



Machine Learning Techniques To Analyze Operator's Behavior

Sai Srivatsava Manchala

This thesis is submitted to the Department of Computer Science & Engineering at Blekinge Institute of Technology in partial fulfillment of the requirements for the degree of Master of Science in Computer Science. The thesis is equivalent to 20 weeks of full-time studies.

Contact Information:

Author(s):

Sai Srivatsava Manchala

E-mail: matc17@student.bth.se

University Advisor:

Yulia Sidorova, PhD

E-mail: yulia.sidorova@bth.se

Department of Computer Science and Engineering

External Advisor:

Erik Berglund, PhD

E-mail: erik.berglund@gmail.com

Faculty of Computing
Blekinge Institute of Technology
SE-371 79 Karlskrona, Sweden

Internet : www.bth.se
Phone : +46 455 38 50 00
Fax : +46 455 38 50 57

Abstract

Background: With savvier management teams, airlines are becoming more stable, more productive, and more profitable. The problems plaguing the aviation industry, however, have not gone away and have become more complicated instead. Schedule recovery is the process of recovery from these issues (also known as operating disturbances). The recovery solver from Jeppesen is a software tool that produces a set of solutions to solve these operational disruptions.

Objectives: In this research work, we review the literature related to disruptions in airlines to understand the state of the art of applying machine learning and decrease the recovery time. The primary goal of this research work is to analyze the Jeppesen airline system and recovery solver extensively, which plays an important role and is used when disturbances occur. In the case of a loss, the recovery solver provides several solutions. The operator can either solve it manually, use a solution created by the recovery solver, or use a combination to solve a disturbance. The research also focuses on identifying various machine learning algorithms that can be used to answer two questions: "Will the operator use the solver" and "If the operator uses the solver, which solution will he prefer"

Methods: First, a literature review is performed to classify effective machine learning algorithms and then consider the findings of the discovery that an experiment is conducted to test the chosen machine learning algorithms. Due to unbalanced classes in the dataset, an experiment is performed to generate a synthetic dataset that is similar to the ground truth. Various steps that are done in the experimentation phase like data collection, preprocessing, and training are described in detail. We also test the performance of various algorithms for machine learning.

Results: The results are presented in conjunction with the literature review and the experiments performed to answer research questions. The performance of the models is then measured using different performance metrics.

Conclusions: We finish the research work with an overall review of sections in the paper. It can be inferred that neural network models and the SVM model do not significantly improve predictive performance compared to the XGBoost model by evaluating the results obtained and considering the real-world scenario this study aims at.

Keywords: Airline disruptions, Machine learning, Supervised learning, Neural networks, Schedule recovery.

Acknowledgments

There are no proper words to convey my sincere gratitude and respect for my supervisor Dr. Yulia Sidorova for her guidance and encouragement. Without her continuous support, this study would hardly have been completed.

I am deeply grateful to Dr. Erik Berglund. He has inspired me and always motivated me to push my limits and always gave me his time to offer me valuable comments toward improving my work and provided me constructive criticism, which helped to develop a broader perspective on my thesis.

Special thanks to Dr. Mattias Grönkvist, Mattias Slabanja, Fredrik Johansson, Viktor Almqvist, and Atif Amin for their help and support.

I sincerely thank and dedicate this to my parents, Dr. Srinivasulu Manchala and Varalaxmi Manchala, for their unconditional trust, timely encouragement, and endless patience. It was their love that raised me up again when I got weary. The family of my sister, Sravani Chunchu and Ravi Chunchu have also been generous with their love and encouragement despite the long distance between us.

My joy knows no bounds in expressing my cordial gratitude to my dear friends Sidhartha Singh and Uttej Reddy. Their encouragement and curiosity were a great help throughout the course of this research work, and I cannot forget my BOIZ who went through hard times together, cheered me on, and celebrating each accomplishment: Ashik, Avinash, Yashwanth, Aakash, and Bhargav.

Finally, I thank with love to Monika. She understands me best. She has been my best friend and great companion, loved, supported, encouraged, entertained, and helped me get through stressful situations in the most positive way. BOP!

Contents

Abstract	i
Acknowledgments	ii
1 Introduction	1
1.1 Aim and Objectives	3
1.2 Research Questions:	3
1.3 Problem Statement	4
1.4 Structure of the thesis	4
2 Background	6
2.1 Machine Learning	6
2.2 Supervised Learning	7
2.3 Ensemble Learning	7
2.3.1 XGBoost(Extreme Gradient Boosting)	8
2.4 Neural Networks	8
2.5 Activation Function(Transfer Function)	10
2.5.1 Sigmoid or Logistic activation function(Soft Step)	10
2.5.2 Hyperbolic tangent (TanH)	11
2.5.3 Softmax	11
2.5.4 Rectified Linear Unit(ReLU)	11
2.6 Deep Learning	12
2.7 Forward Propagation	13
2.8 Backpropagation	13
2.9 Definitions	15
2.9.1 Learning Rate:	15
2.9.2 Batch:	15
2.9.3 Training Epochs:	15
2.9.4 Convergence:	15
2.9.5 Overfitting	16
2.9.6 Cross Validation	16
2.9.7 Normalization:	17
2.9.8 Loss Function/Cost Function:	18
2.9.9 Training Algorithms	18
2.10 Probabilistic Neural Network	18
2.11 Multilayer Perceptron	19
2.12 Support Vector Machine (SVM)	20
2.13 Naive Bayes Classifier	21

3	Related Work	22
4	Methodology	26
4.1	Method	26
4.2	Literature Review	26
4.3	Experiment	27
4.4	Tools used	27
4.4.1	Setting up Software Environment	28
4.5	Data Collection	29
4.6	Datsaset Used	29
4.7	Data preprocessing	32
4.8	Feature Selection	35
4.9	Experimental Setup	36
4.10	Adapted approaches and implementation	37
4.10.1	MultiLayer Perceptron	37
4.10.2	Probabilistic Neural Network	37
4.10.3	XGBoost	38
4.10.4	Support Vector Machine(SVM)	38
4.11	Performance Metrics	39
4.11.1	Confusion Matrix	39
4.11.2	Accuracy	40
4.11.3	Precision	40
4.11.4	Recall	40
4.11.5	F1 score	40
4.11.6	AUROC (Area Under the Receiver Operating Characteristics)	40
5	Results	43
5.1	Performance	43
5.1.1	Multilayer Perceptron results	43
5.1.2	Probabilistic Neural Network results	44
5.1.3	XGBoost results	46
5.1.4	Support Vector Machine (SVM) results	47
5.1.5	AUROC Curve	48
5.2	Results Comparison	51
6	Analysis and Discussion	53
6.1	Answering the Research Questions	53
6.2	Validity Threats	55
6.3	Limitations	56
7	Conclusions and Future Work	57
	References	58

List of Figures

1.1	Crew and Portfolio	2
1.2	Jeppesen airline system	3
2.1	Neural Networks	9
2.2	Sigmoid	10
2.3	Hyperbolic Tangent(TanH)	11
2.4	Softmax	11
2.5	Rectified Linear Unit(ReLU)	12
2.6	Forward Propagation	13
2.7	Backpropagation	14
2.8	Learning rate	15
2.9	Probabilistic Neural Network structure	19
2.10	Multilayer Perceptron with 'l' hidden layers	20
2.11	Support Vector Machine	20
4.1	Steps performed during Literature Review	26
4.2	Selected Solutions	31
4.3	Recovery Solver Solutions	31
4.4	Selected Option	32
4.5	Selected Option	33
4.6	SMOTE	34
4.7	Final dataset	34
4.8	Confusion Matrix	39
4.9	AUC	41
4.10	AUC = 0.8	42
4.11	AUC = 0.5	42
5.1	MLP before cross validation	44
5.2	MLP after cross validation	44
5.3	PNN before cross validation	45
5.4	PNN after cross validation	45
5.5	XGBOOST before cross validation	46
5.6	XGBOOST after cross validation	46
5.7	SVM before cross validation	47
5.8	SVM after cross validation	47
5.9	MLP Confusion Matrix and AUC	48
5.10	PNN Confusion Matrix and AUC	49
5.11	XGBOOST Confusion Matrix and AUC	50

5.12 SVM Confusion Matrix and AUC	51
6.1 Comparison of the Accuracy	54

List of Tables

4.1	Features	30
4.2	Feature Importance	35
4.3	Feature Selection using Recursive Feature Elimination	36
4.4	Selected Features for the Experiment	36
5.1	Comparison of Classifiers using the performance metrics	52
6.1	Accuracy of selected models	55
6.2	Classification Report	55

Over the past century, few inventions have changed the way people around the globe live and experience the world as much as the invention of the aircraft. As we know, air travel has become a normal and routine part of life, to the extent that it is difficult to imagine life without it. It is by far the most convenient and time-saving mode of long-distance transport. The airline industry generally uses different methods and tools for optimization during planning. The plan is made several months before the day of service [15], [3]. An airline's operation requires the allocation of resources to air services, such as aircraft and crew members. In practice, operations are usually associated with disruptions that include severe weather conditions, sick crew members, congested airspace, mechanical failure of aircraft or damaged aircraft, and other causes. Since the airline industry is extremely capital-intensive, airlines are trying to minimize the amount of time spent on the ground. For example, for many commercial airlines, the average, typical time at the gate between flights is only 20 minutes. A single cancellation or extended delay can cause ripple effects all day long. Also, security issues can hold airports closed for hours, causing hundreds of flight cancellations [50]. If these operational disturbances are not properly dealt with, it causes not only a sharp decline in cost efficiency but also a bad reputation that can affect the success streak of the airline [54], [2], [27].

Schedule recovery is the process of recovery on the day of operation from operational disturbances. A disruption management system is used for this purpose, which addresses such situations and reduces the impact on operations. The Operations Control Centers (OCC), consisting of aircraft dispatchers, maintenance operators, and other operational personnel, monitor operations and manage these unplanned situations by implementing control measures [54].

Jeppesen Systems AB is developing, marketing, and providing airline and railway software solutions. The solutions of the company consist of research-based applications combined with services such as analyzing, stimulating, developing, and supporting business processes of customers. The services of the company include airspace solutions, crew solutions, data management solutions, flight and fuel optimization, flight planning and dispatch, navigation solutions overview, network and operations management, training, and pilot supplies. In Jeppesen, the planning process for aircraft and crew usually begins months before the day of operations (Figure 1.1). The scheduling is divided primarily into three stages, including crew planning, fleet planning, and maintenance planning. Except for crew tracking, fleet tracking, and

tail allocation, everything is scheduled days-weeks-months before the day of service.

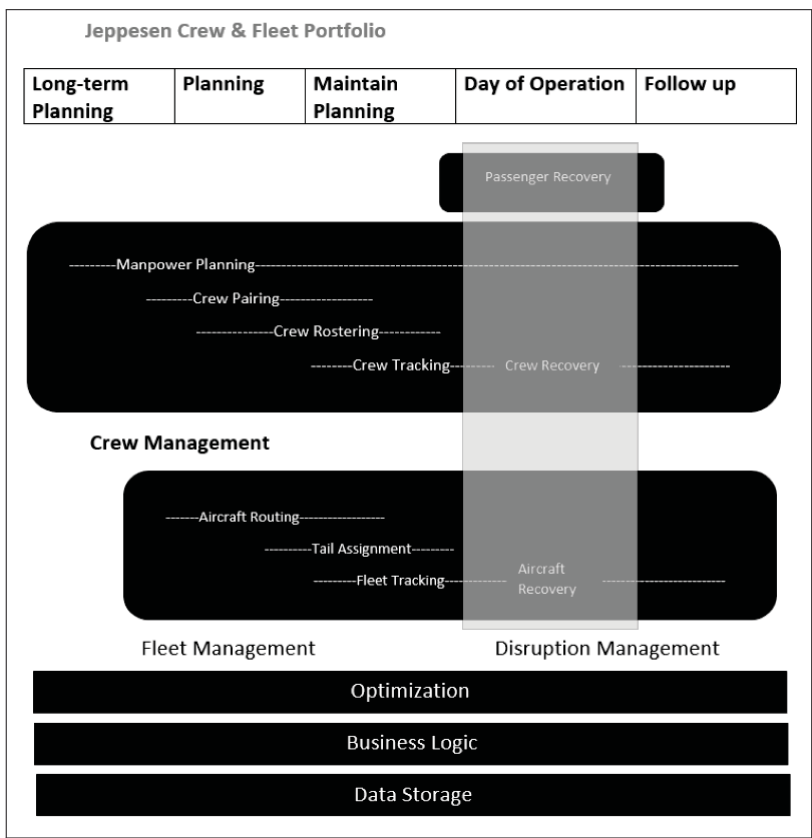


Figure 1.1: Crew and Portfolio

The Jeppesen airline system (Figure 1.2) has the specifics of the airline’s ongoing schedule. The colors in Figure 1.2 show different activities of aircraft, green means scheduled and on-time, whereas red means delayed or canceled, and gray means under maintenance. In the case of disruptions, the schedule recovery can be made in this Jeppesen airline system as it contains the ongoing schedule. Jeppesen has built a disruption management system to restore the plan, known as a recovery solver (Figure 1.2), that creates a set of solutions in which a solution can be used to resolve a disruption. On the day of service, the recovery solver comes into play to address the disruption. The operators of the Jeppesen airline system can either choose a solution created by the recovery solver to recover the service for a specific disturbance; the operator can also manually create a solution or a combination of manual solution and recovery solution.

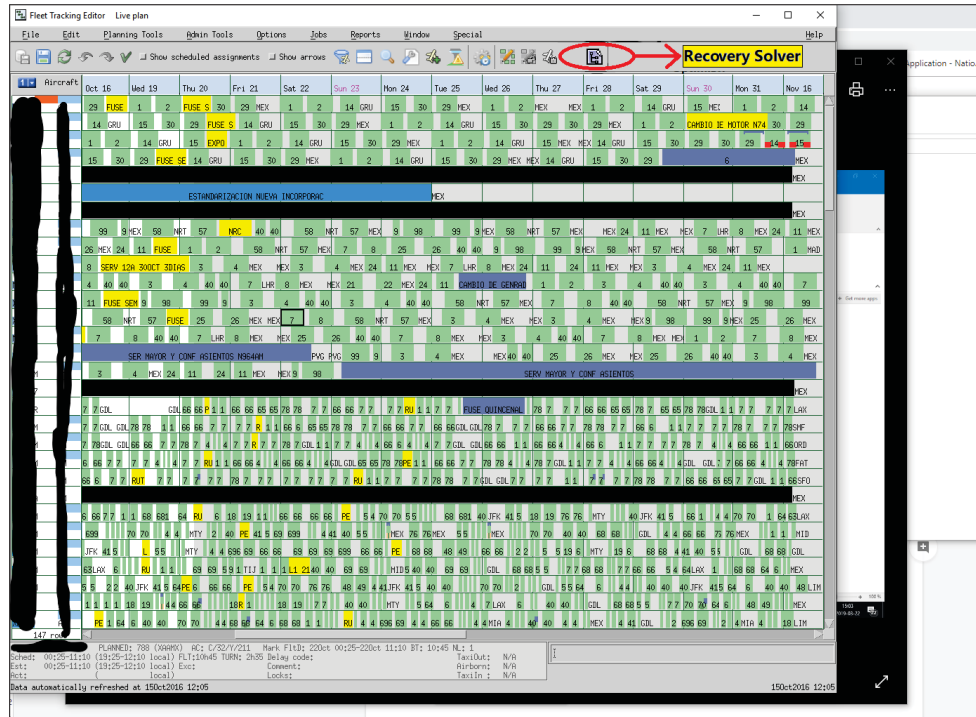


Figure 1.2: Jeppesen airline system

1.1 Aim and Objectives

The thesis focuses on transforming the data obtained from the Jeppesen airline system operators and further defining the suitable machine learning algorithms suitable for problem formulation by conducting a literature review. Using features extracted from the data, several models should be trained and compared to determine which one is best able to answer the question of whether the operator will use the recovery solver or not? Furthermore, if the operator chooses to use the recovery solver, which of the offered solutions will he pick?

Objectives

- To identify suitable machine learning algorithms that can predict which action the operator will choose.
- To evaluate the performance of the selected algorithms and comparing the performance of the selected algorithms to figure out which algorithm for the given problem statement is the best.

1.2 Research Questions:

RQ 1: What are the suitable machine learning techniques for predicting the operator's choice in the case of airline disruptions?

Motivation: The motivation for this research question is to find and gain knowledge of the previous implementation of the problem statement. It is also important

to know which algorithms can be applied to the given data, to find the appropriate algorithms, and to perform experiments to find out which algorithms can give us better results.

RQ 2: What are the performances of machine learning models in the prediction of the operator's selection based on the previous recorded data?

Motivation: The motivation for this research question is to conduct an experiment and use performance metrics to introduce and assess the performance of selected machine learning algorithms in the prediction of the operator's selection and discuss about the performance of the best machine learning model compared to the other selected models.

A literature review is chosen to answer RQ1. Since it can help to gain more information about the research background, it is also useful to review various previously applied machine learning models for similar research and to define suitable machine learning models accordingly

An experiment is chosen to address the RQ2, Models are selected based on a literature review, and an experiment defines the performance of selected models. A comparison is made between the results of the selected machine learning models as the purpose of this research is to find a reliable model. The model with the best performance is chosen and analyzed in comparison of other selected models.

1.3 Problem Statement

For a given disruption, the operator uses the Jeppesen airline system to provide a solution to the problem. The operator can either choose between the recovery solver, manually construct a solution or use a solution that blends manual solution with recovery solver. Resolving disruptions involve combining any number of three possible actions: delaying flights, canceling flights, or finding a replacement plane (also called "swap").

The question here is to determine, "Will the operator use the solver" and "If the operator uses the solver, which solution will, he prefers." Such knowledge could then be used to guide the recovery solver system to generate the type of solutions desired by the operator to boost schedule recovery operations.

1.4 Structure of the thesis

This thesis structure is divided into seven different chapters which are as follows:

- Chapter 1: contains the introduction to this thesis, problem statement, aim and objectives, research questions, and motivation.
- Chapter 2: contains the background of the concepts, used during the research.

- Chapter 3: contains the summary of the works similar to this thesis.
- Chapter 4: contains methods to answer research questions. It includes experimental analysis like data processing, tools used during the experiment, and experimental setup details.
- Chapter 5: contains results obtained from the experiment.
- Chapter 6: consists of analysis and discussions about the results and methods, the contribution of the thesis to existing research, threats to validity and limitations of the thesis.
- Chapter 7: contains the conclusion of the thesis and discussion on possible future work.

