# VISVESVARAYA TECHNOLOGICAL UNIVERSITY

**"Jnana Sangama", Belagavi - 590 018, Karnataka.**

**21INT68 -Innovation/Entrepreneurship /Societal Internship**

# "Traffic Prediction "

**Submitted in partial fulfilment of the requirements for the award of the degree of Bachelor of Engineering**
**In**
**Computer Science & Engineering**


**Submitted by**
**1BI21CS100 Pratibha Kumari**

**Under the Guidance of**
**Prof. Pooja P**
**Assistant Professor**
**Dept. of CSE, BIT**




**DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING**

## BANGALORE INSTITUTE OF TECHNOLOGY

K. R. Road, V. V. Puram, Bengaluru - 560 004


**2023-2024**

# VISVESVARAYA TECHNOLOGICAL UNIVERSITY

**"Jnana Sangama",** Belagavi-590 018, Karnataka

## BANGALORE INSTITUTE OF TECHNOLOGY

Bengaluru-560 004

**DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING**

## *Certificate*

Certified that the **21INT68-Innovation/Entrepreneurship/Societal Internship** (21INT68) work entitled "Data Analysis using Python" carried out by Pratibha Kumari, USN 1BI21CS100 a bonafide student of Bangalore Institute of Technology in partial fulfillment for the award of Bachelor of Engineering in Computer Science & Engineering of the **Visvesvaraya Technological University, Belagavi** during the academic year 2023-2024. It is certified that all corrections/suggestions indicated for Internal Assessment have been incorporated in the report deposited in the departmental library.

The Internship report has been approved as it satisfies the academic requirements in respect of Internship work prescribed for the said degree.

**Guide**
**Prof. Pooja P**
Assistant Professor
Department of CSE, BIT

**Dr. J. Girija**
Professor and Head
Department of CSE, BIT

Ref: iObrain/intern/100-01/24 Dated: 24-01-2024

# CERTIFICATE OF INTERNSHIP

This is to certify that "PRATIBHA KUMARI" USN: 1BI21CS100,

5th Semester from dept. of "**Computer Science and Engineering** from

"**BANGALORE INSTITUTE OF TECHNOLOGY**"

has successfully completed an **Internship Programme** on

"**DATA Analysis using Python**" along with the live projects

for 4 weeks from 25-10-23 to 23-11-23.

Resource Person: Dr. Nagarajan Srinivasan

We wish the student every success in career and future endeavors.

Ramakrishna K S

CEO, iObrain

Bangalore 560 085

# ACKNOWLEDGEMENT

The satisfaction and euphoria that accompanies the successful completion of any task would be incomplete without complementing those who made it possible and whose guidance and encouragement made my efforts successful. So, my sincere thanks to all those who have supported me in completing this Internship successfully.

My sincere thanks to **Dr. M. U. Aswath**, Principal, BIT and **Dr. J. Girija**, HOD, Department of CSE, BIT for their encouragement, support and guidance to the student community in all fields of education. I am grateful to our institution for providing us a congenial atmosphere to carry out the Internship successfully.

I would not forget to remember **Dr. Bhanushree K J,** Associate Professor **21INT68 - Innovation/Entrepreneurship/Societal** Internship Coordinator, Department of CSE, BIT, for her encouragement and more over for her timely support and guidance till the completion of the Internship.

I avail this opportunity to express my profound sense of deep gratitude to my esteemed guide **Prof. Pooja P**, Associate Professor, Department of CSE, BIT, for his/her moral support, encouragement and valuable suggestions throughout the Internship.

I extend my sincere thanks to all the department faculty members and non-teaching staff for supporting me directly or indirectly in the completion of this Internship.

**NAME: PRATIBHA KUMARI**

**USN: 1BI21CS100**

# TABLE OF CONTENTS


| Chapter 1 - Introduction | | | Page No. |
|---|---|---|---|
| 1.1 | Overview | | 1 |
| 1.2 | Objective | | 1 |
| 1.3 | Purpose, Scope and applicability | | 2-4 |
| | 1.3.1 | Purpose | 2 |
| | 1.3.2 | Scope | 3 |
| | 1.3.3 | Applicability | 3 |
| Chapter 2 - Problem Statement | | | 5 |
| Chapter 3 - System Architecture | | | 6 |
| Chapter 4 - Tools/Technologies | | | 10 |
| Chapter 5 - Implementation | | | 12-17 |
| Chapter 6 - Results | | | 18-19 |
| Chapter 7 - Reflection Notes | | | 20-21 |
| Chapter 8 - References | | | 22 |

# Chapter 1

# INTRODUCTION

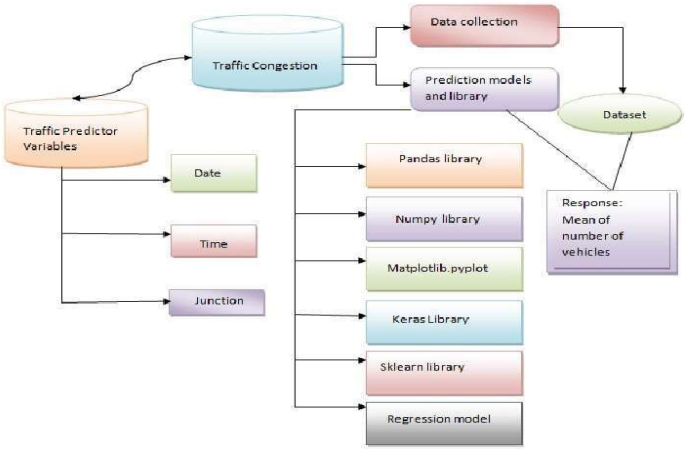
## 1.1 Overview

Machine Learning (ML) is one of the most important and popular emerging branches these days as it are a part of Artificial Intelligence (AI). In recent times, machine learning becomes an essential and upcoming research area for transportation engineering, especially in traffic prediction. Traffic congestion affects the country's economy directly or indirectly by its means. Traffic congestion also takes people's valuable time, cost of fuel every single day. As traffic congestion is a major problem for all classes in society, there has to be a small-scale traffic prediction for the people's sake of living their lives without frustration or tension. For ensuring the country's economic growth, the road user's ease is required in the first place. This is possible only when the traffic flow is smooth. To deal with this, Traffic prediction is needed so that we can estimate or predict the future traffic to some extent.

