A

Mid Term Assessment Project Report

On

Prediction of Traffic Flow using K-NN Algorithm and Random Forest Algorithm

Submitted To University,

In The Partial Fulfillment of The Requirements

For The Awards of The

Submitted By

Prathamesh S. Deshmukh

(Final Year B- Tech in Mechanical Engineering, Div. 04)

Under The Guidance Of

Prof. Vaidehi Deshmukh

MIT World Peace University, Pune – 38

For the Academic Year

2023-2024

Dr. Vishwanath Karad

MIT World Peace University, Pune

School of Mechanical Engineering

CERTIFICATE

This is to certify that ***Mr. Prathamesh Deshmukh***, has successfully completed the mid-term assessment project **"*Prediction of Traffic Flow using K-NN Algorithm and Random Forest Algorithm*"** under my supervision, in the partial fulfillment of final year, seven semester - School of Mechanical Engineering of Dr. Vishwanath Karad MIT World Peace University, Pune for the academic year 2022-2023.

Date: 28/10/2023

Place: Pune

Prof. Vaidehi Deshmukh **Prof. Dr. G. M. Kakandikar**

Subject Guide **Head of School**

Seal

Dr. Vishwanath Karad MIT WORLD PEACE UNIVERSITY PUNE	Vision and Mission

Vision of Faculty of Engineering and Technology

To be a globally recognized leader in Engineering Education having a constructive impacton society.

Mission of Faculty of Engineering and Technology

1. To achieve academic excellence through Continuously Updated Education (CUEd).
2. To create environment of technology research and social innovation by establishing trans-disciplinary centres of excellence.
3. To establish strong partnerships with reputed industry and research organizations.
4. To promote ethical and universal value based professional education.

Vision of School of Mechanical Engineering

To be a center of academic excellence in Mechanical Engineering for sustainable development.

Mission of School of Mechanical Engineering

1. To impart comprehensive knowledge in mechanical engineering and foster skills to enhance economic development of the nation.
2. To facilitate trans-disciplinary research leading to innovative technologies.
3. To collaborate with academia, industry and research organization globally.
4. To nurture students for lifelong learning and ethical professional career.

ACKNOWLEDGEMENT

It gives me great pleasure to present a mid-term assessment project on ***"Prediction of Traffic Flow using K-NN Algorithm and Random Forest Algorithm."*** In presenting this mini project work, several hands helped me directly and indirectly. Therefore, it becomes my duty to express my gratitude towards them.

I am very much obliged to subject guide **Prof. Vaidehi Deshmukh**, in School of Mechanical Engineering, for helping and giving proper guidance. His timely suggestions made it possible to complete this seminar for me. All efforts might have gone in vain without his valuable guidance.

I will fail in my duty if I will not acknowledge a great sense of gratitude to the head of School of Mechanical Engineering **Dr. Shivprakash B. Barve** and the entire staff members in the School of Mechanical Engineering for their cooperation.

Prathamesh Deshmukh

Final Year Mechanical Engg.

PRN No: 1032210678

Roll No: PD54 Div: 04

ABSTRACT

Due to the ever-growing demand and rapid urbanization, traffic pattern prediction has become a focal point for researchers in recent decades. The confluence of abundant traffic-related data and the emergence of machine learning techniques has driven the advancement of data-driven approaches within this domain. Predicting traffic flow enables the implementation of measures to reduce traffic congestion. This not only saves commuters' time but also reduces fuel consumption and greenhouse gas emissions, contributing to environmental sustainability. This research harnesses loop counter data to create models capable of forecasting traffic flow for various future time intervals. The study involves a thorough investigation of the data and rigorous statistical testing to pinpoint high-quality, informative features.

Multiple feature sets are assessed using various machine learning methods, such as Ridge Regression, Support Vector Regression (SVR), and Random Forests. The results emphasize the importance of meticulous feature extraction, indicating that feature selection is equally, if not more, critical than the choice of machine learning methods.

Efficient feature engineering enables the adoption of simpler methods, which are more efficient, dependable, and easier to maintain. In conclusion, the study provides insights into potential avenues for enhancing traffic predictions even further.

Table of Contents

List of Figures

Chapter No. 1
Introduction

1.1 Background Concept

Traffic congestion can exert substantial impacts on people's quality of life, particularly in larger cities. Statistics indicate that in the United States, traffic congestion results in the wastage of approximately two billion gallons of fuel each year, with around 135 million drivers collectively spending a staggering two billion hours annually stuck in traffic. This predicament translates to an overall cost of 100 billion USD solely in fuel expenses, equating to an average annual expense of 800 USD for the typical American driver Beyond the economic implications, there are also environmental concerns, as reducing travel time can significantly curtail emissions and pollution.[1]

The aforementioned data underscores because governments are increasingly investing in Intelligent Transportation Systems (ITS) technologies aimed at enhancing the efficient use of transportation networks. Traffic prediction models have become integral components of most ITS. The provision of precise real-time information and forecasts regarding traffic flow is pivotal within these systems. ITS encompasses a range of technologies, including advanced travel information systems, variable message signs, traffic signal control systems, and user-friendly applications like travel advisors.[2] The overarching goal of these technologies is consistent: to facilitate smoother traffic flow, alleviate congestion, and decrease travel times by guiding drivers regarding their routes, departure times, and even transportation modes.

The wealth of traffic-related data gathered from diverse sources, coupled with the emergence of sophisticated machine learning algorithms, has ushered in a significant shift from analytical modeling to data-driven modeling approaches. [3] This paper's primary focus lies in investigating various machine learning algorithms and crafting features that empower us to predict traffic conditions at multiple points in the future.

