Table of Contents

List of Figures

Abstract

The growing number of vehicles on road has attributed to increased traffic. Even though public transport systems like metro, buses, and trains help in reducing the vehicles on roads but there the problem of traffic congestion is still very prominent. The problem of traffic congestion affects everyone's life almost on daily basis. It is predicted that by 2040, the number of cars is going to be doubled. Therefore, it is important to understand and try to alleviate the problem. There are several machine learning algorithms used to enhance the accuracy of the prediction. The machine learning algorithm used are linear and logistic regression, K- nearest neighbors, Random Forest, SVM, Naïve Bayes, and Artificial neural network ANN. Different models produced different levels of efficiency and the best result was produced by an Artificial neural network. This machine-learning algorithm could fetch a precision of 95.1%

Introduction

Transportation is considered to be one of the most prominent factors in determining the infrastructure of the nation. Transportation affects everyone's life daily. It defines people's movement and the level of ease with which is done. On one hand, it could bring a certain convenience and on the other hand, if not managed properly, it could produce a lot of discomforts. The transportation system of any country plays an integral role in defining the social and economic status of the country. Since the dependency on the transport system is so high, it is important the system in place be reliable. The first step to achieving a high level of efficiency is the prediction of the traffic must be precise. This project aims to understand traffic congestion in different areas and the factors that play a role in it. With the help of data analysis, the dataset would be broken down into useful categories and develop a deeper understanding of the dataset.

1.1 Overview of Traffic Congestion

With the increasing number of vehicles on roads every day, congestion of roads is bound to happen which leads to the slower speed of vehicles and unpleasant driver's experience. This causes a loss of time and money. According to 2018 statistics the U.S incurred a loss of 305 billion dollars due to traffic congestion. There are various reasons for the failure of the "free-flow drive" experience. A few of the reasons that attribute to traffic are traffic incidents, work -zone, accidents, weather, and disruption in traffic control devices.

One of the major challenges in traffic congestion is precise prediction. With the growing use of advanced technology like real-time traffic maps, traffic forecasting has improved significantly. Intelligent transportation system aims to support different aspects like improving the user's experience by correctly predicting the traffic congestion, sustainable management system, and revamping the prediction by taking into account factors like accidents and changes in the route.

This case study consists of traffic congestion prediction based on real-time traffic obtained through Google Maps API connecting all main roads of Islamabad as well as from the roads connecting Rawalpindi and Islamabad.

1.2. Problem Identification

There could be several reasons that define the pattern of traffic congestion. For instance, there is generally a high rush during office hours and school hours. Weekdays and weekends also follow a distinct pattern of traffic. Few roads tend to be busier on weekdays than on weekends. There are a few reasons why the traffic pattern is unpredictable. For instance, prediction for accidents is highly unlikely, or sudden changes in weather or unforeseeable events. Situations like these create uncertainty and pose challenges to the accuracy of the estimation. Therefore, the analysis should incorporate the changes happening in real-time.

1.3.Objective

The objective of this case study is to augment the accuracy level of the prediction of traffic congestion. If the level and reason for the congestion can be accurately defined, traffic management would improve significantly. Better traffic management means enhanced accuracy of estimated time arrival and defining easier routes. Conditions on the road do not remain static, it keeps changing for several hours of the day. Therefore, it is important that the analysis and prediction also get tuned to the changes in traffic.

To achieve a high level of efficiency several machine learning algorithms are employed. Concepts like Random Forest, K-Nearest neighbors, SVM, Linear, and Logistic Regression, and Naïve Bayes to obtain high-level precision.

2. Literature Review

The studies and research on prediction system is not a recent challenge. There has been several and extensive research on traffic congestion ad how to improve the estimated time of arrival. Previous research (Wu, Ho & Lee, 2004) suggested that Support Vector Regression plays a crucial role in travel-time prediction and how SVR was more successful than other baseline models.

(Duraku & Ramadani, 2019) proposed principal component analysis (PCA) to detect key components and then use those key components to integrate and create PCA-MLR and PCA-RBF combined models to reduce the error in the prediction of travel time.

(Jiang, Abdel, Hu & Lee, 2015) used random forest algorithm to identify different hot zones which can underline safety hazardous zones and in turn render the safe design of road management.

(Huang, Xia, Li, Li &Li, 2019) took into consideration Long short-term memory (LSTM) neural network to predict traffic congestion during peak hours. The weather was also taken as a key component consideration. The state of the traffic was divided into five categories and then further analysis was conducted.

(Zafar & Haq , 2020) employed Google Maps API connecting all main roads of Islamabad as well as from the roads connecting Rawalpindi and Islamabad to obtain real-time data. Several machine learning algorithms were used like KNN, Naïve Bayes, and Random Forest to accurately predict the time of travel.

(Kan, Tang, Kwan, Ren, Liu & Li & ,2019) uses a unique method of analysis where sensing turn-level or lane-level traffic conditions are used to monitor congestion. Instead of using GPS trajectory data for further analysis, lane-level traffic is used, and analysis is performed via the clustering method.