## 1.2 Objective

To build and evaluate a classifier that can accurately predict traffic levels in urban areas, using vehicle counts in 15-minute intervals as the main input feature, and compare its performance with existing methods.This project aims to develop a machine learning model using the Random Forest Classifier algorithm to predict traffic situations based on various features such as car count, bike count, bus count, and truck count. The dataset used for this project contains information about traffic situations on different days along with the counts of different vehicle types.

**In traffic congestion forecasting there are data collection and prediction model:**



## 1.3 Purpose, Scope and applicability

### 1.3.1 Purpose

Traffic prediction using machine learning serves several important purposes:

1. Enhanced Traffic Management: Machine learning models can analyze historical traffic data and identify patterns indicative of congestion, accidents, or other disruptions. By predicting future traffic situations, authorities can implement proactive measures such as adjusting signal timings, deploying additional resources, or rerouting traffic to alleviate congestion and improve overall traffic flow.

2. Optimized Transportation Systems: Predictive models can assist in optimizing transportation systems by forecasting traffic demand and identifying peak hours of traffic congestion. This information can be used to schedule public transportation services more efficiently, plan infrastructure upgrades, and prioritize maintenance activities to minimize disruptions.

3. Improved Urban Planning: By analyzing traffic patterns and predicting future traffic conditions, city planners can make informed decisions regarding the development of road networks, public transit routes, and urban infrastructure. This helps in designing cities that are more pedestrian-friendly, bike-friendly, and conducive to sustainable modes of transportation.

4. Reduced Environmental Impact: Traffic prediction models can contribute to reducing the environmental impact of transportation by optimizing traffic flow and reducing vehicle idling time. By minimizing congestion and improving traffic efficiency, emissions of greenhouse gases and pollutants from vehicles can be reduced, leading to improved air quality and mitigating the effects of climate change.

5. Enhanced Safety: Early detection of traffic incidents and hazardous conditions can help in improving road safety by enabling prompt emergency response and traffic management interventions. Predictive models can identify high-risk areas and times, allowing authorities to implement targeted safety measures and reduce the likelihood of accidents and collisions.

Overall, the purpose of traffic prediction using machine learning is to enhance the efficiency, safety, and sustainability of transportation systems, ultimately improving the quality of life for residents and commuters in urban areas.

### 1.3.2 Scope

1. Traffic Flow Optimization: Utilizing historical data to forecast congestion, accidents, and road closures, enabling proactive measures to optimize traffic flow and minimize delays.

2. Transportation Infrastructure Planning: Predicting future traffic demands based on historical trends and population growth, facilitating informed decisions on the expansion, maintenance, and development of transportation networks.

3. Public Transit Efficiency: Forecasting ridership demand to optimize public transit routes, schedules, and capacity utilization, leading to improved service reliability and customer satisfaction.

4. Traffic Safety Enhancement: Analyzing historical data to identify factors contributing to accidents and collisions, allowing for targeted interventions and safety measures to reduce road accidents and injuries.

5. Environmental Impact Mitigation: Predicting traffic patterns to minimize emissions and environmental pollution from vehicles, contributing to improved air quality and reduced environmental degradation.

6. Urban Planning and Development: Integrating traffic prediction models into urban planning processes to guide land use decisions, transportation infrastructure placement, and the creation of sustainable, accessible urban environments.

### 1.3.3 Application

Applications of traffic prediction using machine learning:

1. Urban Traffic Management: Predicting traffic flow and congestion patterns helps urban planners and traffic management authorities optimize traffic signal timings, lane configurations, and road network design to improve overall traffic efficiency and reduce congestion.

2. Smart Transportation Systems: Machine learning models can be integrated into intelligent transportation systems (ITS) to provide real-time traffic predictions, enabling dynamic route guidance and adaptive traffic control strategies for vehicles and public transit.

3. Navigation and Routing Apps: Incorporating traffic prediction algorithms into navigation apps allows users to receive accurate and up-to-date information on traffic conditions along their chosen routes, helping them avoid congested areas and choose the fastest travel options.

4. Logistics and Supply Chain Management: Predicting traffic conditions and delivery times enables logistics companies to optimize their delivery routes, reduce transportation costs, and improve customer satisfaction by ensuring timely deliveries.

5. Emergency Response Planning: Anticipating traffic congestion and identifying optimal routes for emergency vehicles based on predicted traffic conditions can improve response times during emergencies such as accidents, natural disasters, or medical emergencies.

6. Urban Development and Policy Making: Traffic prediction models provide valuable insights for urban planners and policymakers to make data-driven decisions on infrastructure investments, transportation policies, and land use planning to support sustainable urban development and reduce traffic-related emissions.

# Chapter 2

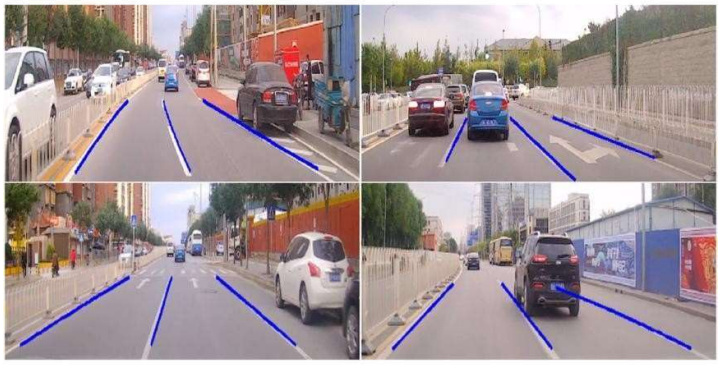
## Problem Statement

Build a classifier to predict traffic levels in urban areas, using vehicle counts in 15-minute intervals. The model will use vehicle counts, historical patterns, and external factors to forecast congestion and inform urban transport decisions. The tool will help authorities, planners, and commuters to improve traffic flow, reduce congestion, and increase transport efficiency and sustainability.