1.2 Objectives

Generally, studies on traffic prediction can be classified into three primary categories: naive approaches, parametric techniques, and non-parametric methods. Naive methods typically consist of straightforward, non-model baseline predictors, which can sometimes yield favorable results. Parametric models are grounded in traffic flow theory and are studied separately alongside non-parametric, more data-centric machine learning methods. In recent years, there has been a notable shift towards non-parametric methods, possibly attributed to increased data availability, advancements in computational capabilities, and the development of more advanced algorithms. [4]

Non-parametric methods should not be misconstrued as devoid of parameters; rather, they pertain to model parameters that are adaptable and not predetermined. Both the model's structure and its parameters are determined from the available data. An important advantage of this approach is that it necessitates less domain-specific expertise compared to parametric methods, but it does require a greater volume of data to establish a model. [5] This also implies that the successful application of data-driven models is closely linked to the quality of the available data.

On the contrary, traffic conditions in urban areas can exhibit significantly greater dynamism and non-linearity, primarily due to the abundance of intersections and traffic signals. In such contexts, data-driven machine learning techniques like neural networks, Random Forests, and K-NN can be more suitable due to their ability to model intricate nonlinear relationships and dynamic processes.

Chapter No. 2
Basics of Algorithms

2.1 K-NN Algorithm

The K-Nearest Neighbors (KNN) algorithm is a simple and intuitive machine learning technique used for both classification and regression tasks. It operates based on the principle that data points with similar features tend to belong to the same class or have similar values. [6]

2.1.1 Step-by-Step Explanation of the K-Nearest Neighbors (KNN) Algorithm:

1. **Input Data:** The KNN algorithm begins with a dataset containing labeled instances. Each instance consists of features (attributes) and a corresponding class label (for classification) or a target value (for regression).
2. **Choose the Number of Neighbors (K):** You need to specify the number of nearest neighbors (K) to consider when making predictions. This is a hyperparameter that you can adjust based on the characteristics of your data and problem.
3. **Distance Metric**: Select a distance metric, such as Euclidean distance or Manhattan distance, which measures the similarity between data points. The choice of distance metric affects how "closeness" is defined.
4. **Calculate Distances**: For a new, unlabeled data point (the one you want to predict for), calculate the distance between that data point and all the data points in the training dataset. You use the chosen distance metric for this calculation.
5. **Sort by Distance:** Sort the distances calculated in the previous step in ascending order, so that the nearest data points come first in the list.
6. **Select the Top K Neighbors:** Choose the top K data points (those with the smallest distances) from the sorted list. These K data points are the "nearest neighbors."
7. **Majority Voting (Classification) or Weighted Averaging (Regression):**
    a. **Classification:** If you are using KNN for classification, count the class labels of the K nearest neighbors. The predicted class for the new data point is the class that occurs most frequently among these K neighbors. This is known as majority voting.

b. **Regression:** If you are using KNN for regression, calculate the average (mean or weighted mean) of the target values of the K nearest neighbors. This average is your prediction for the new data point's target value.

8. **Output Prediction:** The algorithm provides the predicted class (for classification) or the predicted value (for regression) as the final output.

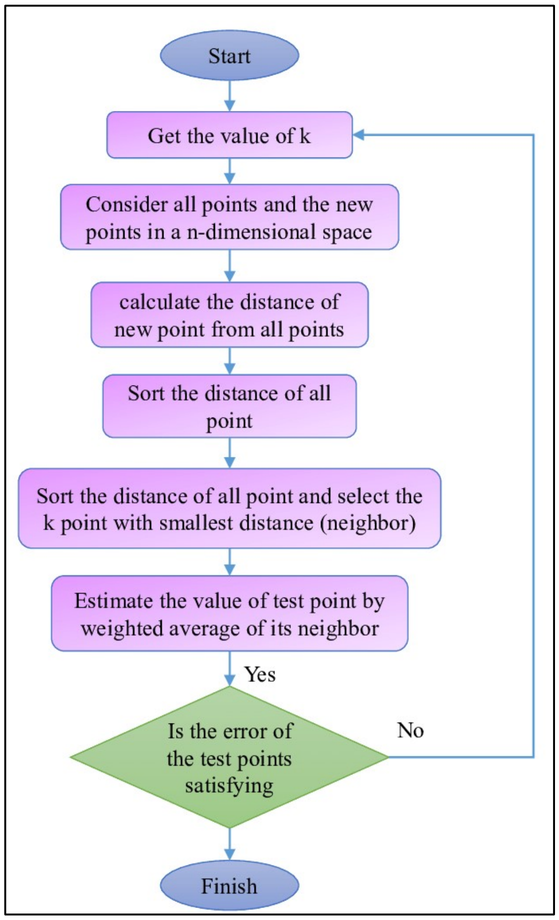

Start
Get the value of k
Consider all points and the new points in a n-dimensional space
calculate the distance of new point from all points
Sort the distance of all point
Sort the distance of all point and select the k point with smallest distance (neighbor)
Estimate the value of test point by weighted average of its neighbor
Yes
Is the error of the test points satisfying
No
Finish

Fig. 2.1 Flowchart of KNN Algorithm

2.2 Random Forest Algorithm

The Random Forest algorithm is an ensemble learning method that is widely used for both classification and regression tasks in machine learning. It is based on the idea of building multiple decision trees and combining their predictions to improve accuracy and reduce the risk of overfitting. [7]

