SDAIA
الهيئة السعودية للبيانات والذكاء الاصطناعي
Saudi Data & AI Authority

أكاديمية طويق
Tuwaiq Academy

Traffic Prediction

Mini Project T5 BootCamp

Team Work

Joud Tarek : Deep Learning model

Retaj Alanzii : Deep Learning Model

Dana Almistdadi : EDA & Visualization

Rana al-Harsan : Evaluation Models

Table of Contact

Traffic Prediction Overview

objective

The idea for the project is to create a deep learning model for predicting traffic jams on roads by utilizing the data that is currently available, such as the time of day, the quantity and kind of cars on the road, and the type of traffic. Based on the given data, the model will forecast the kind and timing of congestion.

Data Source

The data was gathered from Kaggle data source.
The two data files that make up these are called "Traffic" and "Traffic in Two Months."

Source : Kaggle

methodology

We shall first combine the data. Following data cleaning, we will show the cleaned data and then move on to data splitting. Next, we'll use Random Forest to build a machine learning model and evaluate its performance. Next, we will use a neural network to build a deep learning model, show a detailed examination of the model's output, print the results, and make a graph to show the outcomes.

EDA & Visualization

Understanding Data

The dataset focuses on traffic congestion in urban areas. This dataset offers valuable insights for transportation planning, infrastructure development, and congestion management, enabling stakeholders to make informed decisions that enhance urban mobility and contribute to sustainable city planning. Updated every 15 minutes over one month, the dataset provides a comprehensive view of traffic patterns.
It includes a traffic situation column with four categories: Heavy,High, Normal, and Low, which helps assess congestion severity.

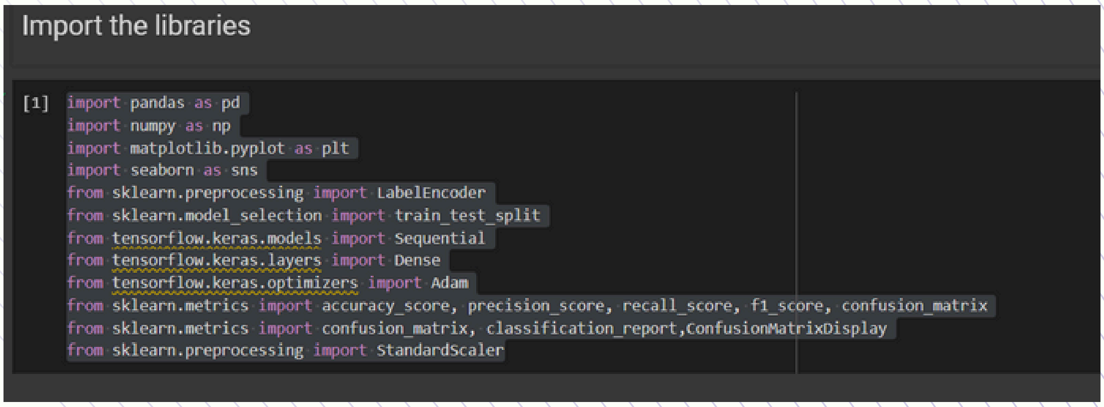
The dataset consists of 18 columns: 12 integer, 4 string, and 2 DateTime.

Column Name	Description
Time : object	The time column is used to indicate the period of congestion.
Date : int	The date of the congestion occurrence.
Day of the week : object	the day of the congestion occur
CarCount : int	Count how many car occur in the Traffic
BikeCount : int	Count how many bike occur in the Traffic
BusCount : int	Count how many bus occur in the Traffic
TruckCount : int	Count how many Truck occur in the Traffic
Total : int	Count Total Cars occur in the Traffic
Traffic Situation : Object	Describe the Traffic situation if it’s normal or havey ..etc

EDA & Visualization

Import libraries

in this first step of creating model , we applied libraries from tensorflow.keras to be able to create Nural_Network model.