3. Solution Methodology

3.1. Data Description

This dataset consists of information regarding traffic congestion based on real-time traffic obtained through Google Maps API connecting all main roads of Islamabad as well as from the roads connecting Rawalpindi and Islamabad covering a little over 120 sites. This data has been recorded over a few days to check for traffic congestion and consists of 317112 rows and 12 columns. To make the analysis most effective several factors have been kept in mind. For instance, special occasions like festivals or celebrations, and events also play a major role in determining the traffic control that day. Factors like fastest route name and time and weather have also been taken into account. To include these factors in the analysis data was fetched from Weather API and combined with data received from Google Maps API. Therfore the attributes that are finally part of the processing are discussed in Figure 1.

Features	**Description**
Day	From Monday to Sunday
Date	Records of the month of July 2019, November 2019, January 2020, and February 2020.
Destination on Location	The ending point of the journey
Fastest route distance	The shortest distance traveled in the journey

Fastest route time	Shortest time taken
Holidays	Yes, No
Special condition	Yes, No
Starting Location	The starting point of the journey
System Time	Duration of the journey
Weather	Showers, Cloudy, Mostly Cloudy, Sunny, Partly Cloudy, Mostly Sunny, Mostly Clear, Clear, Rain
Data prediction	Smooth, highly congested, slightly congested, congested, blockage

Figure 1: *Description of the dataset*

3.2. Data Cleaning and Visualization

Data cleaning is one of the most crucial steps in the analysis of the dataset. The first step is to make sure that are no null or 'NA' values. The dataset didn't have any null or 'NA' values. The next step is data manipulation. For instance, the 'Day' column consists of string values that are converted into numeric values such as Monday is converted into 1, Tuesday as 2, Wednesday as 3, Thursday as 4, Friday as 5, Saturday as 6, and Sunday as 7.

The 'Holiday' column contains yes, and no values, and this column is also converted into numeric values. 'Yes' is termed as 1 and 'No' is termed as 0. The 'Weather' column has string values that

are converted to numeric values such as Showers as 1, Cloudy as 2, Mostly Cloudy as 3, Sunny as 4, Partly Cloudy as 5, Mostly Sunny as 6, Mostly Clear as 7, Clear as 8, Rain as 9.

Traffic congestion is classified as

(0, 0.15) Smooth ~ 0

(0.15, 0.35) Slightly Congested ~ 1

(0.35, 0.65) Congested ~ 2

(0.65, 2.0) Highly Congested ~ 3

Above 2.0 Blockage ~ 4

The data visualization of the congestion along with the count is shown in Figure 2. As per the figure, the experience of most of the rides is either smooth or they experience high congestion.

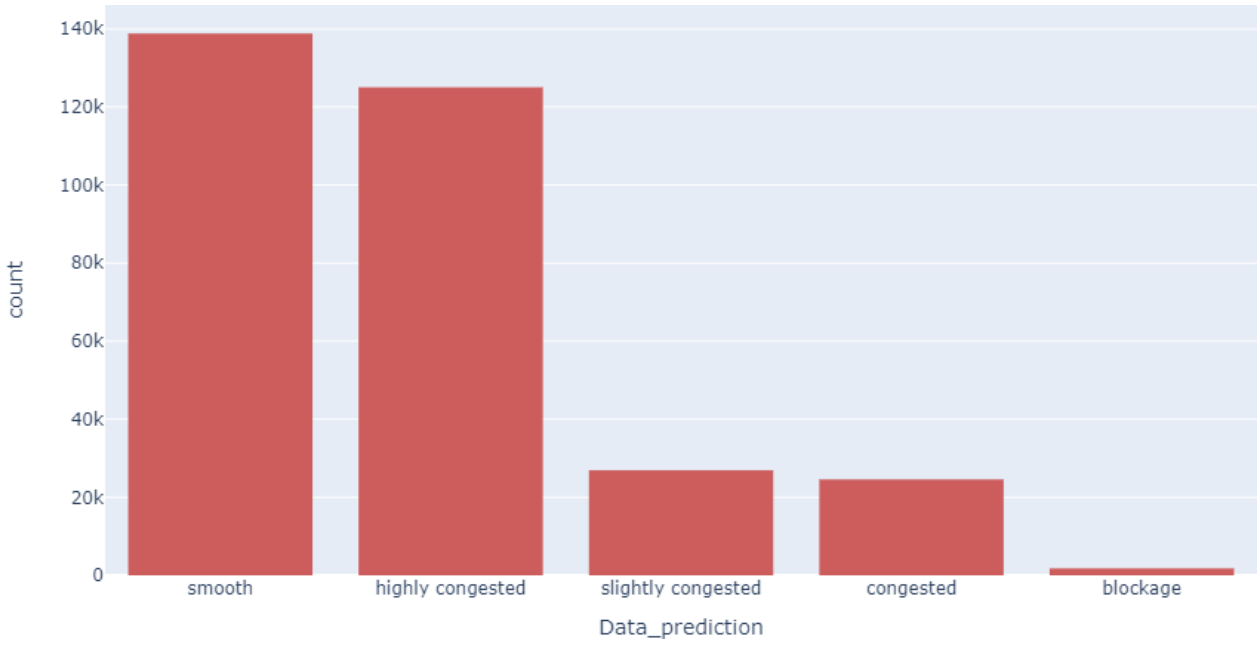


Figure2*: Level of Congestion based on count.*

The relation between fastest route time and fastest route time is shown in Figure 3. Higher the route time is directly propionate to congestion which essentially means that the higher the route time higher is congestion and the lower the route time lower is the congestion.