2.2.1 Step-by-Step Explanation of the Random Forest Algorithm:

1. **Input Data**: The Random Forest algorithm begins with a dataset containing labeled instances. Each instance has features (attributes) and a corresponding class label (for classification) or a target value (for regression).
2. **Bootstrapping (Random Sampling):** A random subset of the training data is sampled with replacement. This process is called bootstrapping. The size of the subset is typically equal to the size of the original dataset, but it may include some instances multiple times (with replacement) and exclude others. This step generates multiple "bootstrap" datasets.
3. **Random Feature Selection**: When constructing each decision tree, a random subset of features is considered at each node of the tree. This random feature selection introduces diversity among the trees, which helps reduce overfitting and improves the generalization of the model.
4. **Build Decision Trees:** For each bootstrap dataset and using the randomly selected features, a decision tree is constructed. These trees are often referred to as "weak learners" because they can overfit to the training data. However, this overfitting is less of a concern when the predictions from multiple trees are combined.
5. **Majority Voting (Classification) or Averaging (Regression):**
    a. **Classification:** When making predictions for a new data point, all the decision trees in the forest "vote" for a class. The class that receives the majority of votes is the predicted class for the new data point.
    b. **Regression:** In regression tasks, the forest combines the predictions of all the decision trees by averaging the target values (or taking a weighted average) to make the final prediction.

6. **Output Prediction:** The algorithm provides the predicted class (for classification) or the predicted value (for regression) as the final output.

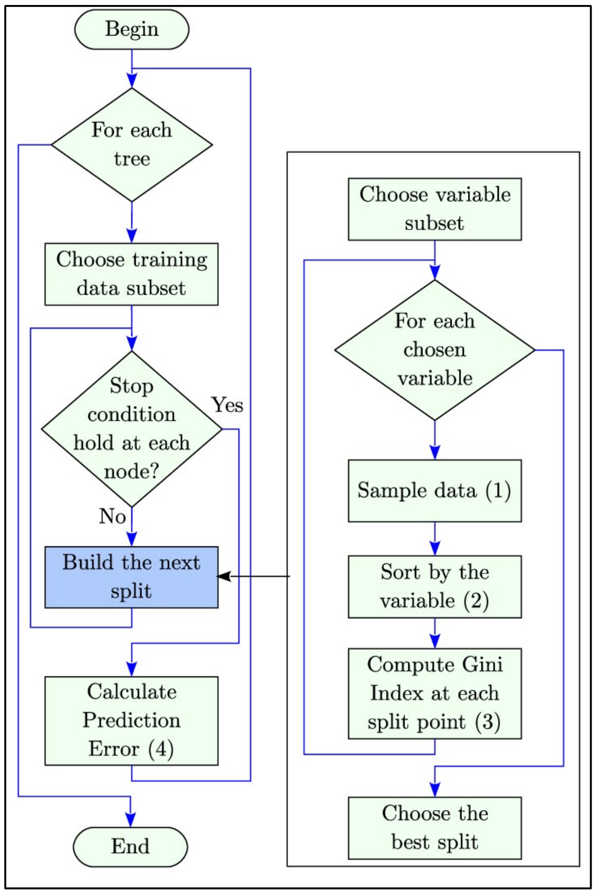


Fig. 2.2 Flowchart of KNN Algorithm

Chapter No. 3
Case Study

3.1 Background Concept

One occurrence that significantly affects our daily routines is the process of commuting. Like many urban inhabitants, we find ourselves commuting to work daily, often adhering to tight schedules. To illustrate, consider the situation where you have an important early morning meeting at the office tomorrow.

To ensure you arrive on time – neither late nor excessively early – you must leave early and also make sure your commute duration falls within the safe time frame for a punctual arrival. What you require is a recommendation system capable of advising you on the optimal departure time from your home, guaranteeing you reach the office precisely when needed. [8]

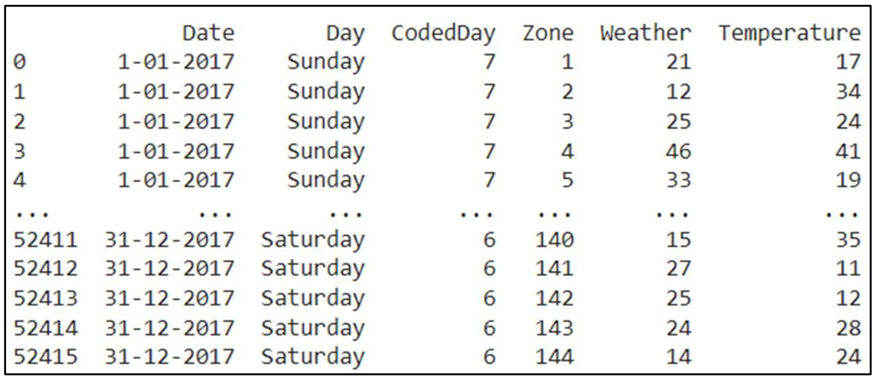
In this study, loop counter data are used to develop models that can predict traffic flow for several different prediction intervals into the future. In depth exploratory data analysis and statistical testing is performed to obtain good quality informative features.

3.2 Factor Influencing Traffic

- **Hour of the Day:** Travel duration is predominantly influenced by this variable, as traffic congestion on city roads peaks during busy hours.
- **Day of the Week:** Weekdays consistently experience higher traffic density, primarily due to the influx of office commuters, in contrast to weekends.
- **Weather Conditions:** Adverse weather conditions can disrupt roadways and public infrastructure, leading to unfavorable effects on travel time.
- **Temperature:** Extremely high or low temperatures often encourage people to remain indoors, resulting in reduced traffic density and faster travel.