This chapter presents the fundamental concepts behind the models used in the research.

2.1 Machine Learning

Machine Learning (ML) is an artificial intelligence (AI) technique that provides systems with the ability to learn from experience and improve without explicit programming. Machine learning generally focuses on creating computer programs that can access data and use it to learn on their own. Usually, the process starts with observations or data, such as examples, instructions, etcetera. To search for trends in data to make better decisions in the future based on the examples we provide. The primary objective is to allow the systems to learn without human interference or help automatically and to adapt actions accordingly. The stages while working with machine learning involve- data gathering, preparing the data, choosing a model, training, evaluation, hyperparameter tuning, and prediction.

Machine Learning offers several different approaches to learning from given information. It usually depends on the output predicted, and the form of input received. ML algorithms are classified according to their learning style. The style is dependent on the type of data available and the expected result. ML techniques can be classified to four types which include the following techniques [52], [30], [19]:

- Supervised machine learning: Finds patterns to develop predictive models using input and output data.
- Unsupervised machine learning: Derives patterns from a given dataset without any reference to known, or labeled outcomes. Unsupervised learning can be considered as detecting patterns in the data up and beyond and is appropriate when one does not have information on desired outcomes [23].
- Semi-supervised machine learning: Lies between supervised and unsupervised machine learning techniques were during the training phase; it uses labeled and unlabeled data. This takes a smaller amount of labeled data and a larger amount of unlabeled data into consideration to build better classifiers [65].
- Reinforcement machine learning: Taking appropriate action in a particular situation to maximize reward. It is adapted by different software and machines

to find the best possible behavior or path it should take in a specific situation [31].

2.2 Supervised Learning

The search for algorithms that reason from externally provided examples to provide general hypotheses is known as supervised machine learning, which then makes predictions about future examples [37]. Supervised machine learning algorithms generally discover insights, relationships, and patterns from a labeled training dataset (i.e., the dataset that already consists of a known target variable value for each record variable). The machine-learning algorithm has the correct answers to a problem during the training phase. In this way, the machine learning algorithm learns how the other features contribute to the goal, allowing insights to uncover, and accurate predictions of future events based on historical data. Types of supervised learning techniques are as follows:

- Regression: The algorithm returns for each case a statistical goal, such as how much revenue a new marketing strategy would produce. Regression deals with discrete output.
- Classification: The algorithm tries to label each of the examples by choosing between two or more different classes. Deciding between two categories is called binary identification, like finding out if someone is going to default on a loan or not. Choosing between more than two classes is known as multi-class classification.

2.3 Ensemble Learning

Ensemble learning is a method of learning algorithms that assembles a set of classifiers and classifies new data points by considering a weighted vote of their predictions [18], [17]. An ensemble contains a set of independently trained classifiers whose predictions are combined during classifying different instances. In simpler words, the ensemble is the art of combining a disparate group of learners or individual models to increase the stability and predictive power of the model [43]. Some commonly used ensemble learning techniques are:

- Bagging: Bagging(Bootstrap Aggregating) is an ensemble technique that creates individuals for its ensemble. It is done by training the classifiers (separately) on a randomly sampled subset of the training set. In simpler words, it trains multiple models on different samples and then takes a mean of all the predictions [9], [20]. It is possible to use different learners on a different population, and this helps in the reduction of the variance error. Bagging techniques help when applied to an over-fitted base model.
- Boosting: Boosting is a general method for improving the performance of weak learning algorithms. It significantly reduces the error of any weak learning algorithms [22]. Incorrectly classified observations are given increased weight

in subsequent training iterations. Boosting, in general, reduces variance and also eliminates the effect of high bias of the weak learner.

- **Stacking:** This is an exciting way of merging models. The learner combines output from different learners. This can lead to a decrease in either bias or variance error, depending on the combining learner used.

2.3.1 XGBoost(Extreme Gradient Boosting)

XGBoost refers to Extreme Gradient Boosting. XGBoost is also a method to push the limit of computations resources for boosted tree algorithms. It is a scalable tree boosting system that is widely used by data scientists and provides state-of-the-art results on many problems and it is used very widely [48], [13]. It is a decision tree-based ensemble machine learning algorithm that implements the gradient boosting framework. Boosting is a technique in which new models are added and trained on the examples misclassified by the existing models; these models are added sequentially till no further improvements can be made. Gradient boosting is a way where new models are designed that predict the residuals or errors of previous models and then combined collectively to gain the final prediction. It is termed gradient boosting because it uses a gradient descent algorithm to decrease the loss when combining new models. This procedure supports both regression and classification predictive modeling problems [48]. XGBoost is composed of several base learners.

$$F = \{f_1, f_2, f_3, f_4, \dots, f_m\} \quad (2.1)$$

$$Final\ Prediction : \hat{y}_i = \sum_{t=1}^m f_t(x_i) \quad (2.2)$$

$$O = \{x_1, x_2, x_3, x_4, \dots, x_n\} \quad (2.3)$$

$$L^{<t>} = \sum_{i=1}^n l(y_i, \hat{y}_i^{<t-1>} + f_t(x_i)) + \Omega(f_t) \quad (2.4)$$

In the gradient boosting algorithm stated above, F in Equation 2.1 is the set of base learners. Functions that minimize the overall loss are chosen for each iteration. In Equation 2.4, l is the loss term and Ω is the regularization term. The $f_t(x_i)$ is obtained at each iteration by fitting a base learner to the negative gradient of the loss function concerning previous iteration's value. In XGBoost, we explore several base learners or functions and pick a function that minimizes the loss.

2.4 Neural Networks

Neural Networks (NN) have been used to structure the functionality of the biological nature of the human brain [1]. NNs are computational modeling tools that have emerged and are found to be accepted in many disciplines for real-world modeling problems. NN is a collection of interconnected processing elements known as artificial nodes or neurons, which are modeled to reproduce the flexibility and power of

a biological brain that is capable of performing massively parallel computations for data processing and representation of knowledge by artificial means [7], [53]. The most popular NN is the feedforward network, described as a directed graph in which each node implements a transfer function of the form [51]

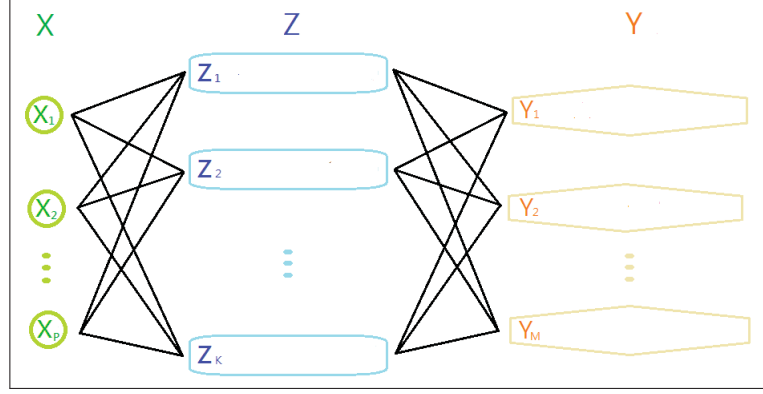


Figure 2.1: Neural Networks

$$\text{Input layer(layer1)} : X = (X_1, X_2, \dots, X_P) \quad (2.5)$$

$$\text{Middle layer(layer2)} : Z = (Z_1, Z_2, \dots, Z_K) \quad (2.6)$$

$$\text{Output layer(layer3)} : Y = (Y_1, Y_2, \dots, Y_M) \quad (2.7)$$

Figure 2.1 shows a schematic view of a neural network. The first layer (X) is the input layer, which receives input values and generates graded output values. These output values act as input to the next layer, which is considered to be a hidden layer that has neurons(nodes) that apply different transformations to the input data. Finally, the output from the hidden layer acts as an input to the final layer of the network, i.e., the output layer. The output layer is the last in the network, which receives input from the hidden layer. With this layer, the desired number of values and in the desired range is obtained. It consists of the same number of neurons as that of the number of classes. Thus, each output neuron represents a class while mutually contributing to the final decision of the network. The weights represent the strength of the connection between units. If the weight from the first node to the second node has a bigger magnitude, it means the first node has a more significant influence over the second node. Weights near-zero mean that changing this input will not change the output. Negative weights mean increasing this input will decrease the output. A weight decides how much influence the input will have on the output.

$$\text{Middle Layer Neuron} : Z_k = \sigma_k(\alpha_{0k} + \alpha_k^T * X), \forall k = 1, 2, \dots, K \quad (2.8)$$

$$\alpha_k = (\alpha_{k1}, \alpha_{k2}, \dots, \alpha_{kp})^T \quad (2.9)$$

$$\text{Output Layer Neuron} : Y_m = G_m[Z] = \beta_0 + \beta_m^T * Z, \forall m = 1, 2, \dots, M \quad (2.10)$$

$$\beta_m = (\beta_{m1}, \beta_{m2}, \dots, \beta_{mk})^T \quad (2.11)$$

Here 2.9 and 2.11 are vectors of regression coefficient. NNs, depending on their connectivity, can be divided into feed-forward and recurrent groups. A NN is referred to as feed-forward if there is a mechanism that numbers all nodes in the network so that there is no relation between a large number of nodes and a smaller number of nodes. All links range from small numbered nodes to larger numbered nodes. A NN is defined as recurrent if there is no such method of counting. A NN's architecture can be determined by its topological structure, i.e., each node's overall connectivity and transfer function in the network [29].

2.5 Activation Function(Transfer Function)

Activation Function is used to introduce non-linearity to neural networks. Activation functions sometimes scale the value to a smaller, pre-determined range, usually $[0,1]$ or $[-1,1]$, but there are also transfer functions that are not limited, for example, ReLu. Basically, neural networks are universal function approximators, and deep neural networks are trained using backpropagation, which requires differentiable activation functions. Backpropagation uses gradient descent on this function to update the network weights.

2.5.1 Sigmoid or Logistic activation function(Soft Step)

It is one of the most popular and widely used activation function for binary classification problems (i.e., outputs values that range 0–1). The only problem with the sigmoid activation function is of vanishing gradients. This function is more prone to saturation of the later layers, making training more difficult. The calculation of the derivative of the sigmoid function is very easy.

$$\sigma_k(x) = \frac{1}{1 + e^{-x}} \quad (2.12)$$

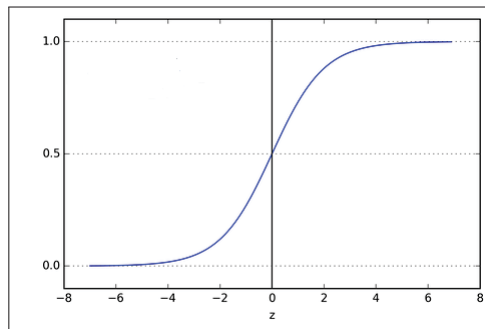


Figure 2.2: Sigmoid

2.5.2 Hyperbolic tangent (TanH)

It looks similar to a scaled sigmoid function. Data is concentrated around zero, so the derivatives will be higher. Tanh converges more quickly than sigmoid and logistic activation functions.

$$\tanh(x) = \frac{e^x - e^{-x}}{e^x + e^{-x}} \quad (2.13)$$

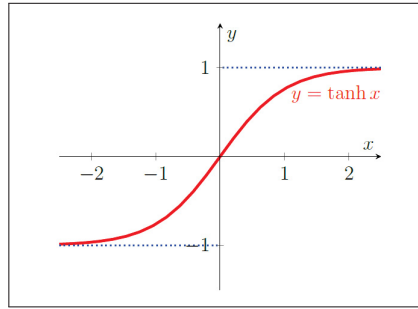


Figure 2.3: Hyperbolic Tangent(TanH)

2.5.3 Softmax

Softmax converts a vector of raw values into a vector of probabilities and provides a measure of certainty. It compresses the outputs between 0 and 1 similar to a sigmoid function; each output is divided in such that the total sum of the outputs is equal to 1.

$$f_j(z) = \frac{e^j z}{\sum_k e^{z_k}} \quad (2.14)$$

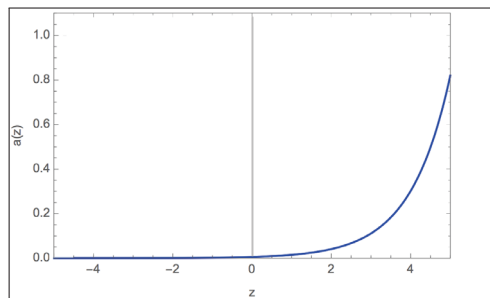


Figure 2.4: Softmax

The output of the softmax function is similar to a categorical probability distribution; it tells you the probability that any of the classes are true.

2.5.4 Rectified Linear Unit(ReLU)

A rectifier linear unit is one of the most common activation functions in neural network models. It trains six times faster than tanh. The output value will be zero when the input value is less than zero. If the input is greater than or equal to zero,

the output is equal to the input. When the input value is positive, the derivative is 1; hence, there will be no squeezing effect that occurs in the case of backpropagating errors from the sigmoid function.

$$\text{relu} = \max(0, x) \quad (2.15)$$

$$\text{if } x > 0, \frac{\partial(\text{relu})}{\partial x} = 1 \quad (2.16)$$

$$\text{otherwise, } \frac{\partial(\text{relu})}{\partial x} = 0 \quad (2.17)$$

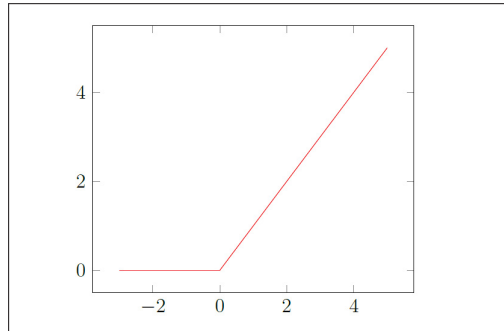


Figure 2.5: Rectified Linear Unit(ReLU)

ReLUs are faster to train faster when compared to sigmoid or tanH, and constant gradient achieves faster learning in ReLUs.

2.6 Deep Learning

Deep learning is artificial intelligence (AI) techniques that mimic the brain in data processing and creating patterns. Deep learning has gained much recognition due to the great advancements it has contributed to, particularly in computer vision. Deep learning procedures can classify, recognize, identify, and detail. Deep learning is used for image classification, speech recognition, and object detection. Various improvements are now advancing deep learning:

- Algorithmic improvements have boosted the performance of deep learning methods.
- New machine learning approaches have improved the accuracy of models.
- New classes of neural networks have been developed that fit well for applications like text translation and image classification.
- Deep learning is capable of learning from data that is both unstructured and unlabeled. Large amounts of data are available to develop neural networks with many deep layers.

Typically such deep learning approaches adopt the structure of the neural network, and deep learning algorithms are often called deep neural networks. Deep learning is used primarily in autonomous driving, medical research, aviation, and telecommunications.

2.7 Forward Propagation

Forward propagation is a process of feeding the network input values and producing an output that we call the predicted value. We often refer to forward propagation as inference. The first layer does no operations on the input. The next (second) layer takes values from the first layer and applies operations of multiplication, addition, and activation, moving this value to the next layer. For subsequent layers, the same process repeats, and finally, we get an output value from the last layer.

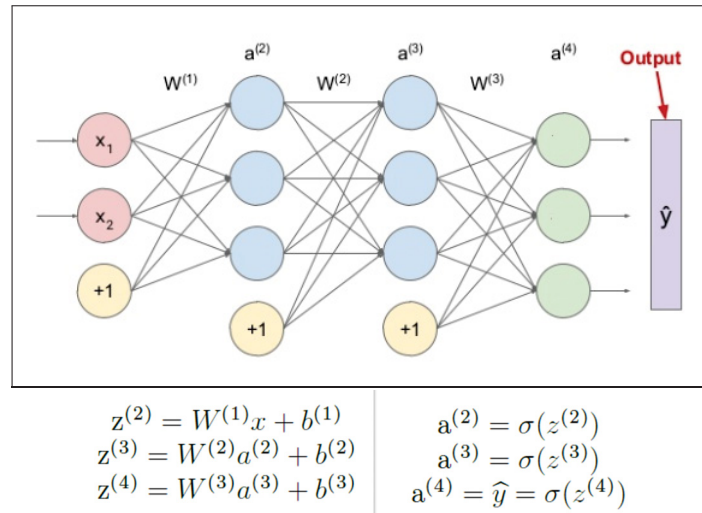


Figure 2.6: Forward Propagation

As shown in Figure 2.6, x_l is the input value, $W^{(l)}$ is the weight matrix for layer l , $a^{(l)}$ is the activation vector for layer l , $z^{(l)}$ is the input into layer l and $+1$ is the bias (constant) unit. Forward propagation works in the following way. As shown in Figure 2.6, the input vector is passed into the network. It is multiplied with $W^{(1)}$ weight matrix and added with layer one biases to calculate $z^{(2)}$. Activation value for the second layer is calculated by passing $z^{(2)}$ into some function let's assume sigmoid function. $z^{(3)}$ is calculated by multiplying $a^{(2)}$ vector with $W^{(2)}$ weight matrix and adding layer two biases. Similar to previous layer, $a^{(3)}$ is calculated by passing $z^{(3)}$ into the sigmoid function $z^{(4)}$ is calculated by multiplying $a^{(3)}$ vector with $W^{(2)}$ weight matrix and adding layer three biases. For the final layer, we calculate $a^{(4)}$ by passing $z^{(4)}$ into the activation function. We then make our prediction based on the final layer's output.

2.8 Backpropagation

Backpropagation or the backward propagation of the errors is a method to adjust the weights based on the output. Rummelhart et al. [49] devised the backpropagation algorithm for training, and that is how it became the basic concept behind the learning of a neural network. Backpropagation algorithm was actually discovered years earlier by a Finnish master's student named Seppo Linnainmaa [39]. The training

process adjusts the connection weight and bias of the network to minimize the error function. To calculate the error, we compare the predicted value with the actual output value. A loss function is used to calculate the error value. The derivative of the error value is calculated concerning every weight in the neural network. This error is then backpropagated to the previous layer, and all the weights are adjusted accordingly.

