Chapter – 5
Utilization of AI and ML Prediction Algorithms

5.1 Traffic Control Systems for Smart Cities
5.1.1 IoT-Based Traffic Prediction Models
5.1.2 Machine Learning-Based Traffic Prediction Models

5.2 Machine Learning Predictive Model for Smart Transportation

5.3 Analysis of Machine Learning Models for Smart Transportation
5.3.1 Performance Measure
5.3.2 Error Measures
5.3.3 Cross-Validation Configuration Setting (10-folds) Results
A. Performance Measures
i. Accuracy Measures
ii. Confusion Matrix Parameters – Low Traffic
iii. Confusion Matrix Parameters – Heavy Traffic
B. Error Measure Results
C. Execution Time Results
5.3.4 Cross-Validation Configuration Setting (25-folds) Results
A. Performance Measures
i. Accuracy Measures
ii. Confusion Matrix Parameters – Low Traffic
iii. Confusion Matrix Parameters – Heavy Traffic
B. Error Measure Results
C. Execution Time Results
5.3.5 Cross-Validation Configuration Setting (30% Split) Results
A. Performance Measures
i. Accuracy Measures
ii. Confusion Matrix Parameters – Low Traffic
iii. Confusion Matrix Parameters – Heavy Traffic
B. Error Measure Results
C. Execution Time Results
5.3.6 Consolidated Result
5.3.7 Dominance Chart
5.3.8 Weighted Sum Model Analysis Using Python
5.3.9 Rank and Percentile Method

5.4 Hypothesis Testing Results

5.5 Summary

Artificial Intelligence and Machine Learning prediction algorithms are revolutionizing industries and domains by harnessing data to make precise forecasts and informed decisions. In smart cities it can be used to solve traffic congestion problems, traffic prediction and vehicle maintenance insights, In healthcare, they aid in disease diagnosis and drug discovery, In finance it benefits from risk assessment and stock market predictions, e-commerce relies on recommendation systems and demand forecasting, manufacturing optimizes operations with predictive maintenance and quality control, Energy sector uses it for forecasting consumption and renewable energy utilization, Agriculture improves crop yields and pest detection, Customer service employs chatbots and sentiment analysis, Weather forecasting becomes more accurate, Education adopts personalized learning and student success prediction, all contributing to enhanced efficiency, cost reduction, and better decision-making across various sectors.

5.1 Traffic Control Systems for Smart Cities

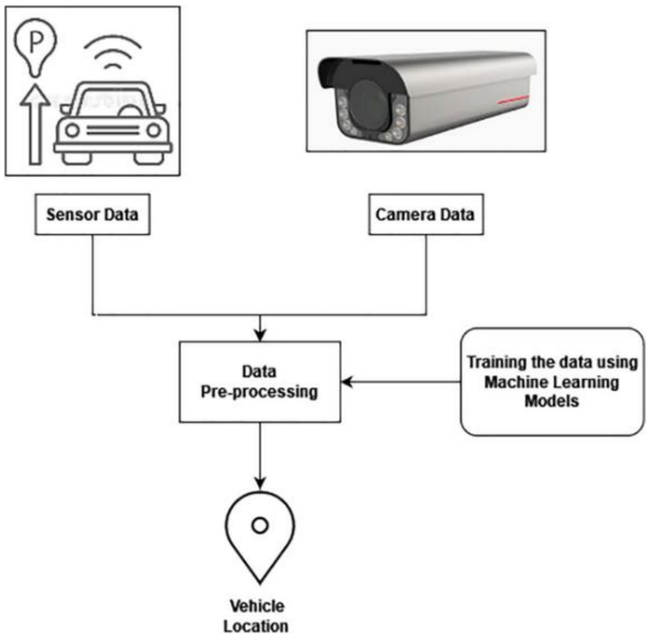
Traffic prediction and control systems in smart cities are essential for managing urban congestion and improving overall transportation efficiency. Various machine learning and IoT-based models have been developed to address these challenges. Some of the existing approaches in this field are:

Adaptive Traffic Signal Control: Using real-time traffic data collected from IoT sensors, adaptive traffic signal control systems can adjust signal timings based on current traffic conditions. These systems aim to minimize congestion and improve traffic flow efficiency.

Intelligent Transportation Systems (ITS): ITS integrates various technologies, including IoT, machine learning, and data analytics, to manage traffic in real-time. It involves strategies such as dynamic route guidance, incident detection, and congestion pricing to optimize traffic control in smart cities.

Predictive Traffic Control: By combining machine learning-based traffic prediction models with control algorithms, predictive traffic control systems can anticipate traffic conditions and adjust signal timings proactively. These systems help prevent congestion before it occurs.

Figure 5.1: Vehicle Location Tracking Using IoT and Machine Learning



The Figure 5.1 shown above explains the vehicle tracking system using IOT and Machine learning. Various IoT sensors, like Camera, GPS Probes, Motion Sensors etc. are used to collect raw traffic data, which is preprocessed for missing values and other non-linearities. Collected data is used for training the machine learning model, which will be used to forecast vehicle location on live data in future.

5.1.1 IoT-Based Traffic Prediction Models

Today traffic density is increasing in smart cities due to rise in population. This traffic rise results in time wastage, fuel wastage, environmental problems and casualties. Many solutions and methods are suggested in the past by many researchers but they lack in accuracy and reliability. IoT based traffic prediction models have shown us new ray of hope to overcome congestion problems in smart cities. Some of the IoT methods are described below.

a. Sensor Networks: IoT devices and sensors deployed across road networks can collect real-time data on traffic flow, vehicle speeds, and occupancy. By analyzing this data, traffic prediction models can provide accurate and up-to-date traffic forecasts.

b. **Vehicle-to-Infrastructure Communication:** IoT-enabled vehicles can communicate with smart infrastructure systems, such as traffic lights and road sensors, to gather real-time data. This information can be used to predict traffic patterns and optimize traffic control strategies.

These IoT models integrates data from different sources, including traffic cameras, GPS devices, and proximity sensors to predict and optimize vehicle density, traffic congestion, vehicle routing and improve the overall transportation experience.

5.1.2 Machine Learning-Based Traffic Prediction Models

In last one decade machine learning algorithms are increasingly becoming popular to solve traffic congestion problems due to static and error prone traditional statistical methods. Today's ever increasing city limits and population growth, has made city traffic management very difficult and demanding. Improvement in Machine learning technology is a new ray of hope for us. Some of the machine learning technologies used are described below.

a. Time-Series Forecasting Models: Models like ARIMA[1] and SARIMA[2] use historical traffic data to predict future traffic patterns. They consider factors like time of day, day of the week, and seasonality to forecast traffic conditions accurately.

b. Artificial Neural Networks: ANNs, such as MLP[3] and RNNs[4] like Long Short-Term Memory, are capable of learning complex patterns in traffic data. These models can capture temporal dependencies and perform well in long-term traffic prediction.

c. Support Vector Machines: SVMs are used for both classification and regression tasks. They can be employed to predict traffic conditions based on historical data, considering features like weather, events, and road characteristics.

[1] Auto Regressive Integrated Moving Average
[2] Seasonal Auto Regressive Integrated Moving Average
[3] Multilayer Perceptron
[4] Recurrent Neural Networks

d. Random Forests: Random Forest models combine multiple decision trees to make predictions. They can handle both numerical and categorical features, making them suitable for traffic prediction tasks involving multiple input variables.

In summary above set of algorithms have the capability to learn to perform tasks such as prediction and classification effectively using data. Learning is achieved using additional data and or additional models. A machine learning algorithm uses the following steps:

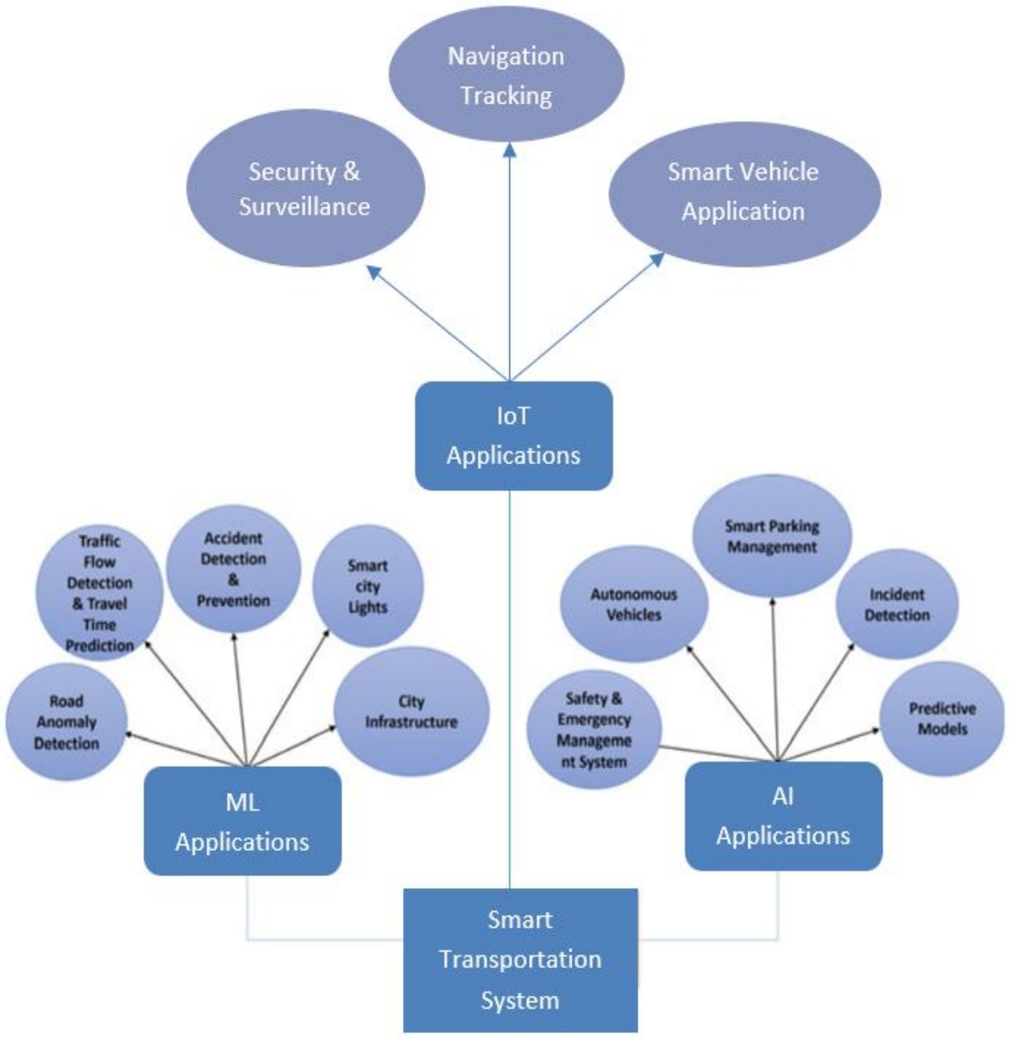
1. Identify the problems.
2. Identify sources of information / data.
3. Pre-process the data for missing and incorrect data and transform the data if required.
4. Divide the data into training and testing datasets.
5. Build ML models and identify the best model performance in validation data.
6. Implement solution / develop product.

An algorithm can be called as learning algorithm when it improves on a performance metric while performing a task.

5.2 Machine Learning Predictive Model for Smart Transportation

A Machine Learning Predictive Model for a Smart Transportation System integrates data from diverse sources, including traffic cameras, GPS devices, and weather forecasts, to predict traffic congestion, optimize vehicle routing, enhance public transportation efficiency, and improve the overall transportation experience. By analyzing historical and real-time data, these models offer solutions such as rerouting traffic, suggesting efficient routes for logistics, predicting public transportation demand, enabling predictive maintenance, promoting sustainability, enhancing security, and informing government policies, ultimately revolutionizing the way transportation is managed and transforming urban mobility for the better.

Figure 5.2: Smart Transportation System



The Smart Transportation System is being divided into three modules which includes ML applications, AI applications and IoT applications as shown in Figure 5.2. These modules are further being divided into submodules such as the ML application module includes traffic flow detection and travel time prediction, accident detection and prevention, smart city lights city, infrastructure and road anomaly detection. Similarly, the AI application is being subdivided into safety and emergency management, autonomous vehicles, smart parking management, incident detection and predictive models. The IoT applications include security surveillance, smart vehicle application and navigation.

5.3 Analysis of Machine Learning Models for Smart Transportation

Machine Learning Predictive Models for Smart Transportation provide data-driven insights and solutions to address complex urban mobility challenges. These models harness vast datasets from traffic sensors, GPS devices, and various sources to forecast traffic patterns, optimize routes, and improve public transportation systems. By leveraging historical and real-time data, they enable more efficient traffic management, reduce congestion, enhance user experience, and promote sustainable transportation practices. Additionally, these predictive models have the potential to play a pivotal role in shaping future transportation policies and infrastructure development for smarter and more accessible cities. Various Algorithms were used for feature/attribute extraction and selection. Based on the results provided by Algorithms out of twenty one attributes following seven attributes are selected for Machine Learning Algorithms.

1. SPEED
2. NUM_READS
3. HOUR
4. ZIP CODES
5. REGION
6. BUS_COUNT
7. CLASS LABEL

5.3.1 Performance Measure

To analyze different prediction models, the performance measure like accuracy, incorrectly classified instances, Kappa statistic, precision, recall, F-measure , ROC Area ,TP Rate, FP Rate, precision and recall were being used, which are explained below.

Accuracy: It is a commonly used metric to evaluate the performance of machine learning classification models. It measures the ratio of correctly predicted instances to the total number of instances in the dataset. The formula for precision is:

$$\text{Accuracy} = \frac{Number\ of\ Correct\ Predicted\ values}{Total\ Number\ of\ instances\ in\ the\ data\ set} \qquad [i]$$

Incorrectly Classified Instances: In machine learning, a misclassified instance is a data point or instance in a data set that is incorrectly predicted or labeled by a machine learning model. These instances represent errors that the model made in its predictions. There are two types of errors:

- **False Positive:** Negative Instances are predicted as Positive by Model. In terms of Traffic Congestion this can be low traffic is predicted as heavy traffic.

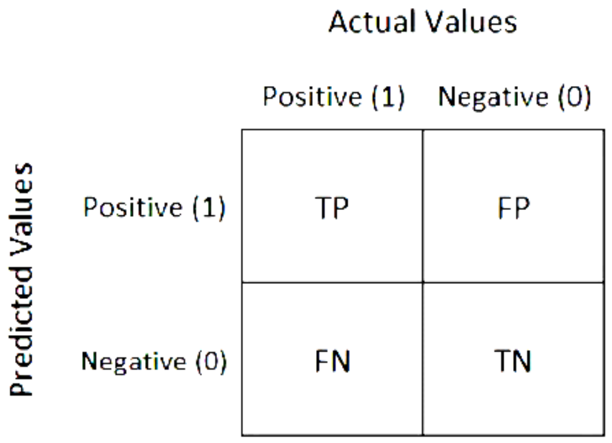
- **False Negative:** Positive Instances are predicted as Negative by Model. In terms of Traffic Congestion this can be heavy traffic is predicted as low traffic.

Kappa Statistics: The kappa statistic, also known as Cohen's kappa, is a measure of agreement or reliability between classification algorithms. It is often used to assess agreement between two classification algorithms. Kappa value ranges from -1 to +1. Positive 1 indicates perfect agreement and Negative 1 indicates worst agreement. The various agreement interpretations are given below.

- Kappa > 0.8: Excellent Agreement
- 0.6 < Kappa < 0.8: Good Agreement
- 0.4 < Kappa < 0.6: Moderate Agreement
- Kappa ≤ 0.4: Poor Agreement

Confusion Matrix Parameters: Confusion matrix shows the different ways in which the classification model gets confused when making predictions. The predicted values are compared with Actual values to find out various performance parameters. The Parameter like TP Rate, FP Rate , Precision, Recall and ROC Area are derived from a confusion matrix.

Figure 5.3: Confusion Matrix



Source: Towards Data Science [23]

TP[5]: It refers to the number of predictions where the classifier correctly predicts the positive class as positive. For example in terms of Traffic Congestion this can be heavy traffic is predicted as heavy traffic.

TN[6]: It refers to the number of predictions where the classifier correctly predicts the negative class as negative. For example in terms of Traffic Congestion this can be low traffic is predicted as low traffic.

FP[7]: It refers to the number of predictions where the classifier incorrectly predicts the negative class as positive. For example in terms of Traffic Congestion this can be low traffic is predicted as heavy traffic.

FN[8]: It refers to the number of predictions where the classifier incorrectly predicts the positive class as negative. For example in terms of Traffic Congestion this can be heavy traffic is predicted as low traffic.

Precision: It is the quality of a positive prediction made by the model. Precision refers to the number of True Positives divided by the total number of Positive predictions.

$$\text{Precision} = \frac{TP}{TP+FP} \qquad [ii]$$

Recall: It is the measures of how well a machine learning model can detect positive instances. It is also called as Sensitivity. Sensitivity refers to the number of true

[5] True Positive
[6] True Negative
[7] False Positive
[8] False Negative

positives divided by the sum of True Positives and False Negatives. The model with high Sensitivity will have significantly fewer False Negatives.

$$\text{Recall} = \frac{TP}{TP+FN} \qquad \text{[iii]}$$

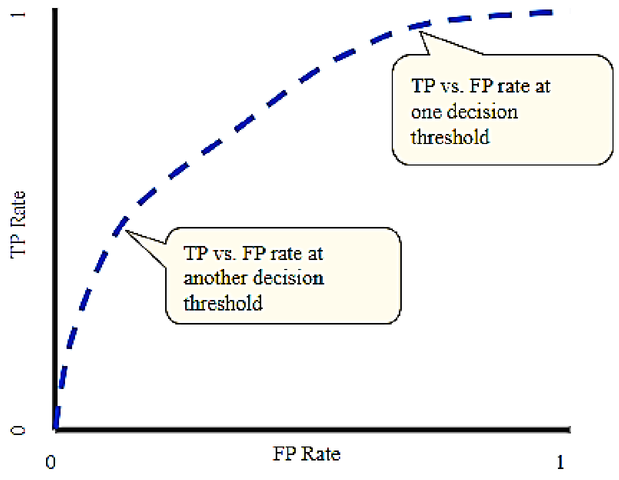
F-Measure: It also known as F1-score, It is a machine learning metric that combines precision and recall into one value.

$$\text{F-Measure} = 2 \times \frac{Presion \times Recall}{Precision+Recall} \qquad \text{[iv]}$$

F-Measure value changes between 0 and 1, Where 1 indicates ideal Precision and Recall and 0 indicates poor performance.

ROC[9]: It is a graph showing the performance of a classification mode. This curve plots two parameters: TPR[10] and FPR[11]. TP Rate is used to measure the percentage of actual positives which are correctly identified by model . TPR is synonym for Recall. FP Rate also known as Type - I error is used to measure the percentage of actual positives which are incorrectly identified by model.

Figure 5.4: ROC Curve

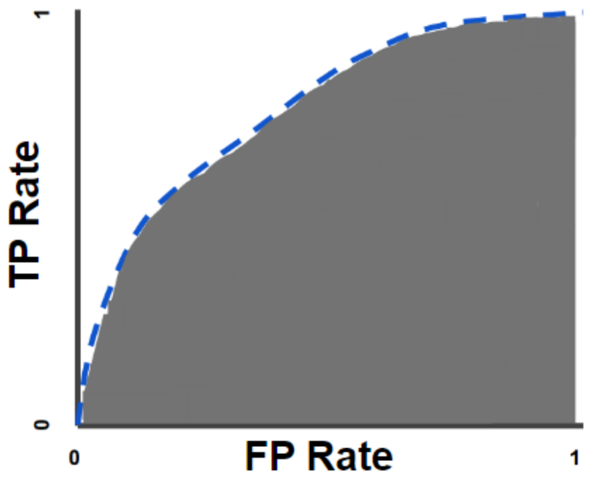


Source: Google developer site [27]

[9] Receiver Operating Characteristics
[10] True Positive Rate
[11] False Positive Rate

ROC Area: AUC[12] measures the entire two-dimensional area underneath the entire ROC curve. AUC indicates how well predictions are ranked. AUC ranges in value from 0 to 1. A model whose predictions are 100% wrong has an AUC of 0.0; one whose predictions are 100% correct has an AUC of 1.0.

Figure 5.5: Area Under ROC Curve



Source: Google developer site [27]

In summary, performance measurements play a critical role in evaluating the effectiveness of machine learning algorithms, providing insight into their ability to make accurate predictions and transform appropriately to new, unseen data. Choosing the most appropriate metric depends on the nature of your particular problem, the characteristics of your data, and your analysis goals.

5.3.2. Error Measures

Machine learning uses various error measures to evaluate the performance of algorithms and models. These measurements help quantify the difference between predicted and actual values and provide insight into model performance. Common error remedies includes:

Mean Absolute Error: it is Average of absolute differences between Actual values and Predicted values. Lower the value of Mean Absolute Error, better the Prediction Algorithm.

[12] Area Under ROC Curve

Mean Absolute Error = $\frac{1}{n}\sum_{i=1}^{n}|y_{a_i} - y_{p_i}|$ [v]

Where

- y_a = Actual Output Value
- y_p = Predicted Output Value

Root Mean Square Error: It is the root of mean square error. It considers the effect of negative as well positive errors in considerations.

Root Mean Square Error = $\sqrt{\frac{1}{n}\sum_{i=1}^{n}(y_{a_i} - y_{p_i})^2}$ [vi]

Relative Absolute Error: It is used to measure accuracy of predictions. It compares the distance of actual values from predicted values with distance of actual values from average values. Its value lie between 0 to positive infinite. A lower Relative Absolute Error indicates better performance.

Relative Absolute Error = $\frac{\sum_{i=1}^{n}|y_{a_i} - y_{p_i}|}{\sum_{i=1}^{n}|y_{a_i} - \bar{y}|}$ [vii]

Where

$\bar{y}$ = Mean of Actual Values

Root Relative Square Error: It is the root of relative square error. The relative square error is the ratio of total square error to average of actual values. Relative square error normalizes total square error.

Relative Square Error = $\frac{\sum_{i=1}^{n}(y_{a_i} - y_{p_i})^2}{\bar{y}}$ [viii]

By Taking root of Relative Square Error we are reducing the normalize square error value and bringing it closer to predicted value.

Root Relative Square Error = $\sqrt{Relative\ Square\ Erro}$ [ix]

Root Relative Square Error = $\sqrt{\frac{\sum_{i=1}^{n}(y_{a_i} - y_{p_i})^2}{\bar{y}}}$ [x]

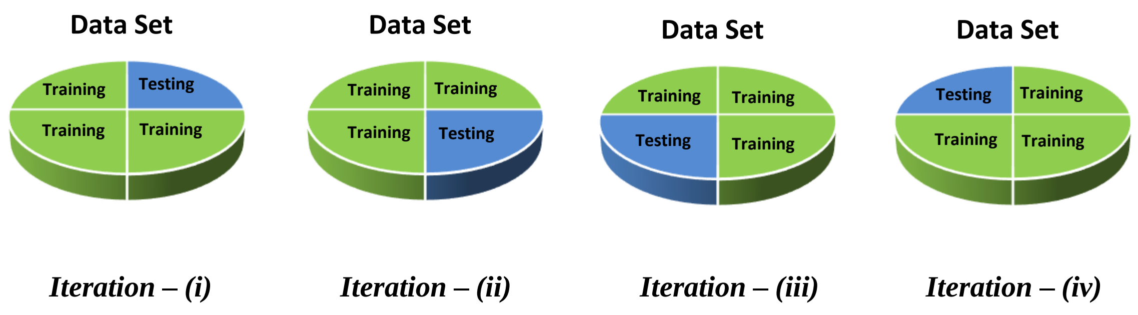
Lower the value of Root Relative Square Error, better the performance of Machine Learning prediction model. The choice of error measure depends on the nature of the problem and the specific objectives of the analysis. When choosing an appropriate error measure, it is important to consider the characteristics of the data and the objectives of the modeling task.

In summary, understanding and effectively using error counter measures is a fundamental aspect of creating and improving machine learning algorithms. Error measurements provide a quantitative means of assessing model accuracy and performance, allowing practitioners to make informed decisions, compare different algorithms, and optimize predictive capabilities.

5.3.3 Cross-Validation Configuration Setting (10-folds) Results

The Weka tool was being used for the analysis of various classification machine learning algorithms. K-folds Cross-Validation Approach is used for evaluating the Performance of Machine learning Algorithms, where K value is changed to study different cases. For example in 4 folds Cross-Validation K value is 4 where data set is divided into 4 parts, out of which 3 parts are used for training the Machine learning Algorithm and only One part is used for testing the Algorithm.

Figure 5.6: 4-fold Cross-Validation Example



As shown in above figure 5.6, four iterations are executed for 4-folds, In first iteration part one will be used for testing and remaining three parts will be used for training. In second iteration part two will be used for testing and remaining three parts will be used for training. This recursion will continue up to the last iteration to complete the Cross-Validation. For 10-fold Cross-Validation K value is 10 and data set will be divided into 10 parts, Nine parts for Training the algorithm and One part for testing the algorithm.

Similarly for 25-fold Cross-Validation K value is 25 and data set will be divided into 25 parts, Twenty four parts for Training the algorithm and One part for testing the algorithm.

Data Analysis Credentials:

Dataset: Udaipur_Traffic **Source:** TOMTOM Server

Date: October 2023 **Duration:** One Month

Number of Instances: 1000

Number of Attributes (After Feature Extraction and Selection): 7

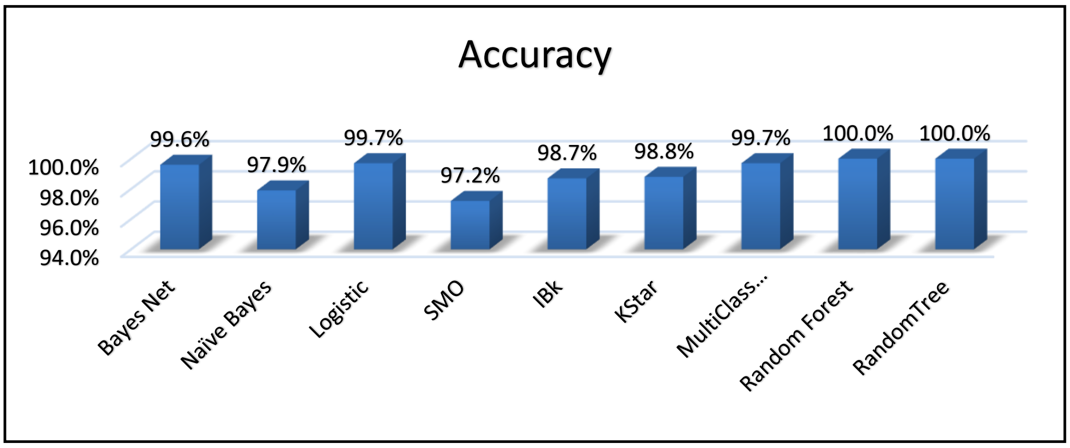
A. Performance Measures

i. Accuracy Measures

Table 5.1: Classifiers and Accuracy Measures (Cross Validation: 10-Folds)

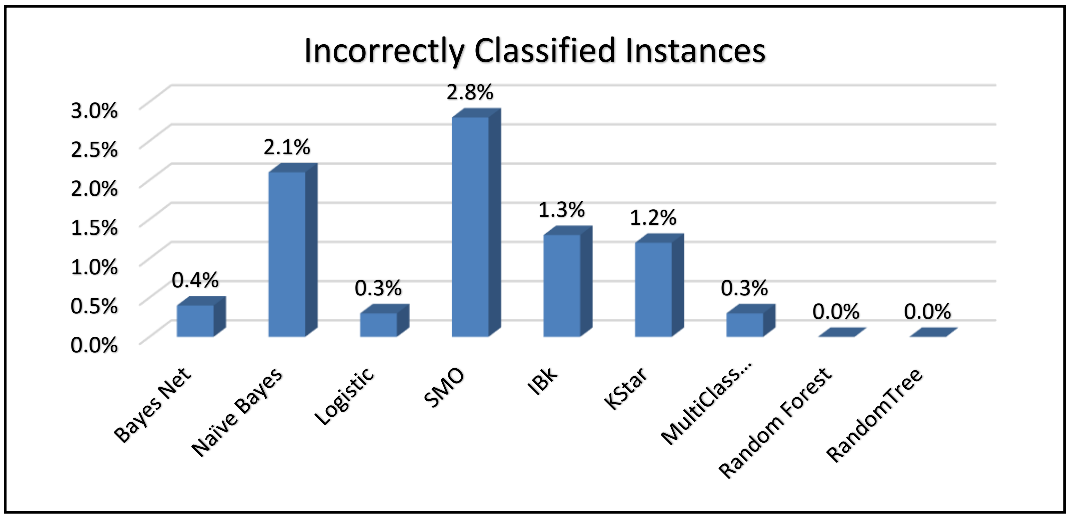
Classifier	Accuracy	Incorrectly Classified Instances	Kappa statistic
Bayes Net	99.6%	0.4%	0.967
Naïve Bayes	97.9%	2.1%	0.815
Logistic	99.7%	0.3%	0.976
SMO	97.2%	2.8%	0.722
IBk	98.7%	1.3%	0.898
KStar	98.8%	1.2%	0.905
MultiClass Classifier	99.7%	0.3%	0.976
Random Forest	100.0%	0.0%	1.000
RandomTree	100.0%	0.0%	1.000

Figure 5.7: Performance Measure Accuracy (Cross-Validation: 10 Folds)



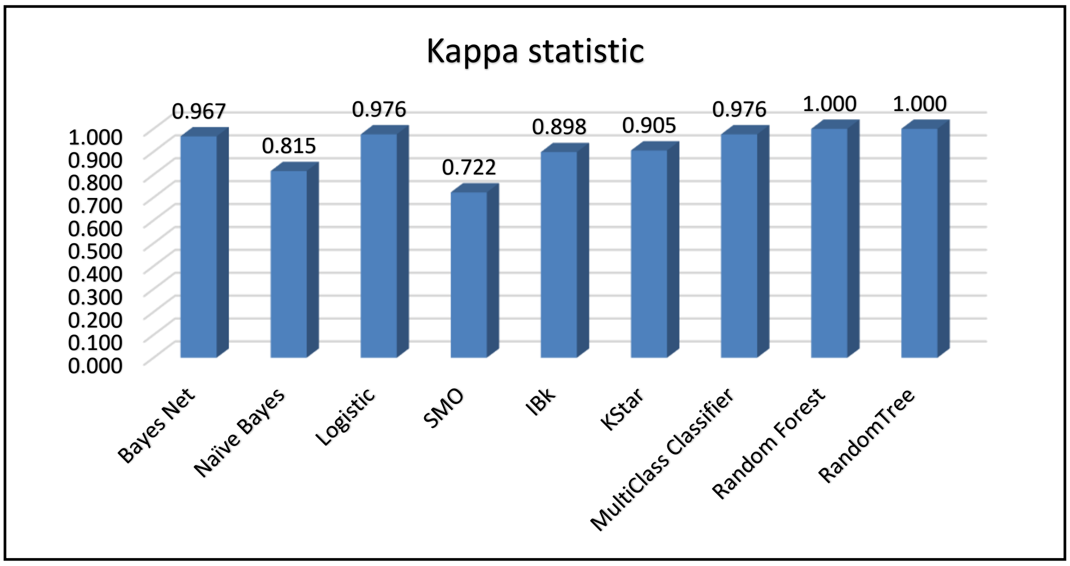
Based on the performance measure accuracy it can be interpreted that Random Forest classifier was the most appropriate one as it was having the highest accuracy value of 100% whereas SMO and Naïve Bayes were having the lowest value of accuracy 97.2% and 97.9%.

Figure 5.8: Incorrectly Classified Instances (Cross-Validation: 10 Folds)



According to the performance measure incorrectly classified instances it can be interpreted that Random Forest and Random Tree classifiers were the most appropriate one as these were having the lowest number of incorrectly classifies instances accounting for 0% each whereas SMO and Naïve Bayes classifiers were having the highest number of incorrectly classifies instances accounting for 2.8% and 2.1% respectively.

Figure 5.9: Kappa Statistic Values (Cross-Validation: 10 Folds)



The provided data consists of a set of Kappa statistic values, which are used to assess the agreement or consistency between classifiers in different situations. These Kappa values range from 0.722 to 1.000, indicating varying levels of agreement. The highest Kappa value, 1.000, suggests a very good level of agreement between the classifiers in that particular scenario, while the lowest value, 0.722, falls into the good to moderate agreement range. Overall, the data suggests that there is a generally positive level of agreement in the assessed situations, with some instances demonstrating higher agreement than others.

In correctly classified instances and kappa statistics are performance metrics which are important to explore to get more comprehensive insights to develop accurate machine learning model.

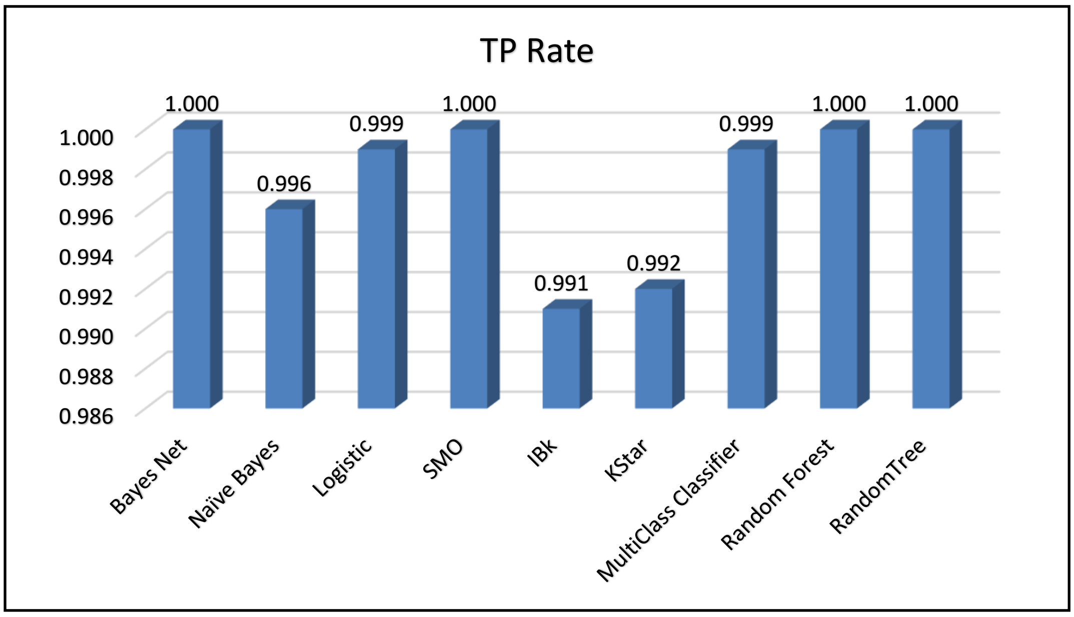
ii. Confusion Matrix Parameters – Low Traffic

Table 5.2 shows confusion matrix parameters TP Rate, FP Rate, Precision, Recall, F-Measure and ROC Area for Low Traffic case. Machine learning often requires a limited amount of data when dealing with low-traffic scenarios. In such cases, the challenge is to create a robust model despite data limitations. Data Augmentation techniques are used to artificially increase the size of data set which can be helpful especially in low traffic scenarios.