3.3 Dataset of Traffic

The require dataset of the traffic from a specific region, specifically classified as from destination A to destination B was collected from the open source website called Mapquest API.



```
            Date       Day  CodedDay  Zone  Weather  Temperature
0      1-01-2017    Sunday         7     1       21           17
1      1-01-2017    Sunday         7     2       12           34
2      1-01-2017    Sunday         7     3       25           24
3      1-01-2017    Sunday         7     4       46           41
4      1-01-2017    Sunday         7     5       33           19
...          ...       ...       ...   ...      ...          ...
52411 31-12-2017  Saturday         6   140       15           35
52412 31-12-2017  Saturday         6   141       27           11
52413 31-12-2017  Saturday         6   142       25           12
52414 31-12-2017  Saturday         6   143       24           28
52415 31-12-2017  Saturday         6   144       14           24
```

Fig. 3.3 Insights of Dataset

1 **Time of Day (Zone Column):** Each 10-minute interval of the day is represented by a numerical code, dividing the 24-hour day into 144 zones. For instance, the code 1 corresponds to the 10-minute period from 00:00 to 00:10 Hrs, and code 2 represents the subsequent 10-minute slot from 00:10 to 00:20 Hrs, and so forth.

2 **Day of the Week (Coded Day Column):** The day of the week is transformed into a numerical code, with each of the 7 weekdays assigned a unique number, starting from 1 (Sunday) and ending at 7 (Saturday).

3 **Weather Conditions (Coded Weather Column):** Weather conditions are encoded using numerical values. You can refer to the provided codes to understand the representation of different weather conditions within the training set.

4 **Temperature (Temperature Column**): This column contains the average daily temperature, measured in Fahrenheit.

Furthermore, the training dataset includes the real-time travel duration in minutes, denoted in the Realtime Column, for each individual record. In summary, here's a comprehensive list of the six parameters that make up a single data entry in the training set:

1. The initial departure time from the source or point A (indicated by the Date Column).
2. The 10-minute time interval of the day (depicted in the Zone Column).

3. The day of the week (represented by both the Day and CodedDay Columns, which convey the same information).
4. Weather conditions on that specific day (expressed through the CodedWeather Column).
5. The temperature on that particular day (recorded in the Temperature Column).
6. The actual traffic flow from point A to the destination, or point B, when an individual commenced their journey from the source at the time specified in the Date Column (measured in the Realtime Column).

Chapter No. 4
Model Construction

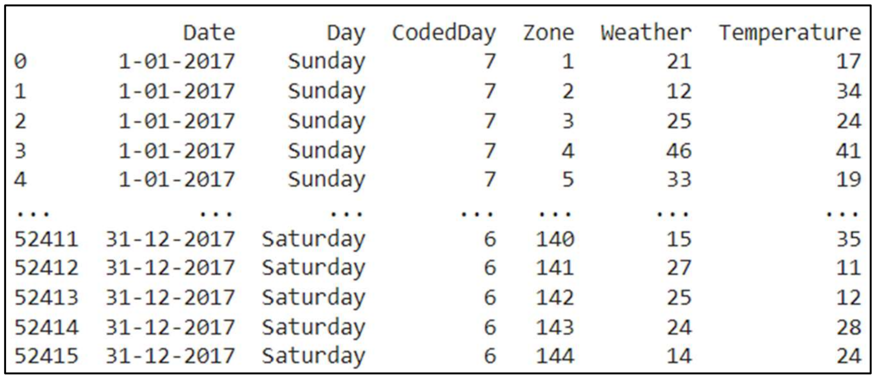
4.1 Importing the Dataset and Pandas

To utilise the functions in a module, you must first import it with an import statement. An import statement consists of the import keyword and the name of the module. This will be stated at the top of the Python file, under any shebang lines or general comments.
Pandas is a robust data analysis tool. It simplifies data exploration and manipulation. It provides a number of functions for reading data from various sources.

```
import matplotlib.pyplot as plt
import pandas as pd
from sklearn.preprocessing import StandardScaler
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LinearRegression
from sklearn.neighbors import KNeighborsClassifier
from sklearn.metrics import accuracy_score, confusion_matrix, f1_score

# Importing the training dataset
train_dataset = pd.read_csv('train_set.csv')
X_train = train_dataset.iloc[:, [2, 3, 4, 5]].values
y_train = train_dataset.iloc[:, 6].values

# Importing the testing dataset
test_dataset = pd.read_csv('test_set.csv')
X_test = test_dataset.iloc[:, [2, 3, 4, 5]].values
y_test = test_dataset.iloc[:, 6].values
print(train_dataset)
```



```
             Date       Day  CodedDay  Zone  Weather  Temperature
0       1-01-2017    Sunday         7     1       21           17
1       1-01-2017    Sunday         7     2       12           34
2       1-01-2017    Sunday         7     3       25           24
3       1-01-2017    Sunday         7     4       46           41
4       1-01-2017    Sunday         7     5       33           19
...           ...       ...       ...   ...      ...          ...
52411  31-12-2017  Saturday         6   140       15           35
52412  31-12-2017  Saturday         6   141       27           11
52413  31-12-2017  Saturday         6   142       25           12
52414  31-12-2017  Saturday         6   143       24           28
52415  31-12-2017  Saturday         6   144       14           24
```