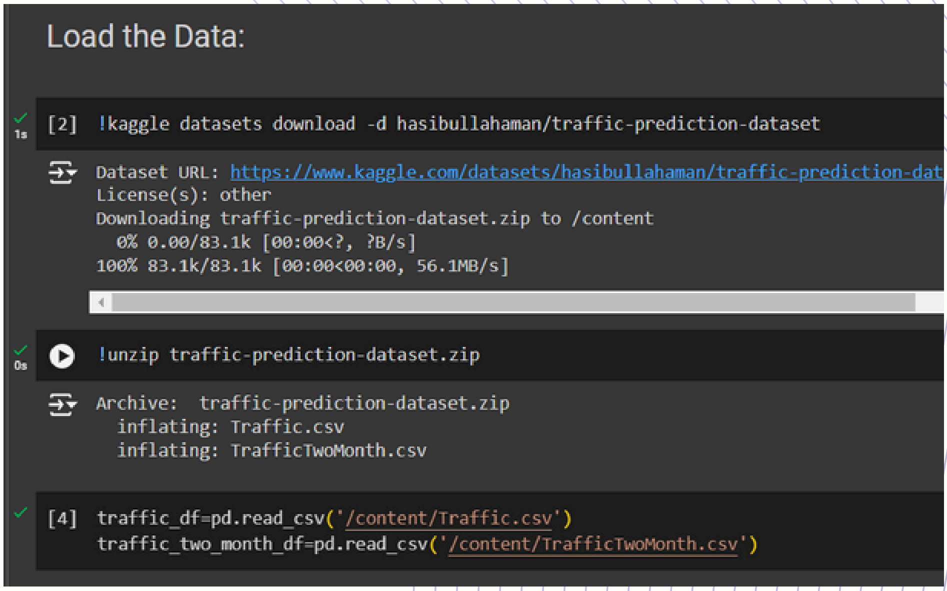


Import the libraries

```
[1] import pandas as pd
    import numpy as np
    import matplotlib.pyplot as plt
    import seaborn as sns
    from sklearn.preprocessing import LabelEncoder
    from sklearn.model_selection import train_test_split
    from tensorflow.keras.models import Sequential
    from tensorflow.keras.layers import Dense
    from tensorflow.keras.optimizers import Adam
    from sklearn.metrics import accuracy_score, precision_score, recall_score, f1_score, confusion_matrix
    from sklearn.metrics import confusion_matrix, classification_report,ConfusionMatrixDisplay
    from sklearn.preprocessing import StandardScaler
```

Load Dataset

in loading dataset step we call the data from the sours it's self by using the link and add it to google colab , the data was to file and we call each one of them



Load the Data:

```
[2] !kaggle datasets download -d hasibullahaman/traffic-prediction-dataset

Dataset URL: https://www.kaggle.com/datasets/hasibullahaman/traffic-prediction-dat
License(s): other
Downloading traffic-prediction-dataset.zip to /content
  0% 0.00/83.1k [00:00<?, ?B/s]
100% 83.1k/83.1k [00:00<00:00, 56.1MB/s]
```

```
!unzip traffic-prediction-dataset.zip

Archive:  traffic-prediction-dataset.zip
  inflating: Traffic.csv
  inflating: TrafficTwoMonth.csv
```

```
[4] traffic_df=pd.read_csv('/content/Traffic.csv')
    traffic_two_month_df=pd.read_csv('/content/TrafficTwoMonth.csv')
```

EDA :

Step 3: Data Preparation and Preliminary Analysis

Initial Dataset Examination

We commenced our analysis with the Traffic_csv dataset. The following steps were undertaken to understand and prepare the data:

Previewing the Dataset:

The first and last five rows of the dataset were examined to gain an overview of the data structure and the types of values present in each column.
This preliminary inspection helped in identifying the nature of the data and provided initial insights into the distribution of traffic-related features.

Data Type Analysis:

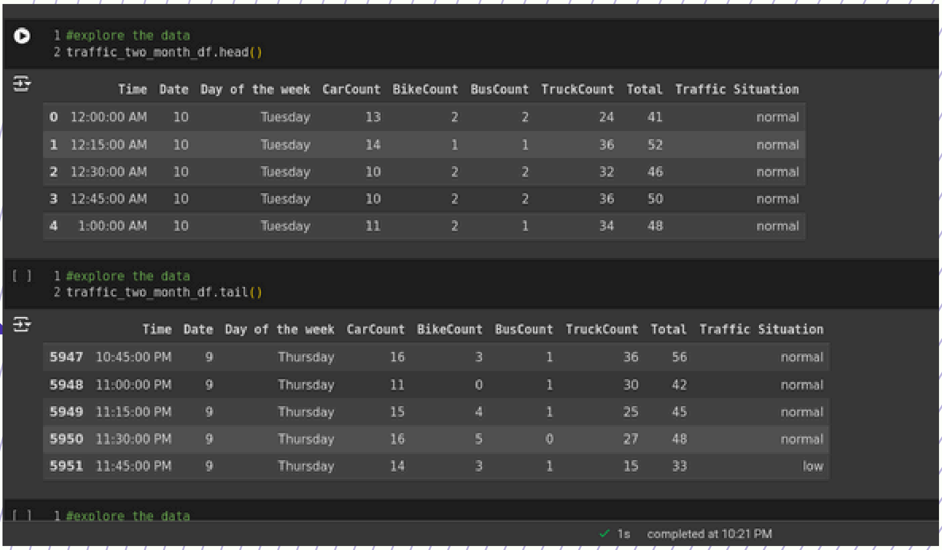
The data types of each column were reviewed using the info() method. Understanding the data types was crucial for identifying categorical and numerical features, as well as for determining any necessary conversions or adjustments.