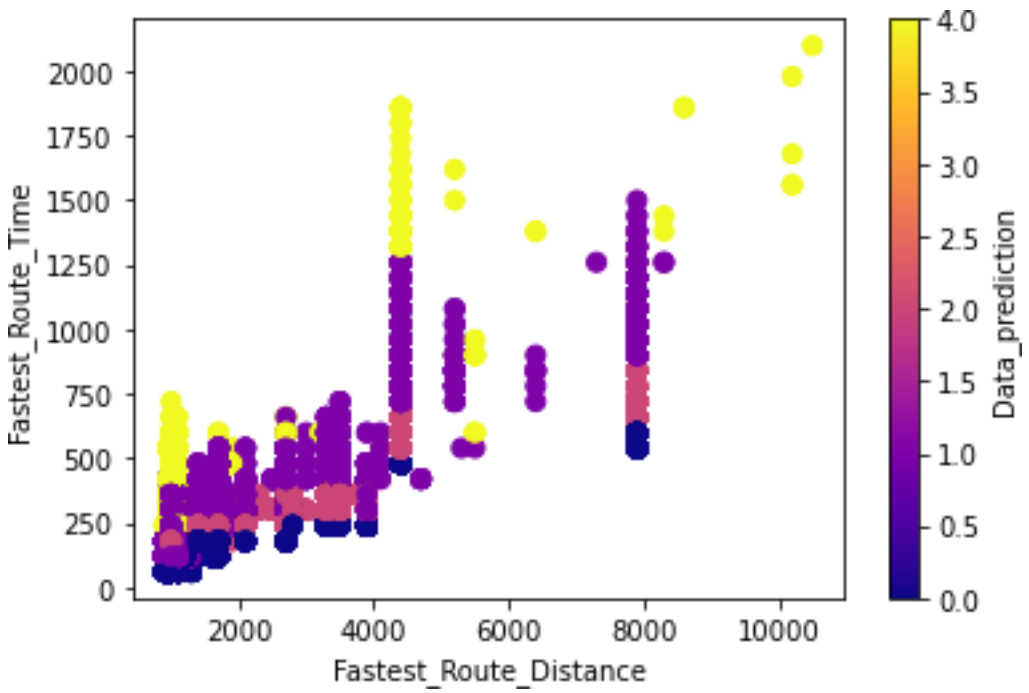


Figure 3*: The relationship between fastest route time and fastest route distance based on congestion.*

The data visualization for the congestion on the different weeks of the days is shown in Figure 4. Few interesting facts are observed in data exploration. On close observation, it is evident that traffic is never smooth on Wednesday while Monday and Sunday have the least amount of traffic.