Fig. 4.1 Dataset Output

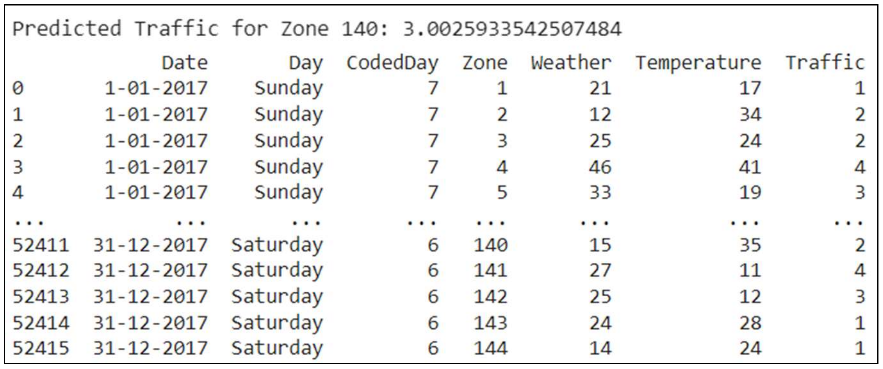
4.2 Depicting the Traffic Index

The test dataset does not contain any index of the traffic congestion which would help us to realize on what day and in which zone the traffic was highest and lowest. And, to find such index, simple linear regression model has been implemented on the previous dataset using scaler transform.

```
X = data[['Zone', 'Weather', 'Temperature']]
y = data['Traffic']
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)
scaler = StandardScaler()
X_train = scaler.fit_transform(X_train)
X_test = scaler.transform(X_test)

#  Linear Regression model
model = LinearRegression()
model.fit(X_train, y_train)
specific_zone = 5
specific_weather = 33
specific_temperature = 19
specific_data = scaler.transform([[specific_zone, specific_weather, specific_temperature]])
```

```
traffic_prediction = model.predict(specific_data)
print(f"Predicted Traffic for Zone {specific_zone}: {traffic_prediction[0]}")
print(train_dataset)
```



```
Predicted Traffic for Zone 140: 3.0025933542507484
             Date       Day  CodedDay  Zone  Weather  Temperature  Traffic
0       1-01-2017    Sunday         7     1       21           17        1
1       1-01-2017    Sunday         7     2       12           34        2
2       1-01-2017    Sunday         7     3       25           24        2
3       1-01-2017    Sunday         7     4       46           41        4
4       1-01-2017    Sunday         7     5       33           19        3
...           ...       ...       ...   ...      ...          ...      ...
52411  31-12-2017  Saturday         6   140       15           35        2
52412  31-12-2017  Saturday         6   141       27           11        4
52413  31-12-2017  Saturday         6   142       25           12        3
52414  31-12-2017  Saturday         6   143       24           28        1
52415  31-12-2017  Saturday         6   144       14           24        1
```

Fig. 4.2 Traffic Index Prediction

4.3 Label Encoder & OneHotEncoder

We frequently deal with a range of labels in supervised learning. These can be either numbers or words. If they are numbers, the algorithm can immediately utilise them. However, labels must frequently be in human readable form. As a result, users often label the training data using words. Label encoding is the process of converting word labels into numerical representations so that algorithms can comprehend how to act on them. Let's have a look at how to accomplish this.

One hot encoding is a procedure that converts categorical information into a form that can be fed into ML algorithms to help them predict better.

```
from sklearn.preprocessing import LabelEncoder, OneHotEncoder
labelencoder_X = LabelEncoder()

X_train[:, 2] = labelencoder_X.fit_transform(X_train[:, 2])
onehotencoder = OneHotEncoder(categorical_features = [2])
X_train = onehotencoder.fit_transform(X_train).toarray()

X_train[:, 3] = labelencoder_X.fit_transform(X_train[:, 3])
onehotencoder = OneHotEncoder(categorical_features = [3])
```

```
X_train = onehotencoder.fit_transform(X_train).toarray()

X_train[:, 4] = labelencoder_X.fit_transform(X_train[:, 4])
onehotencoder = OneHotEncoder(categorical_features = [4])
X_train = onehotencoder.fit_transform(X_train).toarray()

X_train[:, 5] = labelencoder_X.fit_transform(X_train[:, 5])
onehotencoder = OneHotEncoder(categorical_features = [5])
X_train = onehotencoder.fit_transform(X_train).toarray()

X_test[:, 2] = labelencoder_XY.fit_transform(X_test[:, 2])
onehotencoder = OneHotEncoder()
X_test = onehotencoder.fit_transform(X_test[:,[2]]).toarray()

X_test[:, 3] = labelencoder_XY.fit_transform(X_test[:, 3])
onehotencoder = OneHotEncoder()
X_test = onehotencoder.fit_transform(X_test[:,[3]]).toarray()

X_test[:, 4] = labelencoder_XY.fit_transform(X_test[:, 4])
onehotencoder = OneHotEncoder()
X_test = onehotencoder.fit_transform(X_test[:,[4]]).toarray()

X_test[:, 5] = labelencoder_XY.fit_transform(X_test[:, 5])
onehotencoder = OneHotEncoder()
X_test = onehotencoder.fit_transform(X_test[:,[5]]).toarray()
```