The ultimate aim of this classifier is to provide accurate, real-time forecasts of congestion levels. With this information, urban transport decisions can be made more effectively. Authorities can implement strategies to improve traffic flow, such as adjusting traffic signal timings or implementing congestion charges. Planners can design more efficient transport networks, taking into account predicted traffic hotspots. And commuters can make informed decisions about when and how to travel, helping to reduce overall congestion.

By achieving these goals, the classifier will contribute to increased transport efficiency and sustainability. It will help to reduce the time and fuel wasted on congested roads, leading to lower carbon emissions. And by improving the predictability of travel times, it can also enhance the quality of life for urban residents.

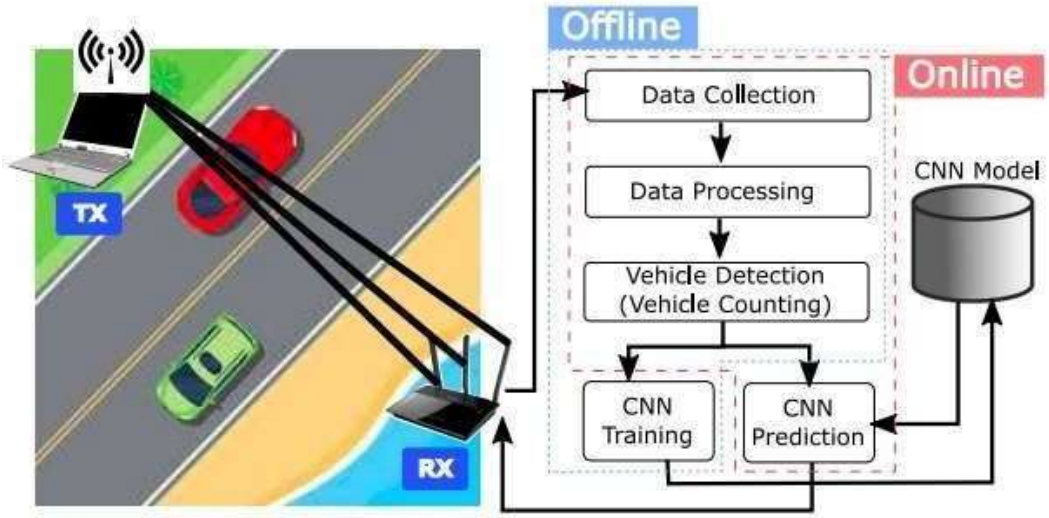
*a*



*b*

# Chapter 3

## System Architecture




The flow chart explains a process of vehicle detection and counting using a Random Forest Classifier (RFC) model. Here are the steps:

**Data Collection:** A transmitter (TX) sends signals that are received by a receiver (RX), likely detecting vehicles on the road. This data is collected and stored for offline processing.

**Data Processing:** The collected data is processed to extract relevant features, such as the signal strength, frequency, and duration, that can be used for vehicle detection.

Vehicle Detection (Vehicle Counting): A vehicle detection algorithm is applied to the processed data to identify and count the vehicles present on the road. This step generates labels for the data that can be used for training the RFC model.

**RFC Training:** A RFC model is trained with the labeled data, learning to recognize the patterns and features that indicate the presence and number of vehicles on the road. A RFC model is an ensemble learning method that operates by constructing multiple decision trees during training time and outputting the class that is the mode of classes (classification) or mean prediction (regression) of individual trees. It helps in improving prediction accuracy and controlling over-fitting.

**RFC Prediction:** The trained RFC model is used to make predictions on new data in real-time, using the TX/RX system as the input. The model outputs the vehicle count for each data point, indicating the traffic level on the road.

**Data Splitting:**Preprocessed data, including vehicle counts, historical patterns, and external factors, is split into training and test datasets. Typically, 80% of the data is used for training and 20% for testing.

**Load Train Data:**The training data, comprising features such as vehicle counts, historical patterns, and external factors, is loaded into memory for model training.

**Train Model:**Using the training data, a machine learning model, such as a Random Forest classifier, is trained to predict traffic levels and congestion in urban areas. The model is optimized to leverage historical patterns and external factors to improve prediction accuracy.

**Confusion Matrix:**After training, a confusion matrix is generated using the model's predictions on the test dataset. This matrix helps evaluate the model's performance by quantifying True Positives, True Negatives, False Positives, and False Negatives.

**Export Trained Model:**The trained model, built using the Random Forest algorithm, is exported and saved for further use, such as testing or deployment in production environments.

**Load Trained Model:**The exported trained model is loaded from storage to ensure consistency between training and testing phases.

**Load Test Data:**Test data, containing features similar to those used for training, is loaded into memory to evaluate the model's performance.

**Predictions:**The loaded test data is provided as input to the trained model to predict traffic levels and congestion in urban areas. The model analyzes the features and generates predictions based on learned patterns and external factors.

**Result Display:**The results of the predictions, including predicted traffic levels and congestion statuses, are displayed on a user interface. This interface may visualize predictions, compare them with actual data, and provide insights for decision-making by urban transport authorities, planners, and commuters.

## Methodology

## 1)Classification

The process of recognizing, understanding, and grouping ideas and objects into preset categories or “sub-populations.” Using pre-categorized training datasets, machine learning programs use a variety of algorithms to classify future datasets into categories. Classification algorithms in machine learning use input training data to predict the likelihood that subsequent data will fall into one of the predetermined categories.

## 2) Decision TreeClassifier

A Decision Tree Classifier is a supervised machine learning algorithm used primarily for classification tasks. It works by recursively partitioning the feature space into smaller regions, guided by the values of input features, to make decisions about the target variable.