Table 5.2: Classifiers and Performance Measures Class Label: Low Traffic
Cross Validation: 10-Folds

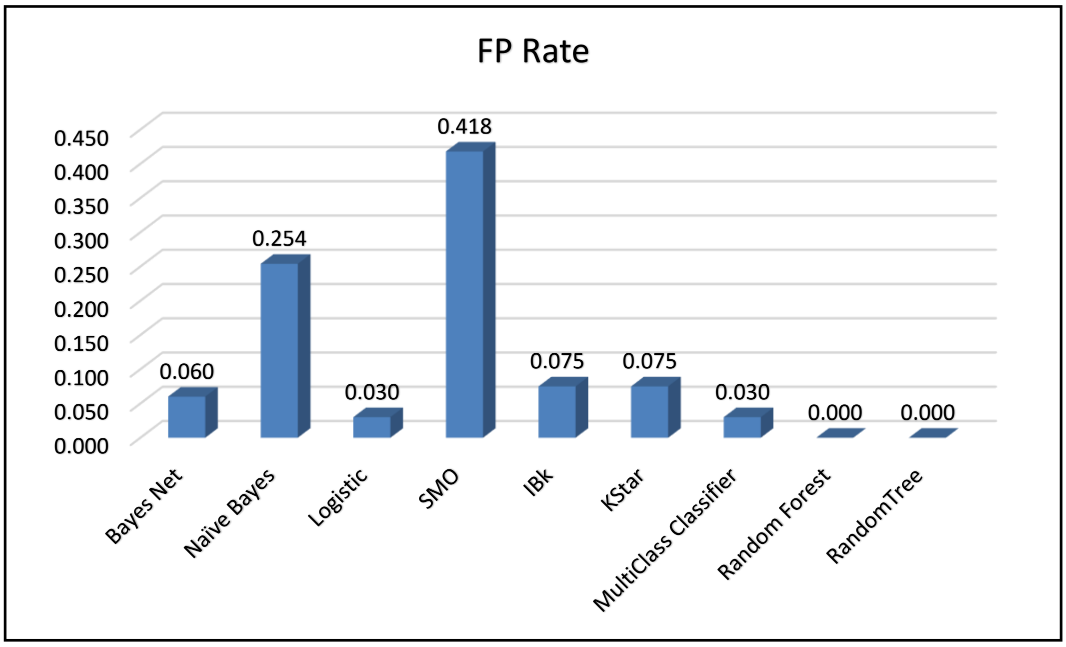
Classifier	TP Rate	FP Rate	Precision	Recall	F-Measure	ROC Area
Bayes Net	1.000	0.060	0.996	1.000	0.998	1.000
Naïve Bayes	0.996	0.254	0.982	0.996	0.989	0.988
Logistic	0.999	0.03	0.998	0.999	0.998	1.000
SMO	1.000	0.418	0.971	1.000	0.985	0.791
IBk	0.991	0.075	0.995	0.991	0.993	0.958
KStar	0.992	0.075	0.995	0.992	0.994	0.997
MultiClass Classifier	0.999	0.030	0.998	0.999	0.998	1.000
Random Forest	1.000	0.000	1.000	1.000	1.000	1.000
RandomTree	1.000	0.000	1.000	1.000	1.000	1.000

Figure 5.10: TP Rate (Cross Validation: 10-Folds – Low Traffic)



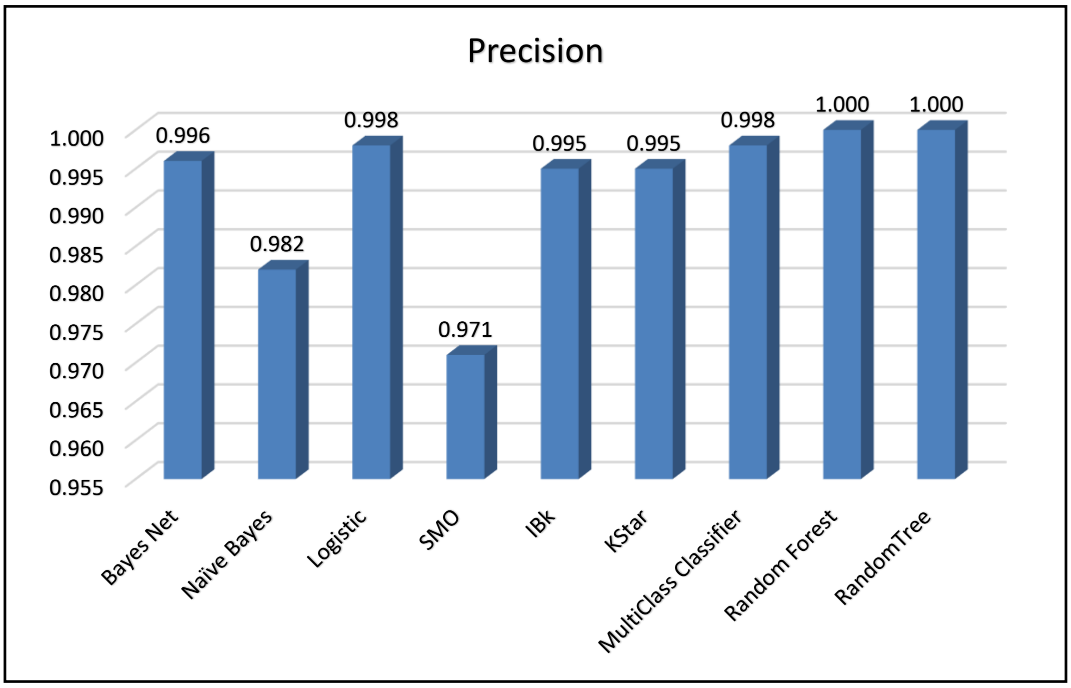
According to the performance measure TP rate it was found that the highest true positive rate was of the classifiers Bayes Net, Random Forest and Random Tree with value 1.000, followed by 0.999 and 0.999 of Logistic and MultiClass Classifier respectively whereas the lowest TP rate was found to be of the classifiers IBK with value 0.991. Overall it can be interpreted that there are three most appropriate classifiers based on the performance measure TP rate with the value of 1.000.

Figure 5.11: FP Rate (Cross Validation: 10-Folds – Low Traffic)



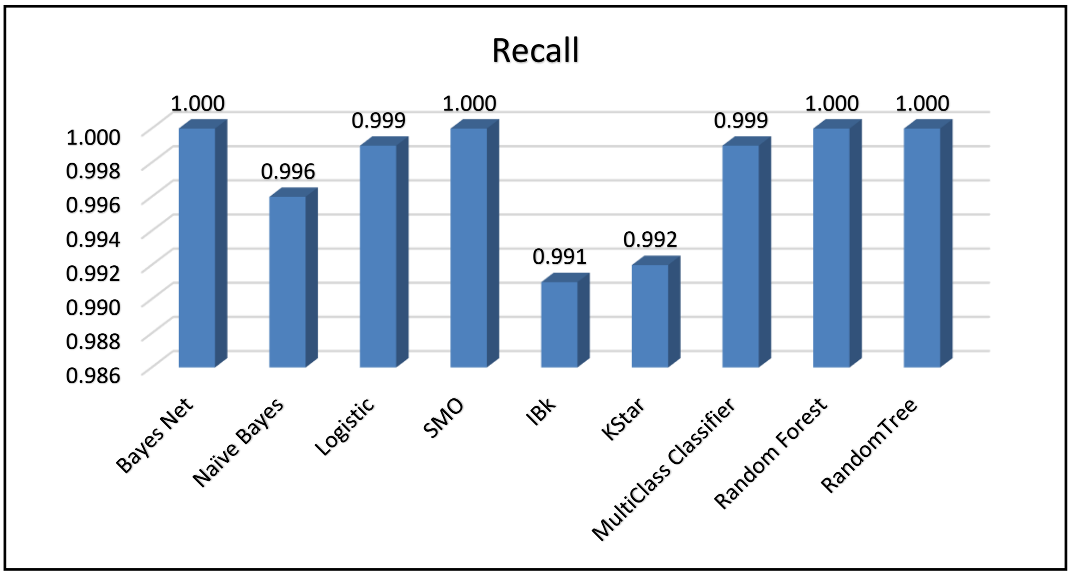
Based on the performance measure FP rate it was found that the lowest false positive rate was of the classifiers Random Forest and Random Tree with value 0.000, followed by 0.030 of Logistic and MultiClass Classifier whereas the highest FP rate was found to be of the classifiers SMO with value 0.418. Overall, it can be interpreted the most appropriate classifier based on the performance measure FP rate is found to be Random Forest and Random Tree with lowest FP rate value.

Figure 5.12: Precision (Cross Validation: 10-Folds – Low Traffic)



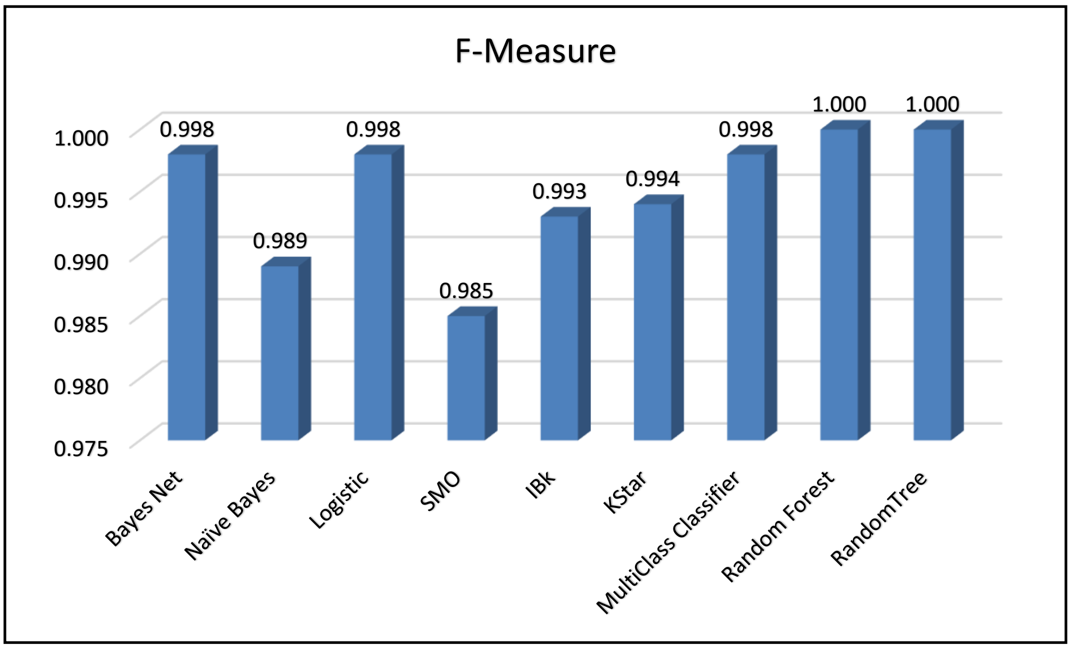
According to the performance measure precision it was found that the highest precision value was of the classifier Random Forest and Random Tree with value 1.000, followed by 0.998 of Logistic and MultiClass Classifier respectively whereas the lowest precision value was found to be of the classifier SMO with value 0.971. Overall, it can be interpreted the most appropriate classifiers based on the performance measure precision are found to be Random Forest and Random Tree.

Figure 5.13: Recall (Cross Validation: 10-Folds – Low Traffic)



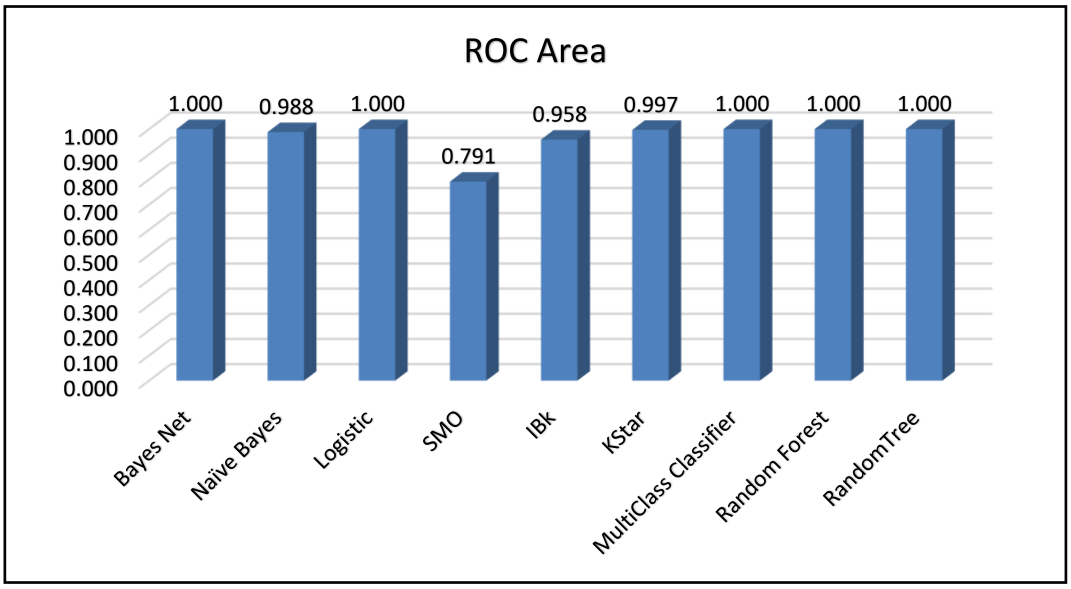
Based on the performance measure recall it was found that the highest recall value was of the classifiers Bayes Net, SMO, Random Forest and Random Tree with value 1.000, followed by 0.999 of Logistic and MultiClass Classifier respectively whereas the lowest recall value was found to be of the classifier IBK with value 0.991. Overall, it can be interpreted the most appropriate classifier based on the performance measure recall is found to be four algorithms Bayes Net, SMO, Random Forest and Random Tree.

Figure 5.14: F-Measure (Cross Validation: 10-Folds – Low Traffic)



According to the performance measure F-Measure it was found that the highest F-Measure values were of the classifier Random Forest and Random Tree with value 1.000, followed by 0.998 value of Bayes Net, Logistic and MultiClass Classifier respectively whereas the lowest F-Measure value was found to be of the classifier SMO classifier with values 0.985. Overall, it can be interpreted the most appropriate classifiers based on the performance measure F-Measure is found to be Random Forest and Random Tree.

Figure 5.15: ROC Area (Cross Validation: 10-Folds – Low Traffic)



Based on the performance measure ROC it was found that the highest ROC Area value was of the classifiers Bayes Net, Logistic, MultiClass Classifier, Random Forest and Random Tree with value 1.000, followed by 0.997 and 0.988 of KStar and Navie Bayes respectively whereas the lowest ROC Area value was found to be of the classifier SMO with value 0.791 respectively. Overall, it can be interpreted the most appropriate classifiers based on the performance measure ROC Area are found to be five Algorithms.

In summary, performance measurements play a critical role in evaluating the effectiveness of machine learning algorithms, providing insight into their ability to make accurate predictions and transform appropriately to new, unseen data. Choosing the most appropriate metric depends on the nature of your particular problem, the characteristics of your data, and your analysis goals.

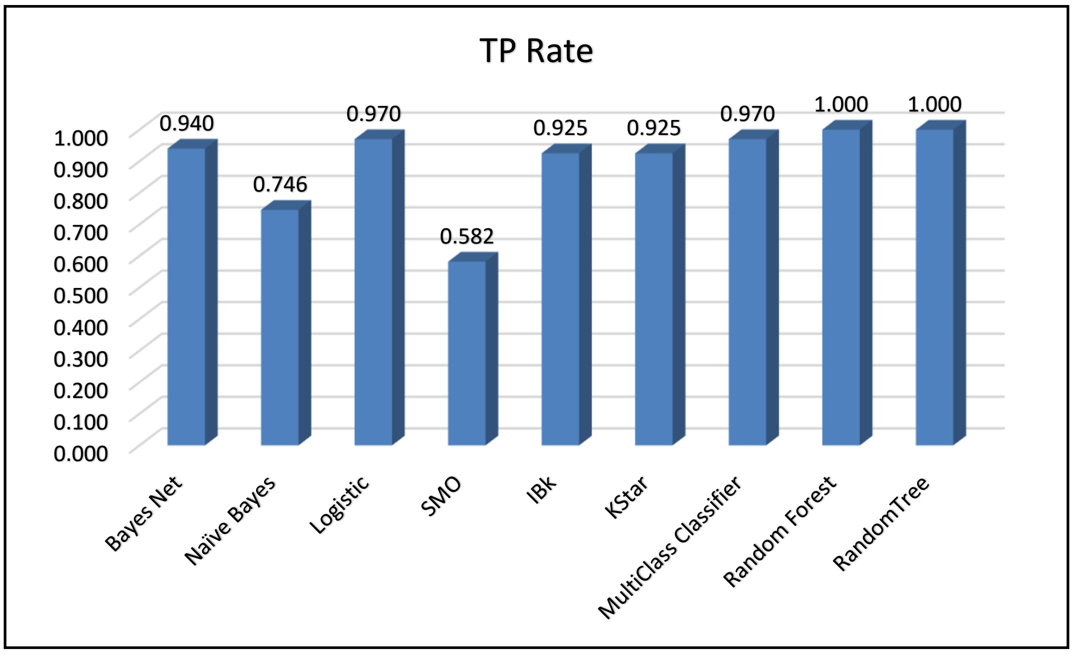
iii. Confusion Matrix Parameters – Heavy Traffic

Heavy Traffic generates huge amounts of data from the various IOT sensors. These data sets can be used by Machine Learning Algorithms to develop prediction models. Table 5.3 shows the Confusion Matrix parameters obtained for Heavy Traffic conditions.

Table 5.3: Classifiers Performance Measure Class Label: Heavy Traffic
Cross Validation: 10-Folds

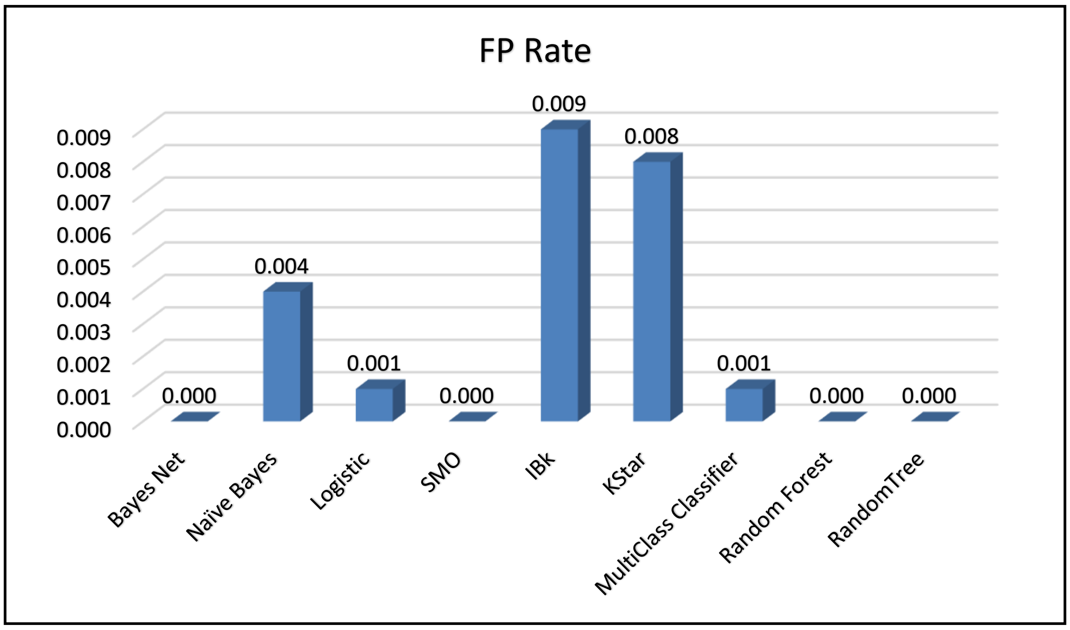
Classifier	TP Rate	FP Rate	Precision	Recall	F-Measure	ROC Area
Bayes Net	0.940	0.000	1.000	0.940	0.969	1.000
Naïve Bayes	0.746	0.004	0.926	0.746	0.826	0.988
Logistic	0.970	0.001	0.985	0.970	0.977	1.000
SMO	0.582	0.000	1.000	0.582	0.736	0.791
IBk	0.925	0.009	0.886	0.925	0.905	0.958
KStar	0.925	0.008	0.899	0.925	0.912	0.997
MultiClass Classifier	0.970	0.001	0.985	0.970	0.977	1.000
Random Forest	1.000	0.000	1.000	1.000	1.000	1.000
RandomTree	1.000	0.000	1.000	1.000	1.000	1.000

Figure 5.16: TP Rate (Cross Validation: 10-Folds – Heavy Traffic)



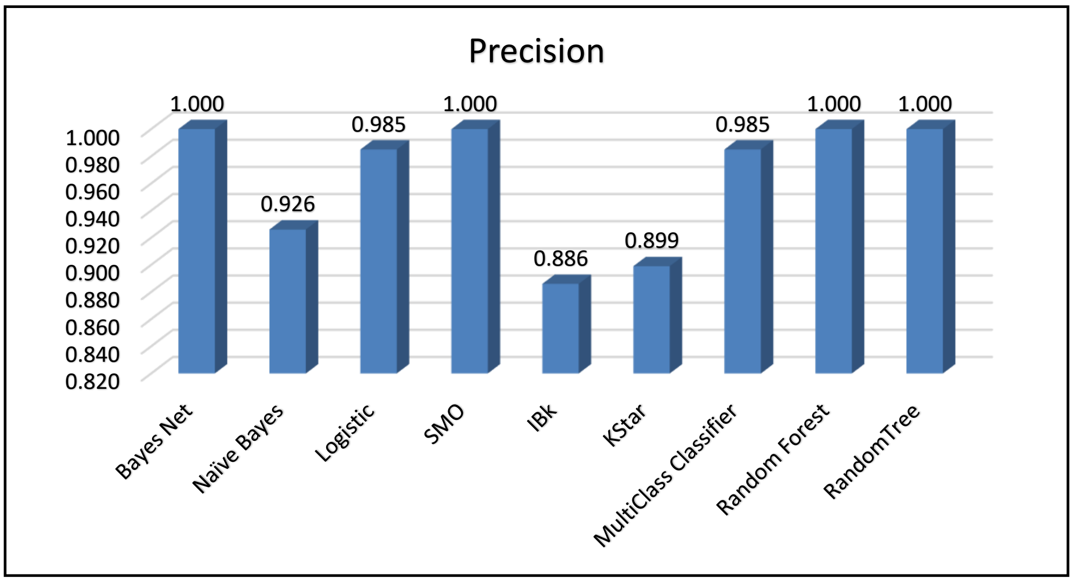
According to the performance measure TP rate for class label: Heavy Traffic it was found that the highest true positive rate was of the classifiers Random Forest and Random Tree 1.000, followed by 0.970 of Logistic and MultiClass Classifier respectively whereas the lowest TP rate was found to be of the classifiers SMO with values 0.582. Overall, it can be interpreted the most appropriate classifier based on the performance measure TP rate are Random Forest and Random Tree.

Figure 5.17: FP Rate (Cross Validation: 10-Folds – Heavy Traffic)



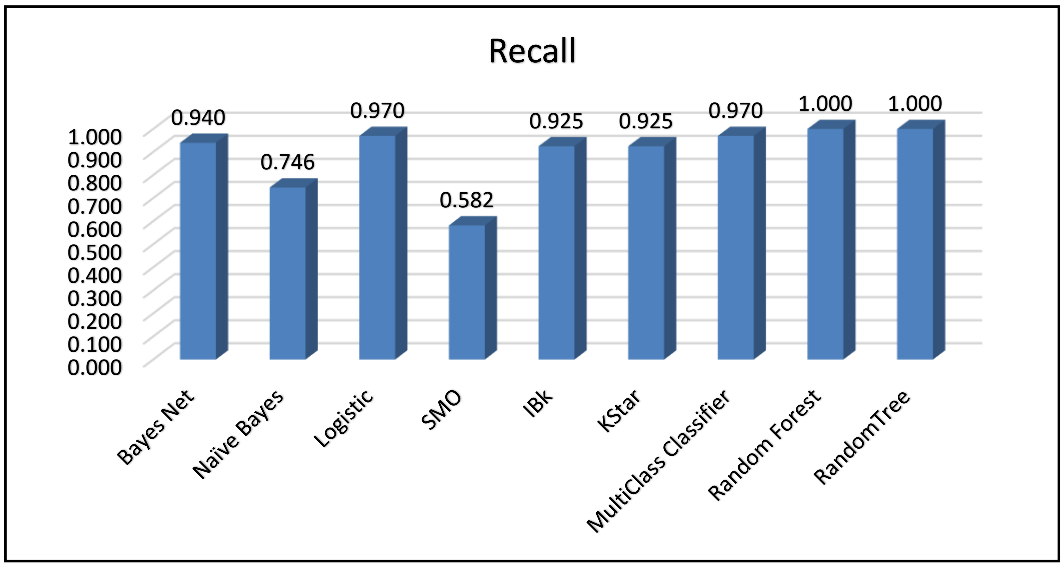
Based on the performance measure FP rate for class label: Heavy Traffic it was found that the lowest false positive rate was of the classifiers Bayes Net, SMO, Random Forest and Random Tree with value 0.00, followed by 0.001 of Logistic and MultiClass Classifier, whereas the highest FP rate was found to be of the classifiers KStar and IBK with values 0.008 and 0.009 respectively. Overall, it can be interpreted the most appropriate classifier based on the performance measure FP rate are found to be four Algorithms with lowest FP rate value 0.000.

Figure 5.18: Precision (Cross Validation: 10-Folds – Heavy Traffic)



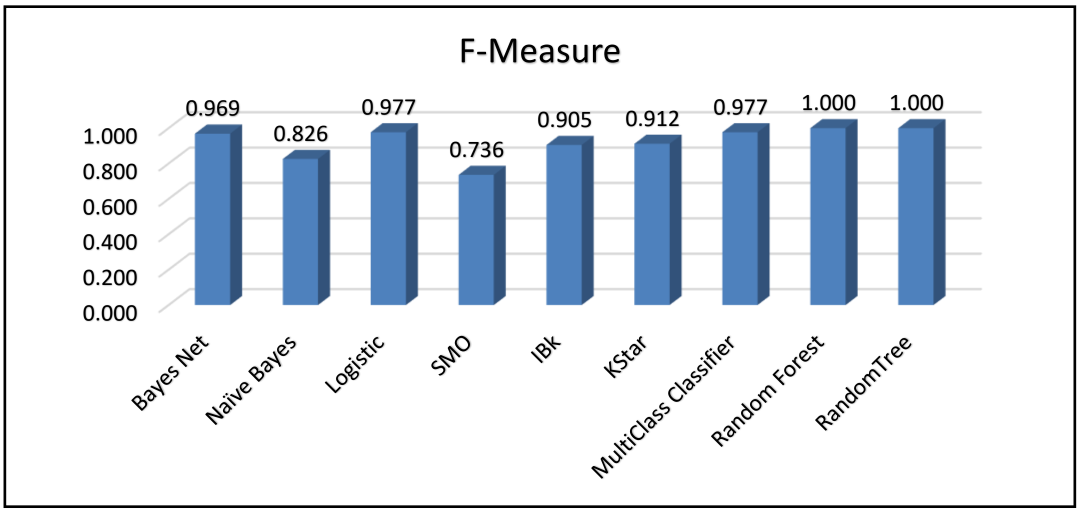
According to the performance measure precision class label: Heavy Traffic it was found that the highest precision value was of the classifiers Bayes Net, SMO, Random Forest and Random Tree with value 1.000, followed by 0.985 and 0.926 of Logistic, MultiClass Classifier and Naïve Bayes respectively whereas the lowest precision value was found to be of the classifier IBK with values 0.886 respectively. Overall, it can be interpreted that the most appropriate classifier based on the performance measure precision is found to be Four Algorithms.

Figure 5.19: Recall (Cross Validation: 10-Folds – Heavy Traffic)



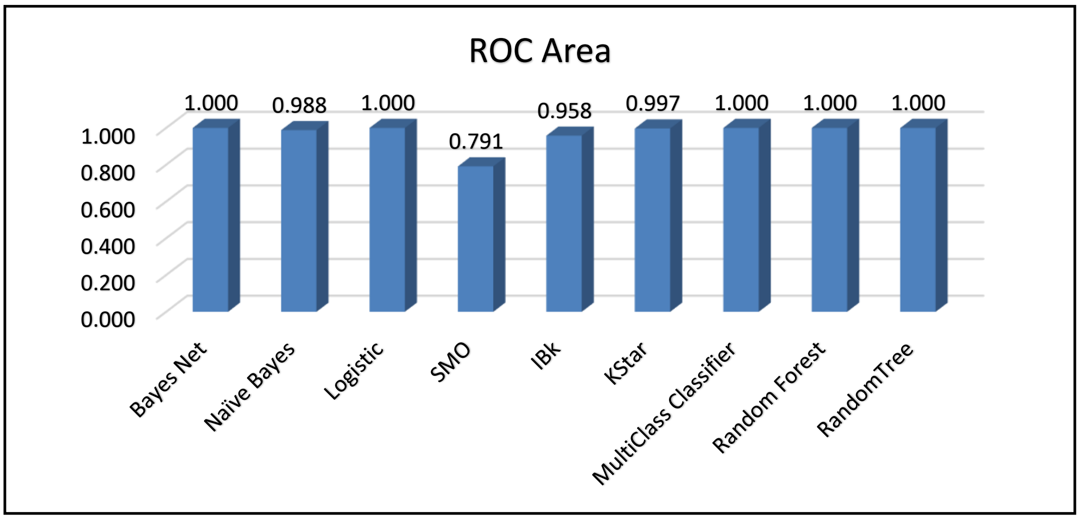
Based on the performance measure recall class label: Heavy Traffic it was found that the highest recall value was of the classifiers Random Forest and Random Tree with value 1.000, followed by 0.970 of Logistic and Multiclass Classifier respectively whereas the lowest recall value was found to be of the classifiers SMO with value 0.582. Overall, it can be interpreted the most appropriate classifier based on the performance measure recall are found to be Random Forest and Random Tree.

Figure 5.20: F-Measure (Cross Validation: 10-Folds – Heavy Traffic)



According to the performance measure F-Measure class label: Heavy Traffic it was found that the highest F-Measure value was of the classifiers Random Forest and Random Tree with value 1.000, followed by 0.9777 of Logistic and Multiclass Classifier respectively whereas the lowest F-Measure value was found to be of the classifier SMO with value 0.736. Overall, it can be interpreted the most appropriate classifiers based on the performance measure F-Measure is found to be Random Forest and Random Tree.

Figure 5.21: ROC Area (Cross Validation: 10-Folds – Heavy Traffic)



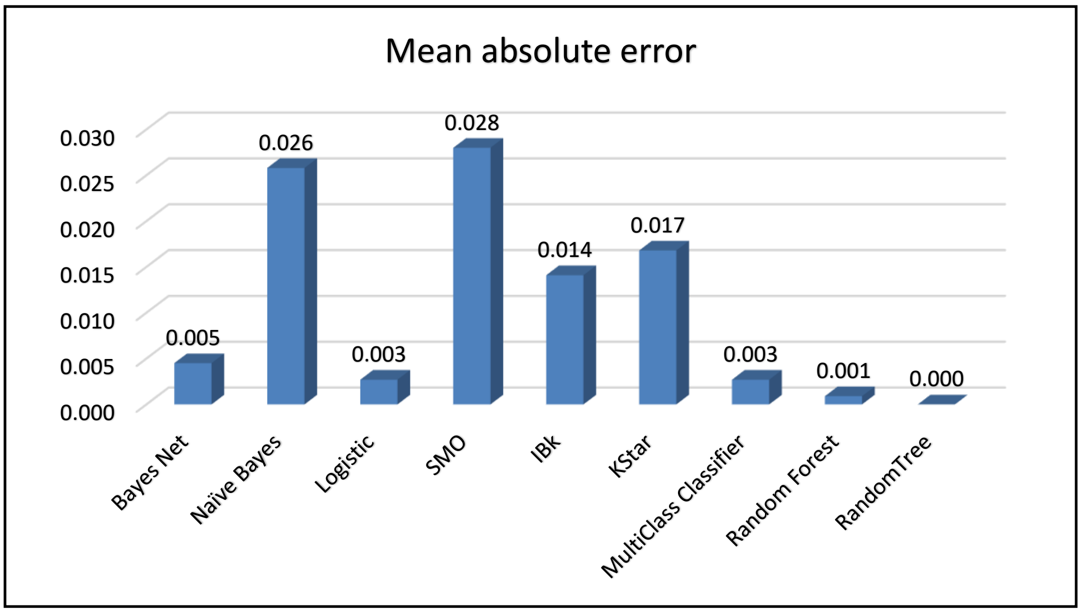
Based on the performance measure ROC class label: Heavy Traffic it was found that the highest ROC Area value was of the classifiers Bayes Net, Logistic, MultiClass Classifier, Random Forest and Random Tree with value 1.00, followed by 0.988 of Naive Bayes respectively whereas the lowest ROC Area value was found to be of the classifier SMO with values 0.791 respectively. Overall, it can be interpreted that the most appropriate classifiers based on the performance measure ROC Area are found to be five Algorithms.

