TECHNICAL UNIVERSITY OF CRETE

SCHOOL OF ELECTRICAL AND COMPUTER ENGINEERING

DIPLOMA THESIS

Predicting the Occurrence of Flight Delay based on Machine Learning Techniques

AUTHOR

Alexandros N. Chatzipetros

THESIS COMMITTEE

Prof. Michail G. Lagoudakis (Advisor)

Prof. Michalis Zervakis

Dr. Vassilios Diakoloukas

A thesis submitted in partial fulfillment of the requirements for the degree of Diploma in Electrical and Computer Engineering

Chania, October 2024

ΠΟΛΥΤΕΧΝΕΙΟ ΚΡΗΤΗΣ

ΣΧΟΛΗ ΗΛΕΚΤΡΟΛΟΓΩΝ ΜΗΧΑΝΙΚΩΝ

ΚΑΙ ΜΗΧΑΝΙΚΩΝ ΥΠΟΛΟΓΙΣΤΩΝ

ΔΙΠΛΩΜΑΤΙΚΗ ΕΡΓΑΣΙΑ

Πρόβλεψη Συμβάντος Καθυστέρησης Πτήσης βάσει Τεχνικών Μηχανικής Μάθησης

ΣΥΓΓΡΑΦΕΑΣ

Αλέξανδρος Ν. Χατζηπέτρος

ΕΞΕΤΑΣΤΙΚΗ ΕΠΙΤΡΟΠΗ

Καθ. Μιχαήλ Γ. Λαγουδάκης (Επιβλέπων)

Καθ. Μιχάλης Ζερβάκης

Δρ. Βασίλειος Διακολουκάς

Διπλωματική εργασία που υποβλήθηκε σε μερική εκπλήρωση των απαιτήσεων για την απόκτηση του Διπλώματος Ηλεκτρολόγου Μηχανικού και Μηχανικού Υπολογιστών

Χανιά, Οκτώβριος 2024

With a true view, all the data harmonize, but with a false one the facts soon clash.

Aristotle

Abstract

In recent years, there has been a great deal of interest in flight delay prediction, since flight delays have a negative impact on both the economy and the environment, as they increase fuel consumption and therefore carbon emissions. Statistical and operational methods for prediction have been employed in this area, and with the advent of technology, Machine Learning techniques serve also a catalytic role in the study and forecasting of flight delays. Prompted by the aforementioned facts, in this Diploma Thesis we applied Machine Learning methods to forecast a) the average departure delay of a flight, b) the average delay throughout a flight, and c) the average total delay of a flight, using real-world, chronological data. In addition, two strategies were used to estimate the overall delay: either by aggregating the findings of the departure delay and flight delay prediction models, or by applying a single model to predict the total delay directly. Traditional Machine Learning approaches, such as Linear Regression, Polynomial Regression, and Support Vector Regression, as well as Neural Networks, were employed to develop our predictions. To determine the efficacy of the algorithms, the Mean Absolute Error and Root Mean Square Error metrics were employed, along with the R-Squared Determination Coefficient metric, in conjunction with the Cross Validation approach, while the execution time of each implementation was also considered. In addition, the effect of a new heuristic factor, namely the knowledge of previous delays at both the arrival and departure airports, was examined. In conclusion, the application of our methodology to annual flights between two of the busiest international airports in the United States of America (USA) demonstrates that the use of Machine Learning algorithms can contribute to flight delay prediction, while the novel feature of knowing previous delays positively affects the accuracy of the predictions.

Περίληψη

Τα τελευταία χρόνια, έχει παρατηρηθεί έντονο ενδιαφέρον για την πρόβλεψη καθυστερήσεων πτήσεων, καθώς οι καθυστερήσεις πτήσεων είναι ένα πρόβλημα το οποίο έχει αντίκτυπο στην οικονομία μιας κοινωνίας, αλλά και στο περιβάλλον, αφού αυξάνουν την κατανάλωση καυσίμων και κατ' επέκταση τις εκπομπές διοξειδίου του άνθρακα. Σε αυτήν την κατεύθυνση, έχουν υλοποιηθεί προσεγγίσεις πρόβλεψης, οι οποίες βασίζονται σε στατιστικές και επιχειρησιακές μεθόδους, ενώ με την πρόοδο της τεχνολογίας οι τεχνικές Μηχανικής Μάθησης επίσης διαδραματίζουν καταλυτικό ρόλο στην ανάλυση και την πρόβλεψη καθυστέρησης μιας πτήσης. Ορμώμενοι από αυτά τα δεδομένα, στην παρούσα Διπλωματική Εργασία, εφαρμόσαμε αλγορίθμους Μηχανικής Μάθησης που αφορούν στην πρόβλεψη α) της μέσης καθυστέρησης αναχώρησης μιας πτήσης, β) της μέσης καθυστέρησης κατά την διάρκεια μιας πτήσης και γ) της μέσης συνολικής καθυστέρησης μιας πτήσης, χρησιμοποιώντας πραγματικά, χρονολογικά δεδομένα. Επιπλέον, η πρόβλεψη της συνολικής καθυστέρησης υλοποιήθηκε με δύο μεθόδους, είτε συναθροίζοντας τα αποτελέσματα των μοντέλων πρόβλεψης καθυστέρησης αναχώρησης και καθυστέρησης κατά τη διάρκεια μίας πτήσης, είτε εφαρμόζοντας ένα ενιαίο μοντέλο για την απευθείας πρόβλεψη της συνολικής καθυστέρησης. Για την υλοποίηση των αλγορίθμων πρόβλεψης, χρησιμοποιήσαμε παραδοσιακές τεχνικές Μηχανικής Μάθησης, και συγκεκριμένα Γραμμική Παλινδρόμηση, Πολυωνυμική Παλινδρόμηση, Παλινδρόμηση μέσω Διανυσμάτων Υποστήριξης (Support Vector Regression), αλλά και Νευρωνικά Δίκτυα. Με στόχο τη διερεύνηση της αποτελεσματικότητας των αλγορίθμων χρησιμοποιήθηκαν οι μετρικές του Μέσου Απολύτου Σφάλματος (Mean Absolute Error) και η Ρίζα του Μέσου Τετραγωνικού Σφάλματος (Root Mean Square Error), καθώς και η μετρική Συντελεστή Προσδιορισμού R-Squared, σε συνδυασμό με την τεχνική Cross Validation, ενώ λήφθηκε υπ’ όψιν και ο χρόνος εκτέλεσης της εκάστοτε υλοποίησης. Επιπλέον, μελετήθηκε και η συνεισφορά ενός νέου χαρακτηριστικού, αυτού της γνώσης προηγούμενων καθυστερήσεων, τόσο στο αεροδρόμιο άφιξης, όσο και στο αεροδρόμιο αναχώρησης. Εν κατακλείδι, η εφαρμογή της μεθοδολογίας μας σε ετήσιες πτήσεις ανάμεσα σε δύο πολυσύχναστα διεθνή αεροδρόμια των Ηνωμένων Πολιτειών Αμερικής (ΗΠΑ) καταδεικνύει ότι η χρήση αλγορίθμων Μηχανικής Μάθησης μπορεί να συμβάλλει στην πρόβλεψη της χρονικής καθυστέρησης πτήσεων, ενώ το νέο χαρακτηριστικό γνώσης προηγούμενων καθυστερήσεων έχει θετική συνεισφορά στις προβλέψεις.

Acknowledgments

This journey would not have been possible without the help of numerous individuals, and I would like to express my gratitude to each of them. My parents, Georgia and Nikolaos, have always supported me and encouraged me to test and surpass my limits. Therefore, I would like to thank my friends Giorgos, Kostas, Christos, and Konstantinos for their unwavering support and tenacity over the years, which kept me going.

Last, but not least, I would like to thank my thesis advisor, Professor Michail G. Lagoudakis, for letting me implement this thesis and for his support.

Table of Contents

List of Figures

List of Tables

Chapter 1

Introduction

Since the beginning of its existence, humanity has created a variety of tools to utilize effectively for its survival. Therefore, human ingenuity led to the advancement of these tools, ushering in the era of computing devices. These innovations have aided humanity in its struggle for existence and the enhancement of its quality of life. Artificial intelligence, utilizing its subsector of Machine Learning that contains Deep Learning, is another weapon in humanity's arsenal. Today, people can conclude and make predictions in many areas of research, such as travel, industry, aviation, and science in general, by using artificial intelligence to analyze data.

1.1 Motivation

Air transportation is a vital infrastructure that facilitates almost 7 billion passenger boardings each year, including around 800 million in the United States (U.S.) [1], [2]. It is also a complicated system with multiple interdependent components.

Constrained airspace and airport resources, thousands of aircraft, air traffic control systems (ATC), and weather disturbances all contribute to the complexity and make delays unavoidable. For instance, 18% of domestic flights in the United States were delayed in 2015, and 1.5% were canceled [2], nearly 40% of these delays were attributable to the late arrival of inbound planes Figure 1.1 shows a complete picture of arrival delays in the USA in 2015. This proportion of delays attributable to late inbound arrivals indicates the interdependent nature of the delay dynamics.

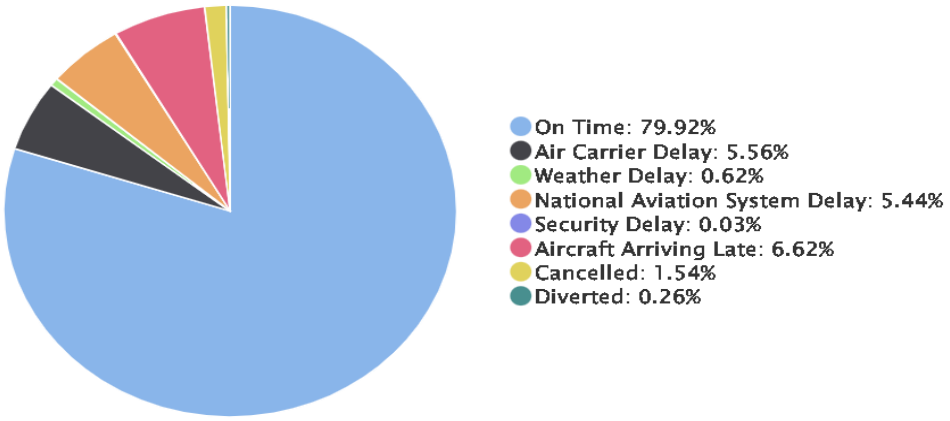

Figure 1.1: On-Time arrival performance U.S. Airports 2015 [2]

According to estimates, delays cost the US economy up to $41 billion annually in 2007 [3], [4] including $31 billion in direct costs and $10 billion in spillovers Figure 1.2 [5]. The inherent intricacies and size of the system make predicting delays a difficult task. The forecast of air traffic delays, even a few hours in advance, has the potential to improve system performance by allowing ATC to take proactive preventive actions and by assisting airlines in better planning recovery operations. The aforementioned facts motivated us to model some parts of the flight delay problem and, using Machine Learning techniques, to generate a numerical prediction of delays between two busy airports in the United States of America, which we cite in this thesis.

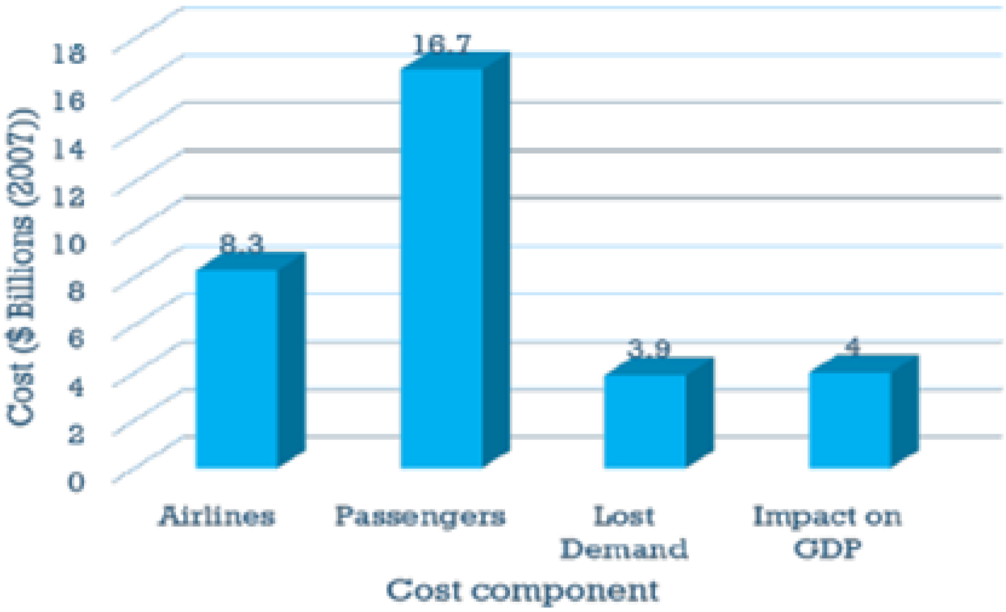

Figure 1.2: Direct cost of air transportation delays (2007 in billions USD) [5]

1.2 Thesis Contribution

This thesis describes an implementation developed to predict flight delays at airports in the United States of America. Consequently, three internal approaches (models) to the flight delay problem between JFK (John F. Kennedy International Airport) and LAX (Los Angeles International Airport) are constructed utilizing data derived from the 2015 dataset of the United States Bureau of Transportation Statistics [6]. In addition, the third delay model is constructed in two directions to determine which of the two has a superior structure and yields more accurate predictions.

Furthermore, the delays of each model are predicted numerically (Regression) using both traditional Machine Learning and Feed-Forward Neural Network approaches. To obtain a more comprehensive implementation of the approximations, the simple average and approximation forecasts derived by Machine Learning techniques is incorporated into the verification process. When performing k-fold Cross-Validation, the predictions of the simple average delay of each element is also be validated, such that the Regression metrics indicate whether or not the Machine Learning techniques contributed to a more precise forecast of the flight delay problem.

To sum up, this thesis contributes the proposal of three distinct predictions for the multivariate flight delay prediction problem between two airports. In addition, it reveals the role of Machine Learning techniques in the aforementioned implementations, and through a comparison of the two directions of the third approach, it determines which implementation structure, splitting the problem into two sub-problems or utilizing a single model, is more accurate for delay prediction.

1.3 Thesis Outline

A concise summary of the thesis's material is provided in this paragraph as follows. In the following Chapter 2, a survey of the pertinent literature that has been applied to related flight delay issues and has provided useful information for this thesis is presented. Continuing in Chapter 3, information regarding the theory of tools for the realization of the thesis is offered, including algorithm theory, Machine Learning and methods, and Neural Networks. Chapter 4 describes the utilized data, their statistical description, research questions, how they were set up for prediction, how to use them, and how to evaluate the approaches. Chapter 5 includes a description of the experiments, and their outcomes, and a discussion, and comparison of these, which leads to the answers to the research questions posed in the previous chapter. At last, Chapter 6 is the conclusion of the thesis, where conclusions and proposals for further research on the flight delay problem are drawn.

Chapter 2

Related Work

Numerous attempts have been made to predict flight delays and their durations using a variety of methodologies, such as [7]. When constructing these prediction models, researchers and analysts have encountered several obstacles. Among these obstacles are recognizing whether data may be pertinent to their work, where to access it, and how to analyze it efficiently.

In a similar manner, numerous scientists have attempted to develop a model that envelops the influential components and analyzes the impact of each aspect and their relationship. In conclusion, flight delay prediction methods can be categorized as follows: Statistical Methods, Operational Methods, Machine Learning Methods, and Network-Based Methods. We will elaborate on each of these category methods in more detail below.

2.1 Research based on Statistical methods

Introductory, utilizing statistical tools and approaches to solve a problem is a cornerstone of science. In one statistically-based research [8] conducted, simulations reveal disparities between operational delays and inherent schedule delays, as shown in Figure 2.3. Their research has examined significant factors before the flight and those that occur on the ground. At a subsequent stage, the delay at the destination is projected based on circumstances that occur close to the arrival time at the destination. Subsequently, studies [4], [5] demonstrated that whenever a delay is accurately predicted, passenger dissatisfaction and fuel use decrease, resulting in a rise in the number of flights. In addition, the agencies' benefits can be increased by minimizing the number of passengers who

incorrectly select their routes or by specifying the probability for certain flights and optimizing delay time forecast.

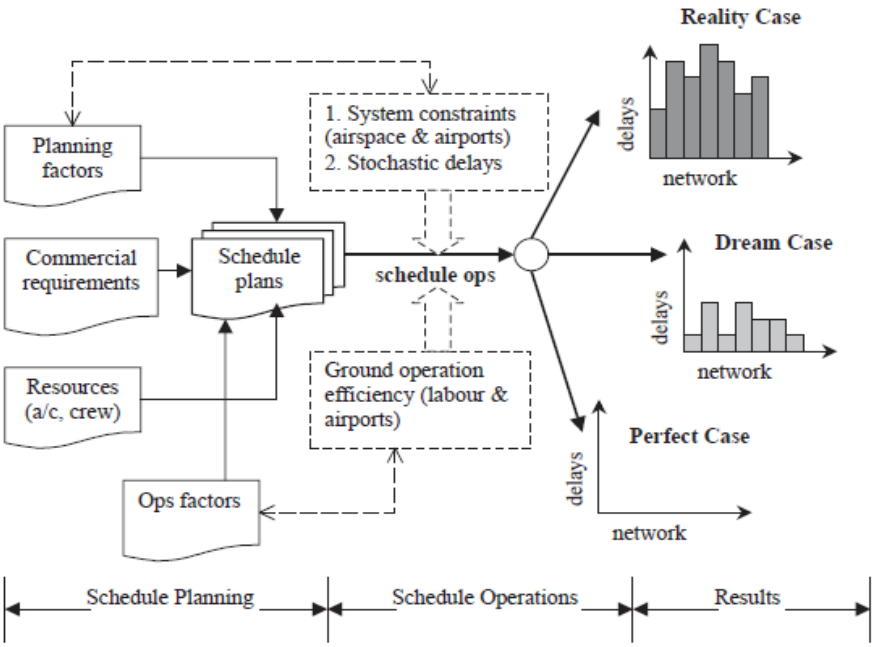

Figure 2.3: Key factors influencing on time performance (OTP) of airline flight schedules [8]

Another significant analysis based on probability has been conducted [9], and according to the author's perspective, the substantial storm has exerted a noteworthy influence on the occurrence of aircraft delays. This study was devoted to predicting aircraft delays based on mathematical calculations and analyzing the duration of flight delays caused by storms on the same day. According to meteorological records, the effects of a storm one hour before and after an event generate a fleeting environment in the region. At a subsequent stage, a Monte-Carlo simulation was performed to estimate the airport runway capacity [9], hence estimating the traffic on each runway. Due to the study's reliance on a single variable, the model's accuracy is insufficient, although it is possible to improve the regional air capacity path structure.

2.2 Research based on Operational Methods

The aviation and optimization groups have conducted extensive research on these operational methods and related subproblems, however, this research [10] has primarily concentrated on a single subproblem at a time in isolation. In [11] an integer optimization model is presented, regarding the arrival/departure runway balancing problem (ADRB) [11] utilizing a runway configuration to simulate runway capacity, as shown in Figure 2.4. So, researchers then solved the airport runway configuration management problem and the

ADRB within a single optimization model and proposed an extension to the case of airports within a metropolitan area with shared airspace.

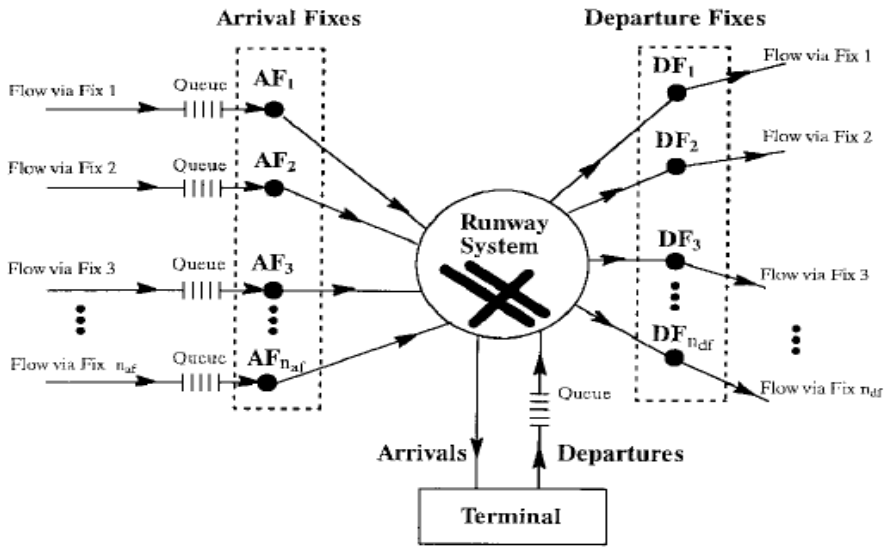

Figure 2.4: The arrival-departure scheme of an airport [11]

Additional studies concentrate on delay and its propagation stability and dependability. For example, congestion issues were studied in [12]. Then, a queue-based model for studying delay propagation among consecutive planes at the Los Angeles International Airport (LAX) was given. At least 15 percent of the flights arriving at LAX on the sample day cause more congestion delay than they save in schedule delay, according to this study. These flights are typically commuter flights that operate during periods of peak demand, which are favored by current pricing rules. In the long-term, numerous airports, including LAX, have contemplated the addition of dedicated commuter runways. This may be justifiable, but only if a full-length runway is impractical and the incremental value of commuter flights warrants a substantial investment.

Furthermore, another operational research [13] analyzed delay propagation in a network of airports using a queuing model. The Approximate Network Delays model described, is a stochastic and dynamic queuing model developed to compute delays at each airport in a network and, more importantly, how these delays spread from airport to airport over a day or other period of interest.

2.3 Research based on Machine Learning Methods

Using Data Mining and Machine Learning approaches, a Classification study [14] examines flight information for American Airlines' domestic flights serving the five busiest airports in

the United States and predicts the possibility of a flight delay. By employing Grid Search [15] on Gradient Boosting Classifier Model [16] on flight data, this research provided a hyper-parameter-tuned method. Also in this research, the Over-Sampling technique Randomized SMOTE [17], [18] is used for Data Balancing, and the ensuing performance gain has been demonstrated. In conclusion, the Validation Accuracy is 85.73%.

The creation of an effective model to predict the flight delay is considered as a task of great importance. In this study [19], a method is proposed for modeling the arriving flights and delay prediction through Machine Learning techniques, such as Multiple Linear Regression [20], Naive-Bayes [21], and the C4.5 approach [22], shown in Figure 2.5.

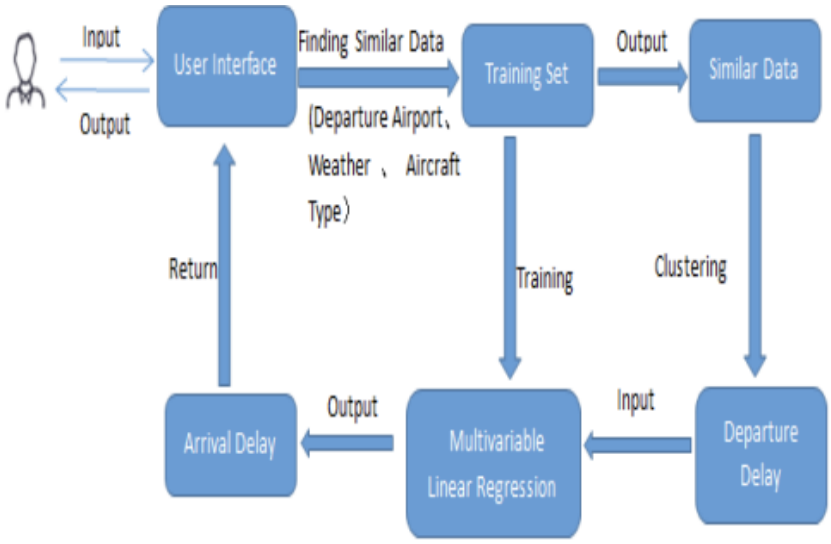

Figure 2.5: The detailed computational framework of flight delay [19]

Experiments with an actual dataset of domestic airports indicate that the suggested model's accuracy is approximately 80%, which is superior to the Naive-Bayes and C4.5 techniques. Test results indicate that this method is convenient for calculation and may accurately anticipate flight delays, and it can provide airport authorities with a basis for decisions.

To continue with, using Machine Learning models, such as Logistic Regression [23], Random Forest Regressor [24], Decision Tree Regression [25], Bayesian Ridge [26], and Gradient Boosting Regression [16], a group of academics [27] has developed five models for predicting flight delays, as shown in Figure 2.6. They gathered information from the Bureau of Transportation and U.S. Statistics on all domestic flights taken in 2015 [6] and projected whether a given flight will arrive on time or be delayed. Mean Squared Error (MSE) [28], Mean Absolute Error (MAE) [28], Score for Explained Variance [29], Median Absolute error [28], and R2 score [30] were used to evaluate the performance of the models. Due to the use of unbalanced data sets, there was a significant amount of computed inaccuracy. Random Forest Regressor was determined to be the best model for predicting arrival and departure delays based on the data.

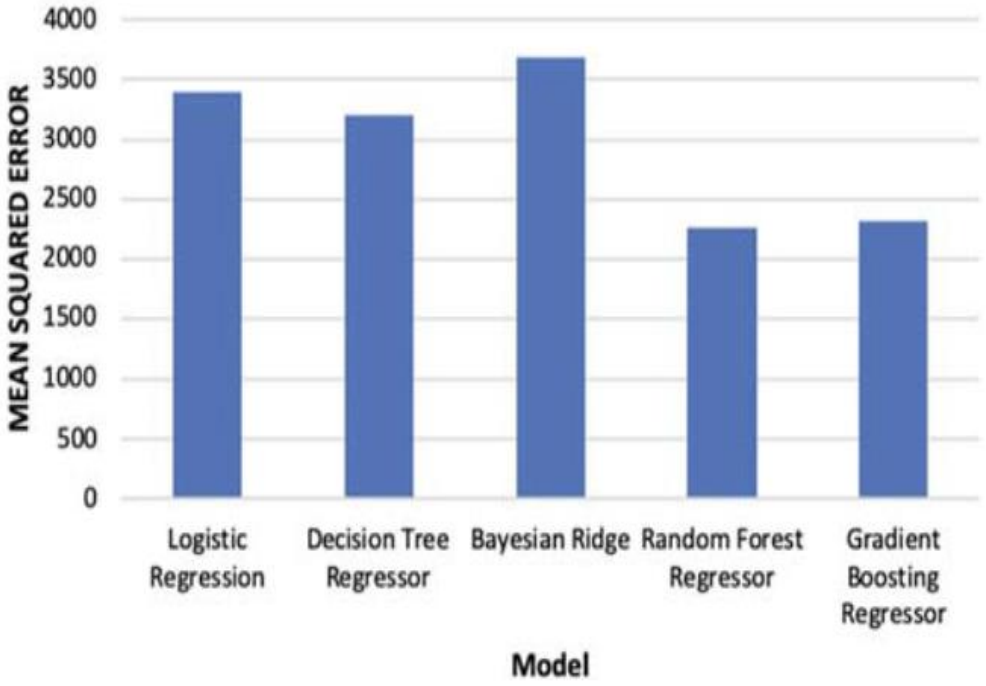

Figure 2.6: Algorithm comparison Mean Squared Error [27]

In addition, research on neural networks will be mentioned at this point. Predictive neural networks, a subfield of deep learning [31], are conceptually a complex network of interconnected nodes that "learn" from the data structure. Neural networks start by studying historical data to determine how to forecast the known output values using the provided predictor variables. In [32] researchers investigate the application of deep learning algorithms to the processing of air traffic data and the performance of deep learning models in tasks involving the prediction of air traffic delay. By combining many models based on the deep learning paradigm, an accurate and robust prediction model has been constructed that enables an in-depth investigation of air traffic delay patterns. In particular, Recurrent Neural Networks (RNN) have proven their accuracy [32] in modeling sequential data. Long Short-Term Memory (LSTM) RNN architectures have been used to model the daily sequences of departure and arrival flight delays at an individual airport, as shown in Figure 2.7. Finally, the proposed prediction model's accuracy was assessed, examined, and compared to earlier prediction methods. It is the most accurate procedure compared to all others [32].

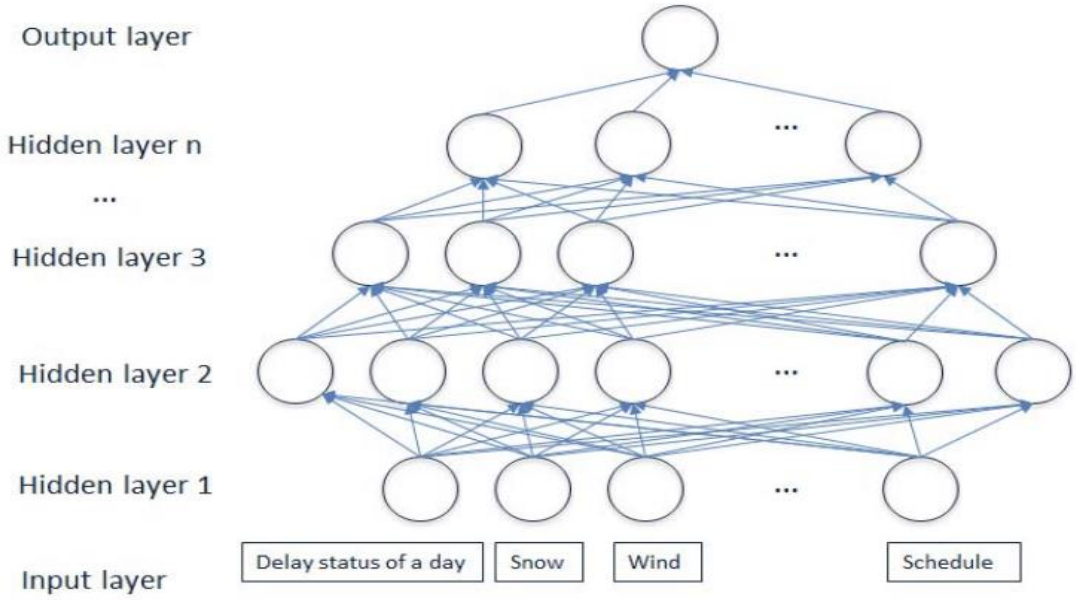

Figure 2.7: Individual flight delay model [32]

2.4 Research based on Network Methods

Flight delays are inherently stochastic phenomena. Congestion at the origin airport, weather, rising demand, and air traffic management (ATM) choices, such as the Ground Delay Programs (GDP), are major contributors to flight delays. In [33] researchers propose a stochastic model that makes use of Bayesian Networks (BNs) to understand the links between the various components of aircraft delay and the causes that cause delays. A case study on departure flight delays from Chicago O'Hare International Airport (ORD) to Hartsfield-Jackson Atlanta International Airport (ATL) reveals how local and system-level environmental and human-caused factors interact to affect delay components and how these components contribute to the overall arrival delay at the destination airport.

The model and results provided in [34] primarily examine the propagation of delays throughout the airport system and the queuing phenomena associated with such propagation. Analytical queuing and network decomposition models were created to examine the complicated phenomena of delay propagation across a wide network of major airports in the USA. The Approximate Network Delays (AND) developed model calculates the delays caused by local congestion at individual airports and captures the "ripple effect" that causes these delays to propagate. The model operates by repeatedly cycling between its two major components: a Queuing Engine (QE) that computes delays at individual airports and a Delay Propagation Algorithm (DPA) that updates flight schedules and demand rates at all airports in the model in response to the local delays computed by the QE. This research demonstrated the significance of incorporating delay propagation into airport network models.

Chapter 3

Background

In this chapter, a summary of the theoretical knowledge underlying the instruments utilized to implement this thesis will be presented. To begin with, due to the complexity of aviation Big Data, data analysts and researchers confront a variety of obstacles, while attempting to analyze, comprehend, and detect trends in aviation Big Data. Particularly challenging are the ingestion, storage, exploration, analysis, and visualization of aviation Big Data. Traditional approaches to delay modeling, in particular, are incapable of analytically analyzing the huge volume of data associated with traffic, the effects of bad weather, or airport-related operations (runway closures, construction, etc.) that cause Ground Delay Programs. After ingesting and processing Big Data in the aviation industry, analysts and researchers must be able to promptly and efficiently interpret data that is crucial to their operations. Machine Learning is one strategy for addressing this issue. Using information regarding traffic volume, adverse weather, etc., new approaches that employ Machine Learning techniques have demonstrated some promise in their capacity to predict traffic delays.

3.1 Machine Learning theory

Machine Learning is a data analysis technique that focuses on the creation of computer algorithms that translate data into useful actions [35]. Machine Learning techniques have been widely implemented across numerous industries. Forecasts of weather behavior and long-term climate changes, decrease in fraudulent credit card transactions, prediction of popular election outcomes, finding of genetic sequences connected to diseases, and image recognition are examples of applications. Machine Learning has also been widely utilized to

complement the subject-matter expertise of subject-matter experts and to improve data production and aggregation procedures. It is important to note, however, that Machine Learning has limitations, despite the enormous benefits it has provided to society. It has a limited ability to extrapolate beyond the constrained boundaries it has learned. To avoid over- or under-fitting, the model must be trained precisely and thoroughly.

In addition to being a non-intuitive process, Machine Learning algorithms frequently function as black boxes. Similarly, Machine Learning techniques and implementations rely on numerous assumptions. The deployment of any Machine Learning technique/algorithm, therefore, requires a thorough comprehension of these assumptions. The three categories of Machine Learning algorithms are Supervised Learning, Reinforcement Learning, and Unsupervised Learning. Understanding these categories is a necessary prerequisite for creating accurate prediction models.

3.1.1 Supervised Learning

Supervised learning is the process of training a model to predict one value using other values in the dataset. This method will be also used in this thesis. To continue with, predictive models are the models utilized in supervised learning. In supervised learning, the algorithm attempts to uncover and model the link between the anticipated value (target) and other values (predictors). Not only may predictive models be used to forecast future events, but also past and current events.

Importantly, Supervised Learning does not include human involvement. Instead, it relates to the fact that target features allow learners to evaluate how effectively they have mastered the intended tasks. The techniques for Supervised Machine Learning can be utilized for two tasks: Regression and Classification [35], as shown in Figure 3.8. In Supervised Learning, data is labeled and algorithms are taught to predict output from input data.

Classification

Classification is frequently used to forecast the class to which a given instance belongs. This entails learning the relationship between predictors and a target to map predictors to the objective. The computer program is tasked with identifying which of k categories a given piece of information belongs to [36]. Typically, this assignment requires the learning method to construct a function $f: \mathbb{R}^n \rightarrow \{1, \ldots, k\}$. When $y = f(x)$, the model allocates an input denoted by the vector x to the category denoted by the discrete code y. Classification challenges include predicting whether an email is spam, whether a person has cancer, whether a football team will win or lose, or whether there will be a flight delay [35].

Regression

Using a group of predictors, a target number can be anticipated through the use of numerical predictions. There are no discontinuities or gaps in the values that the objectives

can assume, hence they are continuous. Given some input, the computer software must anticipate a numerical as output. To complete this objective, the learning algorithm must generate the function $f : \mathbb{R}^n \rightarrow \mathbb{R}$. This task is similar to Classification [36], except for the output format (continuous vs. discrete). Predicting revenue, test results, or the number of objects are examples of numerical prediction problems [35].

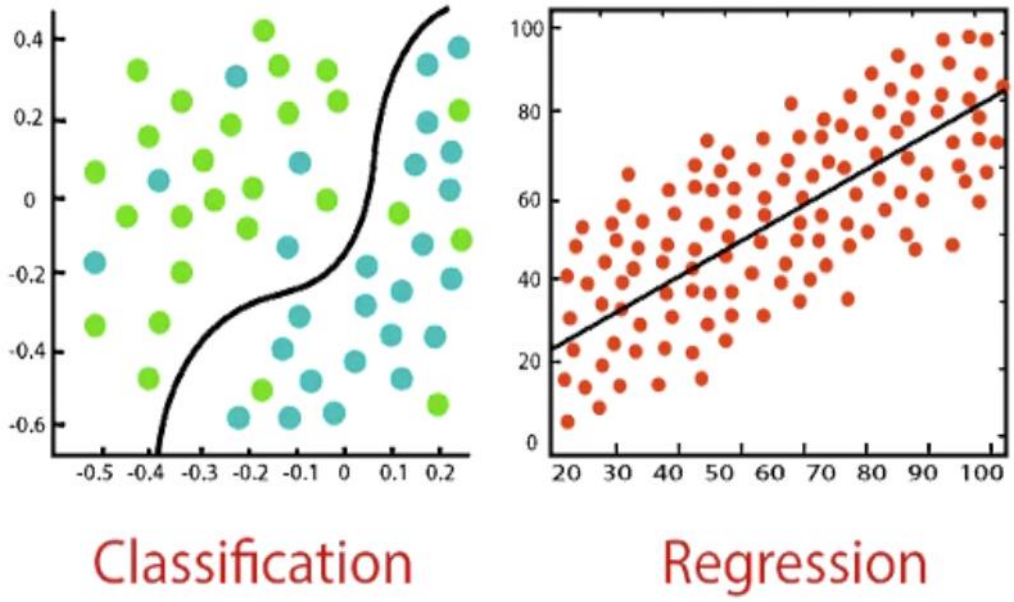

Figure 3.8: Differences between Regression and Classification algorithms [37]

3.1.2 Unsupervised Learning

Unsupervised learning is the process of training a model that benefits from the knowledge gained from summarizing data in novel and intriguing ways. Descriptive models are those utilized in unsupervised learning. In contrast to predictive models that forecast goal values, descriptive models do not prioritize any particular aspect over the others. Algorithms for unsupervised Machine Learning can be employed for pattern discovery and clustering [35]. In unsupervised learning, data is not labeled and algorithms learn to comprehend the data's structure.

Clustering

Generally, clustering is used to split a dataset into distinct categories (clusters), as shown in Figure 3.9. It is often used to discover hidden patterns in data through exploring data. Typically, clusters are described using a measure of similarity defined by metrics like probabilistic distance. Identifying groups of persons with similar behaviors for a marketing

effort is an example of clustering [35]. The k-means algorithm is the most prevalent clustering technique [37].

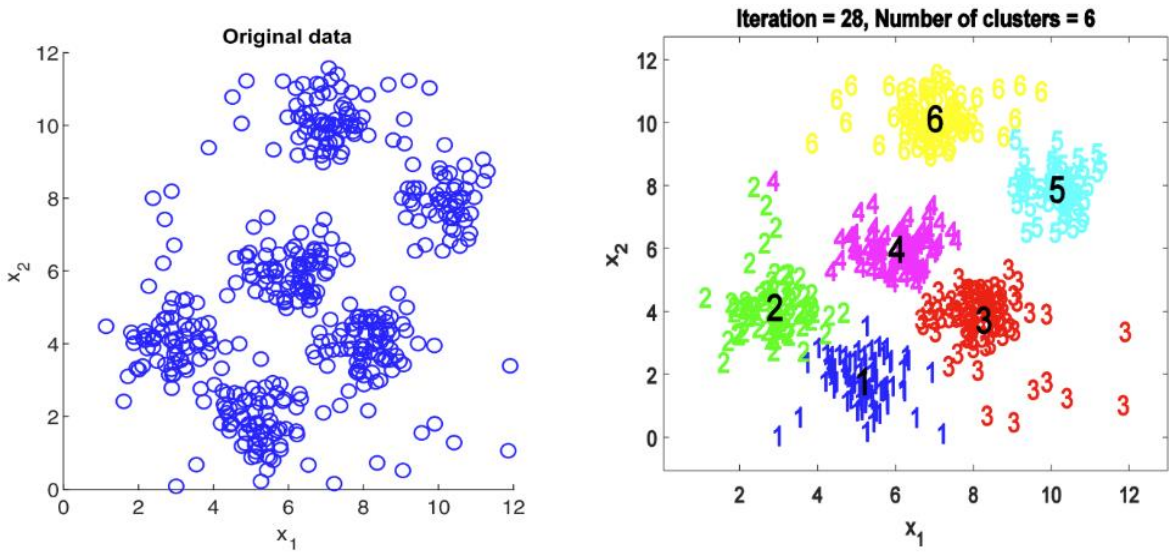

Figure 3.9: A 6-cluster dataset with 50 noisy points [38]

Pattern Discovery

Pattern discovery is used to establish useful data relationships by extracting knowledge from databases without prior knowledge of existing data patterns. Identifying items that are frequently purchased together is an example of a pattern discovery task [35].