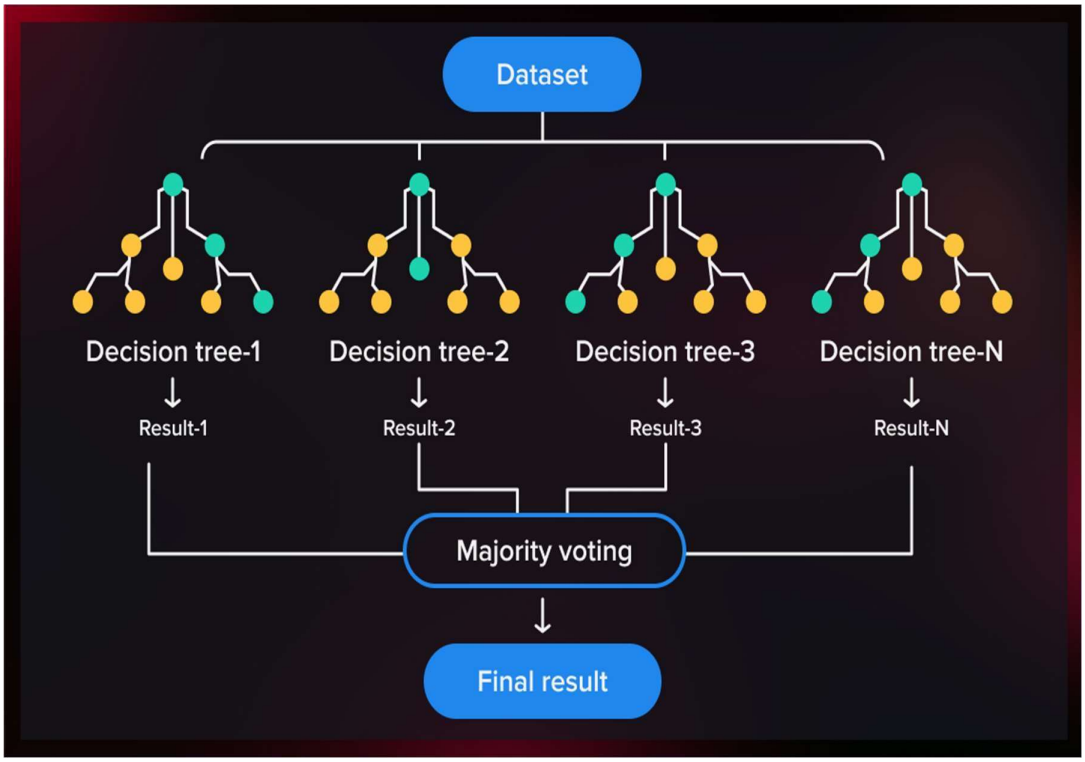
## 3) Random forest

Random forests or random decision forests is an ensemble learning method for classification, regression and other tasks that operates by constructing a multitude of decision trees at training time.

For classification tasks, the output of the random forest is the class selected by most trees. For regression tasks, the mean or average prediction of the individual trees is returned.

Instead of relying on one decision tree, the random forest takes the prediction from each tree and based on the majority votes of predictions, and it predicts the final output

The random forest algorithm solves overfitting to a great extent



**Working of algorithm**

The following steps explain the working Random Forest Algorithm:

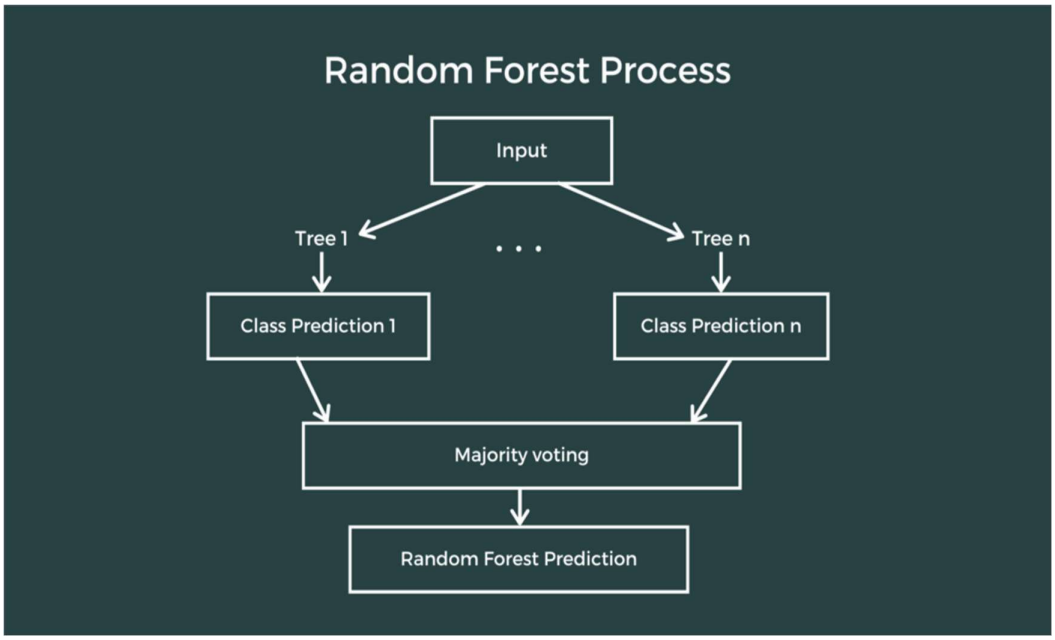
• Step 1: Select random samples from a given data or training set.

• Step 2: This algorithm will construct a decision tree for every training data.

• Step 3: Voting will take place by averaging the decision tree

.• Step 4: Finally, select the most voted prediction result as the final prediction result.

This combination of multiple models is called Ensemble.



# Chapter 4

# TOOLS/TECHNOLOGIES

**NumPy**: NumPy is a fundamental package for scientific computing in Python. It provides support for multi-dimensional arrays and matrices, along with a collection of mathematical functions to operate on these arrays efficiently. NumPy is widely used in various fields such as machine learning, data analysis, engineering, and scientific research due to its speed and versatility. It forms the basis for many other Python libraries in the scientific computing ecosystem.

**Matplotlib**: It is a comprehensive Python library used for creating static, animated, and interactive visualizations in Python. It provides a wide variety of plotting functions and customization options to create publication-quality figures and visualizations. Matplotlib is highly customizable, allowing users to create plots ranging from simple line plots to complex 3D plots.