B. Error Measure Results

Table 5.4: Classifiers and Error Measures (Cross Validation: 10-Folds)

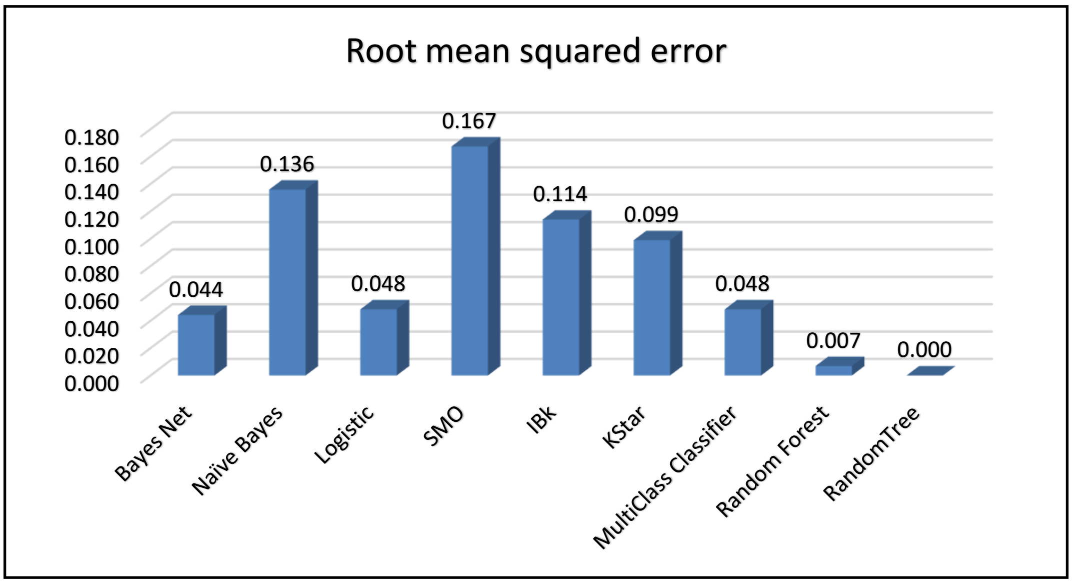
Classifier	Mean absolute error	Root mean squared error	Relative absolute error	Root relative squared error
Bayes Net	0.005	0.044	3.549%	17.692%
Naïve Bayes	0.026	0.136	20.501%	54.312%
Logistic	0.003	0.048	2.171%	19.311%
SMO	0.028	0.167	22.247%	66.924%
IBk	0.014	0.114	11.180%	45.553%
KStar	0.017	0.099	13.340%	39.526%
MultiClass Classifier	0.003	0.048	2.171%	19.311%
Random Forest	0.001	0.007	0.739%	2.762%
RandomTree	0.000	0.000	0.000%	0.000%

Figure 5.22: Mean Absolute Error (Cross-Validation: 10 Folds)



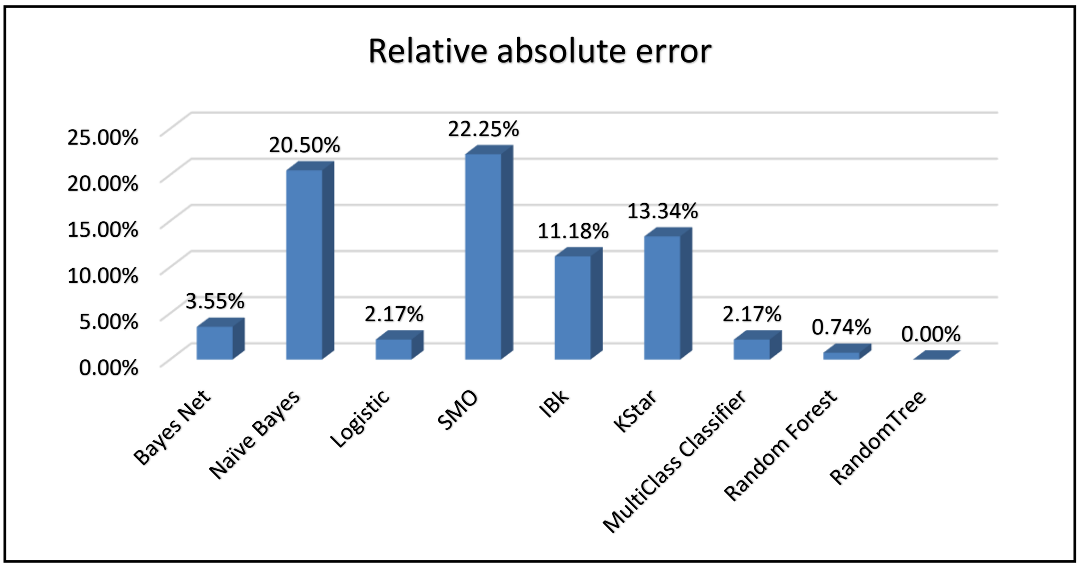
The mean absolute error is found to be lowest in case of Random Tree with the value 0.000 Whereas the mean absolute error value of SMO is found to be highest with value 0 .028. So, it can be interpreted that based on the measure Mean absolute Error the most appropriate algorithm is found to be Random Tree at configuration setting – 10-fold cross validation.

Figure 5.23: Root Mean Squared Error (Cross-Validation: 10 Folds)



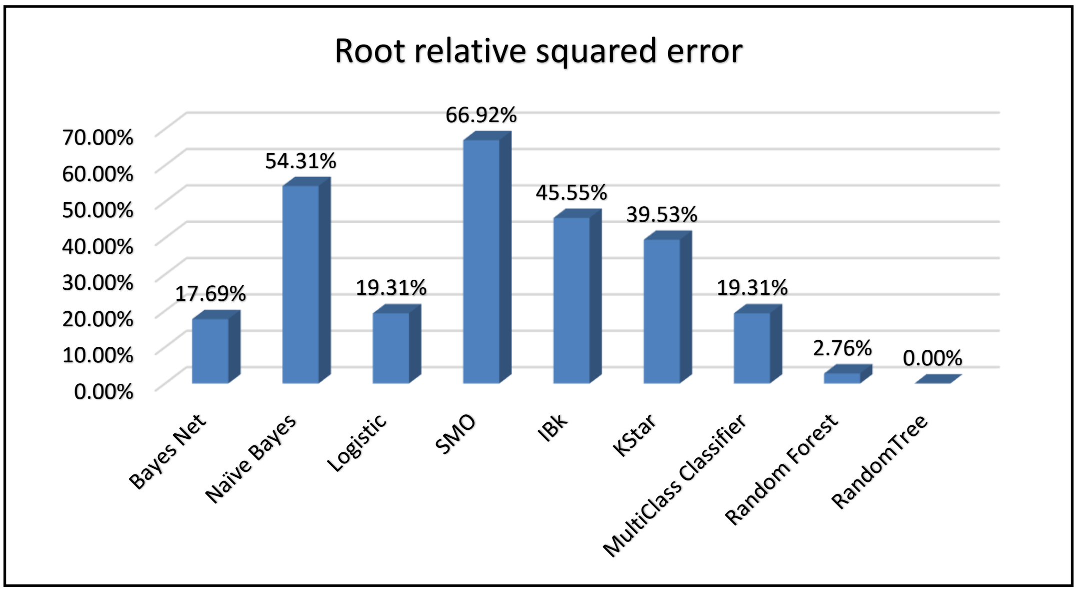
The root mean squared error value is found to be highest in case of SM0 with the value of 0 .167 whereas the lowest value is found to be of Random Tree with value 0.000. So, it can be interpreted that based on the measure RMSE the most appropriate algorithm is found to be Random Tree at configuration setting – 10-fold cross validation.

Figure 5.24: Relative Absolute Error (Cross Validation: 10-Folds)



Accordingly, the relative absolute error value is found to be lowest in case of Random Tree classifier with 0.00% whereas the highest relative absolute error percentage value is found to be in case of SMO with 22.25%. So, it can be suggested that based on the measure Relative Absolute Error the most appropriate algorithm is found to be Random Tree with lowest value when evaluated at configuration setting – 10-fold cross validation.

Figure 5.25 : Root Relative Squared Error (Cross Validation: 10-Folds)



The root relative squared error value is found to be lowest in case of Random Tree classifier with 0.00% whereas the root relative squared error percentage value is found to be highest in case of SMO with percentage value of 66.92%. So, it can be interpreted that based on the measure RRSE the most appropriate algorithm is found to be Random Tree with lowest percentage value when evaluated at configuration setting – 10-fold cross validation.

In summary, it is important to understand error measures in Machine learning for assessing the performance of model properly and making informed decisions. The choice of specific metrics depends on the nature of the problem, characteristics of the dataset, and goals of the analysis.

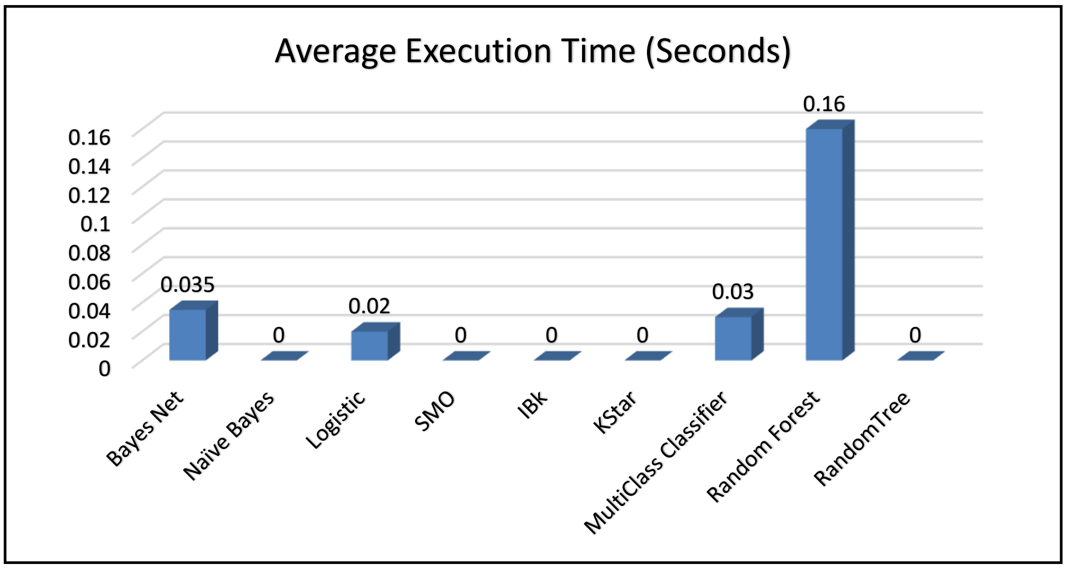
C. Execution Time Results

The execution time of a machine learning algorithm refers to the time it takes for the algorithm to process and analyse input data, train the model (if applicable), and produce predictions or results. Execution time is an important factor when evaluating the efficiency and scalability of machine learning algorithms, especially when dealing with large data sets and real-time applications. The Average Execution Time of Nine Classifier Algorithm is given below.

Table 5.5: Classifiers and Average Execution Time (Cross Validation: 10-Folds)

Classifier	Average Execution Time (Seconds)
Bayes Net	0.035
Naïve Bayes	0
Logistic	0.02
SMO	0
IBk	0
KStar	0
MultiClass Classifier	0.03
Random Forest	0.16
RandomTree	0

Figure 5.26 : Average Execution Time (Cross Validation: 10-Folds)



According to the performance measure average execution time it was found that the lowest average execution time were of the classifiers Naïve Bayes, SMO, IBK, KStar and Random Tree with values nearly 0 Seconds each whereas the highest average execution time was found to be of the classifiers Random Tree classifier with values 0.16 respectively. Overall, it can be interpreted the most appropriate classifiers based on the performance measure average execution time are found to be Naïve Bayes, SMO, IBK, KStar and Random Tree.

There are Several factors like Algorithm complexity, Data Size, Model Training, Hardware resources, Optimizations, Feature Engineering and Software implementation etc. that can affect the execution time of a machine learning algorithm.

5.3.4 Cross-Validation Configuration Setting (25-Folds) Results

Cross-validation is a valuable technique in machine learning for assessing the performance of predictive models. It involves splitting a dataset into multiple subsets or "folds" to train and test the model on different portions of the data. The number of folds, such as the "25-folds" configuration setting you mentioned, determines how many times this process is repeated. The Weka tool was being used for the analysis of various classification machine learning algorithms. K-folds Cross-Validation Approach is used for evaluating the Performance of Machine learning Algorithms, where K value is changed to study difference cases. In 25 folds Cross-Validation K value is 25 where data set is divided into 25 parts, out of which 24 parts are used for training the Machine learning Algorithm and only One part is used for testing the Algorithm. Twenty five iterations are executed for 25-folds, In first iteration part one will be used for testing and remaining Twenty four parts will be used for training. In second iteration part two will be used for testing and remaining twenty four parts will be used for training. This recursion will continue up to the last iteration to complete the Cross-Validation. Following credentials are used for Data Analysis.

Dataset: Udaipur_Traffic	**Source:** TOMTOM Server
Date: October 2023	**Duration:** One Month

Number of Instances: 1000

Number of Attributes (After Feature Extraction and Selection): 7

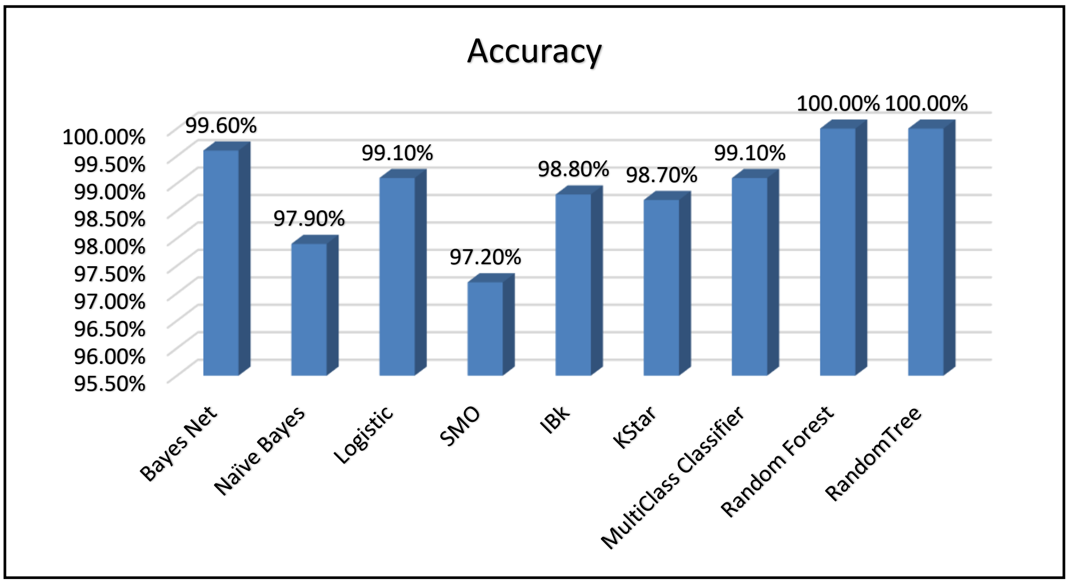
A. Performance Measures

i. Accuracy Measures

Table 5.6: Classifiers and Accuracy Measures (Cross-Validation: 25-Folds)

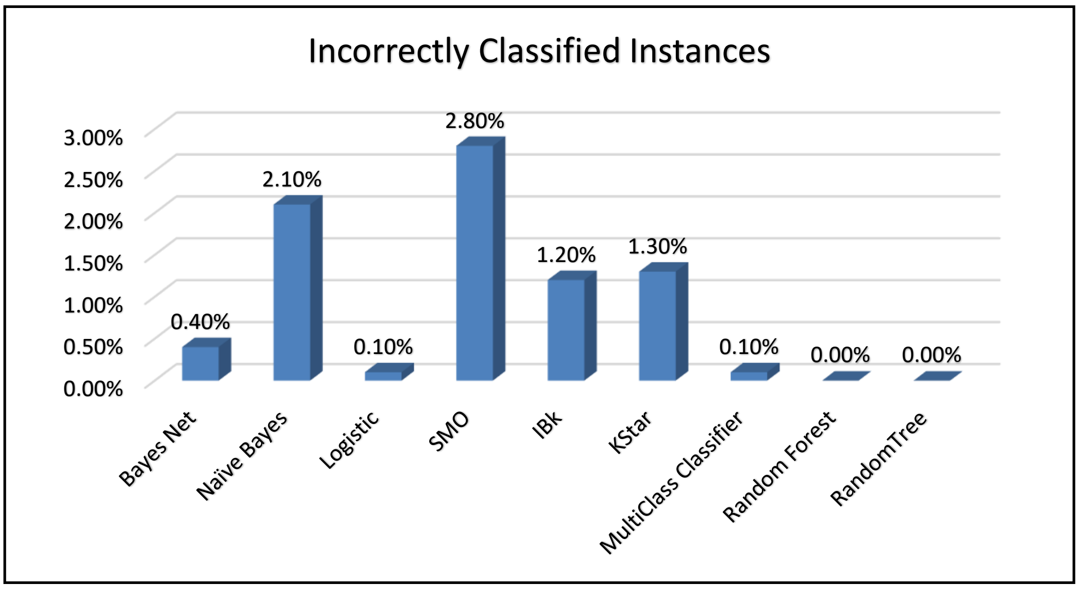
Classifier	Accuracy	Incorrectly Classified Instances	Kappa statistic
Bayes Net	99.60%	0.40%	0.967
Naïve Bayes	97.90%	2.10%	0.818
Logistic	99.10%	0.10%	0.992
SMO	97.20%	2.80%	0.722
IBk	98.80%	1.20%	0.905
KStar	98.70%	1.30%	0.897
MultiClass Classifier	99.10%	0.10%	0.992
Random Forest	100.00%	0.00%	1.000
RandomTree	100.00%	0.00%	1.000

Figure 5.27: Performance Measure Accuracy (Cross-Validation: 25 Folds)



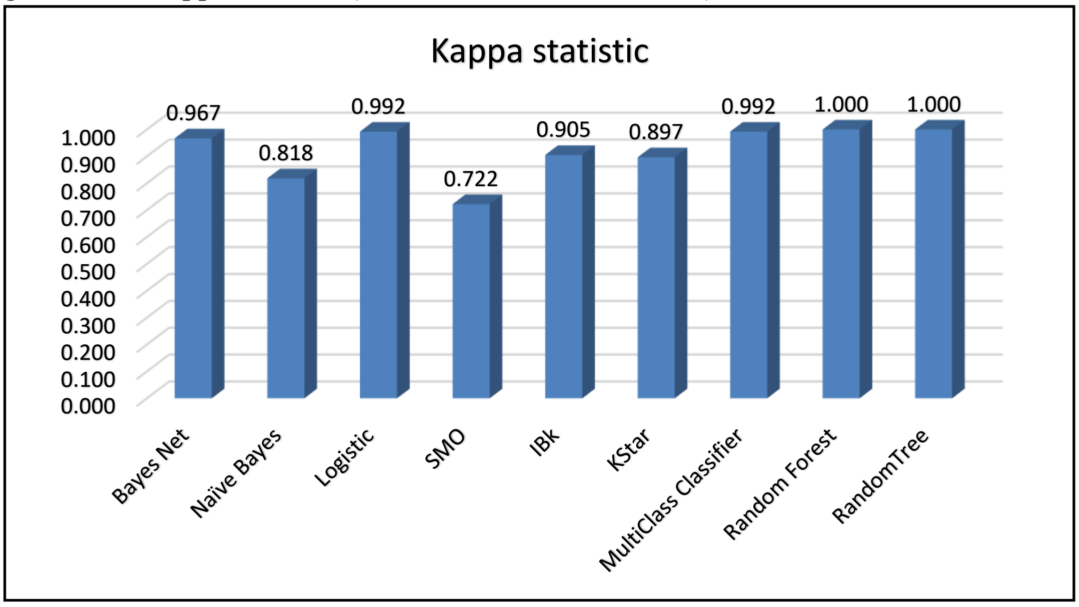
Based on the performance measure accuracy it can be interpreted that Random Forest and Random Tree classifiers were the most appropriate one as they were having the highest accuracy value of 100% whereas classifier Naïve Bayes and SMO were having the lowest value of accuracy 97.90% and 97.20% each.

Figure 5.28: Incorrectly Classified Instances (Cross-Validation: 25 Folds)



According to the performance measure incorrectly classified instances it can be interpreted that Random Forest and Random Tree classifiers were the most appropriate one as these Algorithms were having the lowest number of incorrectly classifies instances, whereas classifiers SMO was having the highest number of incorrectly classified instances.

Figure 5.29: Kappa Statistic (Cross-Validation: 25 Folds)



The provided data consists of a set of Kappa statistic values, which are used to assess the agreement or consistency between classifiers in different situations. These Kappa values range from 0.722 to 1.000, indicating varying levels of agreement. The highest

Kappa value, 1.000, suggests a very good level of agreement between the classifiers in that particular scenario, while the lowest value, 0.722, falls into the good to moderate agreement range. Overall, the data suggests that there is a generally positive level of agreement in the assessed situations, with some instances demonstrating higher agreement than others.

In correctly classified instances and kappa statistics are performance metrics which are important to explore to get more comprehensive insights to develop accurate machine learning model.

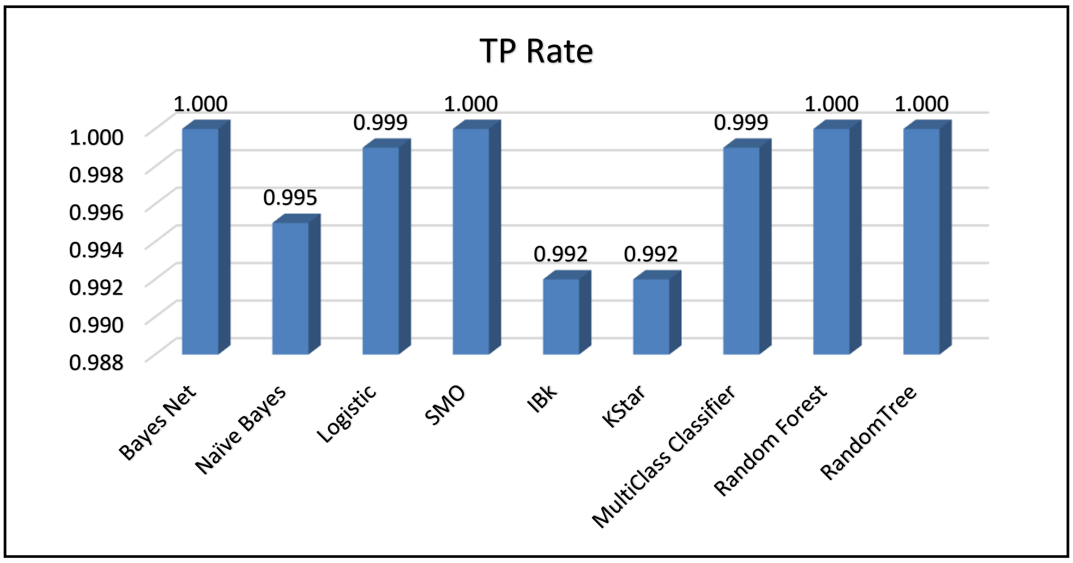
ii. Confusion Matrix Parameters – Low Traffic

Table 5.7 shows confusion matrix parameters TP Rate, FP Rate, Precision, Recall, F-Measure and ROC Area for Low Traffic case. Machine learning often requires a limited amount of data when dealing with low-traffic scenarios. In such cases, the challenge is to create a robust model despite data limitations. Data Augmentation techniques are used to artificially increase the size of data set which can be helpful especially in low traffic scenarios.

Table 5.7: Classifiers and Performance Measures Class Label: Low Traffic
Cross Validation: 25-Folds

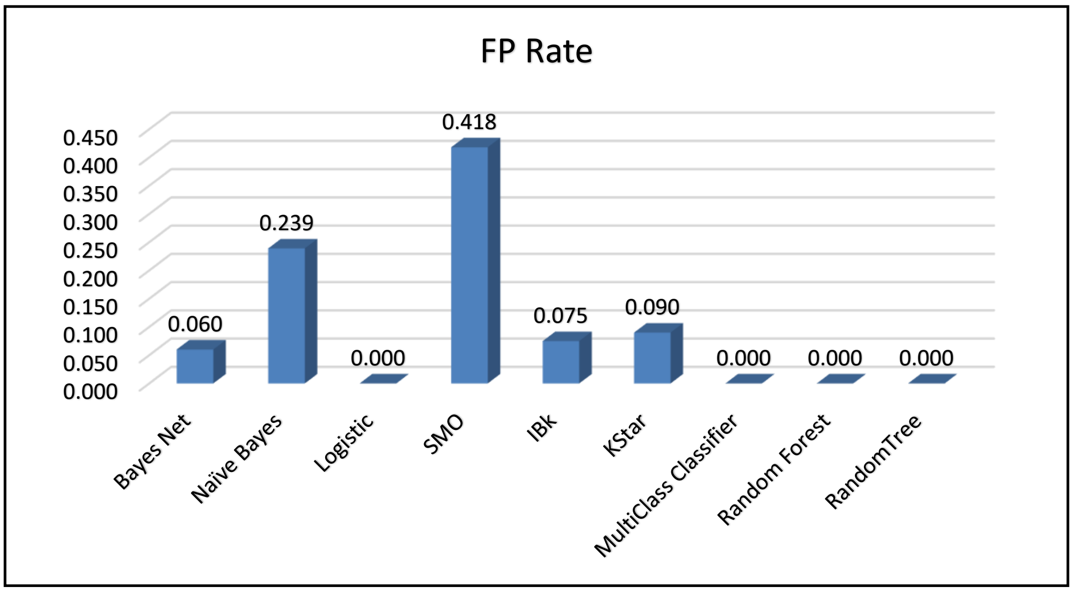
Classifier	TP Rate	FP Rate	Precision	Recall	F-Measure	ROC Area
Bayes Net	1.000	0.060	0.996	1.000	0.998	1.000
Naïve Bayes	0.995	0.239	0.983	0.995	0.989	0.990
Logistic	0.999	0.000	1.000	0.999	0.999	1.000
SMO	1.000	0.418	0.971	1.000	0.985	0.791
IBk	0.992	0.075	0.995	0.992	0.994	0.959
KStar	0.992	0.090	0.994	0.992	0.993	0.997
MultiClass Classifier	0.999	0.000	1.000	0.999	0.999	1.000
Random Forest	1.000	0.000	1.000	1.000	1.000	1.000
RandomTree	1.000	0.000	1.000	1.000	1.000	1.000

Figure 5.30 : TP Rate (Cross Validation: 25-Folds – Low Traffic)



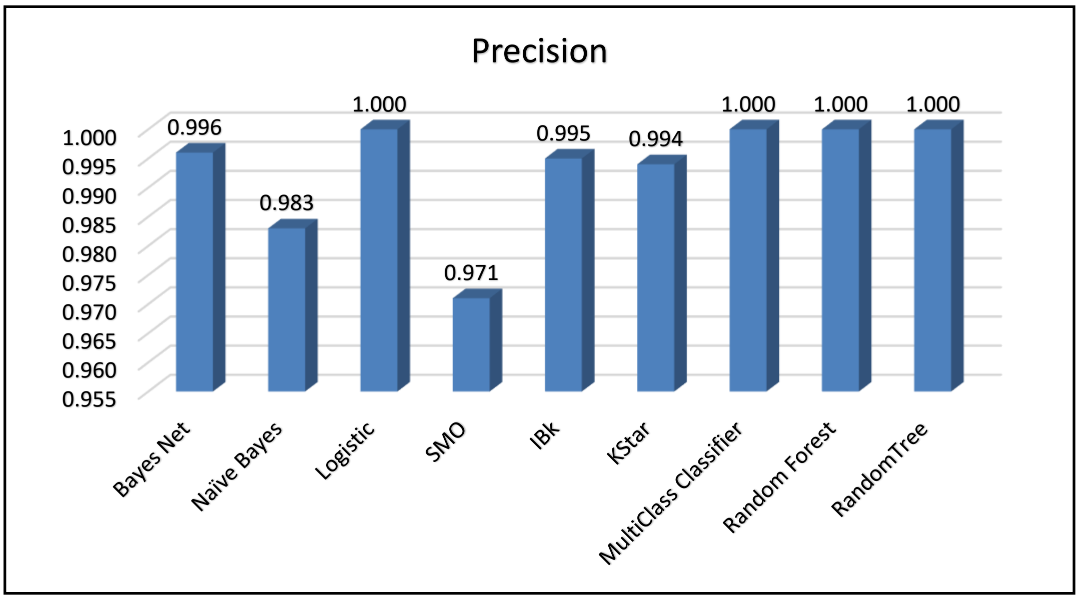
According to the performance measure TP rate for class label: Low Traffic it was found that the highest true positive rate was of the classifiers Bayes Net, SMO, Random Forest and Random Tree with value 1.000, followed by 0.999 of Logistic and MultiClass Classifier, The lowest TP rate was found to be of the classifiers IBK and KStar with value 0.992 respectively.

Figure 5.31 : FP Rate (Cross Validation: 25-Folds – Low Traffic)



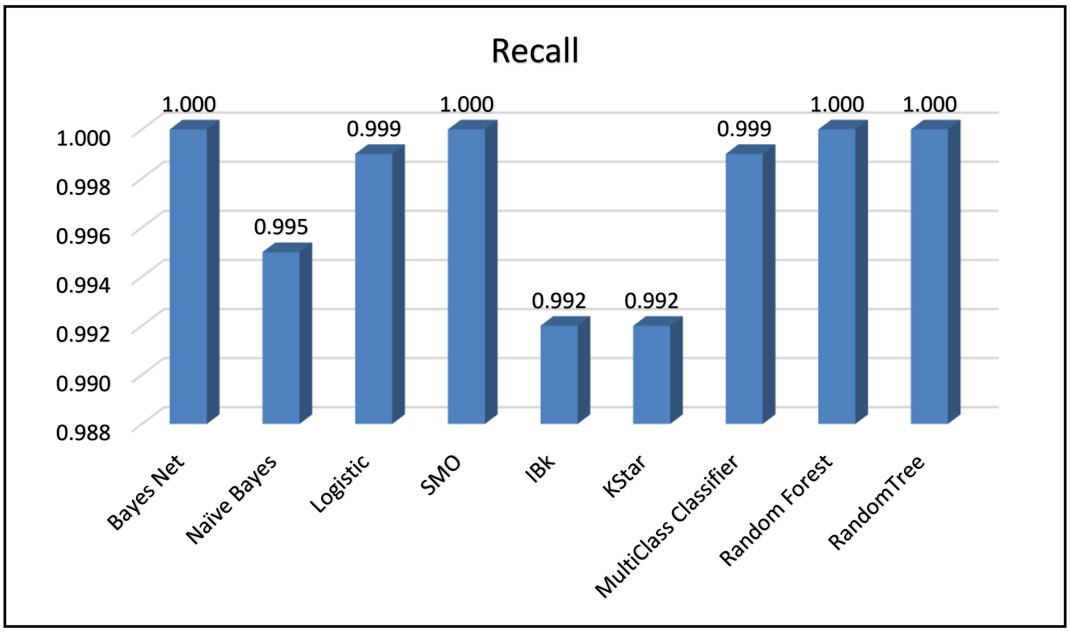
Based on the performance measure FP rate for class label: Low Traffic it was found that the lowest false positive rate was of the classifiers Logistic, MultiClass Classifier, Random Forest and Random Tree with value 0.000, followed by 0.060 of Bayes Net whereas the highest FP rate was found to be of the classifier SMO with value 0.418.

Figure 5.32: Precision (Cross Validation: 25-Folds – Low Traffic)



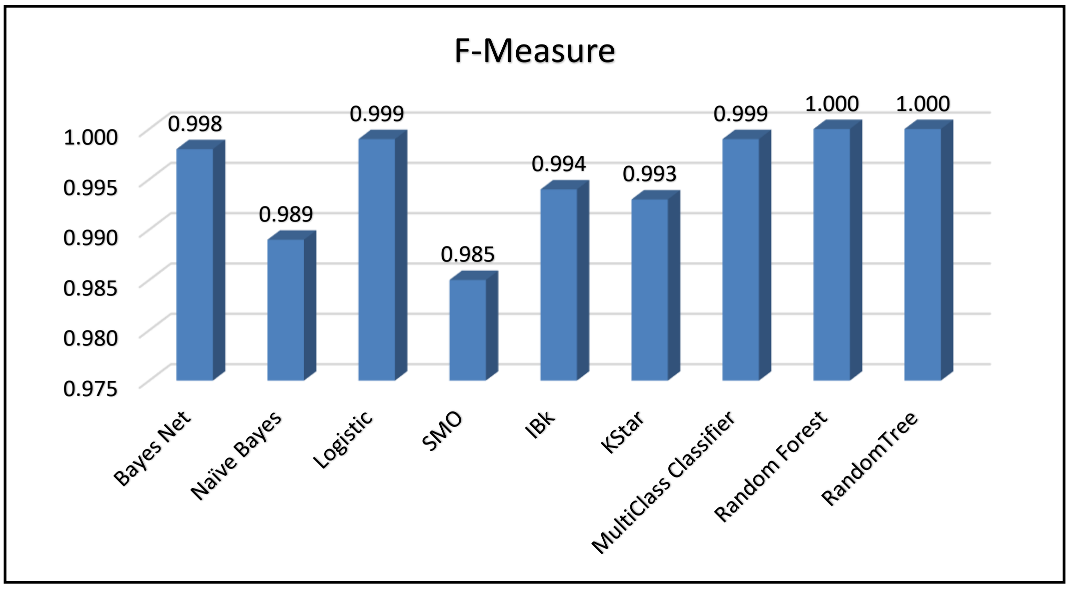
According to the performance measure precision for class label: Low Traffic it was found that the highest precision value was of the classifiers Logistic, MultiClass Classifier, Random Forest and Random Tree with value 1.000, followed by 0.996, 0.995 and 0.994 of Bayes Net, IBK and KStar respectively whereas the lowest precision value was found to be of the classifier SMO with values 0.971 respectively. Overall, it can be interpreted that the most appropriate classifiers based on the performance measure precision are found to be four Algorithms.

Figure 5.33 : Recall (Cross Validation: 25-Folds – Low Traffic)



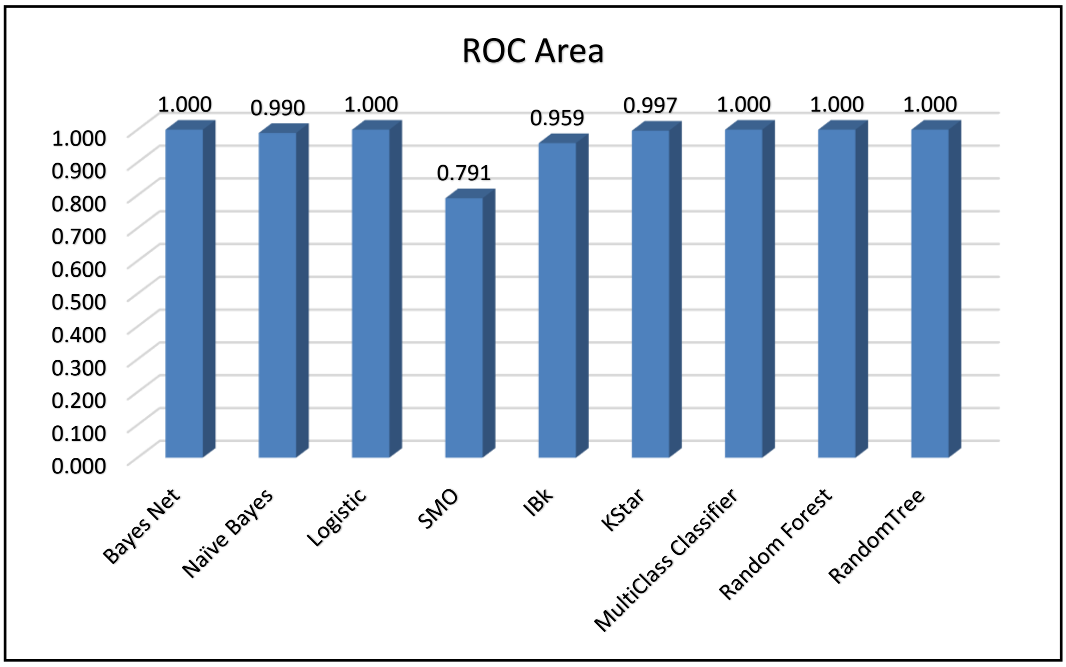
Based on the performance measure recall for class label: Low Traffic it was found that the highest recall value was of the classifiers Bayes Net, SMO, Random Forest and Random Tree with value 1.000, followed by 0.999 of Logistic and MultiClass Classifier respectively whereas the lowest recall value was found to be of the classifiers IBK and KStar with values 0.992 respectively. Overall, it can be interpreted that the most appropriate classifier based on the performance measure recall are found to be Bayes Net, SMO, Random Forest and Random Tree.

Figure 5.34 : F-Measure (Cross Validation: 25-Folds – Low Traffic)



According to the performance measure F-Measure for class label: Low Traffic it was found that the highest F-Measure value was of the classifiers Random Forest and Random Tree with value 1.000, followed by 0.999 and 0.998 of Logistic, MultiClass Classifier and Bayes Net respectively whereas the lowest F-Measure value was found to be of the classifiers SMO with value 0.985. Overall, it can be interpreted that the most appropriate classifier based on the performance measure F-Measure are found to be Random Forest and Random Tree.

Figure 5.35 : ROC Area (Cross Validation: 25-Folds – Low Traffic)



Based on the performance measure ROC for class label: Low Traffic it was found that the highest ROC Area value was of the classifiers Bayes Net, Logistic, MultiClass Classifier, Random Forest and Random Tree with value 1.000, followed by 0.997, 0.990 and 0.959 of KStar, Naïve Bayes and IBK respectively whereas the lowest ROC Area value was found to be of the classifiers SMO with value 0.791 respectively.