3.1.3 Reinforcement Learning

Reinforcement learning [38] is a subfield of Machine Learning that focuses on how software agents behave in an environment to maximize a sense of cumulative rewards. It was mostly used in games (e.g. chess), where its performance rivaled or even surpassed that of humans. Currently, the merging of neural networks has made it possible for these algorithms to accomplish increasingly complex jobs. In addition to the challenges of reinforcement learning, optimal control theory has been studied primarily to construct and characterize optimal solutions and algorithms for exact calculation and less in terms of learning or approximation, especially in the absence of a mathematical environment model. Reinforcement learning can be used by economics and game theory to understand how equilibrium can emerge under constrained reasonableness.

3.2 Neural Networks

Artificial Neural Networks [39], [40] are a form of Machine Learning algorithm influenced by the human brain's structure [41], as shown in Figure 3.10. Each neuron receives inputs, independently weights them, totals them, and then passes this total through a nonlinear function to produce output. The output of a Neural Network is governed by inter-feature correlations. Furthermore, Deep Learning [42] happens when several neuronal layers are connected.

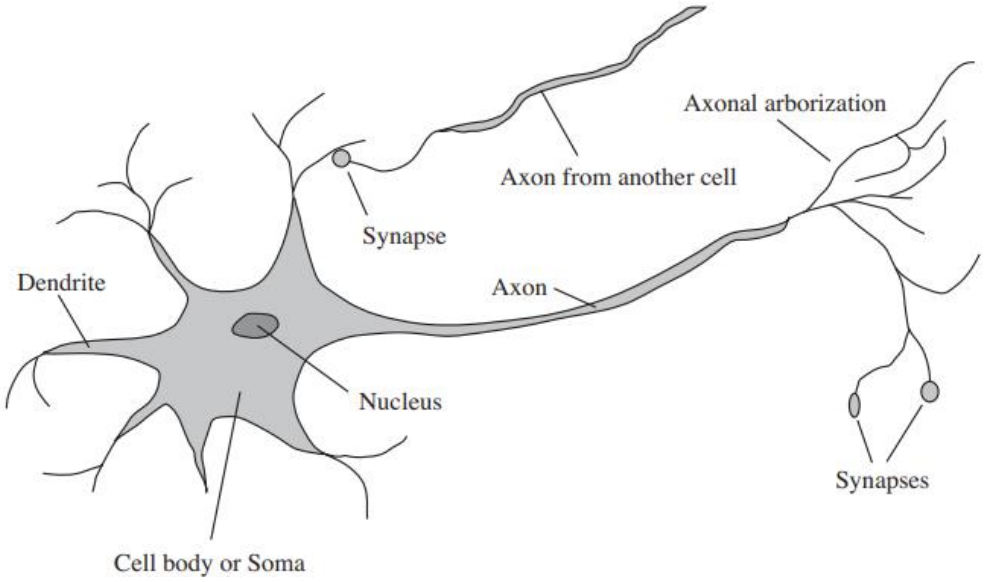

Figure 3.10: The parts of a neuron [42]

A learning mechanism fine-tunes these connection strengths to maximize the neural network's problem-solving effectiveness through trial and error. To solve a problem, the objective is to identify patterns and correlations between the various data elements and to forecast future values or a reward function value. Frequently, the number and configuration of neurons in a neural network, as well as the division of labor among specialized submodules, are tailored to each specific task, basic structure is shown in Figure 3.11. Neural Networks constitute a subset of Machine Learning. A Neural Network is utilized in numerous Machine Learning methods to transform complex data inputs into a format that computers can comprehend.

Currently, they handle self-driving vehicles, offer advertisements, detect individuals, understand messages, and even aid in the creation of new paintings. Neural Networks are currently utilized in a variety of real-world applications, including speech and picture recognition, spam email filtering, and medical diagnosis, to mention a few. Moreover, as demonstrated in this thesis, they can be used for Regression prediction, such as numerical

values for stock market prediction, social media statistics, aerospace, and even aircraft delay prediction.

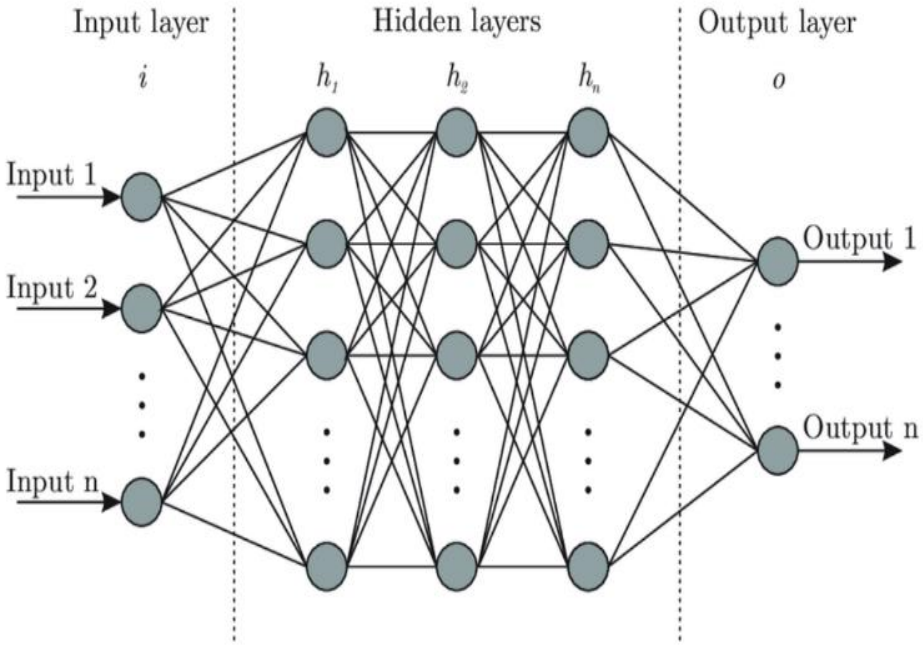

Figure 3.11: Basic Neural Network structure [44]

Input neurons

The number of features used by a neural network to produce predictions. Each dimension character in the input vector requires one input neuron. This is the number of relevant features in the dataset for tabular data. These features must be carefully selected, and any that may have patterns that do not generalize beyond the training set must be eliminated (overfitting) [43].

Output neurons

The number of forecasts. For Regression tasks, a single value is acceptable (e.g. stock price). In Multivariate Regression, however, there is one neuron per predicted value (e.g., for vehicles, there may be four neurons, one for acceleration, maximum speed, width, and length). For binary Classification (spam, non-spam), one output neuron is utilized per positive class, with the output representing the probability of the positive class. One output neuron per class is required for multi-class Classification (such as in object detection, where an instance may be categorized as a flower, a car, or a cat).

3.2.1 Perceptron

Perceptron is the simplest neural network consisting of n number of inputs, one neuron, and one output, where n is the number of features in our dataset. Perceptron was invented by Frank Rosenblatt in 1958 [44]. A perceptron model is also regarded as one of the most effective and specialized types of Artificial Neural networks. As a supervised learning algorithm for binary classifiers, it may also be viewed as a single-layer neural network with four primary parameters: input values, weights and biases, net sum, and activation. Forward Propagation and Back-Propagation are the components of a Neural Network. Forward Propagation is the process of transmitting information through the Neural Network, and Backward Propagation is the process of propagating the error (loss) back to the neural network and then updating the weights of each neuron by altering the weight and bias parameters [45], as shown in Figure 3.12. Following are the five steps of the Neural Network training procedure:

1. Initialize the weight and bias parameters
2. Forward Propagation of some input
3. Compute the Loss [46] of current output with respect to the known target value
4. Backpropagation of error (loss)
5. Update the weight and bias parameters

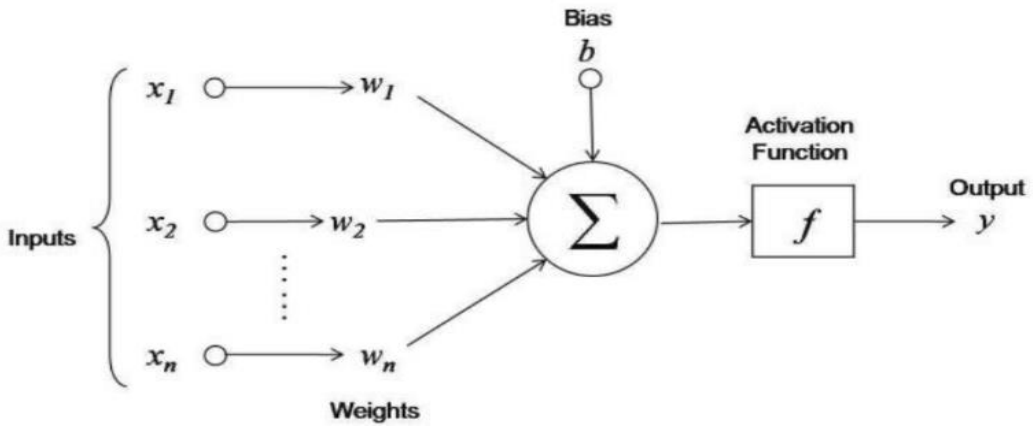

Figure 3.12: Neural network function diagram

Single Layer Perceptron model

As noted previously, Single Layer Perceptron is one of the most basic types of ANN (Artificial Neural Networks) and comprises a feed-forward network with an internal threshold transfer. The basic objective of the single-layer perceptron model is to evaluate binary outcomes that are linearly distinct, for example, image Classification. A single-layer perceptron, see Figure 3.13, is only capable of learning linearly separable patterns. Consequently, it is often referred to as Linear Binary Classifier.

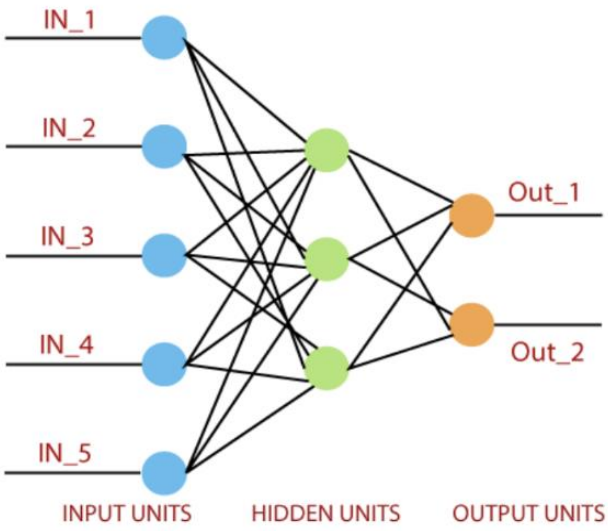

Figure 3.13: Single layer Perceptron model [48]

Multi-Layered Perceptron model

A Multilayer Perceptron, consists of input and output layers, as well as one or more hidden layers with stacked neurons, as shown in Figure 3.14. Moreover, whereas neurons in a Perceptron must have an activation function that imposes a threshold, such as ReLU or Sigmoid, neurons in a Multilayer Perceptron can utilize any activation function. Many Machine Learning approaches to Classification and Regression, utilize multilayer perceptron. Particularly for Classification problems, it has been demonstrated that these methods yield extremely precise outcomes [47]. In Regression problems, a single output neuron with no activation function can be considered.

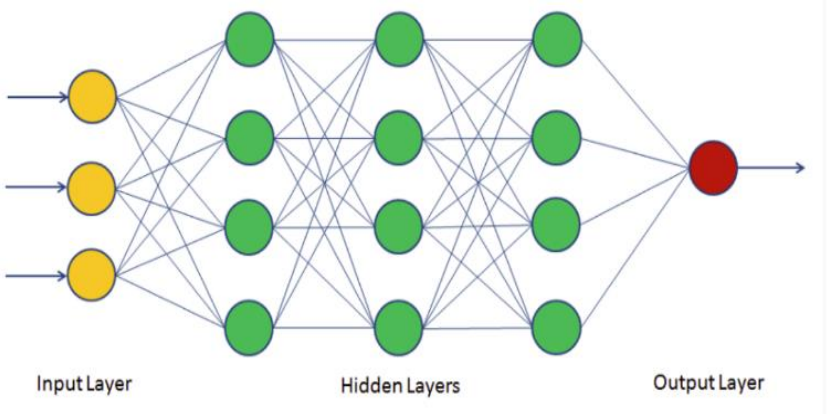

Figure 3.14: Multilayer Perceptron model

Activation Functions of Perceptron

On the x, y plane, the arrangement of data points may not be a straight line, as shown in Figure 3.15. If so, a basic straight-line linear Regression in the form of $y = mx + b$ would not be adequate for predicting y values. Using activation functions we overcome this problem, a modeling instrument for data points that allows the Regression line to be curved. In a nutshell, the activation function transforms Regression into a nonlinear prediction. Several of the most frequent activation functions are listed below:

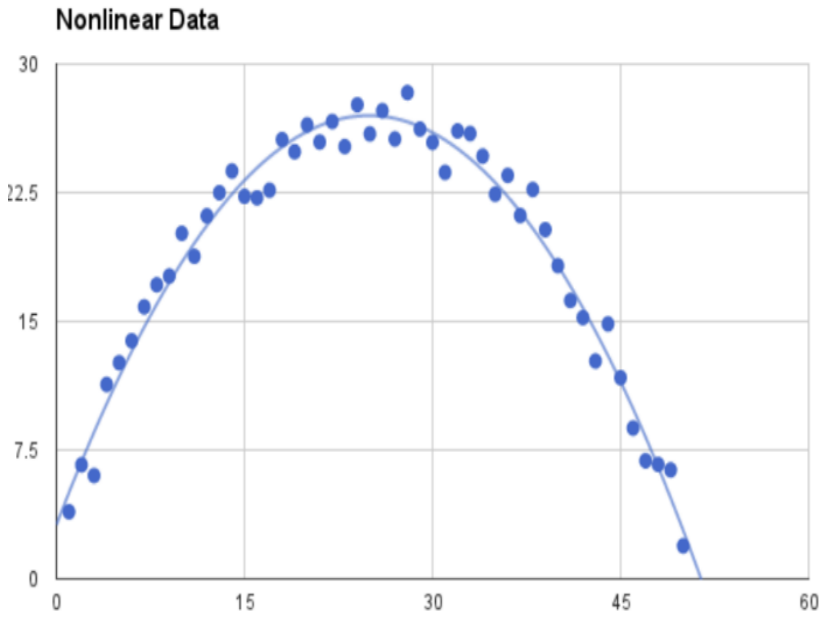

Figure 3.15: Non-linear function [50]

Rectified Linear Unit Activation Function (ReLU)

A node that implements the ReLU activation function is known as ReLU or rectified linear activation unit. Rectified networks are a common term for networks that employ the rectifier function for the hidden layers. The rectified linear activation function is a straightforward calculation that returns the input value immediately, or 0 if the input is less than zero, as shown in Figure 3.16. And defined as follows:

$$ReLU(x) = \max(0, x)$$

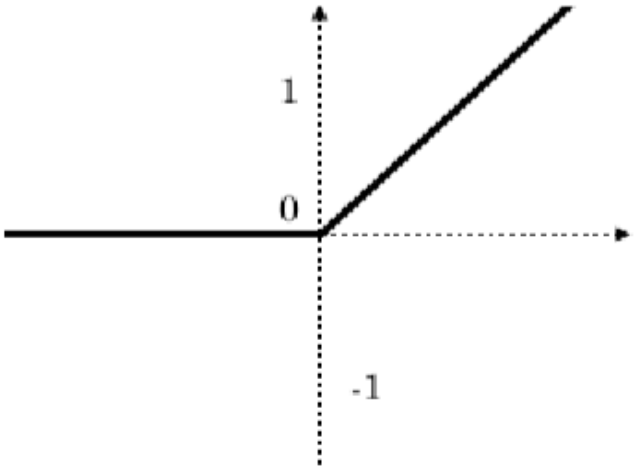

Figure 3.16: ReLU activation function [51]

Leaky Rectified Linear Unit Activation Function (LReLU)

Leaky Rectified Linear Unit [48], or Leaky ReLU, is an activation function based on the ReLU. The Leaky ReLU Activation Function (LReLU) differs from the ReLU Activation Function in that, rather than sending negative values to zero, a very small slope parameter is used to integrate negative value information, as shown in Figure 3.17. Several benefits of this activation function include: Sparse activation, for instance, in a randomly initialized network, only about fifty percent of hidden units are activated (have a non-zero output), and better gradient propagation, such as fewer vanishing gradient problems [49], compared to sigmoidal activation functions that saturate in both directions. In a conclusion it is, defined as:

$$f(y) = \begin{cases} y, & if\ y > 0 \\ ay, otherwise, where\ a\ small\ positive\ number \end{cases}$$

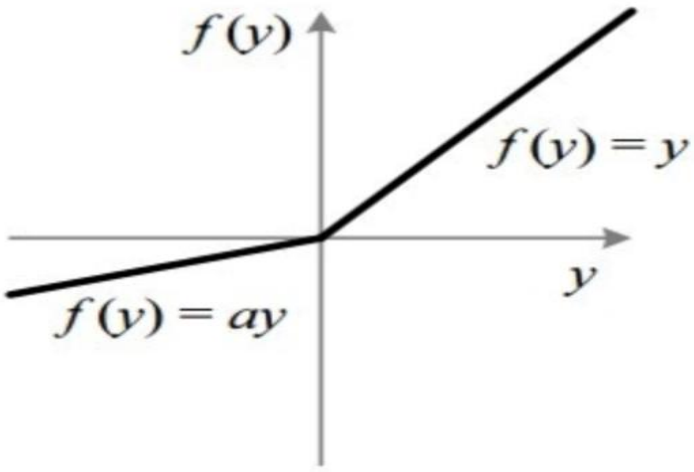

Figure 3.17: Leaky ReLU activation function [54]

Sigmoid or Logistic Activation Function

In neural networks, the sigmoid function is utilized as an activation function. An activation function is applied to a weighted sum of inputs, and the outcome serves as an input to the subsequent layer. When the activation function of a neuron is a sigmoid function, the output of this unit is guaranteed to be between 0 and 1 at all times. Additionally, as the sigmoid is a nonlinear function, the output of this unit would be a nonlinear function of the weighted sum of inputs, as shown in Figure 3.18. Using a non-linear function yields non-linear boundaries; hence, neural networks can use the sigmoid function to learn complex decision functions. The sigmoid function is defined as:

$$sigmoid(z) = \frac{1}{1 + e^{-z}}$$

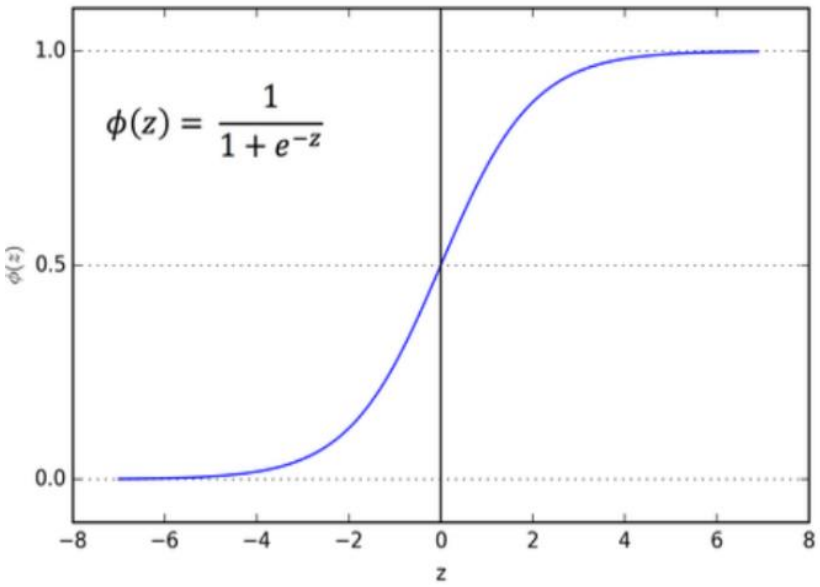

Figure 3.18: Sigmoid function [50]

Summary of Perceptron's characteristics

The following are the characteristics of a Perceptron Model: it is an algorithm for Machine Learning that employs supervised learning of binary classifiers, the weight coefficients are automatically learned, also initial weights are multiplied with input features and finally, a decision is made as to whether or not a neuron is fired. The activation function employs a step rule to determine if the function is greater than zero, the linear decision border is then constructed, permitting the differentiation between the two linearly distinct classes, +1 and -1. If the sum of all input values exceeds the threshold value, an output signal must be generated, otherwise, no signal will be generated.

Some Perceptron restrictions include: due to the hard-edge transfer function, the output of a perceptron can only be a binary number (0 or 1), and it can only be used to classify the linearly separable sets of input vectors. If the input vectors are nonlinear, it is

difficult to appropriately classify them. In the multi-layered perceptron paradigm, computations are difficult and time-consuming. It is difficult to estimate how much each independent variable is influenced by the dependent variable. In conclusion, the functionality of the model depends on the quality of training.

Mathematics underlying Perceptron

Forward propagation steps

Forward propagation consists of multiplying the input value x_i by the weights w_i for each input, and then adding the products. Weights reflect the strength of the connection between neurons and determine how much the supplied input will affect the output of the neuron. If the absolute weight w_1 is greater than the weight w_2, the input x_1 will have a greater impact on the output than w_2 [36].

$$\Sigma = (x_1 w_1) + (x_2 w_2) + \ldots + (x_n w_n)$$

The row vectors of the inputs and weights are denoted by $x = [x_1, x_2, \ldots, x_n]$ and $w = [w_1, w_2, \ldots, w_n]$ respectively, and their dot product is denoted by:

$$x \cdot w = (x_1 w_1) + (x_2 w_2) + \ldots + (x_n w_n)$$

Consequently, the output equals the dot product of the vectors x and w:

$$\Sigma = x \cdot w$$

Add bias [50] b to the product of the multiplied values and denote this as z. Bias, also known as the offset, is typically required to shift the entire activation function to the left or right in order to get the desired output values:

$$z = x \cdot w + b$$

Transfer z to a nonlinear activation function [51]. Without activation functions, the output of neurons would be linear and the neural network would be a linear function. Moreover,

they have a substantial effect on the neural network's learning rate. The binary step function is the activation function of the perceptron. However, activation function can be sigmoid [52], also known as the logistic function:

$$\hat{y} = \sigma(z) = \frac{1}{1 + e^{-z}}$$

where σ represents the sigmoid activation function and stands for the expected value $\hat{y}$ following forward propagation.

Backpropagation steps

Backpropagation refers to the algorithm for calculating the gradient of the loss function [46] concerning the weights. However, the phrase is frequently used to refer to the full algorithm for learning. The backpropagation performed by a perceptron is described in the steps below.

Utilizing a loss function, the distance between an estimate and the desired solution can be calculated. Typically, the Mean Squared Error is selected as the loss function for Regression problems, while cross-entropy is selected for Classification problems. The loss function for a Regression problem is the Mean Squared Error, which squares the difference between the actual y_i and predicted value $\hat{y}_i$.

$$MSE_i = (y_i - \hat{y}_i)^2$$

The loss function is computed for the complete training dataset, and the cost function C [53] is the average MSE.

$$C = \frac{1}{n}\sum_i^n MSE_i = \frac{1}{n}\sum_{i=1}^n (y_i - \hat{y}_i)^2$$

To determine the better weights and bias for the Perceptron, it is necessary to understand how the cost function varies concerning weights and bias. This is accomplished with the use of gradients (rate of change) – how one quantity changes concerning another. In this case we determine the gradient of the cost function concerning the weights and bias.

The gradient of the cost function C concerning the weight w_i is computed by partial derivation. Since the cost function is not directly related to the weight w_i, the chain rule will be used.

$$\frac{\partial C}{\partial w_i} = \frac{\partial C}{\partial \hat{y}} \frac{\partial \hat{y}}{\partial z} \frac{\partial z}{\partial w_i}$$

Beginning with the gradient of the cost function (C) concerning the predicted value ($\hat{y}$)

$$\frac{\partial C}{\partial \hat{y}} = \frac{\partial}{\partial \hat{y}} \frac{1}{n} \sum_{i=1}^{n} (y_i - \hat{y}_i)^2 = 2 \frac{1}{n} \sum_{i=1}^{n} (y_i - \hat{y}_i)$$

Let $y = [y_1, y_2, \ldots, y_n]$ and $\hat{y} = [\hat{y}_1, \hat{y}_2, \ldots, \hat{y}_n]$ be the actual and projected value row vectors. Therefore, the preceding equation may be reduced to:

$$\frac{\vartheta C}{\vartheta \hat{y}} = \frac{2}{n} \sum (y - \hat{y})$$

Furthermore, determine the gradient of the anticipated value relative to z:

$$\begin{aligned}
\frac{\vartheta \hat{y}}{\vartheta z} &= \frac{\partial}{\partial z} \sigma(z) \\
&= \frac{\partial}{\partial z} \left(\frac{1}{1 + e^{-z}} \right) \\
&= \frac{e^{-z}}{(1 + e^{-z})^2} \\
&= \frac{1}{(1 + e^{-z})} \frac{e^{-z}}{(1 + e^{-z})} \\
&= \frac{1}{(1 + e^{-z})} \left(1 - \frac{1}{(1 + e^{-z})} \right) \\
&= \sigma(z) \big(1 - \sigma(z) \big)
\end{aligned}$$

The gradient of z relative to w_i is:

$$\frac{\partial z}{\partial w_i} = \frac{\partial}{\partial w_i} (z)$$

$$= \frac{\partial}{\partial w_i} \sum_{i=1}^{n} (x_i . w_i + b)$$
$$= x_i$$

Therefore, in summary, we have:

$$\frac{\partial C}{\partial w_i} = \frac{2}{n} \sum (y - \hat{y}) \times \sigma(z) \times \big(1 - \sigma(z)\big) \times x_i$$

Similarly, the gradient of C with respect to b is computed as follows:

$$\frac{\partial C}{\partial b} = \frac{2}{n} \sum (y - \hat{y}) \sigma(z) \big(1 - \sigma(z)\big)$$

Optimization – Gradient Descent

Optimization is the selection of a better than the current element from a set of available options, which in this case is the selection of better weights and bias for the perceptron. In this instance, gradient descent [53] is used as the optimization procedure, which modifies the weights and bias in proportion to the negative gradient of the Cost function concerning the respective weight or bias. The learning rate (α) [54] is a hyperparameter that lies in (0,1) and the amount by which the weights and bias are modified, the effect of the learning rate is shown in Figure 3.20. The weights and biases are updated as follows, and gradient descent and backpropagation are repeated until convergence. This iterative update gradually improves the weights and the bias and may lead even to optimal choices.

$$w_i = w_i - \left(\alpha \frac{\partial C}{\partial w_i}\right)$$
$$b = b - \left(\alpha \frac{\partial C}{\partial b}\right)$$

More specifically, Gradient Descent is an algorithm for optimizing the Cost Function or the model's error. It is used to determine a local (or global) minimum of model error, as its direction leads to the smallest possible mistake. The model's inaccuracy can vary at different times, and the quickest solution to reduce it must be identified to prevent the wastage of data. This is continued until the error values become progressively small. Cost function optimization concludes when Gradient Descent identifies the local minimum of errors. The

significance of the learning rate (a) in locating quickly the local minimum of a Loss Function is illustrated in Figures 3.18, 3.19.

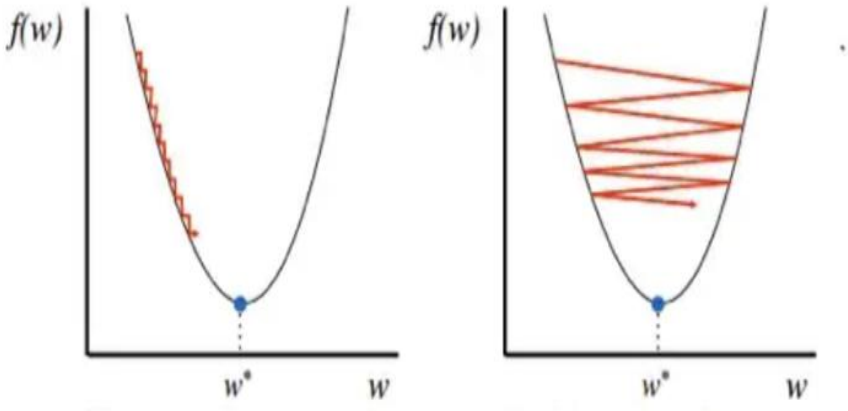

Figure 3.19: Low and high learning rate [60]

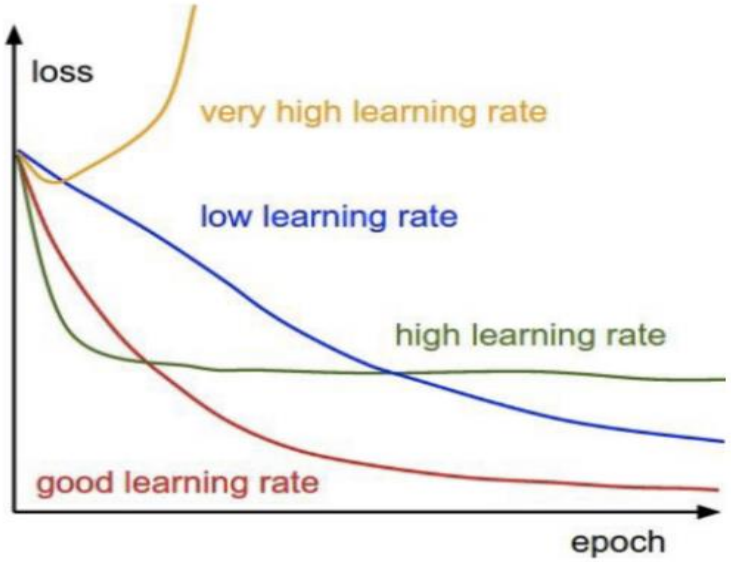

Figure 3.20: Effect of learning rate on the Loss Function [60]

3.3 Machine Learning algorithms

In this section, the Machine Learning algorithms utilized in this thesis will be described. Specifically, we will discuss the mathematical and theoretical foundations of the Regression algorithms employed with the supervised learning technique. These linear algorithms employ flight data from the dataset to estimate flight delays, which is the topic addressed in this study.

3.3.1 Linear Regression

The predictive analytic technique of Linear Regression employs a statistical linear Machine Learning algorithm. The objective [39] is to predict the value of y for a new value of x, given a training data set consisting of N observations x_n, where $n = 1, \dots, N$ and associated target values y_n, as shown in Figure 3.21. This can be accomplished simply by immediately generating a suitable function $y(x)$ whose values for new inputs x represent predictions for the corresponding values of y. In a broader sense, from a probabilistic standpoint, we wish to model the predictive distribution $p(y|x)$, because it reflects our uncertainty regarding the value of y for each value of x. From this conditional distribution, predictions of y can be made for any new value of x to minimize the anticipated value of a suitably selected loss function.

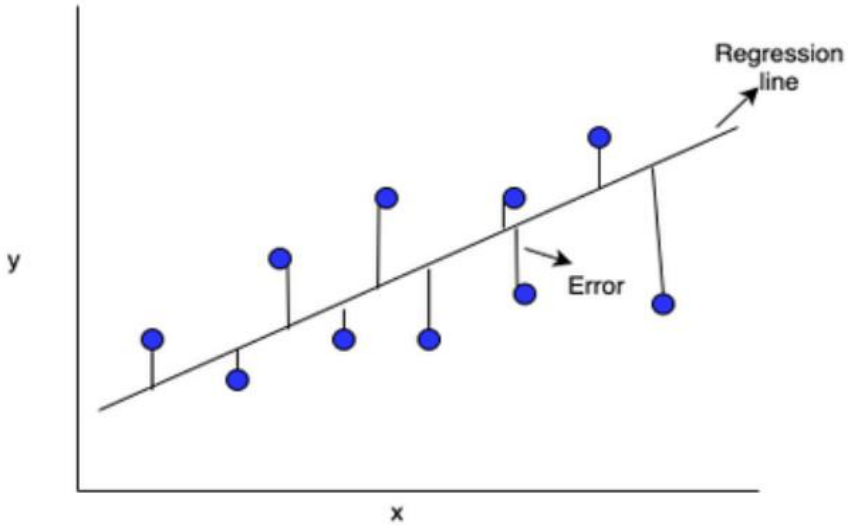

Figure 3.21: Regression prediction line and distance error [61]

Simple and Multiple Linear Regression

The simplest linear model for Regression is the one in which the output is compiled linearly from the input:

$$y = w_1 x + w_o$$

Where y represents the output value. It is also known as the dependent variable in statistical modeling and the target variable in Machine Learning. It is the continuous value that we attempt to forecast. The input variable is x and is referred to as the feature in Machine Learning, whereas in statistics it is known as the independent variable. It depicts

the data received at any particular time. To continue, w_o represents the bias or y-axis intercept. Furthermore, w_1 is the coefficient of Regression or scale factor. In traditional statistics, it corresponds to the slope of the best-fit straight line that is produced after fitting the linear Regression model.

Multiple linear Regression is a type of Regression in which the dependent variable exhibits a linear relationship with two or more independent variables, this is the primary distinction between multiple linear Regression and the simple linear model, which has a single dependent variable. In non-linear models, the dependent and independent variables do not follow a straight line. Graphically, both linear and nonlinear Regression follows a specific response using two or more variables. However, non-linear Regression is typically challenging to implement, because its assumptions are derived from trial and error. However, the data are formed as:

$$y = w_1 x_1 + w_2 x_2 + \cdots + w_n x_n + w_o$$

There are $n + 1$ Regression coefficients and the residual w_o represents other influences on y besides $w_1 x_1 + w_2 x_2 + \cdots + w_n x_n$. To ease the above computation, the equation could be in matrix notation as (where w weights, ε error residuals):

$$Y = wX + \varepsilon$$

The coefficient matrix w is then found by solving the equation below:

$$(X^T X)\widehat{w} = X^T Y, where\ \widehat{w} = (X^T X)^{-1} X^T Y\ Y$$

The simple linear Regression model can be visually represented as the best-fit line between the data points, but the multiple linear Regression model can be represented as a plane (in two dimensions) or a hyperplane (in higher dimensions), as shown in Figure 3.22.

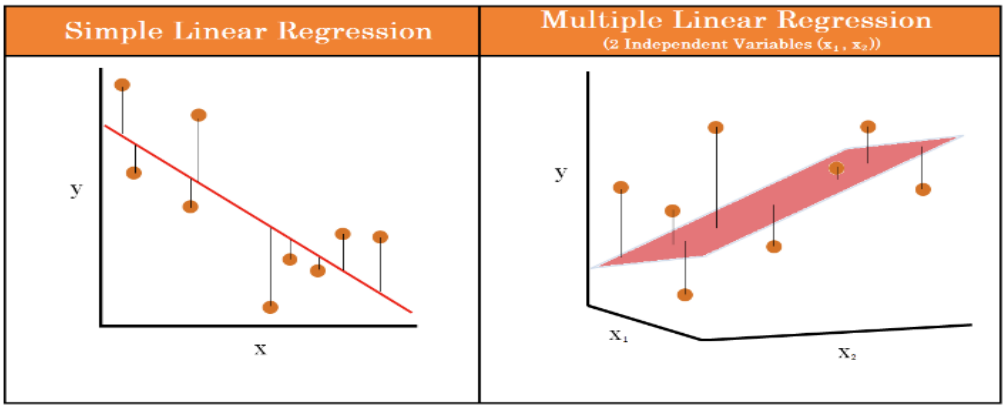

Figure 3.22: Simple Linear – Multiple Linear Regression graphs [62]

The objective of the Regression analysis (modeling) is to determine the values of the equation's unknown parameters. This goal can be achieved by using the cost function, which measures the efficacy of Machine Learning methods. The Cost Function will evaluate the model outputs against the actual outcomes. Therefore, the linear Regression model will iterate through several weight combinations to identify the combination that yields the function with the lowest cost. In addition, the degree of freedom of the first-order polynomial model is $\mathrm{d}+1$, where d is the dimension of the feature vectors.

Polynomial Regression

Simple Linear Regression demonstrates that when a linear model is applied to a linear dataset, it produces a favorable outcome nevertheless if the same model is applied without modification to a non-linear dataset, it will produce a not-so-good output. In instances where data points are ordered non-linearly, a polynomial model is preferred. In polynomial Regression, the relationship between the independent variable x and the dependent variable y is modeled as a polynomial of x's n-th degree. Abbreviated $p(y|x)$, Polynomial Regression [39] depicts the fitting of a nonlinear relationship between the value of x and the conditional mean of y. It corresponds typically to the least-squares approach. The Gauss Markov Theorem [55] states that the least square method reduces coefficient variance. In this sort of Linear Regression, the dependent and independent variables have a curvilinear connection, and a polynomial equation is fitted to the data. Multiple Linear Regression includes Machine Learning as a subset. By adding more polynomial factors, the Multiple Linear Regression equation is transformed into a Polynomial Regression equation, as shown in Figure 3.23.

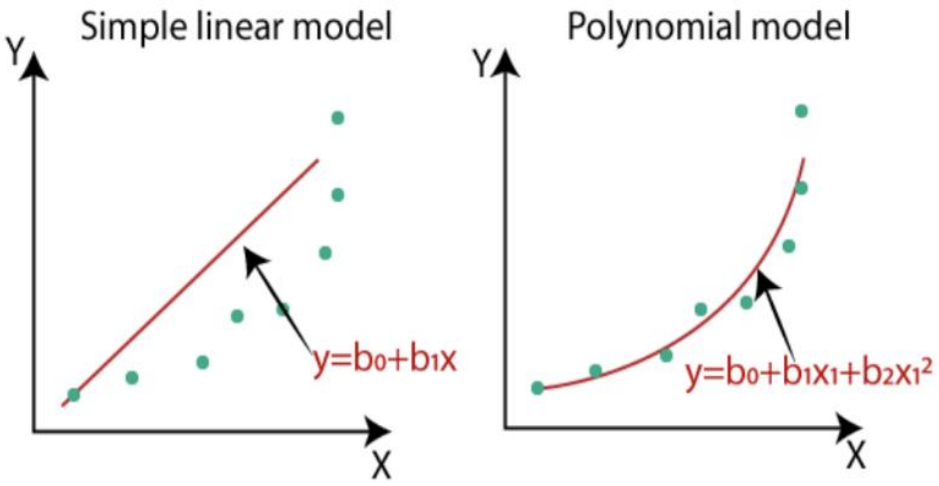

Figure 3.23: Simple Linear model – Polynomial model graph [64]

The equation of Polynomial Regression is:

$$y = w_o + w_1 x_1 + w_2 {x_1}^2 + \cdots + w_n {x_1}^n$$

Polynomial Regression can be easily generalized to inputs of many dimensions, by forming polynomials terms and cross-terms of the input dimensions.

In conclusion, Polynomial Linear Regression is a useful tool in prediction. Since a linear relationship between independent and dependent variables is not always the case, this method can be used when simple Linear Regression fails to adequately fit the data. Furthermore, some advantages are, testing the presence of curvature and its inflections and, nonlinear interactions between variables can also be modeled. However, as the polynomial degree increases, these models are more susceptible to overfitting, and even a single outlier in the data can significantly skew the result.