Null Value Detection:

A thorough check for null values was performed to identify any missing data. This step ensured that potential gaps in the dataset were recognized early, allowing for appropriate handling strategies to be implemented.

Duplicate Data Check:

The dataset was examined for duplicate entries. Ensuring data uniqueness was vital for maintaining the integrity and reliability of the analysis.
These steps were repeated for the second dataset, TrafficTwoMonth_csv, to ensure that both datasets were thoroughly examined and prepared for subsequent analysis.



```
#explore the data
traffic_two_month_df.head()
```

	Time	Date	Day of the week	CarCount	BikeCount	BusCount	TruckCount	Total	Traffic Situation
0	12:00:00 AM	10	Tuesday	13	2	2	24	41	normal
1	12:15:00 AM	10	Tuesday	14	1	1	36	52	normal
2	12:30:00 AM	10	Tuesday	10	2	2	32	46	normal
3	12:45:00 AM	10	Tuesday	10	2	2	36	50	normal
4	1:00:00 AM	10	Tuesday	11	2	1	34	48	normal

```
#explore the data
traffic_two_month_df.tail()
```

	Time	Date	Day of the week	CarCount	BikeCount	BusCount	TruckCount	Total	Traffic Situation
5947	10:45:00 PM	9	Thursday	16	3	1	36	56	normal
5948	11:00:00 PM	9	Thursday	11	0	1	30	42	normal
5949	11:15:00 PM	9	Thursday	15	4	1	25	45	normal
5950	11:30:00 PM	9	Thursday	16	5	0	27	48	normal
5951	11:45:00 PM	9	Thursday	14	3	1	15	33	low

```
#explore the data
```

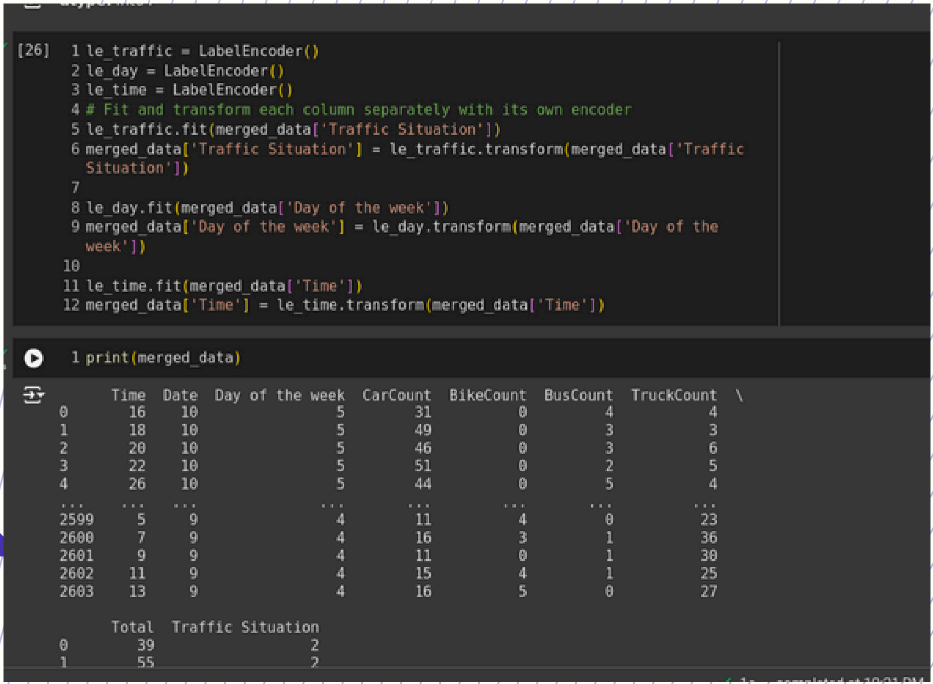
✓ 1s completed at 10:21 PM

EDA :

Data Visualization:
To deepen our understanding of the datasets, various visualizations were created. These visualizations helped in exploring relationships between features, identifying patterns, and gaining insights into the overall distribution of the data. This exploratory analysis was crucial for informing the next stages of the project.

Dataset Merging and Preparation for Modeling
Prior to developing the Neural Network model, the two datasets were merged to create a comprehensive dataset.