In summary, performance measurements play a critical role in evaluating the effectiveness of machine learning algorithms, providing insight into their ability to make accurate predictions and transform appropriately to new, unseen data. Choosing the most appropriate metric depends on the nature of your particular problem, the characteristics of your data, and your analysis goals.

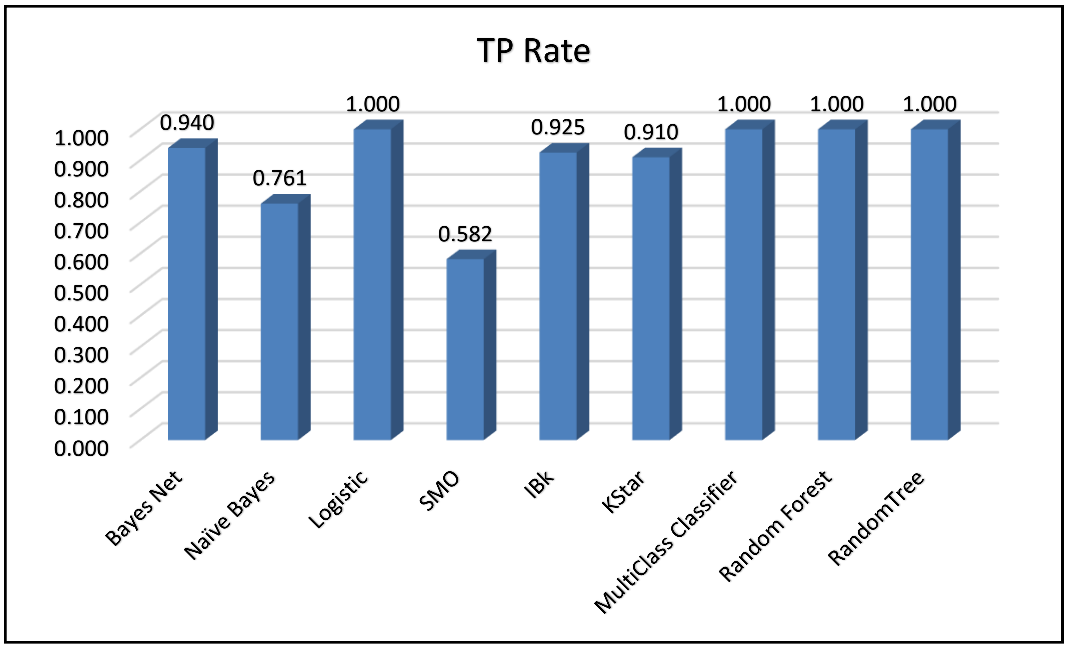
iii. Confusion Matrix Parameters – Heavy Traffic

Heavy Traffic generates huge amounts of data from the various IOT sensors. These data sets can be used by Machine Learning Algorithms to develop prediction models. Table 5.8 shows the Confusion Matrix parameters TP Rate, FP Rate, Precision, Recall, F-Measure and ROC Area obtained for Heavy Traffic conditions.

Table 5.8: Classifiers Performance Measure Class Label: Heavy Traffic: Cross Validation: 25-Folds

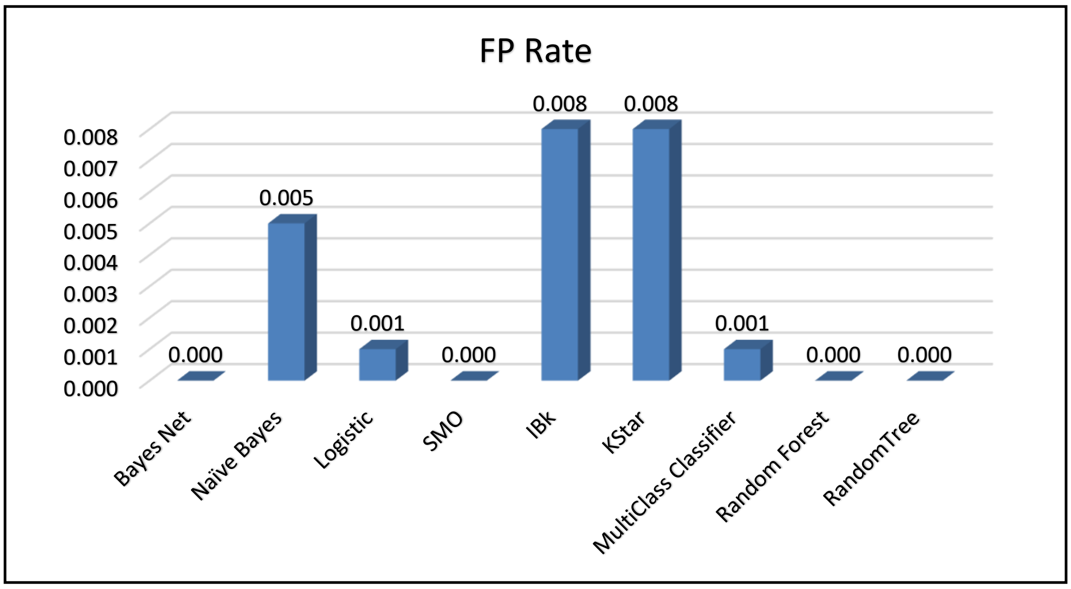
Classifier	TP Rate	FP Rate	Precision	Recall	F-Measure	ROC Area
Bayes Net	0.940	0.000	1.000	0.940	0.969	1.000
Naïve Bayes	0.761	0.005	0.911	0.761	0.829	0.990
Logistic	1.000	0.001	0.985	1.000	0.993	1.000
SMO	0.582	0.000	1.000	0.582	0.736	0.791
IBk	0.925	0.008	0.899	0.925	0.912	0.959
KStar	0.910	0.008	0.897	0.910	0.904	0.997
MultiClass Classifier	1.000	0.001	0.985	1.000	0.993	1.000
Random Forest	1.000	0.000	1.000	1.000	1.000	1.000
RandomTree	1.000	0.000	1.000	1.000	1.000	1.000

Figure 5.36 : TP Rate (Cross Validation: 25-Folds – Heavy Traffic)



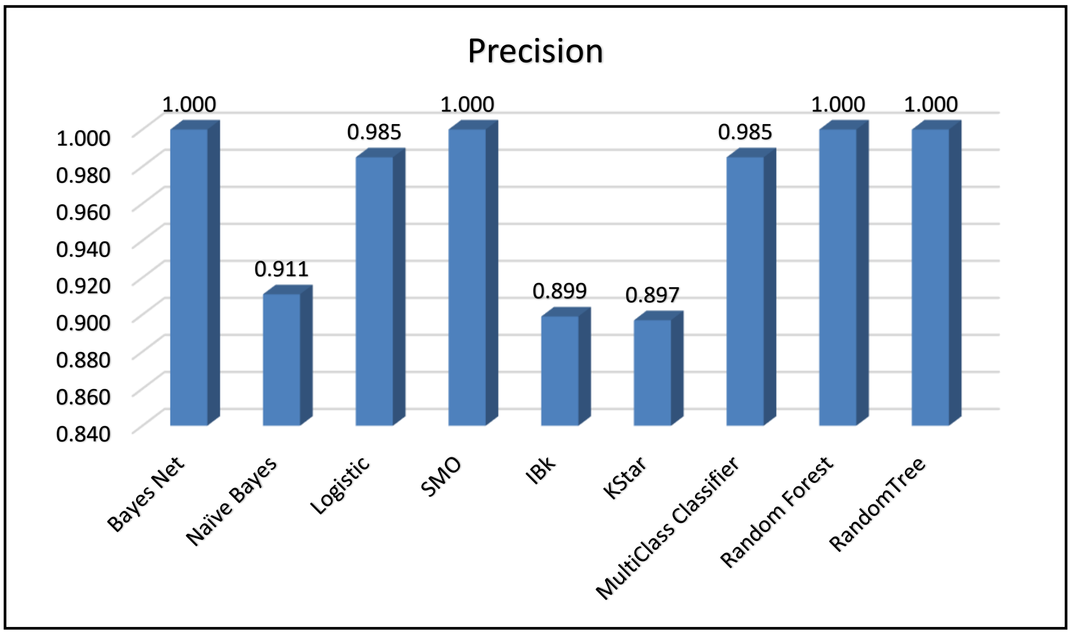
According to the performance measure TP rate for class label: Heavy Traffic it was found that the highest true positive rate was of the classifiers Logistic, MultiClass Classifier, Random Forest and Random Tree with value 1.0, followed by 0.940, 0.925 and 0.910 of Bayes Net, IBK and KStar respectively whereas the lowest TP rate was found to be of the SMO with value 0.582 respectively.

Figure 5.37 : FP Rate (Cross Validation: 25-Folds – Heavy Traffic)



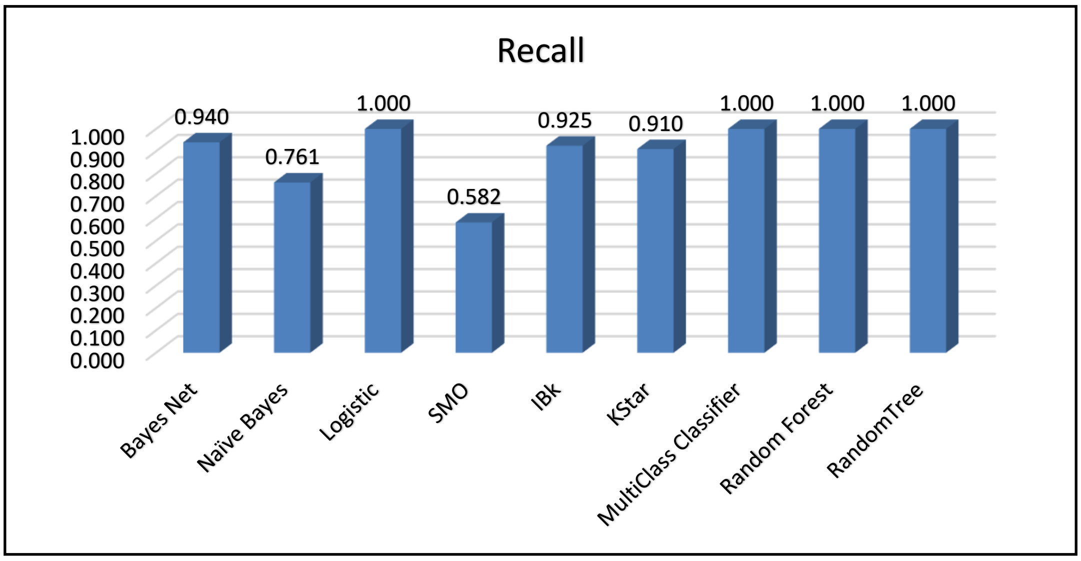
Based on the performance measure FP rate for class label: Heavy Traffic it was found that the lowest false positive rate were of the classifiers Bayes Net, SMO, Random Forest and Random Tree with value 0.00, followed by 0.001 of Logistic and MultiClass Classifier whereas the highest FP rate was found to be of the classifiers IBK and KStar with value 0.008 respectively. Overall, it can be interpreted the most appropriate classifier based on the performance measure FP rate is found to be Bayes Net, SMO, Random Forest and Random Tree with lowest FP rate value.

Figure 5.38: Precision (Cross Validation: 25-Folds – Heavy Traffic)



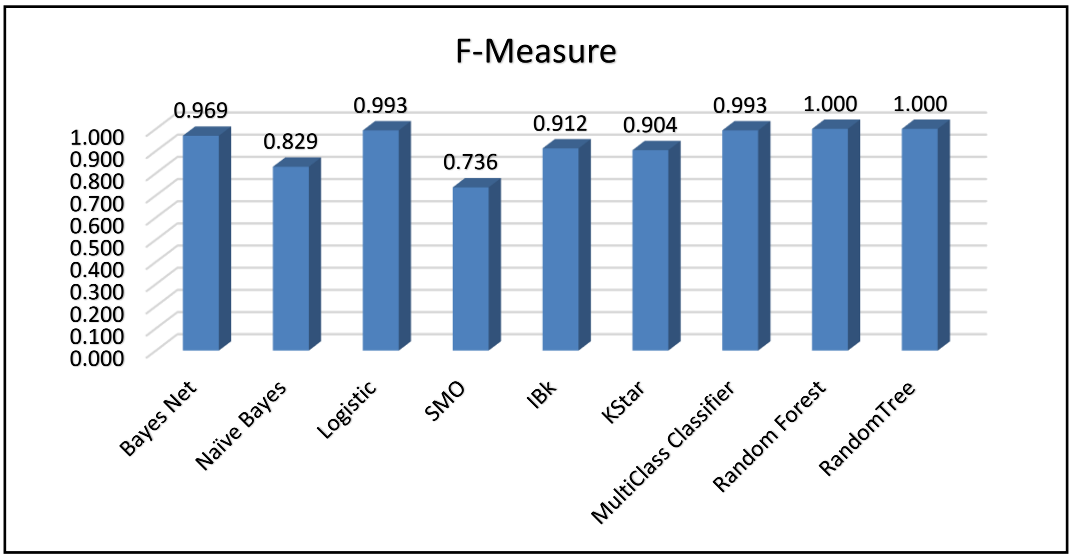
According to the performance measure precision class label: Heavy Traffic it was found that the highest precision value was of the classifiers Bayes Net, SMO, Random Forest and Random Tree with value 1.0, followed by 0.985 of Logistic and MultiClass Classifier respectively whereas the lowest precision values were found to be of the classifiers IBK and KStar with values 0.899 and 0.897 respectively.

Figure 5.39: Recall (Cross Validation: 25-Folds – Heavy Traffic)



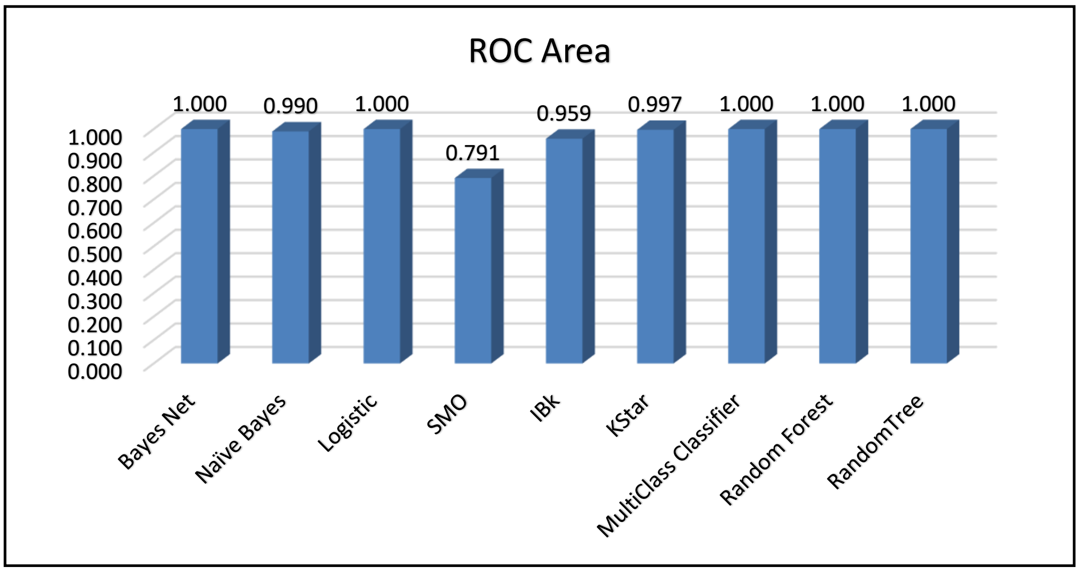
Based on the performance measure recall class label: Heavy Traffic it was found that the highest recall value was of the classifiers Logistic, MultiClass Classifier, Random Forest and Random Tree with value 1.0, followed by 0.940 ,0.925 and 0.910 of Bayes Net, IBK and KStar respectively whereas the lowest recall value was found to be of the classifier SMO with value 0.582 respectively.

Figure 5.40: F-Measure (Cross Validation: 25-Folds – Heavy Traffic)



According to the performance measure F-Measure class label: Heavy Traffic it was found that the highest F-Measure value was of the classifier Random Forest and Random Tree with value 1.0, followed by 0.993 and 0.969 of Logistic, MultiClass Classifier and Bayes Net respectively whereas the lowest F-Measure value was found to be of the classifier SMO with value 0.736. Overall, it can be interpreted that the most appropriate classifier based on the performance measure F-Measure is found to be Random Forest and Random Tree.

Figure 5.41: ROC Area (Cross Validation: 25-Folds – Heavy Traffic)



Based on the performance measure ROC class label: Heavy Traffic it was found that the highest ROC Area value was of the classifiers Bayes Net, Logistic, MultiClass Classifier, Random Forest and Random Tree with value 1.0, followed by 0.997, 0.990 and 0.959 of KStar, Naïve Bayes and IBK respectively whereas the lowest ROC Area value was found to be of the classifier SMO with value 0.791 respectively. Overall, it can be interpreted the most appropriate classifiers based on the performance measure ROC Area are found to be Five Algorithms.

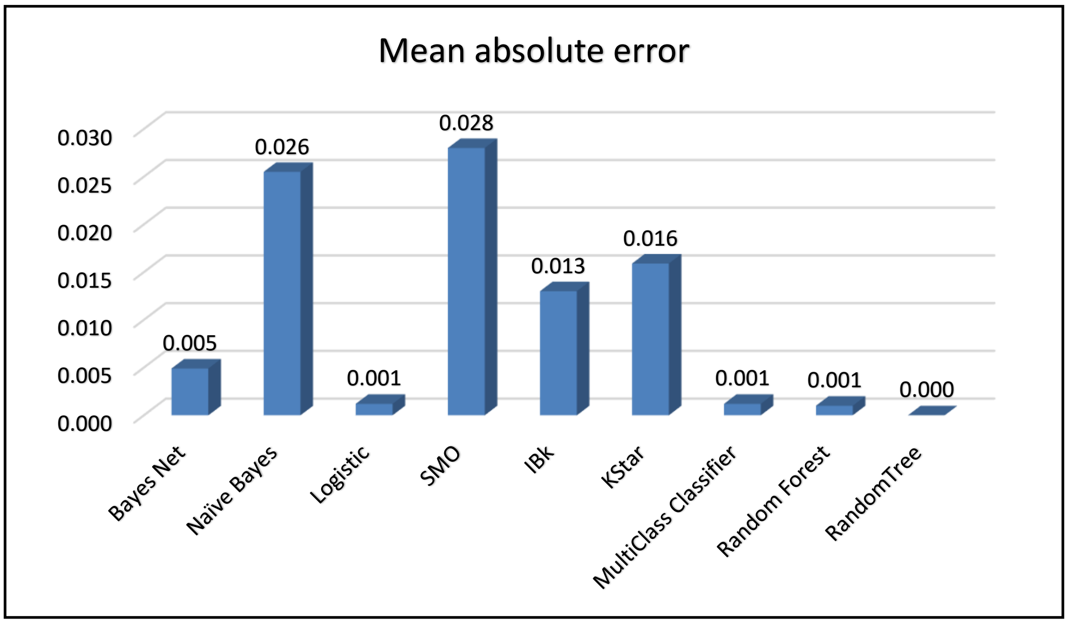
In conclusion the 25 fold Cross validation not only increases the reliability of model but also gives insights of model and explains how model behave under different conditions. More number of folds trains model more accurately to face real life applications and also diminishes the chances of overfitting and underfitting. More number of folds also increases model effectiveness and gives superior model performance, paving the way for more trustworthy and impactful model for real life applications.

B. Error Measure Results

Table 5.9: Classifiers and Error Measures (Cross Validation: 25-Folds)

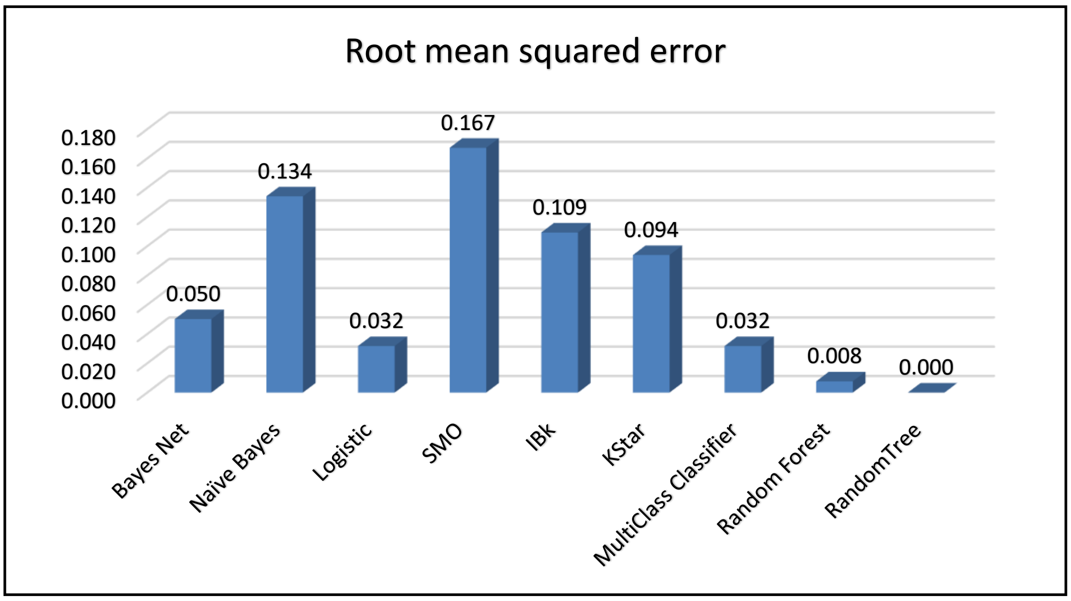
Classifier	Mean absolute error	Root mean squared error	Relative absolute error	Root relative squared error
Bayes Net	0.005	0.050	3.89%	20.08%
Naïve Bayes	0.026	0.134	20.28%	53.62%
Logistic	0.001	0.032	91.68%	12.75%
SMO	0.028	0.167	22.26%	66.92%
IBk	0.013	0.109	10.34%	43.77%
KStar	0.016	0.094	12.63%	37.59%
MultiClass Classifier	0.001	0.032	0.92%	12.73%
Random Forest	0.001	0.008	0.77%	3.08%
RandomTree	0.000	0.000	0.00%	0.00%

Figure 5.42: Mean Absolute Error (Cross-Validation: 25 Folds)



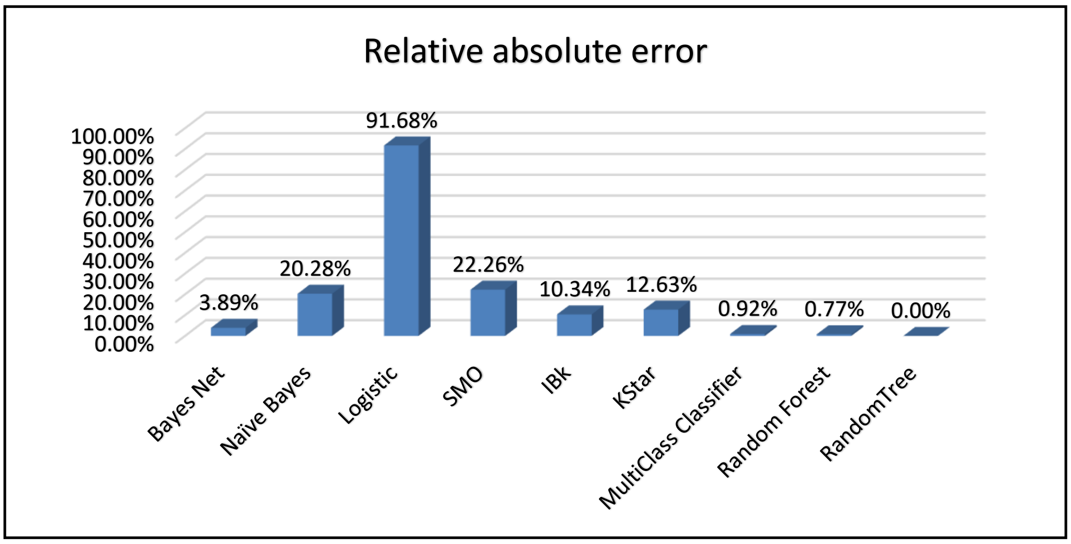
The mean absolute error is found to be lowest in case of Random Tree with the value 0.000 Whereas the mean absolute error value of SMO is found to be highest with value 0 .028. So, it can be interpreted that based on the measure Mean absolute Error the most appropriate algorithm is found to be Random Tree at configuration setting – 25-fold cross validation.

Figure 5.43: Root Mean Squared Error (Cross-Validation: 25 Folds)



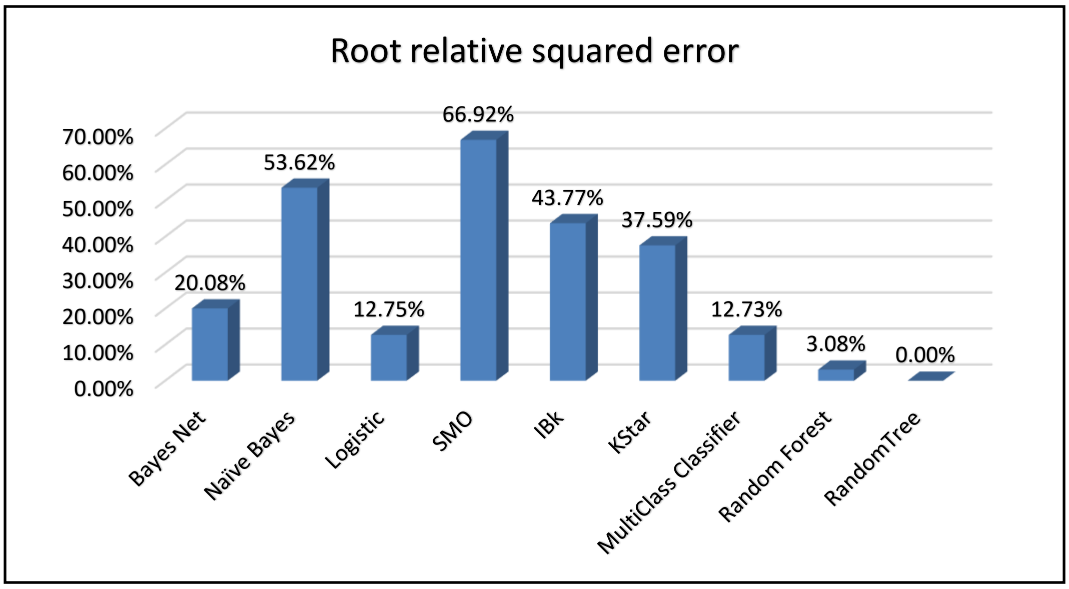
The root mean squared error value is found to be highest in case of SM0 with the value of 0 .167 whereas the lowest value is found to be of Random Tree with value 0.000. So, it can be interpreted that based on the measure RMSE the most appropriate algorithm is found to be Random Tree at configuration setting – 25-fold cross validation.

Figure 5.44: Relative Absolute Error (Cross Validation: 25-Folds)



Accordingly, the relative absolute error value is found to be lowest in case of Random Tree classifier with 0.00% whereas the highest relative absolute error percentage value is found to be in case of Logistic with 91.68%. So, it can be suggested that based on the measure Relative Absolute Error the most appropriate algorithm is found to be Random Tree with lowest value when evaluated at configuration setting – 25-fold cross validation.

Figure 5.45: Root Relative Squared Error (Cross Validation: 25-Folds)



The root relative squared error value is found to be lowest in case of Random Tree classifier with 0.00% whereas the root relative squared error percentage value is found to be highest in case of SMO with percentage value of 66.92%. So, it can be interpreted that based on the measure RRSE the most appropriate algorithm is found to be Random Tree with lowest percentage value when evaluated at configuration setting – 25-fold cross validation.

In summary, it is important to understand error measures in Machine learning for assessing the performance of model properly and making informed decisions. The choice of specific metrics depends on the nature of the problem, characteristics of the dataset, and goals of the analysis.

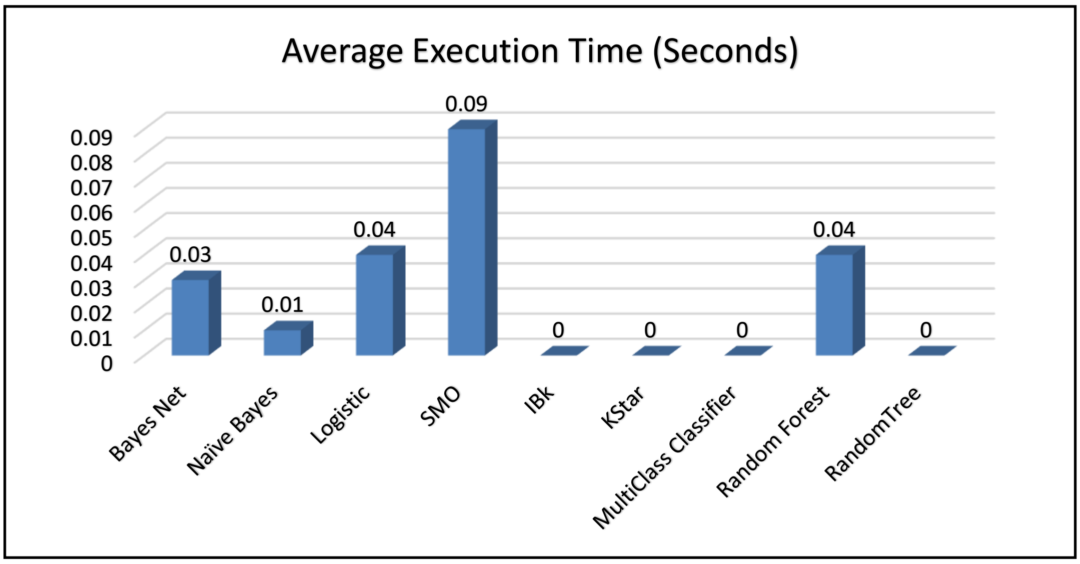
C. Execution Time Results

The execution time of a machine learning algorithm refers to the time it takes for the algorithm to process and analyse input data, train the model (if applicable), and produce predictions or results. Execution time is an important factor when evaluating the efficiency and scalability of machine learning algorithms, especially when dealing with large data sets and real-time applications. The Average Execution Time of Nine Classifier Algorithm for Cross Validation 25 fold is given below.

Table 5.10: Classifiers and Average Execution Time (Cross Validation: 25-Folds)

Classifier	Average Execution Time (Seconds)
Bayes Net	0.03
Naïve Bayes	0.01
Logistic	0.04
SMO	0.09
IBk	0
KStar	0
MultiClass Classifier	0
Random Forest	0.04
RandomTree	0

Figure 5.46 : Average Execution Time (Cross Validation: 25-Folds)

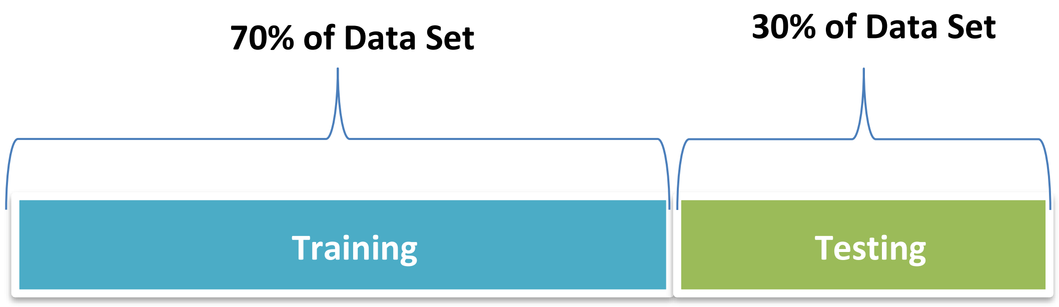


According to the performance measure average execution time it was found that the lowest average execution time were of the classifiers IBK, KStar, MultiClass Classifier and Random Tree with values 0.0 each whereas the highest average execution time was found to be of the classifier SMO with values 0.09 respectively. Overall, it can be interpreted that the most appropriate classifiers based on the performance measure average execution time are found to be IBK, KStar, MultiClass Classifier and Random Tree.

5.3.5 Cross-Validation: Configuration Setting (30% Split) Results

In the context of cross-validation, the "30% Split" configuration setting typically refers to a technique called "holdout validation" or "simple validation." It involves splitting the dataset into two portions: one for training the machine learning model and another for testing its performance. Here 70% of Data set is used for training the machine learning model and 30% is used for testing the Model. Testing is basically used to evaluate the model based on various metrics. Confusion Matrix can be used to evaluate the final performance of the Selected Machine learning models.

Figure 5.47 : Data Set Split



As shown in the Figure 5.47 the entire data set is randomly partitioned into Training set and Testing set. Since The data set is split into only two set, therefore it is constructed very fast on training data and executed for testing very fast. Following credentials are used for Data Analysis.