4.4 Feature Scaling and K-NN Fitting in the Dataset

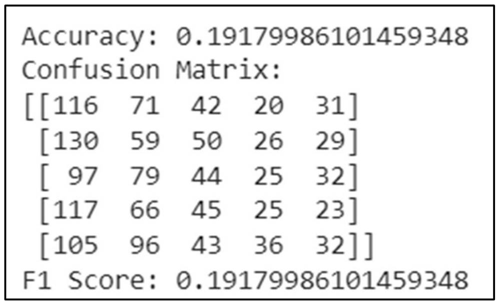
Feature scaling is a technique for standardizing the range of independent variables or data characteristics. It is also known as data normalization in data processing and is often conducted during the data pre-processing stage.

```
from sklearn.preprocessing import StandardScaler
sc = StandardScaler()
X_train = sc.fit_transform(X_train)
X_test = sc.transform(X_test)

from sklearn.neighbors import KNeighborsClassifier
classifier = KNeighborsClassifier(n_neighbors = 3, metric = 'minkowski', p = 2)
classifier.fit(X_train, y_train)

# Predicting the Test set results
y_pred = classifier.predict(X_test)

# Making the Confusion Matrix
from sklearn.metrics import accuracy_score
ac = accuracy_score(y_test, y_pred)
from sklearn.metrics import confusion_matrix
cm = confusion_matrix(y_test, y_pred)
from sklearn.metrics import f1_score
f1_score(y_test, y_pred, average='micro')
```



```
Accuracy: 0.19179986101459348
Confusion Matrix:
[[116  71  42  20  31]
 [130  59  50  26  29]
 [ 97  79  44  25  32]
 [117  66  45  25  23]
 [105  96  43  36  32]]
F1 Score: 0.19179986101459348
```

Fig. 4.4 K-NN Algorithm Output

4.5 Implementing Random Forest Algorithm

A confusion matrix is a table that is frequently used to describe the performance of a classification model (or "classifier") on a set of test data with known true values. The confusion matrix itself is straightforward to grasp, but the associated language can be perplexing.

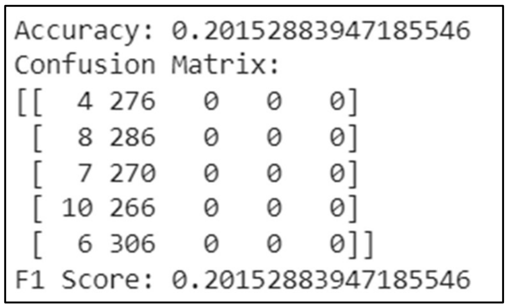
```
from sklearn.ensemble import RandomForestClassifier
rand_classifier = RandomForestClassifier(n_estimators = 1000, criterion = 'entropy', random_state = 0)
rand_classifier.fit(X_train, y_train)

# Predicting the Test set results
y_pred = rand_classifier.predict(X_test)

# Making the Confusion Matrix
from sklearn.metrics import accuracy_score
ac1 = accuracy_score(y_test, y_pred)

# Making the Confusion Matrix
from sklearn.metrics import confusion_matrix
cm1 = confusion_matrix(y_test, y_pred)
from sklearn.metrics import f1_score
f1 = f1_score(y_test, y_pred, average='micro')
```

```
print("Accuracy:", ac1)
print("Confusion Matrix:")
print(cm1)
print("F1 Score:", f1)
```



```
Accuracy: 0.20152883947185546
Confusion Matrix:
[[  4 276   0   0   0]
 [  8 286   0   0   0]
 [  7 270   0   0   0]
 [ 10 266   0   0   0]
 [  6 306   0   0   0]]
F1 Score: 0.20152883947185546
```

Fig. 4.5 Random Forest Algorithm Output

Chapter No. 5
Advantages, Limitation & Applications

In this chapter, we will discuss about the various advantages, limitations and applications of K-NN algorithm and Random Forest Algorithm which can be used to evolve more into the mechanical engineering industry applications.

5.1 Advantages of K-NN Algorithm

The k-Nearest Neighbors (k-NN) algorithm has several advantages, making it a popular choice in machine learning and data analysis. Here are some of the key advantages of the k-NN algorithm:

1 **Simplicity**: k-NN is easy to understand and implement, making it a straightforward algorithm for both beginners and experts. It doesn't require complex mathematical or parametric assumptions.

2 **No Training Phase:** Unlike many other machine learning algorithms, k-NN does not have a training phase. It stores the entire dataset and makes predictions based on the proximity of data points, which can be advantageous in situations where data is constantly changing.

3 **Versatility:** k-NN can be applied to various types of data, including classification and regression tasks. It can handle both numerical and categorical data.

4 **Interpretability:** Predictions made by k-NN are interpretable. The outcome is based on the majority class or average of the k-nearest neighbors, which can be easily understood and explained.

5 **Robust to Outliers**: k-NN is robust to noisy data and outliers because it relies on a voting mechanism from multiple neighbors. Outliers are less likely to significantly impact the prediction.

6 **Non-parametric:** k-NN is a non-parametric algorithm, meaning it doesn't assume any specific underlying data distribution. This makes it suitable for data that doesn't conform to traditional statistical assumptions.

5.2 Limitations of K-NN Algorithm

While the k-Nearest Neighbors (k-NN) algorithm has several advantages, it also has some limitations and challenges that should be taken into consideration:

1 **Sensitivity to Feature Scaling**: k-NN is sensitive to the scale of features. Features with larger scales can dominate the distance calculations, leading to inaccurate results. Standardization or normalization of data is often required to address this issue.

2 **Computational Complexity:** As the dataset grows, the computational cost of k-NN increases significantly. Calculating distances between data points for every prediction can be slow for large datasets, making it less practical in high-dimensional spaces.

3 **Curse of Dimensionality:** In high-dimensional feature spaces, the concept of "nearest" neighbors becomes less meaningful, and the data points tend to be equidistant from each other. This can lead to a reduction in the effectiveness of k-NN in high-dimensional data.

