Weekly Progress Report

Name: CH.Shiva Kumar

Domain: Data science and machine Learning

Date of submission:23/04/2024

Week Ending:04

Project Format:

- ✓ **Name of Project:** Forecasting of Smart city traffic patterns
- ✓ **Problem Statement:**
  - We are working with the government to transform various cities into a smart city. The vision is to convert it into a digital and intelligent city to improve the efficiency of services for the citizens. One of the problems faced by the government is traffic. You are a data scientist working to manage the traffic of the city better and to provide input on infrastructure planning for the future.
  - The government wants to implement a robust traffic system for the city by being prepared for traffic peaks. They want to understand the traffic patterns of the four junctions of the city. Traffic patterns on holidays, as well as on various other occasions during the year, differ from normal working days. This is important to take into account for your forecasting

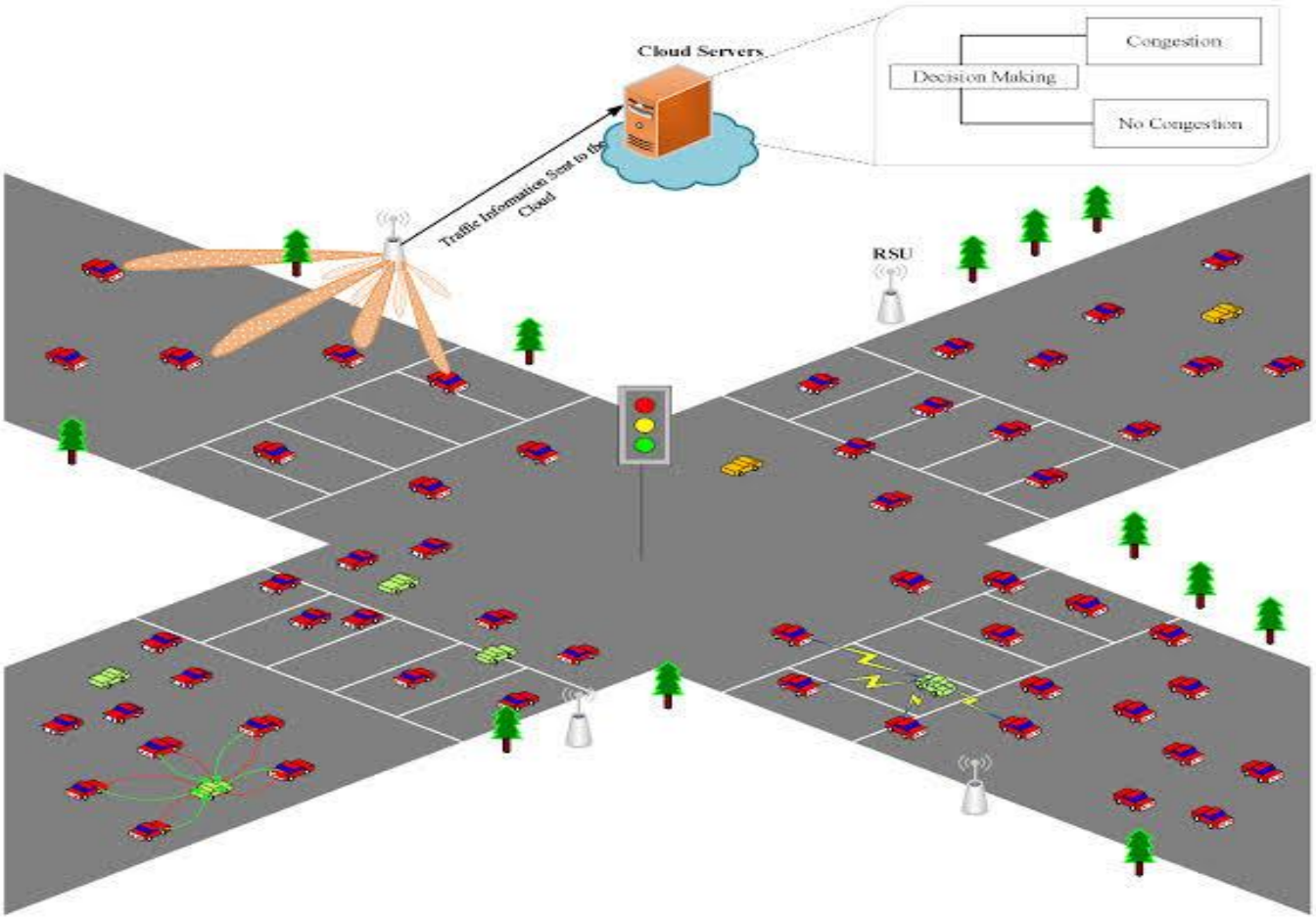
- ✓ **Requirements:**
  - Training data set
  - Testing data set
- ✓ **Software's:**
  - Command prompt to import libraries
  - Anaconda prompt
  - PowerBI
- ✓ **Libraries to be Installed :** By using direct commands
  - pip install numpy
  - pip install pandas
  - pip install matplotlib.pyplot