Dataset: Udaipur_Traffic	**Source:** TOMTOM Server
Date: October 2023	**Duration:** One Month

Number of Instances: 1000

Number of Attributes (After Feature Extraction and Selection): 7

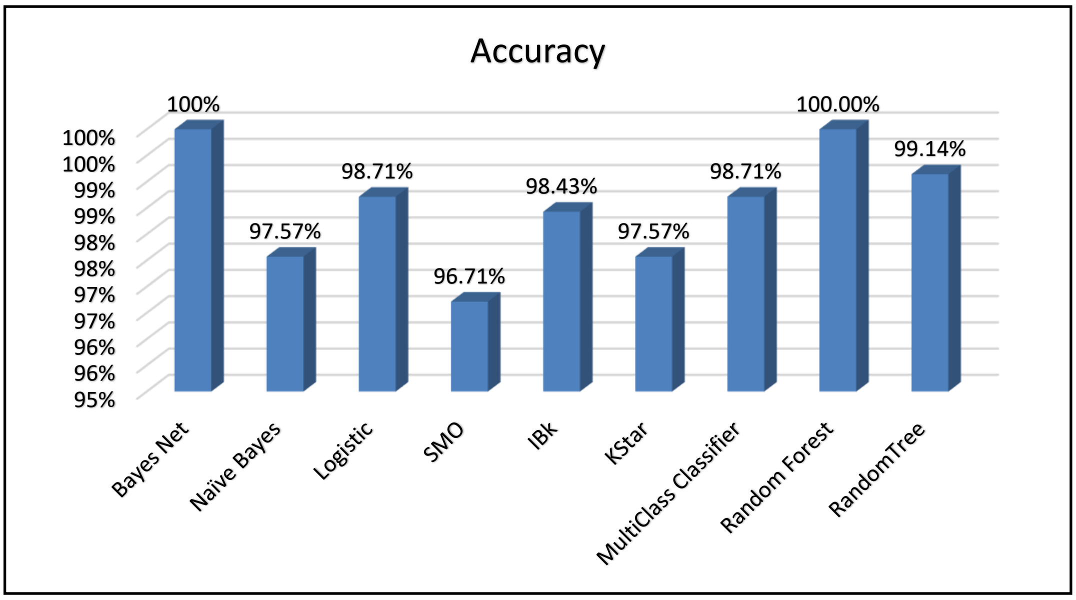
A. Performance Measures

i. Accuracy Measures

Table 5.11: Classifiers and Accuracy Measures (Cross-Validation:30% Split)

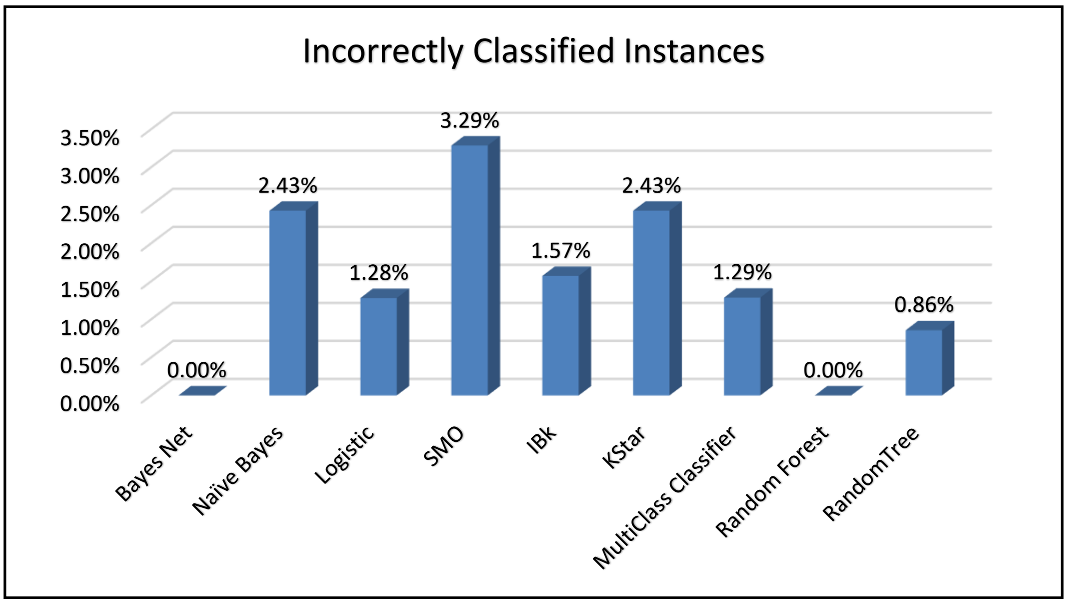
Classifier	Accuracy	Incorrectly Classified Instances	Kappa Statistic
Bayes Net	100%	0.00%	0.010
Naïve Bayes	97.57%	2.43%	0.008
Logistic	98.71%	1.28%	0.009
SMO	96.71%	3.29%	0.707
IBk	98.43%	1.57%	0.881
KStar	97.57%	2.43%	0.797
MultiClass Classifier	98.71%	1.29%	0.900
Random Forest	100.00%	0.00%	1.000
RandomTree	99.14%	0.86%	0.935

Figure 5.48 : Performance Measure Accuracy (Cross-Validation: 30% Split)



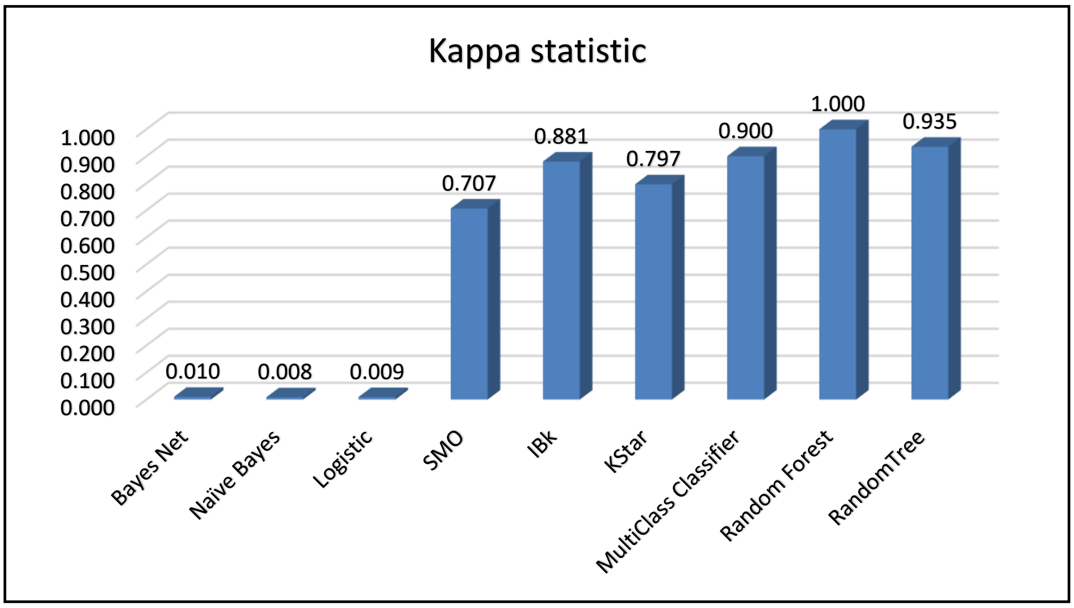
Based on the performance measure accuracy it can be interpreted that Bayes Net and Random Forest classifiers were the most appropriate one as they were having the highest accuracy value of 100% whereas classifier SMO was having the lowest value of accuracy 96.71% respectively.

Figure 5.49: Incorrectly Classified Instances (Cross-Validation: 30% Split)



According to the performance measure incorrectly classified instances it can be interpreted that Bayes Net and Random Forest classifier were the most appropriate one as they were having the lowest number of incorrectly classified instances accounting for 0% whereas classifier SMO classifier was having the highest number of incorrectly classifies instances accounting as 3.29% respectively.

Figure 5.50: Kappa Statistic (Cross-Validation: 30% Split)



The provided data consists of a set of Kappa statistic values, which are used to assess the agreement or consistency between classifiers in different situations. These Kappa values range from 0.008 to 1.000, indicating varying levels of agreement. The highest

Kappa value, 1.000, suggests a very good level of agreement between the classifiers in that particular scenario, while the lowest value, 0.008, falls into the poor agreement range.

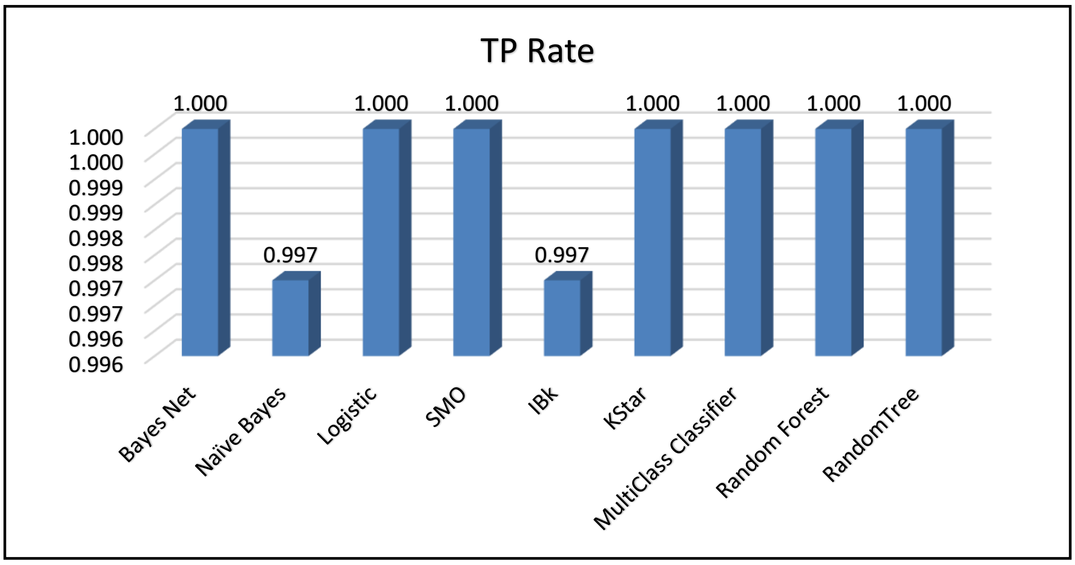
ii. Confusion Matrix Parameters – Low Traffic

Machine learning often requires a limited amount of data when dealing with low-traffic scenarios. In such cases, the challenge is to create a robust model despite data limitations. Data Augmentation techniques are used to artificially increase the size of data set which can be helpful especially in low traffic scenarios.

Table 5.12: Classifiers Performance Measures Class Label: Low Traffic: Cross Validation: 30% Split

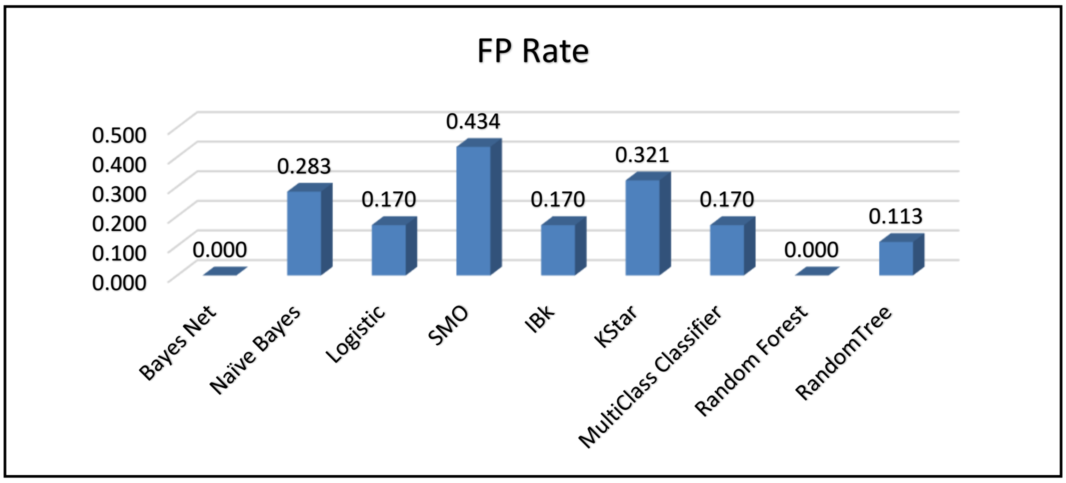
Classifier	TP Rate	FP Rate	Precision	Recall	F-Measure	ROC Area
Bayes Net	1.000	0.000	1.000	1.000	1.000	1.000
Naïve Bayes	0.997	0.283	0.977	0.997	0.987	0.990
Logistic	1.000	0.170	0.986	1.000	0.993	0.990
SMO	1.000	0.434	0.966	1.000	0.983	0.783
IBk	0.997	0.170	0.986	0.997	0.992	0.914
KStar	1.000	0.321	0.974	1.000	0.987	0.997
MultiClass Classifier	1.000	0.170	0.986	1.000	0.993	0.990
Random Forest	1.000	0.000	1.000	1.000	1.000	1.000
RandomTree	1.000	0.113	0.991	1.000	0.995	0.943

Figure 5.51: TP Rate (Cross-Validation: 30% Split)



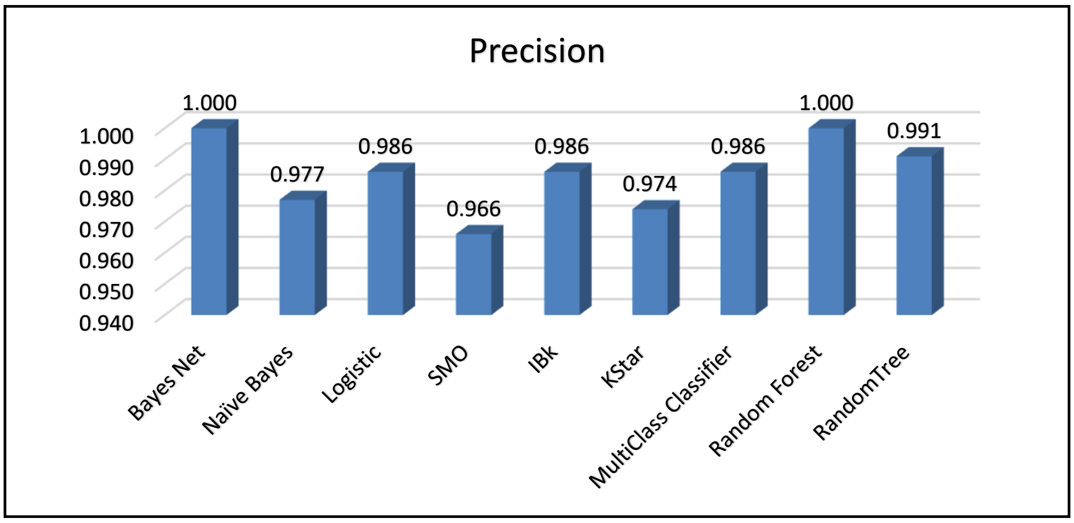
According to the performance measure TP rate it was found that the highest true positive rate was of the classifiers Bayes Net, Logistic, SMO, KStar, MultiClass Classifier, Random Forest and Random Tree with value 1.0, whereas the lowest TP rate was found to be of the classifiers Naïve Bayes and IBK with values 0.997 respectively. Overall, it can be interpreted that the most appropriate classifier based on the performance measure TP rate are seven Algorithms.

Figure 5.52: FP Rate (Cross-Validation: 30% Split)



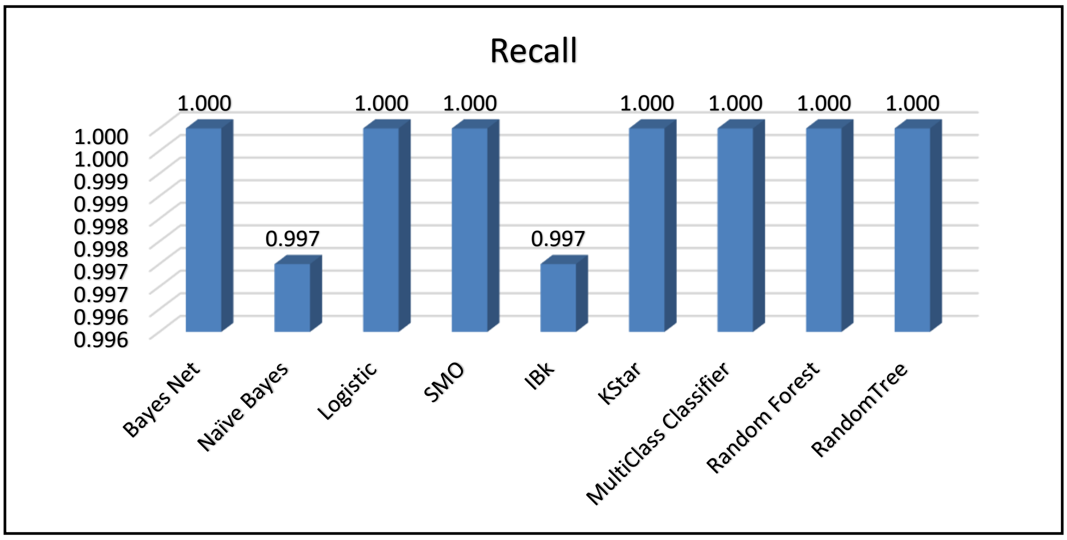
Based on the performance measure FP rate it was found that the lowest false positive rate was of the classifiers Bayes Net and Random Forest with value 0.000, whereas the highest FP rate was found to be of the classifier SMO with value 0.434 respectively. Overall, it can be interpreted that the most appropriate classifier based on the performance measure FP rate is found to be Bayes Net and Random Forest with lowest FP rate value.

Figure 5.53: Precision (Cross-Validation: 30% Split)



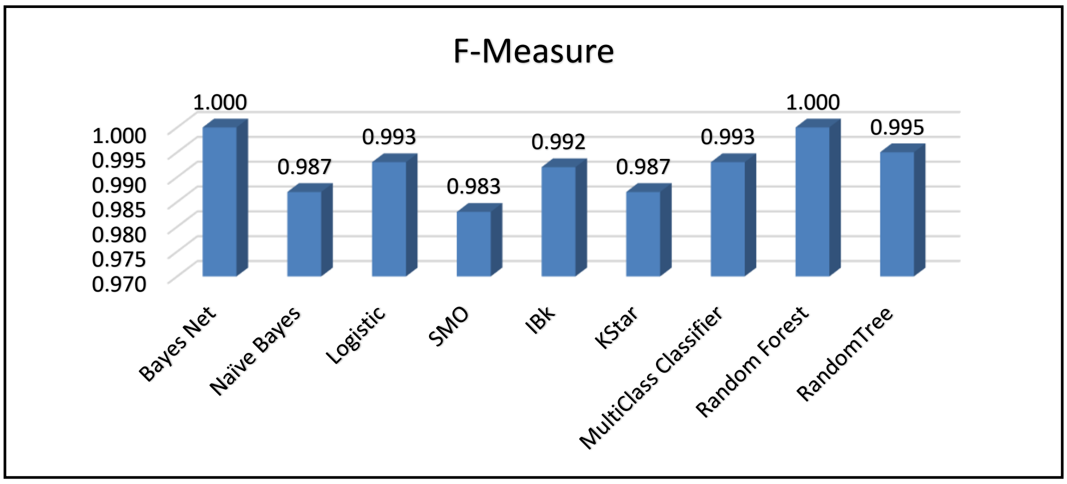
According to the performance measure precision it was found that the highest precision value was of the classifier Bayes Net and Random Forest with value 1.0, followed by 0.991 of Random Tree respectively whereas the lowest precision value was found to be of the classifiers SMO with values 0.966 respectively. Overall, it can be interpreted that the most appropriate classifier based on the performance measure precision is found to be the Bayes Net and Random Forest.

Figure 5.54: Recall (Cross-Validation: 30% Split)



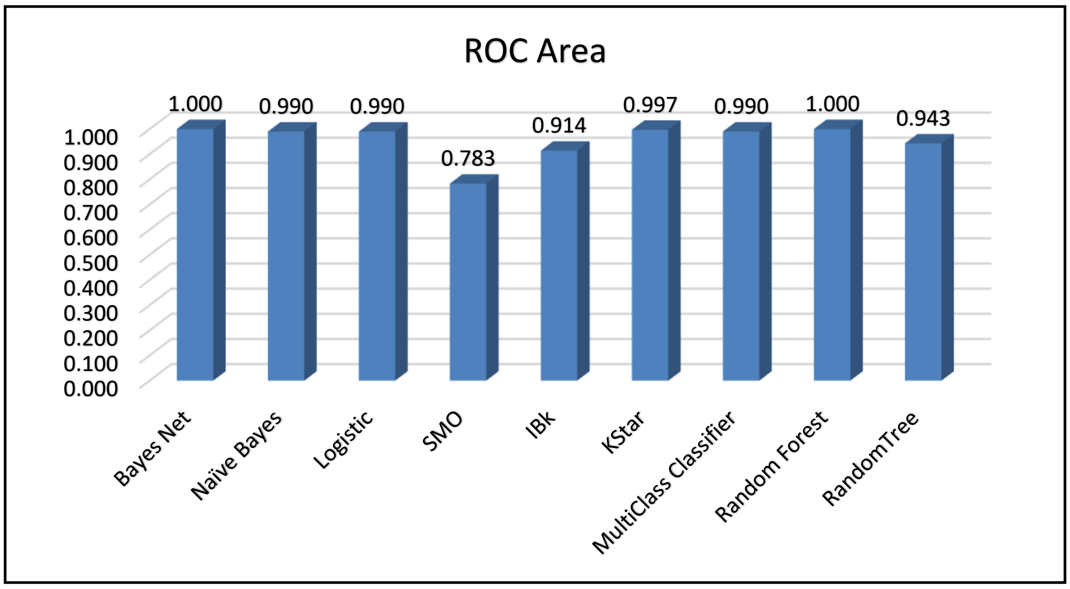
Based on the performance measure recall it was found that the highest recall value was of the classifiers Bayes Net, Logistic, SMO, KStar, MultiClass Classifier, Random Forest and Random Tree with value 1.0, whereas the lowest recall value was found to be of the classifiers Naïve Bayes and IBK with values 0.997 respectively. Overall, it can be interpreted that the most appropriate classifier based on the performance measure recall is found to be Seven Algorithms.

Figure 5.55: F-Measure (Cross-Validation: 30% Split)



According to the performance measure F-Measure it was found that the highest F-Measure value was of the classifiers Bayes Net and Random Forest with value 1.00, followed by 0.995 of Random Tree respectively whereas the lowest F-Measure value was found to be of the classifiers SMO classifier with values 0.983 respectively. Overall, it can be interpreted that the most appropriate classifier based on the performance measure F-Measure is found to be Bayes Net and Random Forest.

Figure 5.56 : ROC Area (Cross-Validation: 30% Split)



Based on the performance measure ROC it was found that the highest ROC Area value was of the classifiers Bayes Net and Random Forest with value 1.0, followed by 0.997 of KStar respectively whereas the lowest ROC Area value was found to be of the classifiers SMO with values 0.783 respectively. Overall, it can be interpreted the most appropriate classifier based on the performance measure ROC Area is found to be Bayes Net and Random Forest.

In summary, performance measurements play a critical role in evaluating the effectiveness of machine learning algorithms, providing insight into their ability to make accurate predictions and transform appropriately to new, unseen data. Choosing the most appropriate metric depends on the nature of your problem, the characteristics of your data, and your analysis goals.

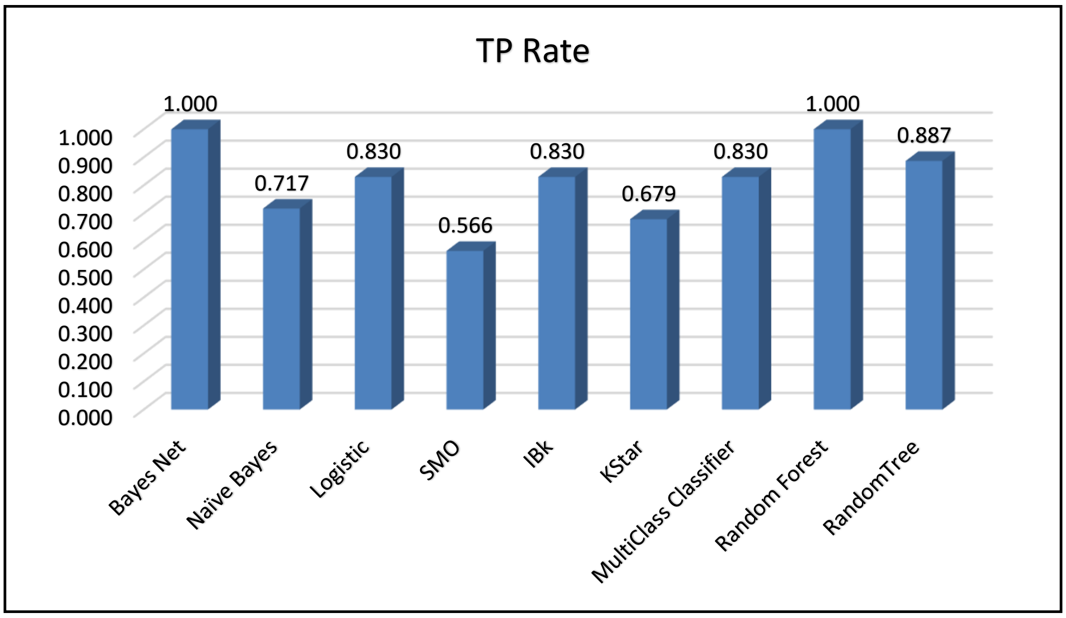
iii. Confusion Matrix Parameters – Heavy Traffic

Heavy Traffic generates huge amounts of data from the various IOT sensors. These data sets can be used by Machine Learning Algorithms to develop prediction models. Table 5.13 shows the Confusion Matrix parameters TP Rate, FP Rate, Precision, Recall, F-Measure and ROC Area obtained for Heavy Traffic conditions.

Table 5.13: Classifiers Performance Measure Class Label: Heavy Traffic: Cross Validation: 30% Split

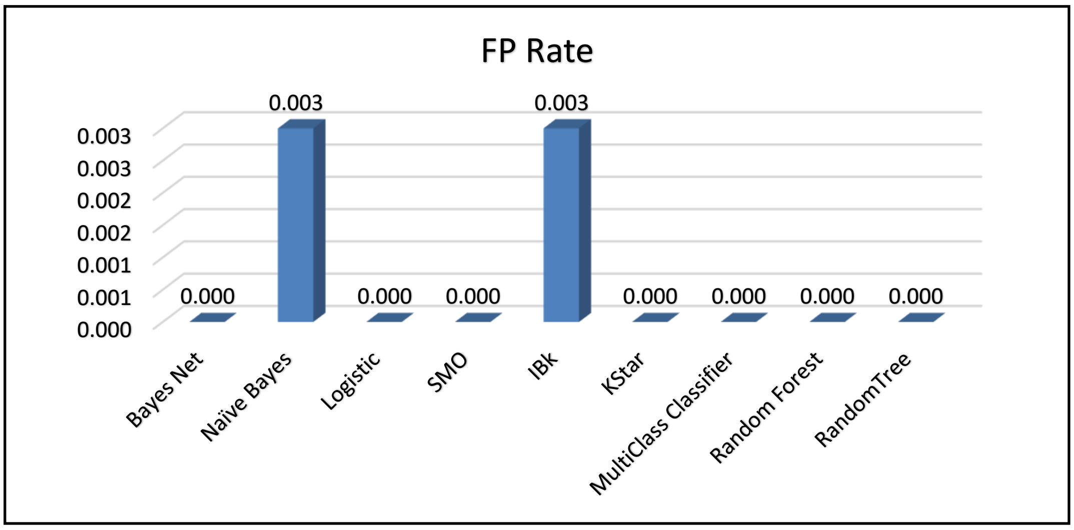
Classifier	TP Rate	FP Rate	Precision	Recall	F-Measure	ROC Area
Bayes Net	1.000	0.000	1.000	1.000	1.000	1.000
Naïve Bayes	0.717	0.003	0.950	0.717	0.817	0.990
Logistic	0.830	0.000	1.000	0.830	0.907	0.999
SMO	0.566	0.000	1.000	0.566	0.723	0.783
IBk	0.830	0.003	0.957	0.830	0.889	0.914
KStar	0.679	0.000	1.000	0.679	0.809	0.097
MultiClass Classifier	0.830	0.000	1.000	0.830	0.907	0.999
Random Forest	1.000	0.000	1.000	1.000	1.000	1.000
RandomTree	0.887	0.000	1.000	0.887	0.940	0.943

Figure 5.57: TP Rate (Cross-Validation: 30% Split)



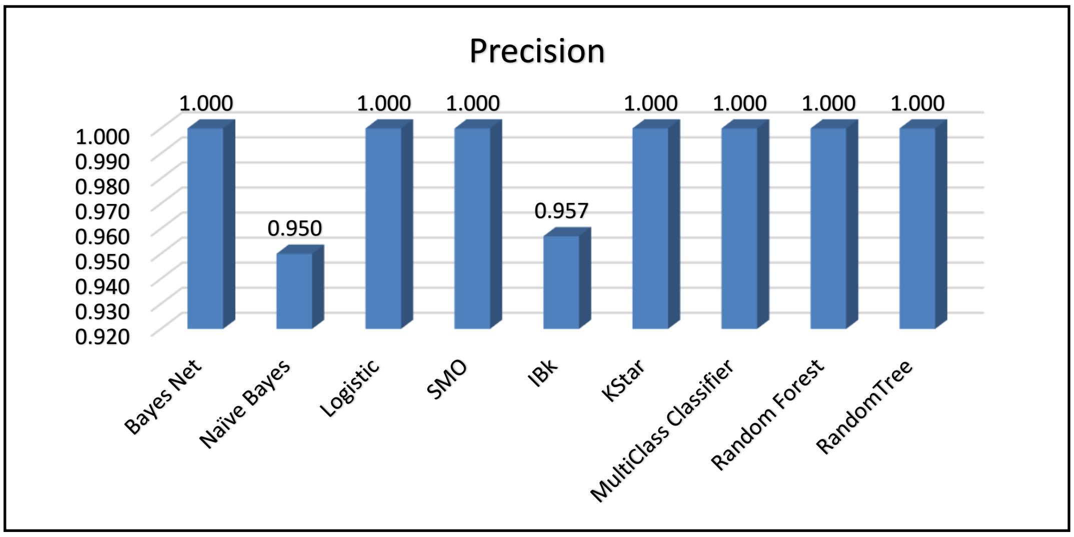
According to the performance measure TP rate for class label: Heavy Traffic it was found that the highest true positive rates were of the classifier Bayes Net and Random Forest with value 1.0, followed by 0.887 of Random Tree respectively whereas the lowest TP rate was found to be of the classifiers SMO with value 0.566 respectively. Overall, it can be interpreted that the most appropriate classifier based on the performance measure TP rate is Bayes Net and Random Forest.

Figure 5.58: FP Rate (Cross-Validation: 30% Split)



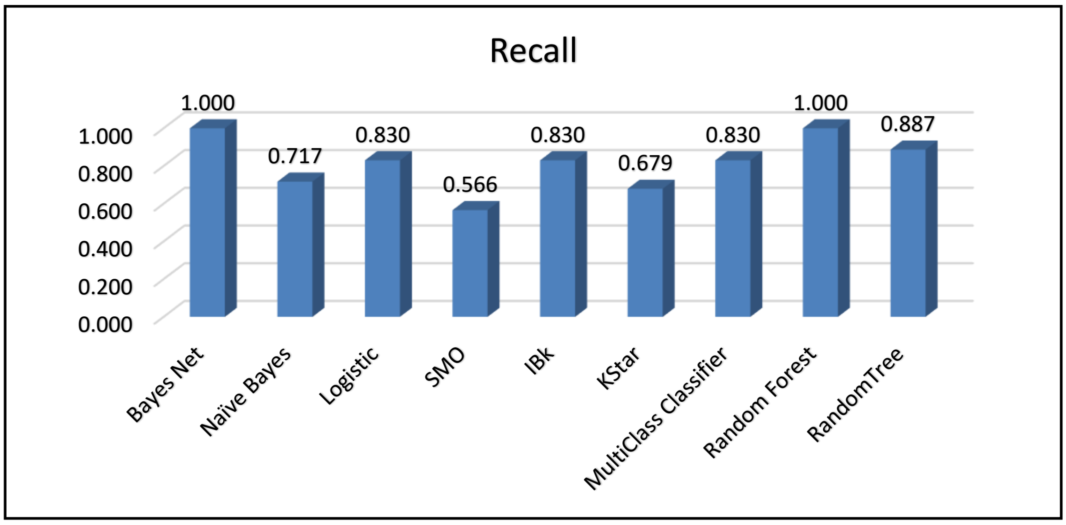
Based on the performance measure FP rate for class label: Heavy Traffic it was found that the lowest false positive rates were of the classifier Bayes Net, Logistic, SMO, KStar, MultiClass Classifier, Random Forest, and Random Tree with value 0.00 each, whereas the highest FP rate was found to be of the classifiers Naïve Bayes and IBK with values 0.003.

Figure 5.59 : Precision (Cross-Validation: 30% Split)



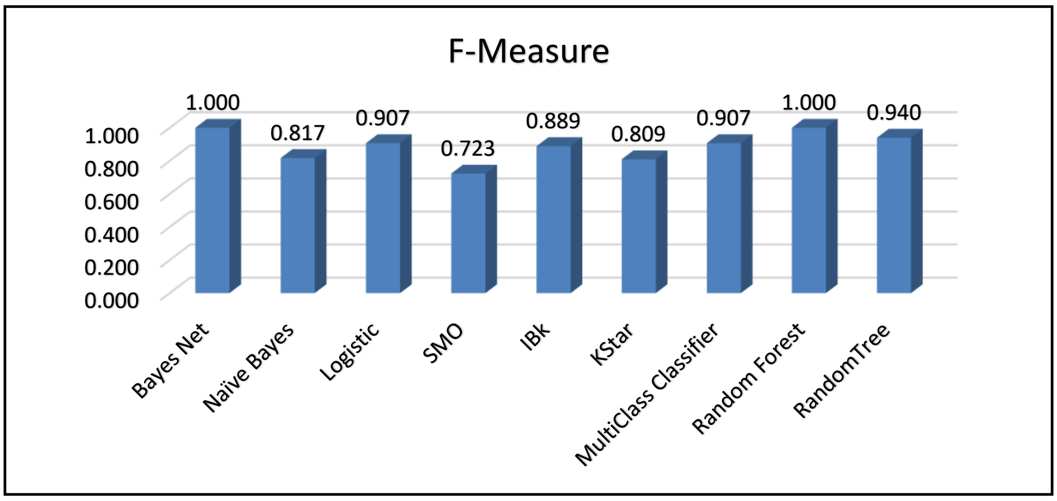
According to the performance measure precision class label: Heavy Traffic it was found that the highest precision value was of the classifiers Bayes Net, Logistic, SMO, KStar, MultiClass Classifier, Random Forest, and Random Tree with value 1.0, whereas the lowest precision value was found to be of the classifiers Naïve Bayes with value 0.950 respectively. Overall, it can be interpreted that the most appropriate classifier based on the performance measure precision is found to be Seven Algorithms.

Figure 5.60: Recall (Cross-Validation: 30% Split)



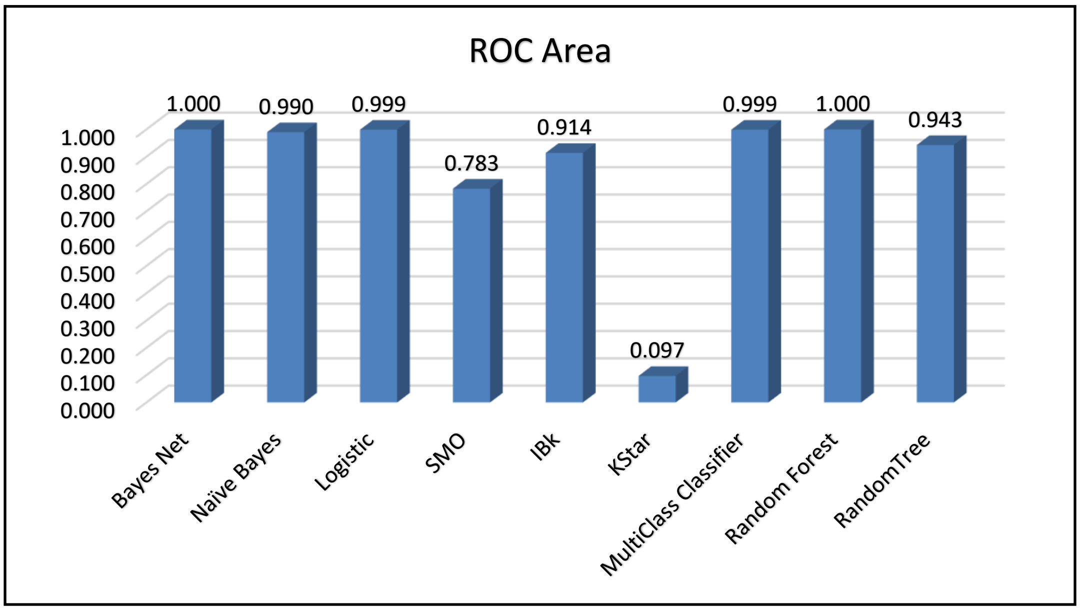
Based on the performance measure recall class label: Heavy Traffic it was found that the highest recall value was of the classifiers Bayes Net and Random Forest with value 1.0, followed by 0.887 of Random Tree respectively whereas the lowest recall value was found to be of the classifiers SMO with value 0.566 respectively. Overall, it can be interpreted that the most appropriate classifier based on the performance measure recall is found to be Bayes Net and Random Forest.

Figure 5.61: F-Measure (Cross-Validation: 30% Split)



According to the performance measure F-Measure class label: Heavy Traffic it was found that the highest F-Measure value was of the classifiers Bayes Net and Random Forest with value 1.0, followed by 0.940 of Random Tree respectively whereas the lowest F-Measure value was found to be of the classifier SMO classifier with value 0.723 respectively. Overall, it can be interpreted that the most appropriate classifier based on the performance measure F-Measure is found to be Bayes Net and Random Forest.