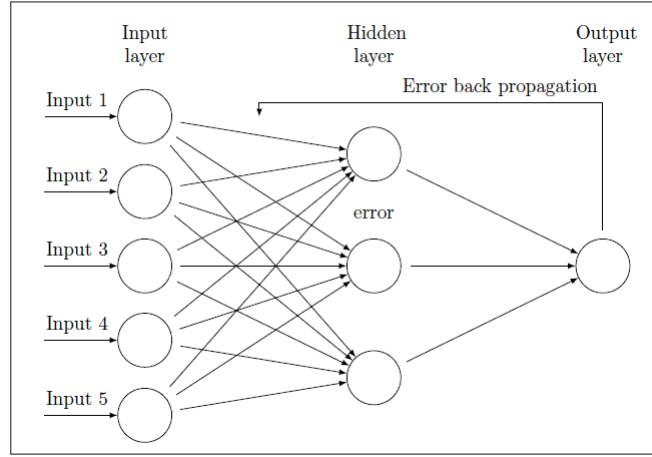


Figure 2.7: Backpropagation

The equation of backpropagation are as follows:

$$dZ^{[2]} = A^{[2]} - y \quad (2.18)$$

$$dW^{[2]} = \frac{1}{m} dZ^{[2]} A^{[1]T} \quad (2.19)$$

$$dZ^{[1]} = W^{[2]T} dZ^{[2]} g^{[1]'}(Z^{[1]}) \quad (2.20)$$

$$dW^{[1]} = \frac{1}{m} dZ^{[1]} X^{[T]} \quad (2.21)$$

Where,

- $A^{[i]}$ is the activation values for the i^{th} layer.
- y is the target value.
- $Z^{[i]}$ is the input for the i^{th} layer.
- $W^{[i]}$ is the weight between the i^{th} layer and the $(i - 1)^{th}$ layer.
- $g^{[i]}()$ is the activation function for the i^{th} layer.
- X is the input to the neural network.

Generally, in chain rule, first, we calculate the derivatives of error value concerning the weight values of the last layer. These are called derivatives or gradients and use these gradient values to calculate the gradients. This process is repeated until we get all the gradients for every weight in our neural network. Gradient values are subtracted from the weight value to reduce the error value. Thus, this moves descent closer to the local minima, which is also known as minimum loss.

2.9 Definitions

Here are a few terms which are used while discussing neural networks. These terms are defined as the following:

2.9.1 Learning Rate:

It determines how quickly or how slowly the weights(parameter) are updated. The learning rate should be low enough such that it can find its local minima, and it also affects how quickly our model can arrive at the best accuracy (local minima). The lower the value, the slower we travel along the downward slope.

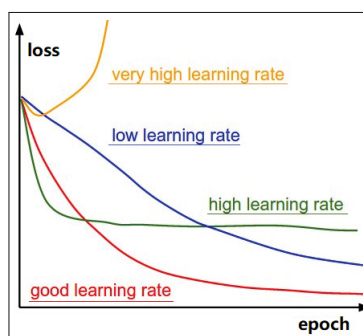


Figure 2.8: Learning rate

2.9.2 Batch:

As the entire dataset cannot be feed into the network at once, the dataset is split into sets or parts known as batches.

- Batch Size: The number of training examples in one forward/backward pass. The larger the batch size, the more memory space will be needed.

2.9.3 Training Epochs:

It is the number of times that the model is revealed to the training dataset.

- One epoch = one forward pass and one backward pass of all the training examples.

2.9.4 Convergence:

Convergence is when series $s(n) = loss_{w_n}(\hat{y}, y)$ (Where w_n is the set of weights after the n^{th} iteration of back-propagation and $s(n)$ is the n^{th} term of the series) is a converging series. As the iterations advance, the output gets closer and closer to the target value.

2.9.5 Overfitting

Overfitting is a scenario where the inclusive cost is minimal, but the performance becomes unreliable. An overfitted model is when we train it with a large amount of data, it learns from the noise and incorrect data entries in our dataset. The data is not categorized in the right way in the model because of too many details. Non-parametric, non-linear processes cause overfitting because specific machine learning algorithms have freedom in creating the model based on the dataset, and therefore they can build unrealistic models. Detecting overfitting is necessary because it is unknown how well a model will perform on new data until it is tested. The following are a few ways to overcome overfitting.

- Cross-validation: This is an effective and trendy preventative solution for overfitting. Here, the initial training data is used to generate multiple mini train-test splits.
- Remove features: Generalizability can be improved by removing irrelevant input features.
- Early stopping: Early stopping points to ending the training method before the learner reaches that point.
- Regularization: Regularization applies to an extensive range of methods for artificially limiting the model to be more straightforward. In other words, it is possible to measure how well each iteration of the model performs during the training phase. Iterations improve the model up to a particular stage, and after that point, the model's generalization ability weakens, leading to overfitting, and this is where regularization comes handy.

2.9.6 Cross Validation

Cross-validation is a statistical method involving partitioning the data into subsets, training the data on a subset and using the other subset to determine the efficiency of the model. To minimize uncertainty we perform multiple rounds of cross-validation from the same data with different subsets. We combine the validation results from these multiple rounds to arrive at a predictive performance estimate for the model. Cross-validation gives us a more precise estimate of the performance of a model. The following are few popular techniques in cross-validation.

- K Fold: This technique involves randomly dividing the dataset into k groups or folds of approximately equal size. The first fold is kept for testing and the model is trained on k-1 folds. The process is repeated K times and each time different fold or a different group of data points are used for validation. Generally, KFold will provide train/test indices to split data in train and test sets. It will split dataset into k consecutive folds (without shuffling by default). Each fold is then used a validation set once while the k - 1 remaining folds form the training set.
- Stratified cross-validation: Stratification is a technique where we rearrange the data in a way that each fold has a good representation of the whole dataset.

It forces each fold to have at least m instances of each class. This approach ensures that one class of data is not over represented especially when the target variable is unbalanced. The Stratified cross validation shuffles the data, after the shuffling it splits the data into n_splits parts and done. Now, it will use each part as a test set. Important thing to be noted is that stratified cross validation only and always shuffles data one time before splitting.

- Leave one out cross-validation (LOOCV): In LOOCV we divide the data set into two parts. In one part we have a single observation, which is our test data and in the other part, we have all the other observations from the dataset forming our training data. Leave one out cross-validation is essentially an estimate of the generalisation performance of a model trained on $n-1$ samples of data, which is generally a slightly pessimistic estimate of the performance of a model trained on n samples.
- Nested cross-validation: Nested cross-validation is an extension of the K-fold cross validation and it fixes one of the problems that are caused when normal cross-validation are considered. Generally, in normal cross-validation best hyper-parameters are found using training and testing set which might cause leakage of information. The nested CV has an inner loop CV nested in an outer loop CV. The inner loop is responsible for model selection/hyperparameter tuning (similar to validation set), while the outer loop is for error estimation (test set). In nested cross validation, firstly the desired parameters are set to tune to to some value and the dataset is split into k 'folds' (sections) and then train the model using $k-1$ folds using the parameter value and the model is tested on the remaining fold. This process is repeated until every fold is considered as test data once and also this process is repeated for every possible value of the parameter and finally the parameters with the best results are produced.

2.9.7 Normalization:

Data normalization is a perfect example of pre-processing data to remove or reduce the burden of machine learning (ML) to learn certain invariants, i.e., things that do not change the meaning of the symbol but only change the representation. Normalization is an excellent technique to use when the distribution of the data is unknown or when the distribution is not Gaussian (a bell curve). It is good to speed up the learning process. There are different types of normalization.

- Z Normalization (Standardization): This transformation sets the mean of data to 0 and the standard deviation to 1. Standardization is used feature-wise.

$$\hat{X}[:, i] = \frac{X[:, i] - \mu_i}{\sigma_i}, (\mu_i = \frac{1}{N} * \sum_{k=1}^N X[k, i], \sigma_i = \sqrt{\frac{1}{N-1} * \sum_{k=1}^N (X[k, i] - \mu_i)^2}) \quad (2.22)$$

- Min-Max Normalization: The data is rescaled to the range $[0,1]$.

$$\hat{X}[:, i] = \frac{X[:, i] - \min(X[:, i])}{\max(X[:, i]) - \min(X[:, i])} \quad (2.23)$$

- Unit Vector Normalization: Scaling to unit portion shrinks or expands a vector to a unit sphere as the row of data can be viewed as a D-dimensional vector. When applied to the whole dataset, the converted data can be visualized as a collection of vectors with various positions on the D-dimensional unit sphere.

$$\hat{X}[j,:] = \frac{X[j,:]}{\|X[j,:]\|} \quad (2.24)$$

In the Equations 2.22, 2.23, 2.24: X is the dataset, N is the number of entries, X[:,i] represent feature i and X[j,:] represent entry j.

2.9.8 Loss Function/Cost Function:

The loss function computes the error for a single training example. The cost function is the average of the loss functions of the entire training set.

- MSE: For mean squared error.
- Binary crossentropy: For binary logarithmic loss (logloss).
- Categorical crossentropy: For multi-class logarithmic loss (logloss).

2.9.9 Training Algorithms

The training algorithms are a search technique, which is used to update weights in the model.

- SGD: Stochastic Gradient Descent, with support for momentum.
- RMSprop: Adaptive learning rate optimization method proposed by Geoff Hinton.
- Adam: Adaptive Moment Estimation (Adam) that also uses adaptive learning rates.
- Lbfgs: Lbfgs is an optimizer in the family of quasi-Newton methods.

2.10 Probabilistic Neural Network

The Probabilistic Neural Network (PNN) is closely related to the Parzen window pdf estimator. It can compute nonlinear decision boundaries [55], and training is instantaneous. The decision surfaces can be made as complex as necessary and also as simple as desired. A PNN consists of four layers:

- Input layer: The input nodes are the set of measurements, and Each neuron in the input layer represents a predictor variable. Input neurons feed the values to each of the neurons present in the hidden layer.
- Pattern layer: The second layer consists of the Gaussian functions formed using the given set of data points as centers. This layer comprises one neuron for each case in the training data set. It saves the values of the predictor variables for the problem, along with the target value.

- Summation layer: The third layer performs a regular operation of the outputs from the second layer for each class.
- Output layer: The fourth layer performs a vote, selecting the most significant value. The associated class label is then determined.

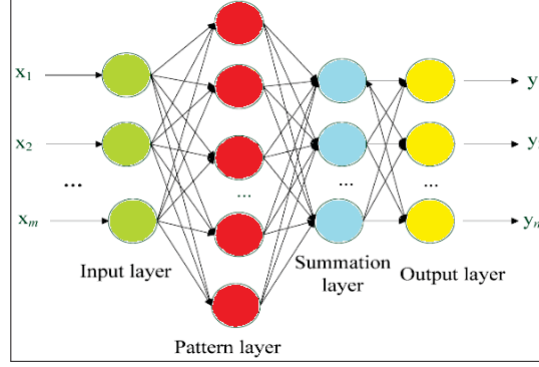


Figure 2.9: Probabilistic Neural Network structure

The output of each unit in the pattern layer is calculated by the following equation.

$$\Phi_{ij}(x) = \frac{1}{(2\pi)^{\frac{D}{2}} \sigma^D} \exp \left[-\frac{(x - x_{ij})^T (x - x_{ij})}{2\sigma^2} \right] \quad (2.25)$$

Here in Equation 2.25, $i = 1, \dots, M$, $j = 1, \dots, N$, M is the total number of classes in the training samples, x_{ij} is the j^{th} implicit center vector for the i^{th} mode, N_i is the number of neurons in the pattern layer of the PNN i^{th} class, σ is the smoothing parameter and d is the data dimension of the samples space. The summation layer is the cumulative probability belonging to a particular class, which is calculated as follows.

$$f_{iN_i}(x) = \frac{1}{N_i} \sum_{j=1}^{N_i} \Phi_{ij}(x) \quad (2.26)$$

All probability density functions in the summation layer are the input of the output layer neurons, which can be described as follows.

$$\rho(x) = \operatorname{argmax} [\alpha_i f_{iN_i}(x)] \quad (2.27)$$

Where α_1 is the prior probability of class i , and $\rho(x)$ is the estimated class obtained by PNN.

2.11 Multilayer Perceptron

A perceptron is an algorithm intended for binary classification, i.e., whether an input belongs to specific class $Y = 0$ or $Y = 1$. The Multilayer Perceptron (MLP) is a class of artificial neural network consists of one or more perceptron. MLP consists of an input layer, at least one arbitrary hidden layer and one output layer. MLP [5] is one

of the commonly used classifiers, and it uses backpropagation technique, an iterative method in which the calculated weights (from input layer through hidden layers to output layer) are measured against ground truth, and the weights are updated to improve the network until it can make its best prediction similar to ground truth.

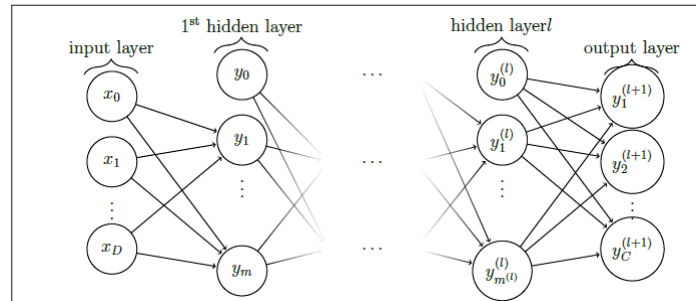


Figure 2.10: Multilayer Perceptron with 'l' hidden layers

The activation functions for MLP used to be sigmoid functions (Section 2.5.1), but most of the time tanh (Section 2.5.2) can converge more quickly than the sigmoid function and gives better accuracy. However, ReLU (Section 2.5.4) train faster than tanh to reach the same training error.

2.12 Support Vector Machine (SVM)

A Support Vector Machine (SVM) is one of the oldest and most popular machines learning algorithm and was introduced in [16], [24], that analyzes data for classification and regression analysis. SVM is a supervised learning approach that studies the data and classifies it into one of two divisions. An SVM outputs a map of the distributed data with the boundaries within the two. SVMs are applied in text categorization, image classification, handwriting recognition, etcetera. A support vector machine is also known as a support vector network.

SVMs are based on the concept of discovering a hyperplane that best separates a dataset into two classes, as shown in the image below.

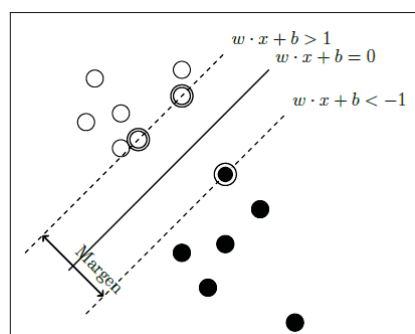


Figure 2.11: Support Vector Machine

Support vectors are the data points closest to the hyperplane, the points of a dataset

that, if excluded, would alter the state of the dividing hyperplane. Because of this, they are recognized as the critical elements of the dataset.

2.13 Naive Bayes Classifier

A Naive Bayes classifier is a simple and widely used classifier in the field of data classification. It is one of the most efficient and effective inductive learning algorithms for machine learning and data mining. Naive Bayes classifier comes under Probabilistic classifiers, which generally uses mixture models to classify data. Naive Bayes is not an individual algorithm but a family of algorithms where all of them share a universal principle. Naive Bayes's methods are a set of supervised learning algorithms based on applying Bayes's Theorem with the "naive" presumption of limited freedom between every pair of features given the value of the class variable. Its competitive performance in classification is surprising, because the conditional independence assumption on which it is based, is rarely true in real-world applications [64]. Bayes theorem presents a method to compute the posterior probability, $P(c|x)$, from $P(c)$, $P(x)$, and $P(x|c)$. The classifier implies that the outcome of the value of a predictor (x) for a given class (c) is independent when compared to the values of other predictors. This presumption is called class conditional independence.

$$P(c|X) = \frac{P(x|c)P(c)}{P(x)} \quad (2.28)$$

$$P(c|X) = P(x_1|c) \times P(x_2|c) \times \dots \times P(x_n|c) \times P(c) \quad (2.29)$$

- $P(c|x)$ is the posterior probability of class (target) given predictor (attribute).
- $P(x|c)$ is the likelihood which is the probability of predictor given class.
- $P(x)$ is the prior probability of predictor.
- $P(c)$ is the prior probability of class.

The research on recovery operations problems started way back in the 1980's [58], the authors' goal was to minimize the total passenger delays on an airline network when one or more aircraft are unavailable. The earliest operation research simulated the manual approach that airline operators use, splitting the recovery of aircraft, crew, and passengers into separate problems and solving them sequentially. By implementing this, many problems were easily managed, but some problems could not be solved in a reasonable time. However, the research was further extended in 1990 [59] and 1995 [60] by Teodorovic and Stojkovic where new factors like airport curfews and crew considerations were added, and the proposed model had a greedy approach and crew before aircraft methodology respectively. These researches were the first developed on aircraft recoveries To operate the daily operations of airlines, a real-time decision support system was developed [46], When it was implemented in 1992 it was a game-changer, and it was later extended to deal with flight cancellations. This research mainly focused on managing airline operations and gave pretty detailed insights on how the changes in the schedule impacted the possibility of huge savings. However, starting with [35], the authors have presented a detailed and structured introduction about airline disruption management and schedule planning in the airline industry. The manual methods of dealing with disruptions and recovery are also discussed. This research gives us a basic idea and introduction to airline disruption management, which includes functions of OCC(Organisation of operations control), decision making, disruption management process, and The Descartes project, which is short for "decision support for integrated crew and aircraft recovery."

In [14], the authors proposed a model which uses data mining and supervised machine learning algorithms to predict airline delays caused by inclement weather, and in this particular, the model was built on the previous data of weather and traffic data by using supervised machine learning algorithms which include random forests, Adaptive Boosting, k-Nearest-Neighbours, and decision trees. Based on receiver operating characteristics(ROC) Moreover, an individual's algorithm's prediction accuracy is measured. Due to unbalanced dataset, sampling technique (SMOTE combine with random under-sampling) was applied and the experiment was carried out and it was observed that random forest performed well during the testing when compared to the other approaches but it was believed that still possible approaches which can improve the model in the future. The author gives us better insights about delays and the implemented machine learning algorithms on predicting the delays, and tackling with unbalanced dataset but the research we are focusing on is related to the actions

done by the operator.