- pip install Sequential
- pip install Dense
- pip install LSTM
- pip install mean_squared_error
- pip install sklearn
- pip install tensorflow
- pip install MinMaxScaler, StandardScaler

Then install other as required

✓ **Operations in order:**

- install the required libraries
- import the libraries
- read the training and testing data sets by using pandas
- train the data set as per the daily analysis
- test the data set to improve it's performance
- check it by running in the anaconda prompt
- verify with expected output



Learning from above:

Understanding the importance of analyzing traffic patterns, including variations on holidays and special occasions, and the significance of integrating data science into urban planning.

Integration:

Integrating data from various sources such as traffic cameras, sensors, historical data, and event calendars to analyze and forecast traffic patterns effectively.

Project complexity:

Moderate to high, considering the need for analyzing diverse data sources, handling seasonal variations, and predicting traffic peaks accurately.

Learning resources:

Utilize machine learning algorithms for time series forecasting, study urban planning principles, and explore data visualization techniques for communicating insights effectively.

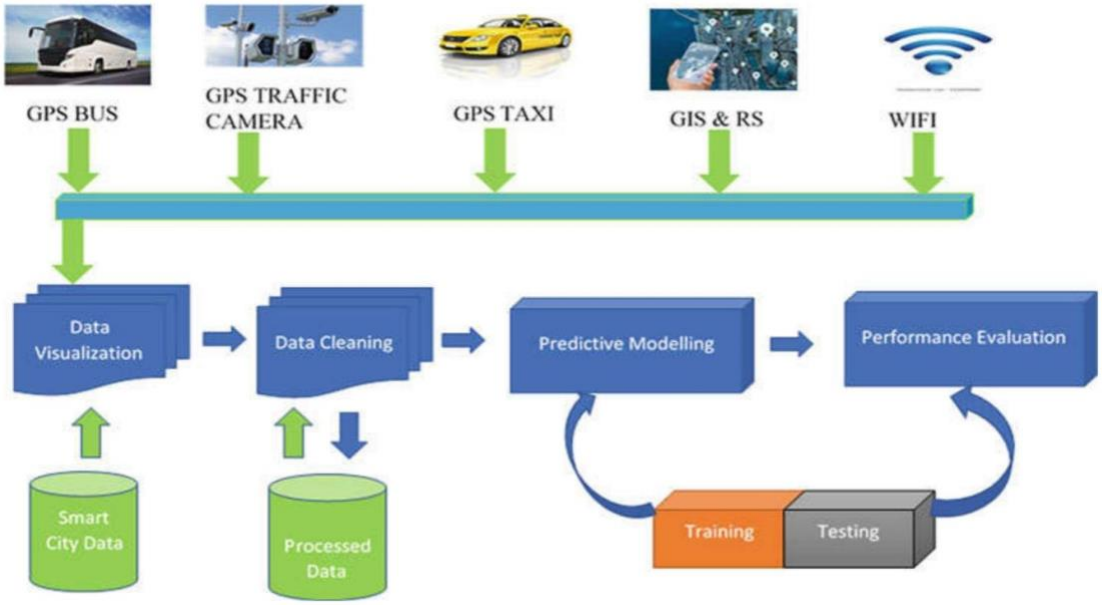
Additional comments:

Collaboration with urban planners and stakeholders is crucial for aligning the traffic forecasting model with the city's infrastructure development plans.

Goals:

Gather and clean historical traffic data, explore different forecasting models, and initiate discussions with urban planners to understand future infrastructure projects and their impact on traffic patterns.