Figure 5.62: ROC Area (Cross-Validation: 30% Split)



Based on the performance measure ROC class label: Heavy Traffic it was found that the highest ROC Area values were of the classifiers Bayes Net and Random Forest with value 1.00, followed by 0.999 of Logistic and MultiClass Classifier respectively whereas the lowest ROC Area value was found to be of the classifier KStar with value 0.097 respectively. Overall, it can be interpreted the most appropriate classifier based on the performance measure ROC Area are found to be Bayes Net and Random Forest.

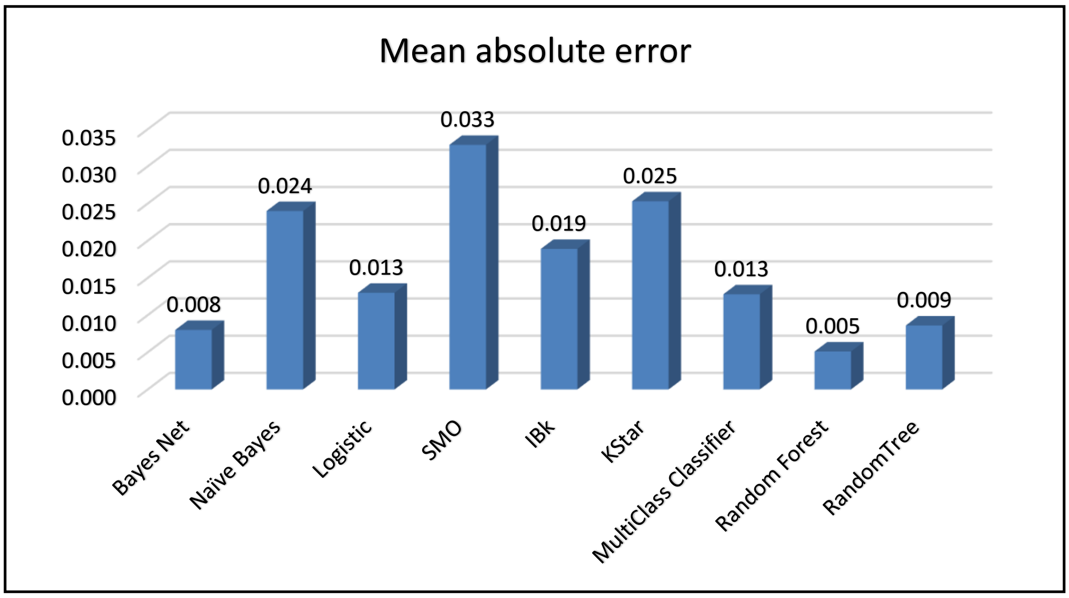
In conclusion the 30% split Cross validation not only increases the reliability of model but also gives insights of model and explains how model behave under different conditions. Sometimes 80:20 model is also used to face real life applications where 80% data is used for Training the model and 20% data is used for testing the model. More percentage data for training also increases model effectiveness and gives superior model performance, paving the way for more trustworthy and impactful model for real life applications.

B. Error Measure Results

Table 5.14: Classifiers and Error Measures (Cross Validation: 30% Split)

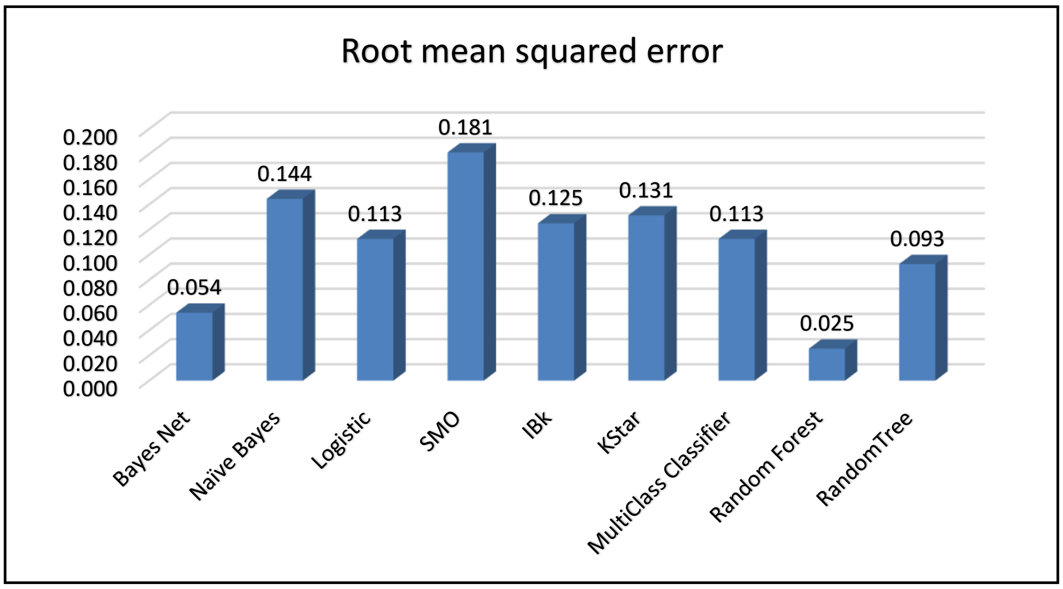
Classifier	Mean absolute error	Root mean squared error	Relative absolute error	Root relative squared error
Bayes Net	0.008	0.054	7.50%	20.50%
Naïve Bayes	0.024	0.144	20.34%	54.31%
Logistic	0.013	0.113	10.86%	42.35%
SMO	0.033	0.181	27.88%	68.19%
IBk	0.019	0.125	16.05%	47.02%
KStar	0.025	0.131	21.48%	49.39%
MultiClass Classifier	0.013	0.113	10.86%	42.35%
Random Forest	0.005	0.025	4.30%	9.54%
RandomTree	0.009	0.093	7.27%	34.83%

Figure 5.63: Mean Absolute Error (Cross-Validation: 30% Split)



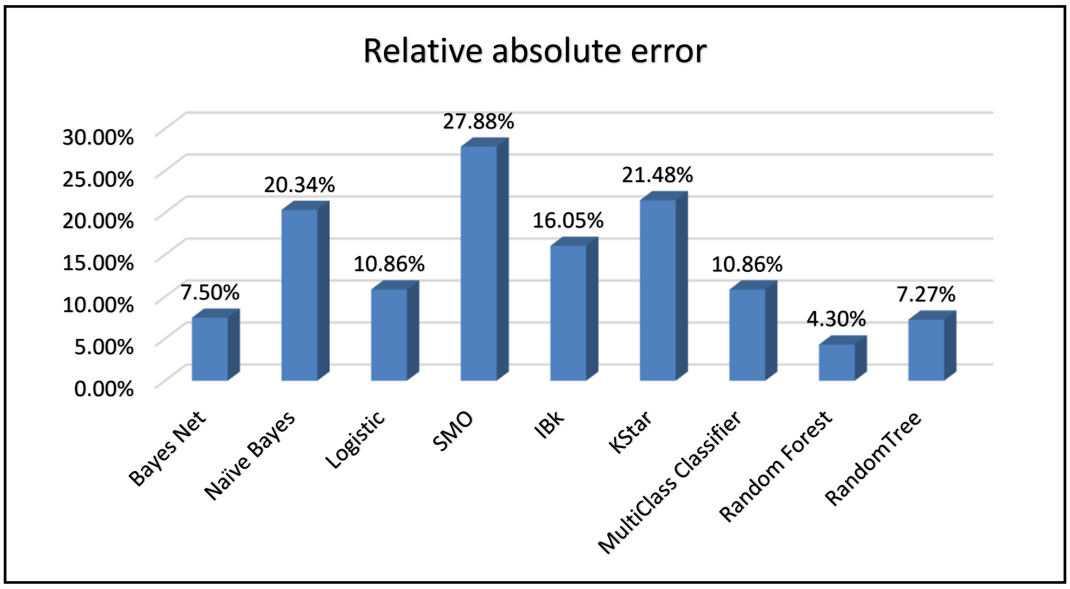
The mean absolute error is found to be lowest in the case of Random Forest with the value 0.005 Whereas the mean absolute error value of SMO is found to be highest with value 0.033.

Figure 5.64: Root Mean Squared Error (Cross-Validation: 30% Split)



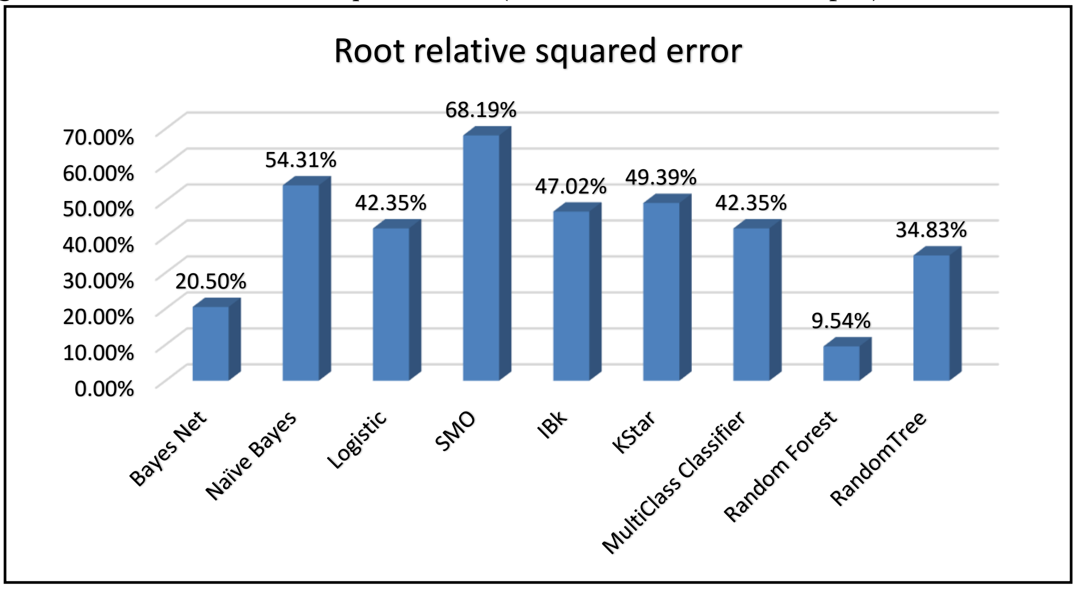
The root mean squared error value is found to be highest in case of SMO classifier with the value of 0.181 respectively whereas the lowest value is found to be of Random Forest with value 0.025. So, it can be interpreted that based on the measure RMSE the most appropriate algorithm is found to be Random Forest at configuration setting – 30% Split Method.

Figure 5.65: Relative Absolute Error (Cross-Validation: 30% Split)



Accordingly, the relative absolute error value is found to be lowest in case of Random Forest classifier with 4.30% whereas the highest relative absolute error percentage value is found to be in case of SMO with percentage value 27.88% respectively. Based on the measure RAE the most appropriate algorithm is found to be Random Forest.

Figure 5.66: Root Relative Square Error (Cross-Validation: 30% Split)



The root relative squared error value is found to be lowest in case of Random Forest classifier with 9.54% whereas the root relative squared error percentage value is found to be highest in case of SMO with percentage value of 68.19%. So, it can be interpreted that based on the measure RRSE the most appropriate algorithm is found to be Random Forest with lowest percentage value when evaluated at configuration setting – 30% Split cross validation.

In summary, it is important to understand error measures in Machine learning for assessing the performance of model properly and making informed decisions. The choice of specific metrics depends on the nature of the problem, characteristics of the dataset, and goals of the analysis.

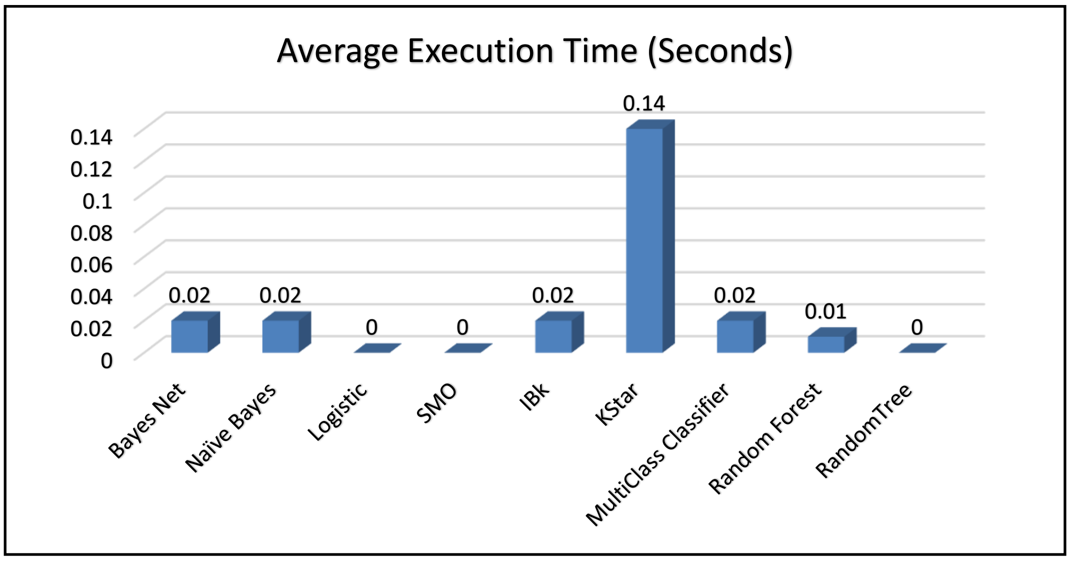
C. Execution Time Results

The execution time of a machine learning algorithm refers to the time it takes for the algorithm to process and analyse input data, train the model (if applicable), and produce predictions or results. Execution time is an important factor when evaluating the efficiency and scalability of machine learning algorithms, especially when dealing with large data sets and real-time applications. The Average Execution Time of Nine Classifier Algorithm for Cross Validation 30% split is given below.

Table 5.15: Classifiers and Average Execution Time (Cross Validation: 30% Split)

Classifier	Average Execution Time (Seconds)
Bayes Net	0.02
Naïve Bayes	0.02
Logistic	0
SMO	0
IBk	0.02
KStar	0.14
MultiClass Classifier	0.02
Random Forest	0.01
Random Tree	0

Figure 5.67: Average Execution Time (Cross-Validation: 30% Split)



According to the performance measure average execution time it was found that the lowest average execution time were of the classifiers Logistic, SMO and Random Tree with values 0.00 each whereas the highest average execution time was found to be of the classifier KStar classifier with value 0.14 respectively. Overall, it can be interpreted that the most appropriate classifiers based on the performance measure average execution time are found to be Logistic, SMO and Random Tree.

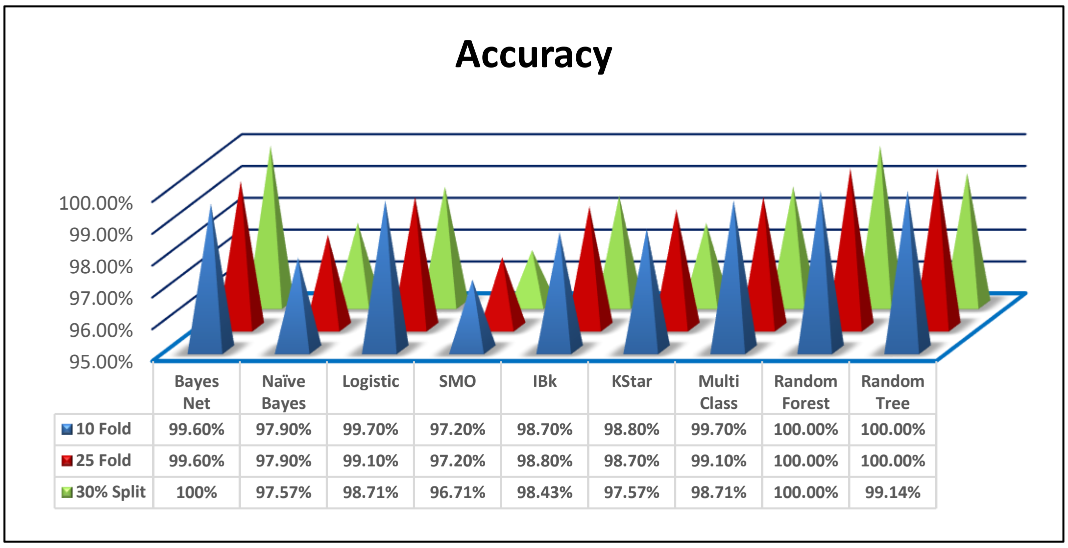
5.3.6 Consolidated Result

Nine Machine Learning Algorithms are Analyzed using Cross – Validation of 10-Fold, 25-Fold and 30% split on various Performance Accuracy Measures, Confusion Matrix Parameters and Error Measures in two different conditions “Low Traffic” and “Heavy Traffic”. Summary of 10 – Fold, 25 – Fold and 30% split is shown below in 3D plots.

A. Performance Measures

i. Accuracy Measures

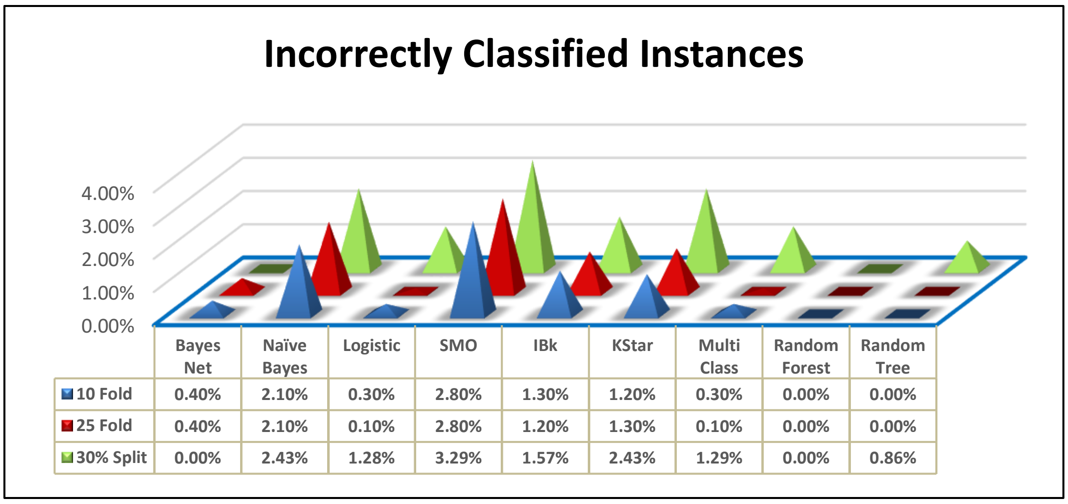
Figure 5.68: Accuracy (Cross-Validation: 10-Fold, 25-Fold and 30% Split)



	Bayes Net	Naïve Bayes	Logistic	SMO	IBk	KStar	Multi Class	Random Forest	Random Tree
🟦 10 Fold	99.60%	97.90%	99.70%	97.20%	98.70%	98.80%	99.70%	100.00%	100.00%
🟥 25 Fold	99.60%	97.90%	99.10%	97.20%	98.80%	98.70%	99.10%	100.00%	100.00%
🟩 30% Split	100%	97.57%	98.71%	96.71%	98.43%	97.57%	98.71%	100.00%	99.14%

From above three-Dimensional plot it is clear that the maximum average Accuracy score for Random Forest is 100% therefore it is concluded that Random Forest is best algorithms for getting best Accuracy using Cross – Validation of 10-Fold, 25-Fold and 30% Split.

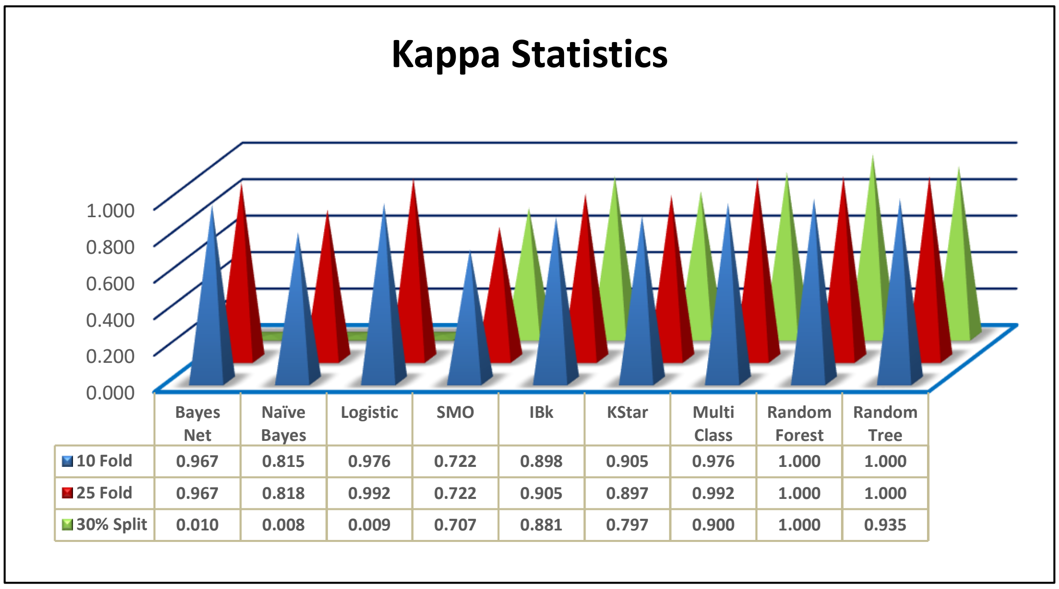
Figure 5.69: Incorrectly Classified Instances (10-Fold, 25-Fold and 30% Split)



	Bayes Net	Naïve Bayes	Logistic	SMO	IBk	KStar	Multi Class	Random Forest	Random Tree
🟦 10 Fold	0.40%	2.10%	0.30%	2.80%	1.30%	1.20%	0.30%	0.00%	0.00%
🟥 25 Fold	0.40%	2.10%	0.10%	2.80%	1.20%	1.30%	0.10%	0.00%	0.00%
🟩 30% Split	0.00%	2.43%	1.28%	3.29%	1.57%	2.43%	1.29%	0.00%	0.86%

From above three-Dimensional plot it is clear that the minimum average incorrectly Classified Instance score for Random Forest is 0% therefore it is concluded that Random Forest is best algorithms for getting best Incorrectly Classified Instances using Cross – Validation of 10-Fold, 25-Fold and 30% Split.

Figure 5.70: Kappa Statistics (Cross-Validation: 10-Fold, 25-Fold and 30% Split)

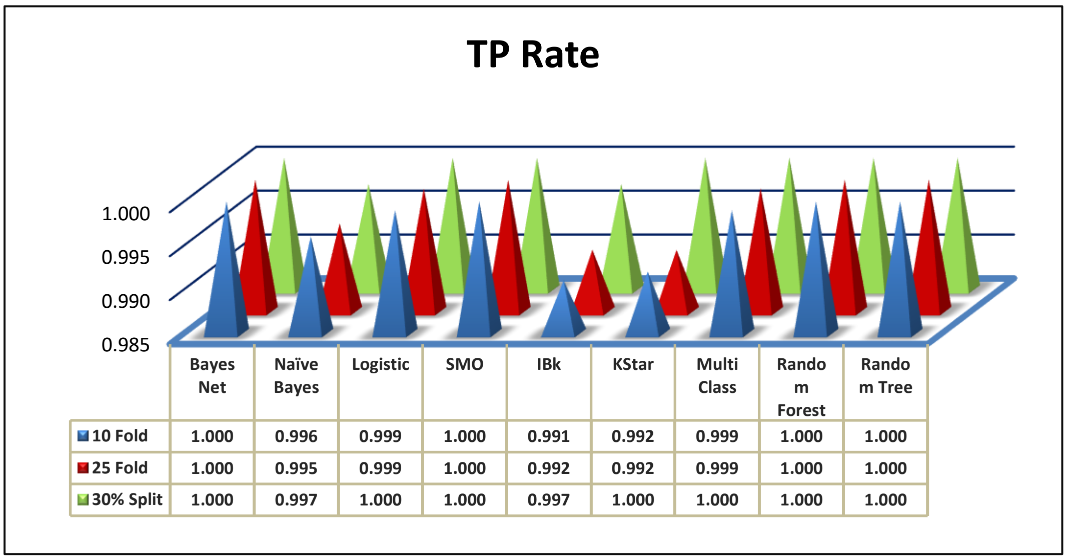


	Bayes Net	Naïve Bayes	Logistic	SMO	IBk	KStar	Multi Class	Random Forest	Random Tree
🟦 10 Fold	0.967	0.815	0.976	0.722	0.898	0.905	0.976	1.000	1.000
🟥 25 Fold	0.967	0.818	0.992	0.722	0.905	0.897	0.992	1.000	1.000
🟩 30% Split	0.010	0.008	0.009	0.707	0.881	0.797	0.900	1.000	0.935

From above three-Dimensional plot it is clear that the maximum average Kappa Statistics score for Random Forest is 1.0% therefore it is concluded that Random Forest is best algorithms for getting best Kappa Statistics using Cross – Validation of 10-Fold, 25-Fold and 30% Split.

ii. Confusion Matrix Parameters – Low Traffic

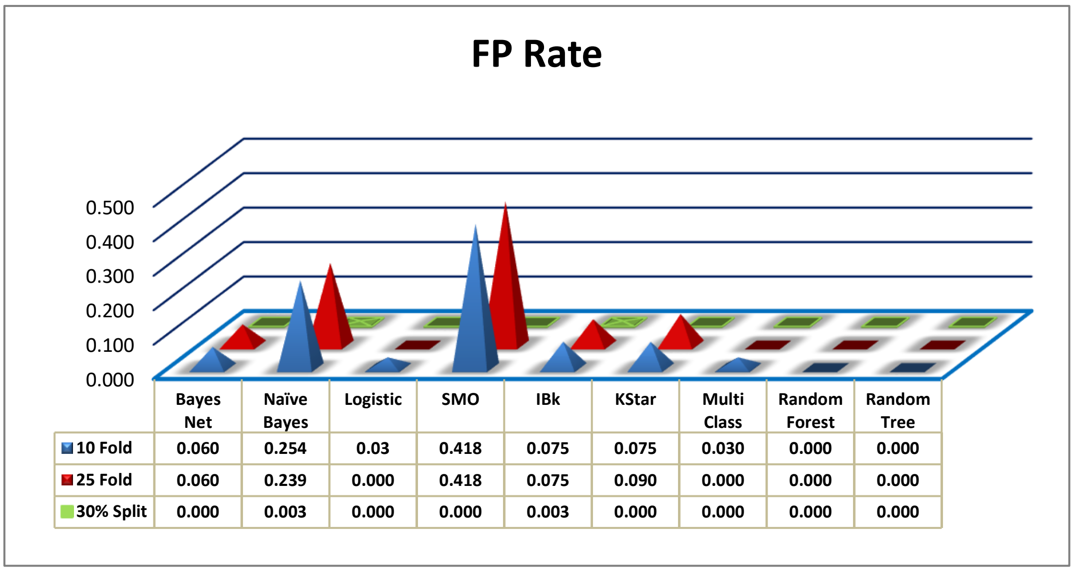
Figure 5.71: TP Rate (Cross-Validation: 10-Fold, 25-Fold and 30% Split)



	Bayes Net	Naïve Bayes	Logistic	SMO	IBk	KStar	Multi Class	Rando m Forest	Rando m Tree
🟦 10 Fold	1.000	0.996	0.999	1.000	0.991	0.992	0.999	1.000	1.000
🟥 25 Fold	1.000	0.995	0.999	1.000	0.992	0.992	0.999	1.000	1.000
🟩 30% Split	1.000	0.997	1.000	1.000	0.997	1.000	1.000	1.000	1.000

From above three-Dimensional plot it is clear that the maximum average score for Bayes Net, SMO, Random Forest and Random Tree is 1.0 therefore it is concluded that Bayes Net, SMO, Random Forest and Random Tree are best algorithms for getting best True Positive Rate using Cross – Validation of 10-Fold, 25-Fold and 30% Split in Low Traffic Conditions.

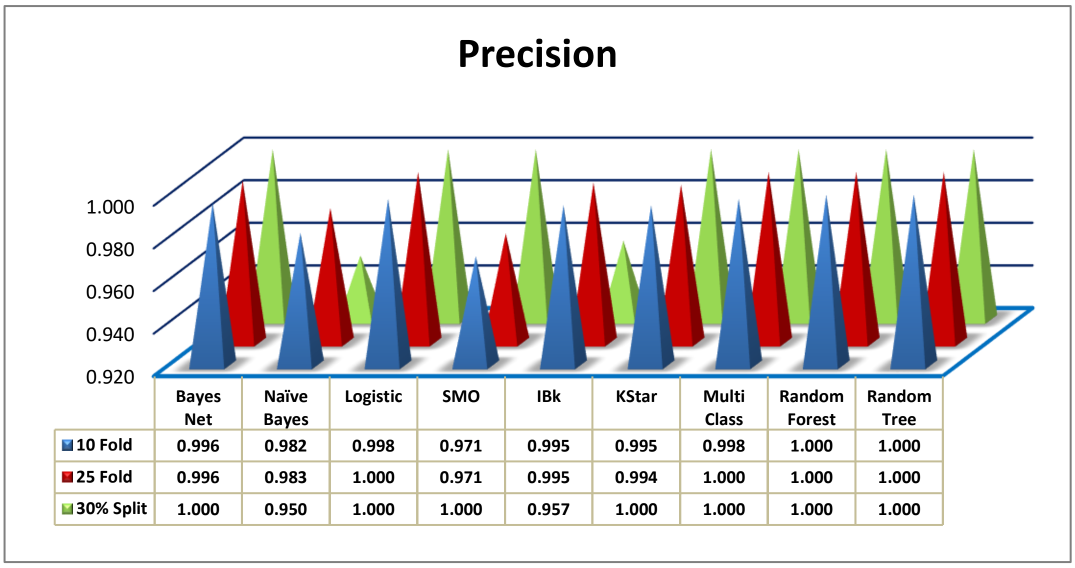
Figure 5.72: FP Rate (Cross-Validation: 10-Fold, 25-Fold and 30% Split)



	Bayes Net	Naïve Bayes	Logistic	SMO	IBk	KStar	Multi Class	Random Forest	Random Tree
🟦 10 Fold	0.060	0.254	0.03	0.418	0.075	0.075	0.030	0.000	0.000
🟥 25 Fold	0.060	0.239	0.000	0.418	0.075	0.090	0.000	0.000	0.000
🟩 30% Split	0.000	0.003	0.000	0.000	0.003	0.000	0.000	0.000	0.000

From above three-Dimensional plot it is clear that the minimum average score for Random Forest and Random Tree is 0.0, therefore it is concluded that Random Forest and Random Tree are the best algorithms for getting best False Positive Rate using Cross – Validation of 10-Fold, 25-Fold and 30% Split in Low Traffic Conditions.

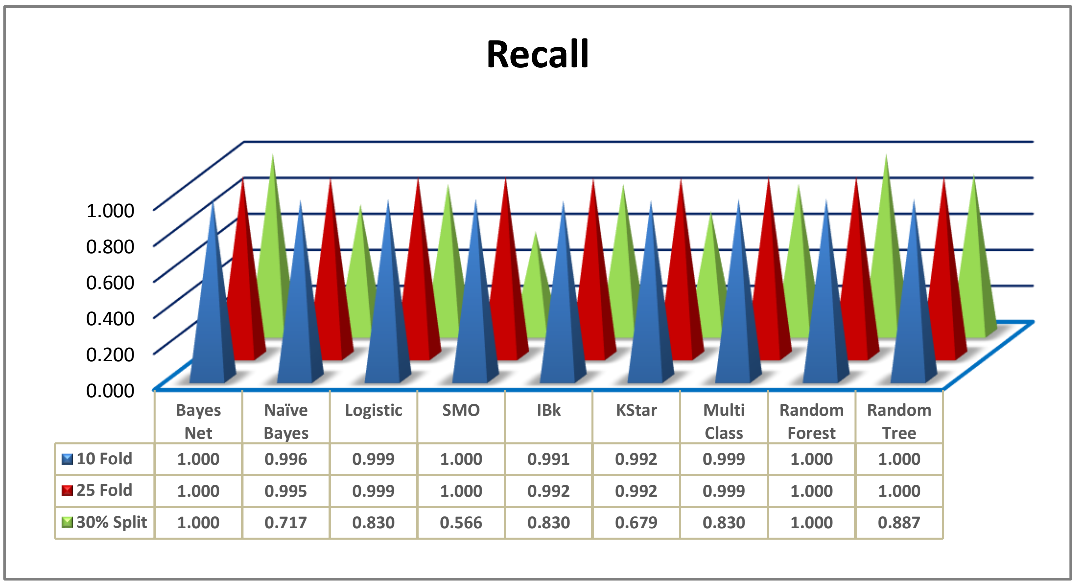
Figure 5.73: Precision (Cross-Validation: 10-Fold, 25-Fold and 30% Split)



	Bayes Net	Naïve Bayes	Logistic	SMO	IBk	KStar	Multi Class	Random Forest	Random Tree
🟦 10 Fold	0.996	0.982	0.998	0.971	0.995	0.995	0.998	1.000	1.000
🟥 25 Fold	0.996	0.983	1.000	0.971	0.995	0.994	1.000	1.000	1.000
🟩 30% Split	1.000	0.950	1.000	1.000	0.957	1.000	1.000	1.000	1.000

From above three Dimensional plot it is clear that the maximum average score for Random Forest and Random Tree is 1.0, therefore it is concluded that Random Forest and Random Tree are the best algorithms for getting best Precision using Cross – Validation of 10-Fold, 25-Fold and 30% Split in Low Traffic Conditions.

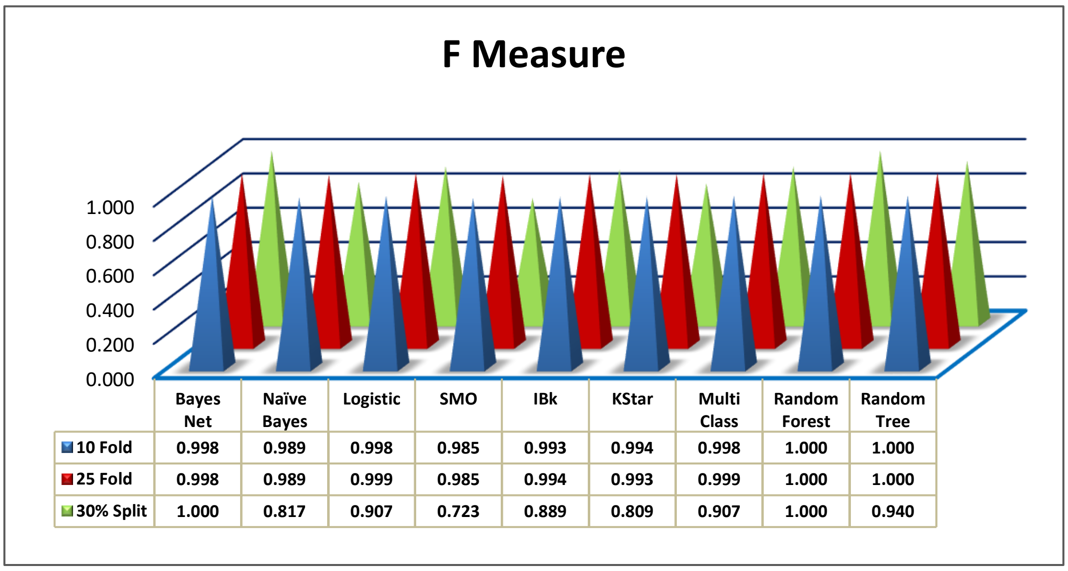
Figure 5.74: Recall (Cross-Validation: 10-Fold, 25-Fold and 30% Split)



	Bayes Net	Naïve Bayes	Logistic	SMO	IBk	KStar	Multi Class	Random Forest	Random Tree
🟦 10 Fold	1.000	0.996	0.999	1.000	0.991	0.992	0.999	1.000	1.000
🟥 25 Fold	1.000	0.995	0.999	1.000	0.992	0.992	0.999	1.000	1.000
🟩 30% Split	1.000	0.717	0.830	0.566	0.830	0.679	0.830	1.000	0.887

From above three-Dimensional plot it is clear that the maximum average score for Bayes Net and Random Forest is 1.0 therefore it is concluded that Bayes Net and Random Forest are the best algorithms for getting best Recall using Cross – Validation of 10-Fold, 25-Fold and 30% Split in Low Traffic Conditions.