The library is extensively used for data exploration, analysis, and presentation in fields such as data science, engineering, finance, and academia. It integrates well with other Python libraries such as NumPy and Pandas, making it a versatile tool for data visualization and analysis in Python ecosystem.

**Seabron**: Seaborn is a Python data visualization library based on Matplotlib. It provides a high-level interface for creating attractive and informative statistical graphics. Seaborn simplifies the process of creating complex visualizations by providing convenient functions for tasks like grouping data, summarizing distributions, and visualizing relationships between variables.

Seaborn is particularly useful for creating statistical plots such as scatter plots, bar plots, box plots, violin plots, and heatmaps. It also offers support for visualizing linear regression models, time series data, and categorical data.

**Scikit-learn** is a widely used open-source machine learning library for Python. It provides simple and efficient tools for data mining and data analysis, built on NumPy, SciPy, and Matplotlib. Scikit-learn offers a consistent interface for various machine learning algorithms, including supervised and unsupervised learning techniques such as classification, regression, clustering, dimensionality reduction, and model selection.

One of the key strengths of scikit-learn is its ease of use and accessibility, making it suitable for both beginners and experienced practitioners. It includes well-documented APIs, clear documentation, and a rich set of tutorials and examples.

Scikit-learn also supports various features such as data preprocessing, feature extraction, model evaluation, and model optimization. It's widely used in academia, industry, and research for tasks ranging from simple data analysis to complex machine learning projects.

**Pandas** is a powerful open-source data analysis and manipulation library for Python. It provides easy-to-use data structures like DataFrame and Series, which allow users to work with structured data efficiently. Pandas is widely used for tasks such as data cleaning, transformation, exploration, and analysis in various domains including data science, finance, and research.

**Google Colab**: It is the short form for Google Colaboratory, is a free cloud-based platform provided by Google that allows users to write and execute Python code in a collaborative environment. It's built on top of Jupyter Notebooks and provides access to GPU and TPU hardware for machine learning tasks. Users can write code, execute it, and store and share their work via Google Drive. It's widely used for data analysis, machine learning, and education purposes due to its accessibility and integration with other Google services

# Chapter 5

# IMPLEMENTATION

## 5.1 Code Requirements:

1. A stable internet connection is required to access the dataset from the provided URL.
2. The Python libraries `pandas`, `numpy`, `matplotlib`, `sklearn`, `joblib`, and `seaborn` need to be installed in your working environment.
3. A Python environment where you can run the code, such as Google Colab or Jupyter notebook embedded with Visual Studio Code.

## 5.2 Concepts Used:

1. **Data Collection:** The data is collected from a Google Spreadsheet using the `pandas` function `read_html`.
2. **Data Cleaning:** Unwanted columns are dropped from the dataframe. Missing values in numeric columns are filled with their mean, and missing values in non-numeric columns are filled with their mode (most frequent value).
3. **Data Preprocessing:** The feature columns and target column are defined. The data is then split into training and test sets.
4. **Model Building:** A Random Forest Classifier is initialized and trained on the training data.
5. **Model Evaluation:** Predictions are made on the test set, and the model's performance is evaluated using a classification report.
6. **Data Visualization:** The importance of each feature is visualized using a bar plot.

## 5.3 Code Explanation

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
from sklearn.model_selection import train_test_split
from sklearn.ensemble import RandomForestClassifier
from sklearn.metrics import classification_report
from sklearn.model_selection import GridSearchCV
import joblib
import seaborn as sns
```

```
# Load the data
tables =
pd.read_html(r'https://docs.google.com/spreadsheets/d/1XX2o_Xe5esnT8SEi45hLBM73
LDI6Jz2yFy-Yu2WGdMk/edit#gid=474392997.html',skiprows=1)
df = tables[0]

# Drop the unwanted columns
df = df.drop(columns=['Unnamed: 10', 'Unnamed: 11'])

# Display the shape of the data and the first few rows
print("Data shape:", df.shape)
print("Sample data:\n", df.head())

# Check for missing values
print("\nMissing values:\n", df.isnull().sum())

# Fill missing values in numeric columns with their mean
num_cols = ['CarCount', 'BikeCount', 'BusCount', 'TruckCount', 'Total']
df[num_cols] = df[num_cols].fillna(df[num_cols].mean())

# Fill missing values in non-numeric columns with their mode (most frequent value)
non_num_cols = ['Date', 'Day of the week', 'Traffic Situation']
df[non_num_cols] = df[non_num_cols].fillna(df[non_num_cols].mode().iloc[0])

# Define your feature columns and target column
feature_cols = ['CarCount', 'BikeCount', 'BusCount', 'TruckCount', 'Total']
target_col = 'Traffic Situation'

# Split the data into training and test sets
X_train, X_test, y_train, y_test = train_test_split(df[feature_cols], df[target_col],
test_size=0.2, random_state=42)

# Initialize the Random Forest Classifier
```

```
clf = RandomForestClassifier(n_estimators=100, random_state=42)

# Train the model
clf.fit(X_train, y_train)

# Make predictions on the test set
y_pred = clf.predict(X_test)

# Evaluate the model
print(classification_report(y_test, y_pred))

# Display the predicted traffic situation for the test set
pred_df = pd.DataFrame({'Actual Traffic Situation': y_test, 'Predicted Traffic Situation': y_pred})
print("\nPredicted Traffic Situation for Test Set:\n", pred_df)

# Feature importance analysis
importances = clf.feature_importances_
indices = np.argsort(importances)[::-1]
plt.figure(figsize=(12,6)) # Increase the size of the figure
plt.title("Feature importances")

# Use feature names instead of numbers
feature_names = [feature_cols[i] for i in indices]

plt.bar(range(X_train.shape[1]), importances[indices], color="r", align="center")
plt.xticks(range(X_train.shape[1]), feature_names) # Use feature names here
plt.xlim([-1, X_train.shape[1]])
plt.show()

# Hyperparameter tuning
param_grid = {
    'n_estimators': [50, 100, 200],
    'max_features': ['auto', 'sqrt', 'log2'],
```