PROJECT EXECUTION STEPS:



- ✓ **Load And Clean Data:**
  - Download the data sets for source link https://drive.google.com/file/d/1y61cDyuO9Zrp1fSchWcAmCxk0B6SMx7X/view?usp=sharing
  - We can clean data by using the Two methods:
    - By Using the Power-Bi desktop (or) Tableau software applications
    - By Using Python Modules
  - By Using the Power-Bi desktop (or) Tableau software applications
    - Open the Power Bi First
    - Let take the data from the Option 'Get data'
    - Next click on from the Common data sources As "Excel workbook"
    - Select the file '2_Ecommerce Sales Data Analysis Excel.xlsx" from the required path
    - Then Select All the Tables in the Dataset LOAD the data field
    - Select the discount column and change the formula as"percentage"
    - Delete the column field "Customer Name"
    - Transform the data when completed of the above steps

- Choose Columns as Year,Order Date,Ship Data,Shipment days,Ship Mode,Segment,country,city,state,Region,Category,Sub_category,Sales,Quantity,Discount,Profit
- Change the Data fields as per Requirements
- Then Close & Apply

(If you want to know better about the dataset then perform the visualization by using different methods like:bar graph,pie-chart with to various locations and for different symbols in the traffic places to make the better smart City prediction)

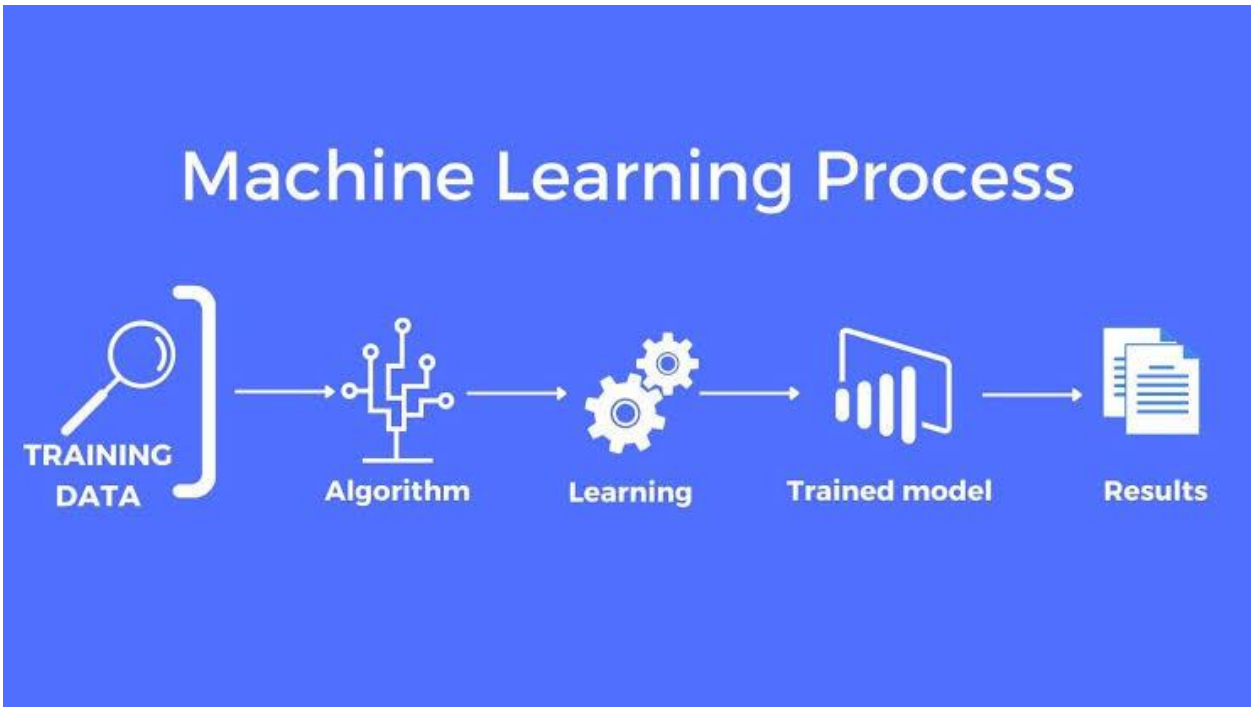
- By Using Python Modules
  - import or load the data set by using python padas module
  - Syntax: Variable_name=pandas.read_csv("file location")
  - import the modules numpy, pandas, matplotlib.pyplot
  - Then load the training dataset and clean by below methods for remove null values, duplicate values and zero values
  - Variable_name=pandas.dropna(inplace=True) Here Variable_name is loaded by the python
  - Variabl_name.drop_duplicates(inplace=True)
  - Then we can remove the duplicate values from the dataset.
- Now we can train the data model with this trained dataset
  - Next wee can take the testing dataset to test the model
  - Before testing we need to clear the duplicate values from the testing dataset as above syntax like the training dataset
- Example operations to CSV file:
  - var.head() : Display the some top of the rows
  - var.tail() : Display the bottom of the rows
  - var.info() : Give the information of dataset
  - var.describe : Give the discription of dataset
  - var.colums : Give the all columns in dataset