In [32], the authors have done a systematic literature review which includes the study done by different researchers in the disruption of airlines in the major airports of America, national airspace system of America [62], [41], [47], [4] and proposed methods such as Bayesian, decision trees, random forest and hybrid classification to predict the delay propagation of airlines especially for developing countries. The selected approaches were analyzed based on real datasets on the U.S. and Iranian airlines network. The results favored the hybrid model. The research generally focuses on a better understanding of different reviews of different researches. Another research done by Chen [12] for flight delay early warning, the research presents a flight delay early warning model based on fuzzy support vector machine with weighted margin. The comparison of OAO-SVM classifier and OAO-WMSVM classifier by their performances shows that the performance of OAO-WMSVM classifier are better fitted to the early warning model, and the difference between OAOSVM and OAO-WMSVM. In [6], the comparison between naive-bayes, SVM and random forests was done on predicting airline delays. The performance metrics considered were classification report and it was observed that SVM does a slightly better job in predicting delays.

Quansheng et al. [38] have done research on disrupted airline scheduling, and they focus on aircraft mechanical problems, severe weather, crew sickness, airport curfews, and security. The aim of the study is to minimize the cost during disruptions. Few things are ignored which are swapping of resources. It is believed to study the swapping resources intensively and believed to develop a system such that optimal decision is produced. The research focuses on additional factors of airline disruptions and how they are handled.

Balasubramanian Thiagarajan et al. [61] have developed a two-staged predictive model employing supervised machine learning algorithms for the prediction of flight on-time performance, i.e., it was built for predicting the arrival and departure delays of flights. The dataset, which was considered in developing the model, consisted of the data of 15 major airports considering on-time performance and weather data of 5 years throughout USA and the features were extracted using recursive feature elimination. Due to unbalanced dataset the researchers used many sampling techniques and it was observed that Synthetic Minority Over-sampling Technique (SMOTE) produced the best results among the implemented other sampling techniques. It was observed that during the classification stage, the gradient boosting classifier performed the best with an accuracy of 86%. when compared to random forest, Adaboost and extra-trees. and in the regression stage, extra tree regressor performed the best. The research focuses on arrival and departure delays of airlines, which is different from our research, but it gives an idea about various machine learning algorithms, sampling techniques and feature selection method and their predictions during disruptions.

A study by Castro and Oliveira [10] focuses on an agent-based approach to help the Airline Operations Control Centre (AOCC) in solving the problems related to

disruption management. The author briefly details about tools and systems used in airline companies, the author also discusses their previous work in which he briefly discusses about Database Query system(DBQS) where the work is done by a human operator, Decision Support Systems(DSS) where it has some characteristics of the DBQS with additional functionalities to support the human operator on decision making. The Automatic or Semi-Automatic Systems(ASAS) which does not need human operators to operate correctly, unlike DSS and DBQS. The roles or functions related to operation monitoring, searching for solutions related to aircraft, crew or passenger problems and re-allocation of resources, are performed by specialist software agents replacing the human operators. The author also discusses their approach to disruption management, experimental setup, and results. This study helps in understanding the different approaches to deal with the disruptions, but our research is based on the operator's selection of solutions.

In [33], the authors have done a research for predicting flight delays. The research was done on the inbound flights of JFK airport in January 2012 were considered. The authors have introduced a new type of multilevel input layer ANN which can handle nominal variables. Further, the new approach was compared with traditional gradient descent back propagation ANN model on the basis of prediction error and training time. The proposed ANN model out performs the traditional approach and the proposed model is applied to a system of airport traffic control where the arriving flights are prioritized for landing based on the expected possible delays. RMSE was used as a performance metric. The results suggest that the proposed method can be effective for specific problems that include many nominal variables, such as the transportation problem.

Sridhar, Banavar, et al. [57] focuses on flight delays and cancellations based on the weather. This research compares the performance of traditional linear regression models with neural networks considering convective weather season (april-september) and non-convective season (october-march) for the period 2005-2008. The performance of the models were measured by the performance metrics mean absolute error and root mean squared error. The experiment was conducted on two operational databases, The multilayered neural networks were validated using five-fold cross-validation and after the experiment it was concluded that neural networks performed slightly better when compared to linear regression and multiple linear regression. Another research done by Henriques and Feiteira [26] on Hartsfield Jackson Atlanta international airport considering flight delays and associated factors. The goal of the research was to predict the occurrence of delays in arrivals of the flights in the airport. The dataset was unbalanced so the authors have used different sampling techniques like SMOTE and undersampling and the models selected for this research were multilayer perceptron, decision trees and random forest and the results showed that multilayer perceptron has outperformed when compared to other bagging models with an accuracy of 85% and another research [25], where different approaches were used in predicting delays in air traffic networks. Three classes of models were considered in this research which are markov jump linear system (MLJs), classification and regression trees (CART) and three candidate artificial neural network architectures namely MLP, GRNN and PNN. It is concluded that ANN architectures performs

better where MLP gives the highest accuracy (93%) among the other methods.

In the above-mentioned papers, various machine learning techniques like gradient boosting techniques, bagging techniques, neural networks and other machine learning techniques like Naive Bayes and support vector machines like have been implemented which have shown significant results on avoiding and overcoming several airline disruptions such as predicting delays and cancellations etcetra. However, choosing algorithms for a problem is not an insignificant decision. There is certainly no optimal algorithm that works for every problem but few algorithms are considered to perform better over others on specific problems. In the above cases algorithms like gradient boosting, multilayer perceptron, Probabilistic neural network and support vector machine have significantly performed well when compared to other machine learning techniques. As per the existing knowledge, there is not much evidence of research on the operator's choice. Therefore, this thesis will be using neural networks and other machine learning algorithms based on the literature review results to predict the choice of the operator of the recovery solver and evaluate the chosen algorithms.

4.1 Method

The research methods chosen to answer the research questions are:

- Literature review
- Experiment

4.2 Literature Review

Literature reviews are the primary basis for study in almost every research area. To answer the first research question, i.e., RQ1, the literature review was performed. As stated in Section 1.2, the literature review focuses on gaining knowledge about airline disruptions and understanding the working of various machine learning algorithms, as well as identifying the various appropriate machine learning algorithms that can be used for predictive purposes.

The steps performed in order to search the relevant sources are shown in Figure 4.1.

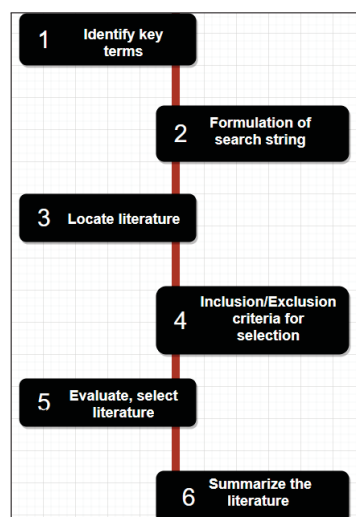


Figure 4.1: Steps performed during Literature Review

The following steps have been performed during the literature review:

- Before formulating the search string, the following keywords were identified. "Machine Learning," "Supervised Machine Learning," "neural network," "Prediction." "Airline Disruption," "Airline Schedule Recovery."newline
- Based on the above-listed keywords, primary keywords were selected to formulate the search string. The list of articles for the Literature Review is obtained by the search strings "Airline Schedule Recovery," "Airline Disruption using neural network," "Prediction of disruption Solutions using Machine Learning," "Airline Disruption Management" used to perform a search in different digital libraries.
- Inclusion and exclusion criteria are implemented after obtaining the articles, conference papers, journals to limit the results.

Inclusion Criteria

- Papers published over the past 25 years have been selected.
- Title and abstract of the papers should match with the problem domain.
- Articles in English.
- Papers related to Airline Disruptions, which also includes Machine Learning Techniques to solve them.

Exclusion Criteria

- Articles which are not in the field of Computer Science.
- Non-English language.
- Full text is unavailable.

4.3 Experiment

The experiment is selected to answer the RQ 2. The experiment is used as a research method because it is the best approach when dealing with quantitative data, and it is possible to achieve better results due to the controlled environment in an experiment. The experiment's primary goal is to implement the selected machine learning algorithms based on the results obtained from the literature review and then evaluate them to report the results.

4.4 Tools used

The following tools and software equipment have been used.

Hardware

- RAM: 32.0 GB
- Processor: Processor Intel(R) Core (TM)

- i7-8650U CPU @ 1.90GHz 2.11 GHz
- Graphic card: Intel UHD Graphics 620
- Disk size: 471 GB

Software

- Programming language: Python 3.7.2
- Operating System: Windows 10 (64- bit)

4.4.1 Setting up Software Environment

Python [63] is a simple, multi-purpose, high-level programming language. The elegant yet straightforward syntax of Python makes it one of the best programming languages for rapid development of applications. Python is widely regarded as a preferred language for implementing machine learning algorithms. Python is therefore used as a primary language for applying the algorithms of the neural network and also for the following reasons:

- Python is simpler to learn and code when compared to another programming languages like C, C++, and Java.
- Python has built-in libraries for easy implementation of algorithms.
- It is an open-source programming language.

In this experiment, the following python libraries are used to develop the machine learning models:

- Pandas: It is a python package that acts as a data analysis tool and deals with data structures. Pandas [40] carry out entire data analysis workflow in Python without having to switch to a more domain-specific language like R.
- Numpy: NumPy [42] is the basic package for computing with Python. It is used to add support to multi-dimensional arrays and matrices, with an extensive collection of high-level mathematical functions .
- Matplotlib: Matplotlib [28] python library, which generates plots, histograms, power spectra, bar charts, etcetera. In this work matplotlib.pyplot module is used to plot the metrics.
- YellowBrick: Yellowbrick [8] extends the Scikit-Learn API to make a model selection and hyperparameter tuning easier. Under the hood, it is using Matplotlib.
- Jupyter: Its an open-source web application allowing developers or researchers to share documents containing code, data visualizations, text, and equations. Jupyter [34] is generally used for data visualization, data cleaning, data transformation, machine learning, etcetera.
- SKLearn: The algorithms used in this experiment are imported from sklearn [45], in which we can directly take the classifiers, train, and test it on the datasets .

4.5 Data Collection

Data collection was the critical and time-consuming part of the research carried out. The data is spread over an SQL database, and periodic XML-formatted data dumps. Jeppesen has its data stored in a database on a different server. To access the database, the plan was to build and deploy a new system and redirect the new system to use the database. To establish the connection to the server, Putty was used as it is a secure shell/terminal emulator for connecting over a network. As mentioned earlier, the recorded data and the applications which use these databases are on two different servers named oracle node and main node. These nodes can be accessed using SSH.

The database includes tables with a record of aircraft-related activities and assignments. This research focuses mainly on extracting alerts from the database during the time of a disruption merged with the preventive method/solution created or taken by the operator for mitigating the disruption. The "DAVE_REVISION" database table is considered because it consists of "REVID" and "COMMITID" and it also connects COMMITID and REVID by which it is possible to know which specific database state corresponds to a given COMMITID.

- COMMITID: These are the identities of each and every action taken by the operator during a disruption.
- REVID: These are the unique identifiers of each and every change made to the database, including changes made by operators in response to a disruption.

Alerts were obtained from the moment an interruption occurred by channelizing the database for the Jeppesen airline system. The database was pruned and restored to the time when there were delays, and warnings were collected accordingly. For each REVID, this cycle has been repeated to get updates from 2017 to the latest data collected day.

By using the REVID and COMMITID we can link a certain database state to a certain operator action. The operator selection is denoted by 0-15, here 0 represents all manual solutions created by the operator, and the remaining options are chosen by the operation containing solutions provided during the disruption by the recovery solver. During ML training, the operator's selected choices are considered to be the target, and the information extracted from the database state is considered to be features.

4.6 Datsaset Used

The dataset used for this research comes from a major Mexico-based airline. The data covers parts of 2017, all of 2018, and parts of 2019. The dataset

consists of 20183 data points and 16 features. After merging features with the target as the operator's choice was collected considering REVID and COMMITID. Table 4.6 shows the feature names.

Feature Number	Feature Name
1	commitid
2	revid
3	alerts
4	softalerts
5	hardalerts
6	routeconstraints
7	buffer
8	inconsistency
9	assignment
10	airporevent
11	paxcapacity
12	curfew
13	totaltimedeficit
14	affectedaircraft
15	affectedairports
16	selectedoption

Table 4.1: Features

- commitid - A unique identifier of the commit. A commit is an act of implementing all changes made by an operator and turning them into the live plan.
- revid - The database reversion identifier. This increases each time something is changed in the database. There is (or should be) one-to-one mapping from commitid to revid.
- alerts - The number of things the system has detected as being wrong, for example, consistency violations (for example, aircraft lands on airport X and takes off from airport Y, which is impossible). This number should be the sum of hard and soft alerts.
- soft alerts - Alerts that are less serious.
- hard alerts - Serious alerts.
- route constraints - More permanent limitations on which airport a type of aircraft can fly to, for example, because of size and weight limitations.
- buffer inconsistency - Violations on time limits. For example there must be a minimum amount of time between an aircraft lands and takes off again.
- assignment - A flight, does not have an aircraft assigned to it.
- airport event - It is referred to as an event happening at the airport, for example bad weather or a strike.
- pax capacity - The aircraft can not accommodate the required number of passengers.
- curfew - The number of alerts caused by curfews. A curfew is when a type of aircraft is not allowed to use an airport at certain times, usually for noise abatement reasons.
- total time deficit - The total number of time deficit minutes. This is basically the sum of the buffer inconsistency alerts.

- affected aircraft - The total number of aircraft with alerts
- affected airports - The total number of airports affected by alerts.
- selected option - The solution selected by the operator.

Although the dataset consisted of 20,183 data points, there was an enormous difference between the number of manual solutions and recovery solver solutions. Of the 20,183 cases and solutions to solve a disruption, the recovery solver solution has been used 966 times, taking us to a situation where manual solutions have been preferred 19,217 times, as shown in 4.2. The recovery solver generates a range of 15 solutions ranked accordingly (i.e., option 1 is the option considered the best, option 2 considered as second best and so on.), and the best solution to solve the disruption is chosen by the operator depending on the situation. recovery solver solutions were selected 966 times in which "option 1" was selected 903 times and other options combined were selected 63 times as shown in Figures 4.2, and 4.3.

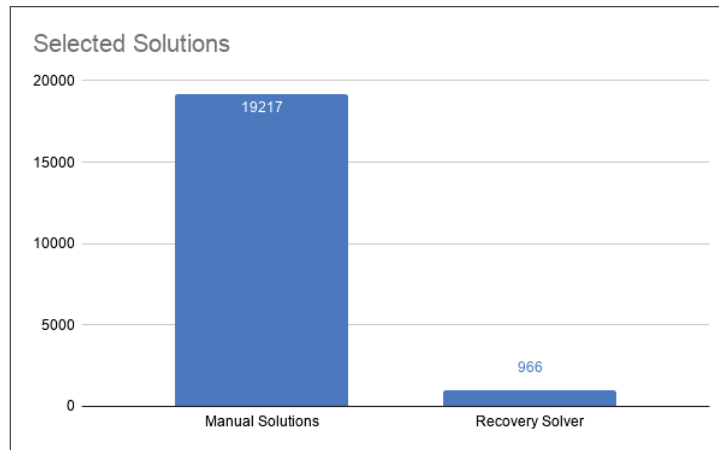


Figure 4.2: Selected Solutions

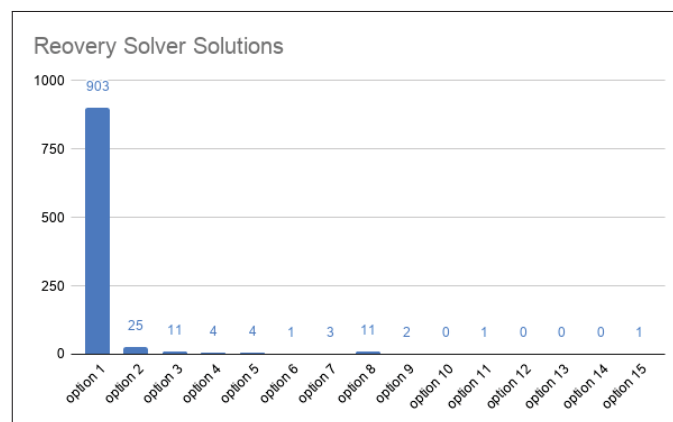


Figure 4.3: Recovery Solver Solutions

4.7 Data preprocessing

After pulling the required data from the database, the first step of data cleaning was to drop duplicate disruption-related solutions as it was found that there were many manual solutions and very few repetitive solutions based on recovery solver. The selected duplicate rows were based on "COMMITID." Total duplicate values of 3541 have been found and dropped. Entries containing null values were also removed, since they are created by the system when the solver is interrupted before a solution is produced. If this type of data is not discarded, noise can be generated, and a classifier's accuracy appears to be decreased. Depending on "REVID," these null values are removed, and a total number of 1689 rows were found and dropped.

Because the recovery solver's solutions were selected so few times, we merged all recovery options (1-15) into one (1) with the aim of training models to predict whether the operator would use a recovery solver solution or create a manual solution.

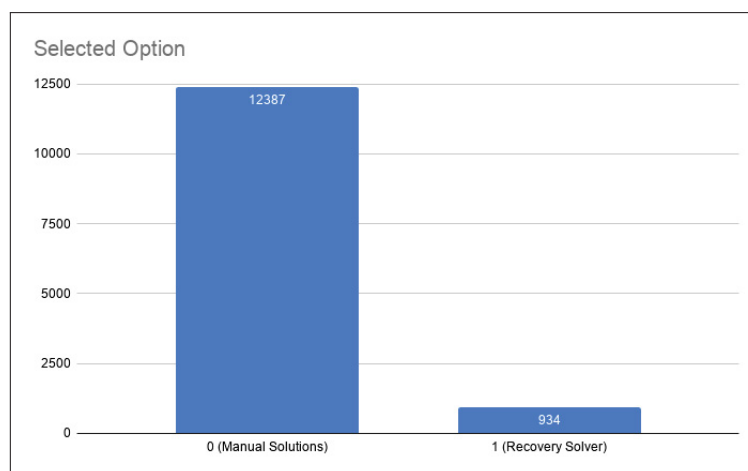


Figure 4.4: Selected Option

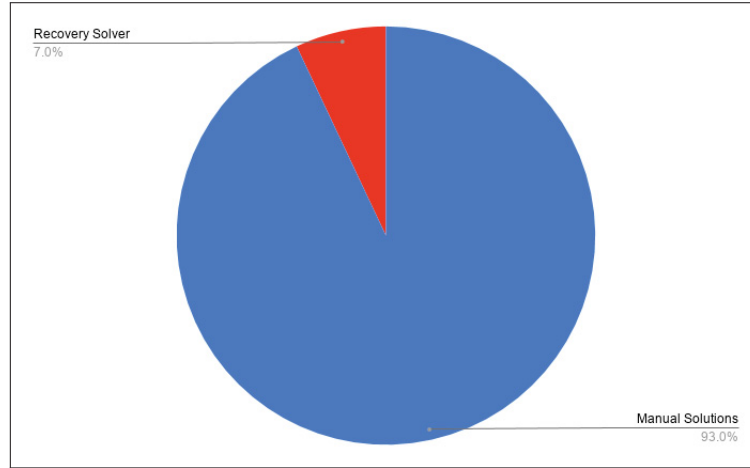


Figure 4.5: Selected Option

As shown in Figure 4.4, 4.5, the classes are imbalanced, i.e., 93% of the data contributes to manual solutions while the remaining 7% contributes to the recovery solution. In such cases, Machine Learning models try to fit the majority class and can provide a biased prediction as well as a false sense of accuracy at the same time. We need more data related to the recovery solver. This can be achieved by creating synthetic data to overcome this problem. For many years, the need for synthetic data has increased in machine learning applications [44]. It is used to represent the original data, and it is very cheap and fast to produce as much as needed to improve the model and training. We could re-sample the dataset that reduces the class of majority or raises the class of minorities. It can be done randomly by using:

- Random Under-Sampling: Here, the majority class observations are under-sampled or removed randomly and uniformly, keeping the minority class as it is.
- Random Over-Sampling: Here, The minority class observations are added randomly by copying some or all of the observations by replicating them multiple times.