The following preparatory steps were then executed:
Label Encoding:
Categorical features were converted into numerical values through label encoding. This transformation was necessary to enable the application of machine learning algorithms that require numerical input.
Standard Scaling:
The dataset was standardized using standard scaling, ensuring that all features had a consistent scale. This step was essential for optimizing the performance of the deep learning model by preventing any single feature from disproportionately influencing the results.
By carefully executing these steps, the dataset was cleaned, analyzed, and fully prepared for the development of the Neural Network model.



```
[26]
le_traffic = LabelEncoder()
le_day = LabelEncoder()
le_time = LabelEncoder()
# Fit and transform each column separately with its own encoder
le_traffic.fit(merged_data['Traffic Situation'])
merged_data['Traffic Situation'] = le_traffic.transform(merged_data['Traffic
Situation'])

le_day.fit(merged_data['Day of the week'])
merged_data['Day of the week'] = le_day.transform(merged_data['Day of the
week'])

le_time.fit(merged_data['Time'])
merged_data['Time'] = le_time.transform(merged_data['Time'])
```

```
print(merged_data)
```

```
      Time  Date  Day of the week  CarCount  BikeCount  BusCount  TruckCount  \
0       16    10                5        31          0         4           4
1       18    10                5        49          0         3           3
2       20    10                5        46          0         3           6
3       22    10                5        51          0         2           5
4       26    10                5        44          0         5           4
...    ...   ...              ...       ...        ...       ...         ...
2599     5     9                4        11          4         0          23
2600     7     9                4        16          3         1          36
2601     9     9                4        11          0         1          30
2602    11     9                4        15          4         1          25
2603    13     9                4        16          5         0          27

      Total  Traffic Situation
0        39                  2
1        55                  2
```

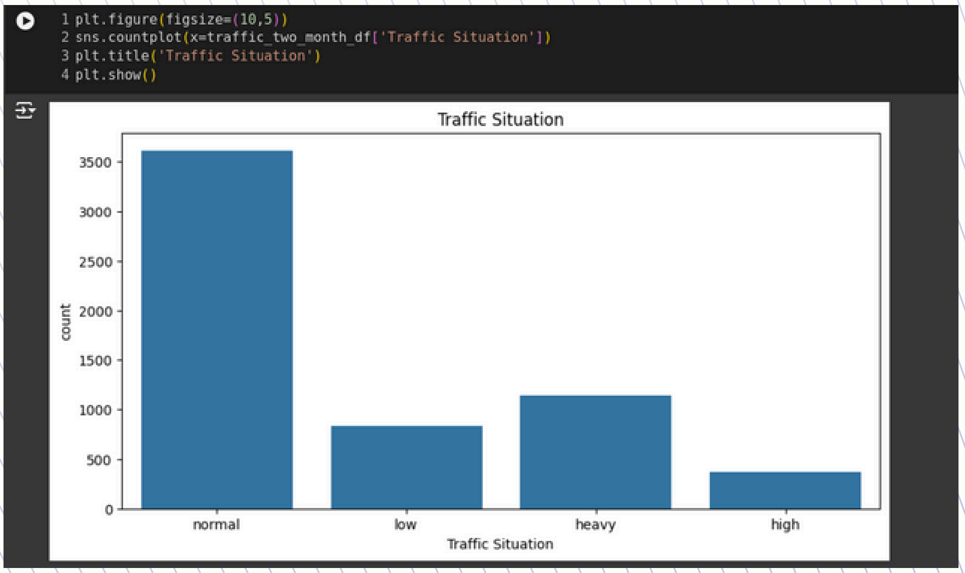
08

visualization the dataset:

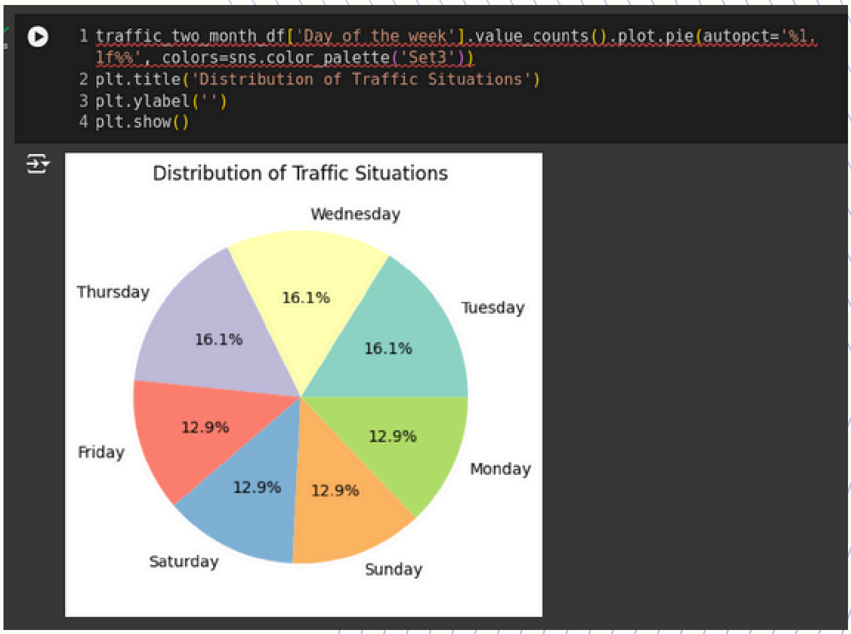
Count Plot:

Distribution of Traffic Situations

The count plot below visualizes the distribution of different traffic situations across the dataset. The Traffic Situation column categorizes traffic flow into distinct situations such as "Heavy," “High”,"Normal," and "Low." Each bar in the plot represents the frequency of occurrences for each traffic situation.



```
plt.figure(figsize=(10,5))
sns.countplot(x=traffic_two_month_df['Traffic Situation'])
plt.title('Traffic Situation')
plt.show()
```



```
traffic_two_month_df['Day of the week'].value_counts().plot.pie(autopct='%1.
1f%%', colors=sns.color_palette('Set3'))
plt.title('Distribution of Traffic Situations')
plt.ylabel('')
plt.show()
```

Pie Chart:

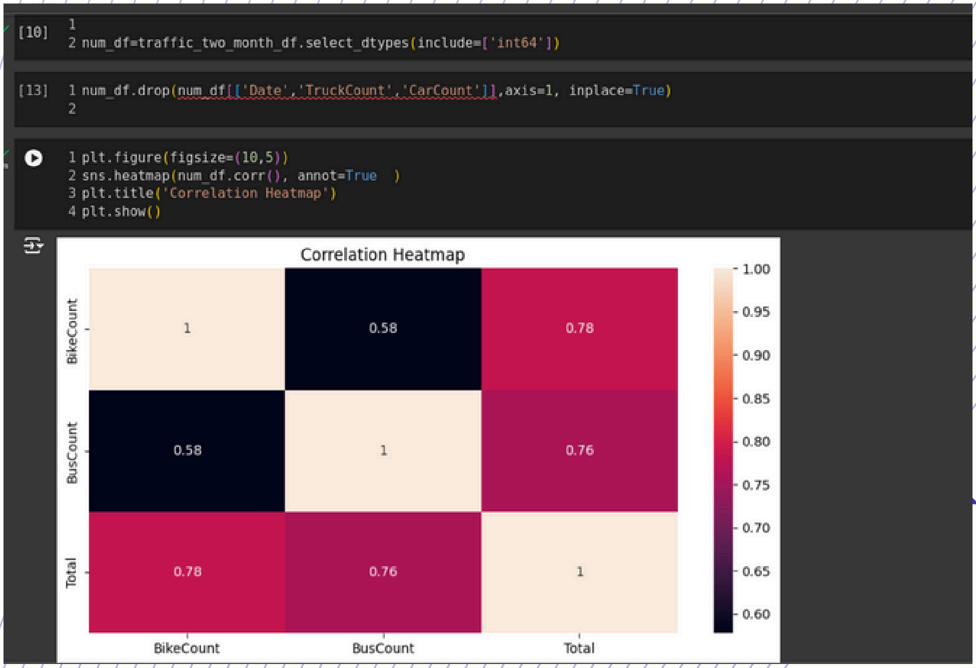
Traffic Distribution by Day of the Week

The pie chart below illustrates the distribution of traffic data across different days of the week. Each segment of the pie represents the proportion of traffic recorded on a specific day.

Heatmap:

Correlation of Numerical Data

The heatmap below visualizes the correlation matrix of the numerical features in the dataset. After removing the Date, TruckCount, and CarCount columns, the remaining numerical columns are analyzed to identify how closely they are related to each other.



```
[10]
num_df=traffic_two_month_df.select_dtypes(include=['int64'])
```

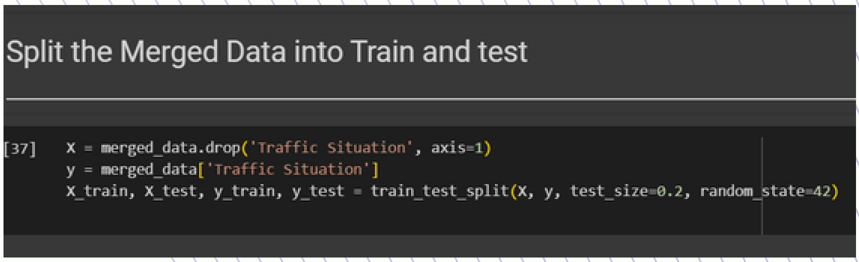
```
[13]
num_df.drop(num_df[['Date','TruckCount','CarCount']],axis=1, inplace=True)
```

```
plt.figure(figsize=(10,5))
sns.heatmap(num_df.corr(), annot=True  )
plt.title('Correlation Heatmap')
plt.show()
```