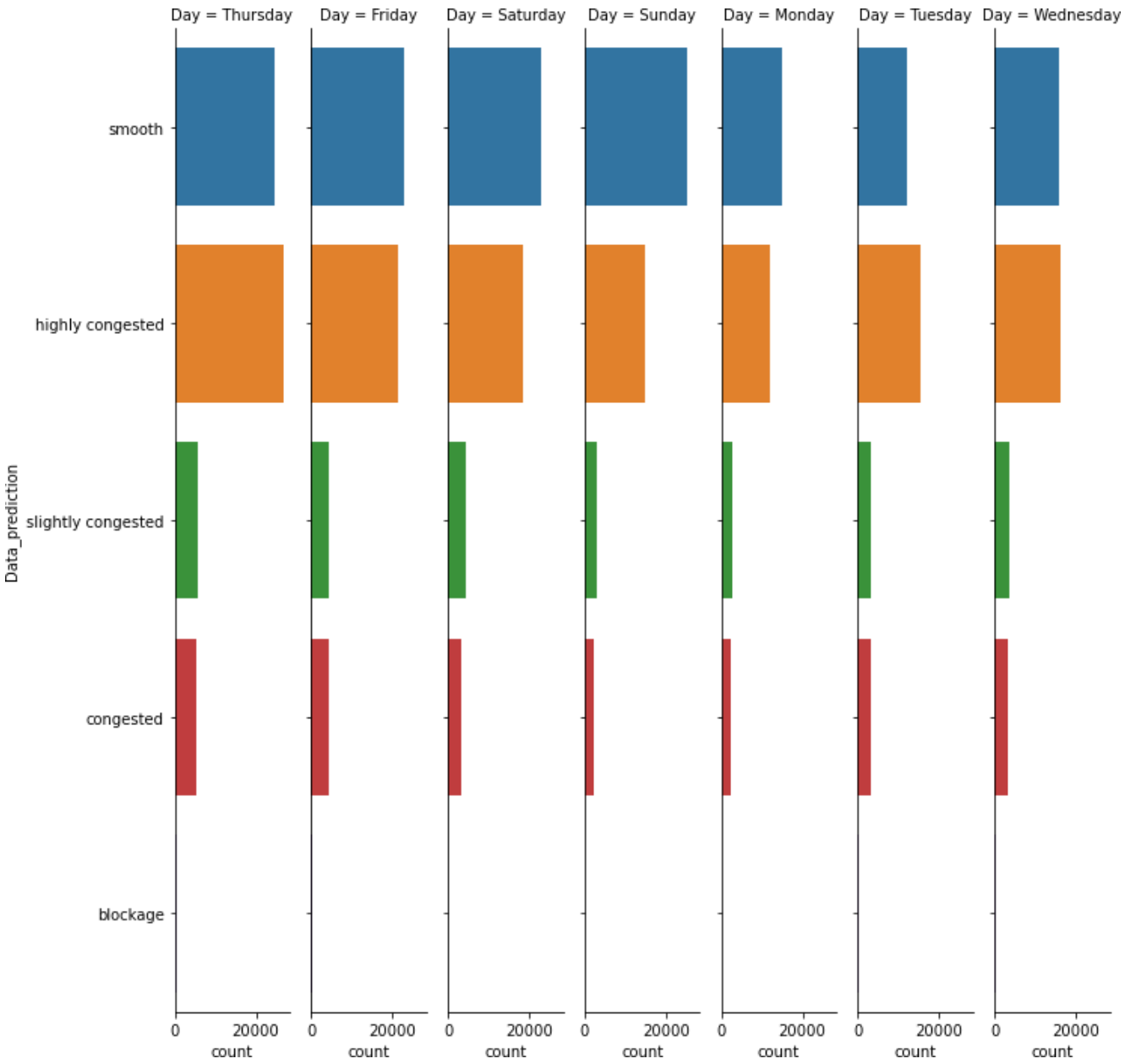


Figure 4*: Relationship between the congestion and days of the week.*

3.3 Methodologies used:

3.3.1 Decision Tree:

A decision tree organizes a series of rules in a tree structure and is used to solve classification problems in the machine learning domain. It provides high information at every node of the tree and therefore classification is performed at every node. The interpretation of the decision tree is much easier as compared to other models and therefore it is very useful in feature engineering.

The tree-like structure is divided into three types of nodes. This tree is upside down and the root is at the top and is referred to as the root node. The features are tested at each node and each step determines the part of the outcome. The root node branches out into decision and terminal nodes. This algorithm is based on classification. The classification algorithm involves separating and dividing the set of data into classes. The dataset had to be manipulated in such a way that it could determine the level of congestion precisely. Their final prediction was divided into five categories, therefore, making it a multi-class classification. The congestion level was decided by the time spent on the road segment and the least time spent on the road. Eighty percent of the data set was extracted for training while twenty percent of the dataset was used for testing the dataset.

Sometimes decision trees may lead to overfitting of the dataset therefore in this analysis, the bagging method is also employed to improve the accuracy. The bagging method is essentially a bootstrap aggregation, used to reduce variance and avoid overfitting. A random set of data is chosen and creates multisets from original data and the average of all sets is taken into account to improve prediction by reducing the variance.

3.3.2 Random Forest:

This machine-learning algorithm consists of several decision trees and each decision tree has one output. Output with maximum vote becomes the final prediction. The major advantage of using random forest is that it doesn't allow overfitting and takes care of high variance. This method utilizes the concept of multiple decision trees for making decisions

3.3.3 SVM:

A support vector machine is used for prediction in both regression and classification. The support vector margins are used to create the margin between the group. The line with greater margins supports better classification among groups. The line that creates maximum distance between the groups in an n-dimensional environment is called a hyperplane. While creating this line of separation or 'hyperplane' there are a few situations that are frequently observed. When the different groups are separated, often the margins could of low or have high gamma values. On the other hand, sometimes, the margins could be larger or have low gamma. Both of the approaches are right, gamma can be chosen depending upon the complexity of the dataset. Similarly, when the dataset has lower margins, it can lead to overfitting (or low regularization). Analysis as such can produce better results but it is quintessential to avoid overfitting, therefore it is pertinent to have high regularization. If the analysis has high regularization, it could erase the problem of overfitting. Since this algorithm incorporates hyperplane to distinguish between classes it acts are the separator between the two classes. This model works well when the dimensions in the dataset are high. This algorithm is also memory efficient.

For data with high complexity, this algorithm utilizes a transformation called kernel which can be represented as $z^{2 =} x^2 + y^2$.

The most common kernel used in SVM is the Linear kernel function, Polynomial kernel function, and Gaussian Radial bias function (RBF). This analysis has employed LinearSVC as this dataset has a large number of rows and columns and this statical approach is much faster in converging large datasets.

3.3.4 KNN:

This algorithm is useful when the dimensions are few as this algorithm calculates the distance between the units. The most common metric used for calculating distance is the Euclidean distance method. Vectors with similar distance ranges are classified into one group. This model is rather a simpler one and the precision gets reduced as the dataset increases. This algorithm works on the concept of proximity. If the data points are closer then they tend to be of the same group. The value of k plays a critical role. The value of k determines the number of data points that are going to be calculated. If the value of k is too low, the model is overfitted while if the value of k is too high, the model becomes unspecific.

3.3.5 Naïve Bayes:

This algorithm falls under the category of the probabilistic classifier. The concept is derived from the Bayes theorem which essentially means to find the probability of B happening given A has occurred. $P(A/B) = P(B/A)\ P(A)/\ P(B)$

In this project, Gaussian Naïve Bayes is used where the distribution of units is continuous.

3.3.6 Linear Regression

This is a linear model governed by the equation y = mx+c where y is a dependent variable and x is an independent variable. This model sketches the best fit line against all the data points.

3.3.7 Logistic Regression

This model is used for binary classification and uses the sigmoid function for prediction. This algorithm predicts categorical variables (as the output) by using dependent variables (as an input). This method is commonly used for solving problems like spam email detection or website advertisements. As we have discussed above linear regression uses y = mx+c for prediction, in logistic regression y uses the sigmoid function which can be represented as $S(x) = 1/(1+e^{-x})$ and y takes values from 0 to 1. If the chance of occurrence is or above 0.5 then it is classified as 1, and if the chance of occurrence is less than 0.5 then it is classified as 0.