These techniques were not adopted due to following reasons:

- Random Under-Sampling helps to balance the dataset. However, the discarded observations might be having valuable information, and this approach could lead to bias.
- In Random Over-Sampling, there is no information loss, but there is a risk of over-fitting due to copying the same information.

To address this problem, SMOTE(Synthetic Minority Oversampling Techniques) [61], [57], [11] is used. New synthetic observations or new data points can be made with SMOTE. The SMOTE method generally involves defining the feature vector and its closest neighbor in the minority class and taking the linear distance between the two points, then multiplying the acquired value with a

random number between 0 and 1 to identify a new line segment point by applying the random number to the feature vector and repeating this procedure for the identified feature. Synthetic data points are the function vectors for the new points as shown in Figure 4.6.

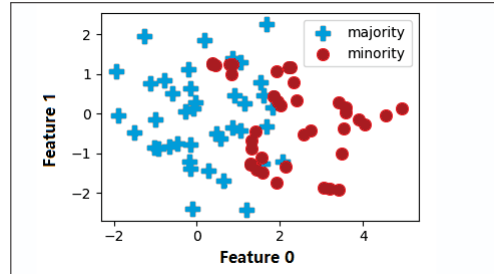


Figure 4.6: SMOTE

Using the Microsoft Azure tool, SMOTE was implemented, the goal was to increase the minority class 1 (recovery solver) by adding synthetic data. The SMOTE percentage parameters were set at 100%, and the number of nearest neighbors was set at 1, which resulted in creating 934 more data points for the minority class. The generated data was cleaned and was fed into a Naive Bayes model to check the accuracy, precision, and recall for the target class. The precision and recall were 74% and 35%.

The SMOTE technique was repeated, setting the percentage of SMOTE 300%, 500%, and 700% consecutively, generating 2802, 4670, and 6538 data points respectively based on the first minority class. Precision and recall stopped increasing and stabilized at 700%, adding 6538 synthetic data points to the data points of the minority class, resulting in a total of 7472 data points.

After cleaning up the newly generated data, i.e., removing the repeated rows, the data consisted of 19283 data points where 12387 belonged to the manual solutions and 6896 belonged to the recovery solver.

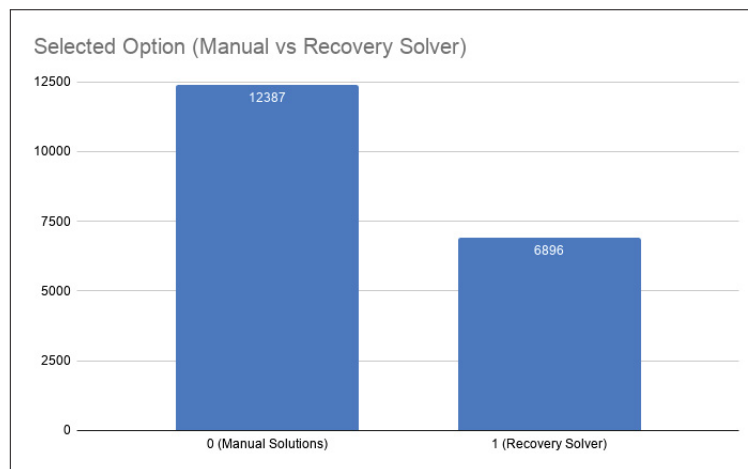


Figure 4.7: Final dataset

4.8 Feature Selection

Feature selection is a technique that is commonly used as it guides the development of an effective predictive model. It helps to identify and pick a small subset of relevant features from the original features based on certain applicable evaluation criteria, which usually leads to better learning results by removing unnecessary, obsolete and redundant attributes from information that do not contribute to improved accuracy [36].

Recursive feature elimination [61] algorithm is a wrapper method that has been used for this research. This operates recursively to delete attributes and to construct a model on the remaining attributes. This uses the accuracy of the model to identify the attributes (and attribute combinations) which contribute the most in predicting the target attribute. This eliminates the features recursively and uses the remaining attributes to build a model and measures the model's accuracy.

Firstly, The importance of all features is calculated, and the mean is taken. The essential features are selected based on their calculated value, i.e.,- the features are considered as necessary if the values of each feature are higher than the acquired mean. Table 4.2 displays the essential features of the dataset.

Mean of all Features = 0.07692307692307693

Feature Name	Feature value	Feature Importance
alerts	0.093	True
softalerts	0.1	True
hardalerts	0.098	True
routeconstraints	0.092	True
buffer	0.102	True
inconsistency	0.091	True
assignment	0.029	False
airportevent	0.01	False
paxcapacity	0.001	False
curfew	0.092	True
totaltimedeficit	0.135	True
affectedaircraft	0.086	True
affectedairports	0.073	False

Table 4.2: Feature Importance

As shown in the Table 4.2, the feature significance of every value and, when compared to the mean of all features, it is shown that out of 13 features, nine feature values are higher than the mean. By applying Recursive Feature Elimination by Gradient Boosting techniques, all features are selected and trained to identify the right number of features that impact the target label to improve accuracy. Table 4.3 shows the number of selected features and their respective accuracies.

Number of Features Selected	Accuracy
1	0.69743
2	0.81799
3	0.90484
4	0.92792
5	0.93207
6	0.93803
7	0.93933
8	0.94399
9	0.94503
10	0.94840
11	0.94607
12	0.94684
13	0.94581

Table 4.3: Feature Selection using Recursive Feature Elimination

It is clearly seen in Table 4.3 that the accuracy reached a good percentage with three features. But for better performance of the model, we should consider the features which give us the maximum accuracy. The maximum accuracy (i.e.,0.94840) is attained by considering the following features.

S.no	Features
1	alerts
2	softalerts
3	hardalerts
4	routeconstraints
5	buffer
6	inconsistency
7	assignment
8	curfew
9	totaltimedeficit
10	affectedaircraft

Table 4.4: Selected Features for the Experiment

4.9 Experimental Setup

The experiment is to implement the selected approaches and compare them concerning their performances. Pandas were used to load, analyze, and manipulate the data. The experimental procedure for the adapted approaches was carried out in two ways, where in the first procedure, the dataset was divided into a training dataset and testing dataset using `train_test_split`, and in the second procedure, the data were divided into training, testing and validation using nested cross-validation. The reason behind using the two approaches is to compare the performances and also to avoid the problem of overfitting. Min-MaxScalar was preferred over StandardScalar for normalization as it rescales the dataset such that all feature values are in the range $[0, 1]$, and it will preserve the shape of the dataset (no distortion). Nested cross validation was used in this research as mentioned in Section 2.9.6. The nested cross-validation was initially conducted for 5,7,10,12 folds, and it was observed that the performance of the model reached its peak when 10 folds were used and the performance deteriorated when the number of folds were increased. GridSearchCV (meta-estimator) was used to construct the inner loop CV in order to determine the

best parameters and `cross_val_score` was used in the outer loop CV to evaluate the model. Matplotlib and Yellowbrick were used for the visualization of the data and results.

The performance metrics are noted for the selected and implemented algorithms. The experimental results are analyzed, and performance metric scores are compared to find the suitable machine learning algorithm for our problem

4.10 Adapted approaches and implementation

The adapted approaches were implemented using SKlearn and the procedure followed in this thesis is discussed below:

4.10.1 MultiLayer Perceptron

Neural networks are well known for their remarkable performance and their capability to find meaning in an unreliable and complicated data. Neural networks can be utilized to obtain the patterns and detect trends that are generally difficult to be noticed by other computing techniques. MLP is known for the ability to learn the tasks based on the data considered for training, and also, they do not make any assumption regarding the underlying probability density functions or other probabilistic information about the pattern classes under consideration in comparison to other probability-based models.

All the inputs were normalized the range of 0 to 1 and by splitting the data using `train_test_split` and nested cross validation the data is fit into the model. After experimenting with a different number of hidden layers and hidden nodes, it was observed that a single hidden layer with many hidden nodes (i.e., 50 nodes) performed better, but when extra hidden layers were added, it did not help much. Instead of increasing the number of nodes gave us better performance. The results are highly influenced by various parameters like `batch_size`, `learning_rate`, and `max_iter`. The parameter range in inner loop CV was set to (`'activation'`: [identity, logistic, tanh, relu], `'solver'`: [lbfgs, sgd, adam], `'batch_size'`: [100, 250, 500, 750, 1000, 1500], `'learning_rate'` = [constant, invscaling, adaptive], `max_iter'` = [1000, 5000, 10000, 15000, 20000]). The best fit was achieved when `solver = lbfgs`, `activation = tanh`, `batch_size = 500`, `hidden_layer_sizes = (50)` and `max_iter = 15000`. The results obtained are shown in the Chapter 5.

4.10.2 Probabilistic Neural Network

Probabilistic neural network is beneficial when it comes to automatic pattern recognition, and the training process of PNN is faster when compared to several known back propagation algorithms [56]. The input data was normalized before

training as the pnn network is sensitive for cases where one input feature has bigger values than the other one. PNN does not need iterative training as it works on the principle of lazy learning. It just stores parameters and uses them to make predictions. From NeuPy Library, the PNN model is constructed. The input features were normalized to the range of 0 to 1, and by splitting the data using `train_test_split` and nested cross-validation, the data is fit into the model. The parameter range in inner loop CV was set to (`'std'` : [0.3, 0.5, 0.7], `'batch_size'`: [100, 250, 500, 750, 1000]). The best fit was achieved when `batch_size = 500` and `standard deviation(std)=0.5`.

4.10.3 XGBoost

The two common terms in machine learning are bagging and boosting. Boosting is similar to bagging; the selection of the sample is, however, done in a different way. XGBoost is a more feasible and efficient algorithm when compared with other machine learning algorithms, and it can take various types of input data. Advanced features for algorithm enhancement and model tuning can be done in XGboost, and additionally, it is also capable of performing gradient boosting(GB), stochastic gradient boosting(SGB), and regularized gradient boosting(RGB) which are known as the three main forms of gradient boosting.

From Sklearn model selection, the XGBoost model is constructed. As XGBoost does not require many parameters tuning to get a good performance. Tuning `n_estimators`, `max_depth`, and `min_child_weight` made an impact on the results. The input features were normalized to the range of 0 to 1, and by splitting the data using `train_test_split` and nested cross-validation, the data is fit into the model. The parameter range in inner loop CV was set to (`'learning_rate'` : [0.1, 0.5, 0.01, 0.05], `'n_estimators'`: [100, 500, 700, 1000, 1500], `'max_depth'`: [10, 15, 20, 25, 30], `'min_child_weigh'` = [0.1, 0.001, 1]). The best fit was acquired at `n_estimators = 1000`, `max_depth = 20`, and `min_child_weight = 0.001` and `learning_rate = 0.5`.

4.10.4 Support Vector Machine(SVM)

The SVM generally uses the kernel trick to transform the data and finds an optimal boundary between the possible outputs. In other words, it is capable of doing extremely complex data transformations and calculates in a way to differentiate the data based on the outputs defined. The benefit is that much more complex relationships can be captured between the data points without having to perform difficult transformations manually. From Sklearn model selection, the SVM model is constructed. The input features were normalized to the range of 0 to 1, and by splitting the data using `train_test_split` and nested cross-validation, the data is fit into the model. The parameter range in inner loop CV was set to (`'C'`: [0.1, 1, 10, 100], `'gamma'`: [1, 0.1, 0.01, 0.001], `'kernel'` = [linear, poly, rbf]). After experimenting with different splits

as mentioned in Section 4.9 The best fit was acquired at 'kernel': 'rbf' 'C': 10, 'gamma': 0.001.

4.11 Performance Metrics

In Machine Learning, performance measurement is an essential task. So when it comes to a classification problem, we can count on Performance metrics like accuracy, precision, recall and f1 score, Confusion Matrix and AUROC (Area Under the Receiver Operating Characteristics).

4.11.1 Confusion Matrix

Confusion matrix is a ubiquitous tool and one of the most intuitive metrics for finding the correctness and determining the accuracy of a model [21]. It is used for classification problems, and it holds information regarding actual and predicted classifications performed by a classification system. Performance of such systems is usually assessed using the data in the matrix.

		Prediction outcome		total
		p	n	
actual value	p'	True Positive	False Negative	P'
	n'	False Positive	True Negative	N'
total		P	N	

Figure 4.8: Confusion Matrix

Generally, the confusion matrix in itself is not a performance measure as such, but essentially all of the performance metrics are based on the confusion matrix and the numbers inside it. the following are the terms related to the confusion matrix:

- True Positives (TP): True positives were the cases when the actual class of the data point was 1(True), and the predicted is also 1(True).
- True Negatives (TN): True negatives were the cases when the actual class of the data point was 0(False), and the predicted is also 0(False).
- False Positives (FP): False positives were the cases when the actual class of the data point was 0(False), and the predicted is 1(True). False is because the model has mispredicted and positive because the class predicted was a positive one (1).

- False Negatives (FN): False negatives were the cases when the actual class of the data point was 1(True), and the predicted is 0(False). False is because the model has mispredicted and negative because the class predicted was a negative one (0).

4.11.2 Accuracy

Accuracy is a measure that tells whether a model/algorithm is being trained correctly and how it performs, or it is the number of correct predictions made as a ratio of all predictions made.

$$Accuracy = \frac{(TruePositives + TrueNegatives)}{(TruePositives + TrueNegatives + FalsePositives + FalseNegatives)} \quad (4.1)$$

4.11.3 Precision

Precision applies to the closeness of two or more measurements to each other. It is the repeatability or reproducibility of the measurement.

$$Precision = \frac{TruePositives}{(TruePositives + FalsePositives)} \quad (4.2)$$

4.11.4 Recall

Recall, also known as "sensitivity," applies to the fraction of related situations that have been reclaimed over the total amount of related situations.

$$Recall = \frac{TruePositives}{(TruePositives + FalseNegatives)} \quad (4.3)$$

4.11.5 F1 score

F1 score also called as F-Score or F-measure. F1 score is the measure of calculating the weighted average of precision and recall.

$$F1 = 2 \times \left(\frac{Precision \times Recall}{Precision + Recall} \right) \quad (4.4)$$

4.11.6 AUROC (Area Under the Receiver Operating Characteristics)

AUROC is used to monitor or visualize classification problems performance. It is one of the most valuable metrics for measuring the performance of any model

of classification. It is a performance measurement for classification problem at various thresholds settings where Area Under Curve (AUC) represent degree or measure of separability and Receiver Operating Characteristics (ROC) a probability curve. It tells how much model is capable of distinguishing between classes. Generally, the higher the area under the roc curve, the better the model is at predicting 0s as 0s and 1s as 1s. The score varies from 0.5 to 1, and if the score is 1, it is known as the ideal case where TPR is 1 and FPR is 0, which means all the positives and negatives are classified correctly. The Figures 4.10, 4.11 give a better understanding. ROC is plotted with True Positive Rate (TPR) against the False Positive Rate (FPR), where TPR is on the y-axis, and FPR is on the x-axis on Figure 4.9. In simpler words, the relationship between TPR and FPR is the ROC curve. TPR is known as sensitivity, and FPR is known as (1 - specificity).

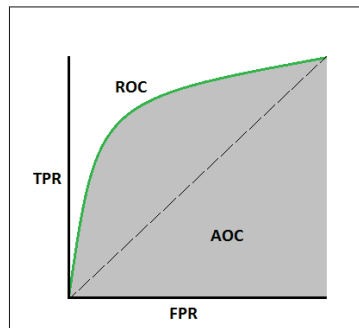


Figure 4.9: AUC

The following is a brief explanation about TPR and FPR.

- TPR: among all the positives, what percentage of them are correctly classified as positive. For example, we are trying to classify cats and dogs. We have 1000 cats (positives) and 400 dogs (negatives) in total, and we correctly identify 700 cats among 1000, the TPR is 70%.

$$TPR = \text{Sensitivity} = \frac{\text{True Positives}}{(\text{True Positives} + \text{False Negatives})} \quad (4.5)$$

- FPR: among all the negatives, what percentage of them are falsely classified as positive. Considering the above example, if we classify 200 dogs to be cats, then the FPR would be 50%.

$$FPR = (1 - \text{Sensitivity}) = \frac{\text{False Positives}}{(\text{False Positives} + \text{True Negatives})} \quad (4.6)$$

From Figure 4.9, it can be seen that there is a new relationship between TPR and FPR. TPR can be maximized by classifying all 1400 animals to be cats, which will cover all the cats (TPR: 100%) to be accurately classified. But it will also make FPR be 100%. And to minimize the FPR to 0, we can mark all like dogs, which will, in turn, make TPR be 0. That's the theory behind why the ROC curve starts from the origin and ends at (1,1).