4 **Optimal Choice of 'k':** Selecting the right value for 'k' (the number of neighbors) can be challenging. A small 'k' may make the model sensitive to noise and outliers, while a large 'k' can lead to oversmoothing and a loss of local patterns.

5 **Imbalanced Datasets:** In cases where the classes are imbalanced, i.e., one class has significantly more instances than the others, k-NN may favor the majority class, leading to biased predictions.

6 **Slow Prediction:** Since k-NN doesn't have a model-building phase and makes predictions based on proximity calculations, it can be slower for real-time or time-sensitive applications, especially when 'k' is large or the dataset is extensive.

In summary, while k-NN is a straightforward and versatile algorithm, it may not always be the best choice, especially in cases of high-dimensional data or when computational efficiency is a primary concern. Careful consideration of its limitations and appropriate preprocessing steps are necessary to make effective use of the algorithm.

5.3 Applications of K-NN Algorithm

1 **Anomaly Detection:** k-NN can be used to identify anomalies or outliers in data by finding data points that have few or no close neighbors. It is used in fraud detection, network

intrusion detection, and quality control in manufacturing. Medical Diagnosis: k-NN is applied in medical diagnosis for tasks like disease classification, determining patient risk factors, and predicting outcomes based on the similarity of a patient's medical history to that of others.

2 **Environmental Monitoring:** k-NN is used in environmental science for tasks like air quality prediction, weather forecasting, and ecological modeling, where data from neighboring locations and times are used to make predictions.

3 **Credit Scoring:** In the financial industry, k-NN can be employed to assess the creditworthiness of individuals by comparing their financial attributes to those of neighbors with known credit histories.

4 **Geographical Information Systems (GIS):** k-NN can help in spatial analysis, such as identifying the closest facilities, neighbors, or points of interest in a geographical area. It is used in route planning, location-based services, and urban planning.

5 **Intrusion Detection in Networks:** k-NN is used to detect network intrusions by comparing network traffic patterns to those of known attacks or anomalies.

6 **Robotics and Autonomous Vehicles:** In robotics and autonomous navigation systems, k-NN can help robots and vehicles make decisions based on the proximity of objects or obstacles in their environment.

These are just a few examples of the wide range of applications for the k-NN algorithm. Its flexibility and simplicity make it a valuable tool in many machine learning and data analysis scenarios, particularly when interpretability and ease of use are crucial.

5.4 Advantages of Random Forest Algorithm

Random Forest and k-Nearest Neighbors (k-NN) are two different machine learning algorithms, each with its own set of advantages and disadvantages. While the choice between them depends on the specific problem and data at hand, here are some advantages of the Random Forest algorithm over k-NN:

1 **Ensemble Method:** Random Forest is an ensemble learning method, which means it combines the predictions of multiple decision trees to produce a more robust and accurate

result. This often leads to improved performance compared to a single k-NN model, which relies on the nearest neighbors' votes.

2 **Less Sensitive to Noisy Data:** Random Forest is generally less sensitive to noisy data and outliers. Each decision tree in the ensemble can be affected by outliers, but the majority voting or averaging across the trees tends to smooth out these effects.

3 **Effective in High-Dimensional Spaces:** Random Forest can handle high-dimensional data more effectively than k-NN. k-NN's performance tends to degrade in high-dimensional spaces due to the "curse of dimensionality," while Random Forest remains robust.

4 **Efficiency:** Random Forest is often more computationally efficient than k-NN, especially for large datasets. The decision trees in the ensemble can be constructed in parallel, and the model can handle substantial amounts of data without significant slowdown.

5 **No Need for Feature Scaling:** Random Forest is not sensitive to feature scaling, making it more convenient when dealing with datasets with features on different scales. In contrast, k-NN performance can be affected by the scale of features.

6 **Out-of-Bag (OOB) Error Estimation:** Random Forest can estimate the model's performance on unseen data through the OOB error. This is a built-in cross-validation technique that does not require a separate validation set, making model evaluation more straightforward.

5.5 Disadvantages of Random Forest Algorithm

While the Random Forest algorithm offers many advantages, it also has some disadvantages and limitations that should be considered when choosing it for a specific machine learning task:

1 **Black Box Model:** Random Forests are ensemble models composed of multiple decision trees. While they provide high predictive accuracy, the ensemble nature can make them less interpretable compared to a single decision tree. Understanding the underlying logic of the model can be challenging.

2 **Resource-Intensive:** Building a Random Forest with a large number of decision trees can be computationally expensive and memory-intensive, especially for large datasets. Training

multiple trees in parallel can mitigate this issue, but it still requires substantial computational resources.

3 **Overfitting:** Although Random Forests are less prone to overfitting than individual decision trees, they can still overfit if the number of trees in the ensemble is too high or the data is noisy. It's important to tune hyperparameters such as the maximum depth of trees and the number of trees to prevent overfitting.

4 **Biased Toward Majority Class:** In imbalanced datasets, where one class significantly outnumbers the others, Random Forests can be biased towards the majority class. While techniques like balancing the class distribution can be used, it's still a limitation to be aware of.

5 **Lack of Good Probability Estimates:** Random Forests do not provide well-calibrated probability estimates. The predicted class probabilities may not be accurate, which can be a limitation in applications where probability estimates are crucial, such as in medical diagnosis or fraud detection.

6 **Less Effective in High-Dimensional Data:** While Random Forests are robust to many types of data, they can become less effective in high-dimensional feature spaces. In extremely high-dimensional data, finding meaningful splits in each tree becomes more challenging.