Cost Function – Gradient Descent

As previously noted in Neural Networks, the Cost Function is a formal representation of the algorithm's intented goal to minimize errors. In the case of linear Regression, the cost function is the minimum of the model's Root Mean Squared Error [56] [39], which is calculated by subtracting the projected values from the actual values. The algorithm handles the minimization problem by attempting to minimize the cost function to determine the best-fitting line with the smallest residual errors. The Regression line described by the

equation $h_\theta(x) = \theta_1 x + \theta_0$ has only two parameters, weight (θ_1) and bias (θ_0), as shown in Figure 3.24:

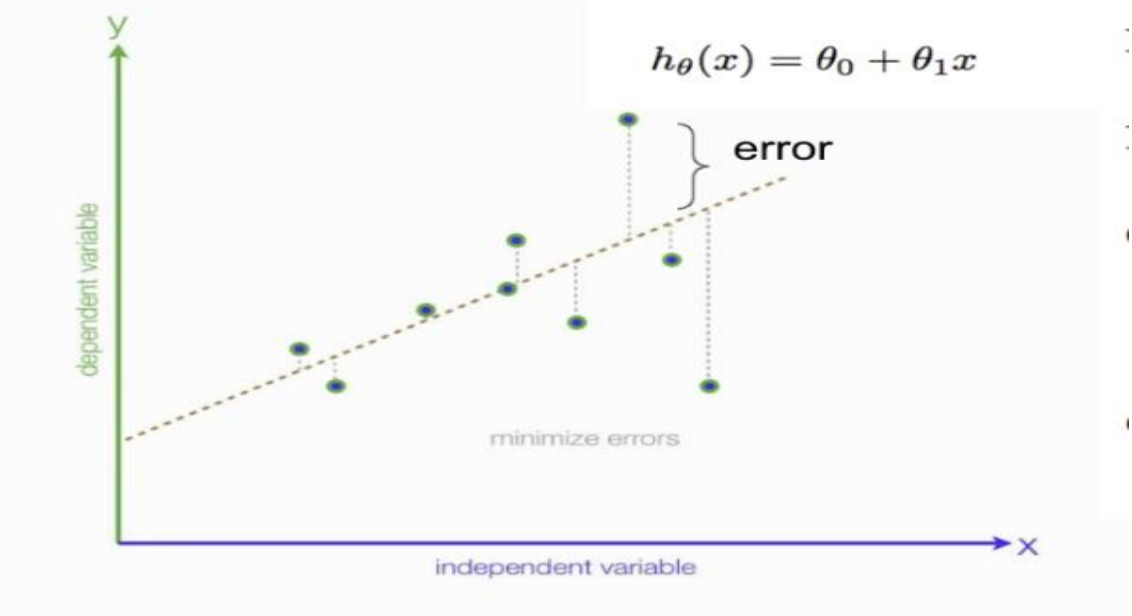

Figure 3.24: Regression line [66]

The Cost Function is defined as follows:

$$J(\theta_0, \theta_1) = \frac{1}{m}\sum_{i=1}^{m}(h_\theta(x^{(i)}) - y^{(i)})^2$$

Cost Function's objective:

$$minimize J(\theta_0, \theta_1)$$

To obtain the lowest error value, the weight (θ_1) and bias (θ_0) must be adjusted to achieve the minimum feasible error. This is because a smaller error between the actual and anticipated numbers indicates that the algorithm has learned effectively. Gradient Descent is an efficient optimization procedure that seeks to identify a function's local or global minimum. Gradient Descent uses calculus to iteratively determine the optimal parameter values that correspond to the minimum value of a particular cost function. Mathematically, the 'derivative' technique is crucial for minimizing the Cost Function, as it aids in obtaining

the minimal point (local or global). The calculus notion of derivative relates to the slope of a function at a specific position. The slope must be calculated so that the direction (sign) indicates the change in coefficient values for the next iteration to have a reduced cost. Specifically, considering the Gradient Descent process, which begins with an initial θ and performs the update repeatedly and simultaneously for all values of $j = [0, \dots, n]$:

$$\theta_j := \theta_j - \alpha \frac{\partial}{\partial \theta_j} J(\theta)$$

Gradient descent enables the learning process to make corrective updates to the learned estimates that bring the model toward an ideal combination of parameters (θ) as a result. For each iteration of the gradient descent process, the cost of a generic technique from Machine Learning is computed across the complete training set. One iteration of the Gradient Descent technique is referred to as one batch, which signifies the total number of samples from a dataset that are utilized to calculate the gradient for each step. To implement this procedure, it is necessary to calculate the partial derivative term:

$$J(\theta) = \frac{1}{m} \sum_{i=1}^{m} (h_\theta(x^{(i)}) - y^{(i)})^2 \quad (3.28)$$

$$\begin{aligned} \frac{\partial}{\partial \theta_0} J(\theta_0, \theta_1) &= \frac{\partial}{\partial \theta_0} \left(\frac{1}{m} \sum_{i=1}^{m} (h_\theta(x^{(i)}) - y^{(i)})^2 \right) \\ &= \frac{1}{m} \sum_{i=1}^{m} \frac{\partial}{\partial \theta_0} (h_\theta(x^{(i)}) - y^{(i)})^2 \\ &= \frac{1}{m} \sum_{i=1}^{m} 2(h_\theta(x^{(i)}) - y^{(i)}) \frac{\partial}{\partial \theta_0} (h_\theta(x^{(i)}) - y^{(i)}) \\ &= \frac{2}{m} \sum_{i=1}^{m} (h_\theta(x^{(i)}) - y^{(i)}) \end{aligned}$$

The size of these steps is known as the learning rate (α), which offers more control over the size of the steps. A greater distance to the local minima can be covered with each step if the learning rate is high, but there is a risk of overshooting the lowest point if the slope of the hill is continually changing. Since it is regularly recalculated, the algorithm may reliably progress in the direction of the negative gradient despite having a very low learning rate. A low learning rate is more accurate, but calculating the gradient is time-consuming, thus reaching the bottom will take a very long time, as shown in Figure 3.25 and Figure 3.26.

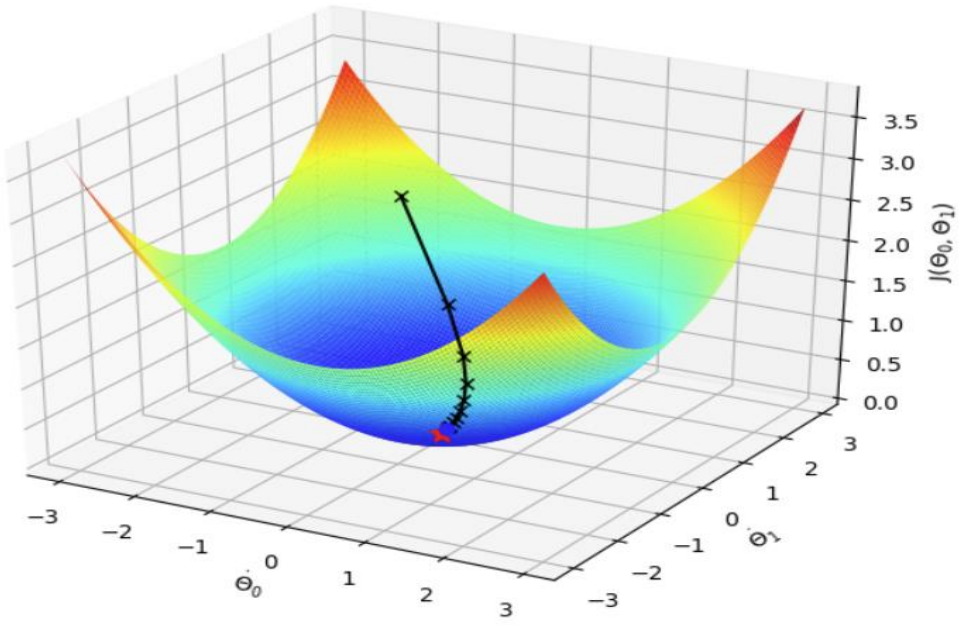

Figure 3.25: Greadient Descent on the convex surface [67]

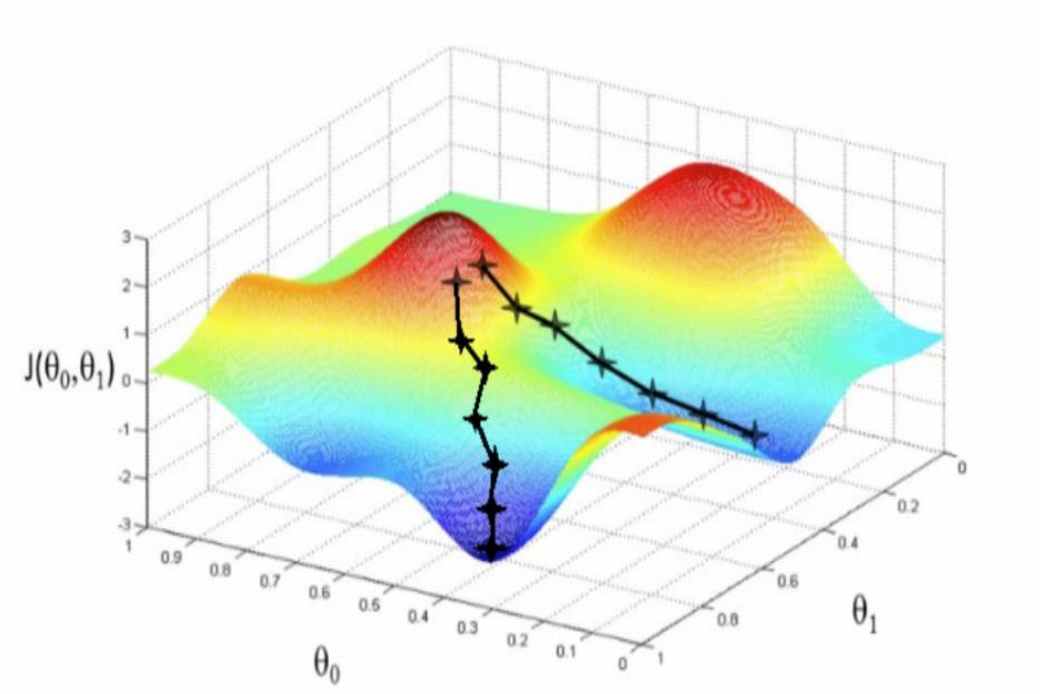

Figure 3.26: Gradient Descent on the non-convex surface [67]

3.3.2 Support Vector Regression

Support-vector machines are supervised learning models with related learning algorithms that examine data utilized for Classification and Regression analysis. In addition, the Regression problem is a generalization of the Classification problem, where the model returns a continuous-valued output rather than an output from a limited collection.

Alternatively, a Regression model estimates a continuous-valued multivariate function. This subsection begins with an introduction to support vector machines, followed by the theory and mathematics of support vector Regression, a technique used to solve the flight delay problem.

Support Vector Machines

Before discussing the function of Support Vector Regression, it is necessary to understand and describe certain fundamental aspects of the function and the concept behind support vector machines. Support Vector Machines (SVM) resolve binary Classification issues by recasting them as convex optimization issues [57]. Finding the largest margin separating hyperplane, while correctly categorizing as many training points as possible constitutes the optimization challenge. This ideal hyperplane is represented by support vectors in SVMs.

The purpose of the support vector machine algorithm is to classify data points using a hyperplane in an N-dimensional space (N — number of characteristics) [39]. Numerous potential hyperplanes might be chosen to divide the two classes of data points, as shown in Figure 3.27. The goal is to locate the plane with the greatest margin, or the greatest distance between data points of both classes. Maximizing the margin distance gives additional support so that subsequent data points can be classified with greater assurance.

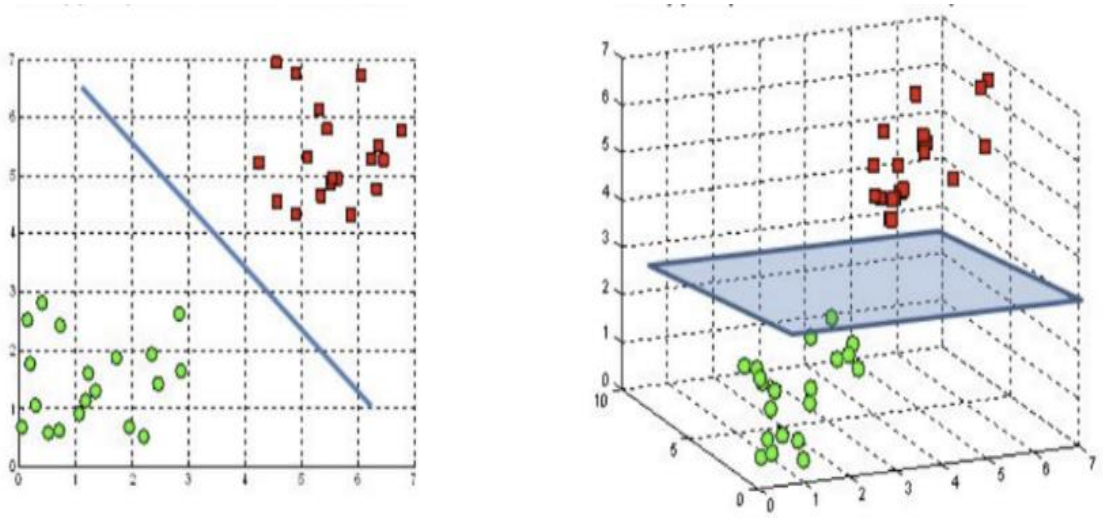

Figure 3.27: SVM hyperplanes in 2D and 3D feature space [69]

Margin is the distance between the hyperplane and the closest data point, while support vectors are the nearest data points defining the hyperplane, as shown in Figure 3.28. Margin is an area in which there are no data points. If the optimal hyperplane is close to the data points, then the margin will be very narrow and it will generalize well for training data, but it may fail to generalize for unknown data. Therefore, the objective is to maximize the margin, such that the classifier can generalize effectively to unseen examples.

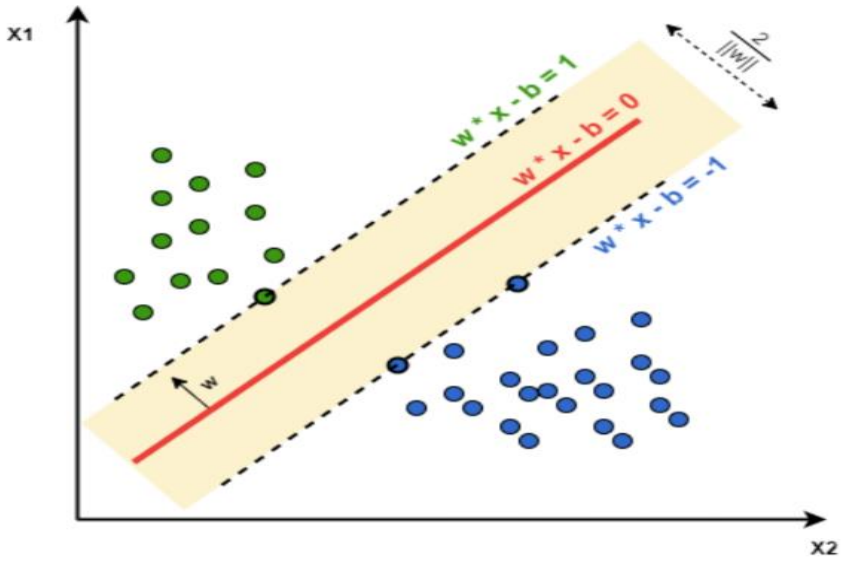

Figure 3.28: SVM maximum margin classifier [70]

Kernel

A kernel aids in locating hyperplanes in higher-dimensional spaces without increasing computing expense. Typically, the dimension of the data will raise the processing cost. It is required to move to a higher dimension, because the SVN may not be able to locate a separating hyperplane in the supplied dimension. The "kernel trick" [58] is that the kernel method only represents the data through a collection of pairwise similarity comparisons between the original data observations x (with the original coordinates in the lower-dimensional space), as opposed to explicitly applying the transformations $\varphi(\chi)$ and depict the information by these transformed coordinates in the higher-dimensional feature space, as shown in Figure 3.29. In kernel methods, the data set X is represented as a nxn kernel matrix of pairwise similarity comparisons with entries (i, j) determined by the kernel function: $k(x_i, x_j)$. This kernel function possesses a certain mathematical characteristic. The kernel function is a technique for determining the dot product of two vectors x and y in a (high-dimensional) feature space and is equivalent to a modified dot product, as shown in Figure 3.30. From a mathematical perspective, a kernel function is defined as:

$$k(x, y) = \langle \varphi(x), \varphi(y) \rangle$$

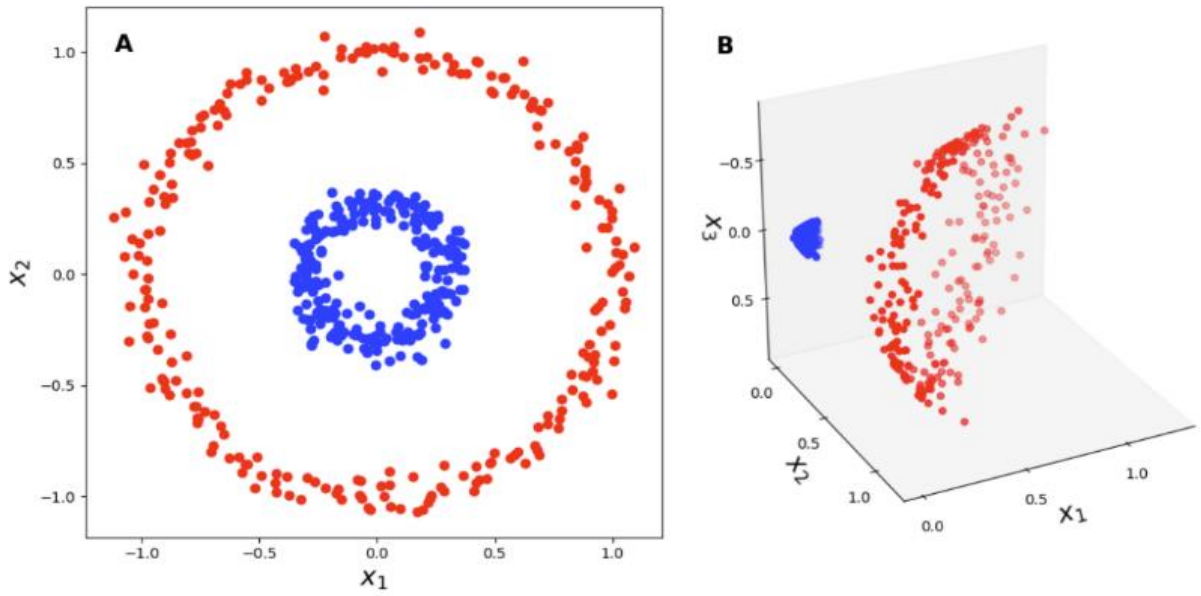

Figure 3.29: The Kernel Trick graphically 2D-3D [72]

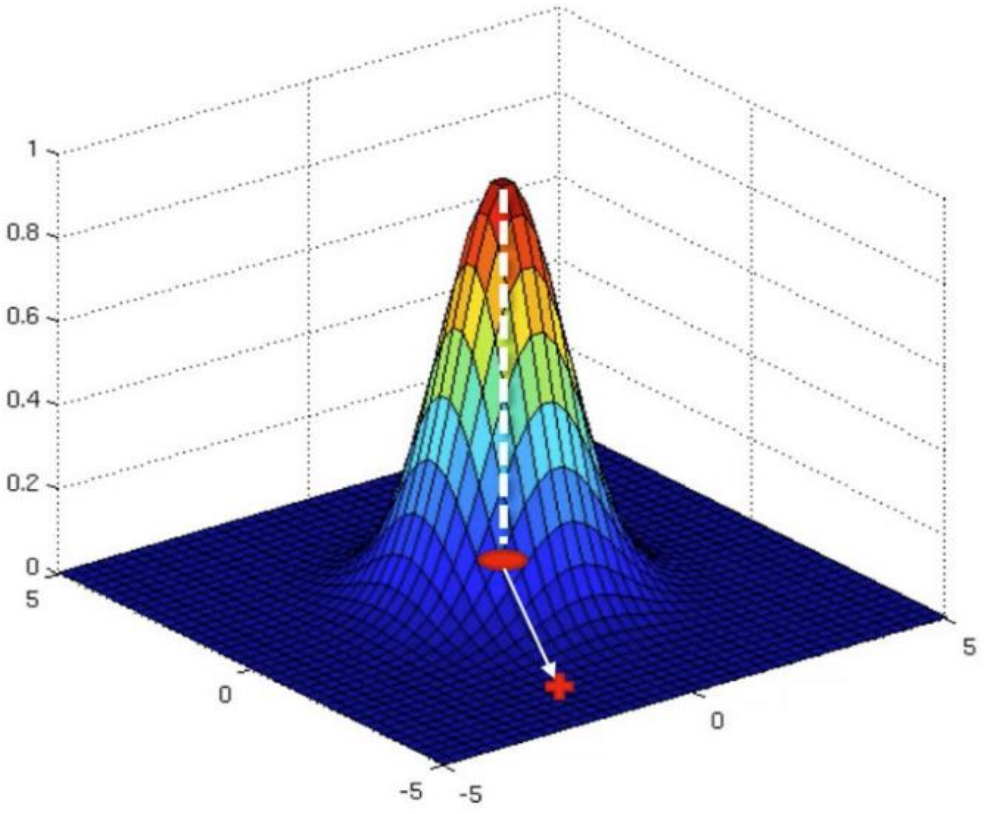

Figure 3.30: Visual representation of data points using Kernel Function [73]

Radial Basis Function Kernel

Radial Basis Function kernel, commonly known as RBF kernel or Gaussian kernel [59], is a kernel in the shape of a radial basis function (particularly a Gaussian function). The RBF kernel is characterized by:

$$K_{RBF}(x, x') = \exp[-\gamma \|x - x'\|^2$$

where γ is a parameter that determines the kernel's "spread." The parameter γ determines the bell-shaped curve's width. The bell will narrow as γ approaches its maximum value. Small values of γ produce broad bells, as shown in Figure 3.31.

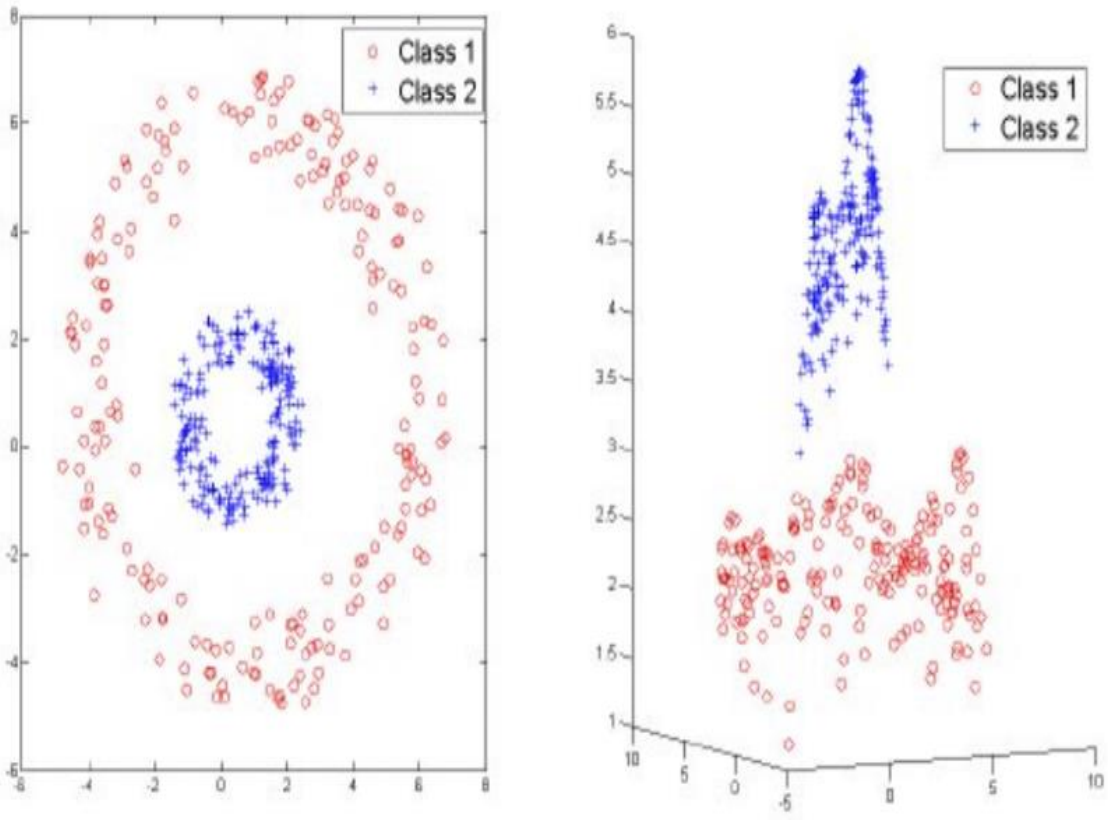

Figure 3.31: Exemplification of RBF kernel from nonlinear to high-dimensional space [60]

Decision Boundary

A decision boundary can be considered as a dividing line, on one side of which are positive elements, and on the other, negative instances. Exactly on the line, the examples can be categorized as positive or negative.

Background Theory and mathematics underlying SVR

Support Vector Machines tackle binary Classification issues by recasting them as convex optimization issues [61]. Finding the largest margin separating hyperplane, while correctly categorizing as many training points as possible, constitutes the optimization challenge. This ideal hyperplane is represented by support vectors in SVMs. The SVM's sparse solution and strong generalizability facilitate its adaption to Regression situations. SVM generalization to SVR is achieved by creating an e-insensitive zone termed the e-tube around the function. This tube reformulates the optimization problem to discover the tube that most closely approximates the continuous-valued function, while balancing model complexity and prediction error.

Specifically, SVR is formulated as an optimization problem by constructing a convex e-insensitive loss function to be reduced and locating the tube containing the majority of training cases that is the flattest. Consequently, a multi-objective function is derived from the loss function and the tube's geometrical features. Using proper numerical optimization algorithms, the convex optimization, which has a unique solution, is then solved. The hyperplane is represented by support vectors, which are training samples located outside of the tube's perimeter.

Consider the training data $\{(x_1, y_1), \dots, (x_n, y_n)\} \in X \times R$, where X specifies the space of the input patterns (e.g., $X = R^d$). The objective of ε-SV Regression is to find a function $f(x)$ that deviates from the actual targets y_i for all training data by no more than ε and is as flat as possible. In other words, we do not mind errors as long as they are less than ε, but we will not accept deviations more than this. The approximate continuous-valued function can be expressed as below, in addition, for multidimensional data, increase x by one and add b to the w vector to simplify the mathematical notation and produce the multivariate Regression Equation.

$$y = f(x) = \langle w, x \rangle + b = \sum_{j=1}^{N} w_j . x_j + b \;\; where\; y, b \in \mathbb{R}, x, w \in \mathbb{R}^N$$

Therefore, N is the degree of the approximating polynomial. Additional w_j are nonzero as the magnitude of the vector w grows, resulting in higher-order solutions. The horizontal line is a 0th-order polynomial solution with a significant divergence from the anticipated outputs, and hence a significant mistake. The linear function, a 1st-order polynomial, yields superior approximations for a subset of the data, but underfits the training data overall. The optimal trade-off between function flatness and the prediction error is achieved by the sixth-order solution. The highest-order solution has zero error, but a high level of complexity, and it will likely overfit data that has not yet been observed, as shown in Figure 3.32. The magnitude of w functions as a regularizing term and delivers the optimization problem control over the solution's flatness.

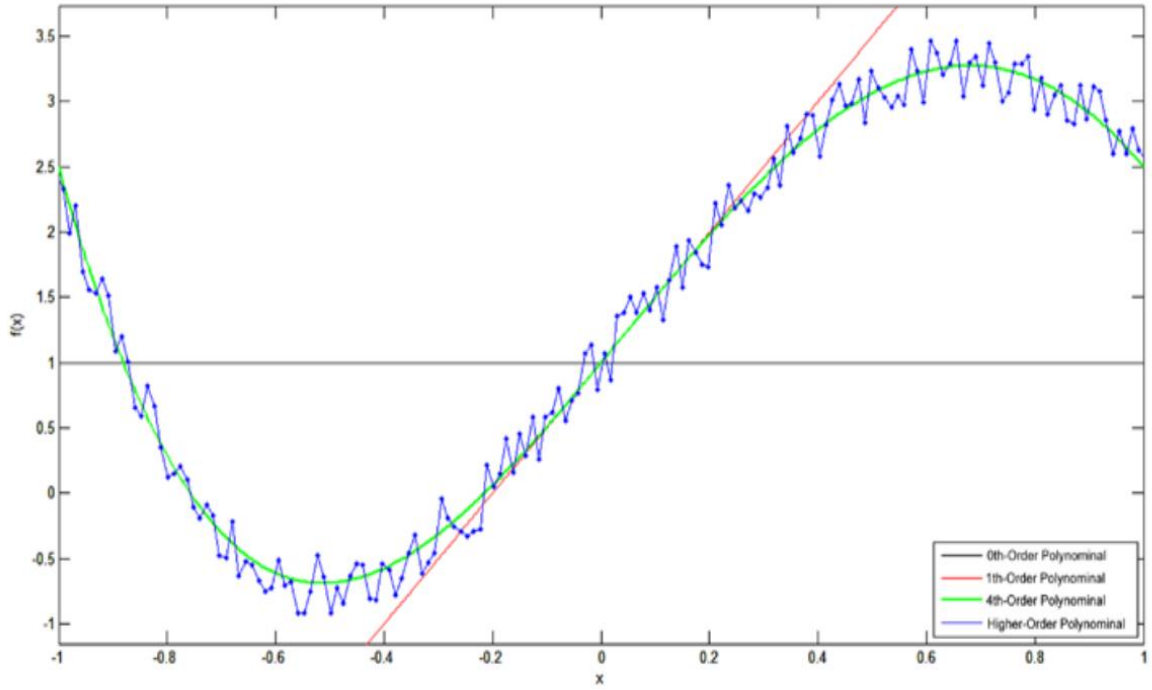

Figure 3.32: SVR Solution with Various Orders [62]

To achieve minimum generalization error, i.e. the ideal solution, given a set of functions with the same empirical/training error, SVM should minimize Machine Learning capacity and, consequently, model complexity. Consequently, this issue may be seen as a convex optimization problem:

$$minimize \ \frac{1}{2}\|w\|^2$$
$$subject\ to \ \begin{cases} y_i - \langle w, x_i \rangle - b \leq \varepsilon \\ \langle w, x_i \rangle + b + y_i \leq \varepsilon \end{cases}$$

In Figure 3.33, the implicit assumption was that a function f exists that approximates all pairings (x_i, y_i) with precision ε, where ε is the radius of a tube within which the Regression function must reside following successful learning. Or, in other words, that the convex optimization problem is feasible. However, this is not always the case, some errors are permitted. Similar to the "soft margin" loss function [63] that Cortes and Vapnik [61] utilized in SV machines, one can incorporate slack variables ξ_i, ξ_i^* to deal with otherwise infeasible restrictions of the optimization problem, equation below. Hence the formulation stated:

$$minimize \quad \frac{1}{2}\|w\|^2 + C\sum_{i=1}^{n}(\xi_i, +\xi_i^*)$$

$$subject\ to \quad \begin{cases} y_i - \langle w, x_i\rangle - b \le \varepsilon + \xi_i \\ \langle w, x_i\rangle + b - y_i \le \varepsilon + \xi_i^* \\ \xi_i, \xi_i^* \ge 0 \end{cases}$$

The constant $C > 0$ sets the trade-off between the flatness of f and the tolerance for deviations greater than ε. This refers to dealing with a so-called "ε-insensitive loss function $|\xi|_\varepsilon$" defined as:

$$|\xi|_\varepsilon := \begin{cases} 0, if\ |\xi| \le \varepsilon \\ |\xi| - \varepsilon, otherwise \end{cases}$$

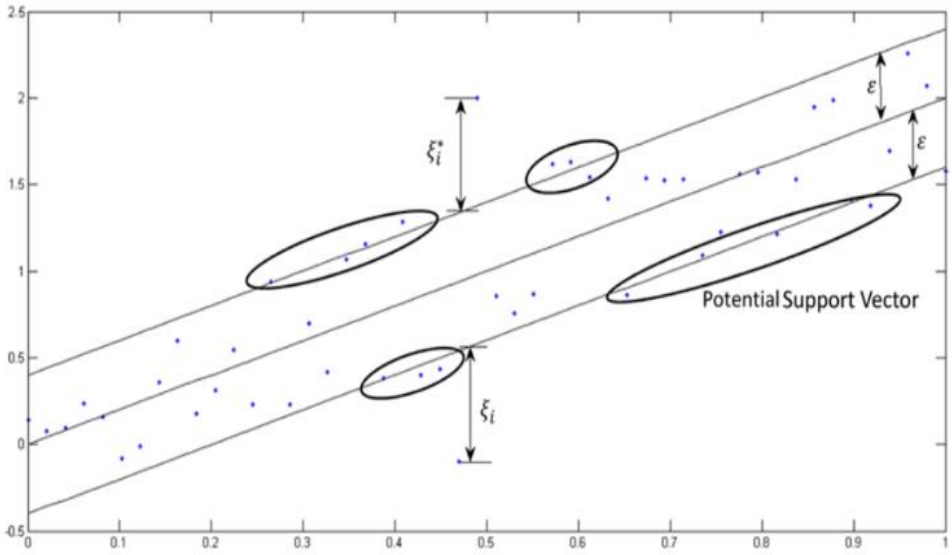

Figure 3.33: One dimensional Linear SVR [77]

The goal is to lower the complexity of the model by tolerating errors to a certain extent. Figure 3.34 compares Vapnik's "ε-insensitivity error function to the quadratic error $(y - f(x, w))^2$ and the absolute error $|y - f(x, w)|$, two classical error functions.

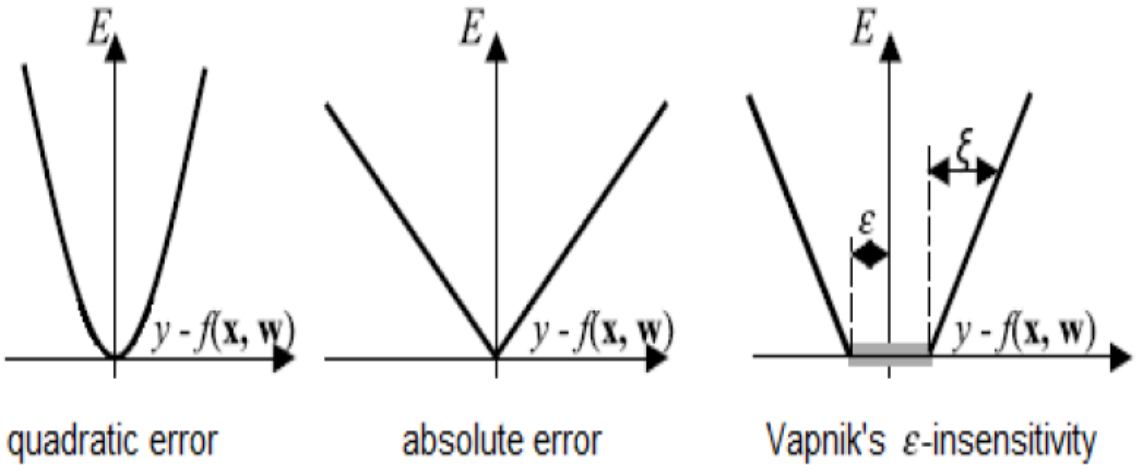

Figure 3.34: Error Function presentation [79]

Dual problem and quadratic programs

By introducing a dual set of variables, the important concept is to construct a Lagrange function from the objective function (primary objective function) and the accompanying constraints [64], [65]. The equation below is a typical quadratic optimization problem with inequality constraints. Using the saddle points λ of the Lagrangian function L, this optimization problem might be addressed as:

$$L(x, \lambda) = f(x) + \sum_{k} \lambda_k g_k$$

Where $f(x)$ is the objective function, while $g_k = 0$ represents the constraints. The domain of f must be an open set containing every point that satisfies the requirements. In addition, f and g_k must possess continuous first partial derivatives and the gradients of g_k cannot be 0 on the domain [66].

The primal Langrangian function is defined as:

$$Lp(w, b, \xi_i, \xi_i^*, a_i, a_i^*, \beta_i, \beta_i^*) = \frac{1}{2}\|w\|^2 + C\sum_{i=1}^{n}(\xi_i, +\xi_i^*)$$

$$-\sum_{i=1}^{n} a_i(\varepsilon + \xi_i - y_i + \langle w, x_i \rangle + b)$$
$$-\sum_{i=1}^{n} a_i(\varepsilon + \xi_i^* - y_i - \langle w, x_i \rangle - b)$$
$$-\sum_{i=1}^{n} (\beta_i \xi_i + \beta_i^* \xi_i^*)$$

Solving a series of equations resulting from setting Langragian's derivation to zero $\nabla L = 0$ yields the stationary positions:

$$\partial_b L = \sum_{i=1}^{n} (\alpha_i^* - \alpha_i) = 0$$
$$\partial_w L = w - \sum_{i=1}^{n} (\alpha_i^* - \alpha_i) x_i = 0$$
$$\partial_{\xi_i} L = C - \alpha_i - \beta_i = 0$$
$$\partial_{\xi_i^*} L = C - \alpha_i^* - \beta_i^* = 0$$

After substituting these derivations into equation and deleting the dual variable (β_i, β_i^*) it yields a dual variable Lagrangian maximization [67] as shown below:

$$L_d(a_i, a_i^*) = \frac{1}{2} \sum_{i,j=1}^{n} (a_i - \alpha_i^*)\left(a_j - \alpha_j^*\right)(x_i, x_j)$$
$$-\varepsilon \sum_{i=1}^{n} (a_i - \alpha_i^*) + \sum_{i=1}^{n} y_i (a_i - \alpha_i^*)$$
$$subject\ to \sum_{i=1}^{n} (a_i - \alpha_i^*) = 0\ and\ a_i, a_i^* \in [0, C]$$

The equation can be expressed as:

$$w = \sum_{i,j=1}^{n} (a_i - \alpha_i^*) x_i$$
$$thus\ f(x) = \langle w, x \rangle + b = \sum_{i,j=1}^{n} (a_i - \alpha_i^*) \langle x_i, x \rangle + b$$

Where $f(x)$ is the linear kernel computation between two vectors.

While b is determined as [67]:

$$\max\{-\varepsilon + y_i - \langle w, x_i\rangle | a_i < C \text{ or } a_i^* > 0\} \leq b \leq min\{-\varepsilon + y_i - \langle w, x_i\rangle | a_i < C\}$$

The number of support vectors is equal to the number of nonzero a_i and a_i^* following the learning process. In addition, because *w* may be characterized as a linear combination of the training patterns x_i, the complexity of a function's representation is independent of the dimensionality of the input space and only depends on the number of support vectors [67]. This independence is a strength of SVM, when dealing with high-dimensional input, and it is also useful for formulating a nonlinear extension.

Kernel Function RBF

Complex data typically requires a more expressive function than a linear one (as mentioned above in the Kernel section). In this instance, SVM transforms the original input space nonlinearly into a higher-dimensional feature space [66]. In its dual formulation, the SVM algorithm relies solely on the inner products of the training samples. Then, the formula would consist of the inner products of the points within the feature space. This mapping into the hypothetical feature space is implicit, hence it is possible to use infinite-dimensional feature spaces. Integrating the kernel function into equation yields the following expression:

$$kernel\ function\colon k(x, y) = \langle \Phi(x), \Phi(y)\rangle$$

$$\begin{aligned} L_d(a_i, a_i^*) = \frac{1}{2}\sum_{i,j=1}^{n} (a_i - \alpha_i^*)\big(a_j - \alpha_j^*\big)k(x_i, x_j) \\ -\varepsilon\sum_{i=1}^{n} (a_i - \alpha_i^*) + \sum_{i=1}^{n} y_i (a_i - \alpha_i^*) \\ subject\ to \sum_{i=1}^{n} (a_i - \alpha_i^*) = 0\ and\ a_i, a_i^* \in [0, C] \end{aligned}$$

Not every kernel can be applied to the equation, only those that meet the symmetric, continuous, and positive semi-definite requirements are admissible [68]. Several confirmed kernel functions are admissible for Classification, such as the polynomial and the radial basis function (RBF) which is mentioned in the SVM section and is the default kernel parameter of the Scikit-learn library used in this thesis [69], nonlinear SVR represented in Figure 3.35.