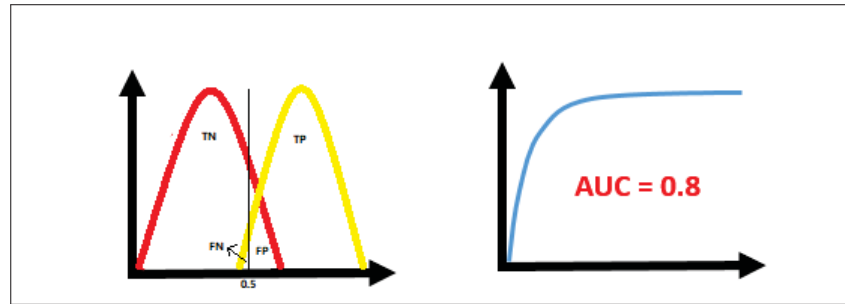


Figure 4.10: AUC = 0.8

When two divisions overlap, type 1 and type 2 errors are proposed. Depending on the threshold, we can minimize or maximize them. When AUC is 0.8, it means there is an 80% chance that the model will be capable to differentiate within positive class and negative class.

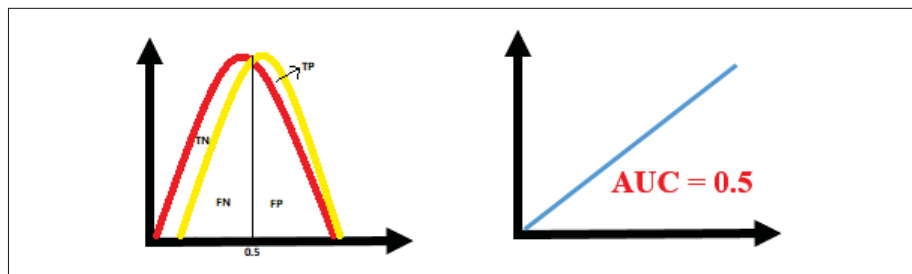


Figure 4.11: AUC = 0.5

This is the worst situation. When AUC is approximately 0.5, the model has no discrimination capacity to distinguish between positive class and negative class.

$$\textit{Specificity} = \frac{\textit{number of true negatives}}{(\textit{number of true negatives} + \textit{number of false positives})} \quad (4.7)$$

Sensitivity and specificity are inversely proportionate to each other. So when sensitivity is increased, specificity decreases, and vice versa.

Multilayer Perceptron, Probabilistic Neural Network, XGBoost, and SVM algorithms are selected based on the literature review done in chapter 3 and are evaluated on the final dataset. Performance metrics for these algorithms are presented in this chapter.

5.1 Performance

The performance metrics mentioned in Section 4.11 are used to evaluate the performance of the selected approaches. The most common evaluation metric for classification problem is the classification accuracy. It was not only the performance metric considered as it is the most misused performance metric. Identifying how the models perform on the target classes (i.e., manual solutions and recovery solver Solutions) is mandatory since the data is unbalanced. Confusion Matrix, Classification Report, and AUROC Curve are the performance metrics that have been chosen to measure the performance and are displayed here.

5.1.1 Multilayer Perceptron results

Multi-layer perceptron algorithm is trained on the training data and tested, and the following results are obtained. On implementing the Multilayer Perceptron algorithm on the dataset, it achieves an accuracy of 94.14%. The classification report is displayed in Figure 5.2. The Classification Report obtained on implementing the MultiLayer Perceptron on the training dataset is presented in Figure 5.1. Figure 5.2 shows the Classification Report obtained after cross-validation.

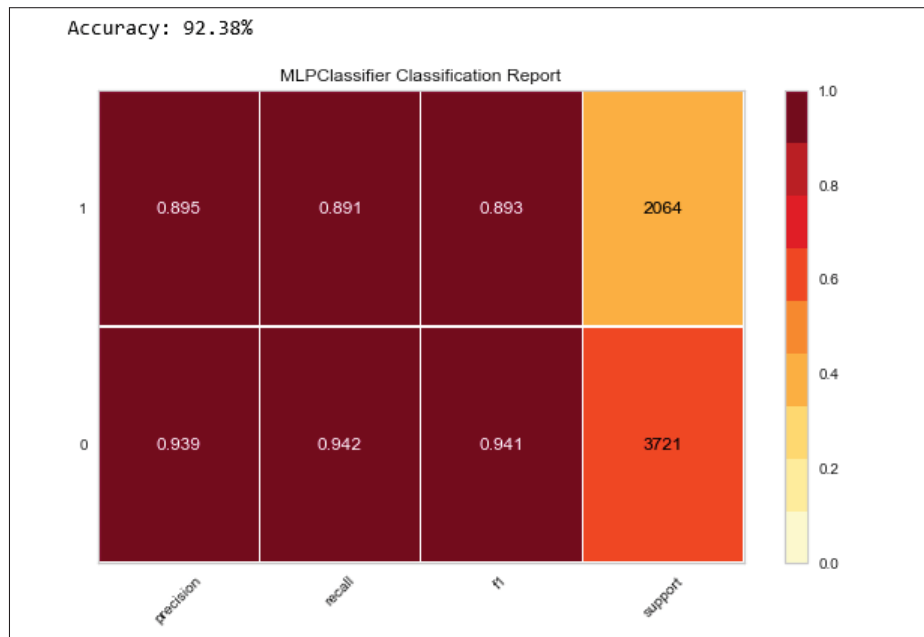


Figure 5.1: MLP before cross validation

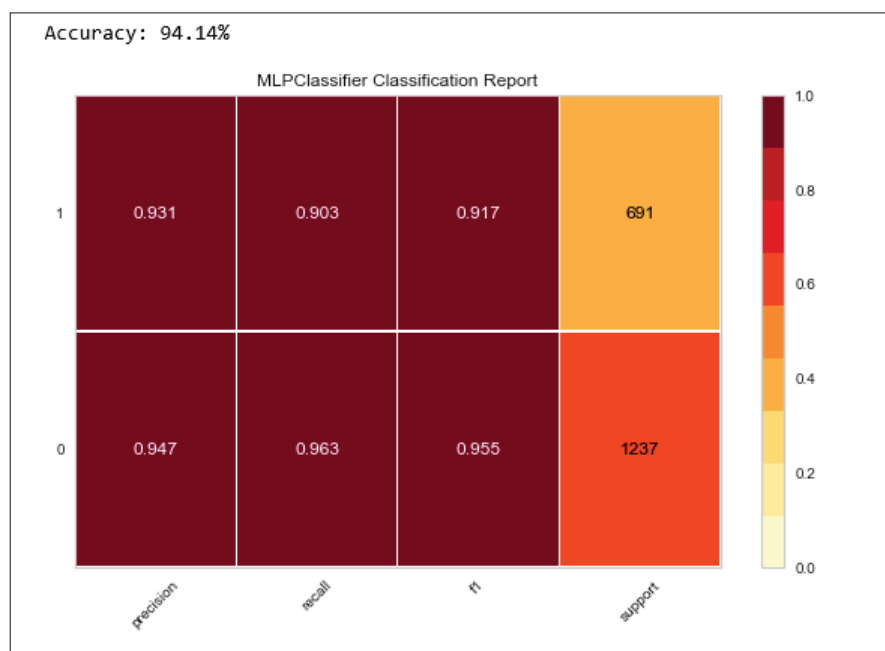


Figure 5.2: MLP after cross validation

5.1.2 Probabilistic Neural Network results

Probabilistic Neural Network algorithm is implemented on the test data, and the following results are obtained. On implementing the Probabilistic Neural Network algorithm on the dataset, it achieves an accuracy of 93.05%. The classification report is displayed in Figure 5.4.

The Classification Report obtained on implementing the Probabilistic Neural Network on the training dataset is presented in Figure 5.3. Figure 5.4 shows the Classification Report obtained after cross-validation.

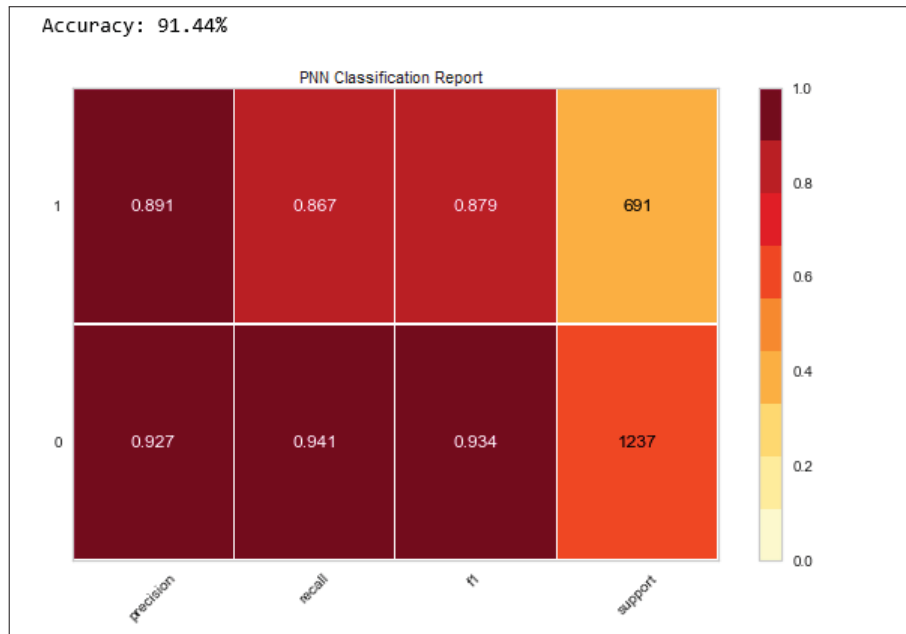


Figure 5.3: PNN before cross validation

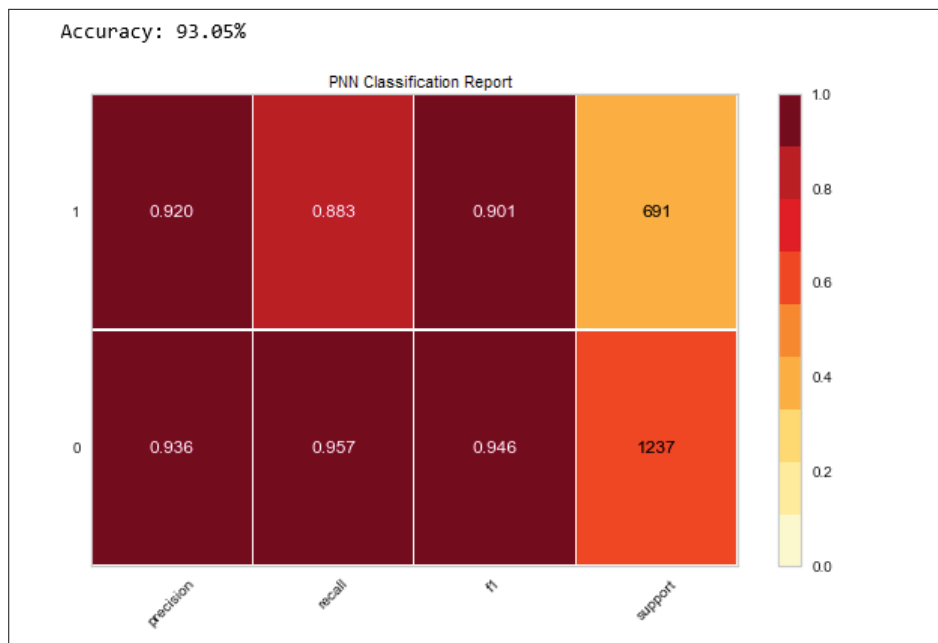


Figure 5.4: PNN after cross validation

5.1.3 XGBoost results

XGBoost algorithm is implemented on the test data, and the following results are obtained. On implementing the XGBoost algorithm on the dataset, it achieves an accuracy of 96.06%. The classification report is displayed in Figure 5.6. The Classification Report obtained on implementing the XGBoost on the training dataset is presented in Figure 5.5. Figure 5.6 shows the Classification Report obtained after cross-validation.

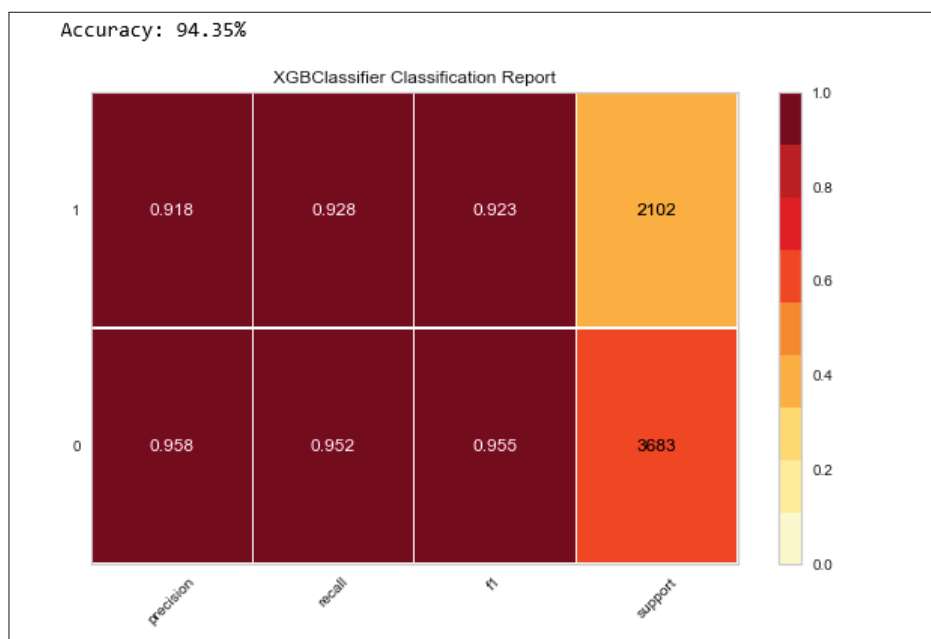


Figure 5.5: XGBOOST before cross validation

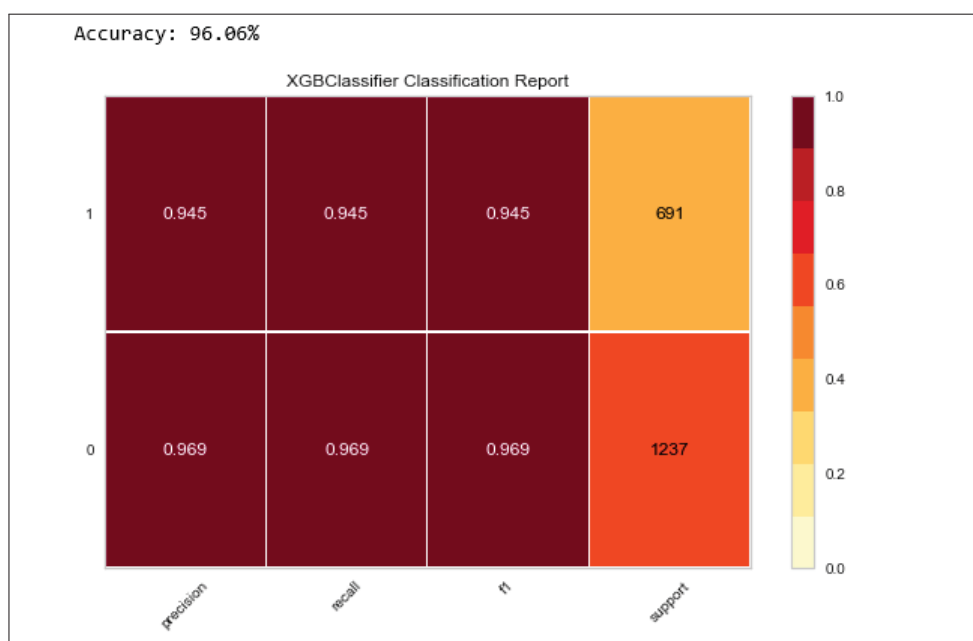


Figure 5.6: XGBOOST after cross validation

5.1.4 Support Vector Machine (SVM) results

SVM algorithm is implemented on the test data, and the following results are obtained. On implementing the SVM algorithm on the dataset, it achieves an accuracy of 88.59%. The classification report is displayed in Figure 5.8. The Classification Report obtained on implementing the SVM on the training dataset is presented in Figure 5.7. The Figure 5.8 shows the Classification Report obtained after GridSearch.

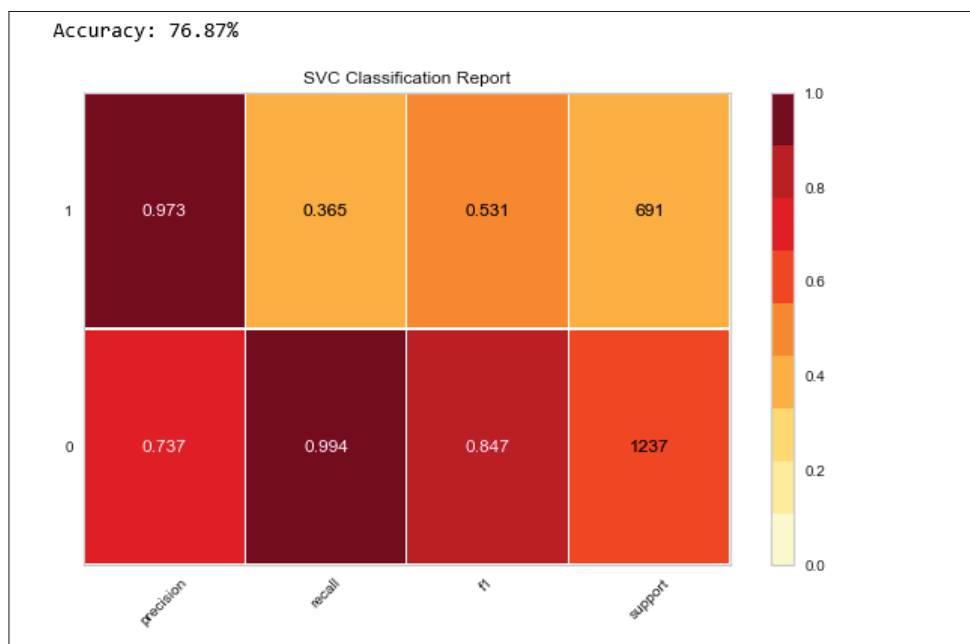


Figure 5.7: SVM before cross validation

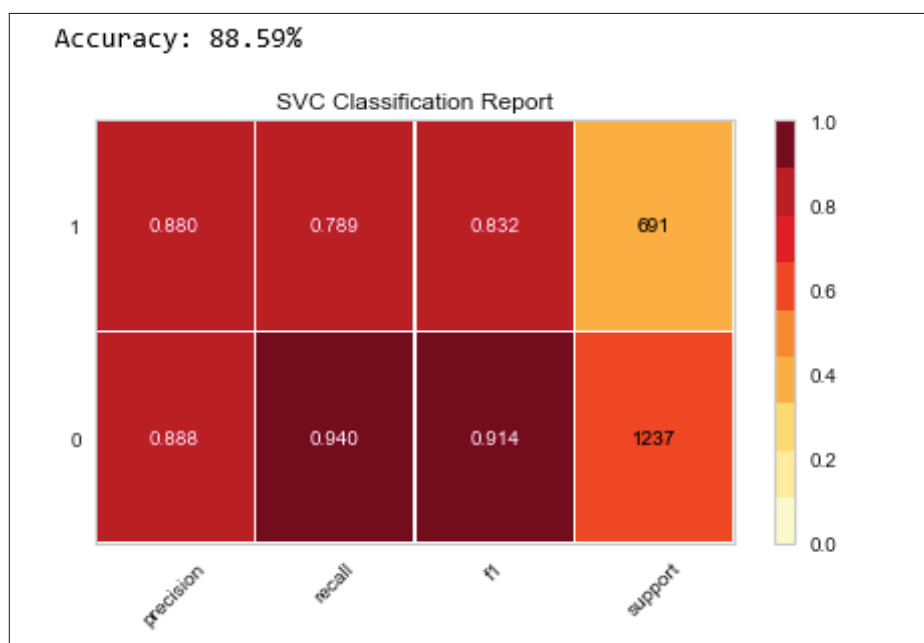


Figure 5.8: SVM after cross validation

5.1.5 AUROC Curve

The ROC graph summarizes all of the confusion matrices that each threshold produced. The threshold can lie in the range $<0,1>$ for threshold values in the range 0 to 1 with a 0.XX step size, the true positive rate, and false-positive rate are calculated, and the resulting graph was plotted. We calculate the Area Under Curve (AUC), which is useful for comparing the performance of different models on the same data.