✓ **TRAIN AND TEST DATASET:**

- Training The Dataset Steps: We are cleaned and imported the dataset in above steps
- Now we train this Dataset.

- import the libraries
  - Example: import numpy as np
- import the datasets
  - Example: Tain_set=pd.read_csv("abc.csv")
  - Note : that file should be in the folder of the used anaconda prompt used folder.

TRAINING DATASET:



These are the main steps in training datasets in data science and machine learning, along with brief explanations and examples:

1) Data Preprocessing:
   - Cleaning the data (removing duplicates, handling missing values).
   - Transforming data into a usable format (scaling, encoding categorical variables).
   - Example (Python – using pandas and scikit-learn):

```python
Import pandas as pd
From sklearn.preprocessing import StandardScaler
#Load dataset
Data = pd.read_csv('data.csv')
# Remove duplicates
Data.drop_duplicates(inplace=True)
# Handle missing values
Data.fillna(data.mean(), inplace=True)
# Scale features
Scaler = StandardScaler()
Scaled_data = scaler.fit_transform(data)
```

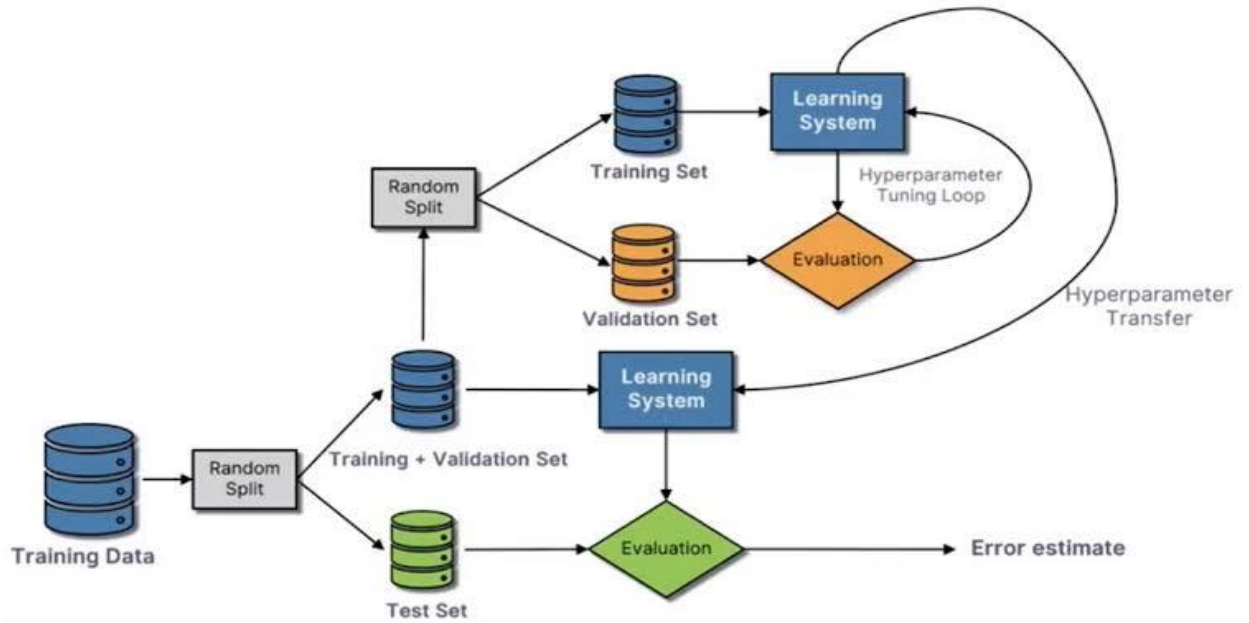
2) Feature Engineering:
    - Creating new features or transforming existing features to improve model performance.
    - Example (Python – using pandas):
      ```python
      Creating new feature: Total income
      Data['TotalIncome'] = data['ApplicantIncome'] + data['CoapplicantIncome']
      ```
3) Model Selection:
    - Choosing the appropriate machine learning algorithm for the problem at hand.
    - Example (Python – using scikit-learn):
      ```python
      From sklearn.ensemble import RandomForestClassifier
      Model = RandomForestClassifier()
      ```
4) Model Training:
    - Training the selected model on the training data.
    - Example (Python – using scikit-learn):
      ```python
      Model.fit(X_train, y_train)
      ```
5) Model Evaluation (Optional):
    - Assessing the performance of the trained model using evaluation metrics.