3.3.8 ANN

This advanced machine learning algorithm tries to work in the same way as neurons in the brain. This algorithm is very useful for non-linear input data. This algorithm consists of neural layers. The layer of the neuron gets the information and input neuron and gets multiplied by a corresponding weight and information is relayed to the next layer of the neuron. This system also has a backpropagation system when the error is detected and the corresponding changes are made.

3.4 Abbreviations

i) KNN- K -nearest neighbors

ii) SVM – Support Vector Machines

iii) ANN – Artificial Neural Networks

iv) PCA – Principal Component Analysis

4. Analysis and Results

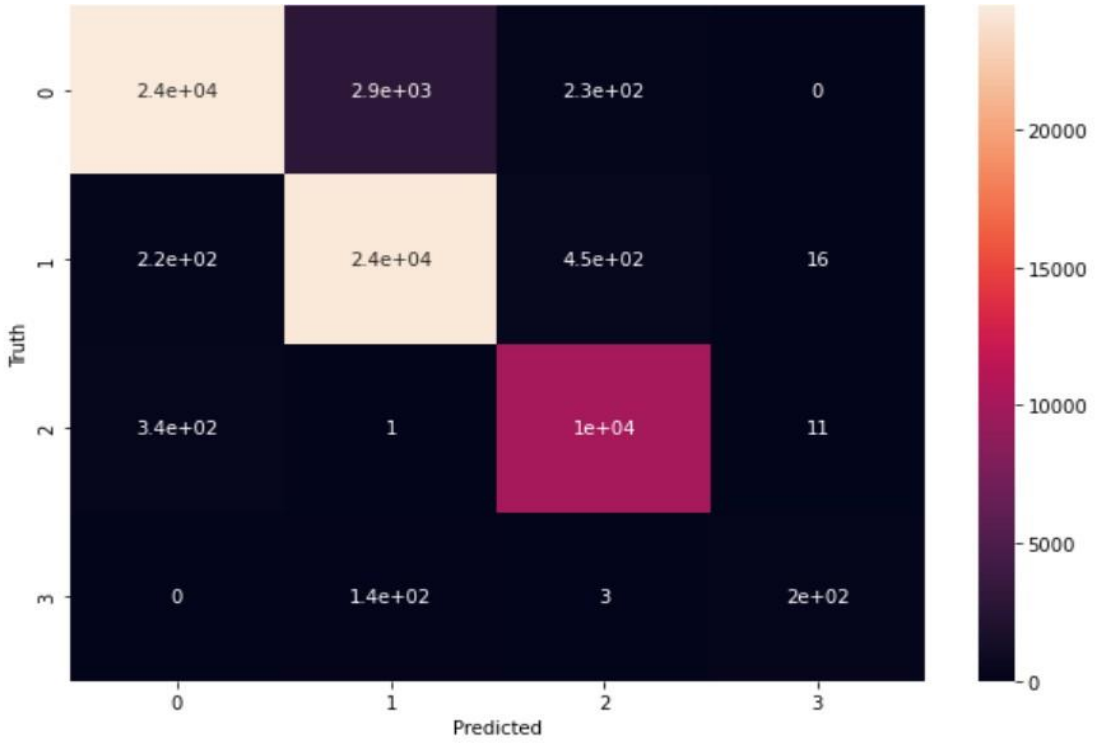
In this analysis the dataset is cleaned and manipulated, the cleaning of the dataset includes removing any null or 'NA' values as shown in Appendix A.1. There are a few columns that contain string values and therefore those columns are converted into numeric values as shown in Appendix A.2. The next step is to make sure that the data types are conducive to further manipulation as shown in A.3.

The first machine learning algorithm used was the decision tree. The DecisionTreeClassifier is imported from sklearn and used to train the dataset. The dataset is split in two ways. The training dataset consists of eighty percent of the dataset while the test dataset consists of twenty percent of the dataset. Dataset can be spilt in different ways as well. For instance, training and testing spilt could be performed in seventy – thirty or ninety – ten percent as well, but for this analysis, we have chosen eighty - twenty spilt. The training and testing analysis is shown in Appendix B.1. The model shows a high success rate of 93.2%. The decision tree is often prone to overfitting so to ensure that model is not prone to overfitting BaggingClassifier has been imported from sklearn.ensemble.

This would help tackle overfitting or getting the model non-generalized. The samples for cross-validation are taken as five and the mean score is 88.2 % which shows that the previous model was too specific. Appendix B.2 shows the detailed work.