```
    'max_depth' : [4,5,6,7,8],
    'criterion' :['gini', 'entropy']
}
CV_rfc = GridSearchCV(estimator=clf, param_grid=param_grid, cv= 5)
CV_rfc.fit(X_train, y_train)
print(CV_rfc.best_params_)

# Save the model
joblib.dump(clf, 'random_forest_model.pkl')
print("Model saved as 'random_forest_model.pkl'")

# Plotting the distribution of traffic situations
plt.figure(figsize=(8,6))
sns.countplot(x=target_col, data=df)
plt.title('Distribution of Traffic Situations')
plt.show()

# Pairplot to visualize the relationships between different features
sns.pairplot(df, hue=target_col)
plt.show()
```

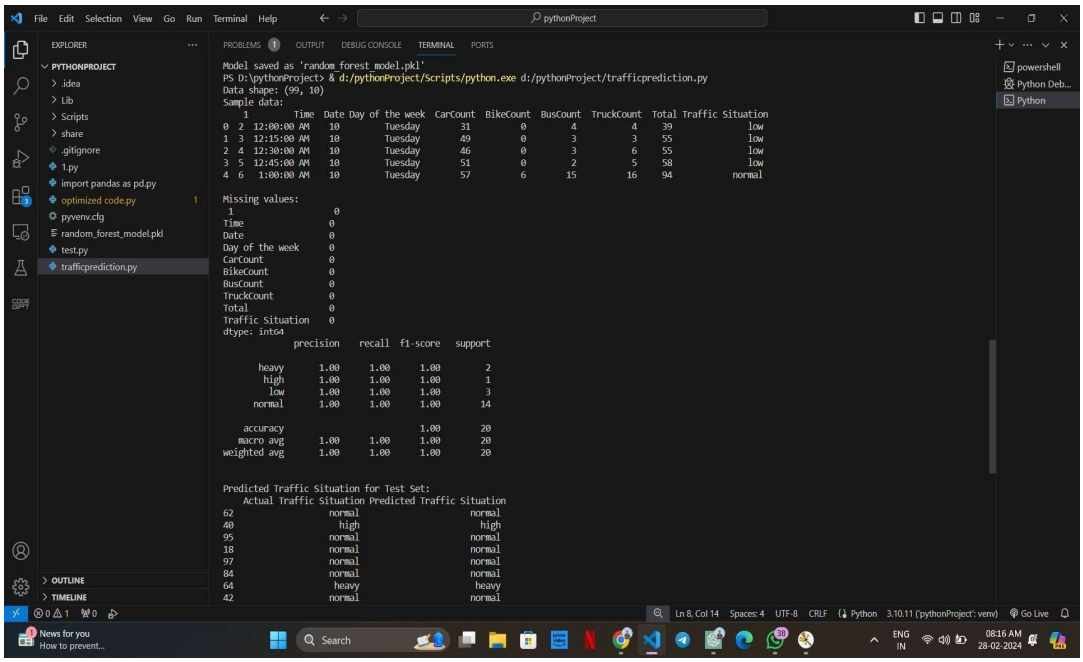
This code performs traffic prediction using a Random Forest Classifier. It first collects and cleans the data, then trains the model, makes predictions, evaluates the model's performance, and finally visualizes the importance of each feature. The model's performance is evaluated using a classification report, which provides precision, recall, f1-score, and support for each class. The feature importance analysis helps in understanding which features contribute the most to the model's predictions. The bar plot visualizes the importance of each feature, with higher bars indicating more important features. The code also handles missing values by filling them with appropriate values (mean for numeric columns and mode for non-numeric columns). This ensures that the model can be trained on a complete dataset. The model is then saved using `joblib` forfuture use. The saved model can be loaded later to make predictions on new data. The code also includes some data exploration and visualization steps to understand the data better. For example, it prints the shape of the data and a few sample rows, and checks for

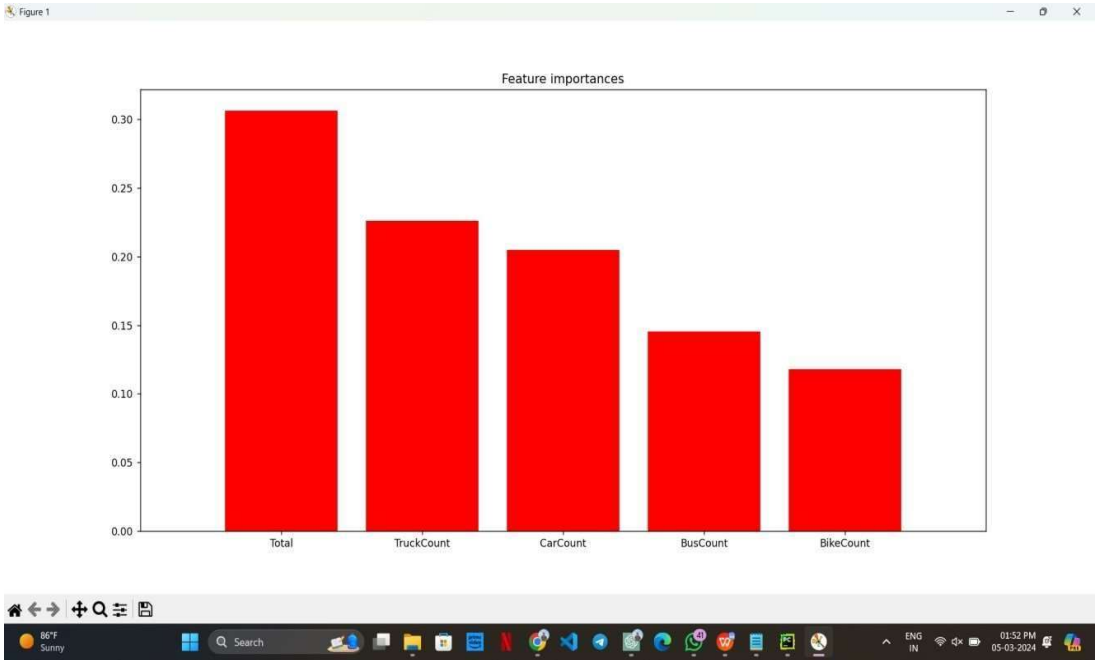
missing values. It also plots the feature importances to understand which features are most important for the model's predictions. This can be useful for feature selection and understanding the model better. The code uses a Random Forest Classifier for prediction. This is a powerful machine learning algorithm that can handle both regression and classification tasks. It is based on the idea of creating a 'forest' of decision trees, each trained on a random subset of the data, and then averaging their predictions. This helps to reduce overfitting and improve generalization. The code uses the `sklearn` library's implementation of Random Forest, which includes many options for customization, such as the number of trees in the forest (`n_estimators`) and the function to measure the quality of a split (`criterion`). The code uses a train-test split to evaluate the model's performance. This is a common method for evaluating machine learning models, where the data is split into a training set to train the model, and a test set to evaluate its performance. The code uses a 80-20 split, meaning 80% of the data is used for training and 20% for testing. This is a common choice, but the exact split can be adjusted depending on the size and nature of the data. The code evaluates the model's performance using a classification report, which provides precision, recall, f1-score, and support for each class. Precision is the ratio of true positives to the sum of true and false positives, recall is the ratio of true positives to the sum of true positives and false negatives, and the f1-score is the harmonic mean of precision and recall. Support is the number of instances of each class in the data. These metrics provide a comprehensive view of the model's performance. The code also displays the predicted traffic situation for the test set, which can be useful for understanding the model's predictions. The code includes a feature importance analysis, which helps to understand which features are most important for the model's predictions. This is done by plotting the feature importances returned