- Example (Python – using scikit-learn):
  ```python
  From sklearn.metrics import accuracy_score
  Predictions = model.predict(X_test)
  Accuracy = accuracy_score(y_test, predictions)
  Print("Accuracy:", accuracy)
  ```

6) Hyperparameter Tuning (Optional):
   - Fine-tuning the model's hyperparameters to improve performance.
   - Example (Python – using scikit-learn's GridSearchCV):
     ```python
     From sklearn.model_selection import GridSearchCV
     Param_grid = {'n_estimators': [100, 200, 300], 'max_depth': [None, 10, 20]}
     Grid_search = GridSearchCV(model, param_grid, cv=5)
     Grid_search.fit(X_train, y_train)
     Best_params = grid_search.best_params_
     ```

These steps provide a comprehensive guide for training machine learning models, ensuring they are well-prepared to make accurate predictions on unseen data.

TESTING DATASET:



These are main steps in testing datasets in data science and machine learning, along with brief explanations and examples:

1) Data Preprocessing (if necessary):
    - Apply the same preprocessing steps used for the training data, such as cleaning and feature scaling.
2) Model Prediction:
    - Use the trained model to make predictions on the testing data.
    - Example (Python – using scikit-learn):
      ```python
      Predictions = model.predict(X_test)
      ```
3) Model Evaluation:
    - Assess the performance of the trained model on the testing data using appropriate evaluation metrics.
    - Example (Python – using scikit-learn):
      ```python
      From sklearn.metrics import accuracy_score, classification_report
      Accuracy = accuracy_score(y_test, predictions)
      Print("Accuracy:", accuracy)
      Report = classification_report(y_test, predictions)
      Print("Classification Report:\n", report)
      ```
4) Visualization (Optional):
    - Visualize the results to gain insights into the model's performance.
    - Example (Python – using matplotlib/seaborn):
      ```python
      Import matplotlib.pyplot as plt
      Import seaborn as sns
      # Confusion matrix visualization
      Sns.heatmap(confusion_matrix(y_test, predictions), annot=True,
      cmap='Blues')
      Plt.xlabel('Predicted labels')
      Plt.ylabel('True labels')
      Plt.title('Confusion Matrix')
      Plt.show()
      ```

```
```

5) Fine-Tuning (Optional):
    - If the model performance is not satisfactory, consider fine-tuning hyperparameters or trying different algorithms.
    - Example (Python – using scikit-learn's GridSearchCV):
      ```python
      From sklearn.model_selection import GridSearchCV
      Param_grid = {'n_estimators': [100, 200, 300], 'max_depth': [None, 10, 20]}
      Grid_search = GridSearchCV(model, param_grid, cv=5)
      Grid_search.fit(X_train, y_train)
      Best_params = grid_search.best_params_
      ```
6) Final Assessment:
    - Evaluate the overall performance of the model and decide whether it meets the desired criteria for deployment.

These steps ensure that the trained model is rigorously tested and evaluated to ensure its effectiveness and reliability in making predictions on new, unseen data.

Conclusion:

Hence we create the data model for the required Data Analysis to perform the Forecasting of Smart city traffic patterns