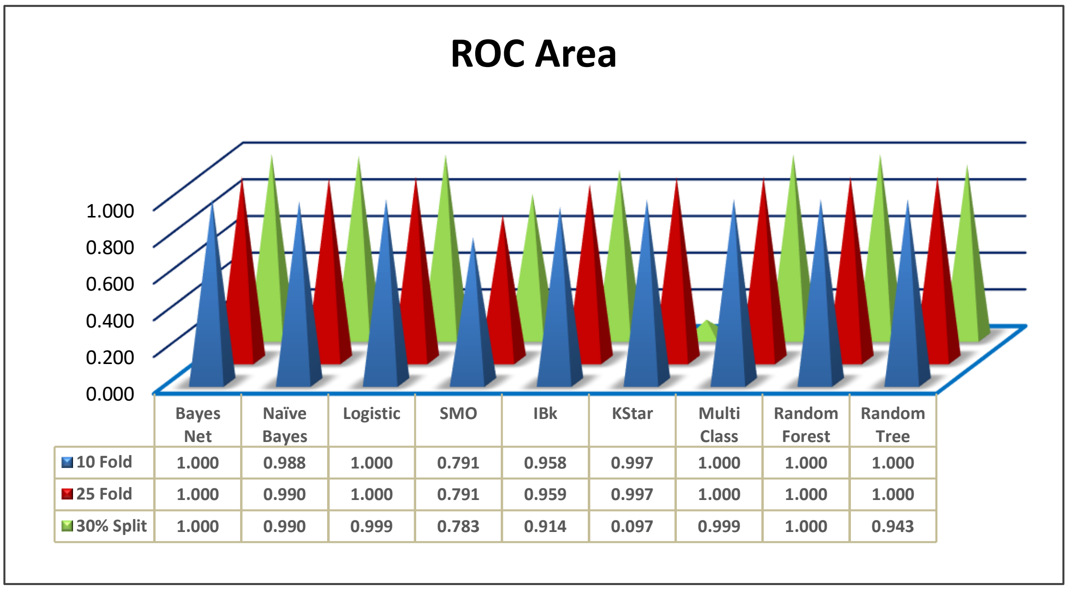
Figure 5.75: F Measure (Cross-Validation: 10-Fold, 25-Fold and 30% Split)



	Bayes Net	Naïve Bayes	Logistic	SMO	IBk	KStar	Multi Class	Random Forest	Random Tree
🟦 10 Fold	0.998	0.989	0.998	0.985	0.993	0.994	0.998	1.000	1.000
🟥 25 Fold	0.998	0.989	0.999	0.985	0.994	0.993	0.999	1.000	1.000
🟩 30% Split	1.000	0.817	0.907	0.723	0.889	0.809	0.907	1.000	0.940

From above three Dimensional plot it is clear that the maximum average score for Random Forest is 1.0, therefore it is concluded that Random Forest is the best algorithms for getting best F Measure using Cross – Validation of 10-Fold, 25-Fold and 30% Split

Figure 5.76: ROC Area (Cross-Validation: 10-Fold, 25-Fold and 30% Split)

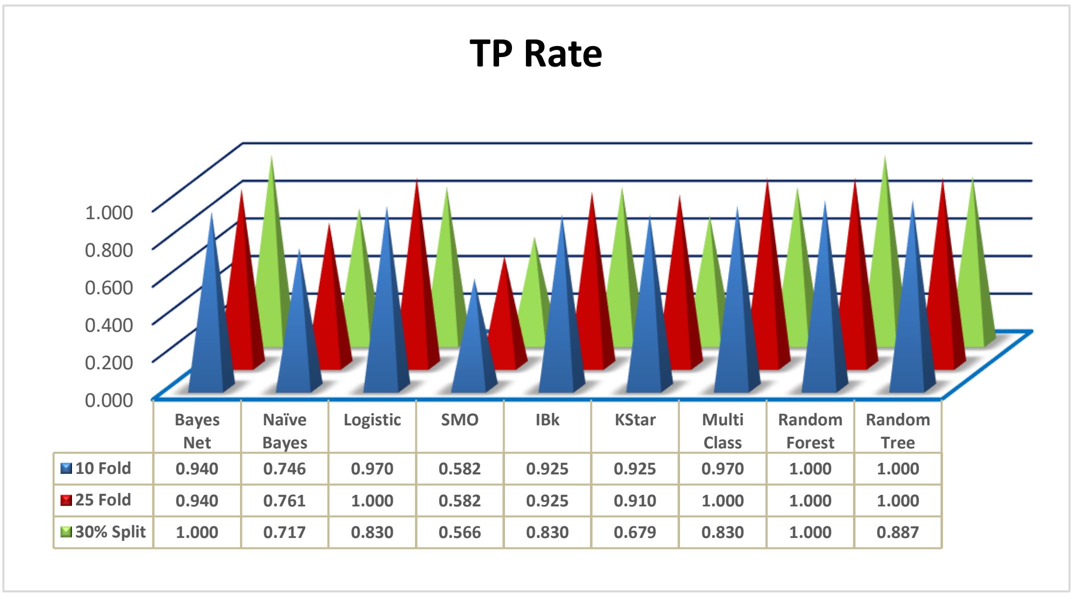


	Bayes Net	Naïve Bayes	Logistic	SMO	IBk	KStar	Multi Class	Random Forest	Random Tree
🟦 10 Fold	1.000	0.988	1.000	0.791	0.958	0.997	1.000	1.000	1.000
🟥 25 Fold	1.000	0.990	1.000	0.791	0.959	0.997	1.000	1.000	1.000
🟩 30% Split	1.000	0.990	0.999	0.783	0.914	0.097	0.999	1.000	0.943

From above three-Dimensional plot it is clear that the maximum average score for Bayes Net and Random Forest is 1.0 therefore it is concluded that Bayes Net and Random Forest are the best algorithms for getting best ROC Area using Cross – Validation of 10-Fold, 25-Fold and 30% Split in Low Traffic Conditions.

iii. Confusion Matrix Parameters – Heavy Traffic

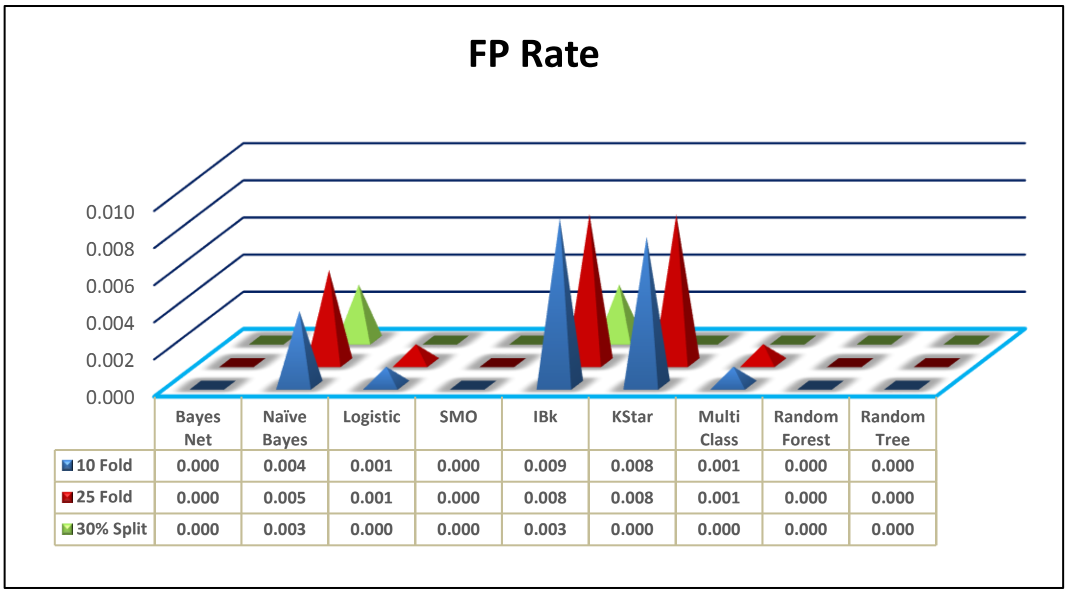
Figure 5.77: TP Rate (Cross-Validation: 10-Fold, 25-Fold and 30% Split)



	Bayes Net	Naïve Bayes	Logistic	SMO	IBk	KStar	Multi Class	Random Forest	Random Tree
🟦 10 Fold	0.940	0.746	0.970	0.582	0.925	0.925	0.970	1.000	1.000
🟥 25 Fold	0.940	0.761	1.000	0.582	0.925	0.910	1.000	1.000	1.000
🟩 30% Split	1.000	0.717	0.830	0.566	0.830	0.679	0.830	1.000	0.887

From above three Dimensional plot it is clear that the maximum average TP Rate score for Random Forest is 1.0, therefore it is concluded that Random Forest is the best algorithms for getting best TP Rate using Cross – Validation of 10-Fold, 25-Fold and 30% Split in heavy traffic conditions.

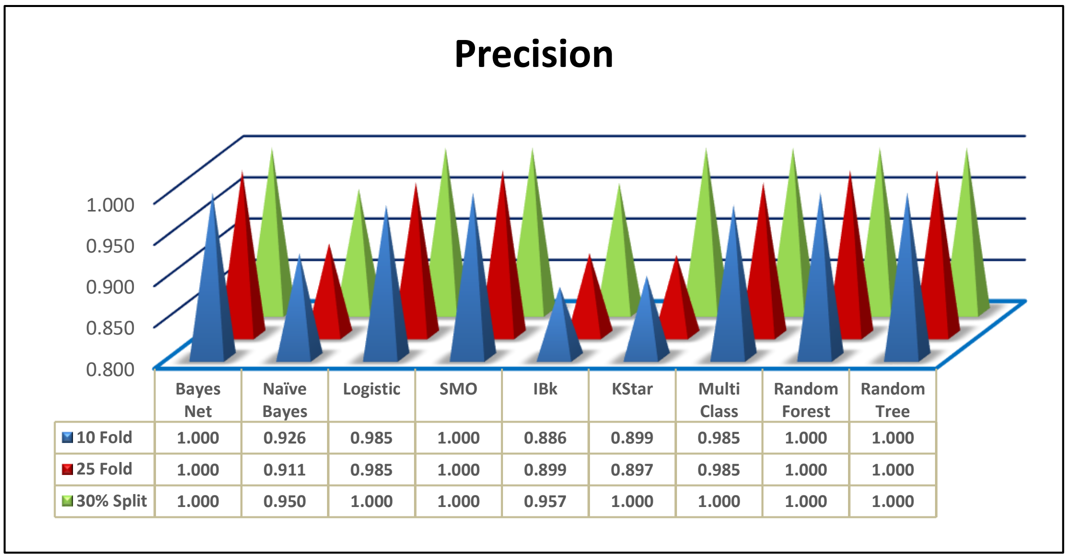
Figure 5.78: FP Rate (Cross-Validation: 10-Fold, 25-Fold and 30% Split)



	Bayes Net	Naïve Bayes	Logistic	SMO	IBk	KStar	Multi Class	Random Forest	Random Tree
🟦 10 Fold	0.000	0.004	0.001	0.000	0.009	0.008	0.001	0.000	0.000
🟥 25 Fold	0.000	0.005	0.001	0.000	0.008	0.008	0.001	0.000	0.000
🟩 30% Split	0.000	0.003	0.000	0.000	0.003	0.000	0.000	0.000	0.000

From above three-Dimensional plot it is clear that the minimum average FP Rate score for Bayes Net, SMO, Random Forest and Random Tree is 0.0, therefore it is concluded that Bayes Net, SMO, Random Forest and Random Tree are the best algorithms for getting best False Positive Rate using Cross – Validation of 10-Fold, 25-Fold and 30% Split in Heavy Traffic Conditions.

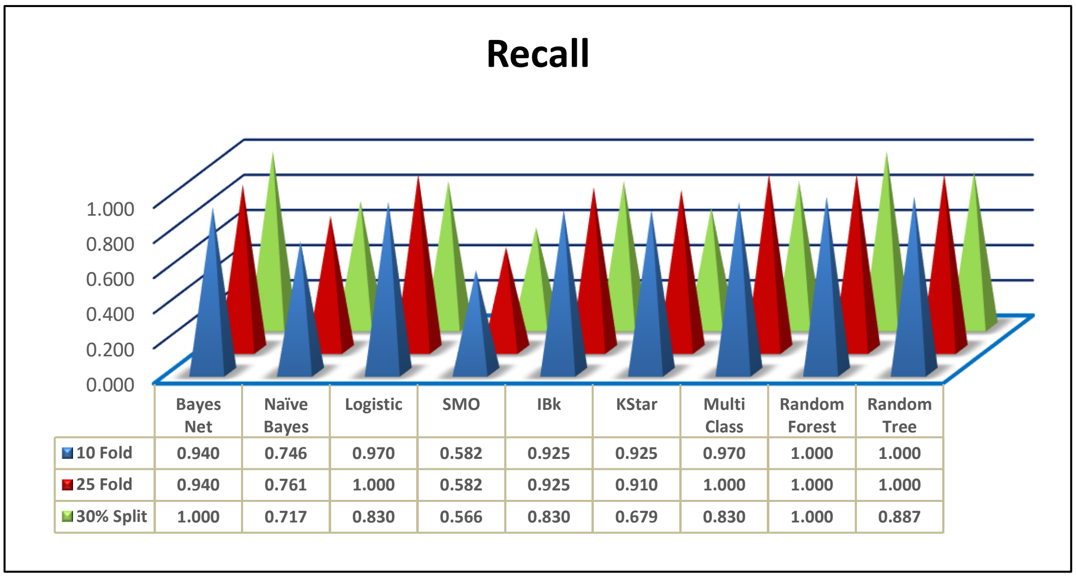
Figure 5.79: Precision (Cross-Validation: 10-Fold, 25-Fold and 30% Split)



	Bayes Net	Naïve Bayes	Logistic	SMO	IBk	KStar	Multi Class	Random Forest	Random Tree
🟦 10 Fold	1.000	0.926	0.985	1.000	0.886	0.899	0.985	1.000	1.000
🟥 25 Fold	1.000	0.911	0.985	1.000	0.899	0.897	0.985	1.000	1.000
🟩 30% Split	1.000	0.950	1.000	1.000	0.957	1.000	1.000	1.000	1.000

From above three-Dimensional plot it is clear that the maximum average Precision score for Bayes Net, SMO, Random Forest and Random Tree is 1.0, therefore it is concluded that Bayes Net, SMO, Random Forest and Random Tree are the best algorithms for getting best Precision using Cross – Validation of 10-Fold, 25-Fold and 30% Split in Heavy Traffic Conditions.

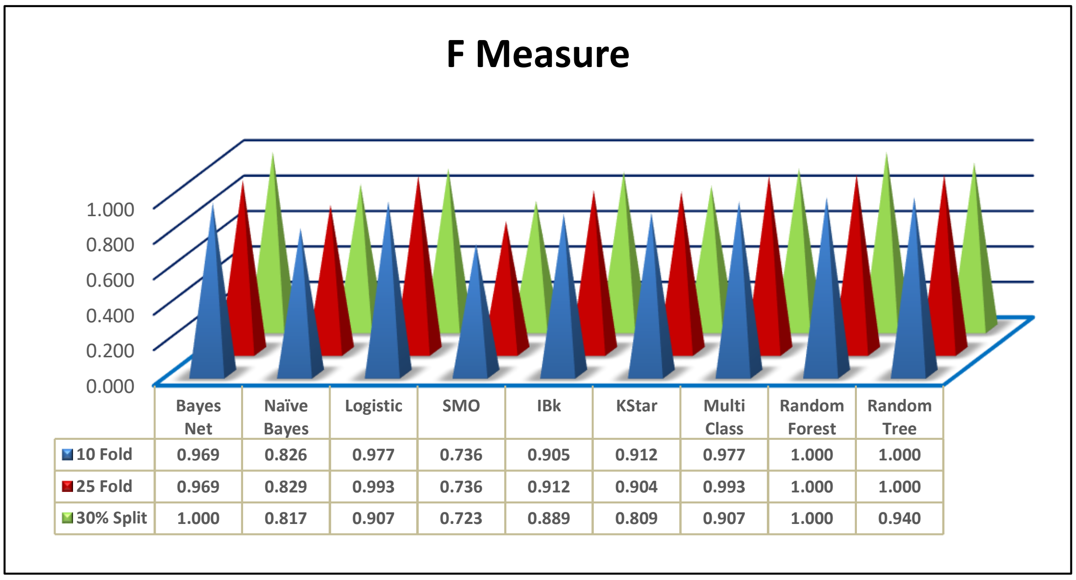
Figure 5.80: Recall (Cross-Validation: 10-Fold, 25-Fold and 30% Split)



	Bayes Net	Naïve Bayes	Logistic	SMO	IBk	KStar	Multi Class	Random Forest	Random Tree
🟦 10 Fold	0.940	0.746	0.970	0.582	0.925	0.925	0.970	1.000	1.000
🟥 25 Fold	0.940	0.761	1.000	0.582	0.925	0.910	1.000	1.000	1.000
🟩 30% Split	1.000	0.717	0.830	0.566	0.830	0.679	0.830	1.000	0.887

From above three-Dimensional plot the maximum average Recall score for Random Forest is 1.0, therefore it is concluded that Random Forest is the best algorithm for getting best Recall using Cross – Validation of 10-Fold, 25-Fold and 30% Split in Heavy Traffic Conditions.

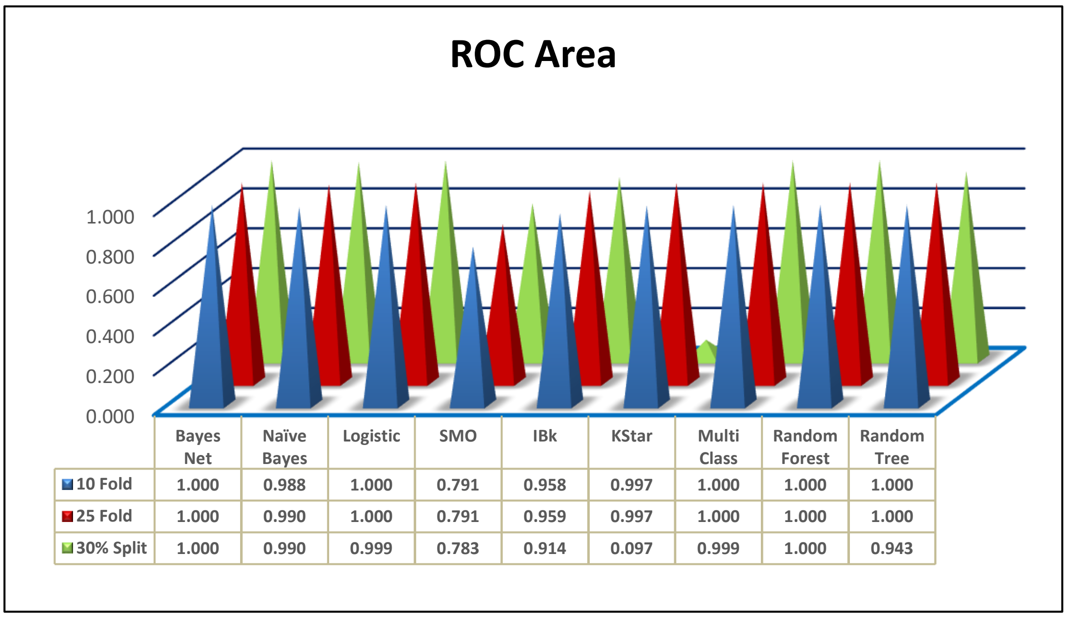
Figure 5.81: F Measure (Cross-Validation: 10-Fold, 25-Fold and 30% Split)



	Bayes Net	Naïve Bayes	Logistic	SMO	IBk	KStar	Multi Class	Random Forest	Random Tree
🟦 10 Fold	0.969	0.826	0.977	0.736	0.905	0.912	0.977	1.000	1.000
🟥 25 Fold	0.969	0.829	0.993	0.736	0.912	0.904	0.993	1.000	1.000
🟩 30% Split	1.000	0.817	0.907	0.723	0.889	0.809	0.907	1.000	0.940

From above three Dimensional plot it is clear that the maximum average F Measure score for Random Forest is 1.0, therefore it is concluded that Random Forest is the best algorithms for getting best F Measure using Cross – Validation of 10-Fold, 25-Fold and 30% Split in Heavy Traffic Conditions.

Figure 5.82: ROC Area (Cross-Validation: 10-Fold, 25-Fold and 30% Split)

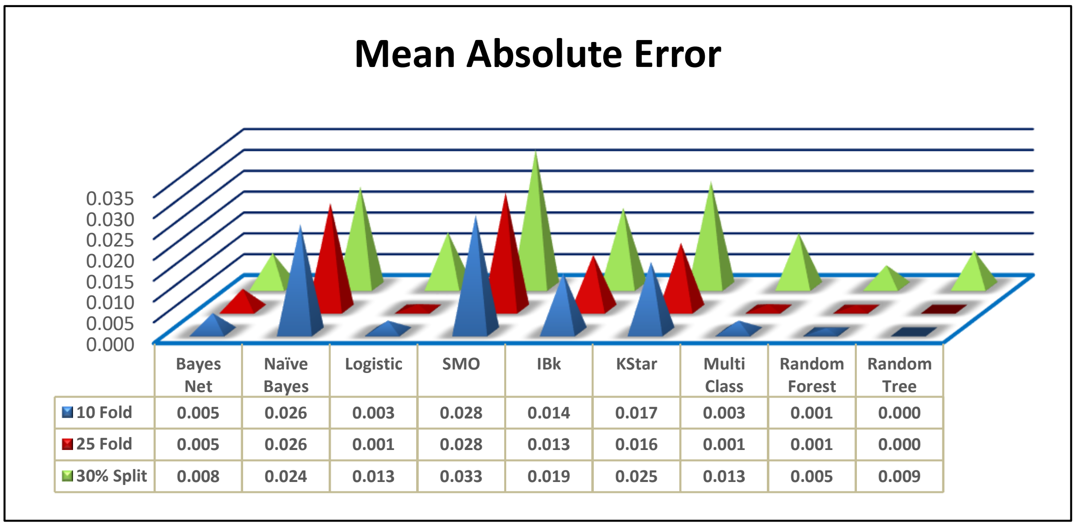


	Bayes Net	Naïve Bayes	Logistic	SMO	IBk	KStar	Multi Class	Random Forest	Random Tree
🟦 10 Fold	1.000	0.988	1.000	0.791	0.958	0.997	1.000	1.000	1.000
🟥 25 Fold	1.000	0.990	1.000	0.791	0.959	0.997	1.000	1.000	1.000
🟩 30% Split	1.000	0.990	0.999	0.783	0.914	0.097	0.999	1.000	0.943

From above three Dimensional plot it is clear that the maximum average ROC Area score for Bayes Net and Random Forest is 1.0 therefore it is concluded that Bayes Net and Random Forest are the best algorithms for getting best ROC Area using Cross – Validation of 10-Fold, 25-Fold and 30% Split in Heavy Traffic Conditions.

B. Error Measure

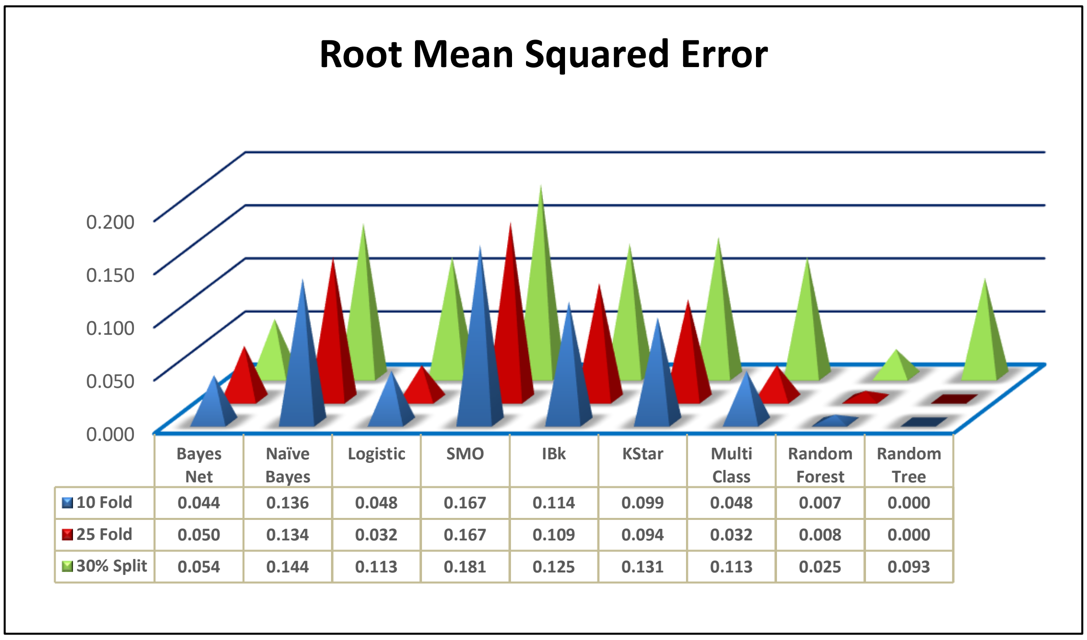
Figure 5.83: Mean Absolute Error (Cross-Validation: 10-Fold, 25-Fold and 30% Split)



	Bayes Net	Naïve Bayes	Logistic	SMO	IBk	KStar	Multi Class	Random Forest	Random Tree
🟦 10 Fold	0.005	0.026	0.003	0.028	0.014	0.017	0.003	0.001	0.000
🟥 25 Fold	0.005	0.026	0.001	0.028	0.013	0.016	0.001	0.001	0.000
🟩 30% Split	0.008	0.024	0.013	0.033	0.019	0.025	0.013	0.005	0.009

From above three Dimensional plot it is clear that the average Mean Absolute Error score for Random Forest is minimum and it’s value is 0.002 therefore it is concluded that Random Forest is the best algorithms for getting Minimum Mean Absolute error using Cross – Validation of 10-Fold, 25-Fold and 30% Split.

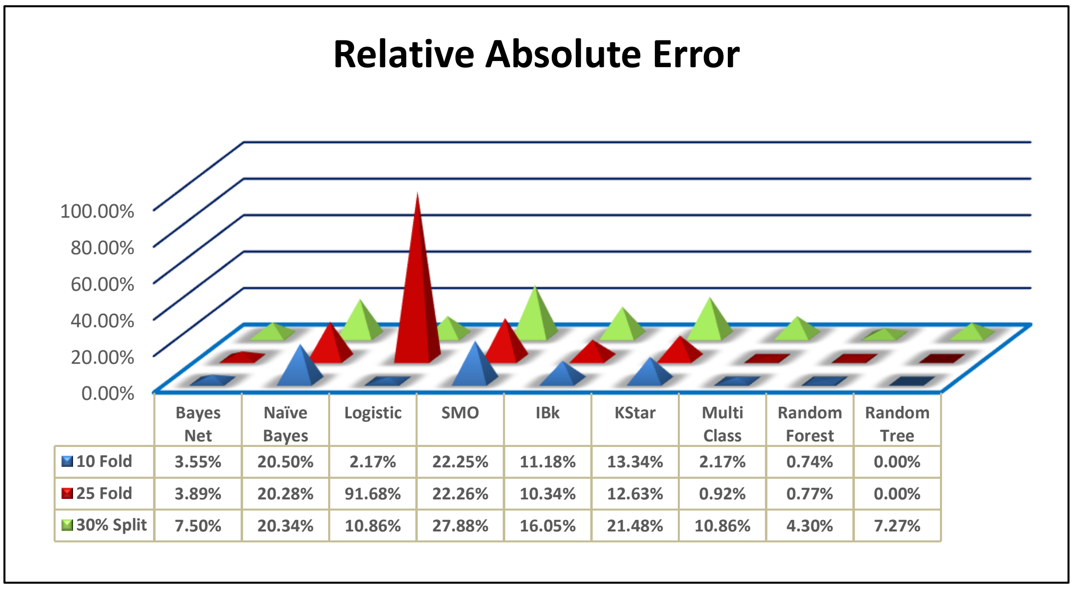
Figure 5.84: Root Mean Squared Error (10-Fold, 25-Fold and 30% Split)



	Bayes Net	Naïve Bayes	Logistic	SMO	IBk	KStar	Multi Class	Random Forest	Random Tree
🟦 10 Fold	0.044	0.136	0.048	0.167	0.114	0.099	0.048	0.007	0.000
🟥 25 Fold	0.050	0.134	0.032	0.167	0.109	0.094	0.032	0.008	0.000
🟩 30% Split	0.054	0.144	0.113	0.181	0.125	0.131	0.113	0.025	0.093

From above three Dimensional plot it is clear that the average Root Mean Squared Error score for Random Forest is minimum and it’s value is 0.013, therefore it is concluded that Random Forest is the best algorithms for getting Minimum Root Mean Squared Error using Cross – Validation of 10-Fold, 25-Fold and 30% Split.

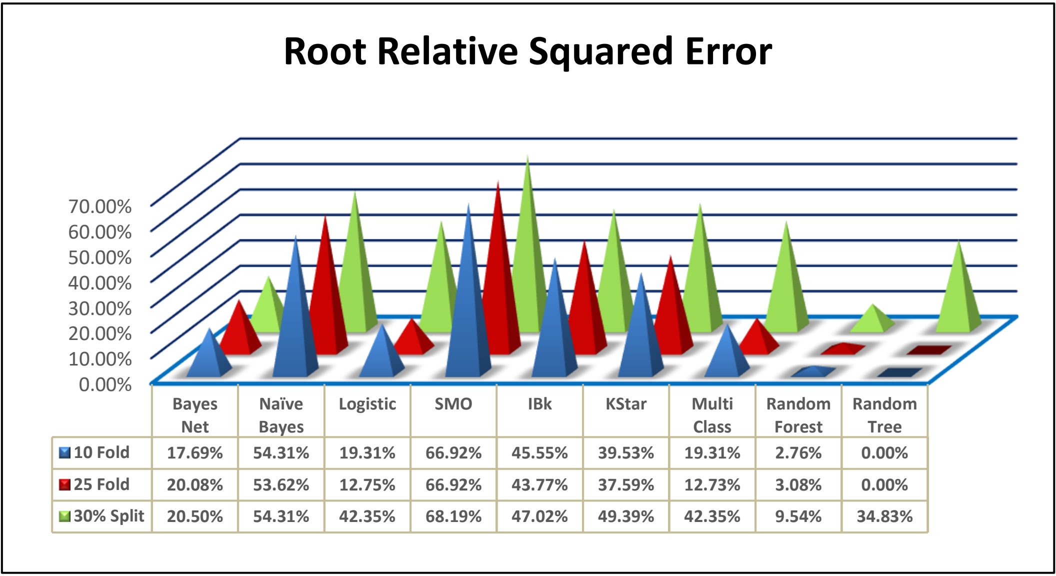
Figure 5.85: Relative Absolute Error(Cross-Validation:10-Fold,25-Fold & 30% Split)



	Bayes Net	Naïve Bayes	Logistic	SMO	IBk	KStar	Multi Class	Random Forest	Random Tree
🟦 10 Fold	3.55%	20.50%	2.17%	22.25%	11.18%	13.34%	2.17%	0.74%	0.00%
🟥 25 Fold	3.89%	20.28%	91.68%	22.26%	10.34%	12.63%	0.92%	0.77%	0.00%
🟩 30% Split	7.50%	20.34%	10.86%	27.88%	16.05%	21.48%	10.86%	4.30%	7.27%

From above three Dimensional plot it is clear that the average Relative Absolute Error score for Random Forest is minimum and it’s value is 1.94%, therefore it is concluded that Random Forest is the best algorithms for getting Minimum Relative Absolute Error using Cross – Validation of 10-Fold, 25-Fold and 30% Split.

Figure 5.86: Root Relative Squared Error (10-Fold, 25-Fold and 30% Split)



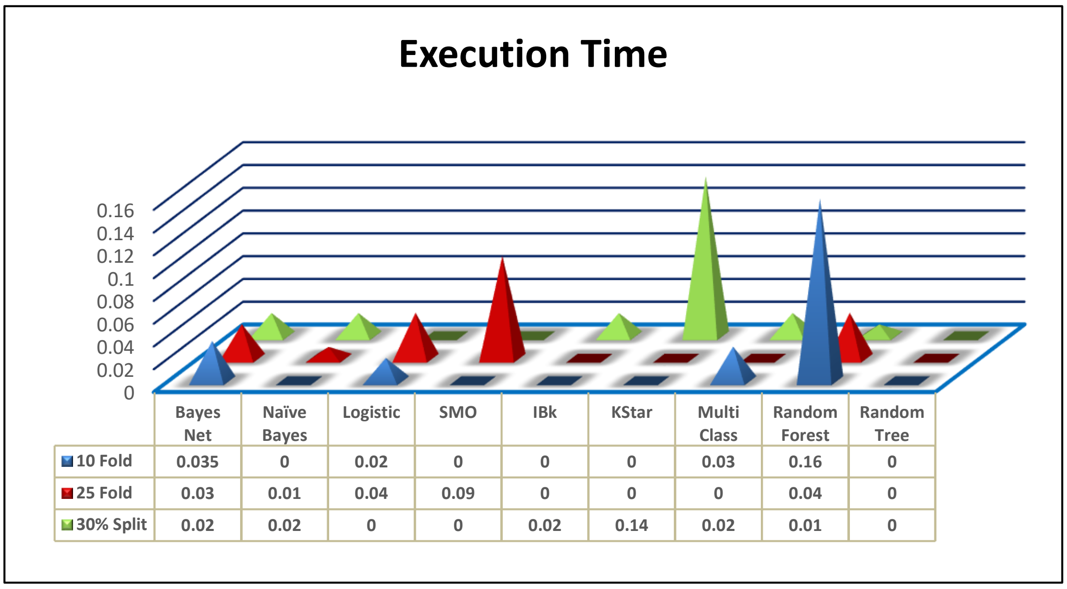
	Bayes Net	Naïve Bayes	Logistic	SMO	IBk	KStar	Multi Class	Random Forest	Random Tree
🟦 10 Fold	17.69%	54.31%	19.31%	66.92%	45.55%	39.53%	19.31%	2.76%	0.00%
🟥 25 Fold	20.08%	53.62%	12.75%	66.92%	43.77%	37.59%	12.73%	3.08%	0.00%
🟩 30% Split	20.50%	54.31%	42.35%	68.19%	47.02%	49.39%	42.35%	9.54%	34.83%

From above three Dimensional plot it is clear that the average Root Relative Squared Error score for Random Forest is minimum and it’s value is 5.13%, therefore it is concluded that Random Forest is the best algorithms for getting Minimum Root Relative Squared Error using Cross – Validation of 10-Fold, 25-Fold and 30% Split.