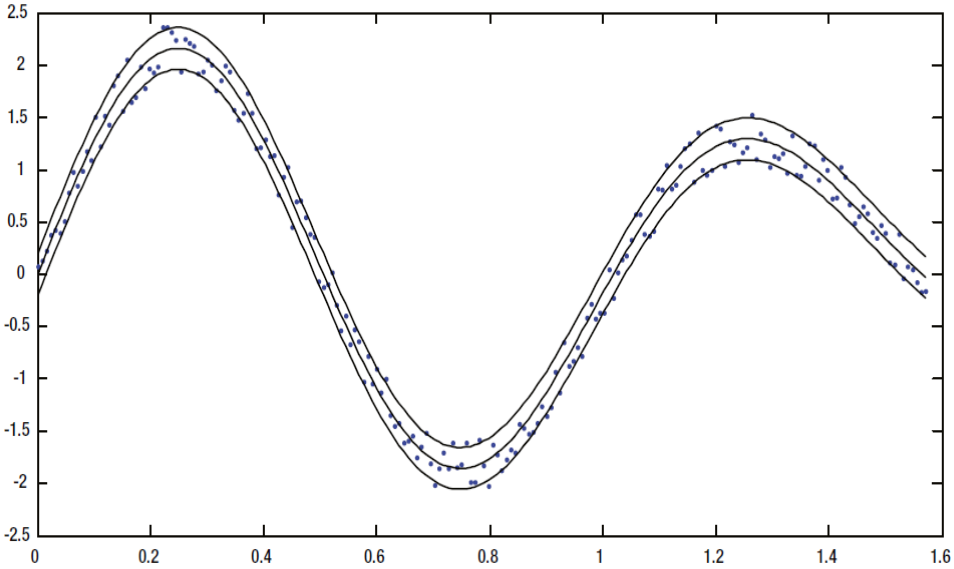

Figure 3.35: Nonlinear SVR [77]

Chapter 4

Implementation and Methodology

Global comparisons and benchmarking, including data analysis, can assist improve performance and find best practices in Air Traffic Management (ATM), as in any other industry. Throughout the years, numerous organizations have attempted to estimate the amount of inefficiency, in several scientific techniques and procedures as seen in Chapter 2 Related Work, that can be mitigated by enhancing the ATM system and reducing flight delays. These investigations inspired the research presented in this thesis and the application of contemporary methods, such as Machine Learning, for the numerical prediction of the Flight Delay problem. This chapter addresses the prediction methodology, the data utilized and preprocessed, the research questions given in this thesis, as well as assessment and validation testing.

4.1 Methodology

The objectives of this research were to:

1. Predict the departure delay from JFK to LAX
2. Predict the inflight delay from JFK to LAX
3. Predict the total delay from JFK to LAX with two different approaches

First, the departure delay problem for each flight from JFK to LAX is analyzed separately using the dataset JFK to LAX. In addition, a new element was added to the dataset, that of prior knowledge of the departure delay of the immediate N previous flights from JFK to any airport in the United States (dataset JFK to all airports), to analyze the contribution of this

element and to structure the prediction more accurately, given that flight delay is a multi-criteria phenomenon and general departure delays at an airport affect all flights during a given period.

Next, the prediction of delay for each individual flight between JFK and LAX using the dataset JFK to LAX. And in this model, the new element of prior knowledge about the in-flight delay of the immediate N previous flights from any U.S. airport to LAX was added to examine its impact and account for the LAX arrival delay factor.

Finally, the total delay problem between JFK and LAX the total delay is the sum of model 1 (departure delay) and model 2 (in-flight delay) whose prediction is re-modeled in two approaches. The first approach simply sums the results of model 1 and model 2 to obtain the total delay. A block diagram is provided below to make the structure of the approach easier to comprehend:

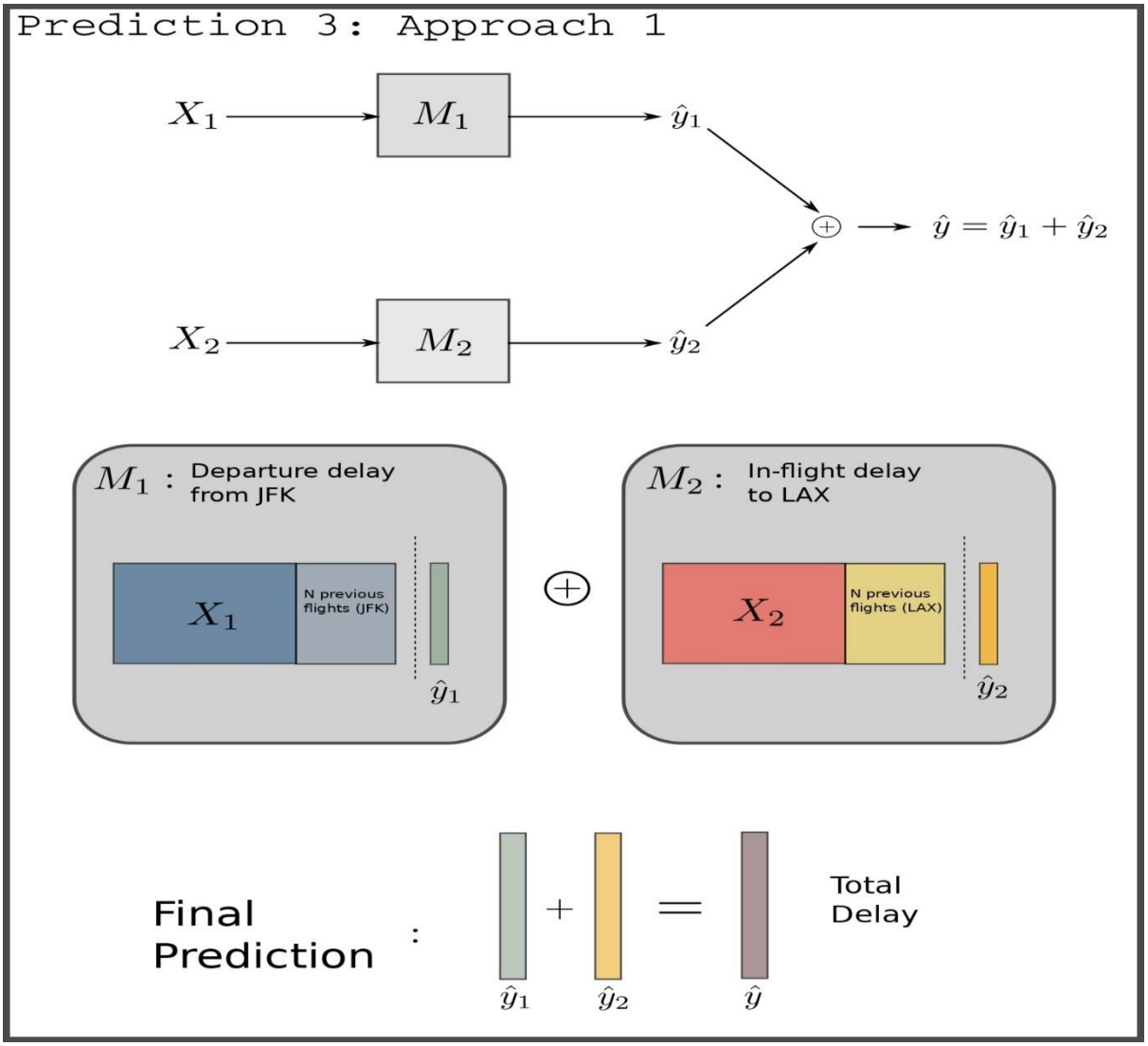

Figure 3.36: Total delay prediction Approach 1

The second approach seeks to anticipate the overall delay that a flight has, but in a single data model using attributes of the data of the model 1 prediction (Departure delay) and the data of the model 2 prediction (Inflight Delay), so that the total delay is recalculated in a single unified model. A block diagram of the second approach is represented below:

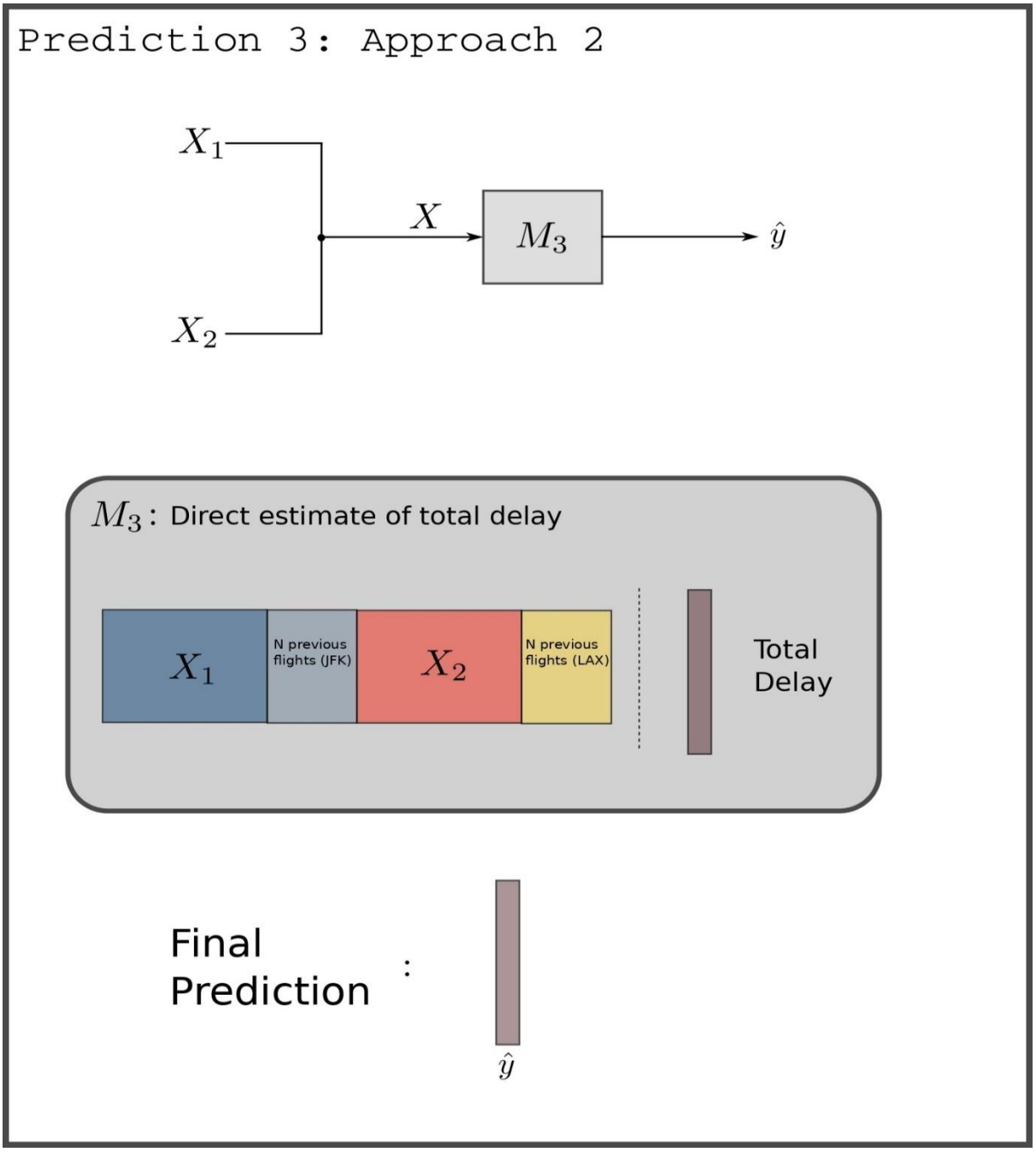

Figure 3.37: Total delay prediction Approach 2

4.2 Dataset

Using Machine Learning techniques for issue-solving and prediction is hindered by the lack of available data. The European Organisation for the Safety of Air Navigation (Eurocontrol) and the US Federal Administration Agency (FAA) provides the most comprehensive data in terms of availability, completeness, integrity, and accuracy following an internet search.

Flight data for the United States are provided by the Traffic Flow Management System (TFMS) [70]. In Europe, data are derived from the European Network Manager's Enhanced Tactical Flow Management System (ETFMS) [71]. The fifth report of joint ATM operational performance comparison [72] between the FAA and Eurocontrol revealed that despite the US CONUS airspace being 10% smaller than the European airspace, the US controlled approximately 57% more flights operating under Instrument Flight Rules (IFR) [73] with 24% fewer full-time Air Traffic Control Officers (ATCOs) than Europe in 2015. The average airspace density in the United States is higher than in Europe, and airports tend to be significantly larger. Due to the aforementioned factors and the ease of access to flight data, US airports were chosen for flight delay prediction research.

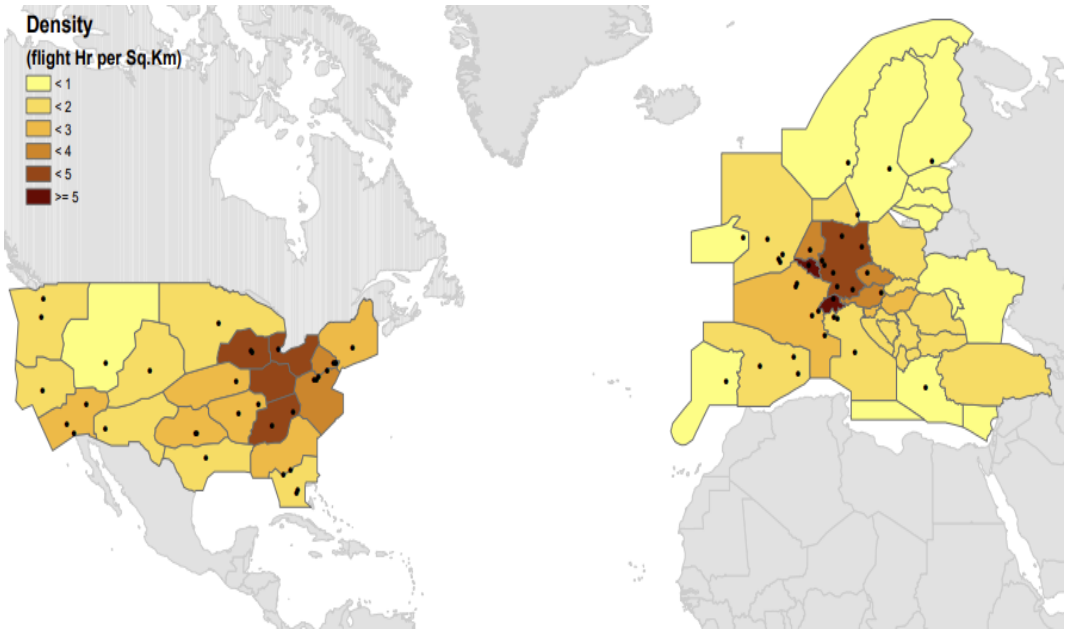

Figure 3.38: Traffic density in the US and Europe (2015) [88]

4.2.1 Dataset Identification and Acquisition

The dataset used to support this thesis was compiled by the Bureau of Transportation Statistics of the United States [6]. The data used for this study were collected from January to December 2015. The dataset is specifically supplied in comma-separated value (CSV) format. Each entry of the flights.csv file corresponds to a flight, and we can see that in 2015, 5.819.079 flights were logged. These flights are described by the following 31 variables:

- **YEAR, MONTH, DAY, DAY_OF_WEEK**: date of the flight
- **AIRLINE, FLIGHT_NUMBER, TAIL_NUMBER**: airline characteristics
- **ORIGIN_AIRPORT, DESTINATION AIRPORT**
- **SCHEDULED_DEPARTURE**: scheduled departure time
- **DEPARTURE_TIME**: actual departure time
- **DEPARTURE_DELAY**: the difference between the schedule and departure time
- **TAXI_OUT**: time spent by a flight between its actual off-block time and actual take-off time
- **WHEELS_OFF**: time that the aircraft lifts off from the origin airport
- **SCHEDULED_TIME**: scheduled date and time of the flight
- **ELAPSED_TIME**: time spent on air by an aircraft
- **AIR_TIME**: see ELAPSED_TIME
- **DISTANCE**: distance in miles covered by a flight
- **WHEELS_ON**: time in hours that an aircraft lands
- **TAXI_IN**: a period in minutes between the actual landing time and actual in-block time
- **SCHEDULED_ARRIVAL**: time of flight scheduled arrival in hours
- **ARRIVAL_TIME**: actual time of arrival in hours
- **DIVERTED**: if the flight for any reason (ex. Weather) diverted to another airport
- **CANCELLED**: if the flight canceled
- **CANCELLATION_REASON**
- **AIR_SYSTEM_DELAY**: delayed provided by FAA Ground Delay Program
- **SECURITY_DELAY**: delay in minutes for security reasons
- **AIRLINE_DELAY**: delay in minutes due to airline
- **LATE_AIRCRAFT_DELAY**: delay in minutes as a result of the late arrival of the previous flight
- **WEATHER_DELAY**: delay in minutes due to weather

The need to study and comprehend the dataset dictated the generation of Table 4.1 below, which details the types of variables in the data frame and the number of null values for each variable.

	Column type	Null values (nb)	Null values (%)
YEAR	Int64	0	0
MONTH	Int64	0	0
DAY	Int64	0	0
DAY_OF_WEEK	Int64	0	0
AIRLINE	Object	0	0
FLIGHT_NUMBER	Int64	0	0
TAIL_NUMBER	Object	14.721	0.25
ORIGIN_AIRPORT	Object	0	0
DESTINATION_AIRPORT	Object	0	0
SCHEDULED_DEPARTURE	Int64	0	0
DEPARTURE_TIME	Float64	86.513	1.48
DEPARTURE_DELAY	Float64	86.513	1.48
TAXI_OUT	Float64	89.047	1.53
WHEELS_OFF	Float64	89.047	1.53
SCHEDULED_TIME	Float64	6	0.0001
ELAPSED_TIME	Float64	105.071	1.8
AIR_TIME	Float64	105.071	1.8
DISTANCE	Int64	0	0
WHEELS_ON	Float64	92.513	1.58
TAXI_IN	Float64	92.513	1.58
SCHEDULED_ARRIVAL	Int64	0	0
ARRIVAL_TIME	Float64	92.513	1.58
ARRIVAL_DELAY	Float64	105.071	1.8
DIVERTED	Int64	0	0
CANCELLED	Int64	0	0
CANCELLATION_REASON	Object	5.729.195	98.45
AIR_SYSTEM_DELAY	Float64	4.755.640	81.72
SECURITY_DELAY	Float64	4.755.640	81.72
AIRLINE_DELAY	Float64	4.755.640	81.72
LATE_AIRCRAFT_DELAY	Float64	4.755.640	81.72
WEATHER_DELAY	Float64	4.755.640	81.72

Table 4.1: Type and null values for each dataset variables

Variables with a filling null factor greater than 80% are regarded as non-informative values and are omitted from the dataset. Such as “CANCELLATION_REASON, AIR_SYSTEM_DELAY, SECURITY_DELAY, AIRLINE_DELAY, LATE_AIRCRAFT_DELAY, WEATHER_DELAY”.

To provide a global picture of the geographical area covered through this information, airports and their locations are plotted, in Python 3.10, along with an indication of the number of flights recorded in each airport during 2015 to provide a global overview, as shown in Figure 3.39. This dataset contains 322 unique values of airports and 14 unique values of airlines, as shown in Table 4.2.

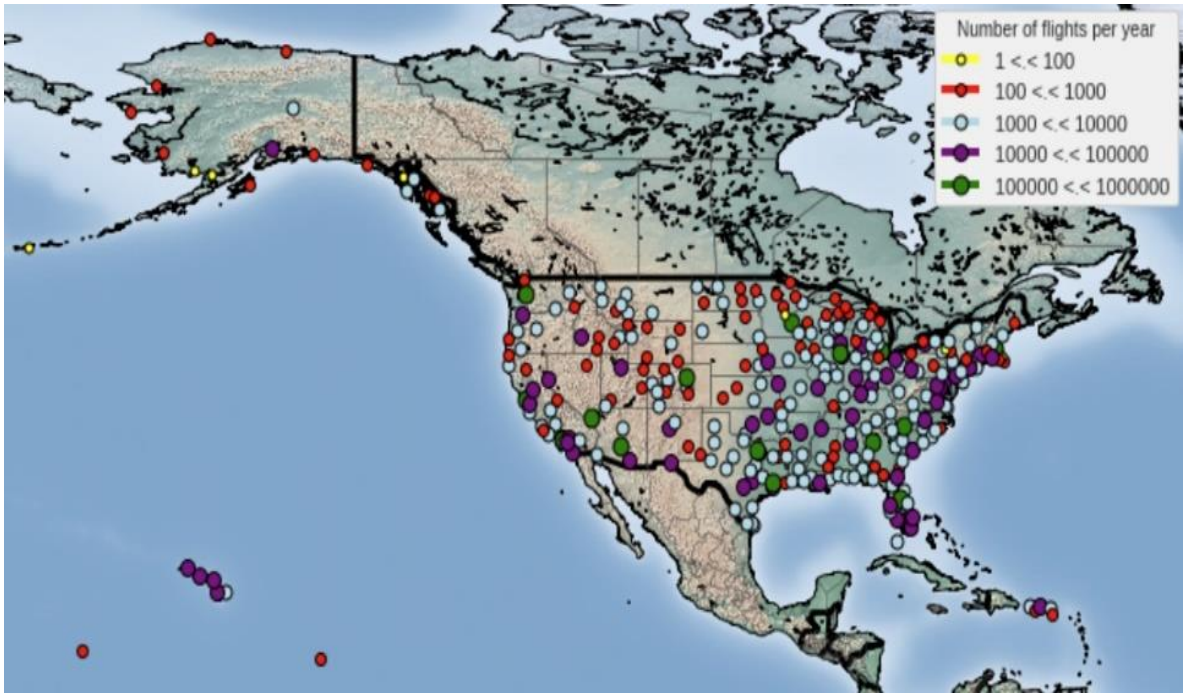

Figure 3.39: Airports and the number of flights in the US 2015 dataset

	IATA CODE [74]	AIRLINE
0	UA	United Air Lines Inc.
1	AA	American Airlines Inc.
2	US	US Airways Inc.
3	F9	Frontier Airlines Inc.
4	B6	JetBlue Airways
5	00	Skywest Airlines Inc.
6	AS	Alaska Airlines Inc.
7	NK	Spirit Air Lines
8	WN	Southwest Airlines Co.
9	DL	Delta Air Lines Inc.
10	EV	Atlantic Southeast Airlines
11	HA	Hawaiian Airlines Inc.
12	MQ	American Eagle Airlines
13	VX	Virgin America

Table 4.2: Airlines operating in the dataset

4.2.2 Data Processing

Many Machine Learning algorithms make data-related assumptions. It is consequently vital to arrange data to disclose the problem's structure most effectively to Machine Learning Techniques.

The thesis dataset contains more than 5,800,000 flights, 322 airports, 14 airlines and 31 unique variables. To model the problem and ultimately reduce the quantity of data and entries, a heuristic strategy was employed. Specifically, a pair of airports in the United States is selected based on the airport pair with the highest number of flights between them in 2015. Using Python 3.1, it was determined that JFK (John F. Kennedy International Airport) and LAX (Los Angeles International Airport) meet this requirement.

To limit the vast size of the dataset to flights and airports, three new datasets covering the entire year 2015 were constructed. The first and main one comprises 11.853 entries for flights between JFK and LAX airports. The second dataset contains flights from JFK to all U.S. airports, totaling 91.663 flights. The objective is to analyze other departure flights from JFK airport (including departures to LAX) that may affect flights from JFK to LAX due to their delays, as well as the departure delay prevalent at each hour at JFK airport. The purpose of the third dataset, which includes flights from all USA airports to LAX airport (including arrivals from JFK) and has 192.136 flight records, is to examine the arrival delay prevalent at each hour during the arrival of a flight from JFK to LAX. That is, whether other flights arriving at LAX from U.S. airports affect JFK-to-LAX arrivals. For greater clarity, the Figure 4.40 illustrates flight delays at US airports at the 12 p.m. peak hour on 7th September 2015:

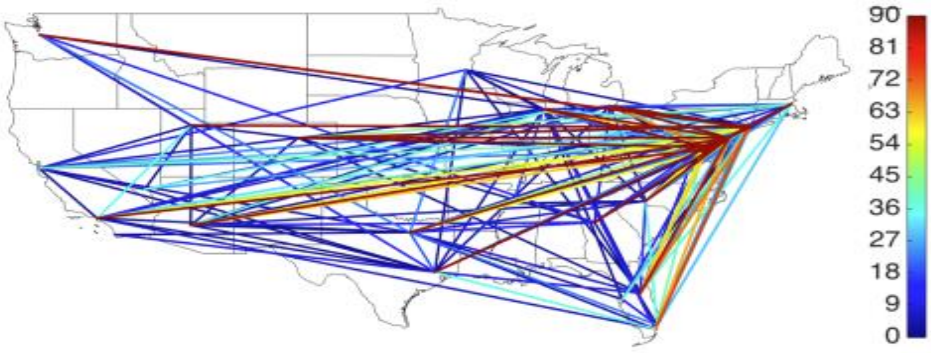

Figure 4.40: Departure delay of flights between US airports in minutes [91]

Then, among the 31 variables, the eleven variables listed below were chosen in a manual investigation and testing: 1) SERIAL FLIGHT NUMBER 2) AIRLINE 3) ORIGIN AIRPORT 4) SCHEDULED DEPARTURE 5) DEPARTURE TIME 6) DEPARTURE DELAY 7) SCHEDULED ARRIVAL 8) ARRIVAL TIME 9) ARRIVAL DELAY 10) SCHEDULED TIME 11) ELAPSED TIME. Thus, the dataset for the first five JFK-LAX flights is shown the following Table 4.3 (DEPARTURE DELAY and ARRIVAL DELAY correspond to target variables):

AIRLINE	ORIGIN AIRPORT	DESTINATION AIRPORT	SCHEDULE DEPARTURE	DEPARTURE TIME	DEPARTURE DELAY	SCHEDULED ARRIVAL	ARRIVAL TIME	ARRIVAL DELAY	SCHEDULED TIME	ELAPSED TIME
B6	JFK	LAX	1/1/2015 6:30	6:27:00	-3	9:47:00	09:51	4	377	384
VX	JFK	LAX	1/1/2015 7:00	6:55:00	-5	10:25:00	10:27	2	385	392
AA	JFK	LAX	1/1/2015 7:00	6:49:00	-11	10:20:00	10:26	6	380	397
DL	JFK	LAX	1/1/2015 7:00	7:02:00	2	10:40:00	10:30	-10	400	388
B6	JFK	LAX	1/1/2015 8:15	8:10:00	-5	11:35:00	11:37	2	380	387

Table 4.3: The initial five components of the dataset

Data cleaning

Data cleaning is the process of preparing data for analysis by removing extraneous or inaccurate information. This is typically information that can have a negative effect on a model or algorithm by reinforcing an incorrect notion. In addition to deleting portions of unneeded data, data cleaning is frequently related to correcting inaccurate information.

Before data can be processed further, unwanted data in the form of outliers must be deleted. Outliers are the most difficult to detect among all other data errors. Before rejecting a data point or series of data points as an outlier, a comprehensive investigation is typically performed. Specific models with an extremely low outlier tolerance can be easily affected by a substantial number of outliers, hence diminishing the quality of the predictions. The dataset contains missing values. Consequently, cases with missing labels are eliminated from the dataset. Another pitfall to avoid is 'random' delays, with a particular focus on excessive delays (over 90 minutes). During the inquiry, it was discovered that delays of many hours (or even tens of hours) were sometimes observed. However, this type of delay is negligible (a few percent) and the cause of these delays is likely due to unforeseen events (weather conditions, breakdowns, accidents). Nonetheless, they generate maximum values that affect the forecasts and generally represent a problem that is not addressed in this thesis and is therefore were eliminated.

In addition, flights with departure delays of less than ten minutes are deemed outliers and are eliminated since they are the result of flight plan errors or causes not examined in this thesis.

One-Hot Encoding

In the thesis dataset, airlines are classified as categorical data. In Regression, Machine Learning algorithms treat the order of numbers as a significant characteristic. In other words, a bigger number will be perceived as superior or more significant. One hot encoding [75] is one approach for preparing data for an algorithm and obtaining a more accurate forecast. Each category value is converted into a new categorical column and assigned a binary value of 1 or 0 using one-hot. Each integer value is represented as a vector of binary digits. Each value is zero, and the index is indicated by a 1. The following Table 4.4 illustrates the categorical to the binary transformation of the five airlines operating from JFK to LAX:

IATA CODE	AIRLINE	ONE HOT ENCODING
AA	American Airlines	10000
DL	Delta Air Lines Inc.	01000
B6	JetBlue Airways	00100
VX	Virgin America	00010
UA	United Air Lines Inc.	00001

Table 4.4: One hot encoding for airlines

Cyclical features encoding

Variables such as SCHEDULED DEPARTURE, which constitute the basic feature of this thesis implementation, have the format DD/MM/YYYY HH: MM. However, there might be some hidden patterns in the dataset that will not be revealed by the regular features. For example cyclical patterns, such as hours of the day, days of the week, months, seasons, etc. The difficulty arises when translating this data into an interpretable characteristic. A sinusoidal and cosinusoidal transform [76], as shown in Figure 4.41, is applied for the time data to maintain homogeneity and periodicity in the data and projections, as follows:

$$Hour_{sin} = \sin\left(\frac{2\pi Hour}{max(Hour)}\right)$$
$$Hour_{cos} = \cos\left(\frac{2\pi Hour}{max(Hour)}\right)$$

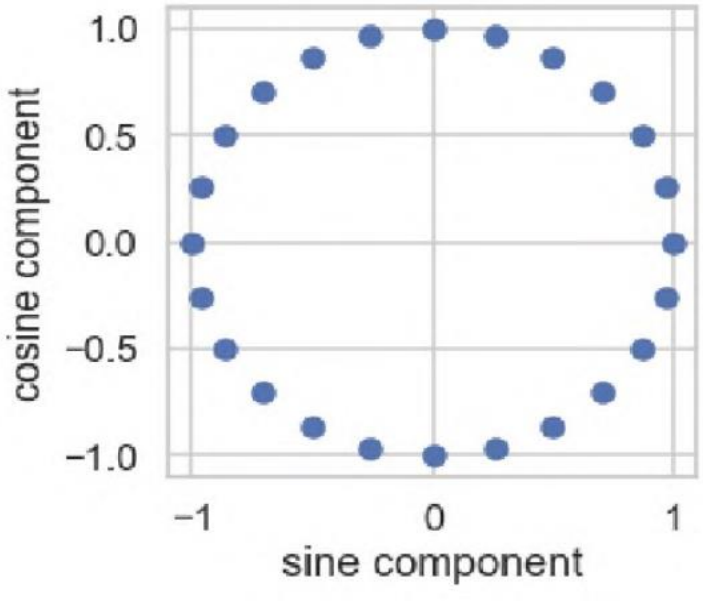

Figure 4.41: The sin/cos encoding in hour-of-day values from 0 through 23 [93]

Moreover, for precise processing and forecasting, the following preparations were made for the data. The day of the week was divided by 7, the week by 52, and the hour of the day by 24 since the year (given as 2015) was omitted. Additionally, we have not scaled in terms of minutes, as flight delays are quantified at a minimum per hour, such as 13:00-14:00. This coding enabled the model to recognize the general trend of the data by distinguishing periods of higher and lower values.

Feature Scaling

Scaling features is one of the most important phases in the preprocessing of data, before building a Machine Learning model. Scaling helps differentiate a weak machine-learning model from a stronger one.

Standardization of independent variables is required in Regression analysis, when the model comprises polynomial terms to model curvature or interaction effects. These terms provide essential information regarding the relationships between the independent factors and the dependent variable, but they also cause substantial multicollinearity. The Standard Scaler [77] assumes that data are normally distributed within each feature and scales them so that the distribution is centered on 0 (mean=0) and has a standard deviation of 1.

$$z = \frac{x - \mu}{\sigma}$$

The standard deviation with $\mu = 0$ and $\sigma = 1$ is also known as the standard normal distribution and is represented by the symbol $N(0,1)$. In addition, feature scaling is utilized since certain algorithms, such as neural network gradient descent, converge significantly faster with feature scaling than without it, as shown in Figure 4.42.

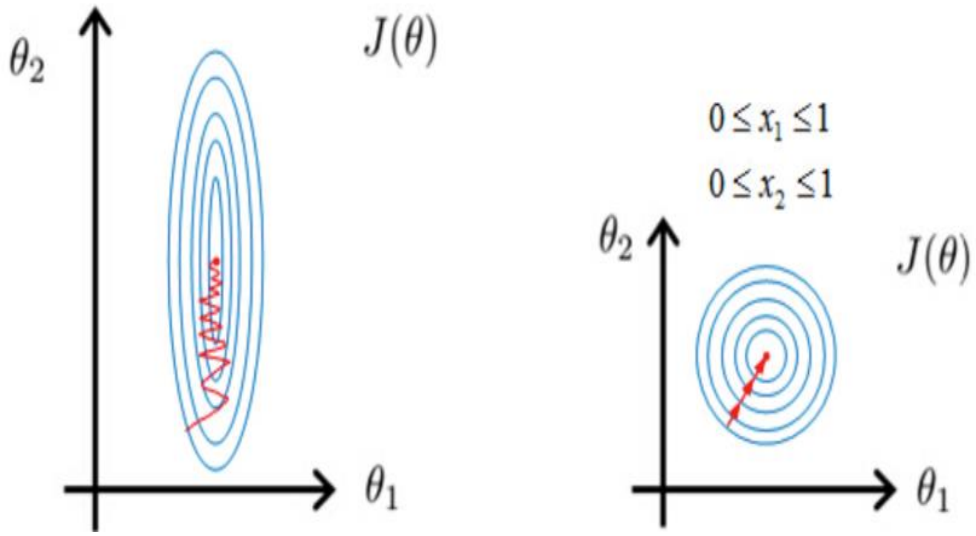

Figure 4.42: Gradient Descent convergence without and with scaling [95]

4.3 Research Questions

Motivated by the research gaps highlighted in Chapter 2 and the overarching goal to predict flight delay, the following research questions served as a guide for the overall research plan. Past research has focused on a variety of methodologies, including statistical, operational, and network methods. In addition, a limited benchmarking of Machine Learning techniques has been conducted to find appropriate techniques for creating prediction models for departure and in-flight delays between two airports, in comparison to the simple mean of delays.

Research Question 1

Could Machine Learning Techniques make a more accurate forecast than utilizing the simple average flight delay of the dataset?

Research Question 2

Does incorporating prior departure and in flight delays, and new feature knowledge, at the respective airports, enhance the Machine Learning predictions?

Research Question 3

Using the metrics of the Regression model, which of the two ways yields better results for the total delay prediction?

Research Question 4

Which Machine Learning model produces the most accurate predictions?

4.4 Validation, Testing and Evaluation

One of the key findings of this study was the successful identification of Machine Learning algorithms that facilitated precise predictions. The most effective approach for assessing the efficacy of an algorithm is to provide predictions for newly introduced data that was not previously encountered during the program's training phase. The assessment of model performance will serve as the concluding stage of this procedure. The assessment of student performance is of utmost importance, as it provides insights into how a learner will perform when faced with unfamiliar or upcoming knowledge. This phenomenon is investigated through the process of cross-validation, wherein the average of predictions generated by various Machine Learning Techniques is compared to the simple average. The evaluation of this comparison is based on the metrics outlined in the thesis.

4.4.1 K-fold Cross Validation

Cross-validation is a method for estimating the performance of a Machine Learning system with less variation than a traditional train-test split [78]. It functions by dividing the dataset into k-parts (such as $k = 5$ or $k = 10$). Each data split is known as a fold. The algorithm is trained on $k - 1$ folds, with one fold reserved for testing. This is repeated such that each fold of the dataset has an opportunity to serve as the held-back test set. Cross-validation yields k unique performance scores, which can be summarized with a mean and standard deviation.

The outcome is a more accurate estimation of the algorithm's performance on new data. It is more precise, because the algorithm is trained and assessed several times on a variety of data. The selection of k must permit the size of each test partition to be large enough to constitute a representative sample of the problem, while allowing sufficient

repetitions of the train-test evaluation of the algorithm to yield an accurate assessment of the method's performance on unseen data. Common k values for datasets containing thousands or tens of thousands of records are 3, 5, and 10. Finally, this thesis involves the use of 5-fold cross-validation.

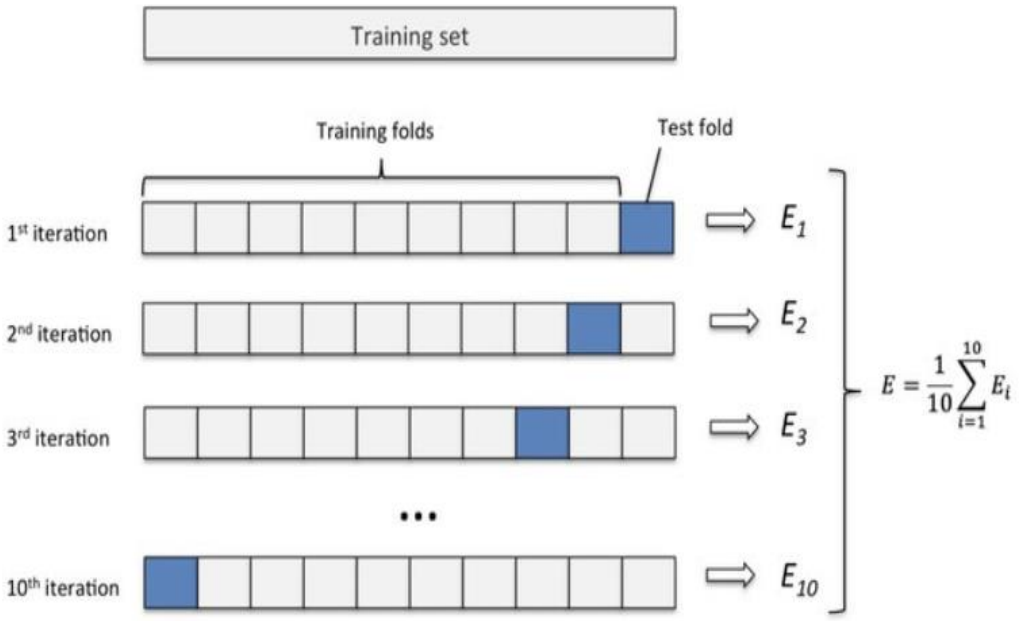

Figure 4.43: Diagram of k-fold cross-validation, where k=10 [97]

4.4.2 Evaluation Metrics

Numerical prediction learners are typically evaluated by analyzing how well the model fits the data. Four evaluation metrics were used to evaluate the numerical predictors: R-Squared, Mean Absolute Error, Root Mean Squared Error, and time performance in seconds.

Mean Absolute Error (MAE)

Mean Absolute Error refers to the distance between a model's predictions and actual values [35]. It is determined by calculating the mean absolute difference between the expected and actual values [79]. The smaller the Mean Absolute Error, the higher the model's performance.

$$MAE = \frac{\sum_{i=1}^{n} |y_i - \hat{y}_i|}{n}$$

Root Mean Squared Error (RMSE)

Root Mean Squared Error is the standard deviation of the difference between a model's predictions and the actual values [79]. The smaller the Root Mean Squared Error, the higher the model's performance.

$$RMSE = \sqrt{\frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2}$$

where $\hat{y}_i$ are the observations, y_i are the projected values of a variable, and n is the number of accessible observations for analysis. As RMSE is scale-dependent, it can only be used to compare the forecasting errors of different models or model configurations for a single variable and not between variables.

R-Squared Error (R^2)

The coefficient of determination, frequently identified as R-Squared is a statistical measure that quantifies the degree to which a model fits the data. Within the realm of regression analysis, it serves as a statistical metric that quantifies the degree to which the regression line accurately represents the observed data [30]. Therefore, it is crucial to utilize a statistical model for either predicting future outcomes or evaluating hypotheses. There are other variations, but one being provided here is extensively utilized:

$$R^2 = 1 - \frac{sum\ squared\ regression\ (SSR)}{total\ sum\ of\ squares\ (SST)}$$

$$= 1 - \frac{\sum(y_i - \hat{y}_i)^2}{\sum(y_i - \bar{y})^2}$$

The sum squared regression is calculated by summing the squared residuals, while the total sum of squares is obtained by summing the squared distances of the data points from the mean. Since it is expressed as a percentage, it will only have values ranging from 0 to 1.

4.4.3 Overfitting, Underfitting

A model is considered to be a good Machine Learning model, if it generalizes any new problem-domain input data appropriately. This allows us to predict future data that the data model has never seen before. Overfitting and underfitting are the primary causes of the poor performance of Machine Learning systems.

Underfitting

Underfitting occurs when a statistical model or Machine Learning algorithm cannot capture the underlying trend of the data, i.e., it performs well only on some training data, but badly on other training data. The Machine Learning model is rendered inaccurate by underfitting. Its recurrence merely indicates that our model or method does not adequately suit the data. It typically occurs, when there are insufficient data to develop an appropriate model, e.g. while attempting to build a linear model with insufficient non-linear data. In such situations, the Machine Learning model's rules are too simple and inflexible to be applied to such; hence, the model will likely produce a large number of incorrect predictions. Underfitting can be avoided by utilizing more data and limiting the number of characteristics through feature selection, or by increasing the model complexity (flexibility).