The next model used is Random Forest. Inherently random forest uses advanced bagging methods, therefore the random tree is a robust model for this analysis since the dataset has a large number of rows and columns. For modeling the data, the dataset has eighty- twenty spilt for training and

testing. The RandomForestClassifier is imported from sklearn and the number of estimators used is forty. The analysis part is shown in Appendix B.3. The model shows a high success rate of 93.1 %. To achieve a better understanding of the analysis, a confusion matrix is constructed. The confusion matrix is also referred to as the error matrix. This matrix has 4*4 entries against all true and predicted values. The column in the confusion matrix corresponds to what machine learning algorithms have predicted while rows correspond to true values provided. The confusion matrix is shown in Figure 5.




```
[61]: cm

Out[61]: array([[24499,  2933,   234,     0],
                [  224, 24306,   448,    16],
                [  337,     1, 10063,    11],
                [    0,   145,     3,   203]], dtype=int64)
```

Few inferences can be made from the confusion matrix. For example, 24499 values are predicted to be ‘zero’ and the true values are also ‘zero’. 24306 values are predicted to be ‘one’ and the true

values are also ‘one’. 10063 values are predicted to be ‘two’ and the true values are also ‘two’. 203 values are predicted to be ‘three’ and the true values are also’ three’. On the other hand, 224 values are ‘one’ but predicted to be ‘zero’. 337 values are predicted to be zero but are ‘two’. If the actual value and predicted values happen to be the same, it adds to the accuracy of the model but if the actual value and predicted values mismatch it leads to an error. Therefore, the confusion matrix is a robust way to portray the efficiency of the model. The analysis part of the confusion matrix is shown in Appendix B.4.

The next model used for prediction is K-nearest neighbors. Again, the data is spilt into an eighty - twenty ratio for training and testing. The success rate of the model is high. To further investigate and evaluate the accuracy of the model, a classification model is drawn. The classification report is useful as it brings light to precision, recall, and F1 Score and support. Precision is considered to have a perfect score of 1.0 when there are no false positives. The macro average and the weighted average for the precision is .92 which is quite close to a perfect 1.0 score. Therefore, we can conclude that the number of false positives is quite low. The second component is called recall which corresponds to false negatives. The recall is said to have a score of 1.0 when there are no false positives. The macro average is 0.84 and the weighted average is 0.92. The most key component is the F1 score, which is the combination of precision and recall. The closer the value is to 1.0, the better the model is. The macro average is 0.87 and the weighted average is 0.92. The support is simply the representation of the number of samples used for calculations. Overall statistics show that the model has high efficiency and precision. The statics is shown in Figure 6.

```
from sklearn.metrics import classification_report

print(classification_report(y_test, y_pred))
```

```
              precision    recall  f1-score   support

           0       0.94      0.90      0.92     27734
           1       0.89      0.94      0.92     25036
           2       0.94      0.94      0.94     10295
           4       0.89      0.56      0.69       358

    accuracy                           0.92     63423
   macro avg       0.92      0.84      0.87     63423
weighted avg       0.92      0.92      0.92     63423
```

There is one other way to check for the accuracy of the model. We have drawn a confusion matrix to evaluate all the parameters. The confusion matrix is shown in Appendix B.8 along with the code.

As observed from the confusion matrix, 24877 values are predicted to be 0 and the true value is also 0. There are 23611 values that are predicted to be 1 and the true value is also 1.

There are 9725 values that are predicted to be 2 and the true value is also 2 lastly there are 202 values that are predicted to be 3 and the true value is also 3. There are 2640 values that are predicted to be 1 and the true value is also 0. This was the summary of the confusion matrix for the KNN algorithm prediction.

The next model for prediction used is the Support Vector Machine. In general, the most common kernel used is the Linear kernel function, Polynomial kernel function, and Gaussian Radial bias function (RBF). In this analysis, we have used LinearSVC. The training-testing data split is similar to the previous one. This particular methodology is efficient when the number of data points is high. This model did not prove to be highly successful in prediction. The analysis is shown in Appendix B.5.

The next model used is Naïve Bayes. It is very useful in prediction related to fraudulent cases or weather forecasting and works well when prediction has to be made based on previous outcomes. The probability of the class is based on like likelihood, class prior probability, and predictor prior probability. Even though it is easy to predict the data with this methodology and less training set is required. We have kept the proportion and training and testing similar to earlier ones. The few common methods used are BernoulliNB, CategoricalNB, ComplementNB, and MultinomialNB. The method employed is GauusianNB. Since this methodology works well with multi-class prediction, the result of the prediction is suboptimal. The analysis is shown in Appendix B.6.

The next model for analysis is Linear Regression. The model shows poor performance, to resolve this issue lasso and ridge techniques are used to implement L1 and L2 regularizations. Even after implementing these methods, the results could not be optimized. . The analysis is shown in Appendix B.7.

The next model implemented is Logistic Regression which uses the sigmoid function. The results from logistic regression were much more precise than linear regression and the Naïve Bayes algorithm. The dataset used was trained and tested with an eighty- twenty ratio. . The analysis is shown in Appendix B.8.

The next machine learning algorithm used is Artificial Neural Networks. This algorithm consists of a dense network of neurons. Pieces of information are fed to the first layer of neurons and then carried forward to the next layer and gets multiplied by the weighted average assigned to the neuron.

This algorithm has the highest success rate.

4.1. Analyzing the results

After implementing various machine learning algorithms, it is evident that ANN provides the best prediction as shown in Figure 9. Other than ANN, Random Forest and K -nearest neighbors provide optimal results. The decision tree algorithm had the challenges of overfitting which was corrected by bagging and cross-validation process. The most inefficient algorithm was Linear Regression and challenges could not be corrected by regularizations.