Multilayer Perceptron Confusion Matrix and AUROC Curve

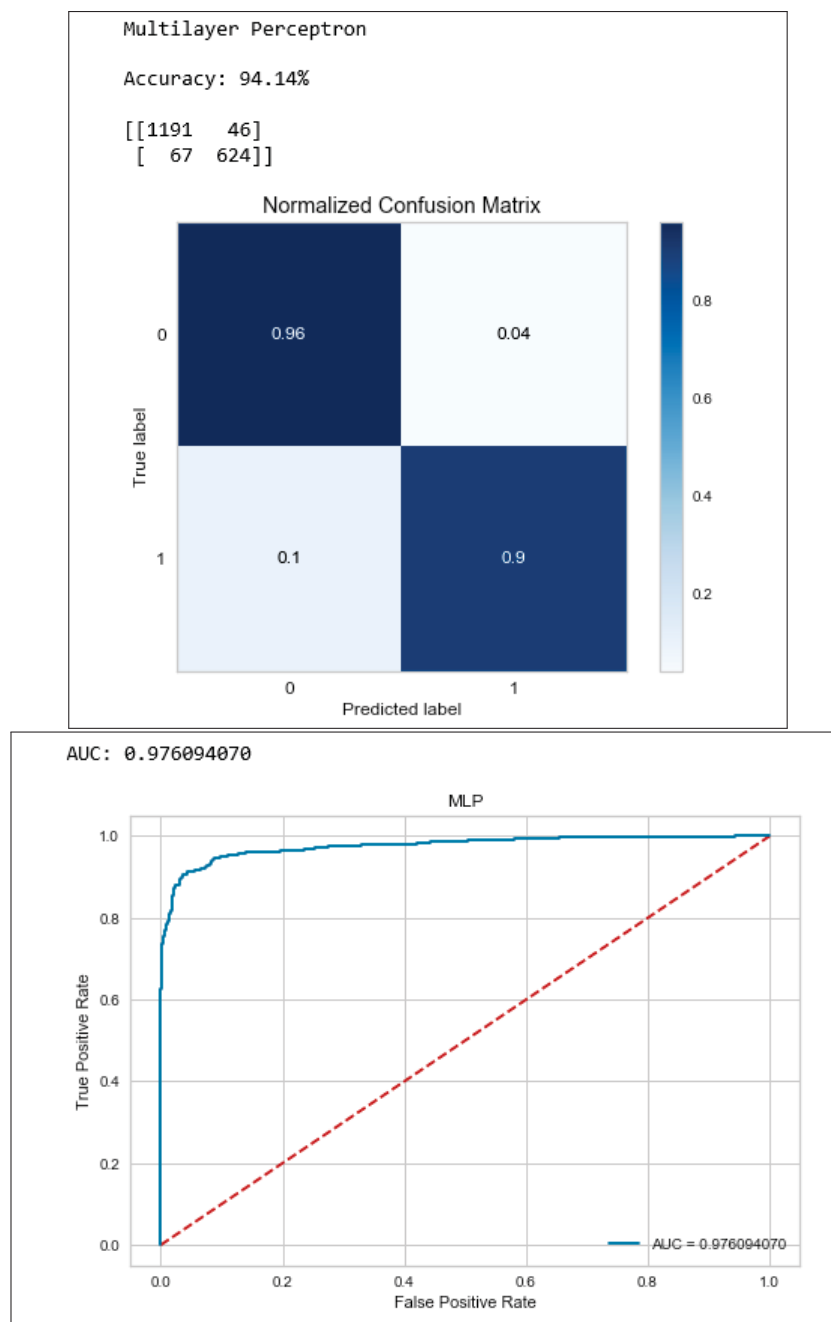


Figure 5.9: MLP Confusion Matrix and AUC

Probabilistic Neural Network Confusion Matrix and AUROC Curve

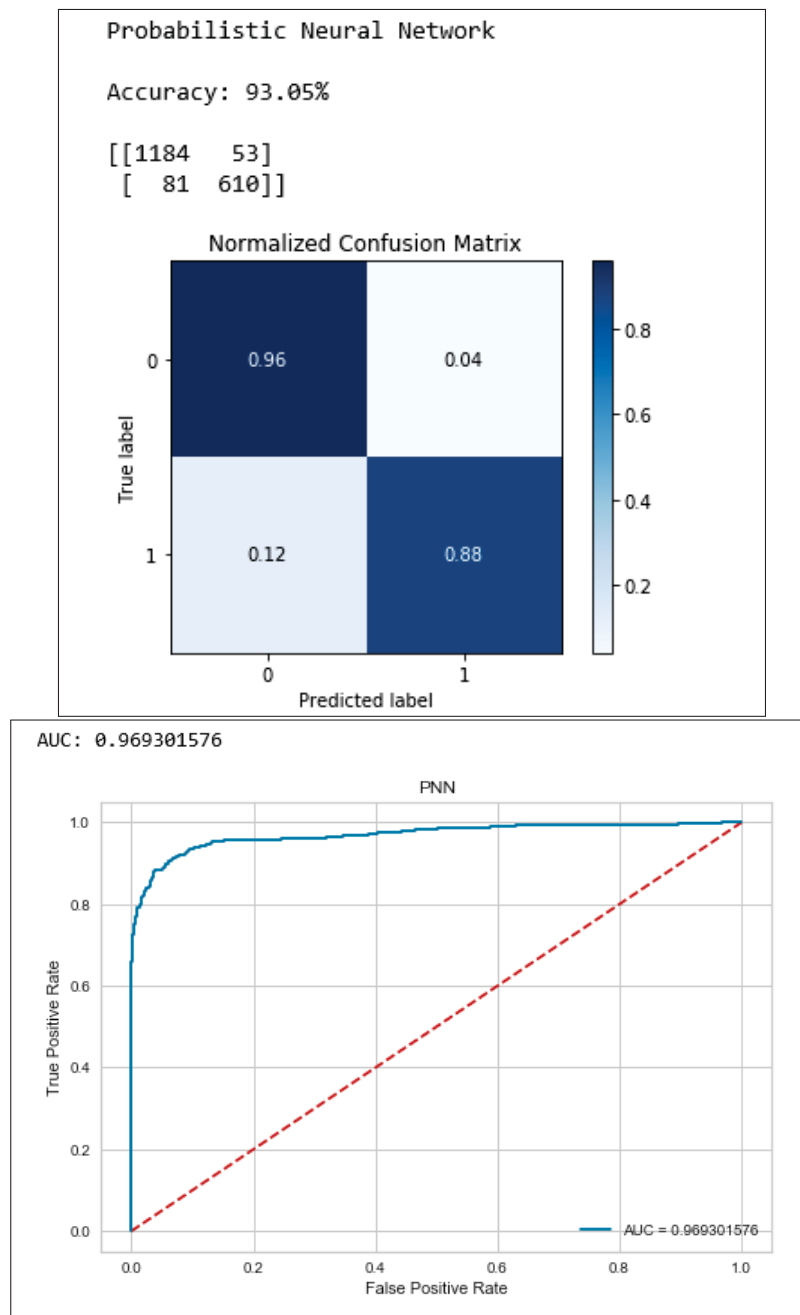


Figure 5.10: PNN Confusion Matrix and AUC

XGBOOST Confusion Matrix and AUROC Curve

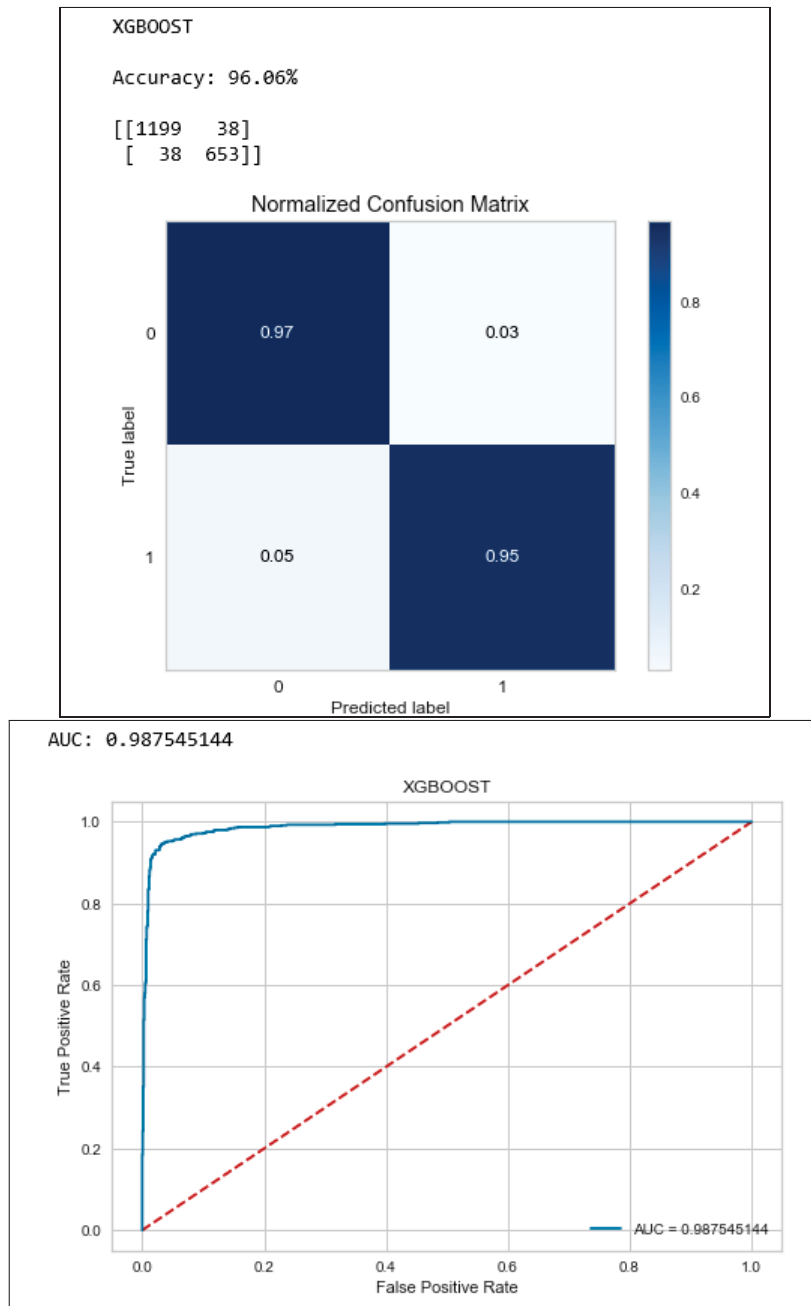


Figure 5.11: XGBOOST Confusion Matrix and AUC

Support Vector Machine Confusion Matrix and AUROC Curve

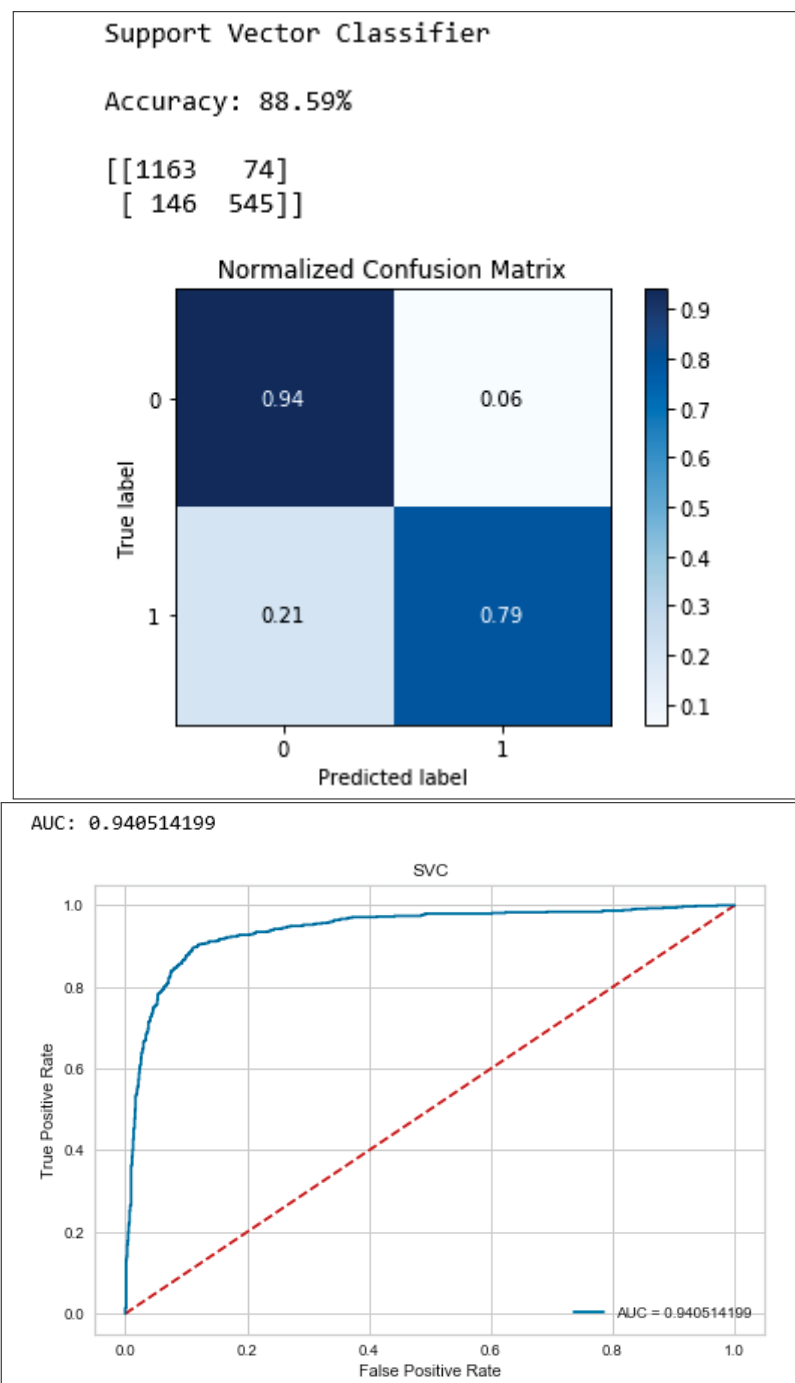


Figure 5.12: SVM Confusion Matrix and AUC

5.2 Results Comparison

Based on the experiment results, the findings of the classifiers are tabulated based on its performance metrics.

Model	Accuracy	Precision	Recall	F1 Score
Multilayer perceptron	0.9414	0.9628	0.9467	0.9547
Probabilistic Neural Network	0.9305	0.9572	0.9360	0.9464
XGBoost	0.9606	0.9693	0.9693	0.9693
SVM	0.8859	0.9402	0.8885	0.9136

Table 5.1: Comparison of Classifiers using the performance metrics

The main aim of this research was achieved by evaluating the performance of each algorithm, namely Multilayer Neural Network, Probabilistic Neural Network, XGBoost, and SVM with the chosen performance metrics, namely accuracy, precision, recall, F1 score and AUROC curve by experimenting. The performance of the algorithms is displayed in Chapter 5, and it is identified that they are working exceptionally well. From the obtained results, a comparison between the models is made, and the model which gives the best results for classifying the two cases (i.e., manual solutions and recovery solver solutions) is identified. It is observed that neural network and other machine learning models have worked exceptionally well, but few factors make them different from each other. XGBoost, Multilayer Perceptron, and Probabilistic Neural network did not have many differences in their performances. It is observed that XGBoost performs comparatively better when compared to the neural network models.

6.1 Answering the Research Questions

RQ 1: What are the suitable machine learning techniques for predicting the operator's choice in the case of airline disruptions?

By conducting a literature review, several examples, and previous studies have been taken into account in connection to the problem statement and context of the study. It was concluded that XGBoost, Multilayer Perceptron, Probabilistic Neural Network, and SVM algorithms could be implemented and compared in this research.

RQ 2: What are the performances of machine learning models in the prediction of the operator's choice based on the previous recorded data?

For the chosen algorithms, the performances are evaluated, and the results are extracted. The clear description of the chosen metrics, namely accuracy, precision, F1 score, recall, is mentioned in Section 4.11. By experimenting, the

results obtained for the performance metrics of the algorithms individually are described below:

The adapted approaches have worked exceptionally well in predicting between the two classes 0 and 1 (i.e., manual solutions and recovery solver solutions) as seen in the Figures 5.2, 5.4, 5.6, 5.8 and Table 6.1. XGBoost has a better score in precision for the two classes when compared to the other algorithms. The recall score of XGBoost and MLP are similar for manual solutions (0), but when recall score for recovery solver(1) is considered, XGBoost outperforms MLP. It is essential to have good scores in terms of precision and recall as high precision states that the algorithm produces more relevant results than irrelevant ones. Similarly, the high recall states that the algorithm has returned most of the relevant results. As the F1 score being the sub-contrary mean of the precision and recall. XGBoost has a higher F1 score when compared to MLP, PNN, and SVM, and additional to these performance metrics, even the AUC score was considered to measure how well the selected models can distinguish between two classes. As mentioned earlier in Section 4.11.6 that the better the classification algorithm is, the higher the area under the roc curve. XGBoost is classifying the two classes accurately as it has the highest AUC score when compared to the other algorithms. It can be clearly observed in the Figures 5.9, 5.10, 5.11, 5.12 and Table 6.2.