Overfitting

A statistical model is considered to overfit, when its predictions on testing data are not correct, although it performs well on the training data. When a model is trained with such a large flexibility in model parameters, it begins to learn from the noise and incorrect data entries in our data set. And when testing using test data, the variance is high. The model fails to appropriately classify the data, because there are too many details and noise. Non-parametric and non-linear approaches are responsible for overfitting, since these types of Machine Learning algorithms have greater leeway in developing the model based on the dataset and can therefore create truly unrealistic models, as shown in Figure 4.44. If we have linear data, we can avoid overfitting by utilizing a linear technique. Overfitting is a situation in which the evaluation of Machine Learning algorithms on training data differs from the evaluation of unseen data.

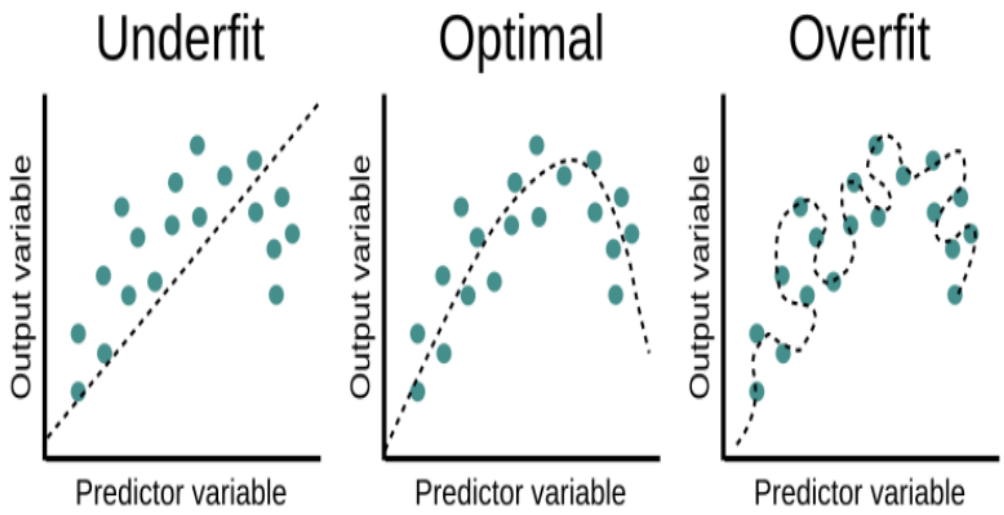

Figure 4.44: Underfit, Optimal and Overfit prediction models [99]

Chapter 5

Analysis and Results

This chapter details the Python 3.1 processes required to construct and evaluate each prediction model. The experimental findings will be presented, followed by a discussion centered on Section 4.4 metrics and training time. This includes detailing any evaluation steps conducted for the models.

5.1 Departure Delay Prediction from JFK to LAX

As previously stated, the purpose of this model is to forecast the mean departure delay between JFK to LAX airports in the United States. A comparative analysis was conducted on four Machine Learning approaches in order to determine the most appropriate methodology for the prediction model. The techniques evaluated were Linear Regression, Polynomial Regression, Support Vector Regression, and Feed Forward Neural Networks. This section outlines the procedures employed to construct the models utilizing the algorithms stated above, and presents an evaluation of their efficacy using the validation and testing datasets. To ensure the appropriate assessment of the algorithms, the data was partitioned into three distinct groups, namely training, 5-fold cross-validation, and testing datasets, by a random process. To establish periodicity in the data throughout the year at various levels (hourly, daily, weekly, and annually), we can apply a periodic transformation to the variable SCHEDULED_DEPARTURE. This transformation involves utilizing trigonometric functions,

such as sine (sin) and cosine (cos), to achieve the desired periodic behavior. By employing these functions, we can ensure that the data exhibits periodic patterns at different time intervals, ultimately enhancing our understanding of the variable's temporal variations. Furthermore, a table was incorporated to forecast the departure delay (DEPARTURE_DELAY) for flights from JFK to LAX and the dataset underwent a reduction resulting in a total of 11,853 flights.

In conjunction with the aforementioned methodology, an additional characteristic was established to compute the mean delay in DEPARTURE_DELAY for the N (the optimal number of flights for enhancing our outcomes can be explored) most recent flights departing from JFK airport to any destination. This attribute, as extra input on the prediction, aims to leverage insights regarding the typical surge in traffic at JFK airport during the departure time of each flight from JFK to LAX. The generation of characteristics in the method involved the utilization of one-hot encoding to represent the information about the airlines that operate between JFK and LAX. In the aforementioned implementation, a novel dataset was generated by extracting flights exclusively from JFK airport to any destination, resulting in a total of 91663 flights.

5.1.1 Linear Regression

The process of developing a prediction model using the Linear Regression technique in Python involves the following steps:

1. Load the data using “read.csv”
2. Train the model using the “LinearRegression” function and the training dataset
3. Test the performance of the model using the “predict” function and the validation dataset
4. Compute the Mean Absolute, Root Mean Squared Errors and R-Squared (R^2) using the difference between the predicted and actual mean values, execution time measured in seconds
5. Repeat steps 3 to 4 with the testing dataset

The results are presented in Figure 4.45:

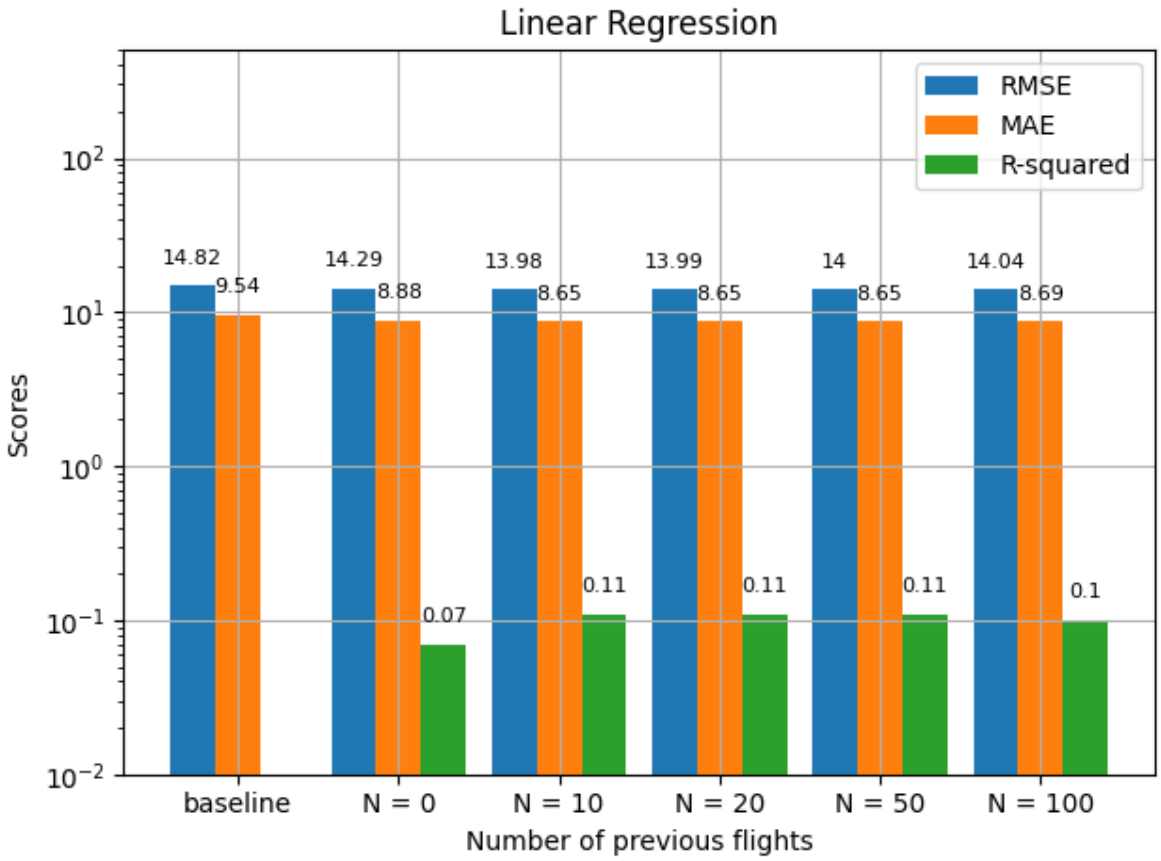

Figure 5.45: Departure delay prediction Linear Regression results

5.1.2 Polynomial Regression

The process of developing a prediction model using the Polynomial Regression technique in Python involves the following steps:

1. Load the data using “read.csv”
2. Train the model using the “LinearRegression” 2nd and 3rd grade function and the training dataset
3. Test the performance of the model using the “predict” function and the validation dataset
4. Compute the Mean Absolute, Root Mean Squared Errors and R-Squared (R^2) using the difference between the predicted and actual mean values, execution time measured in seconds
5. Repeat steps 3 to 4 with the testing dataset

The results are presented in Figures 4.46 and 4.47:

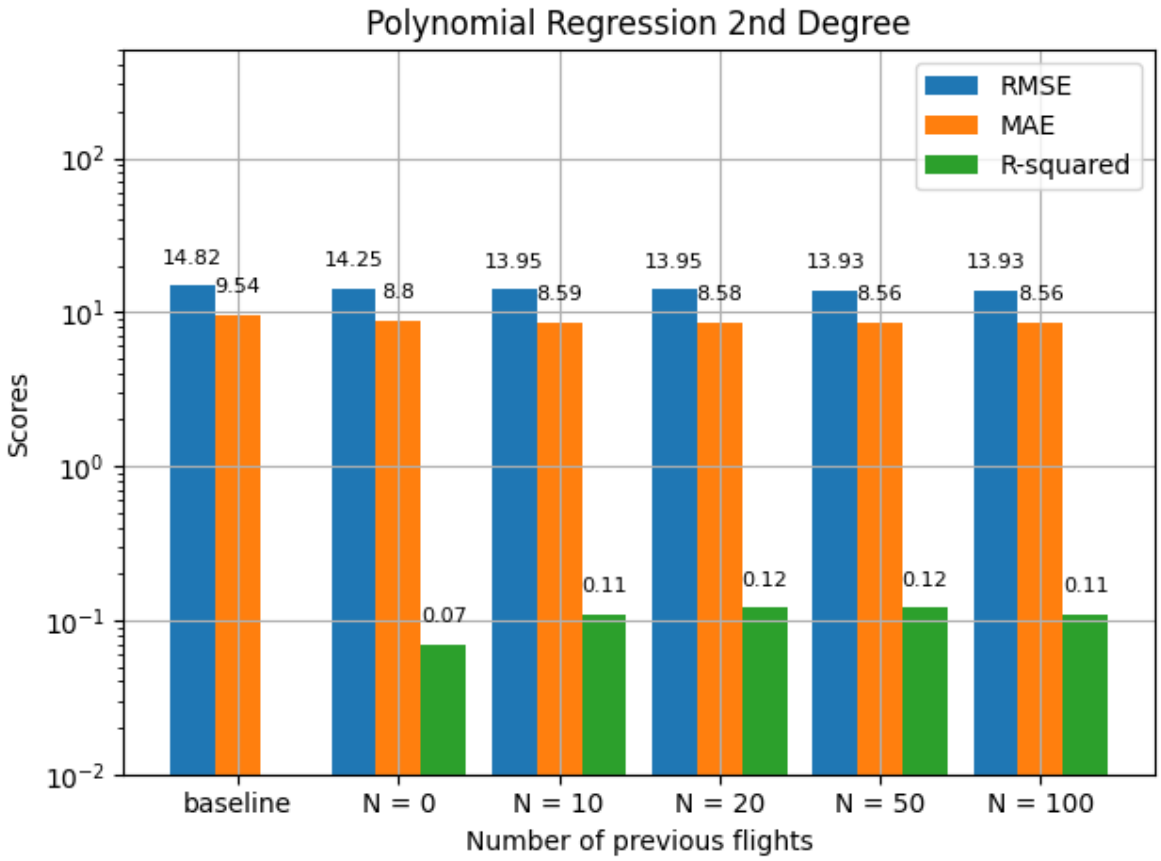

Figure 5.46: Departure delay prediction Polynomial Regression 2nd degree results

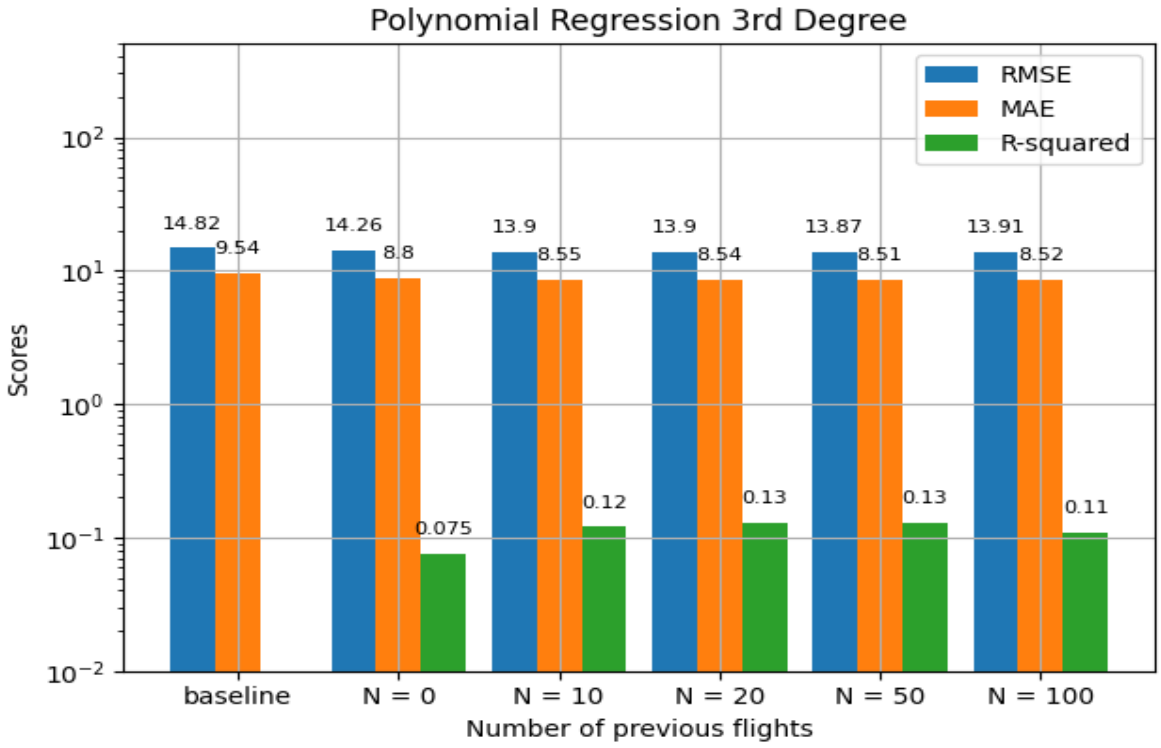

Figure 4.47: Departure delay prediction Polynomial Regression 3rd degree results

5.1.3 Support Vector Regression

The process of developing a prediction model using the Support Vector Regression technique in Python involves the following steps:

1. Load the data using “read.csv”
2. Train the model using the “SVR” function and the training dataset
3. Test the performance of the model using the “predict” function and the validation dataset
4. Compute the Mean Absolute, Root Mean Squared Errors and R-Squared (R^2) using the difference between the predicted and actual mean values, execution time measured in seconds
5. Repeat steps 3 to 4 with the testing dataset

The results are presented in Figure 4.48:

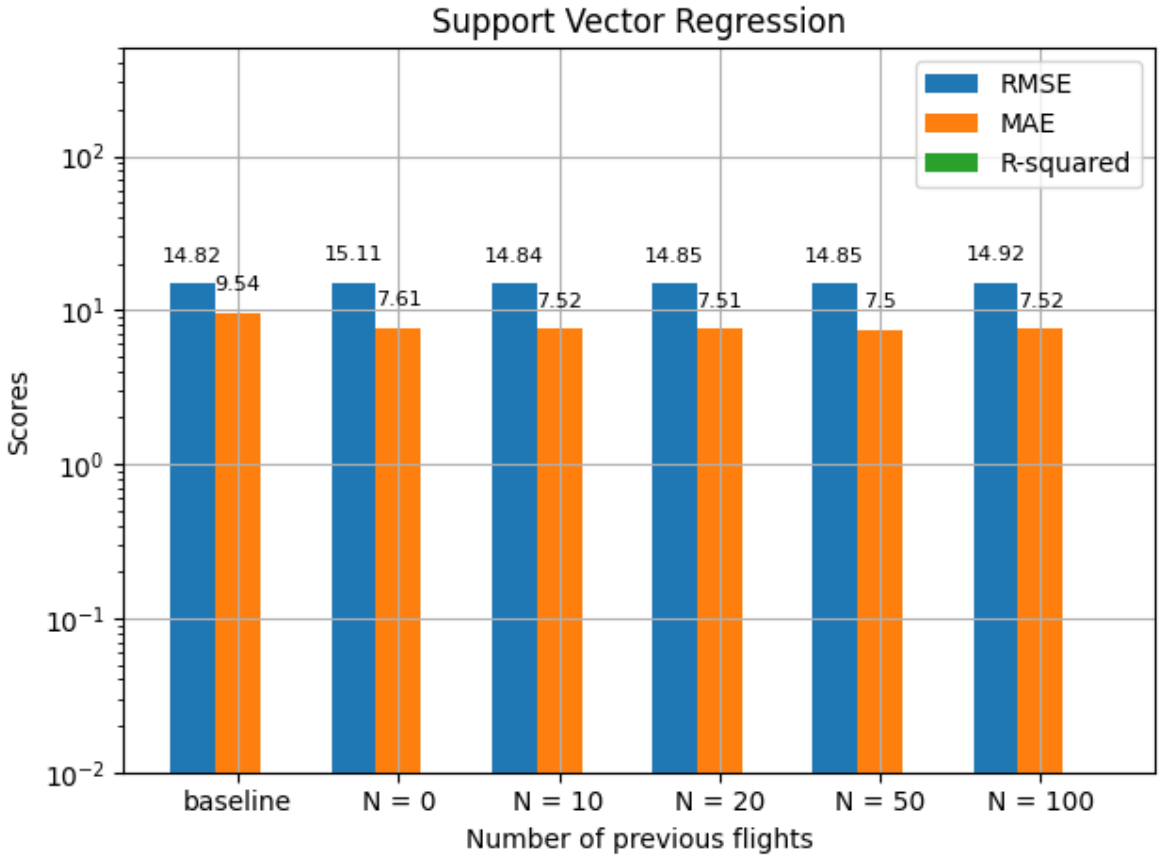

Figure 4.48: Departure delay prediction SVR results

5.1.4 Feed Forward Neural Network

The process of developing a prediction model using the Support Vector Regression technique in Python involves the following steps:

1. Load the data using “read.csv”
2. Normalize continuous variables to a 0 - 1 range, since Neural Networks perform best when predictors are scaled to a narrow range
3. Vary the number of hidden nodes and layers to identify the best hidden node setting for the model.
4. Train the model using “Model” function and the training dataset
5. Test the performance of the model using the “predict” function and the validation dataset
6. Compute the Mean Absolute, Root Mean Squared Errors and R-Squared (R^2) using the difference between the predicted and actual mean values, execution time measured in seconds
7. Repeat steps 5 to 6 with the testing dataset

The results are presented in Figures 4.49, 4.50, 4.51, 4.52:

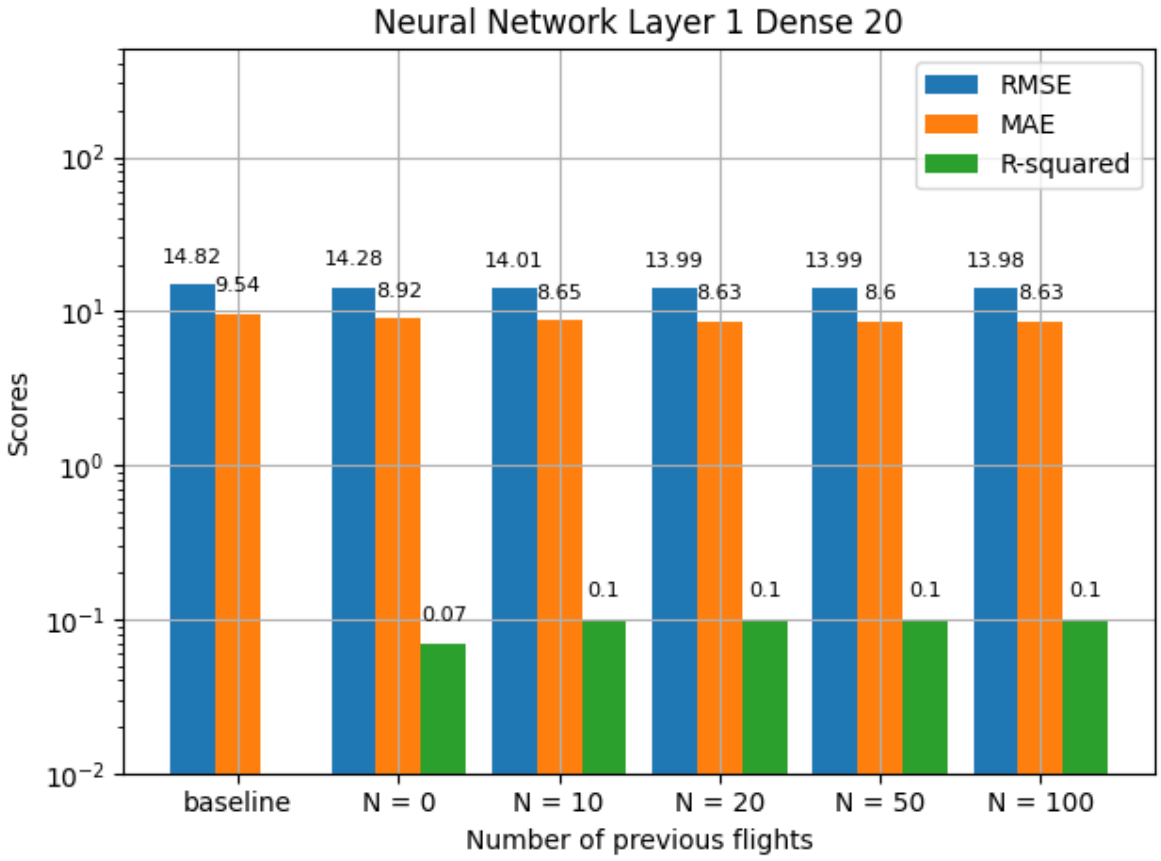

Figure 4.49: Departure delay prediction Neural Network layer=1, dense=20 results

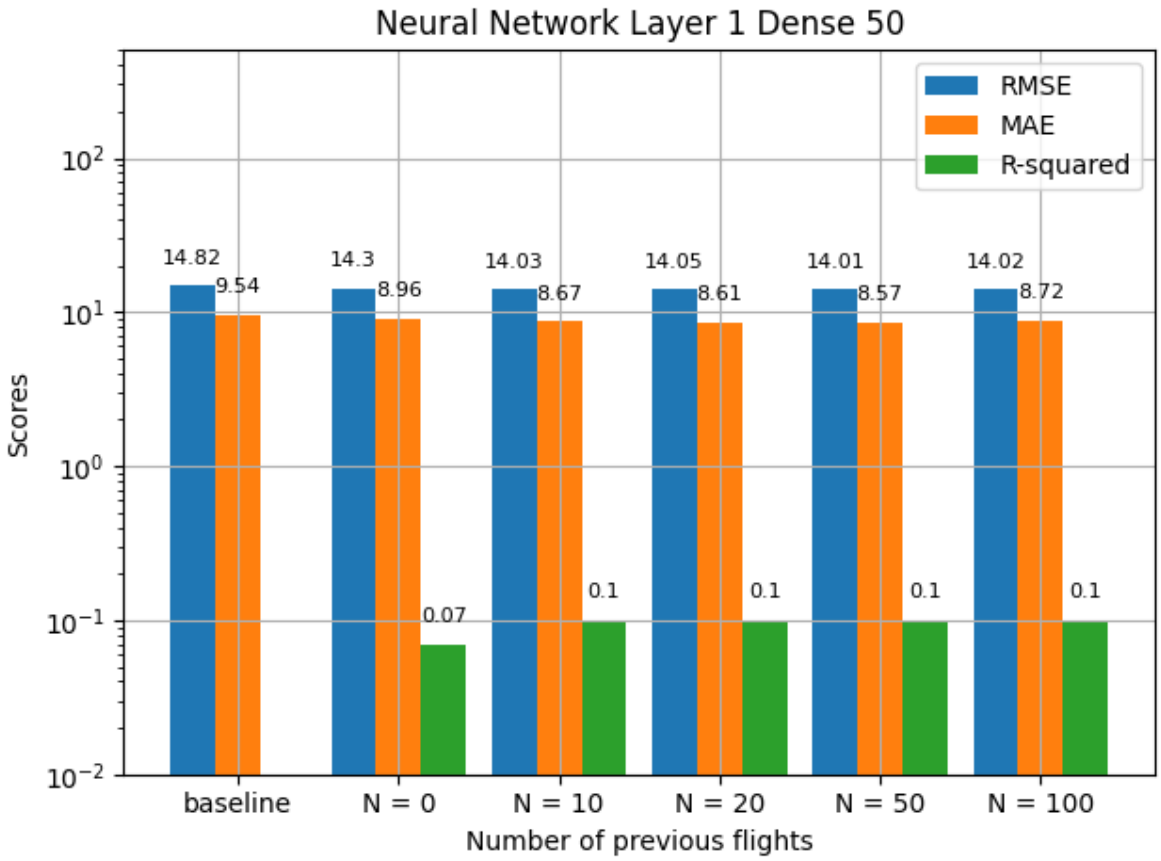

Figure 5.50: Departure delay prediction Neural Network layer=1, dense=50 results

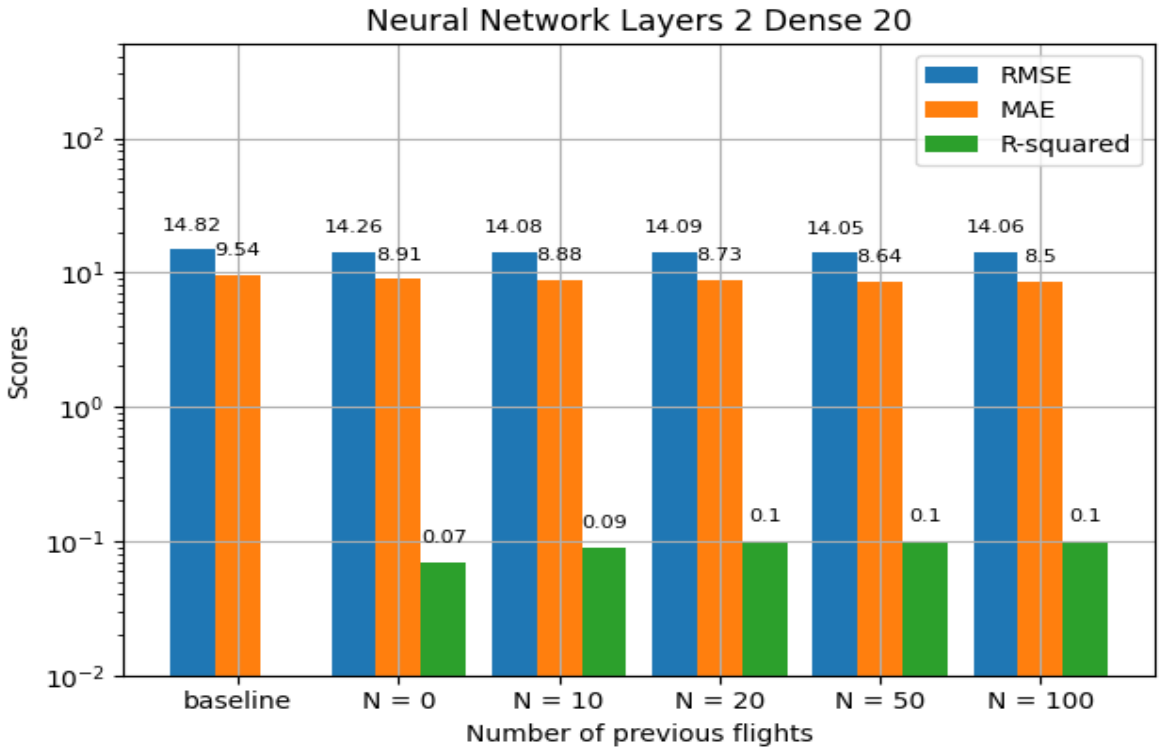

Figure 5.51: Departure delay prediction Neural Network layer=2, dense=20 results

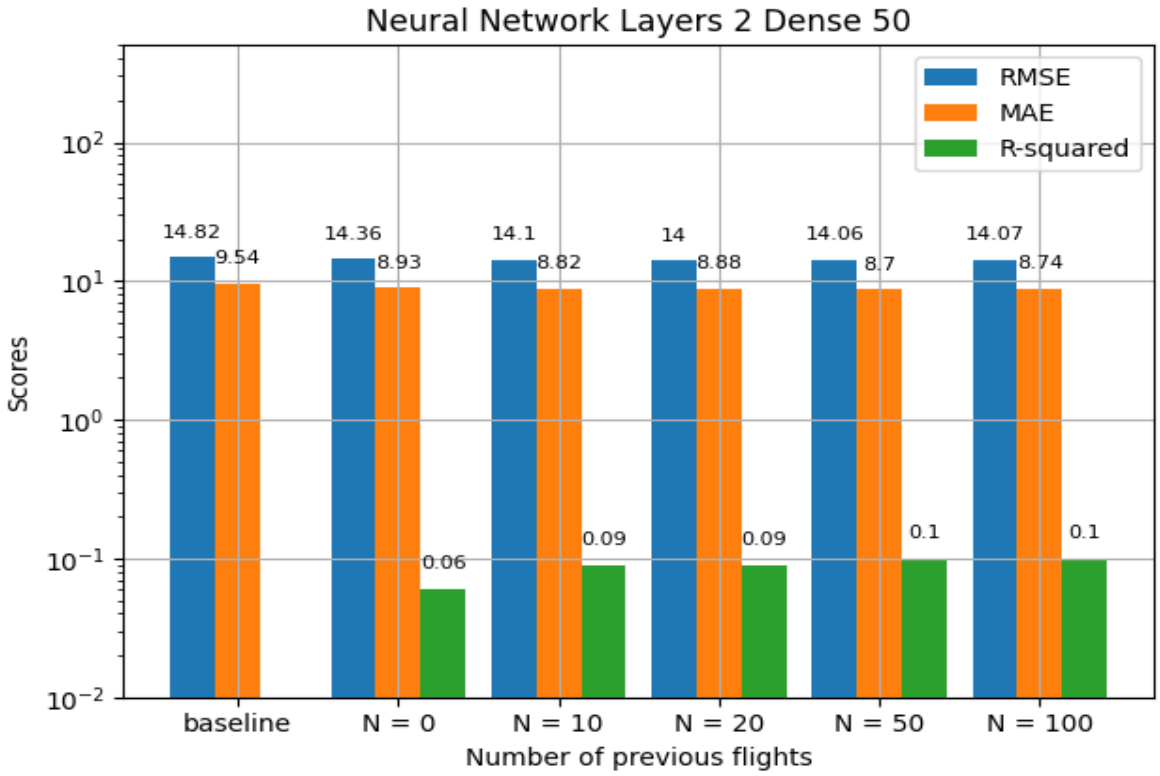

Figure 5.52: Departure delay prediction Neural Network layer=2, dense=50 results

5.1.5 Summary of Departure Delay Prediction from JFK to LAX

The following Table 5.5 presents the best obtained outcomes for Departure Delay Prediction of each algorithm based on the metrics and time measurements (where n=number of previous flights, l=number of hidden layers of Neural Network, d=number of dense):

Algorithms	RMSE	MAE	R-Squared	Time (seconds)
Linear Regression (n=10)	13.98±0,19	8.65±0,18	0.11±0.01	7.76
Polynomial Regression 3rd (n=50)	13.87±0.30	8.51±0.11	0.13±0.01	24.06
SVR (n=10)	14.84±0.68	7.52±0.30	-0.01±0.01	65,73
Neural Networks (n=50, l=1, d=20)	13.99±0.57	8.6±0.22	0,1±0.01	100,71
Baseline (mean)	14.82±0.58	9.54±0.29	0	null

Table 5.5: Summary algorithm comparison Departure Delay Prediction

Upon analyzing the aforementioned results, we note a minimal baseline inaccuracy of approximately 15 minutes in relation to the predicted departure delay. The improvement in our prediction algorithms is quite modest, with a 6.42% decrease in the RMSE metric, a 10.45% decrease in the MAE metric, and a 0.13 increase in the R-Squared metric. The

significance of previous flying knowledge is substantial, yet it may not apply to a vast number of prior trips. Futhermore, it is noticeable that the baseline (mean) has R-Squared value zero.

The 3rd degree Polynomial Regression approach, when provided with prior knowledge about the last 50 flight delays, yields superior results in terms of the RMSE and R-Squared metrics. Although the SVR method shows a 21.71% improvement in the MAE metric, it experiences a decrease in both RMSE and zero R-Squared relative to the baseline, rendering it unsuitable for consideration, as shown on Figure 4.48. Linear Regression demonstrates superior time performance, completing execution in 7.76 seconds, whereas Neural Network exhibits the poorest performance, requiring 100.71 seconds. Best results are shown as radar matrix plot in Figure 5.53.

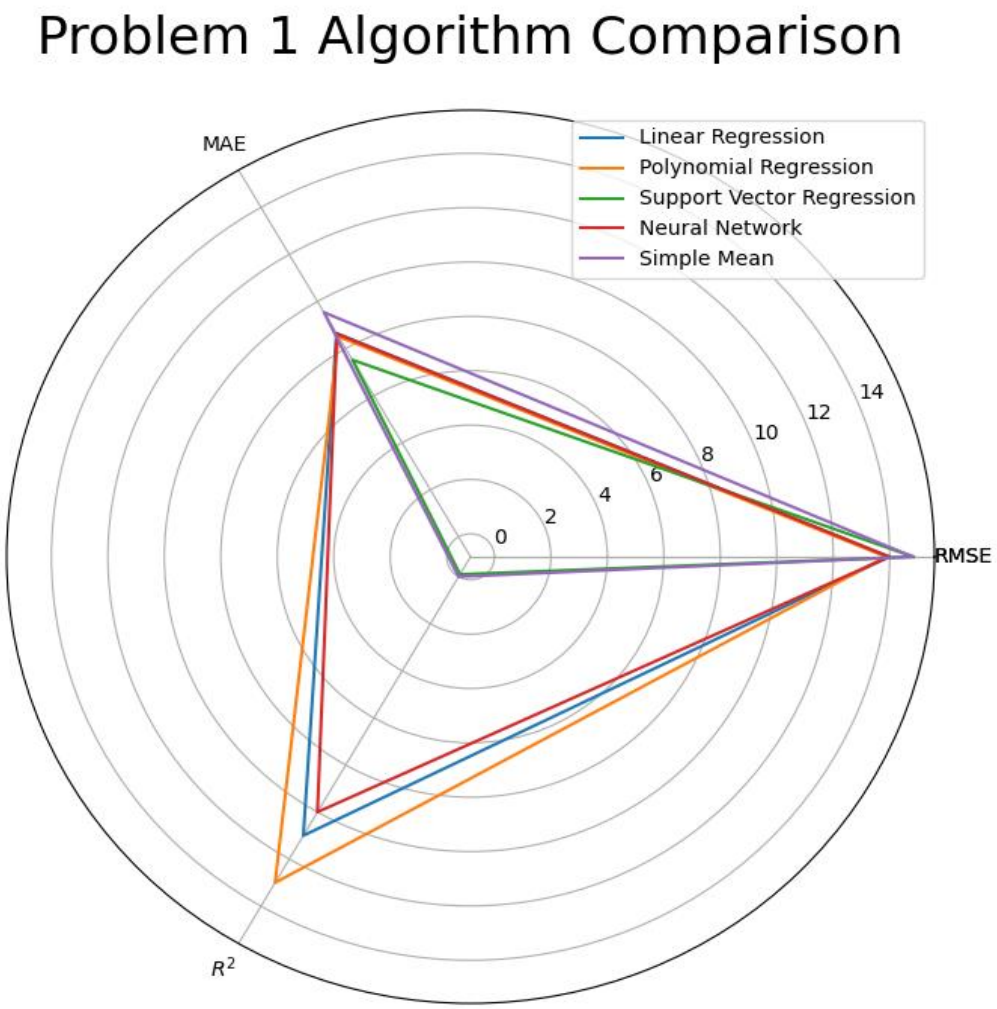

Figure 5.53: Departure delay prediction algorithm comparison radar matrix

5.2 Inflight Delay Prediction from JFK to LAX

This study aims to develop a regression model for predicting the arrival delay of Inflight planes traveling from John F. Kennedy International Airport (JFK) to Los Angeles International Airport (LAX). It should be noted that this particular model is designed to estimate the duration of in-flight delays, as opposed to the overall delay duration recorded in the dataset (Arrival_Delay). Additionally, it is worth mentioning that the model's predictions may also be influenced by departure operations, as this factor was taken into account during the initial approach.

The study utilized a newly acquired dataset named JFK TO LAX. To ensure data periodicity in relation to time, day, and week, the SCHEDULED_ARRIVAL attribute was transformed using periodic functions such as sine and cosine. This transformation facilitated the creation of a table that included a new attribute called ARRIVAL_DELAY, which solely accounted for in-flight delays and did not consider departure delays as in the previous approach. This table was then employed in conjunction with various Machine Learning techniques, namely Linear Regression, Polynomial Regression, and Feed Forward Neural Networks. Furthermore, this approach incorporates a novel property that involves the computation of the average delay, namely the INFLIGHT_DELAY, experienced by the most recent 20 flights originating from any airport in the United States and arriving at LAX airport. A new dataset, named ALL TO LAX, was generated for the aforementioned implementation, containing a total of 192,136 flight entries.