ANN Model

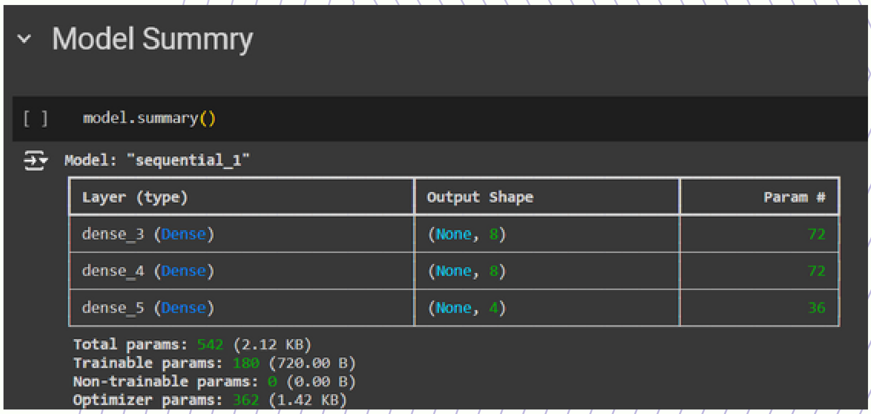
The data was Splitting according to the required target , and we chose the Artificial Neural Network (ANN) because it is considered the most suitable for our data, which includes texts. This choice was made based on ANN's ability to effectively represent the data, making it easier to add more text data.

The network was designed based on the input data, and the model was evaluating multiple times to achieve the best possible results and to ensure that the network's layers are well understood. Our choice of ANN was also based on the possibility of developing it in the future to handle large amounts of images, videos, and additional text data. After that, the model was tested to obtain the best possible results, and from the charts, we were able to clearly understand the outcomes.



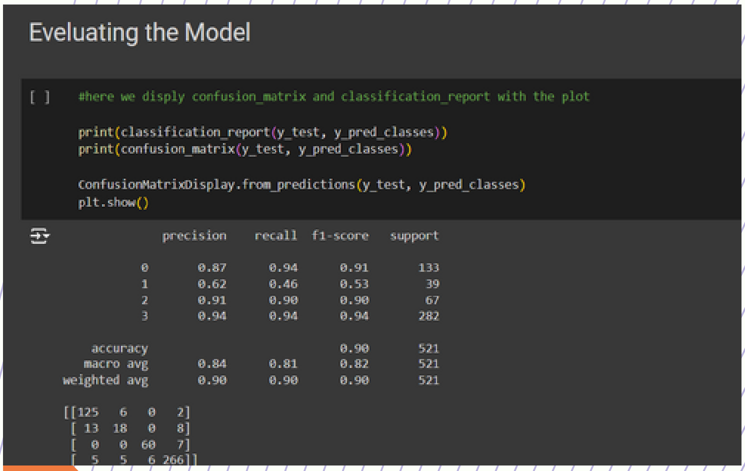
Split the Merged Data into Train and test

```
[37]  X = merged_data.drop('Traffic Situation', axis=1)
      y = merged_data['Traffic Situation']
      X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)
```



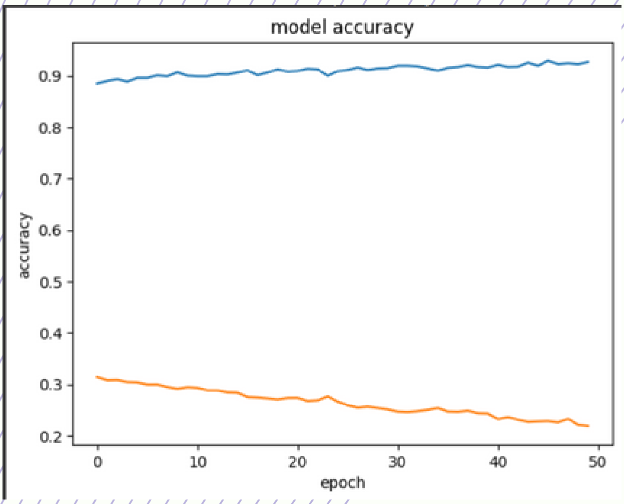
Model Summry

```
[ ]  model.summary()

Model: "sequential_1"
┏━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━┳━━━━━━━━━━━━━━━━━━━━━━━━━━━━━┳━━━━━━━━━━━━━━━━━┓
┃ Layer (type)                         ┃ Output Shape                ┃         Param # ┃
┡━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━╇━━━━━━━━━━━━━━━━━━━━━━━━━━━━━╇━━━━━━━━━━━━━━━━━┩
│ dense_3 (Dense)                      │ (None, 8)                   │              72 │
├──────────────────────────────────────┼─────────────────────────────┼─────────────────┤
│ dense_4 (Dense)                      │ (None, 8)                   │              72 │
├──────────────────────────────────────┼─────────────────────────────┼─────────────────┤
│ dense_5 (Dense)                      │ (None, 4)                   │              36 │
└──────────────────────────────────────┴─────────────────────────────┴─────────────────┘
 Total params: 542 (2.12 KB)
 Trainable params: 180 (720.00 B)
 Non-trainable params: 0 (0.00 B)
 Optimizer params: 362 (1.42 KB)
```