Hence, by experimenting, the results are retrieved based on the performance metrics chosen for all the implemented algorithms are shown in the Tables 6.1, 6.2 and Figure 6.1.

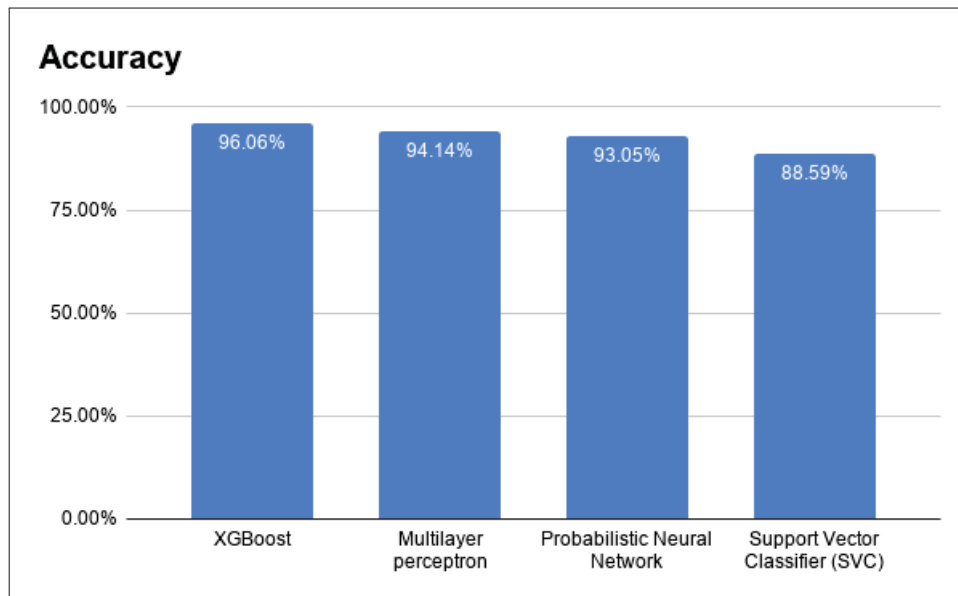


Figure 6.1: Comparison of the Accuracy

Model	Accuracy
XGBoost	96.06%
Multilayer perceptron	94.14%
Probabilistic Neural Network	93.05%
SVM	88.59%

Table 6.1: Accuracy of selected models

Model	Precision	Recall	F1 score	AUC score
XGBoost	0.9693	0.9693	0.9693	0.98754
Multilayer perceptron	0.9628	0.9467	0.9547	0.97609
Probabilistic Neural Network	0.9572	0.9360	0.9464	0.96930
SVM	0.9402	0.8885	0.9136	0.94051

Table 6.2: Classification Report

From the results achieved by addressing the research question 1, the four machine learning models are compared, namely MLP, PNN, XGBoost, and SVM. It is identified that XGBoost as an efficient model with better accuracy, recall, precision, f1 score, and it performs better in predicting the cases of manual and recovery solver solutions. The performance, results, and comparison of the four algorithms presented in Chapter 5. The metric values of each algorithm are categorized and sorted and have a clear view of drawing a conclusion in which algorithm is better over the other. It is concluded that XGBoost performs better when compared to the other machine learning algorithms.

6.2 Validity Threats

By careful considerations during the whole process, we were able to avoid and overcome most of the possible validity threats.

Internal Validity Threats:

This threat arises if the selection of algorithms goes wrong. As this is the first step of the research, enough attention must be given, so there is less chance of picking an unsuitable algorithm.

External Validity Threats

Missing values in the dataset could impact the performance measures of the models.

Other Validity Threats

Initially, three cases were considered which are manually created solutions, recovery solver solutions and combination of both. The combination solutions were taken to identify when does the operator considers both manual and recovery solver to solve a disruption but there was no recorded data where an operator solved a disruption using the combination. Only manual solutions

and recovery solver were considered for this research.

Conclusion Validity

In all machine learning models, overfitting is a potential problem. To address this, cross-validation was applied.

6.3 Limitations

This research successfully answers the first part of the problem statement, which is, "Will the operator use the solver." However, it was not possible to answer, "If the operator uses the solver, which solution will he prefer." due to insufficient data related to the recovery solver.

Conclusions and Future Work

In this research work, we described the problem context of airline disruption and details about the background of various concepts. We generated a synthetic dataset that is similar to the training dataset and expanded it to train the models. We formulated two research questions to address the objectives of this research. We also presented the related work, which includes various research works and evaluation measures. Manual solutions were preferred over recovery solver in most of the cases during disruptions, which resulted in fewer data related to recovery solver.

We have successfully trained various machine learning and neural network models to predict the choice of an operator during a disruption. Data was the most challenging factor in the whole research. From the research, we conclude that the XGBoost model performs slightly better when compared to SVM and neural network models, which are Multilayer Perceptron and Probabilistic Neural Network. Also, we can conclude that Neural Networks could have performed better if the dataset was larger. We feel that the implementation of neural networks using TensorFlow could be beneficial in terms of flexibility as it is essential to know what is going on behind the black box as tuning has a significant impact on the accuracy or lower error rate in the model's prediction.

There is a lot of scope and potential for schedule recovery in airlines. For future work, a recommendation system that could rank the given disruption according to its complexity can be developed. It could be classified into two types and directed to the recovery solver or to the operator accordingly. By doing this, more data could be generated related to the recovery solver. Another possible future work could be working on deep, reinforcement learning models that can solve novel scenarios both faster and with better results than existing algorithms.

References

- [1] Osman Ahmed Abdalla, Mohd Nordin Zakaria, Suziah Sulaiman, and Wan Fatimah Wan Ahmad. A comparison of feed-forward back-propagation and radial basis artificial neural networks: A monte carlo study. In *2010 international symposium on information technology*, volume 2, pages 994–998. IEEE, 2010.
- [2] Ahmed Abdelghany and Khaled Abdelghany. *Modeling applications in the airline industry*. Routledge, 2016.
- [3] Rodrigo Acuña-Agost, Philippe Michelon, Dominique Feillet, and Serigne Gueye. Statistical analysis of propagation of incidents for rescheduling simultaneously flights and passengers under disturbed operations. In *ROADEF 2009. 10^{ème} Congrès de la Société Française de Recherche Opérationnelle et d’Aide à la Décision*, 2009.
- [4] SS Allan, SG Gaddy, and JE Evans. Delay causality and reduction at the new york city airports using terminal weather information systems. Technical report, Citeseer, 2001.
- [5] Barbro Back, Teija Laitinen, and Kaisa Sere. Neural networks and genetic algorithms for bankruptcy predictions. *Expert Systems with Applications*, 11(4):407–413, 1996.
- [6] Ra J Bandyopadhyay and Rafael Guerrero. Predicting airline delays, 2012.
- [7] Imad A Basheer and Maha Hajmeer. Artificial neural networks: fundamentals, computing, design, and application. *Journal of microbiological methods*, 43(1):3–31, 2000.
- [8] Benjamin Bengfort, Rebecca Bilbro, Nathan Danielsen, Larry Gray, Kristen McIntyre, Prema Roman, Zijie Poh, et al. Yellowbrick, 2018.
- [9] Leo Breiman. Bagging predictors. *Machine learning*, 24(2):123–140, 1996.
- [10] António JM Castro and Eugénio Oliveira. Disruption management in airline operations control—an intelligent agent-based approach. *Web Intelligence and Intelligent Agents*, pages 107–132, 2010.
- [11] N. V. Chawla, K. W. Bowyer, L. O. Hall, and W. P. Kegelmeyer. SMOTE: synthetic minority over-sampling technique. *Journal of Artificial Intelligence Research*, 16:321–357, 2002.
- [12] Haiyan Chen, Jiandong Wang, and Xuefeng Yan. A fuzzy support vector machine with weighted margin for flight delay early warning. In *2008*

- Fifth International Conference on Fuzzy Systems and Knowledge Discovery*, volume 3, pages 331–335. IEEE, 2008.
- [13] Tianqi Chen and Carlos Guestrin. Xgboost: A scalable tree boosting system. In *Proceedings of the 22nd acm sigkdd international conference on knowledge discovery and data mining*, pages 785–794. ACM, 2016.
- [14] Sun Choi, Young Jin Kim, Simon Briceno, and Dimitri Mavris. Prediction of weather-induced airline delays based on machine learning algorithms. In *2016 IEEE/AIAA 35th Digital Avionics Systems Conference (DASC)*, pages 1–6. IEEE, 2016.
- [15] Jens Clausen, Allan Larsen, and Jesper Larsen. Disruption management in the airline industry—concepts, models and methods. *Technology Report IMM-Technical Report-2005-01, Informatics and Mathematical Modelling, Technical University of Denmark, DTU*, 2005.
- [16] Corinna Cortes and Vladimir Vapnik. Support-vector networks. *Machine learning*, 20(3):273–297, 1995.
- [17] Thomas G Dietterich. Ensemble methods in machine learning. In *International workshop on multiple classifier systems*, pages 1–15. Springer, 2000.
- [18] Thomas G Dietterich et al. Ensemble learning. *The handbook of brain theory and neural networks*, 2:110–125, 2002.
- [19] David M Dutton and Gerard V Conroy. A review of machine learning. *The knowledge engineering review*, 12(4):341–367, 1997.
- [20] Bradley Efron and Robert J Tibshirani. An introduction to the bootstrap, volume 57 of. *Monographs on Statistics and applied probability*, page 17, 1993.
- [21] A. F.J, R. g. J, and A. n. M. V. Complete control of an observed confusion matrix. In *IGARSS 2018 - 2018 IEEE International Geoscience and Remote Sensing Symposium*, pages 1222–1225, July 2018.
- [22] Yoav Freund, Robert E Schapire, et al. Experiments with a new boosting algorithm. In *icml*, volume 96, pages 148–156. Citeseer, 1996.
- [23] Zoubin Ghahramani. Unsupervised learning. In *Summer School on Machine Learning*, pages 72–112. Springer, 2003.
- [24] Luis Gonzalez, Cecilio Angulo, Francisco Velasco, and Andreu Catala. Unified dual for bi-class svm approaches. *Pattern Recognition*, 38(10):1772–1774, 2005.
- [25] Karthik Gopalakrishnan and Hamsa Balakrishnan. A comparative analysis of models for predicting delays in air traffic networks. ATM Seminar, 2017.
- [26] Roberto Henriques and Inês Feiteira. Predictive modelling: Flight delays and associated factors, hartsfield–jackson atlanta international airport. *Procedia computer science*, 138:638–645, 2018.
- [27] Sheng-Chen Huang. *Airline schedule recovery following disturbances: An organizationally-oriented decision-making approach*. University of California, Berkeley, 2005.

- [28] J. D. Hunter. Matplotlib: A 2d graphics environment. *Computing in Science & Engineering*, 9(3):90–95, 2007.
- [29] Anil K Jain, Jianchang Mao, and K Moidin Mohiuddin. Artificial neural networks: A tutorial. *Computer*, 29(3):31–44, 1996.
- [30] Anil K Jain, M Narasimha Murty, and Patrick J Flynn. Data clustering: a review. *ACM computing surveys (CSUR)*, 31(3):264–323, 1999.
- [31] Leslie Pack Kaelbling, Michael L Littman, and Andrew W Moore. Reinforcement learning: A survey. *Journal of artificial intelligence research*, 4:237–285, 1996.
- [32] H Khaksar and A Sheikholeslami. Airline delay prediction by machine learning algorithms. 2019.
- [33] Sina Khanmohammadi, Salih Tutun, and Yunus Kucuk. A new multi-level input layer artificial neural network for predicting flight delays at jfk airport. *Procedia Computer Science*, 95:237–244, 2016.
- [34] Thomas Kluyver, Benjamin Ragan-Kelley, Fernando Pérez, Brian E Granger, Matthias Bussonnier, Jonathan Frederic, Kyle Kelley, Jessica B Hamrick, Jason Grout, Sylvain Corlay, et al. Jupyter notebooks—a publishing format for reproducible computational workflows. In *ELPUB*, pages 87–90, 2016.
- [35] Niklas Kohl, Allan Larsen, Jesper Larsen, Alex Ross, and Sergey Tiourine. Airline disruption management—perspectives, experiences and outlook. *Journal of Air Transport Management*, 13(3):149–162, 2007.
- [36] Daphne Koller and Mehran Sahami. Toward optimal feature selection. Technical report, Stanford InfoLab, 1996.
- [37] Sotiris B Kotsiantis, I Zaharakis, and P Pintelas. Supervised machine learning: A review of classification techniques. *Emerging artificial intelligence applications in computer engineering*, 160:3–24, 2007.
- [38] Quansheng Lei, DongQing Jiang, Peng Zhao, and TingWei Ma. Research on the disrupted airline scheduling. In *2013 10th International Conference on Service Systems and Service Management*, pages 332–336. IEEE, 2013.
- [39] Seppo Linnainmaa. The representation of the cumulative rounding error of an algorithm as a taylor expansion of the local rounding errors. *Master’s Thesis (in Finnish), Univ. Helsinki*, pages 6–7, 1970.
- [40] Wes McKinney. pandas: a foundational python library for data analysis and statistics. *Python for High Performance and Scientific Computing*, 14, 2011.
- [41] Avijit Mukherjee, D Lovell, M Ball, A Odoni, and G Zerbib. Modeling delays and cancellation probabilities to support strategic simulations. In *Proceedings of the 6th Europe-USA ATM Seminar. Baltimore. US*, 2005.
- [42] Travis E Oliphant. *A guide to NumPy*, volume 1. Trelgol Publishing USA, 2006.
- [43] David Opitz and Richard Maclin. Popular ensemble methods: An empirical study. *Journal of artificial intelligence research*, 11:169–198, 1999.

- [44] Neha Patki, Roy Wedge, and Kalyan Veeramachaneni. The synthetic data vault. In *2016 IEEE International Conference on Data Science and Advanced Analytics (DSAA)*, pages 399–410. IEEE, 2016.
- [45] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12:2825–2830, 2011.
- [46] Ananda Rakshit, Nirup Krishnamurthy, and Gang Yu. System operations advisor: a real-time decision support system for managing airline operations at united airlines. *Interfaces*, 26(2):50–58, 1996.
- [47] Juan Jose Rebollo and Hamsa Balakrishnan. Characterization and prediction of air traffic delays. *Transportation research part C: Emerging technologies*, 44:231–241, 2014.
- [48] Xudie Ren, Haonan Guo, Shenghong Li, Shilin Wang, and Jianhua Li. A novel image classification method with cnn-xgboost model. In *International Workshop on Digital Watermarking*, pages 378–390. Springer, 2017.
- [49] David E Rumelhart, Richard Durbin, Richard Golden, and Yves Chauvin. Backpropagation: The basic theory. *Backpropagation: Theory, architectures and applications*, pages 1–34, 1995.
- [50] Nicholas G Rupp, George M Holmes, and Jeff DeSimone. Airline schedule recovery after airport closures: empirical evidence since september 11th. Technical report, National Bureau of Economic Research, 2003.
- [51] GVR Sagar and Dr S Venkata Chalam. Evolutionary algorithm for connection weights in artificial neural networks. *International Journal of Electronics and Communication Engineering*, 4(5):517–525, 2011.
- [52] R Saravanan and Pothula Sujatha. A state of art techniques on machine learning algorithms: A perspective of supervised learning approaches in data classification. In *2018 Second International Conference on Intelligent Computing and Control Systems (ICICCS)*, pages 945–949. IEEE, 2018.
- [53] Robert J Schalkoff. *Artificial neural networks*. McGraw-Hill Higher Education, 1997.
- [54] Francisco Jesus Jimenez Serrano and Antonin Kazda. Airline disruption management: yesterday, today and tomorrow. *Transportation Research Procedia*, 28:3–10, 2017.
- [55] Donald F Specht. Probabilistic neural networks. *Neural networks*, 3(1):109–118, 1990.
- [56] Donald F Specht. Enhancements to probabilistic neural networks. In *[Proceedings 1992] IJCNN International Joint Conference on Neural Networks*, volume 1, pages 761–768. IEEE, 1992.
- [57] Banavar Sridhar, Yao Wang, Alexander Klein, and Richard Jehlen. Modeling flight delays and cancellations at the national, regional and airport levels in the united states. In *8th USA/Europe ATM R&D Seminar, Napa, California (USA)*, 2009.

- [58] Dušan Teodorović and Slobodan Guberinić. Optimal dispatching strategy on an airline network after a schedule perturbation. *European Journal of Operational Research*, 15(2):178–182, 1984.
- [59] Dušan Teodorović and Goran Stojković. Model for operational daily airline scheduling. *Transportation Planning and Technology*, 14(4):273–285, 1990.
- [60] Dušan Teodorović and Goran Stojković. Model to reduce airline schedule disturbances. *Journal of Transportation Engineering*, 121(4):324–331, 1995.
- [61] Balasubramanian Thiagarajan, Lakshminarasimhan Srinivasan, Aditya Vikram Sharma, Dinesh Sreekanthan, and Vineeth Vijayaraghavan. A machine learning approach for prediction of on-time performance of flights. In *2017 IEEE/AIAA 36th Digital Avionics Systems Conference (DASC)*, pages 1–6. IEEE, 2017.
- [62] Yufeng Tu, Michael O Ball, and Wolfgang S Jank. Estimating flight departure delay distributions—a statistical approach with long-term trend and short-term pattern. *Journal of the American Statistical Association*, 103(481):112–125, 2008.
- [63] G. van Rossum. Python tutorial. Technical Report CS-R9526, Centrum voor Wiskunde en Informatica (CWI), Amsterdam, May 1995.
- [64] Harry Zhang. The optimality of naive bayes. *AA*, 1(2):3, 2004.
- [65] Xiaojin Jerry Zhu. Semi-supervised learning literature survey. Technical report, University of Wisconsin-Madison Department of Computer Sciences, 2005.



Faculty of Computing, Blekinge Institute of Technology, 371 79 Karlskrona,
Sweden