C. Execution Time

Execution Time is one of the important parameter which decides the speed at which algorithm can evaluate and test data sets. It is very important parameter specially when data sets are too large, which is the prime requirement of any machine learning algorithm. If the machine learning algorithm is slow and efficient in all other aspects then cloud computing is the remedy for that. Multitasking in Cloud computing is the solution for slow Machine learning algorithms. The figure 5.87 shown below compares the execution time in seconds of Nine machine learning algorithms.

Figure 5.87: Execution Time (Cross-Validation: 10-Fold, 25-Fold and 30% Split)



	Bayes Net	Naïve Bayes	Logistic	SMO	IBk	KStar	Multi Class	Random Forest	Random Tree
🟦 10 Fold	0.035	0	0.02	0	0	0	0.03	0.16	0
🟥 25 Fold	0.03	0.01	0.04	0.09	0	0	0	0.04	0
🟩 30% Split	0.02	0.02	0	0	0.02	0.14	0.02	0.01	0

The above figure indicates that Random forest Algorithm Average execution time is maximum with average value of 0.07seconds. Random forest algorithm excels in all other parameters except execution time. This drawbacks can be nullified using cloud computing and parallel computing. The Random Tree is the most efficient and fast algorithm with average execution time in nano seconds or nearly zero.

5.3.7 Dominance Chart

Table 5.16 shows the Machine learning Algorithm Dominance chart For 10-fold, 25-fold and 30% split Performance measures. This chart indicates the highest average marks obtained by the machine learning algorithms. One mark is allotted to the machine learning algorithm which got highest score in three categories 10-fold, 25-fold and 30% split. Dominance chart includes twenty parameters from Performance measure, Confusion matrix (Low Traffic and Heavy Traffic), Error measures and Execution time in seconds for finding out Total score. The selected algorithm should have highest marks out of twenty to be the best Machine Learning Algorithm.

Table 5.16 : Machine Learning Algorithms Dominance Chart

Dominance Chart (Cross -Validation 10 Fold, 25 Fold and 30% Split)																					
Classifiers	Performance Measure			Confusion Matrix Parameters Low Traffic						Confusion Matrix Parameters Heavy Traffic						Error Measures				Execution Time	Total Score
	Accuracy	Incorrectly Classified Instances	Kappa statistic	TP Rate	FP Rate	Precision	Recall	F-Measure	ROC Area	TP Rate	FP Rate	Precision	Recall	F-Measure	ROC Area	Mean absolute error	Root mean squared error	Relative absolute error	Root relative squared error		
Bayes Net				**1**			**1**		**1**		**1**	**1**			**1**						**6**
Naïve Bayes																					
Logistic																					
SMO				**1**							**1**	**1**									**3**
IBK																					
K Star																					
Multi Class																					
Random Forest	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**1**		**19**
Random Tree				**1**	**1**	**1**					**1**	**1**								**1**	**6**

It is clear from the Table 5.16 that the highest score is obtained by Random Forest algorithm with total score of 19, followed by Random Tree and Bayes Net with six marks each. Thus considering Performance measures, Confusion matrix parameters, Error Measure and Execution Time, it is concluded that the Random Forest is the best Algorithm for forecasting the traffic flow conditions in smart city.

5.3.8 Weighted Sum Model Analysis Using Python

The Weighted Sum Model is a decision-making approach used to evaluate and rank a set of alternatives based on multiple criteria. It involves assigning weights to each criterion and then calculating a weighted sum of the normalized values of each criterion for each alternative. The alternative with the highest weighted sum is considered the best choice. The procedure for Weighted Sum Model is listed below.

1. **Classifier and Metric Definitions:** Begin by defining the list of classifiers or alternatives you want to evaluate. Each classifier is associated with a set of metrics (criteria) that measure its performance.
2. **Metrics and Weights:** Assign weights to each metric (criterion) based on its importance. These weights reflect the relative significance of each criterion in the decision-making process. The sum of weights should add up to 1 or 100% to ensure a meaningful comparison.
3. **Normalization:** Normalize the metric values for each classifier to a common scale, often within the range of 0 to 1. This step ensures that metrics with different units and scales can be compared effectively.
4. **Weighted Sum Calculation:** For each classifier, calculate the weighted sum of its normalized metric values. This is done by multiplying each normalized metric value by its corresponding weight, and then summing up these weighted values.
5. **Best Alternative:** Identify the alternative with the highest calculated weighted sum. This alternative is considered the best choice according to the chosen criteria and their assigned weights.
6. **Decision:** The alternative with the highest weighted sum is selected as the best choice based on the defined criteria and their weights.

The Weighted Sum Model is widely used in decision analysis when there are multiple factors to consider, and each factor carries a different level of importance. It's important to note that the success of the model heavily depends on the accuracy of the assigned weights and the relevance of the chosen metrics.

Python Pseudocode:

```
// Define classifier names
classifiers = ["Classifier 1", "Classifier 2", ...]

// Provided metrics (Replace with your actual metrics)
metrics = [
    [metric_value_1_classifier_1, metric_value_2_classifier_1, ...],
    [metric_value_1_classifier_2, metric_value_2_classifier_2, ...],
    ...
]

// Define weights for each metric (customize these weights)
metric_weights = [weight_metric_1, weight_metric_2, ...]

// Function to normalize metrics to [0, 1] range
function NormalizeMetrics(metrics):
    normalized_metrics = EmptyMatrix()
    for each classifier_metrics in metrics:
        normalized_classifier_metrics = Normalize(classifier_metrics)
        AddToMatrix(normalized_metrics, normalized_classifier_metrics)
    return normalized_metrics

// Function to calculate the weighted sum for each classifier
function CalculateWeightedSums(normalized_metrics, metric_weights):
    weighted_sums = EmptyList()
    for each classifier_metrics in normalized_metrics:
        weighted_sum = CalculateDotProduct(classifier_metrics, metric_weights)
        AppendToList(weighted_sums, weighted_sum)
    return weighted_sums
```

```
// Function to find the best alternative (index)
function FindBestAlternative(weighted_sums):
    best_alternative_index = IndexOfMax(weighted_sums)
    return best_alternative_index

// Function to print weighted sums
function PrintWeightedSums(classifiers, weighted_sums):
    for i from 0 to length(classifiers) - 1:
        Print(classifiers[i], " - Weighted Sum:", weighted_sums[i])

// Function to print the best alternative
function PrintBestAlternative(best_alternative):
    Print("The best alternative is:", best_alternative)

// Main program
function Main():
    normalized_metrics = NormalizeMetrics(metrics)
    weighted_sums = CalculateWeightedSums(normalized_metrics, metric_weights)
    best_alternative_index = FindBestAlternative(weighted_sums)
    best_alternative = classifiers[best_alternative_index]
    PrintWeightedSums(classifiers, weighted_sums)
    PrintBestAlternative(best_alternative)

// Call the main program
Main()
```

5.3.8.1 Multi-Criteria Decision Making - Weighted Sum Method

The Weighted Sum Method, a fundamental technique in Multi-Criteria Decision Making, facilitates decision-makers in evaluating and ranking alternatives by considering multiple criteria. This approach involves identifying relevant decision criteria, assigning weights to signify their importance, evaluating each alternative's performance on these criteria, normalizing scores to ensure comparability, and calculating a weighted sum for each alternative. The resulting scores enable a systematic ranking of alternatives, aiding decision-makers in selecting the most suitable option that aligns with their preferences and objectives. This method serves as a valuable decision support tool across various domains.

i. Evaluation with Cross Validation-10 Folds

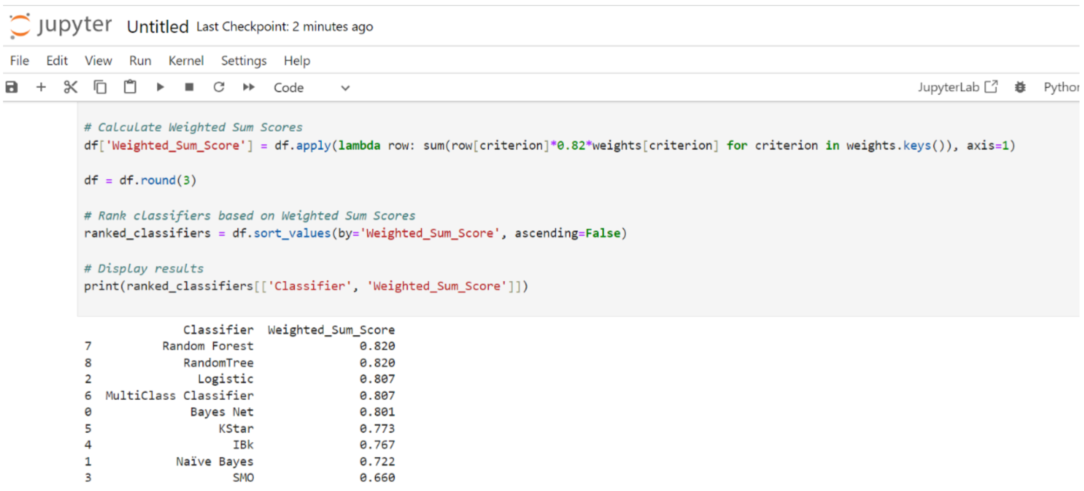
Table 5.17 presents a range of classification models assessed by their "Score (Weighted Sum)," with higher scores indicating superior performance.

Table 5.17: Weighted Sum Score (Cross Validation-10 Folds)

S.No.	Classification Models	Score (Weighted Sum)
1	Bayes Net	0.801
2	Naïve Bayes	0.722
3	Logistic	0.807
4	SMO	0.660
5	IBk	0.767
6	KStar	0.773
7	MultiClass Classifier	0.807
8	Random Forest	0.820
9	Random Tree	0.820

The weighed sum-based score suggests that Random Forest and Random Tree has the highest score of 0.820 followed by Logistic and Multiclass classifier with core 0.807 and Bayes Net with score 0.801. The lowest scores being obtained by the classifiers KStar, IBK, Naïve Bayes and SMO with values 0.773,0.767,0.722 and 0.660 respectively. Further Rank and Percentile analysis would be executed to obtain the final rankings.

Figure 5.88: Running MCDM Method in Python Environment


jupyter
Untitled Last Checkpoint: 2 minutes ago
File Edit View Run Kernel Settings Help
Code
JupyterLab
Python

```
# Calculate Weighted Sum Scores
df['Weighted_Sum_Score'] = df.apply(lambda row: sum(row[criterion]*0.82*weights[criterion] for criterion in weights.keys()), axis=1)

df = df.round(3)

# Rank classifiers based on Weighted Sum Scores
ranked_classifiers = df.sort_values(by='Weighted_Sum_Score', ascending=False)

# Display results
print(ranked_classifiers[['Classifier', 'Weighted_Sum_Score']])
```

```
             Classifier  Weighted_Sum_Score
7         Random Forest               0.820
8            RandomTree               0.820
2              Logistic               0.807
6  MultiClass Classifier               0.807
0             Bayes Net               0.801
5                 KStar               0.773
4                   IBk               0.767
1           Naïve Bayes               0.722
3                   SMO               0.660
```

5.3.9 Rank and Percentile Method:

Ranking and percentile methods can be applied to arrange classification algorithms in machine learning based on their performance metrics. Here's how you can use these methods:

Ranking Method Algorithm:

1. **Collect Performance Metrics**: Gather performance metrics (e.g., accuracy, precision, recall, F1 score, ROC AUC) for each classification algorithm. These metrics are typically computed using cross-validation or other evaluation techniques.
2. **Calculate Ranks**: Calculate ranks for each algorithm based on each performance metric. Assign a rank of 1 to the algorithm with the highest score for a metric, 2 to the second highest, and so on. In the case of ties, you can use methods like averaging ranks.
3. **Calculate Average Rank**: After ranking algorithms for each metric, calculate the average rank for each algorithm across all the metrics. This average rank represents the overall ranking for each algorithm.
4. **Sort and Present Results**: Sort the algorithms based on their average ranks in ascending order. The algorithm with the lowest average rank is considered the top performer, while the one with the highest average rank is considered the lowest performer. You can present these rankings in a table or report.

Percentile Method Algorithm:

1. **Collect Performance Metrics**: Similar to the ranking method, gather performance metrics for each classification algorithm.
2. **Calculate Percentiles**: For each performance metric, calculate the percentile rank of each algorithm. Percentile rank indicates the percentage of algorithms that performed worse than a particular algorithm for a given metric.
3. **Calculate Average Percentile Rank**: After calculating percentiles for each metric, compute the average percentile rank for each algorithm across all the metrics. This average percentile rank represents the overall ranking for each algorithm.

4. **Sort and Present Results**: Sort the algorithms based on their average percentile ranks in ascending order. Lower average percentile rank indicates better performance across multiple metrics.

Python Pseudocode : Rank_Classification_Algorithm

```
Input:
- List of classification algorithms (Algorithms)
- List of performance metrics (Metrics)
- Dictionary of algorithm performance data (AlgorithmMetrics)

Output:
- Sorted list of algorithms based on average rank (RankedAlgorithms)

Begin:
   // Initialize an empty dictionary to store ranks for each algorithm
   Initialize an empty dictionary AlgorithmRanks

   // Step 1: Calculate ranks for each algorithm and metric
   For each Algorithm in Algorithms:
      // Initialize a list to store ranks for each metric
      Initialize an empty list MetricRanks

      For each Metric in Metrics:
         // Calculate the rank for the Algorithm based on the Metric
         Rank = CalculateRank(AlgorithmMetrics[Algorithm][Metric])

         // Append the rank to the MetricRanks list
         Append Rank to MetricRanks
      End For

      // Calculate the average rank for the Algorithm
      AverageRank = CalculateAverageRank(MetricRanks)

      // Store the average rank in the AlgorithmRanks dictionary
      AlgorithmRanks[Algorithm] = AverageRank
   End For

   // Step 2: Sort algorithms based on average rank
   SortedAlgorithms = SortAlgorithmsByRank(AlgorithmRanks)

   // Step 3: Output the sorted list of algorithms
   Return SortedAlgorithms

End
```

```
Function: CalculateRank

Input:
- List of performance scores (Scores)

Output:
- Rank for the algorithm based on the scores (Rank)

Begin:
   // Sort the scores in descending order
   SortedScores = SortScoresDescending(Scores)

   // Initialize the rank as 1
   Rank = 1

   For each Score in SortedScores:
      // Assign the current rank to the Score
      Set Rank for Score = Rank

      // Increment the rank for the next Score if it has the same value
      If NextScoreExists() AND NextScore() = Score Then
         Increment Rank
      End If
   End For

   Return Rank
End

Function: CalculateAverageRank

Input:
- List of ranks (Ranks)

Output:
- Average rank (AverageRank)

Begin:
   // Calculate the mean (average) of the ranks
   AverageRank = Mean(Ranks)

   Return AverageRank
End

Function: SortAlgorithmsByRank

Input:
- Dictionary of algorithm ranks (AlgorithmRanks)

Output:
- Sorted list of algorithms based on rank (SortedAlgorithms)
```

```
Begin:
    // Sort the algorithms based on their average ranks
    SortedAlgorithms = Sort(Algorithms, AlgorithmRanks[Algorithm])

    Return SortedAlgorithms
End
```

The outcomes shows the ranking given to various classification algorithm (applied at cross validation- 10 folds) using the rank and percentile method. The results in the table below shows the classification model (evaluated at cross validation 10-folds), point (classifier ID), Rank and Percentage.

Table 5.18: Classification Models and Ranks (Cross Validation-10 Folds)

Classification Models	Point (Classifier ID)	Score (Weighted Sum)	Rank	Percentile
Random Forest	8	0.820	1	100.00%
Random Tree	9	0.820	1	100.00%
Logistic	3	0.807	3	62.50%
MultiClass Classifier	7	0.807	3	62.50%
Bayes Net	1	0.801	5	50.00%
KStar	6	0.773	6	37.50%
IBK	5	0.767	7	25.00%
Naïve Bayes	2	0.722	8	12.50%
SMO	4	0.660	9	0%

It was found that based on configuration setting: cross validation – 10 folds Random Forest and Random Tree are the best and most appropriate classifier for traffic congestion control and traffic flow as both are having the highest score of 0.820 with percentile 100%. Logistic and MultiClass Classifier are the second most appropriate algorithms having total score of 0.807, rank 3 and percentile of 62.50%. The third best classifier being identified is Bayes net with a total score of 0.801 and percentile of 50.00%. For predicting the Udaipur traffic flow Random Forest and Random Tree are the most appropriate algorithms.

ii. Evaluation with Cross Validation-25 Folds

Table 5.19 shows the ranking given to various classification algorithms (applied at cross validation- 25 folds) using the rank and percentile method.

Table 5.19: Weighted Sum Score (Cross Validation-25 Folds)

S. No.	Classification Models	Score (Weighted Sum)
1	Bayes Net	0.801
2	Naïve Bayes	0.724
3	Logistic	0.815
4	SMO	0.660
5	IBk	0.770
6	KStar	0.769
7	MultiClass Classifier	0.815
8	Random Forest	0.820
9	RandomTree	0.820

The weighed sum-based score suggests that Random Forest and Random Tree has the highest score of 0.820 followed by Logistic and MultiClass Classifier with score 0.815. The results in Table 5.20 show the classification model (evaluated at cross validation 25-folds), point (classifier ID), Weighted Sum, Rank and Percentage.

Table 5.20: Classification Models and Ranks (Cross Validation-25 Folds)

Classification Models	Point (Classifier ID)	Score (Weighted Sum)	Rank	Percent
Random Forest	8	0.820	1	100.00%
Random Tree	9	0.820	1	100.00%
Logistic	3	0.815	3	62.50%
MultiClass Classifier	7	0.815	3	62.50%
Bayes Net	1	0.801	5	50.00%
IBK	5	0.770	6	37.50%
KStar	6	0.769	7	25.00%
Naïve Bayes	2	0.724	8	12.50%
SMO	4	0.660	9	0.00%

It was found that based on configuration setting: cross validation – 25 folds Random Forest and Random Tree are the best and most appropriate classifier for traffic congestion control and traffic flow as it has the highest score of 0.820 with percentile 100%. Logistic and MultiClass Classifier are the second most appropriate algorithm having a total score of 0.815, rank 3 and percentile of 62.50%. The third best classifier being identified is Bayes Net with a total score of 0.801 and percentile of 50.00%. For predicting the Udaipur traffic flow Random Forest is the most appropriate algorithm.

iii. Evaluation with Cross Validation-Split: 30%

Table 5.21 shows the ranking given to various classification algorithms (applied at cross validation- 30% Split) using the rank and percentile method. The weighed sum-based score suggests that Random forest has the highest score of 0.820 followed by Random Tree with score 0.779 and MultiClass Classifier with score 0.765.

Table 5.21: Weighted Sum Score (Cross Validation-30% Split)

S. No.	Classification Models	Score (Weighted Sum)
1	Bayes Net	0.658
2	Naïve Bayes	0.585
3	Logistic	0.619
4	SMO	0.652
5	IBk	0.749
6	KStar	0.638
7	MultiClass Classifier	0.765
8	Random Forest	0.820
9	RandomTree	0.779

The result in the table below shows the classification model ,point (classifier ID), Rank and Percentage evaluated at cross validation 30%-Split.

Table 5.22: Classification Models and Ranks (Cross Validation-30% Split)

Classification Models	Point (Classifier ID)	Score (Weighted Sum)	Rank	Percent
Random Forest	8	0.820	1	100.00%
Random Tree	9	0.779	2	87.50%
Multi Class Classifier	7	0.765	3	75.00%
IBK	5	0.749	4	62.50%
Bayes Net	1	0.658	5	50.00%
SMO	4	0.652	6	37.50%
KStar	6	0.638	7	25.00%
Logistic	3	0.619	8	12.50%
Naïve Bayes	2	0.585	9	0.00%

It was found that based on configuration setting: cross validation – 30% Split Random forest is the best and most appropriate classifier for traffic congestion control and traffic flow as it has the highest score of 0.820 with percentile 100%. Random Tree is the second most appropriate algorithm having a total score of 0.779, rank 2 and percentile of 87.50%. The third best classifier being identified is Multi Class Classifier with a total score of 0.765 and percentile of 75%. For predicting the Udaipur traffic flow Random forest is the most appropriate algorithm while considering configuration setting cross validation – 30%.

Finally, a Random Forest-based predictive model's high ranking in a cross-validation setting, whether it's 10-fold, 25-fold or 30% split indicates its robustness and effectiveness in handling the complexities of traffic management. Its ability to capture non-linear patterns, handle real-time data, and provide insights into important features makes it a valuable tool for improving traffic flow, reducing congestion, and enhancing overall transportation efficiency.

5.4 Hypothesis Testing Results

Hypothesis is nothing but a tentative statement or proposed explanation made based on limited evidence as a starting point for further investigation. The following two hypothesis are being tested for proposed research work.

Hypothesis 1:

The main objective of my research is to solve commuting problems in smart cities using Artificial Intelligence, IoT and Machine Learning technologies. To reach meaningful conclusion of my research I am interested in finding whether there is significant difference between the type of smart technologies used in smart cities. To examine the difference between different categorial variables Chi-Square test is applied after doing survey from different age group participants from different cities. The Hypothesis statements are:

H_o1: There is no significant difference between technologies used for enhancing the transportation system in smart cities.

The related alternative hypothesis is as follows.

H_a1: **There is a significant difference between technologies used for enhancing the transportation system in smart cities.**

Test Applied: To test the hypothesis H_o1 the Chi-Square Test was being used. The outcomes of the Chi-Square test are shown below in the table.

Table 5.23: Type of Technology and Level of Enhancement

Type of Technology and Level of Enhancement in Smart Transportation System: Crosstabulation				
Count				
		Enhancement in Smart Transportation System		**Total**
		High	**Low**	
Type of Technology	AI Based	12	0	12
	Fog Computing	7	0	7
	IoT-Based Traffic Prediction Models	7	6	13
	Machine Learning-Based Traffic Prediction Models	14	4	18
Total		40	10	50

Table 5.24: Chi-Square Test Results

Chi-Square Test			
	Value	**df**	**Asymptotic Significance (2-sided)**
Pearson Chi-Square	10.363[a]	3	.016
Likelihood Ratio	13.026	3	.005
N of Valid Cases	50		
a. 4 cells (50.0%) have expected count less than 5. The minimum expected count is 1.40.			

Table 5.25: Calculation of Expected Frequency

Calculation of Expected Frequency			
Total of Technology	**Total of Enhancement in Smart Transportation System**	**Expected Frequency**	**EF**
12	40	12*40/50	9.6
7	40	7*40 /50	5.6
13	40	13*40 /50	10.4
18	40	18*40 /50	14.4
12	10	12*10 /50	2.4
7	10	7*10 /50	1.4
13	10	13*10 /50	2.6
18	10	18*10 /50	3.6

Table 5.26: Observed and Expected Frequency calculations.

Observed and Expected Frequency for the calculation of X^2			
Observed Frequency (OF)	**Expected Frequency (EF)**	**$(OF - EF)^2$**	**$(OF - EF)^2 / EF$**
12	9.6	5.76	0.6
7	5.6	1.96	0.35
7	10.4	11.56	1.11
14	14.4	0.16	0.01
0	2.4	5.76	2.4
0	1.4	1.96	1.4
6	2.6	11.56	4.45
4	3.6	0.16	0.04
		Total (å)	10.36

Degree of Freedom =(r-1) (c-1)

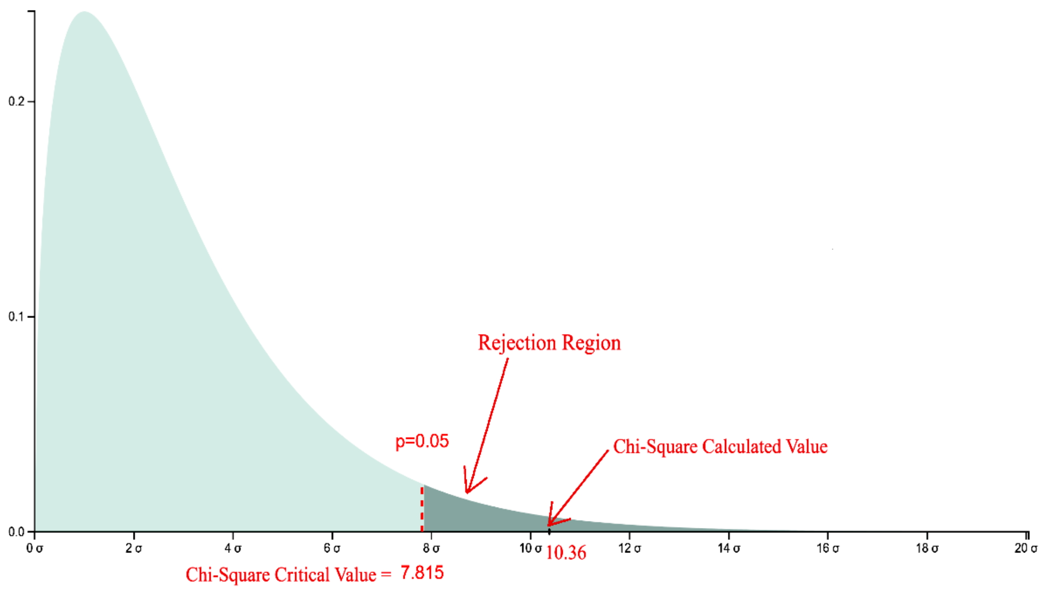
= (4-1) *(2-1) =3

Table value @5% level of significance = 7.815

Calculated Value of Chi-Square = 10.36

Result: The Chi-Square test results confirm that as the Pearson Chi-Square value was found to be 10.36 at degree of freedom 3 and the corresponding p-value is found to be 0.016 which is lesser than the standard alpha value of 0.05 this interpret that the null hypothesis $H_o1$is rejected and alternate hypothesis H_a1 is being accepted and it can be concluded that there is significant difference between technologies used for enhancing the transportation system for smart cities.

Figure 5.89: Right Tailed Chi-Square curve



Source: Chi-Square Distribution Calculator With Graph Generator [28]

From above figure Right Tailed Chi-Square curve it is clear that Calculated Chi-Square value lies in the rejection region therefore H_o1 is rejected and H_a1 is accepted. This indicates that there is a significant difference between the listed technologies like AI based, Fog Computing based, IoT based and Machine learning based traffic prediction models in solving traffic congestion problems in smart cities.

Hypothesis 2:

The weighted sum method algorithm for traffic prediction model generates different performance scores for 10-fold, 25-fold and 30% split case with different weights to each criterion. We are interested to know whether all machine learning algorithms average performance score is more than 75%. Following are the Hypothesis statements.

H_o2: The Machine learning-based traffic prediction models have average performance scores of greater than or equal to 75%.

The related alternative hypothesis is as follows.

H_a2: The Machine learning-based traffic prediction models have average performance scores of less than 75%.

$$\mathbf{H_o}: \mu \geq 0.75$$

$$\mathbf{H_a}: \mu < 0.75$$

Test Applied: As Sample mean is known and number of samples are less than 30, therefore One-Sample lower tail t-test is applied to test the hypothesis H_o2. Calculations of t-test are shown in the table shown below.

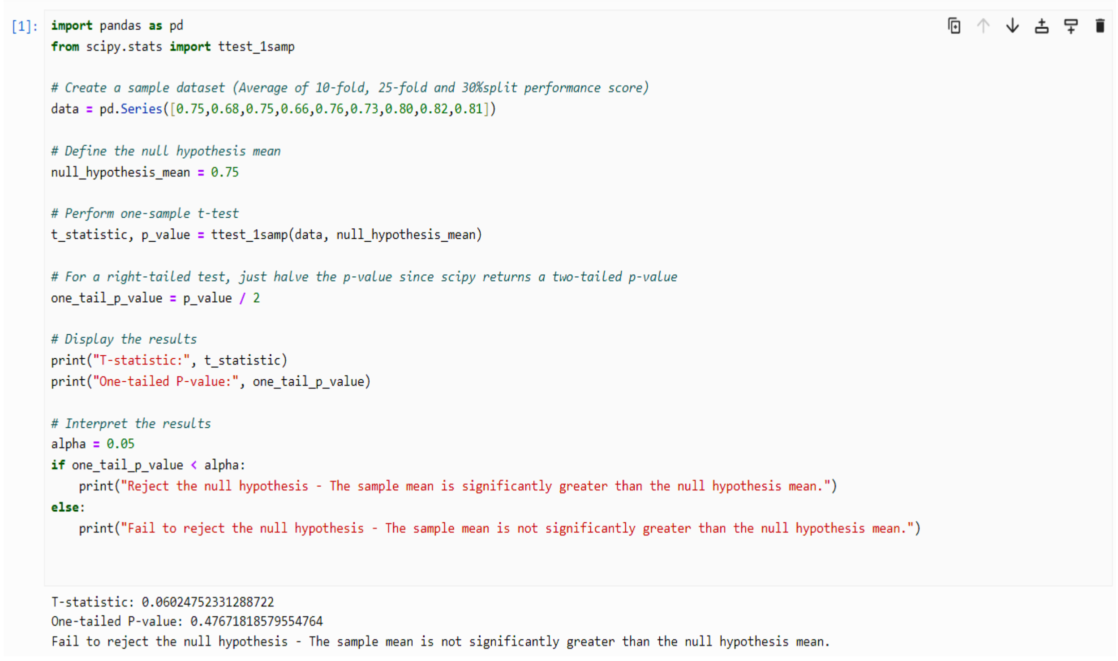
Table 5.27: T-test calculation

Classifier	WSM average Performance Score Values
Bayes Net	0.75
Naïve Bayes	0.68
Logistic	0.75
SMO	0.66
IBk	0.76
KStar	0.73
MultiClass Classifier	0.80
Random Forest	0.82
RandomTree	0.81
Sample Mean ($\bar{X}$)	0.751
Standard Deviation(S)	0.055
Number of Samples(n)	9
Claim (μ)	0.75
√n	3
S/√n	0.018

One Sample T-Test: $T = (\bar{X} - \mu) / S/\sqrt{n} = 0.060$

Degree of Freedom: 8

Figure 5.90: T-Test Python Program Output

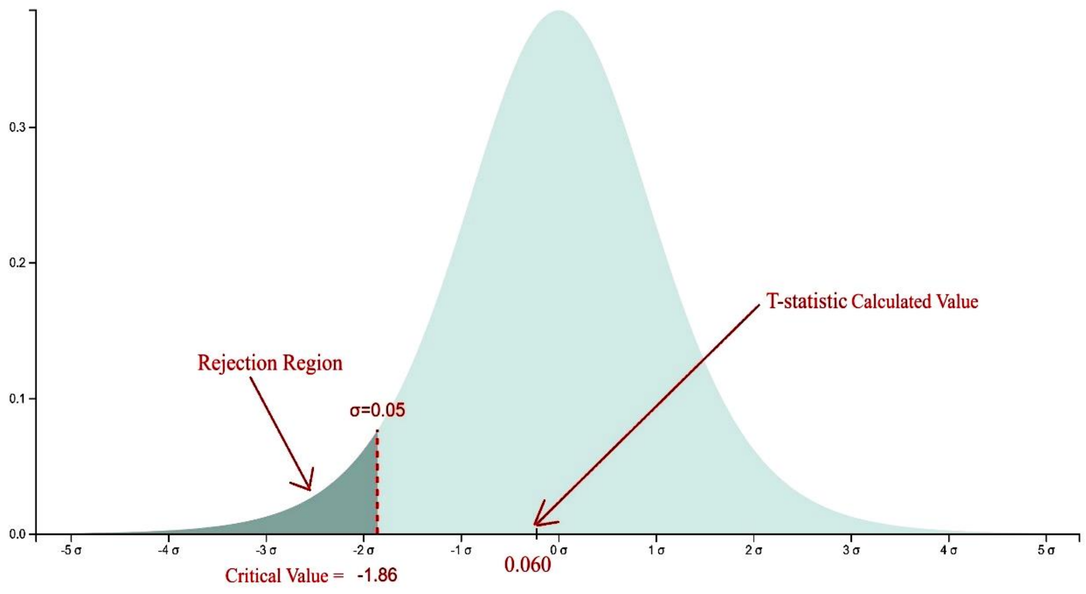


```
[1]: import pandas as pd
     from scipy.stats import ttest_1samp

     # Create a sample dataset (Average of 10-fold, 25-fold and 30%split performance score)
     data = pd.Series([0.75,0.68,0.75,0.66,0.76,0.73,0.80,0.82,0.81])

     # Define the null hypothesis mean
     null_hypothesis_mean = 0.75

     # Perform one-sample t-test
     t_statistic, p_value = ttest_1samp(data, null_hypothesis_mean)

     # For a right-tailed test, just halve the p-value since scipy returns a two-tailed p-value
     one_tail_p_value = p_value / 2

     # Display the results
     print("T-statistic:", t_statistic)
     print("One-tailed P-value:", one_tail_p_value)

     # Interpret the results
     alpha = 0.05
     if one_tail_p_value < alpha:
         print("Reject the null hypothesis - The sample mean is significantly greater than the null hypothesis mean.")
     else:
         print("Fail to reject the null hypothesis - The sample mean is not significantly greater than the null hypothesis mean.")
```

```
T-statistic: 0.06024752331288722
One-tailed P-value: 0.47671818579554764
Fail to reject the null hypothesis - The sample mean is not significantly greater than the null hypothesis mean.
```

Result: Table value @ 5% level of significance is -1.860 and the calculated T-statistics value is 0.060 which is larger than the critical T-value of -1.860, Also the p-value found to be 0.476 which is greater than the standard alpha value of 0.05 therefore null hypothesis is failed to be rejected. Therefore, we can say that the Machine learning-based traffic prediction models have average performance score more than or equal to 75%.

Figure 5.91: Left Tailed T-test curve



Source: T-test Distribution Calculator With Graph Generator [29]

From above figure Left Tailed T-test curve it is clear that calculated T-statistic value lies in the acceptance region therefore H_o2 is failed to reject and H_a2 is rejected. This indicates machine learning algorithms are useful in solving traffic congestion problems and generates average performance score of more than 75%.

5.5 Summary

In a comparative analysis of nine machine learning algorithms conducted utilizing the Weka tool, Random Forest and Random Tree emerged as the foremost viable classifiers for anticipating traffic congestion. Bayes Net moreover illustrated solid performance, ranking second respectively.

The investigation included assessing different performance parameters and error measures to evaluate the adequacy of each calculation. Among these measurements, Random Forest and Random Tree reliably beat other classifiers over multiple criteria. Furthermore, two hypotheses were tried during the analysis. The first hypothesis looked for to investigate the differences between Artificial Intelligence, Fog-based, IoT-based technologies, and Machine Learning approaches in traffic forecast models. This hypothesis was rejected and alternate hypothesis was accepted, recommending that there is significant differences between these technologies in terms of their effectiveness for traffic forecast. The second hypothesis aimed to determine if the average performance score of the classifiers surpassed 75%. This hypothesis failed to be rejected, demonstrating that the classifiers achieved satisfactory performance levels more than 75%.

Overall, the discoveries recommend that Random Forest and Random Tree classifiers are well-suited for traffic congestion forecast, with Bayes Net also offering strong performance. Furthermore, the analysis demonstrates that the choice between AI, Fog-based, IoT-based advances, and Machine Learning approaches may essentially affect the effectiveness of traffic prediction models.