Eveluating the Model

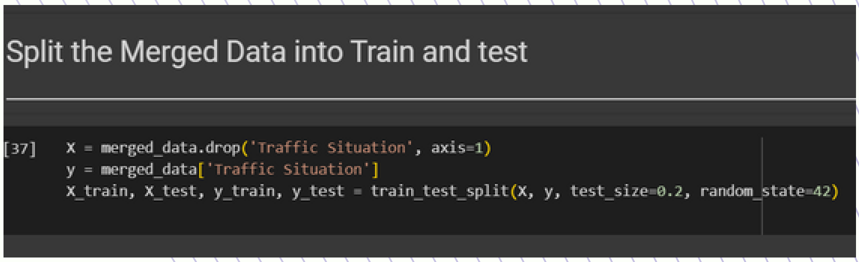
```
[ ]  #here we disply confusion_matrix and classification_report with the plot

     print(classification_report(y_test, y_pred_classes))
     print(confusion_matrix(y_test, y_pred_classes))

     ConfusionMatrixDisplay.from_predictions(y_test, y_pred_classes)
     plt.show()

              precision    recall  f1-score   support

           0       0.87      0.94      0.91       133
           1       0.62      0.46      0.53        39
           2       0.91      0.90      0.90        67
           3       0.94      0.94      0.94       282

    accuracy                           0.90       521
   macro avg       0.84      0.81      0.82       521
weighted avg       0.90      0.90      0.90       521

[[125   6   0   2]
 [ 13  18   0   8]
 [  0   0  60   7]
 [  5   5   6 266]]
```



ANN Model

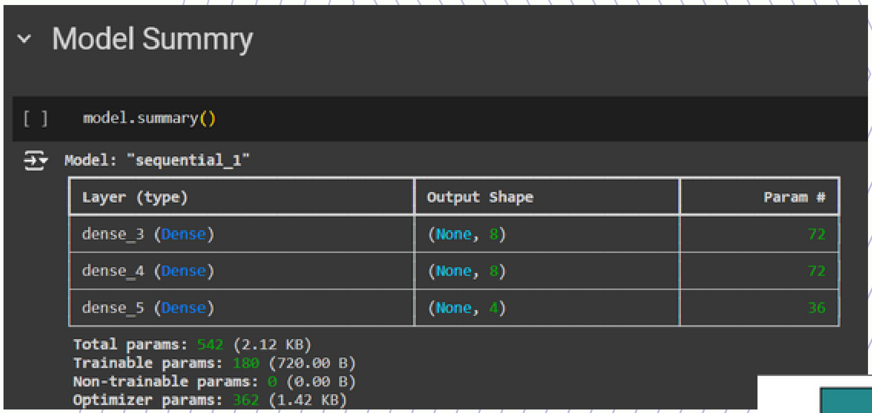
The data was Splitting according to the required target , and we chose the Artificial Neural Network (ANN) because it is considered the most suitable for our data, which includes texts. This choice was made based on ANN's ability to effectively represent the data, making it easier to add more text data.

The network was designed based on the input data, and the model was evaluating multiple times to achieve the best possible results and to ensure that the network's layers are well understood. Our choice of ANN was also based on the possibility of developing it in the future to handle large amounts of images, videos, and additional text data. After that, the model was tested to obtain the best possible results, and from the charts, we were able to clearly understand the outcomes.