Decision Tree	93.2 %
Decision Tree with Bagging and Cross – Validation	88.2%
Random Forest	93.1%
K- nearest Neighbors	92.1%
SVM	71.7%
Naïve Bayes	52.3%
Logistic Regression	81.9%
Linear Regression	18.9%
Linear Regression with Ridge	18.03%
ANN	**95.8%**

Figure 9: *Table for models and their accuracy*

5. Conclusions and Future Recommendations

In this project, the main aim was to find the best methods for the detection of traffic congestion. Artificial neural networks provide the best solution as it has a few numbers of layers it uses to extract features. Even though the analysis has produced robust solutions, there are a few factors that could improve the quality of work. For example, this analysis did not account 'type of vehicles' into account. The type of vehicle plays a major role in determining traffic control on the road. The dataset did not include any impact of the disturbance or interruption on the road. Interruptions like accidents, road maintenance, or any other hindrances could impact the traffic on the roads. The dataset had a column named 'special events' to evaluate the level of impact events or specific occasions has on traffic. Even though this feature was included in the analysis, all the data point in this feature was 'no'. This happened because all the data collection was done on a non-event day. This defeats the purpose of evaluating traffic under any unusual conditions. Similarly, there is another column called 'Holidays' where all the entries in the column are 'no', therefore this analysis gives a narrow vision of traffic congestion. If the data collection was performed in a way that it could hold valuable information like 'special events' and 'holidays', then the information gained through the analysis would be far wider.

6. Reference

Huang, Xia, Li, Li &Li. (2019). A Peak Traffic Congestion Prediction Method Based on Bus Driving Time.

Kan, Tang, Kwan, Ren, Liu & Li. (2019). Traffic congestion analysis at the turn level using Taxis' GPS trajectory data.

Wu, Ho & Lee. (2004). Travel-Time Prediction with Support Vector Regression.

Duraku & Ramadani. (2019). Development of Traffic Volume Forecasting Using Multiple Regression Analysis and Artificial Neural Network.

Jiang, Abdel, Hu & Lee. (2015). Investigating macro-level hot zone identification and variable importance using big data: A random forest models approach.

Zafar & Haq. (2020). Traffic congestion prediction based on Estimated Time of Arrival.

Han, Wang & Shaw. (2006). Visual Exploratory Data Analysis of Traffic Volume.

Vlahogiann, Karlaftis, &Orfanou. (2012). Modeling the Effects of Weather and Traffic on the Risk of Secondary Incidents.

Wang, Wright, Thatcher &Wu. (2019). Traffic Volume Prediction with Segment-Based Regression and its Implementation in Assessing the Impact of Heavy Vehicles.

Appendix A

A.1 Data cleaning involves removing null and ‘NA’ values.

```
In [5]: df.isna().any()

Out[5]: Date                    False
        Day                     False
        Destination_Location    False
        Fastest_Route_Distance  False
        Fastest_Route_Name      False
        Fastest_Route_Time      False
        Holiday                 False
        Special_Condition       False
        Starting_Location       False
        Sys_Time                False
        Weather                 False
        Data_prediction         False
        dtype: bool

In [6]: df.isnull().sum()

Out[6]: Date                    0
        Day                     0
        Destination_Location    0
        Fastest_Route_Distance  0
        Fastest_Route_Name      0
        Fastest_Route_Time      0
        Holiday                 0
        Special_Condition       0
        Starting_Location       0
        Sys_Time                0
        Weather                 0
        Data_prediction         0
        dtype: int64
```

Converting the string values into numeric values. For instance, the ‘Holiday’ and ‘Special Condition’ columns have yes and no values which are converted into ‘1 ‘and ‘0’. Similarly, the columns like ‘Weather’ which has different string values are converted into numeric groups ranging from 1 to 9. Along similar lines, the data prediction values range from 0 to 4. By making these changes the data types are altered and the final data types are favorable for further analysis.

A.2

```
[13]: df1 = df1.replace({'Holiday': {'no': 0, 'yes': 1}})

[14]: df1 = df1.replace({'Special_Condition': {'no': 0, 'yes': 1}})
```

A.3

```
In [17]: df1.dtypes

Out[17]: Date                    object
         Day                      int64
         Destination_Location    object
         Fastest_Route_Distance   int64
         Fastest_Route_Name      object
         Fastest_Route_Time       int64
         Holiday                  int64
         Special_Condition        int64
         Starting_Location       object
         Sys_Time                object
         Weather                 object
         Data_prediction         object
         dtype: object
```

Appendix B

B.1 Training and splitting of data in eighty – twenty ratios.

```
In [ ]: # Decision Tree

In [44]: from sklearn import tree
         model = tree.DecisionTreeClassifier()

In [47]: x_train, x_test, y_train, y_test = train_test_split(x,y,test_size=0.2)

In [48]: model.fit(x_train,y_train)

Out[48]: DecisionTreeClassifier()
```

B.2 Importing Bagging Classifier to overcome the challenges of overfitting model**.**

```
In [52]: from sklearn.model_selection import cross_val_score
         from sklearn.tree import DecisionTreeClassifier

         scores = cross_val_score(DecisionTreeClassifier(), x, y, cv=5)
         scores
         from sklearn.ensemble import BaggingClassifier

         bag_model = BaggingClassifier(
             base_estimator=DecisionTreeClassifier(),
             n_estimators=100,
             max_samples=0.8,
             oob_score=True,
             random_state=0
         )
         bag_model.fit(x_train, y_train)
         bag_model.oob_score_

Out[52]: 0.9306079491030356

In [54]: bag_model = BaggingClassifier(
             base_estimator=DecisionTreeClassifier(),
             n_estimators=100,
             max_samples=0.8,
             oob_score=True,
             random_state=0
         )
         scores = cross_val_score(bag_model, x, y, cv=5)
         scores

Out[54]: array([0.96133895, 0.96058212, 0.92523099, 0.83436347, 0.73323137])

In [55]: scores.mean()

Out[55]: 0.8829493805901117
```