5.2.1 Linear Regression

The process of developing a prediction model using the Linear Regression technique in Python involve following steps as section 5.1.1:

The results are presented in Figure 5.54:

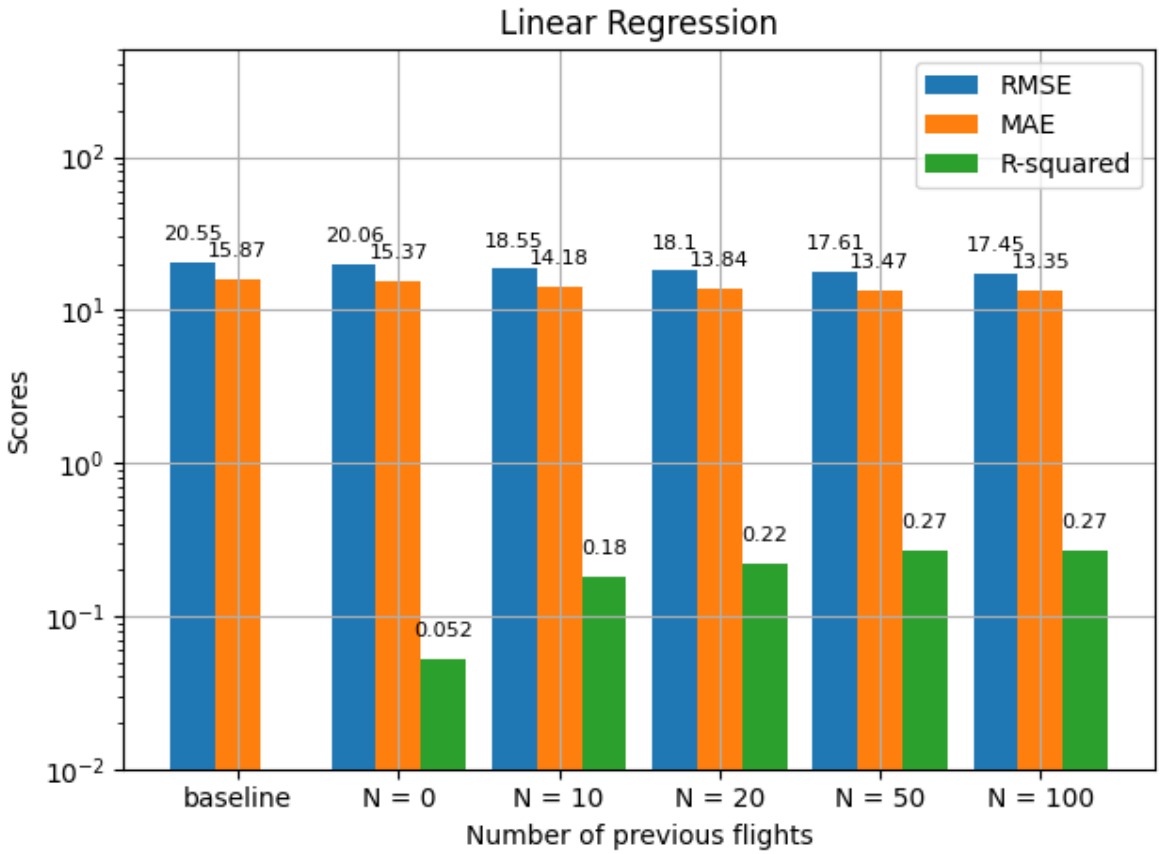

Figure 5.54: Inflight delay prediction Linear Regression results

5.2.2 Polynomial Regression

The process of developing a prediction model using the Polynomial Regression technique in Python involve following steps as section 5.1.2:

The results are presented in Figures 5.55, 5.56:

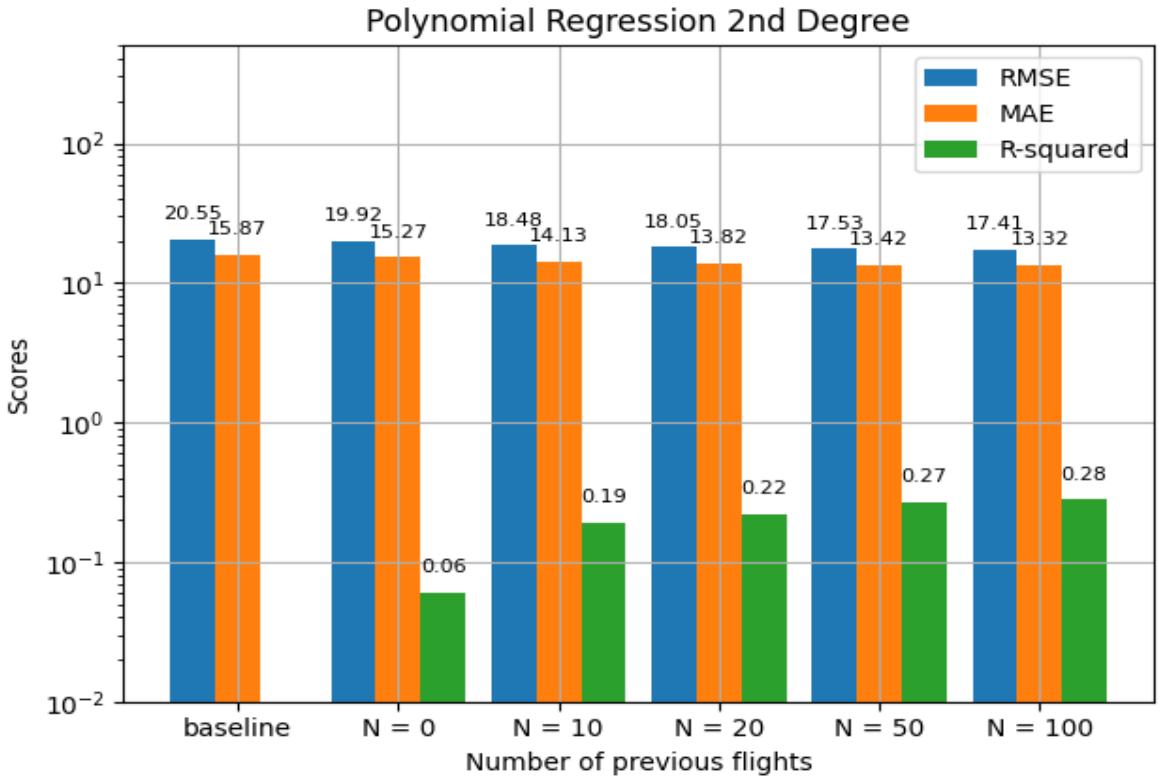

Figure 5.55: Inflight delay prediction Polynomial Regression 2nd degree results

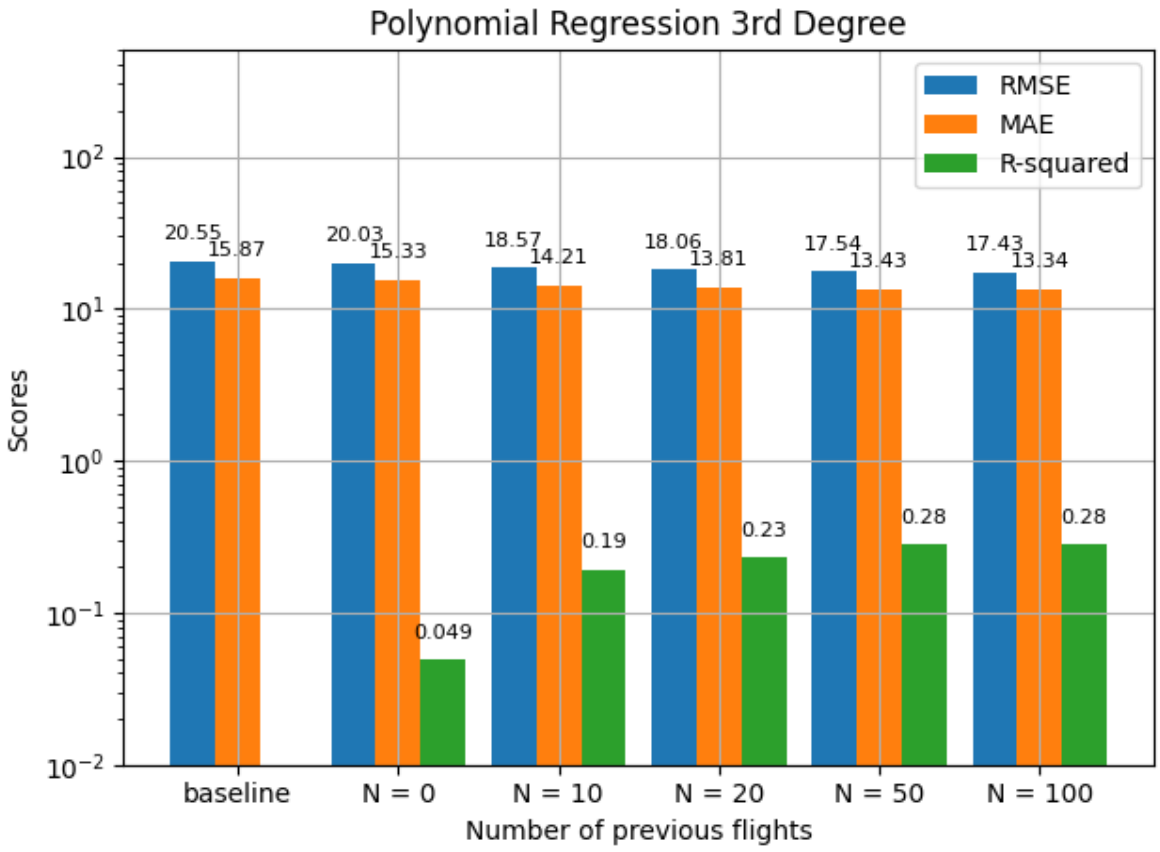

Figure 5.56: Inflight delay prediction Polynomial Regression 3rd degree results

5.2.3 Support Vector Regression

The process of developing a prediction model using the Support Vector Regression technique in Python involve following steps as section 5.1.3:

The results are presented in Figure 5.57:

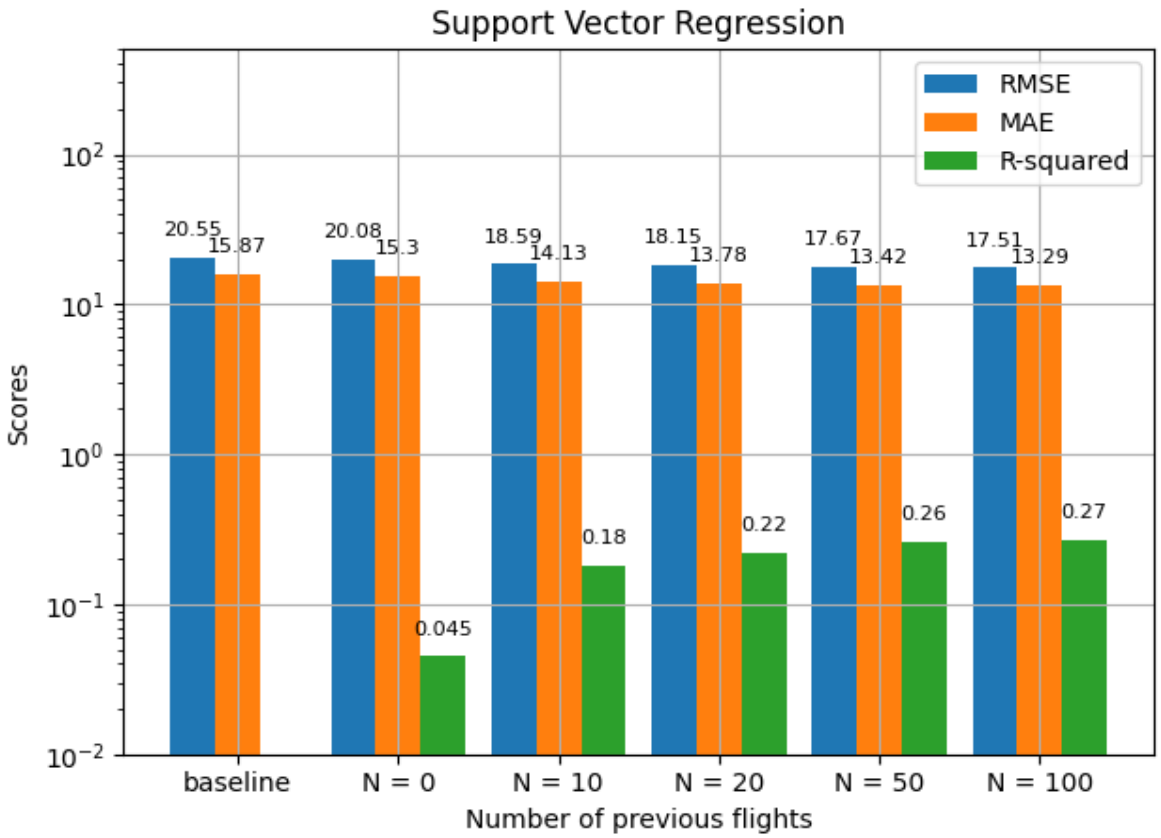

Figure 5.57: Inflight delay prediction SVR results

5.2.4 Feed Forward Neural Network

The process of developing a prediction model using the Support Vector Regression technique in Python involve following steps as section 5.1.4:

The results are presented in Figures 5.58, 5.59, 5.60, 5.61:

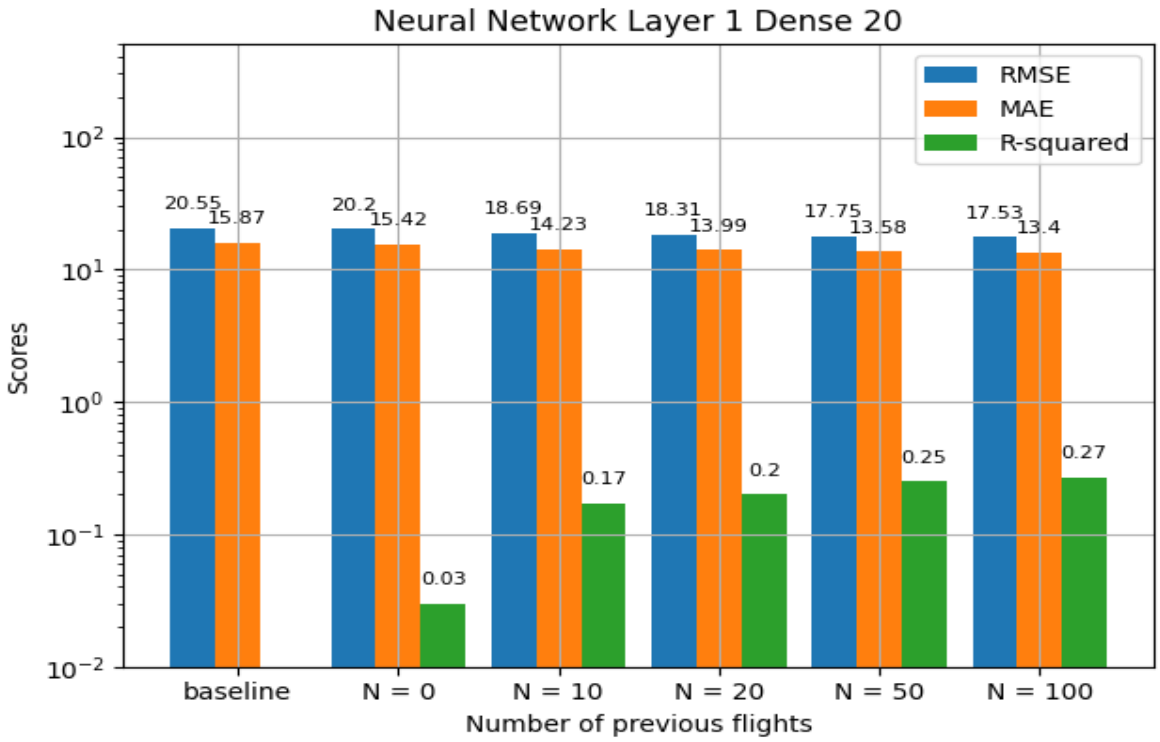

Figure 5.58: Inflight delay prediction Neural Network layer=1, dense=20 results

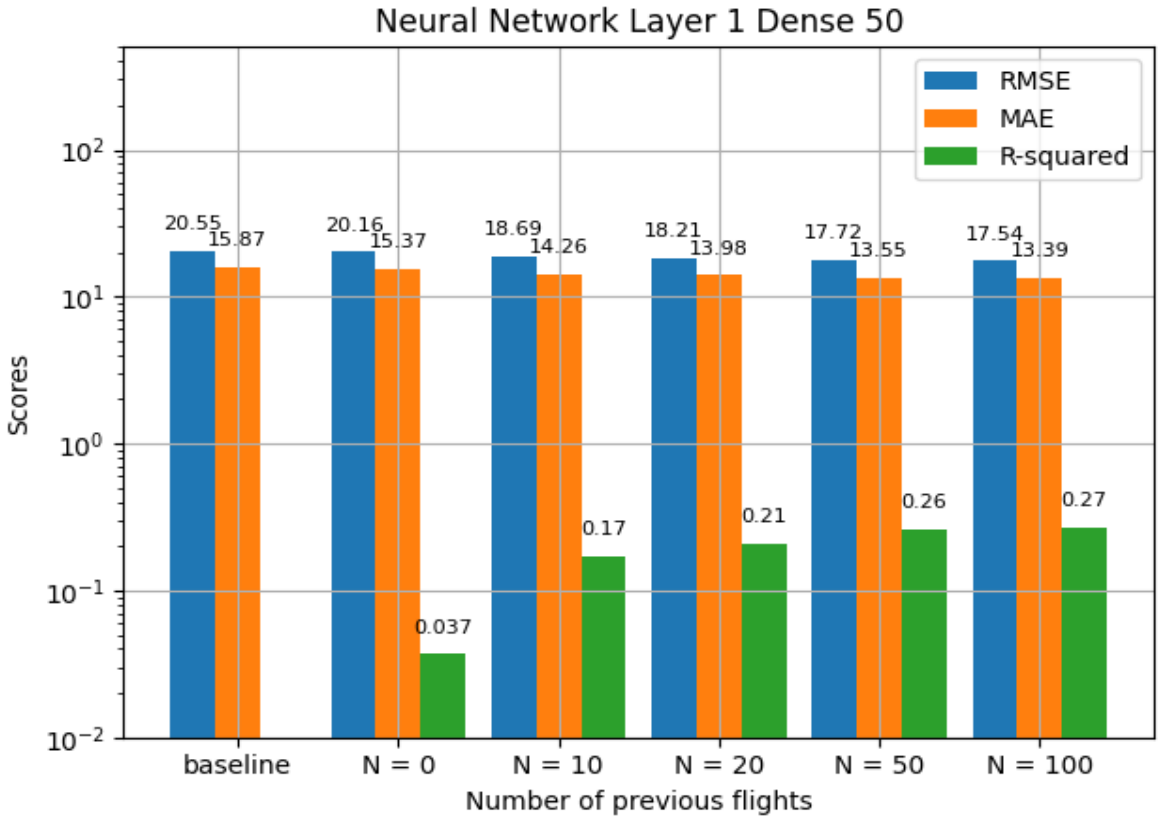

Figure 5.59: Inflight delay prediction Neural Network layer=1, dense=50 results

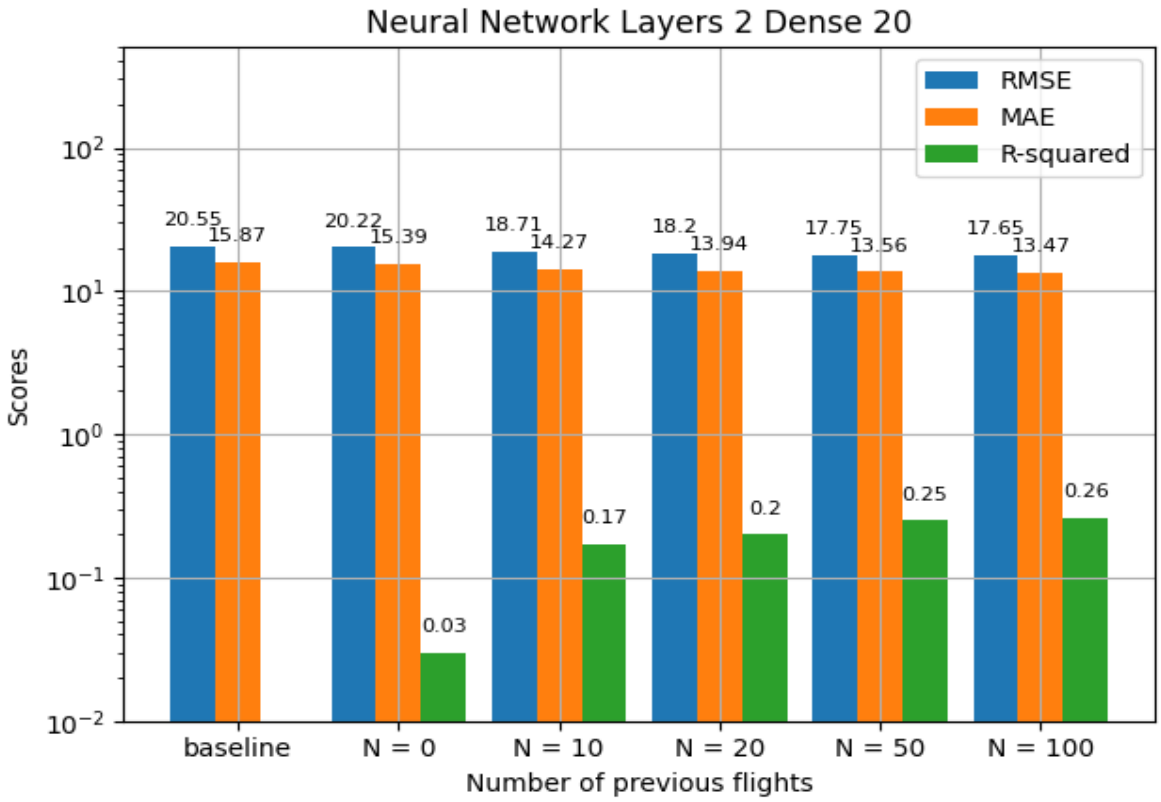

Figure 5.60: Inflight delay prediction Neural Network layer=2, dense=20 results

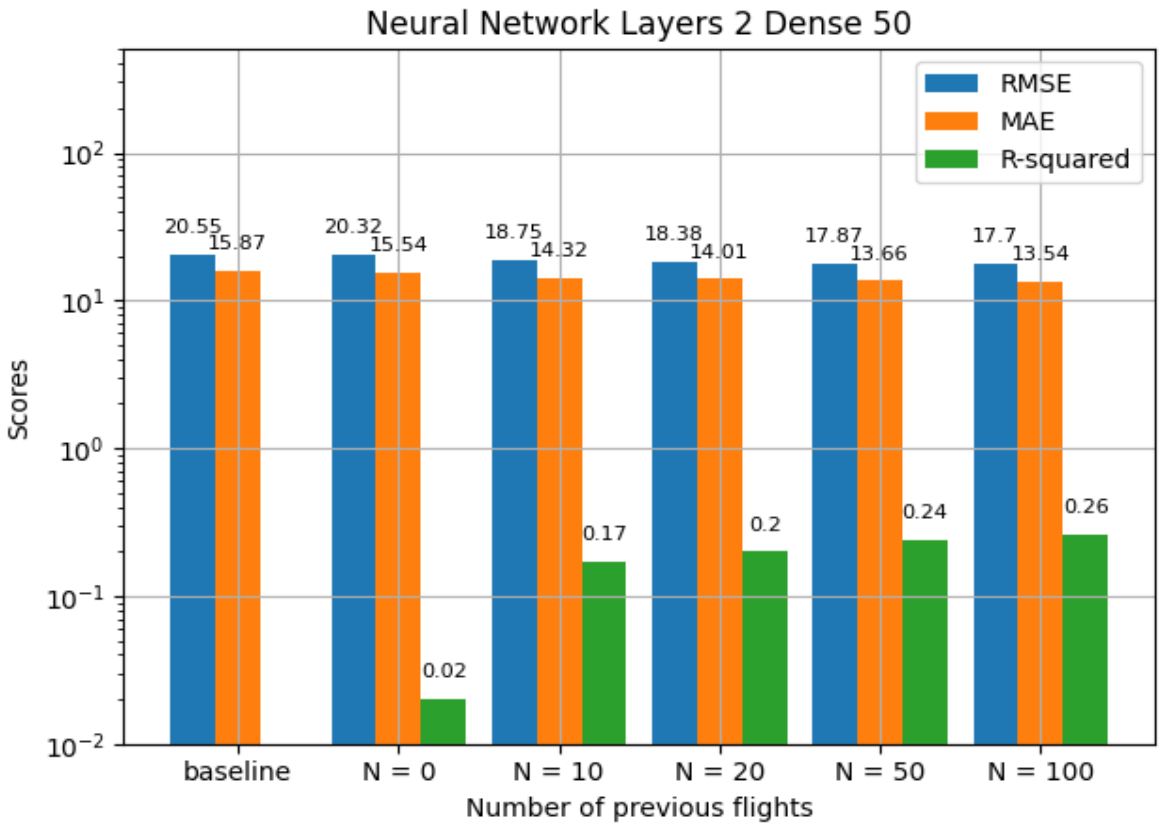

Figure 5.61: Inflight delay prediction Neural Network layer=2, dense=50 results

5.2.5 Summary of Inflight Delay Prediction from JFK to LAX

The following Table 5.6 presents the best obtained outcomes for Inflight Delay Prediction of each algorithm based on the metrics and time measurements (where n=number of previous flights, l=number of layers of Neural Network, d=number of dense):

Algorithms	RMSE	MAE	R-Squared	Time (seconds)
Linear Regression (n=100)	17.45±0.47	13.35±0.30	0.27±0.01	7.97
Polynomial Regression 2nd (n=100)	17.41±0.10	13.32±0.06	0.28±0.02	8.75
SVR (n=100)	17.51±0.14	13.29±0.16	0.27±0.02	66,39
Neural Networks (n=100, l=1, d=20)	17.53±0.48	13.4±0.33	0.27±0.01	105,09
Baseline (mean)	20.55±0.46	15.87±0.30	0	null

Table 5.6: Summary algorithm comparison Inflight Delay Prediction

From the above statistics, it is evident that predicting inflight delay is more challenging compared to departure delay, as indicated by a bigger baseline error of. The algorithms' forecasts demonstrate a significant enhancement in the RMSE measure by 15.27%, a 16.01% improvement in the MAE metric, and a 0.28 increase in the R-Squared value. The knowledge gained from past flights is likewise highly crucial for a multitude of subsequent missions. All algorithms provide comparable outcomes with minor variations of 0.1%. Futhermore, it is noticeable that the baseline (mean) has R-Squared value zero.

The 2nd degree Polynomial Regression prediction, when provided with prior knowledge of the last 100 aircraft delays, yields improved performances in the metrics RMSE, MAE, and R-Squared. When considering execution time, Linear Regression demonstrates the most efficient performance, taking only 7.97 seconds. On the other hand, Neural Network exhibits the poorest performance, requiring 105.09 seconds. Best results are shown as radar matrix plot in Figure 5.62.

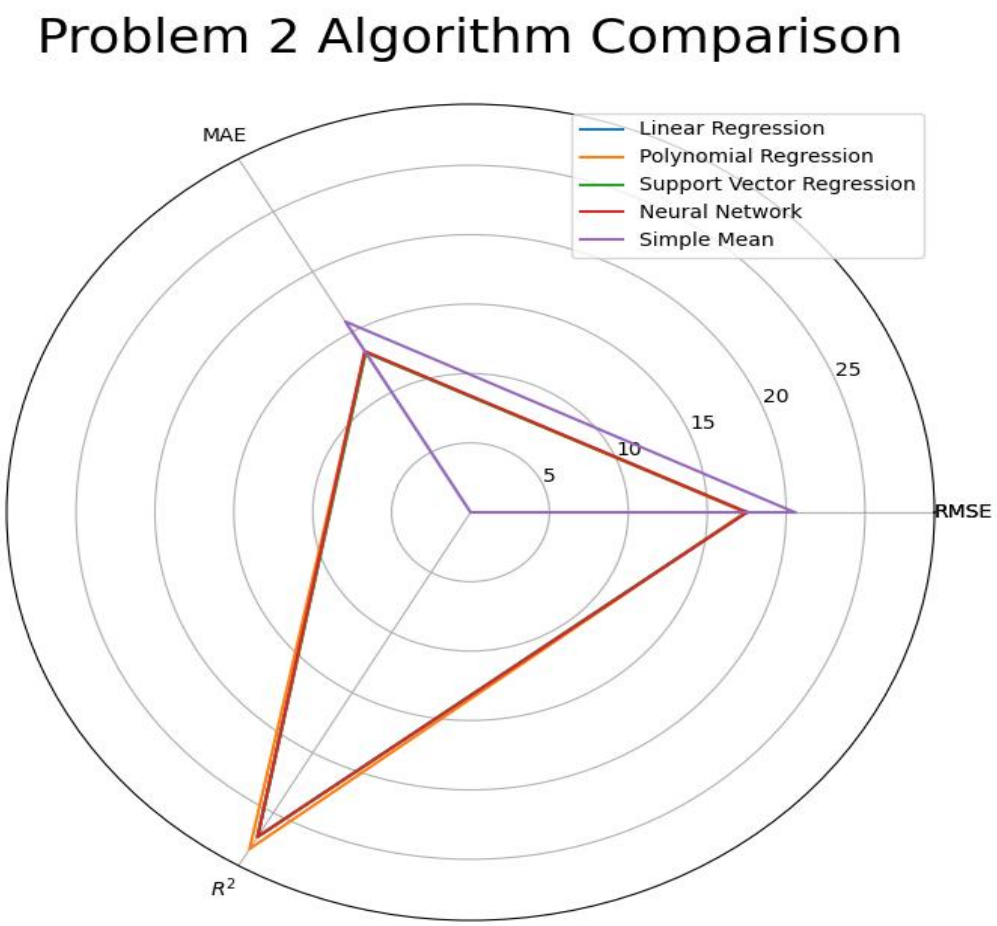

Figure 5.62: Inflight delay prediction algorithm comparison radar matrix

5.3 Total Delay Prediction sum of models

The estimation of overall delay the overall delay of a flight can be determined by adding the calculated values for departure delay and inflight delay, as obtained from the first and second problems, respectively.

5.3.1 Linear Regression

The process of developing a prediction model using the Linear Regression technique in Python involves following steps as section 5.1.1:

The results are presented in Figure 5.63:

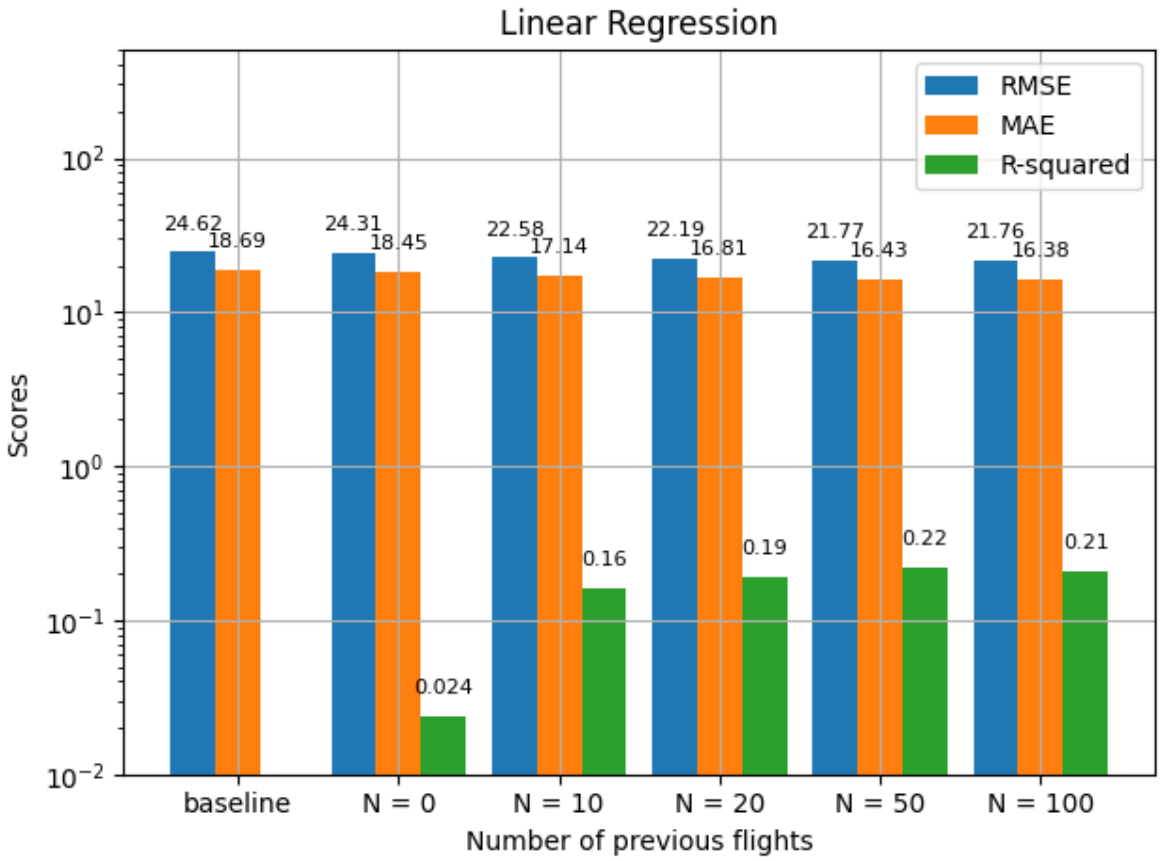

Figure 5.63: Total delay prediction sum of models Linear Regression results

5.3.2 Polynomial Regression

The process of developing a prediction model using the Polynomial Regression technique in Python involves following steps as section 5.1.2:

The results are presented in Figures 5.64, 5.65:

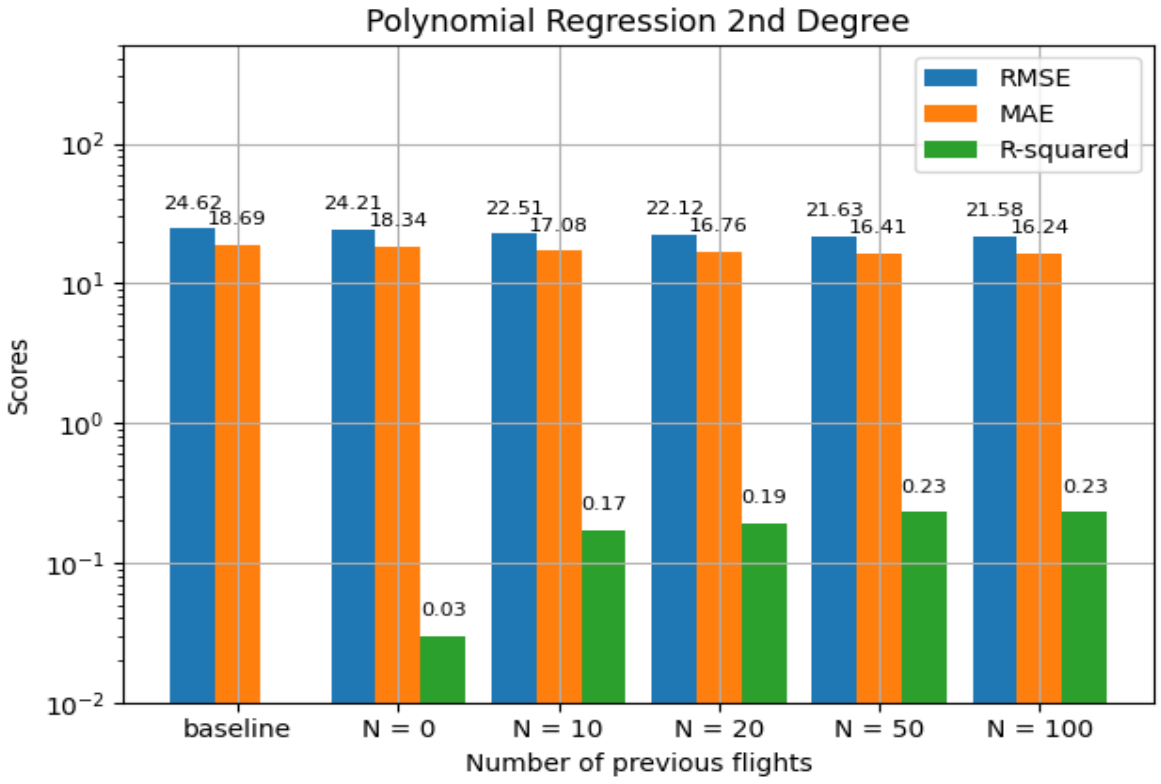

Figure 5.64: Total delay prediction sum of models Polynomial Regression 2nd degree results

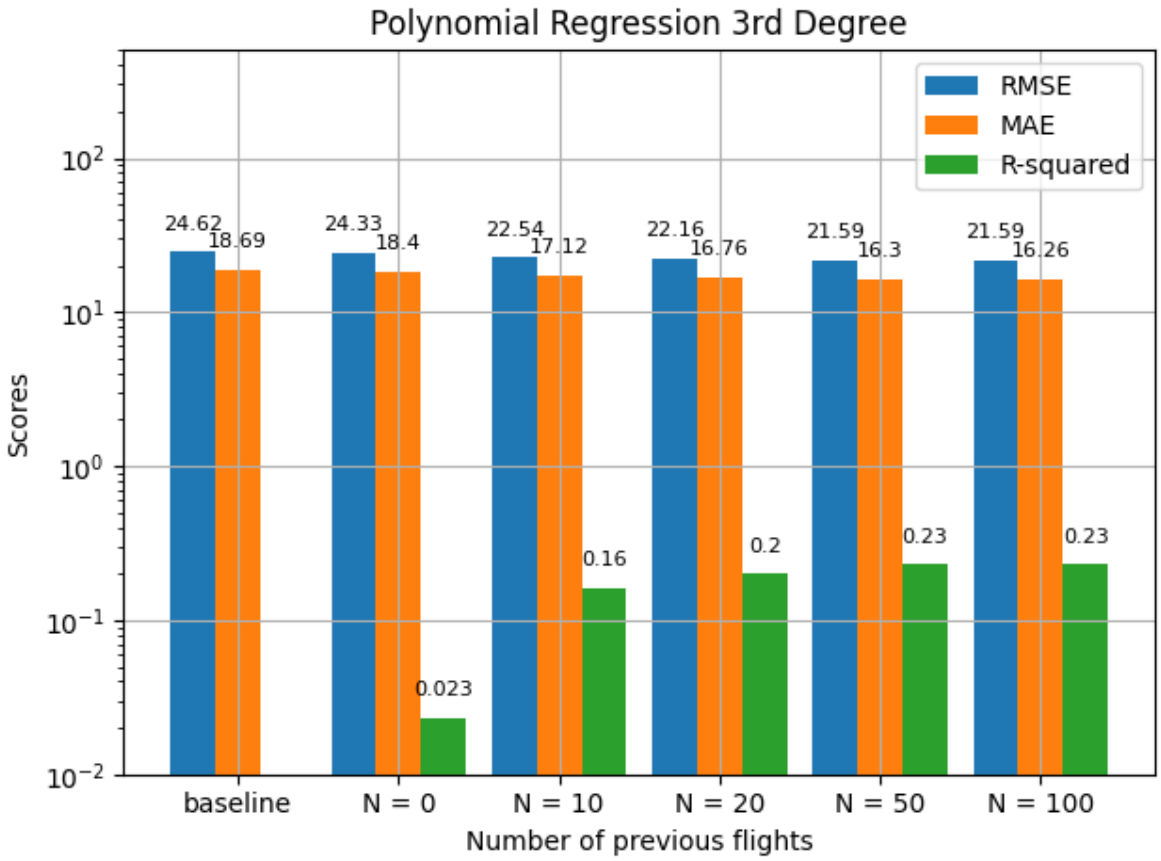

Figure 5.65: Total delay prediction sum of models Polynomial Regression 3rd degree results

5.3.3 Support Vector Regression

The process of developing a prediction model using the Support Vector Regression technique in Python involves following steps as section 5.1.3:

The results are presented in Figure 5.66:

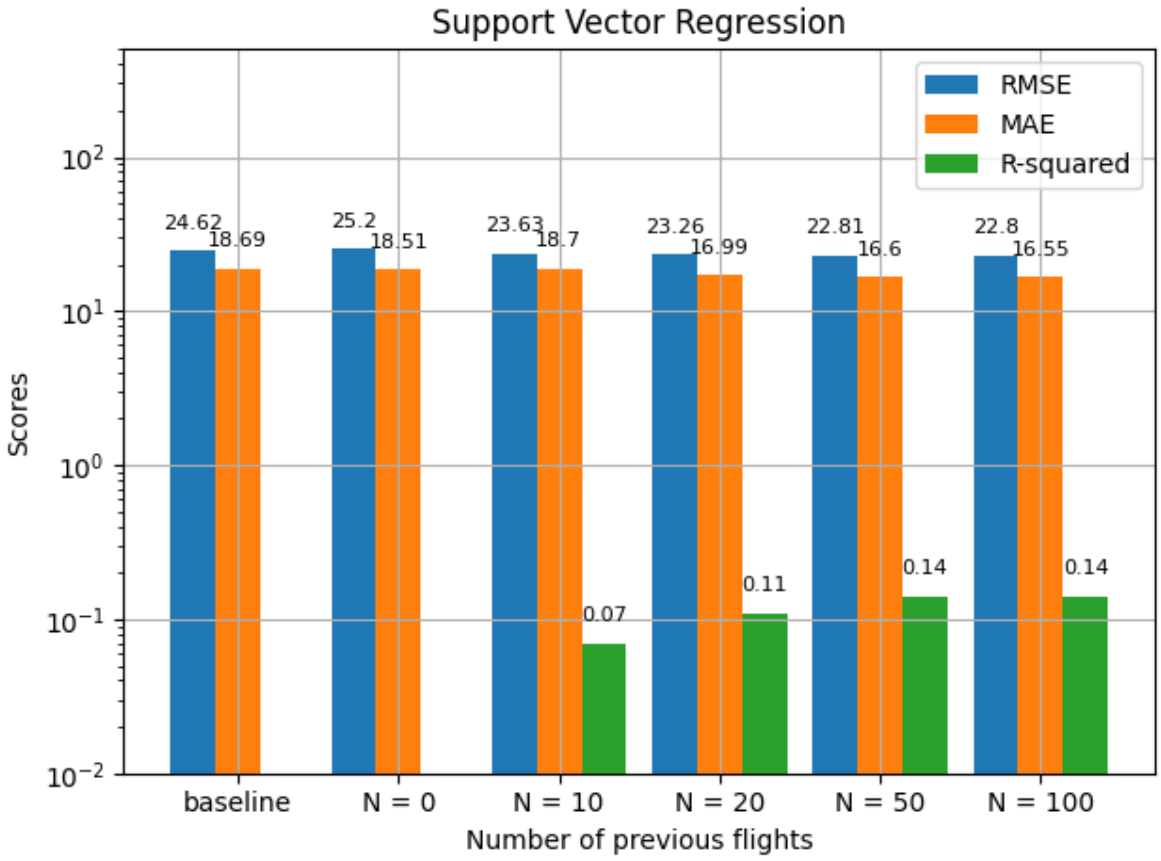

Figure 5.66: Total delay prediction sum of models SVR results

5.3.4 Feed Forward Neural Network

The process of developing a prediction model using the Support Vector Regression technique in Python involves following steps as section 5.1.4:

The results are presented in Figures 5.67, 5.68, 5.69, 5.70:

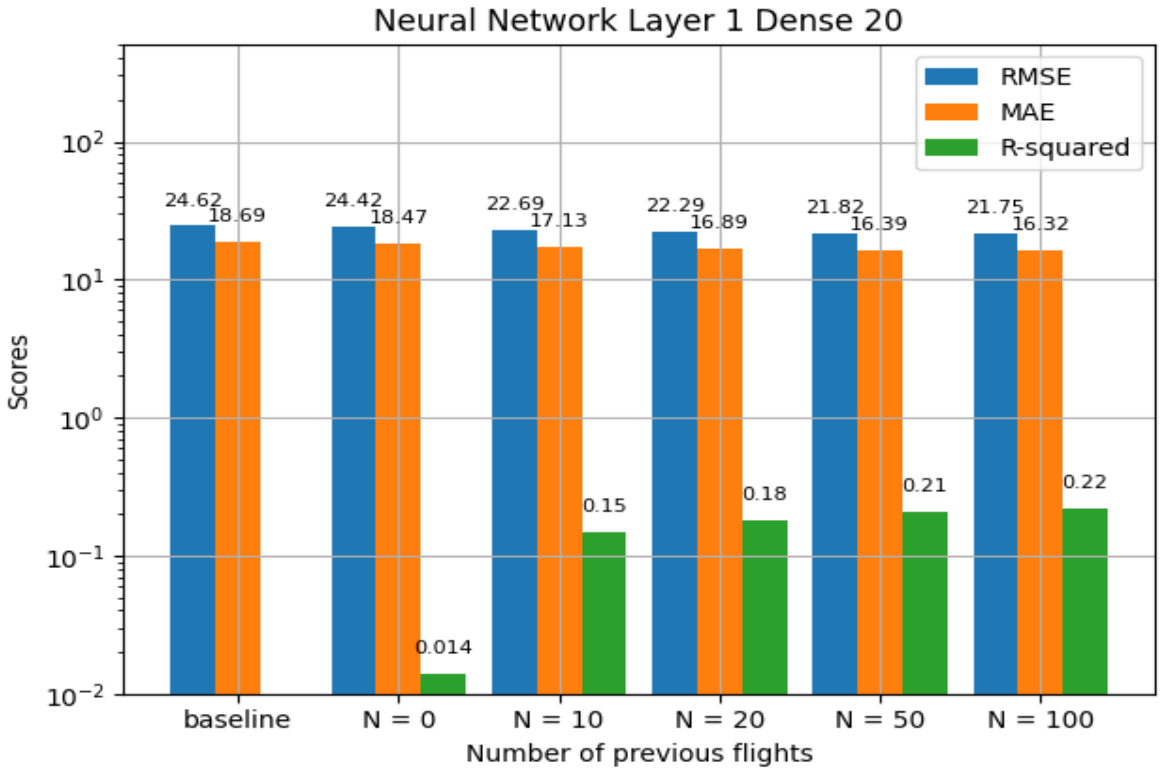

Figure 5.67: Total delay prediction sum of models Neural Network layer=1, dense=20 results

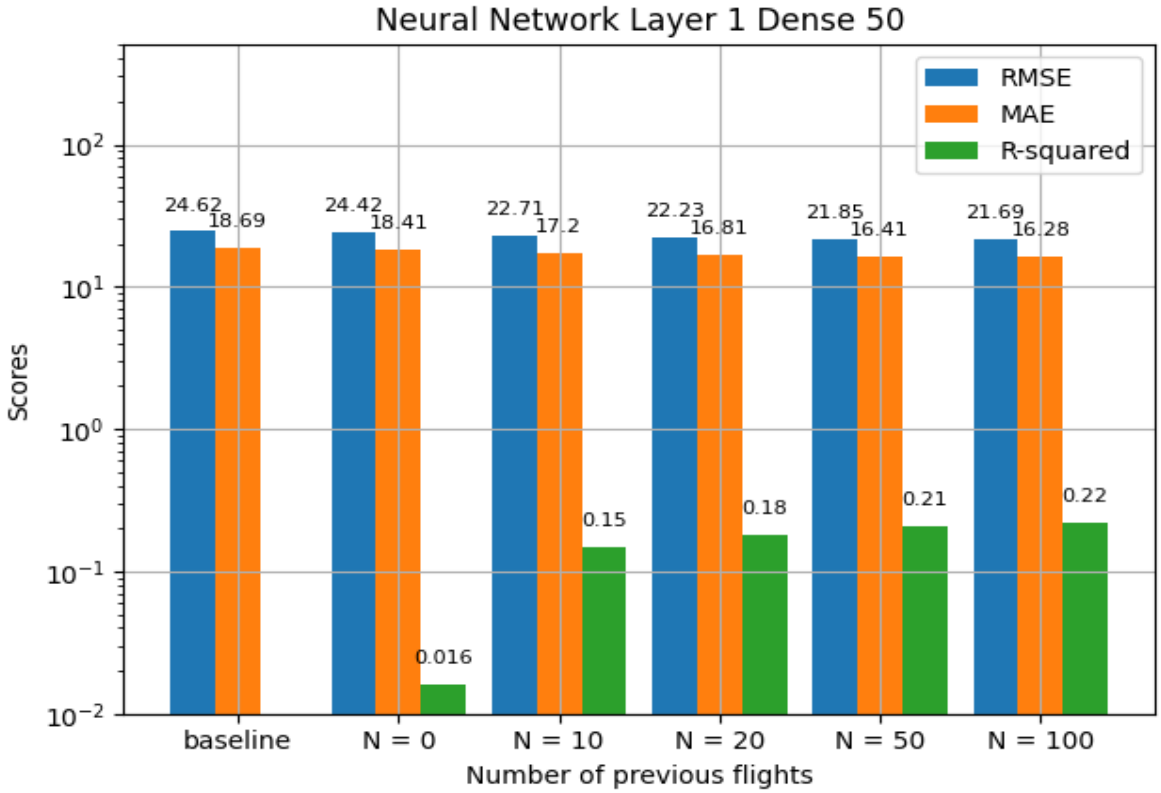

Figure 5.68: Total delay prediction sum of models Neural Network layer=1, dense=50 results

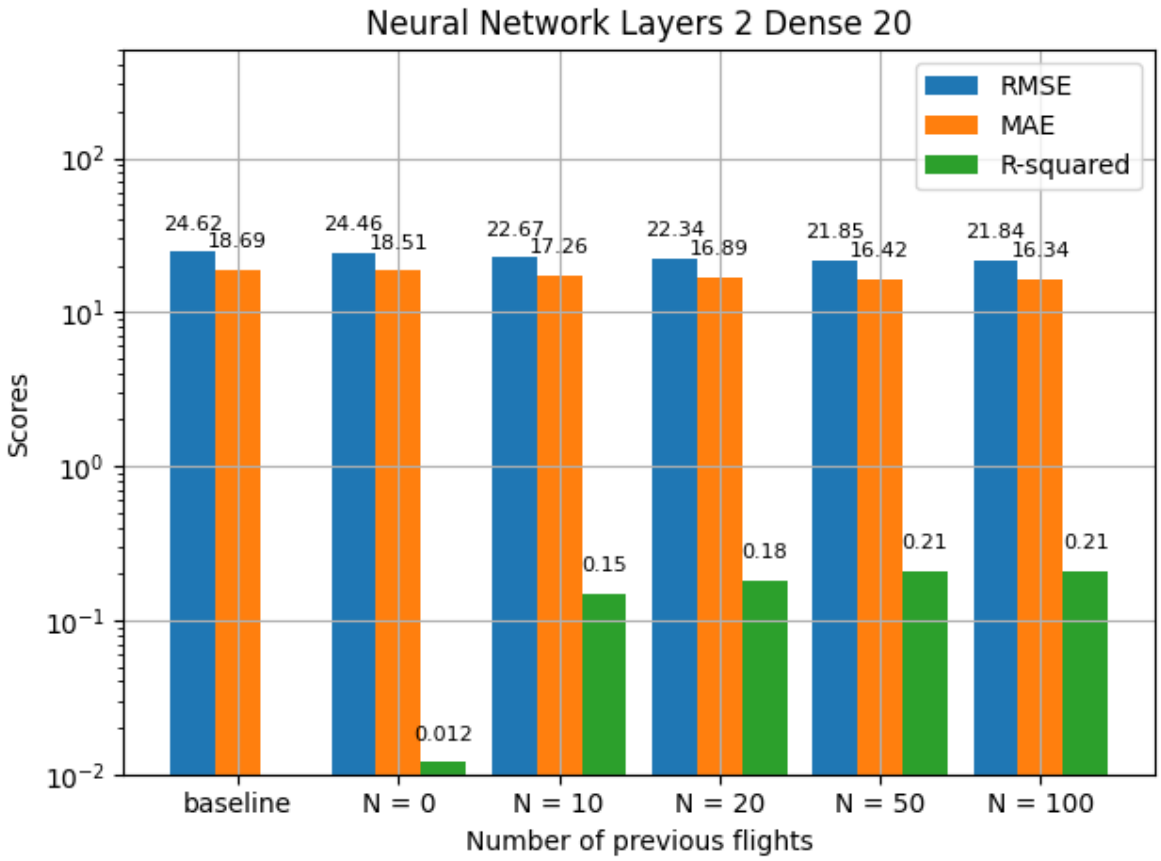

Figure 5.69: Total delay prediction single data model Neural Network layer=2, dense=50 results

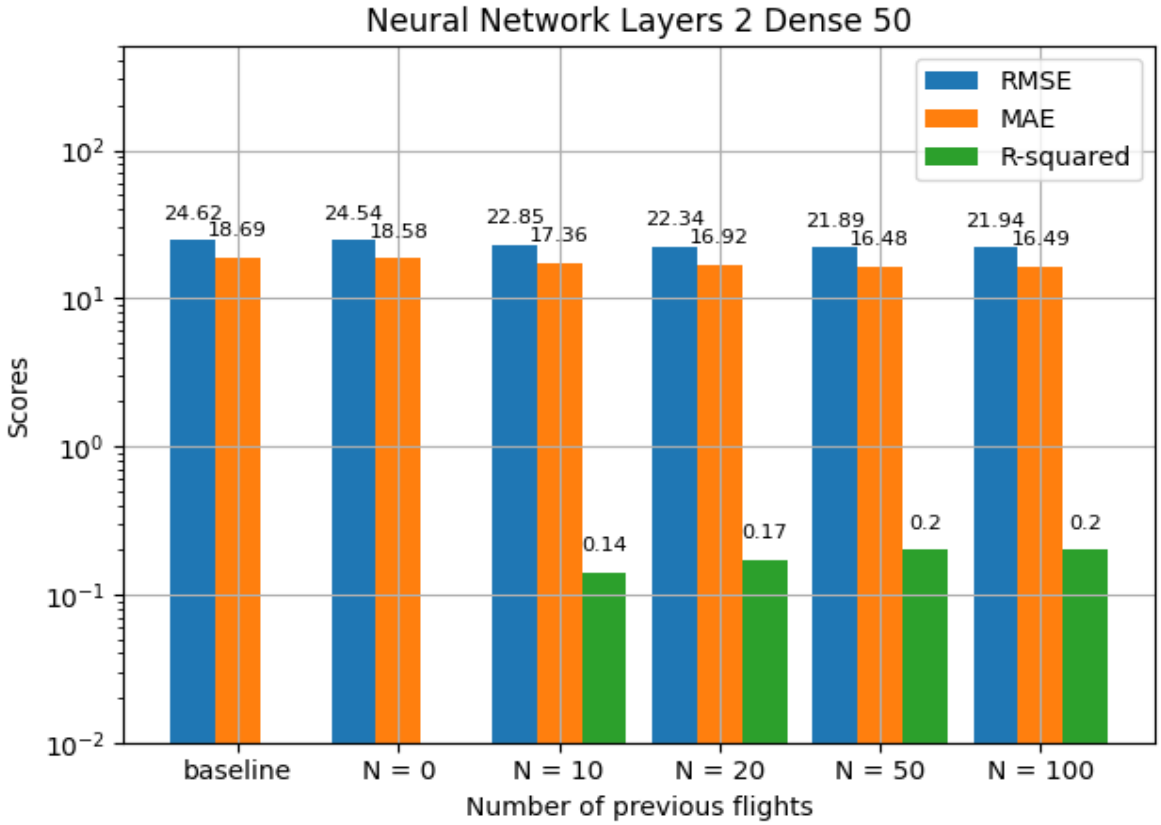

Figure 5.70: Total delay prediction sum of models Neural Network layer=2, dense=50 results

5.3.5 Summary of Total Delay Prediction sum of models

The following Table 5.7 presents the best obtained outcomes for Total Delay Prediction single data model of each algorithm based on the metrics and time measurements (where n=number of previous flights, l=number of layers of Neural Network, d=number of dense):

Algorithms	RMSE	MAE	R-Squared	Time (seconds)
Linear Regression (n=100)	21.76±0.73	16.38±0.49	0.21±0.02	7.99
Polynomial Regression 3rd (n=100)	21.58±0.33	16.24±0.25	0.23±0.01	29.63
SVR (n=100)	22.58±0.28	16,55±0.23	0.13±0.02	66.39
Neural Networks (n=100, l=1, d=50)	21.69±0.27	16.28±0.11	0.22±0.02	107.1
Baseline (mean)	24.62±0.18	18.69±0.22	0	null

Table 5.7: Summary algorithm comparison Total Delay Prediction sums of models

In the results mentioned above, we observe a significant baseline error of 24.62 in the cumulative delay prediction sum of models, which is, nevertheless, smaller than the combined sum of the individual errors. The predictions generated by our algorithms demonstrate a significant enhancement in the RMSE metric by 12.34%, a 13.1% improvement in the MAE metric, and a 0.23 R-Squared value. The knowledge gained from past flights is likewise highly crucial for a multitude of subsequent missions. All algorithms provide comparable outcomes with minor variations of 0.2%. Futhermore, it is noticeable that the baseline (mean) has R-Squared value zero.

The 3rd degree Polynomial Regression algorithm, when provided with prior knowledge of the last 100 flight delays, yields superior results in the metrics RMSE, MAE, and R-Squared. Linear Regression demonstrates superior time performance, completing execution in 7.99 seconds, while Neural Network exhibits the poorest performance, requiring 107.1 seconds. Best results are shown as radar matrix plot in Figure 5.71.

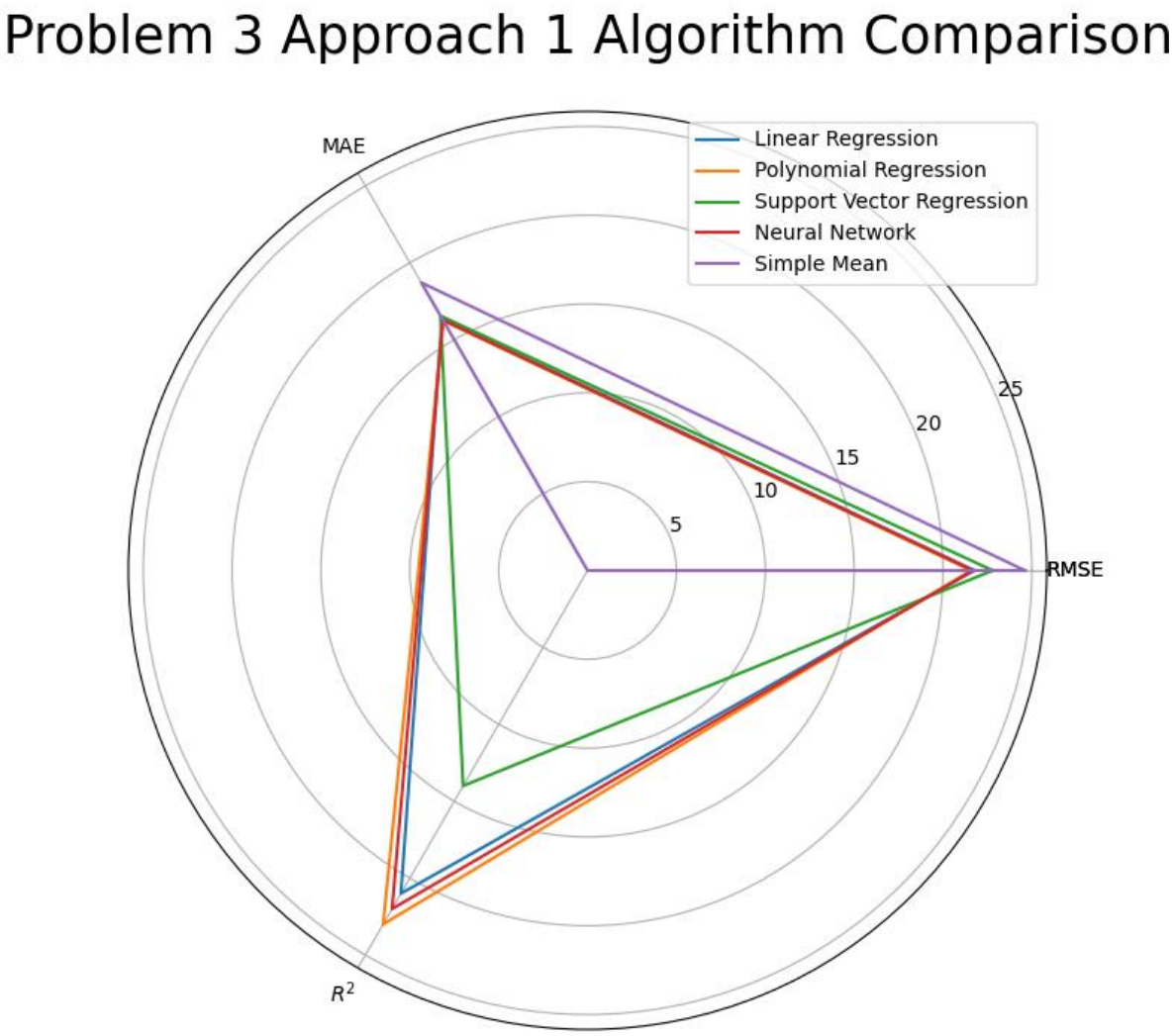

Figure 5.71: Total delay prediction sum of models algorithm comparison radar matrix

5.4 Total Delay Prediction single data model

The objective is to estimate the cumulative delay of a flight using a unified data model that incorporates the qualities from two distinct problems: Departure Delay and Inflight Delay. This approach aims to reevaluate the average overall delay by combining the data from both problems into a single model.

5.4.1 Linear Regression

The process of developing a prediction model using the Linear Regression technique in Python following steps as section 5.1.1

The results are presented in Figure 5.72:

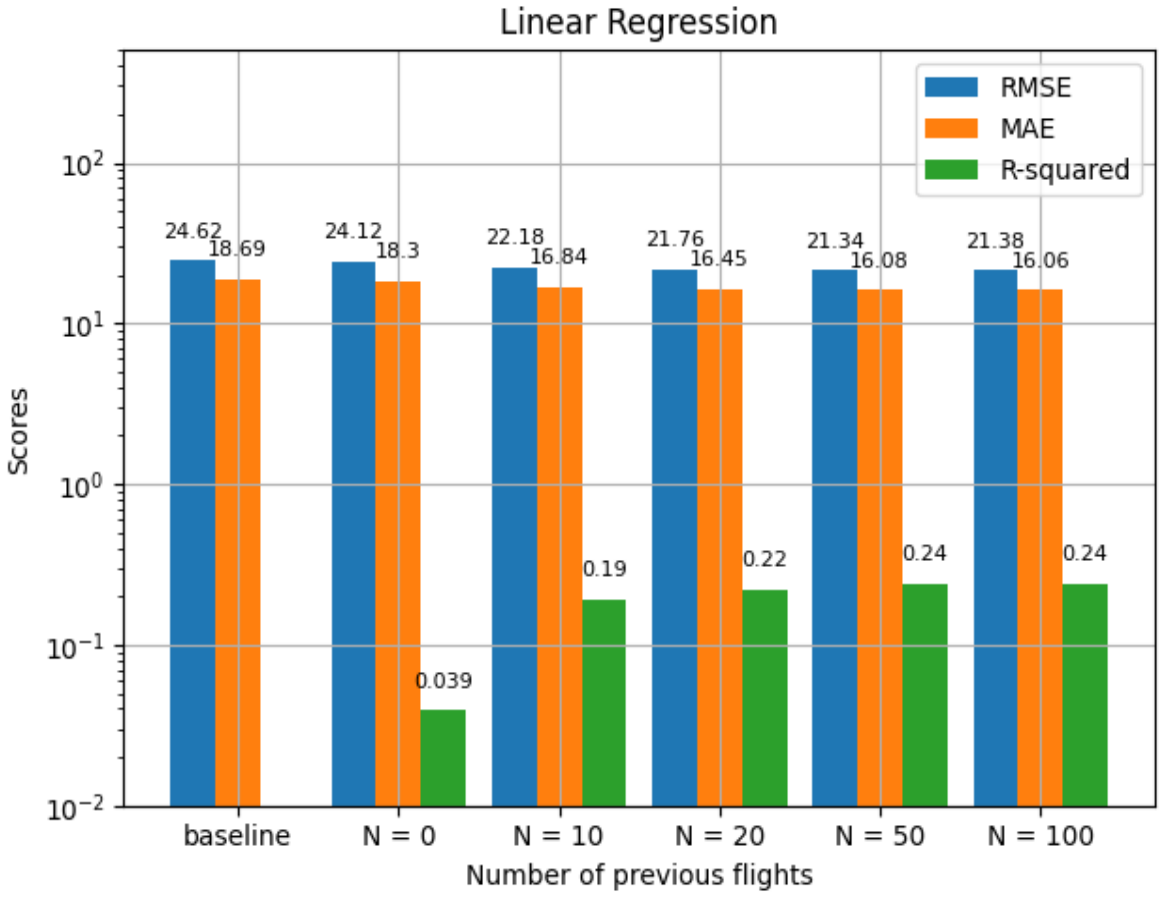

Figure 5.72: Total delay prediction single data model Linear Regression results

5.4.2 Polynomial Regression

The process of developing a prediction model using the Polynomial Regression technique in Python following steps as section 5.1.2:

The results are presented in Figures 5.73, 5.74:

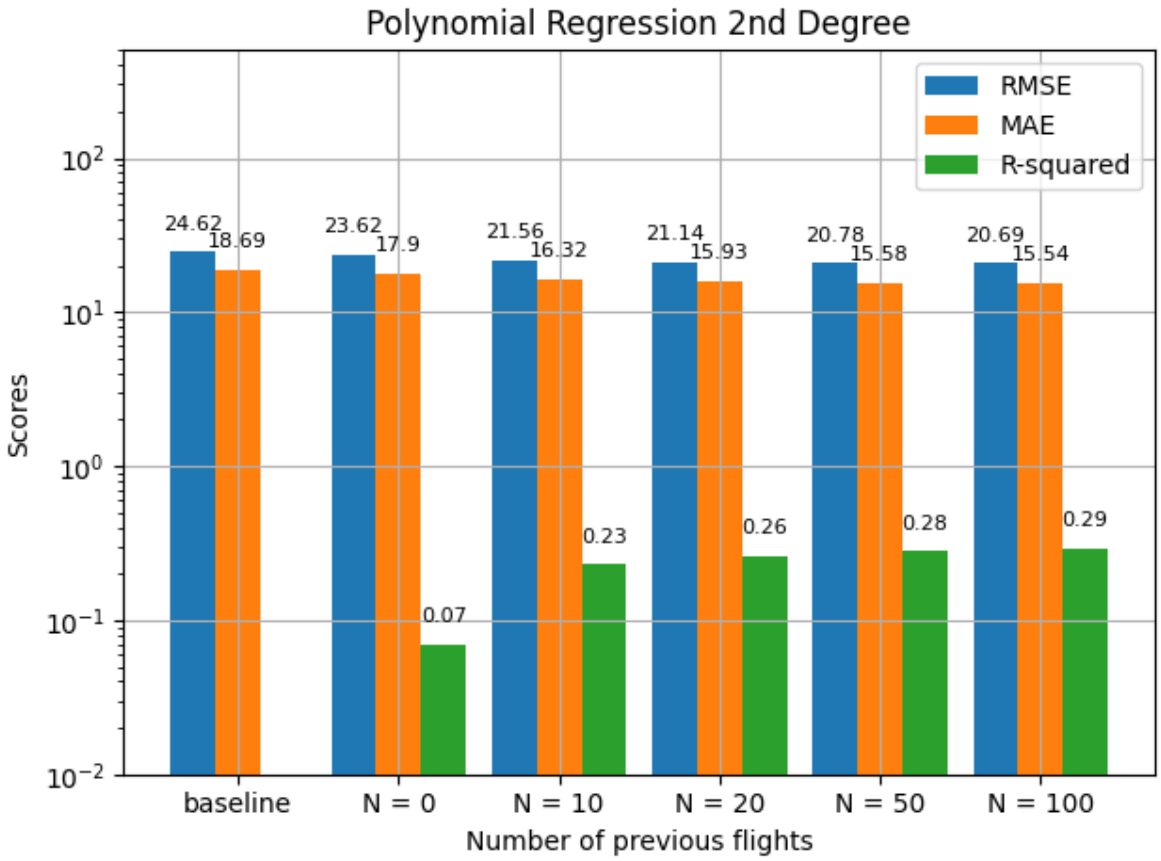

Figure 5.73: Total delay prediction single data model Polynomial Regression 2nd degree results

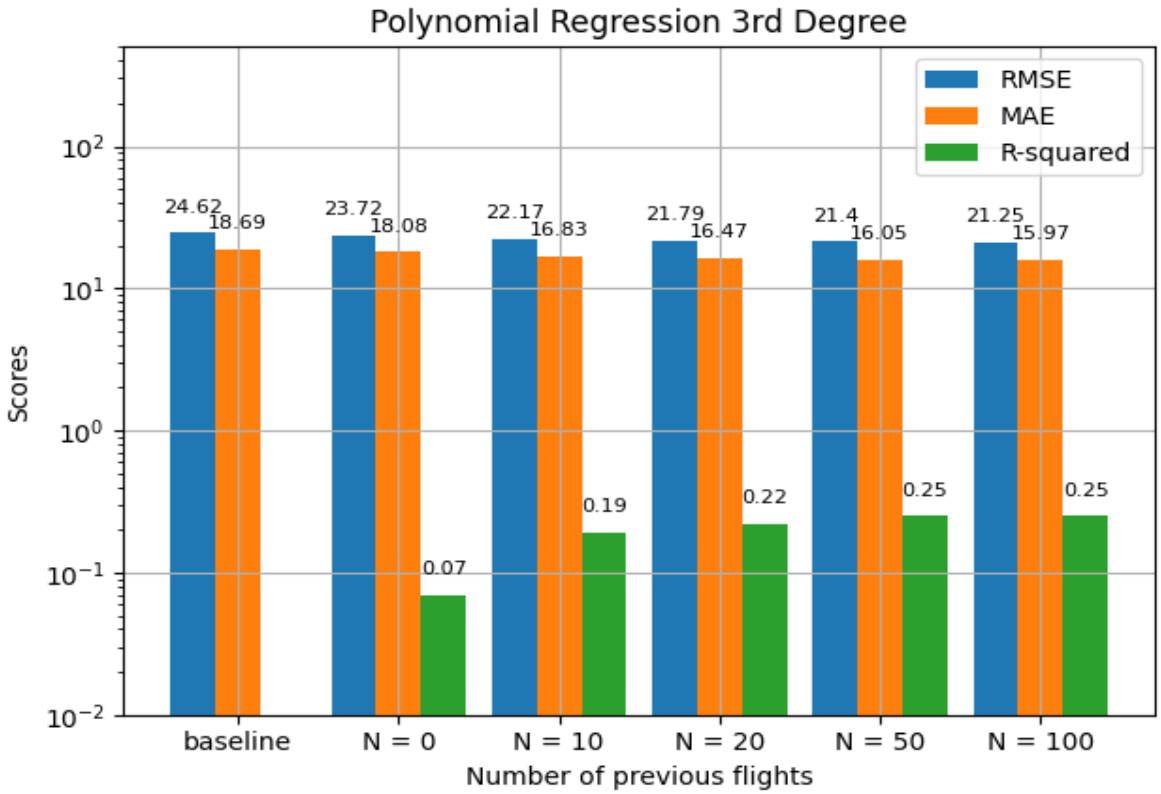

Figure 5.74: Total delay prediction single data model Polynomial Regression 3rd degree results

5.4.3 Support Vector Regression

The process of developing a prediction model using the Support Vector Regression technique in Python following steps as section 5.1.3:

The results are presented in Figure 5.75:

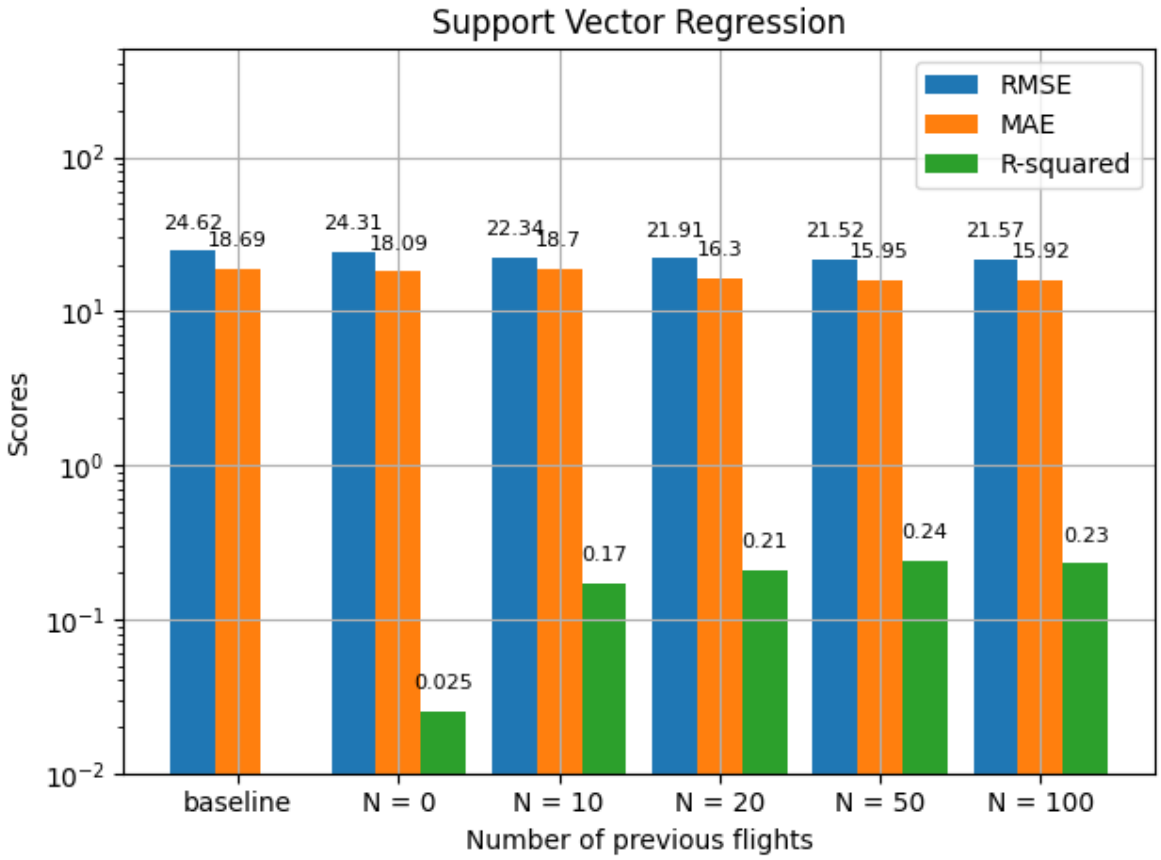

Figure 5.75: Total delay prediction single data model SVR results

5.4.4 Feed Forward Neural Network

The process of developing a prediction model using the Support Vector Regression technique in Python following steps as section 5.1.4:

The results are presented in Figure 5.76, 5.77, 5.78, 5.79:

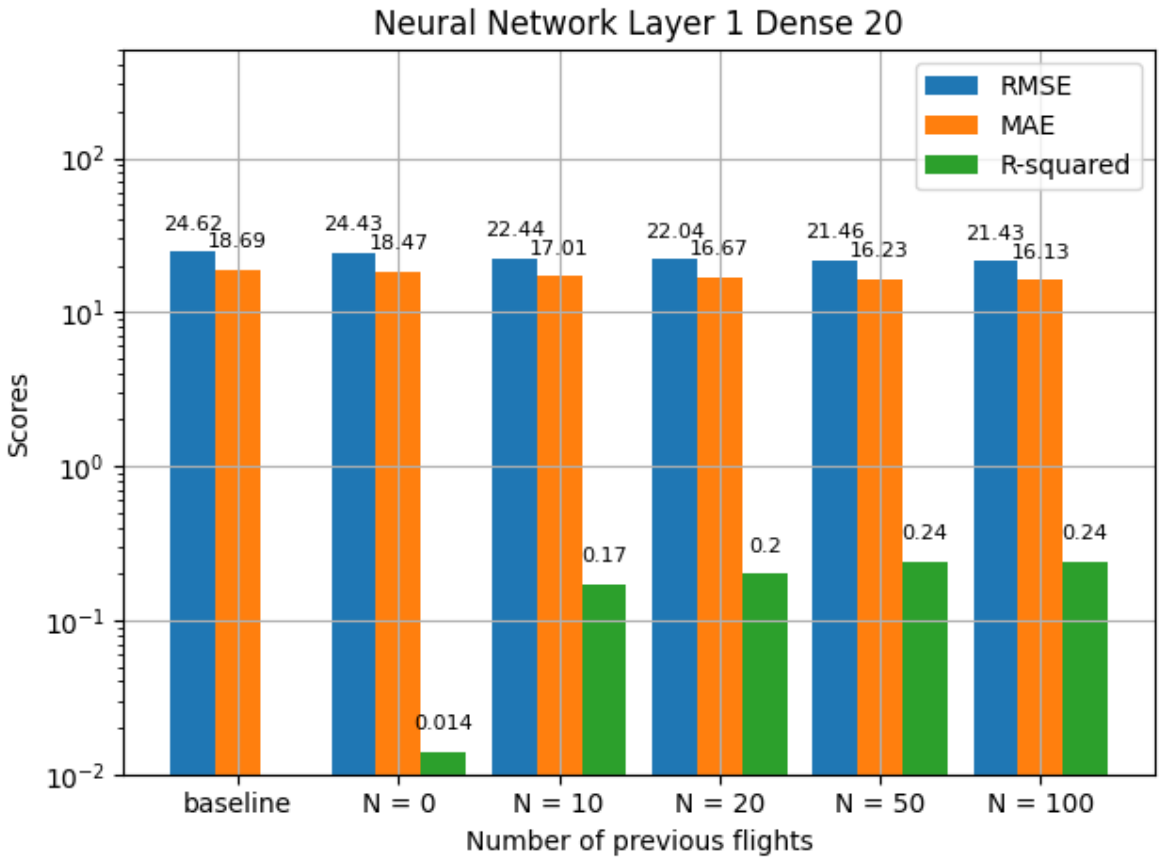

Figure 5.76: Total delay prediction single data model Neural Network layer=1, dense=20 results

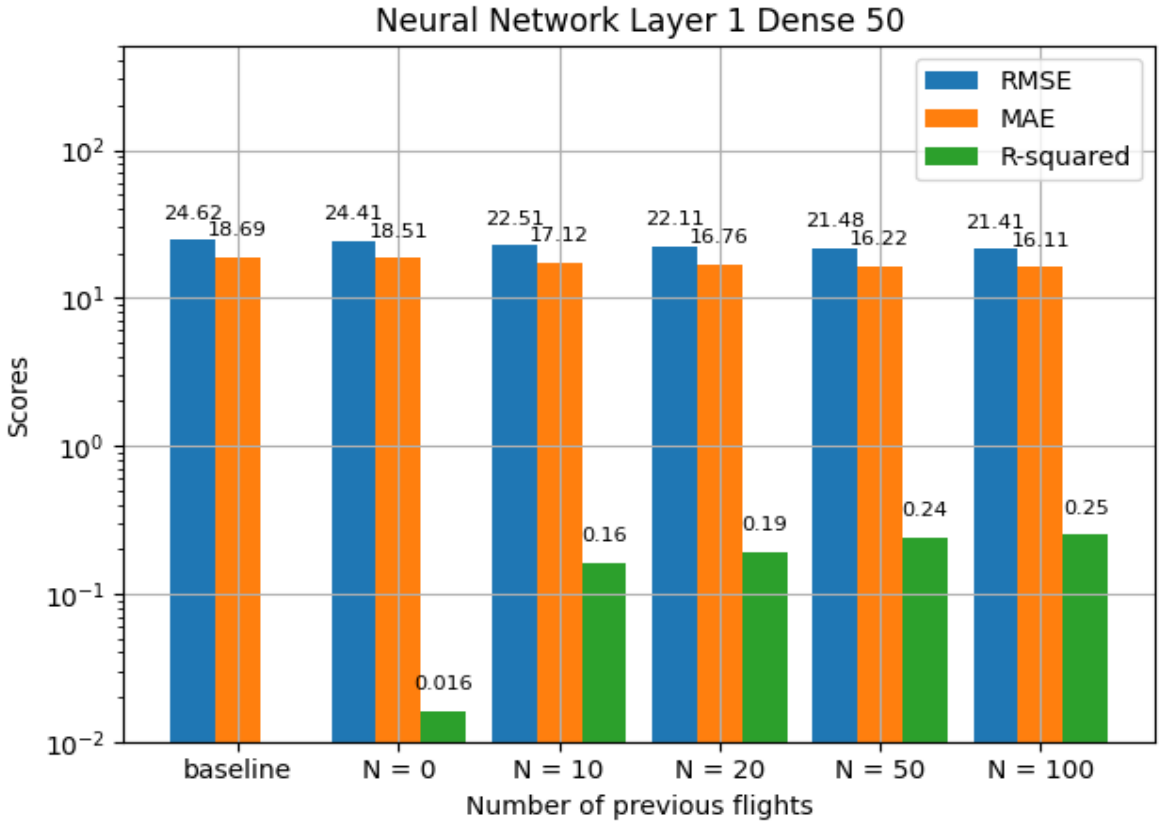

Figure 5.77: Total delay prediction single data model Neural Network layer=1, dense=50 results