by the Random Forest model. Features with higher importance are more influential in the model's predictions. This can be useful for feature selection and understanding the model better. The code uses the `matplotlib` and `seaborn` libraries for visualization. These are powerful libraries for creating static, animated, and interactive visualizations in Python. The code uses them to create a bar plot of feature importances, which provides a clear and intuitive visualization of the importance of each feature. The code is well-structured and follows good programming practices. It includes comments to explain what each section of the code does, and it uses meaningful variable names. This makes the code easier to understand and maintain. The code also handles potential issues such as missing values and unbalanced classes, which can affect the performance of the model. It uses

appropriate methods to handle these issues, such as filling missing values with the mean or mode, and using a Random Forest model, which can handle unbalanced classes. The code also includes some data exploration and visualization steps, which can help to understand the data and the model's predictions. For example, it prints the shape of the data and a few sample rows, checks for missing values, and plots the feature importances. These steps can provide valuable insights into the data and the model. The code uses a Random Forest Classifier for prediction. This is a powerful machine learning algorithm that can handle both regression and classification tasks. It is based on the idea of creating a 'forest' of decision trees, each trained on a random subset of the data, and then averaging their predictions. This helps to reduce overfitting and improve generalization. The code uses the `sklearn` library's implementation of Random Forest, which includes many options for customization, such as the number of trees in the forest (`n_estimators`) and the function to measure the quality of a split (`criterion`). The code uses a train-test split to evaluate the model's performance. This is a common method for evaluating machine learning models, where the data is split into a training set to train the model, and a test set to evaluate its performance. The code uses a 80-20 split, meaning 80% of the data is used for training and 20% for testing. This is a common choice, but the exact split can be adjusted depending on the size and nature of the data. The code evaluates the model's performance using a classification report, which provides precision, recall, f1-score, and support for each class. Precision is the ratio of true positives to the sum of true and false positives, recall is the ratio of true positives to the sum of true positives and false negatives, and the f1-score is the harmonic mean of precision and recall. Support is the number of instances of each class in the data. These metrics provide a comprehensive view of the model's performance. The code also displays the predicted traffic situation for the test set, which can be useful for understanding the model's predictions.

# Chapter 6

# RESULTS



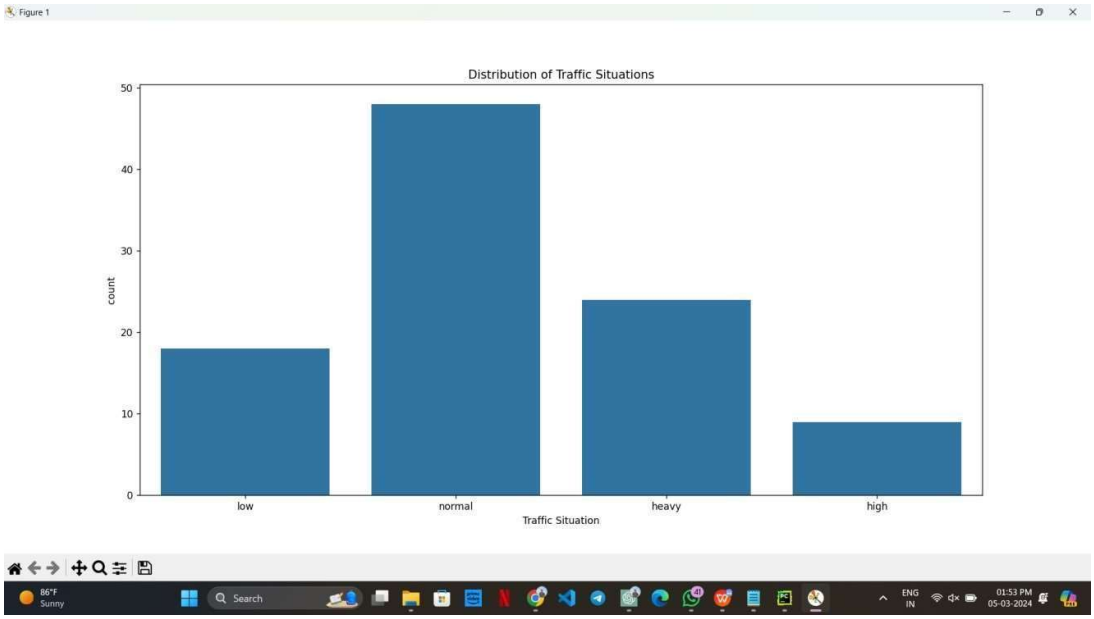


Figure 1
Distribution of Traffic Situations
count
Traffic Situation
low
normal
heavy
high
0
10
20
30
40
50
Search
86°F
Sunny
ENG
IN
01:53 PM
05-03-2024

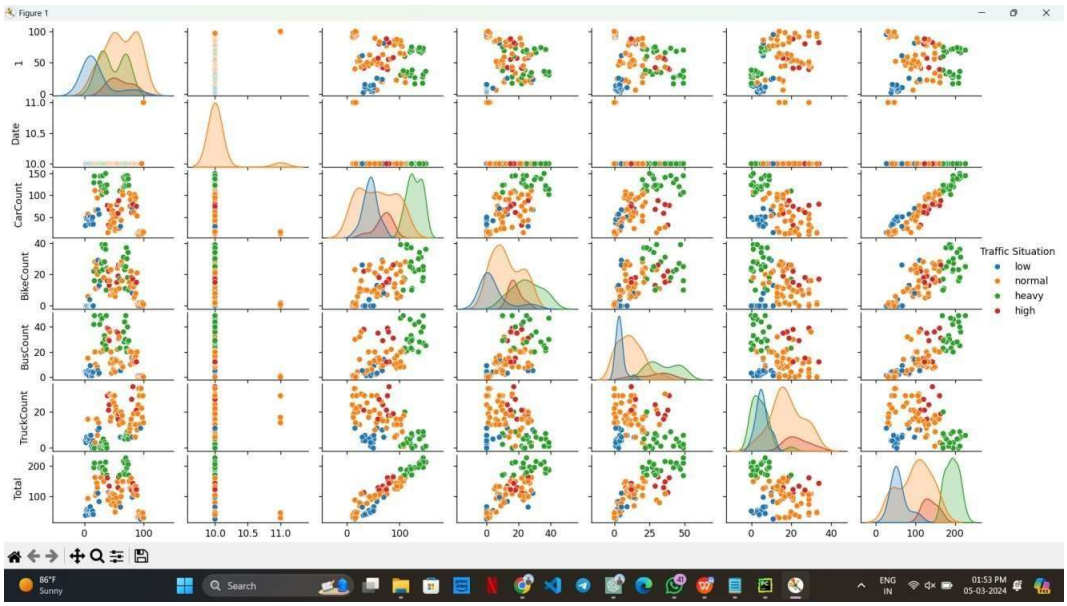


Figure 1
1
Date
CarCount
BikeCount
BusCount
TruckCount
Total
Traffic Situation
low
normal
heavy
high
Search
86°F
Sunny
ENG
IN
01:53 PM
05-03-2024

# Chapter 7

# REFLECTION NOTES

Our research underscores the effectiveness of employing Random Forest (RF) models for predicting traffic situations. The RF model's capability to capture intricate, non-linear relationships among various factors presents a promising avenue for managing urban traffic flow. By utilizing the RF model, urban planners and traffic management authorities gain a practical means to simulate diverse scenarios, aiding in the development of strategies to mitigate the adverse effects of urbanization on traffic congestion.