5.6 Applications of Random Forest Algorithm

1 **Image and Object Recognition:** Random Forest is used for image classification and object recognition tasks, such as identifying objects in images, classifying handwritten digits, and recognizing faces in photos.

2 **Customer Churn Prediction:** Random Forest can help businesses predict customer churn by analyzing historical customer behavior and identifying factors that lead to attrition.

3 **Biomedical Applications:** It is used in medical and biological research for tasks like disease diagnosis, drug discovery, and identifying genetic markers associated with diseases.

4 **Text and Natural Language Processing (NLP):** Random Forest is applied in text classification tasks, such as sentiment analysis, document categorization, and spam email detection.

5 **Credit Scoring:** Financial institutions use Random Forest to assess the creditworthiness of individuals by analyzing their financial history and demographics.

6 **Genomic Data Analysis:** It is used in genomics to classify genetic variations, identify biomarkers, and predict disease susceptibility.

These applications demonstrate the flexibility and effectiveness of Random Forest across a wide range of fields, making it a popular choice for many machine learning and data analysis tasks.

Chapter No. 6
Future Scope and Conclusion

6.1 Future Scope

The future scope of using k-Nearest Neighbors (k-NN) and Random Forest algorithms together lies in harnessing the strengths of both approaches to address specific challenges and create more robust machine learning systems. Here are some potential areas where combining these algorithms could be advantageous:

1 **Hybrid Ensemble Models**: Creating hybrid ensemble models that integrate the local learning capabilities of k-NN with the ensemble power of Random Forest. This can lead to improved predictive accuracy and generalization.

2 **Feature Engineering:** Combining k-NN for feature selection or feature engineering with Random Forest for classification or regression. The k-NN algorithm can be used to identify the most relevant features, which can then be used as input to Random Forest.

3 **Outlier Detection:** Using k-NN for outlier detection and then feeding the identified outliers to a Random Forest classifier to make decisions based on the majority class. This can enhance the robustness of the system.

4 **Meta-Learning**: Utilizing k-NN to perform meta-learning or hyperparameter tuning for Random Forest. k-NN can help in selecting optimal parameters for Random Forest models based on the characteristics of the data.

5 **Local vs. Global Analysis:** Leveraging k-NN for local, fine-grained analysis and using Random Forest for global, coarse-grained analysis. This approach can provide a comprehensive understanding of complex datasets.

6 **Transfer Learning:** Combining both algorithms in transfer learning scenarios where knowledge from one domain is adapted to another. k-NN can be used to identify similar instances in the target domain, and Random Forest can incorporate this knowledge for prediction.

The future scope of using k-NN and Random Forest algorithms together is particularly promising for complex machine learning tasks where a combination of local and global learning is needed to achieve the best results. The specific application and data characteristics will determine the most effective way to blend these two algorithms.

6.2 Conclusion

This study conducted a comparison of the performance of two machine learning methods, namely KNN and Random Forests, for the purpose of predicting traffic flow. The results have revealed that when it comes to long-term predictions (more than one hour into the future), simplistic approaches like using historical averages surprisingly prove to be quite effective. On the other hand, for shorter-term predictions (less than 1 hour), employing current traffic measurements as a straightforward predictive method yields favourable results. This outcome is as expected, as the current traffic situation has a more immediate impact on traffic soon compared to its influence on traffic several hours or days later.

By utilizing fewer complex models, we can more easily identify optimal model parameters, enhance computational speed, and ensure greater model comprehensibility and maintainability. It is worth noting that the primary drawback of the models discussed in this study is their incapacity to predict unusual traffic events. While common traffic patterns offer valuable insights for commuters in unfamiliar areas, local commuters who are familiar with regular traffic conditions find information about unusual traffic events most beneficial. This limitation stems from the fact that current models solely rely on historical traffic data. Since some unusual traffic events are linked to other factors like nearby traffic accidents, adverse weather conditions, holidays, and more, we believe that incorporating additional data sources into the model could significantly enhance the prediction of such events.

As a result, future involves gathering a variety of high-quality traffic-related data sources, including traffic alerts, status updates for special days, information about significant social events, and more. We intend to integrate these additional data sources with the existing loop counters data to develop more robust traffic prediction models.

REFERENCES

[1] A. Moldagulova and R. B. Sulaiman, "Using KNN algorithm for classification of textual documents," 2017 8th International Conference on Information Technology (ICIT), Amman, Jordan, 2017, pp. 665-671, doi: 10.1109/ICITECH.2017.8079924.

[2] Yong, Zhou, Lishi Youwen, and Xia Shixiong. "An improved KNN text classification algorithm based on clustering." Journal of computers 4.3 (2009): 230-237.

[3] Soucy, Pascal, and Guy W. Mineau. "A simple KNN algorithm for text categorization." Proceedings 2001 IEEE international conference on data mining. IEEE, 2001.

[4] Akar, Özlem, and Oguz Güngör. "Classification of multispectral images using Random Forest algorithm." Journal of Geodesy and Geoinformation 1.2 (2012): 105-112.

[5] Kulkarni, Arun D., and Barrett Lowe. "Random forest algorithm for land cover classification." (2016).

[6] Paul, Angshuman, et al. "Improved random forest for classification." IEEE Transactions on Image Processing 27.8 (2018): 4012-4024.

[7] Zhang, Lun, et al. "An improved k-nearest neighbor model for short-term traffic flow prediction." Procedia-Social and Behavioral Sciences 96 (2013): 653-662.

[8] Xu, Dongwei, et al. "Real-time road traffic state prediction based on kernel-KNN." Transportmetrica A: Transport Science 16.1 (2020): 104-118.