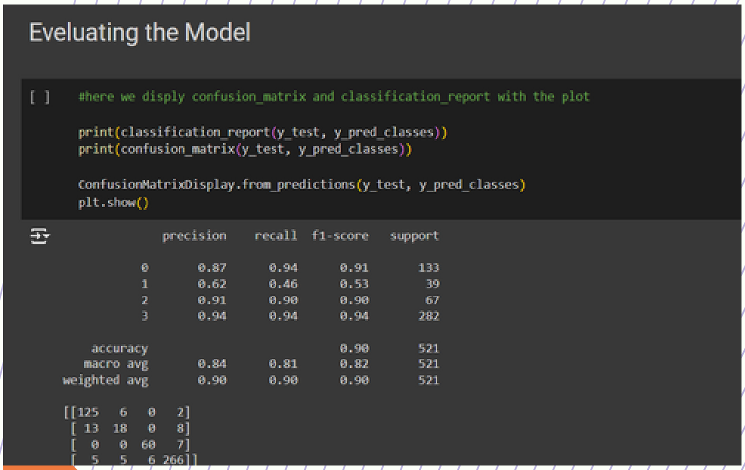
Split the Merged Data into Train and test

```
[37]  X = merged_data.drop('Traffic Situation', axis=1)
      y = merged_data['Traffic Situation']
      X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)
```



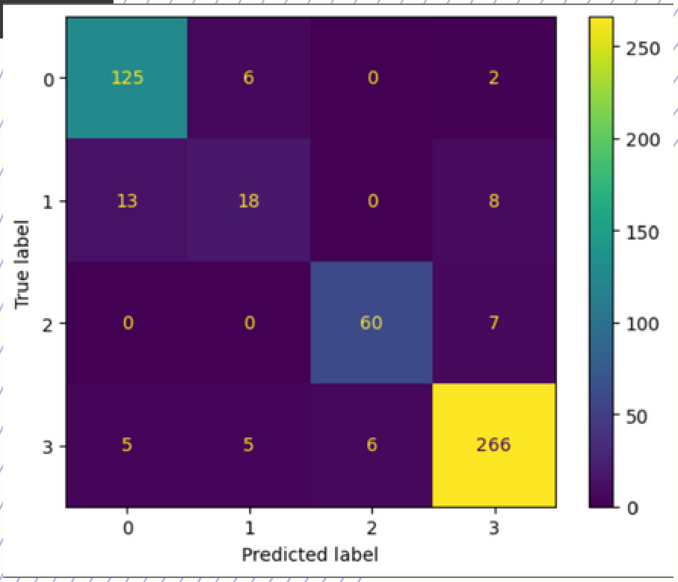
Model Summry

```
[ ]   model.summary()

Model: "sequential_1"
┌──────────────────────────────┬───────────────────────┬──────────┐
│ Layer (type)                 │ Output Shape          │  Param # │
├──────────────────────────────┼───────────────────────┼──────────┤
│ dense_3 (Dense)              │ (None, 8)             │       72 │
├──────────────────────────────┼───────────────────────┼──────────┤
│ dense_4 (Dense)              │ (None, 8)             │       72 │
├──────────────────────────────┼───────────────────────┼──────────┤
│ dense_5 (Dense)              │ (None, 4)             │       36 │
└──────────────────────────────┴───────────────────────┴──────────┘
 Total params: 542 (2.12 KB)
 Trainable params: 180 (720.00 B)
 Non-trainable params: 0 (0.00 B)
 Optimizer params: 362 (1.42 KB)
```



Eveluating the Model

```
[ ]   #here we disply confusion_matrix and classification_report with the plot

      print(classification_report(y_test, y_pred_classes))
      print(confusion_matrix(y_test, y_pred_classes))

      ConfusionMatrixDisplay.from_predictions(y_test, y_pred_classes)
      plt.show()

              precision    recall  f1-score   support

           0       0.87      0.94      0.91       133
           1       0.62      0.46      0.53        39
           2       0.91      0.90      0.90        67
           3       0.94      0.94      0.94       282

    accuracy                           0.90       521
   macro avg       0.84      0.81      0.82       521
weighted avg       0.90      0.90      0.90       521

[[125   6   0   2]
 [ 13  18   0   8]
 [  0   0  60   7]
 [  5   5   6 266]]
```



Solutions

Smart Traffic Management:
Real-time traffic signal timing adjustments can be made by applying prediction results.

Emergency Management:
Using predictions, direct emergency vehicles (fire trucks, ambulances) along less crowded routes.

Dedicated Routes for Public Transportation:
Assign particular lanes for public transportation according to predictions.

Conclusion

In conclusion, the deep learning-based traffic congestion prediction study successfully showed how advanced algorithms can evaluate complicated information to predict traffic patterns. This project offers important insights that can optimize urban design, improve traffic management, and raise overall road safety by accurately predicting congestion. This creative method is a big step toward building more intelligent, effective transportation systems.