The application of machine learning algorithms for traffic prediction emerges as a valuable approach, facilitating the identification of critical traffic indicators and forecasting future trends based on existing data patterns.

Through this project, I gained valuable insights into several key concepts:

**Data Collection and Preprocessing:**Emphasizing the importance of collecting reliable traffic datasets, including variables like vehicle volume, speed, road conditions, and weather.Conducting thorough data preprocessing, including handling missing values and normalizing features to enhance model performance.

**Feature Selection and Engineering:**Recognizing the significance of identifying relevant features affecting traffic prediction, leveraging domain knowledge for informed feature selection.Exploring feature engineering techniques to create new features or transform existing ones, thereby improving the model's predictive ability.

**Model Selection:**Understanding the necessity of selecting appropriate machine learning models based on problem characteristics and data attributes.

Exploring various models such as Random Forest, Decision Trees, Support Vector Machines, and Neural Networks for traffic prediction tasks.

**Model Evaluation:**Employing suitable evaluation metrics like accuracy, precision, recall, and F1-score to assess the model's performance.

Utilizing cross-validation techniques to estimate the model's generalization ability on unseen data.

**Model Interpretability:**Acknowledging the importance of understanding how the model makes predictions, particularly in contexts like traffic management with significant real-world implications.Leveraging techniques such as feature importance analysis to gain insights into the model's decision-making process.

**Continuous Monitoring and Updating:**Recognizing the dynamic nature of traffic patterns and the importance of continuously monitoring and updating the model with fresh data to ensure its accuracy and relevance over time.

**Integration with Sensor Networks and IoT:**Exploring opportunities to integrate machine learning models with sensor networks and IoT devices for real-time traffic monitoring and prediction, facilitating timely interventions to alleviate congestion.

**Ethical Considerations:**Highlighting the importance of addressing biases in data and algorithms, and considering the social and environmental impacts of traffic prediction models and interventions.

**Collaboration and Knowledge Sharing:**Encouraging collaboration among stakeholders including urban planners, traffic management authorities, and local communities to foster interdisciplinary approaches to traffic prediction and management.

**Challenges and Future Directions:**Identifying challenges such as data scarcity, interpretability, and scalability, which need to be addressed for further advancements in traffic prediction.Considering future research directions, including the exploration of advanced machine learning techniques and integration with emerging technologies like satellite imagery and remote sensing for enhanced traffic management

# Chapter 8

## References

[1] https://www.researchgate.net/publication/359387501_Traffic_Prediction_Using_Machine_Learning

[2] https://github.com/Nupurgopali/Traffic-Prediction-using-SVR-and-RFR/blob/master/README.md

[3] [2305.19591] Traffic Prediction using Artificial Intelligence: Review of Recent Advances and Emerging Opportunities (arxiv.org)

[4] A traffic prediction model based on multiple factors | The Journal of Supercomputing (springer.com)

[5] Artificial intelligence-based traffic flow prediction: a comprehensive review | Journal of Electrical Systems and Information Technology | Full Text (springeropen.com)