B.3 Analysis using Random Forest algorithms

```
#Random Forest
```

```
from sklearn.model_selection import train_test_split
x_train, x_test, y_train, y_test = train_test_split(x,y,test_size=0.2)
```

```
from sklearn.ensemble import RandomForestClassifier
model = RandomForestClassifier(n_estimators=40)
model.fit(x_train, y_train)
```

```
57]: RandomForestClassifier(n_estimators=40)
```

```
model.score(x_test, y_test)
```

```
58]: 0.931381360074421
```

B.4 Analysis of the confusion matrix

```
In [59]: y_predicted = model.predict(x_test)
```

```
In [60]: from sklearn.metrics import confusion_matrix
cm = confusion_matrix(y_test, y_predicted)
cm
%matplotlib inline
import matplotlib.pyplot as plt
import seaborn as sn
plt.figure(figsize=(10,7))
sn.heatmap(cm, annot=True)
plt.xlabel('Predicted')
plt.ylabel('Truth')
```

B.5 Analysis using Support Vector Machine algorithms

```
In [ ]: #svm
```

```
In [68]: from sklearn import svm
from sklearn.svm import LinearSVC

x_train, x_test, y_train, y_test = train_test_split(x, y, test_size=0.2 )
```

```
In [70]: model_svm = svm.LinearSVC(dual=False)
model_svm.fit(x_train, y_train)
```

```
Out[70]: LinearSVC(dual=False)
```

```
In [71]: model_svm.score(x_test, y_test)
```

```
Out[71]: 0.7176103306371505
```

B.6 Analysis using Naïve Bayes method

```
## Naive Bayes
```

```
from sklearn.model_selection import train_test_split
x_train, x_test, y_train, y_test = train_test_split(x,y,test_size=0.2)
```

```
from sklearn.naive_bayes import GaussianNB
model = GaussianNB()
```

```
model.fit(x_train,y_train)
```

```
GaussianNB()
```

```
model.score(x_test,y_test)
```

```
0.5238320483105497
```

B.7 Analysis using Linear Regression

```
##Linear Regression(Lasso and Ridge)
```

```
from sklearn.linear_model import LinearRegression
reg =  LinearRegression().fit(X_train, y_train)
```

```
reg.score(X_test, y_test)
```

```
0.1897635018867735
```

```
from sklearn.linear_model import Ridge
ridge_reg = Ridge(alpha = 50 , max_iter = 100, tol = 0.1)
ridge_reg.fit(X_test, y_test)
```

```
Ridge(alpha=50, max_iter=100, tol=0.1)
```

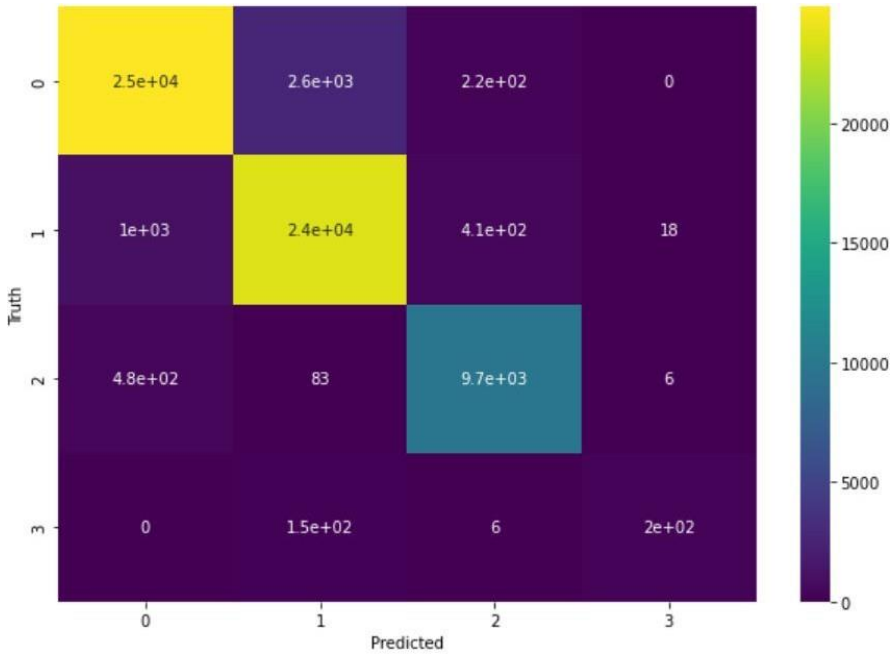
```
ridge_reg.score(X_test, y_test)
```

```
0.18031927811516013
```

```
ridge_reg.score(X_train, y_train)
```

```
0.1806682900526534
```

B.8 Confusion matrix for KNN Model



In [57]:
```
from sklearn.metrics import confusion_matrix
y_pred = knn.predict(x_test)
cm = confusion_matrix(y_test, y_pred)
cm
```

Out[57]:
```
array([[24877,  2640,   217,     0],
       [ 1001, 23611,   406,    18],
       [  481,    83,  9725,     6],
       [    0,   150,     6,   202]], dtype=int64)
```