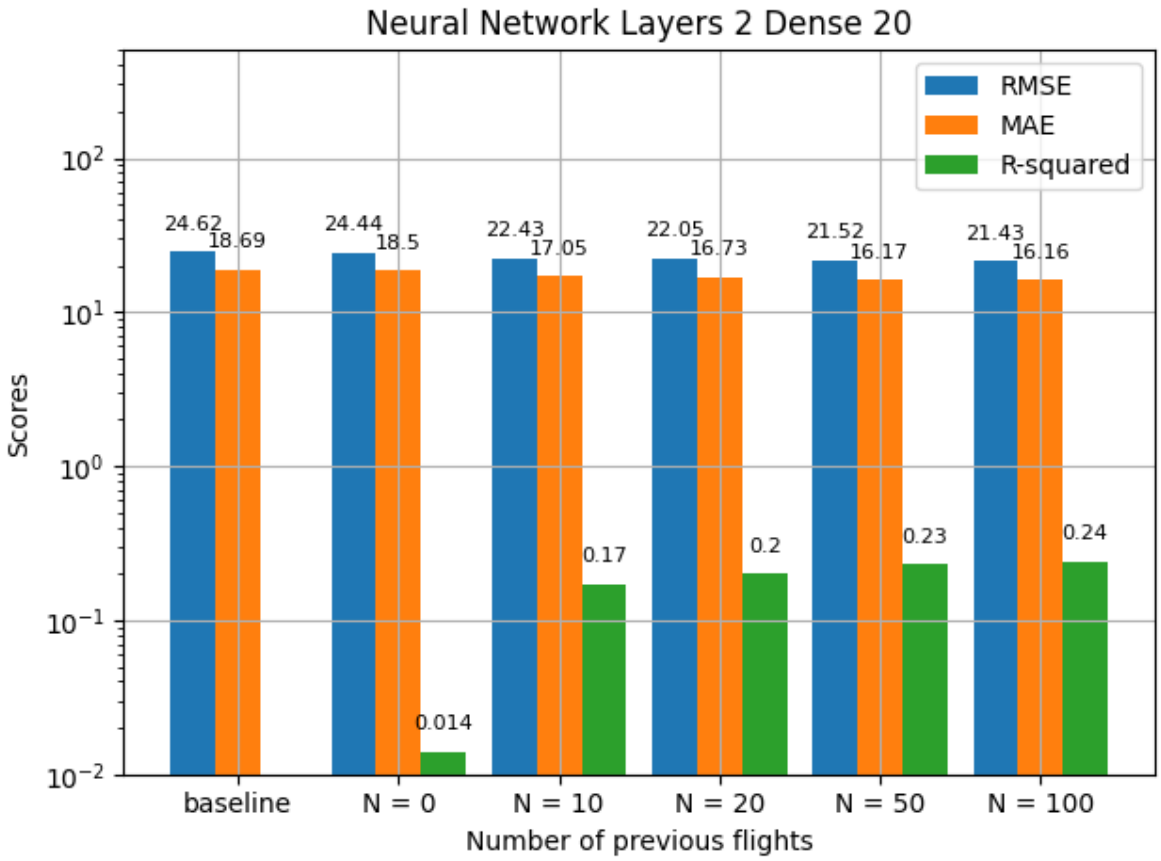

Figure 5.78: Total delay prediction single data model Neural Network layer=2, dense=20 results

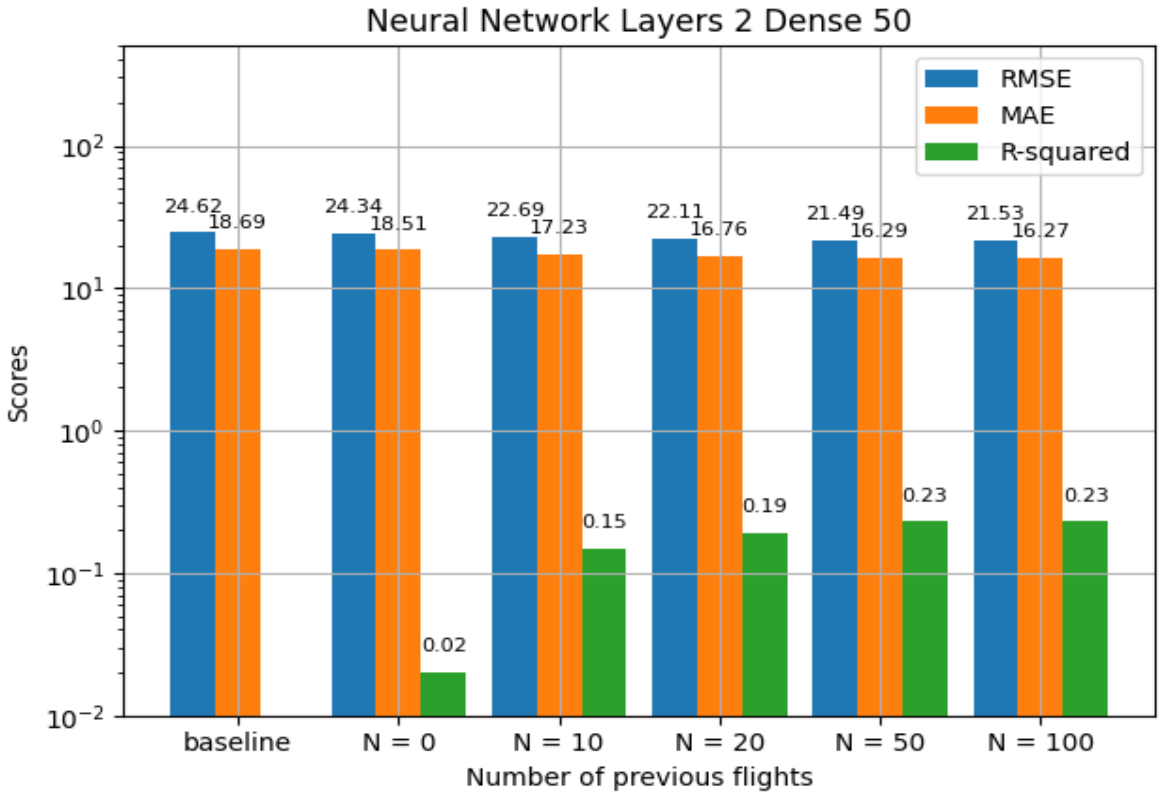

Figure 5.79: Total delay prediction single data model Neural Network layer=2, dense=50 results

5.4.5 Summary of Total Delay Prediction single data model

The following Table 5.8 presents the best obtained outcomes for Total Delay Prediction single data model of each algorithm based on the metrics and time measurements (where n=number of previous flights, l=number of layers of Neural Network, d=number of dense):

Algorithms	RMSE	MAE	R-Squared	Time (seconds)
Linear Regression (n=50)	21.34±0.19	16.08±0.12	0.24±0.01	7.87
Polynomial Regression 2nd (n=100)	20.69±0,51	15.54±0.20	0.29±0.02	8.75
SVR (n=50)	21.52±0.50	15.95±0.29	0.24±0.02	66.37
Neural Networks (n=100, l=1, d=20)	21.41±0.28	16.11±0.15	0.25±0.001	107.1
Baseline (mean)	24.12±0.35	18.69±0.31	0	null

Table 5.8: Summary algorithm comparison Total Delay Prediction single data model

From the data presented above, we can see that the overall delay prediction for a single dataset has a significant baseline error of 24.12. However, this error is smaller than the combined errors of the separate datasets and slightly better than the first strategy. The predictions generated by our algorithms demonstrate a significant enhancement in the RMSE metric by 14.22%, an improvement of 16.85% in the MAE metric, and a 0.29 increase in the R-Squared metric. The expertise gained from earlier flights is highly valuable for a multitude of subsequent flights. All algorithms yield comparable outcomes with variances on the order of 4%. Futhermore, it is noticeable that the baseline (mean) has R-Squared value zero.

Overall, the strategy of using a single dataset to calculate the total delay tends to yield less errors compared to summing the total delay from sum of models. The 2nd degree Polynomial Regression algorithm, when provided with prior knowledge of the last 100 flight delays, yields superior results in terms of RMSE, MAE, and R-Squared metrics. Linear Regression demonstrates superior time performance, completing execution in 7.87 seconds, while Neural Network exhibits the poorest performance, requiring 107.1 seconds. Best results are shown as radar matrix plot in Figure 5.80.

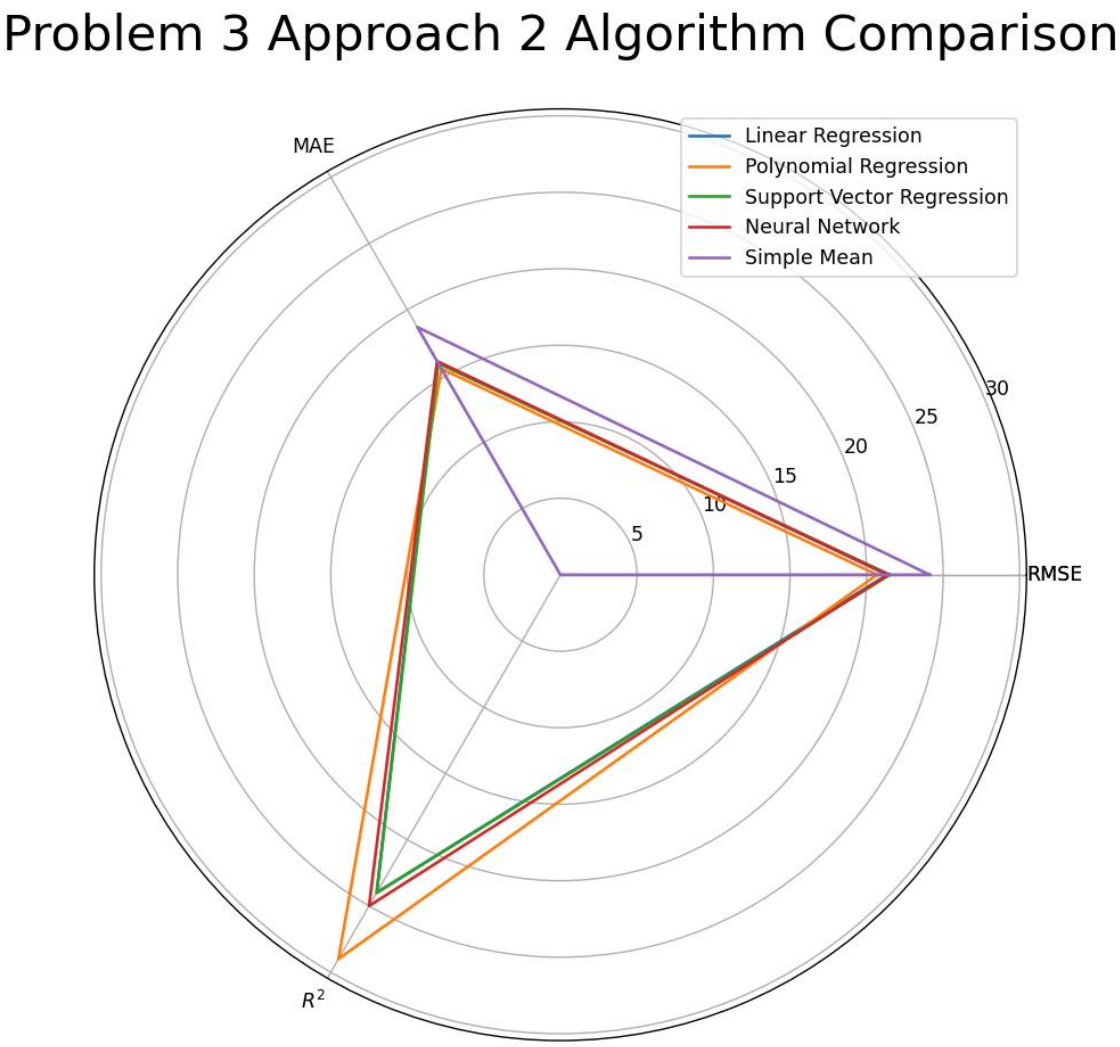

Figure 5.80: Total delay prediction single model algorithm comparison radar matrix

Chapter 6
Conclusion and Future Work

6.1 Review of Research Questions

Could Machine Learning Techniques make a more accurate forecast than utilizing the simple average flight delay of the dataset?

Machine learning techniques have been found to yield superior outcomes compared to the simple average. However, it has been observed that neural networks, when not incorporating the previous knowledge feature, do not exhibit improved predictive capabilities over the simple average.

Does incorporating prior departure and inflight delays, and new feature knowledge, at the respective airports, enhance the Machine Learning predictions?

It is evident that possessing knowledge of the previous delay of N flights yields superior outcomes in all three issues and across all models. However, when the level of information about flights rises, the improvement in performance does not necessarily follow.

Using the metrics of the Regression model, which of the two ways yields better results for the total delay prediction?

The second methodology employed in addressing the problem of total delay prediction yields marginally superior outcomes. The performance of the single model in comparison to the cumulative predictions of multiple models exhibits a minor improvement.

Which Machine Learning model produces the most accurate predictions?

Superior performance in comparison to the tested machine learning models and implementations is attained through the utilization of the following alternative approaches. To forecast the average delay for the initial departure, a 3rd degree polynomial regression model was employed, utilizing data from 50 preceding flights. Conversely, for predicting in-flight delays, a 2nd degree model was utilized, this model was trained using information from the past 100 flights. In terms of total delay prediction, the first approach, which combines the simple sum of the departure and inflight delay models, yielded slightly inferior results compared to the second approach. The second approach, which utilizes a 2nd degree polynomial regression model with prior knowledge of the last 100 flights, demonstrated the most accurate predictions when employed as a single prediction model.

6.2 Future Work

Due to a lack of time, numerous modifications, tests, and experiments have been postponed in the future (i.e., studies using real data are typically quite time-consuming, requiring days to complete a single run). Future research will involve a more in-depth examination of particular mechanisms, new recommendations to test out alternative ways, or simple interest.

There are numerous prospective extensions of the project as future work. As indicated previously, it would be highly exciting to test the performance of all baseline models by substituting the employed Regression algorithms with others, such as Ridge Regression, Lasso Regression, and Decision Tree Regression, to compare their performance.

In conclusion, this implementation does not account for meteorological data, airport safety, the national aviation system, or flight cancellations due to mechanical failure. Therefore, a strategy that incorporates these aspects would be more plausible. Lastly, it would be beneficial to test the developed predictions and models on other pairs of airports in the United States and, if a comparable dataset is obtained on other airport pairs in Europe or even Asia and the Middle East.

Appendices

Appendix A

SETUP

All scripts are developed in Python 3.10 and can be accessed by email to be sent at: chatzipetrosa@gmail.com.

Laptop: DELL XPS 15

Processor: Intel(R) Core(TM) i7-10750H CPU @ 2.60GHz 2.59 GHz

RAM: 20GB

Coding environment: PyCharm

Python libraries: sklearn, keras, matplotlib, pandas, numpy, seaborn

Appendix B

ADVANTAGES AND DISADVANTAGES OF THE MACHINE LEARNING ALGORITHMS CONCERNED

Table B.1: Strengths and weakness of Linear Regression [80]

Strengths	Weaknesses
The implementation of Linear Regression is straightforward and the interpretation of the output coefficients is quite uncomplicated.	Conversely, in the context of linear regression, outliers can exert significant influence on the regression model, although the bounds established by this technique remain linear.
When the presence of a linear relationship between the independent and dependent variables is known, this approach is considered the most suitable due to its lower complexity in comparison to alternative algorithms.	Linear regression presupposes a linear association between the dependent and independent variables, exhibiting limited diversity in its use. This implies that it presupposes the existence of a linear correlation between the variables. The assumption of independence is made between attributes.
Linear regression is prone to overfitting, but this issue can be mitigated through the utilization of dimensionality reduction techniques, regularization methods, such as L1 and L2 regularization, as well as cross-validation.	Linear regression examines the association between the average values of the dependent variables and the independent variables. Similar to how the mean does not provide a comprehensive depiction of a singular variable, linear regression fails to offer a comprehensive depiction of the relationships between variables.

Table B.2: Strengths and weakness of Polynomial Regression [81]

Strengths	Weaknesses
A diverse array of functions can be encompassed within it.	Variables have a high sensitivity to outliers.
Polynomials has the ability to effectively accommodate a diverse array of curvatures.	The inclusion of a small number of outliers within a dataset has the potential to significantly impact the outcomes of nonlinear analysis.
The polynomial function offers the most accurate approximation of the association between the dependent and independent variables.	Moreover, it is regrettable that the availability of model validation techniques for identifying outliers in nonlinear regression is comparatively limited in comparison to those available for linear regression.

Table B.3: Strengths and weakness of Support Vector Regression [82]

Strengths	Weaknesses
The method exhibits robustness in the presence of outliers.	Large datasets are not appropriate for this particular application
The model demonstrates exceptional generalization capacity, exhibiting a high degree of accuracy in its predictions.	When the amount of features for each data point surpasses the number of training data samples, it may be observed that the performance of the Support Vector Regression (SVR) is suboptimal.
The decision model exhibits a high degree of flexibility in terms of its ability to be altered, and its implementation process is characterized by simplicity and ease.	The performance of the Decision model is adversely affected in scenarios when the data set has a higher level of noise, specifically when the target classes exhibit overlapping characteristics.

Table B.4: Strengths and weakness of Feed Forward Neural Networks [51]

Strengths	Weaknesses
Feedforward neural networks can express complex, non-linear input-goal relationships. This makes them ideal for detecting complicated data patterns that linear models miss.	When model complexity exceeds training data, neural networks, especially multi-layer ones, overfit. Checking validation loss may solve this problem, but they require careful calibration.
Feature Learning: Neural networks may automatically learn relevant features from data, eliminating the need for feature engineering. This functionality can be useful for datasets with several dimensions or when distinctive interrelationships are unknown.	Neural networks may need lots of training data to generalize. Neural networks may function poorly with minimal data. Regression scenarios might be limited by data availability.
Neural networks can be scaled and applied to a variety of regression situations, including small datasets and large big data scenarios. Deep networks with several hidden layers can solve challenging regression problems.	Due to their difficulties in deciphering representations and understanding predictions, neural networks are frequently called "opaque" models. In industries or regulatory settings where openness and explainability are vital, the lack of interpretability may be a disadvantage.

Bibliography

[1] "Airports Council International releases 2015 World Airport Traffic Report: The busiest become busier; the year of the international hub airport - ACI World." Accessed: Oct. 19, 2022. [Online]. Available: https://aci.aero/2016/09/09/airports-council-international-releases-2015-world-airport-traffic-report-the-busiest-become-busier-the-year-of-the-international-hub-airport/

[2] "OST_R | BTS | Title from h2." Accessed: Oct. 19, 2022. [Online]. Available: https://www.transtats.bts.gov/OT_Delay/OT_DelayCause1.asp?20=E

[3] "Schumer Releases Report Detailing Over $40 Billion in Costs of Flight Delays To Passengers, Airlines, and U.S. Economy As Memorial Day Travel Season Approaches - Schumer Releases Report Detailing Over $40 Billion in Costs of Flight Delays To Passengers, Airlines, and U.S. Economy As Memorial Day Travel Season Approaches - United States Joint Economic Committee." Accessed: Oct. 19, 2022. [Online]. Available: https://www.jec.senate.gov/public/index.cfm/democrats/2008/5/schumer-releases-report-detailing-over-$40-billion-in-costs-of-flight-delays-to-passengers-airlines-and-u.s-economy-as-memorial-day-travel-season-approaches_1137

[4] M. Ball *et al.*, "Total delay impact study : a comprehensive assessment of the costs and impacts of flight delay in the United States," Oct. 2010. Accessed: Oct. 19, 2022. [Online]. Available: https://rosap.ntl.bts.gov/view/dot/6234

[5] "Your Flight Has Been Delayed Again - Your Flight Has Been Delayed Again - United States Joint Economic Committee." Accessed: Oct. 26, 2022. [Online]. Available: https://www.jec.senate.gov/public/index.cfm/democrats/2008/5/your-flight-has-been-delayed-again_1539

[6] "OST_R | BTS | Transtats." Accessed: Oct. 25, 2022. [Online]. Available: https://www.transtats.bts.gov/

[7] T. Zhou, Q. Gao, X. Chen, and Z. Xun, "Flight Delay Prediction Based on Characteristics of Aviation Network," *MATEC Web Conf.*, vol. 259, p. 02006, 2019, doi: 10.1051/matecconf/201925902006.

[8] C.-L. Wu, "Inherent delays and operational reliability of airline schedules," *J. Air Transp. Manag.*, vol. 11, no. 4, pp. 273–282, Jul. 2005, doi: 10.1016/j.jairtraman.2005.01.005.

[9] Z. W. Zhong, D. Varun, and Y. J. Lin, "Studies for air traffic management R&D in the ASEAN-region context," *J. Air Transp. Manag.*, vol. 64, pp. 15–20, Sep. 2017, doi: 10.1016/j.jairtraman.2017.06.020.

[10] D. Bertsimas and M. Frankovich, "Unified Optimization of Traffic Flows Through Airports," *Transp. Sci.*, vol. 50, no. 1, pp. 77–93, Feb. 2016, doi: 10.1287/trsc.2015.0590.

[11] E. P. Gilbo, “Optimizing airport capacity utilization in air traffic flow management subject to constraints at arrival and departure fixes,” *IEEE Trans. Control Syst. Technol.*, vol. 5, no. 5, pp. 490–503, Sep. 1997, doi: 10.1109/87.623035.

[12] M. Hansen, “Micro-level analysis of airport delay externalities using deterministic queuing models: a case study,” *J. Air Transp. Manag.*, vol. 8, no. 2, pp. 73–87, Mar. 2002, doi: 10.1016/S0969-6997(01)00045-X.

[13] N. Pyrgiotis, K. Malone, and A. Odoni, “Modelling delay propagation within an airport network,” *Transp. Res. Part C Emerg. Technol.*, vol. 27, Jan. 2011, doi: 10.1016/j.trc.2011.05.017.

[14] N. Chakrabarty, "*A Data Mining Approach to Flight Arrival Delay Prediction for American Airlines*." IEMECON 2019, p. 107. doi: 10.1109/IEMECONX.2019.8876970.

[15] R. Joseph, “Grid Search for model tuning,” Medium. Accessed: Oct. 27, 2022. [Online]. Available: https://towardsdatascience.com/grid-search-for-model-tuning-3319b259367e

[16] A. Natekin and A. Knoll, “Gradient Boosting Machines, A Tutorial,” *Front. Neurorobotics*, vol. 7, p. 21, Dec. 2013, doi: 10.3389/fnbot.2013.00021.

[17] Y. Dong and X. Wang, *A New Over-Sampling Approach: Random-SMOTE for Learning from Imbalanced Data Sets", KSEM 2011,* vol. 7091. 2011, p. 352. doi: 10.1007/978-3-642-25975-3_30.

[18] J. Li, H. Li, and J.-L. Yu, “Application of Random-SMOTE on Imbalanced Data Mining,” in *2011 Fourth International Conference on Business Intelligence and Financial Engineering*, Oct. 2011, pp. 130–133. doi: 10.1109/BIFE.2011.25.

[19] Y. Ding, “Predicting flight delay based on multiple linear regression,” *IOP Conf. Ser. Earth Environ. Sci.*, vol. 81, no. 1, p. 012198, Aug. 2017, doi: 10.1088/1755-1315/81/1/012198.

[20] J. D. Jobson, “Multiple Linear Regression,” in *Applied Multivariate Data Analysis: Regression and Experimental Design*, J. D. Jobson, Ed., in Springer Texts in Statistics. , New York, NY: Springer, 1991, pp. 219–398. doi: 10.1007/978-1-4612-0955-3_4.

[21] G. I. Webb, “Naïve Bayes,” in *Encyclopedia of Machine Learning*, C. Sammut and G. I. Webb, Eds., Boston, MA: Springer US, 2010, pp. 713–714. doi: 10.1007/978-0-387-30164-8_576.

[22] J. R. Quinlan, Ed., “The Morgan Kaufmann Series in Machine Learning in *C4.5*,”, San Francisco (CA): Morgan Kaufmann, 1993, p. ii. doi: 10.1016/B978-0-08-050058-4.50001-2.

[23] S. L. Gortmaker, D. Hosmer, and S. Lemeshow, “Applied Logistic Regression,” *Contemp Sociol*, vol. 23, Jan. 2013, doi: 10.1002/9781118548387.ch4.

[24] L. Breiman, “Random Forests,” *Mach. Learn.*, vol. 45, no. 1, pp. 5–32, Oct. 2001, doi: 10.1023/A:1010933404324.

[25] W.-Y. Loh, “Classification and Regression Trees,” *Wiley Interdiscip. Rev. Data Min. Knowl. Discov.*, vol. 1, pp. 14–23, Jan. 2011, doi: 10.1002/widm.8.

[26] W. Koehrsen, “Introduction to Bayesian Linear Regression,” Medium. Accessed: Oct. 27, 2022. [Online]. Available: https://towardsdatascience.com/introduction-to-bayesian-linear-regression-e66e60791ea7

[27] P. Meel, M. Singhal, M. Tanwar, and N. Saini, "*Predicting Flight Delays with Error Calculation using Machine Learned Classifiers*.", SPIN 2020, p. 76. doi: 10.1109/SPIN48934.2020.9071159.

[28] C. Sammut and G. I. Webb, Eds., "Mean Squared Error," in *Encyclopedia of Machine Learning*, Boston, MA: Springer US, 2010, pp. 653–653. doi: 10.1007/978-0-387-30164-8_528.

[29] Stephanie, "Explained Variance / Variation," Statistics How To. Accessed: Oct. 27, 2022. [Online]. Available: https://www.statisticshowto.com/explained-variance-variation/

[30] J. Miles, "R Squared, Adjusted R Squared," in *Wiley StatsRef: Statistics Reference Online*, John Wiley & Sons, Ltd, 2014. doi: 10.1002/9781118445112.stat06627.

[31] H. Schulz and S. Behnke, "Deep Learning," *KI - Künstl. Intell.*, vol. 26, no. 4, pp. 357–363, Nov. 2012, doi: 10.1007/s13218-012-0198-z.

[32] Y. J. Kim, S. Choi, S. Briceno, and D. Mavris, "*A deep learning approach to flight delay prediction*.", DASC 2016, p. 6. doi: 10.1109/DASC.2016.7778092.

[33] K. Laskey, N. Xu, and C.-H. Chen, "*Propagation of Delays in the National Airspace System*. ", 22nd Conference on Uncertainty in UAI 2006, source arXiv.

[34] N. Pyrgiotis, K. M. Malone, and A. Odoni, "Modelling delay propagation within an airport network," *Transp. Res. Part C Emerg. Technol.*, vol. 27, pp. 60–75, Feb. 2013, doi: 10.1016/j.trc.2011.05.017.

[35] B. Lantz, *Machine Learning with R: Discover how to Build Machine Learning Algorithms, Prepare Data, and Dig Deep Into Data Prediction Techniques with R*. Packt Publishing, 2015.

[36] I. Goodfellow, Y. Bengio, and A. Courville, *Deep Learning*. MIT Press, 2016.

[37] "Regression vs. Classification in Machine Learning for Beginners | Simplilearn," Simplilearn.com. Accessed: Nov. 09, 2022. [Online]. Available: https://www.simplilearn.com/regression-vs-classification-in-machine-learning-article

[38] K. P. Sinaga and M.-S. Yang, "Unsupervised K-Means Clustering Algorithm," *IEEE Access*, vol. 8, pp. 80716–80727, 2020, doi: 10.1109/ACCESS.2020.2988796.

[39] L. P. Kaelbling, M. L. Littman, and A. W. Moore, "Reinforcement Learning: A Survey," *J. Artif. Intell. Res.*, vol. 4, pp. 237–285, May 1996, doi: 10.1613/jair.301.

[40] C. M. Bishop and P. of N. C. C. M. Bishop, *Neural Networks for Pattern Recognition*. Clarendon Press, 1995.

[41] L. V. Fausett, *Fundamentals of Neural Networks: Architectures, Algorithms, and Applications*. Pearson Education, 1994.

[42] S. J. Russell and P. Norvig, *Artificial intelligence: a modern approach*. in Prentice Hall series in artificial intelligence. Englewood Cliffs, N.J.: Prentice Hall, 1995. Accessed: Nov. 09, 2022. [Online].

[43] L. Alzubaidi *et al.*, "Review of deep learning: concepts, CNN architectures, challenges, applications, future directions," *J. Big Data*, vol. 8, no. 1, p. 53, Mar. 2021, doi: 10.1186/s40537-021-00444-8.

[44] L. Shukla, "Designing Your Neural Networks," Medium. Accessed: Nov. 09, 2022. [Online]. Available: https://towardsdatascience.com/designing-your-neural-networks-a5e4617027ed

[45] F. Rosenblatt, “The perceptron: A probabilistic model for information storage and organization in the brain,” *Psychol. Rev.*, vol. 65, pp. 386–408, 1958, doi: 10.1037/h0042519.

[46] X. Zhou, W. Zhang, Z. Chen, S. DIAO, and T. Zhang, “Efficient Neural Network Training via Forward and Backward Propagation Sparsification,” in *Advances in Neural Information Processing Systems*, Curran Associates, Inc., 2021, pp. 15216–15229. Accessed: Nov. 11, 2022. [Online]. Available: https://proceedings.neurips.cc/paper/2021/hash/80f2f15983422987ea30d77bb531be86-Abstract.html

[47] H. Zhao, O. Gallo, I. Frosio, and J. Kautz, “Loss Functions for Neural Networks for Image Processing.” IEEE, Dec. 23, 2016. doi: 10.1109/TCI.2016.2644865.

[48] “Single Layer Perceptron in TensorFlow - Javatpoint,” www.javatpoint.com. Accessed: Nov. 15, 2022. [Online]. Available: https://www.javatpoint.com/single-layer-perceptron-in-tensorflow

[49] F. Murtagh, “Multilayer perceptrons for classification and regression,” *Neurocomputing*, vol. 2, no. 5, pp. 183–197, Jul. 1991, doi: 10.1016/0925-2312(91)90023-5.

[50] S. SHARMA, “Activation Functions in Neural Networks,” Medium. Accessed: Nov. 15, 2022. [Online]. Available: https://towardsdatascience.com/activation-functions-neural-networks-1cbd9f8d91d6

[51] L. Pauly, H. Peel, S. Luo, D. Hogg, and R. Fuentes, *Deeper Networks for Pavement Crack Detection*. 2017. doi: 10.22260/ISARC2017/0066.

[52] A. L. Maas, A. Y. Hannun, and A. Y. Ng, “Rectifier nonlinearities improve neural network acoustic models,” in *Proc. ICML*, Atlanta, Georgia, USA, 2013, p. 3.

[53] X. Glorot, A. Bordes, and Y. Bengio, “Deep Sparse Rectifier Neural Networks,” in *Proceedings of the Fourteenth International Conference on Artificial Intelligence and Statistics*, JMLR Workshop and Conference Proceedings, Jun. 2011, pp. 315–323. Accessed: Nov. 18, 2022. [Online]. Available: https://proceedings.mlr.press/v15/glorot11a.html

[54] “Papers with Code - Leaky ReLU Explained.” Accessed: Nov. 18, 2022. [Online]. Available: https://paperswithcode.com/method/leaky-relu

[55] S. Geman, E. Bienenstock, and R. Doursat, “Neural Networks and the Bias/Variance Dilemma,” *Neural Comput.*, vol. 4, no. 1, pp. 1–58, Jan. 1992, doi: 10.1162/neco.1992.4.1.1.

[56] M. Leshno, V. Ya. Lin, A. Pinkus, and S. Schocken, “Multilayer feedforward networks with a nonpolynomial activation function can approximate any function,” *Neural Netw.*, vol. 6, no. 6, pp. 861–867, Jan. 1993, doi: 10.1016/S0893-6080(05)80131-5.

[57] “Population dynamics: Variance and the sigmoid activation function - ScienceDirect.” Accessed: Nov. 11, 2022. [Online]. Available: https://www.sciencedirect.com/science/article/abs/pii/S1053811908005132

[58] L. Bottou and T. B. Laboratories, “Stochastic Gradient Learning in Neural Networks,” Proceedings of Neuro-Nımes, 1991.

[59] Y. Li, C. Wei, and T. Ma, “Towards Explaining the Regularization Effect of Initial Large Learning Rate in Training Neural Networks,” in *Advances in Neural Information Processing Systems*, Curran Associates,

Inc., 2019. Accessed: Nov. 12, 2022. [Online]. Available: https://proceedings.neurips.cc/paper/2019/hash/bce9abf229ffd7e570818476ee5d7dde-Abstract.html

[60] S. V. and S. Das, *Deep Learning*. Accessed: Nov. 18, 2022. [Online]. Available: https://srdas.github.io/DLBook/GradientDescentTechniques.html

[61] "Understanding Linear Regression." Accessed: Nov. 18, 2022. [Online]. Available: https://community.cloudera.com/t5/Community-Articles/Understanding-Linear-Regression/ta-p/281391

[62] "The Ultimate Guide to Linear Regression for Machine Learning." Accessed: Nov. 18, 2022. [Online]. Available: https://www.keboola.com/blog/linear-regression-machine-learning

[63] Kimura, Yoshio, and Hitoshi Kondo. "A Note on the Generalised Gauss-Markov Theorem on Least Squares Estimation." Studies in Regional Science 28, no. 2 (1998): 67–75. http://dx.doi.org/10.2457/srs.28.2_67..

[64] "Machine learning Polynomial Regression - Javatpoint," www.javatpoint.com. Accessed: Nov. 18, 2022. [Online]. Available: https://www.javatpoint.com/machine-learning-polynomial-regression

[65] "RMSE: Root Mean Square Error," Statistics How To. Accessed: Nov. 20, 2022. [Online]. Available: https://www.statisticshowto.com/probability-and-statistics/regression-analysis/rmse-root-mean-square-error/

[66] "5 Concepts You Should Know About Gradient Descent and Cost Function," KDnuggets. Accessed: Nov. 20, 2022. [Online]. Available: https://www.kdnuggets.com/5-concepts-you-should-know-about-gradient-descent-and-cost-function.html

[67] "Stanford Engineering Everywhere | CS229 - Machine Learning." Accessed: Nov. 20, 2022. [Online]. Available: https://see.stanford.edu/course/cs229

[68] V. V. N, "Statistical Learning Theory," *Adapt. Learn. Syst. Signal Process. Commun. Control*, 1998, Accessed: Nov. 22, 2022. [Online]. Available: https://cir.nii.ac.jp/crid/1573668924951096832

[69] R. Gandhi, "Support Vector Machine — Introduction to Machine Learning Algorithms," Medium. Accessed: Nov. 22, 2022. [Online]. Available: https://towardsdatascience.com/support-vector-machine-introduction-to-machine-learning-algorithms-934a444fca47

[70] A. A. Tokuç, "Why Feature Scaling in SVM? | Baeldung on Computer Science." Accessed: Nov. 22, 2022. [Online]. Available: https://www.baeldung.com/cs/svm-feature-scaling

[71] G. Baudat and F. Anouar, "Kernel-based methods and function approximation," in *IJCNN'01. International Joint Conference on Neural Networks. Proceedings (Cat. No.01CH37222)*, Jul. 2001, pp. 1244–1249 vol.2. doi: 10.1109/IJCNN.2001.939539.

[72] "Implicit Lifting and the Kernel Trick." Accessed: Nov. 23, 2022. [Online]. Available: https://gregorygundersen.com/blog/2019/12/10/kernel-trick/

[73] P. Parashar, "Support Vector Regression and it's Mathematical Implementation," The Startup. Accessed: Nov. 23, 2022. [Online]. Available: https://medium.com/swlh/support-vector-regression-and-its-mathematical-implementation-4800456e4878

[74] J. Wang, Q. Chen, and Y. Chen, “RBF Kernel Based Support Vector Machine with Universal Approximation and Its Application,” in *Advances in Neural Networks – ISNN 2004*, F.-L. Yin, J. Wang, and C. Guo, Eds., in Lecture Notes in Computer Science. Berlin, Heidelberg: Springer, 2004, pp. 512–517. doi: 10.1007/978-3-540-28647-9_85.

[75] T.-T. Pham, “Modele De Graphe Et Modele De Langue Pour La Reconnaissance De Scenes Visuelles,” Conference: COnférence en Recherche d'Infomations et Applications - CORIA 2009, 6th French Information Retrieval Conference, Presqu'île de Giens, France, May 5-7, 2009.

[76] C. Cortes and V. Vapnik, “Support-vector networks,” *Mach. Learn.*, vol. 20, no. 3, pp. 273–297, Sep. 1995, doi: 10.1007/BF00994018.

[77] M. Awad and R. Khanna, “Machine Learning and Knowledge Discovery,” in *Efficient Learning Machines: Theories, Concepts, and Applications for Engineers and System Designers*, M. Awad and R. Khanna, Eds., Berkeley, CA: Apress, 2015, pp. 19–38. doi: 10.1007/978-1-4302-5990-9_2.

[78] K. P. Bennett and O. L. Mangasarian, “Robust linear programming discrimination of two linearly inseparable sets,” *Optim. Methods Softw.*, vol. 1, no. 1, pp. 23–34, Jan. 1992, doi: 10.1080/10556789208805504.

[79] “Kernel Based Algorithms for Mining Huge Data Sets | SpringerLink.” Accessed: Nov. 26, 2022. [Online]. Available: https://link.springer.com/book/10.1007/3-540-31689-2

[80] O. L. Mangasarian, *Nonlinear programming*. in McGraw-Hill series in systems science. New York: McGraw-Hill, 1969.

[81] G. P. McCormick, *Nonlinear Programming: Theory, Algorithms and Applications*, 1st edition. New York: Wiley, 1983.

[82] Nello Christianini, John Swawe-Taylor,“An Introduction to Support Vector Machines and Other Kernel-based Learning Methods.”, Cambridge University Press 2000, Accessed: Nov. 26, 2022. [Online]. Available: https://www.cambridge.org/core/books/an-introduction-to-support-vector-machines-and-other-kernelbased-learning-methods/A6A6F4084056A4B23F88648DDBFDD6FC

[83] A. J. Smola and B. Schölkopf, “A tutorial on support vector regression,” *Stat. Comput.*, vol. 14, no. 3, pp. 199–222, Aug. 2004, doi: 10.1023/B:STCO.0000035301.49549.88.

[84] T. Hofmann, B. Schölkopf, and A. J. Smola, “Kernel methods in machine learning,” *Ann. Stat.*, vol. 36, no. 3, Jun. 2008, doi: 10.1214/009053607000000677.

[85] “sklearn.svm.SVR,” scikit-learn. Accessed: Nov. 26, 2022. [Online]. Available: https://scikit-learn/stable/modules/generated/sklearn.svm.SVR.html

[86] “TFMS | Traffic Flow Management System.” Accessed: Dec. 10, 2022. [Online]. Available: https://www.faa.gov/about/office_org/headquarters_offices/ang/offices/tc/library/storyboard/detailed webpages/tfms.html

[87] “Enhanced tactical flow management system (ETFMS).” Accessed: Dec. 10, 2022. [Online]. Available: https://www.eurocontrol.int/system/enhanced-tactical-flow-management-system

[88] Eurocontrol, FAA,"U.S./Europe comparison of air traffic management-related operational performance for 2015.", Eurocontrol Publication Aug. 2016, Accessed: Dec. 10, 2022. [Online]. Available: https://www.eurocontrol.int/publication/useurope-comparison-air-traffic-management-related-operational-performance-2015

[89] "Instrument Flight Rules (IFR) | SKYbrary Aviation Safety." Accessed: Dec. 12, 2022. [Online]. Available: https://www.skybrary.aero/articles/instrument-flight-rules-ifr

[90] International Air Transport Association, "About us.", IATA website 2022, Accessed: Dec. 13, 2022. [Online]. Available: https://www.iata.org/en/about/

[91] K. Gopalakrishnan and H. Balakrishnan, "A comparative analysis of models for predicting delays in air traffic networks," *MIT Web Domain*, Jun. 2017, Accessed: Dec. 17, 2022. [Online]. Available: https://dspace.mit.edu/handle/1721.1/114750

[92] "One-Hot Encoding - an overview | ScienceDirect Topics." Accessed: Dec. 18, 2022. [Online]. Available: https://www.sciencedirect.com/topics/computer-science/one-hot-encoding

[93] T. Mahajan, G. Singh, and G. Bruns, "An Experimental Assessment of Treatments for Cyclical Data.", 2021 Computer Science, Conference for CSU Undergraduates

[94] J. Brownlee, "How to Use StandardScaler and MinMaxScaler Transforms in Python," MachineLearningMastery.com. Accessed: Dec. 22, 2022. [Online]. Available: https://machinelearningmastery.com/standardscaler-and-minmaxscaler-transforms-in-python/

[95] B. Roy, "All about Feature Scaling," Medium. Accessed: Dec. 22, 2022. [Online]. Available: https://towardsdatascience.com/all-about-feature-scaling-bcc0ad75cb35

[96] P. Refaeilzadeh, L. Tang, and H. Liu, "Cross-Validation," in *Encyclopedia of Database Systems*, L. LIU and M. T. ÖZSU, Eds., Boston, MA: Springer US, 2009, pp. 532–538. doi: 10.1007/978-0-387-39940-9_565.

[97] J. Ashfaque and A. Iqbal, "Introduction to Support Vector Machines and Kernel Methods," Researchgate Apr. 2019 uploaded by Johar M. Ashfaque, Available: https://www.researchgate.net/publication/332370436_Introduction_to_Support_Vector_Machines_and_Kernel_Methods

[98] T. Chai and R. R. Draxler, "Root mean square error (RMSE) or mean absolute error (MAE)? – Arguments against avoiding RMSE in the literature," *Geosci. Model Dev.*, vol. 7, no. 3, pp. 1247–1250, Jun. 2014, doi: 10.5194/gmd-7-1247-2014.

[99] "Overfitting and underfitting," Educative: Interactive Courses for Software Developers. Accessed: Dec. 22, 2022. [Online]. Available: https://www.educative.io/answers/overfitting-and-underfitting