Traffic Prediction (congestion Level)

**Description**

This program generates a synthetic dataset of traffic data, preprocesses it by encoding categorical variables and normalizing features, and then splits it into training and testing sets. A neural network model, built using TensorFlow and Keras, is employed to predict traffic congestion levels. The model architecture comprises an input layer with 32 neurons and ReLU activation, a hidden layer with 16 neurons and ReLU activation, and an output layer with 3 neurons and softmax activation to classify traffic congestion levels into three categories: Low, Medium, and High. The model is compiled with the Adam optimizer and categorical crossentropy loss function, and its performance is evaluated based on accuracy. In addition to the model training and evaluation, the program includes visualization code to analyze the distribution of traffic congestion levels and explore the dataset. The model can also make predictions based on new input data, making it a valuable tool for traffic management and analysis.

**Training Models:** Training a model involves feeding it data so it can learn patterns and make predictions. The training process usually consists of multiple steps:

1. **Data Preparation:** Clean and preprocess data to make it suitable for the model.
2. **Model Architecture:** Define the structure of the model (e.g., layers in a neural network).
3. **Compilation:** Choose an optimizer (e.g., Adam), loss function (e.g., categorical crossentropy), and evaluation metrics (e.g., accuracy).
4. **Training:** Use training data to adjust the model's weights through backpropagation and gradient descent.
5. **Validation:** Monitor the model’s performance on validation data to prevent overfitting.
6. **Evaluation:** Test the model on unseen data to assess its generalization ability.

**Modules:** Modules are reusable pieces of code that perform specific functions. In machine learning, several key modules are used:

1. **Data Preprocessing:** Tools for encoding, scaling, and splitting data (e.g., LabelEncoder, StandardScaler, train\_test\_split).
2. **Model Building:** Frameworks for defining and training models (e.g., TensorFlow, Keras).
3. **Optimization:** Algorithms for minimizing the loss function (e.g., Adam, SGD).
4. **Evaluation:** Functions to evaluate model performance (e.g., accuracy\_score, evaluate).
5. **Visualization:** Libraries for visualizing data and model performance (e.g., Matplotlib).

**PROGRAM**

!pip install tensorflow pandas numpy scikit-learn

# Import required libraries

!pip install matplotlib

import matplotlib.pyplot as plt

 import numpy as np

 import pandas as pd

 import tensorflow as tf

 from sklearn.model\_selection import train\_test\_split

 from sklearn.preprocessing import StandardScaler, LabelEncoder

 from tensorflow.keras.models import Sequential

 from tensorflow.keras.layers import Dense

 import matplotlib.pyplot as plt

 # Step 1: Create or Load Dataset

 data = {

     'Time': np.random.randint(0, 24, 1000), # Hour of the day

     'Location': np.random.randint(1, 5, 1000), # Location IDs (1-4)

     'Weather': np.random.choice(['Sunny', 'Rainy', 'Cloudy'], 1000), # Weather conditions

     'Congestion\_Level': np.random.choice([1, 2, 3], 1000) # 1=Low, 2=Medium, 3=High

 }

 df = pd.DataFrame(data)

 # Encode categorical data (Weather)

 encoder = LabelEncoder()

 df['Weather'] = encoder.fit\_transform(df['Weather']) # Sunny=2, Rainy=1, Cloudy=0

 # Features (X) and Target (y)

 X = df[['Time', 'Location', 'Weather']].values

 y = df['Congestion\_Level'].values

 # Normalize the features

 scaler = StandardScaler()

 X = scaler.fit\_transform(X)

 # One-hot encode the target for classification

 y = tf.keras.utils.to\_categorical(y - 1, num\_classes=3) # Convert to 0, 1, 2

 # Split the data into training and testing sets

 X\_train, X\_test, y\_train, y\_test = train\_test\_split(X, y, test\_size=0.2, random\_state=42)

 # Step 3: Build the Neural Network Model

 model = Sequential([

     Dense(32, input\_dim=3, activation='relu'), # Input layer (3 features)

     Dense(16, activation='relu'), # Hidden layer

     Dense(3, activation='softmax') # Output layer (3 classes)

 ])

 # Compile the model

 model.compile(optimizer='adam', loss='categorical\_crossentropy', metrics=['accuracy'])

 # Step 4: Train the Model

 history = model.fit(X\_train, y\_train, epochs=20, batch\_size=16, validation\_split=0.2)

 # Step 5: Evaluate the Model

 loss, accuracy = model.evaluate(X\_test, y\_test)

 print(f"Test Loss: {loss}")

 print(f"Test Accuracy: {accuracy}")

 # Step 6: Make Predictions

 example\_input = np.array([[8, 2, 0]]) # 8 AM, Location 2, Weather=Cloudy

 example\_input = scaler.transform(example\_input) # Normalize

 predicted = model.predict(example\_input)

 predicted\_class = np.argmax(predicted) + 1 # Convert back to class

 print(f"Predicted Traffic Congestion Level: {predicted\_class}")

 # Visualization Code

 # Plot Congestion Level Distribution

 plt.figure(figsize=(10, 6))

 plt.hist(df['Congestion\_Level'], bins=[1, 2, 3, 4], edgecolor='black', alpha=0.7)

 plt.xticks([1, 2, 3], ['Low', 'Medium', 'High'])

 plt.xlabel('Congestion Level')

 plt.ylabel('Frequency')

 plt.title('Distribution of Traffic Congestion Levels')

 plt.show()

 # Plot Congestion Level by Time of Day

 plt.figure(figsize=(10, 6))

 plt.scatter(df['Time'], df['Congestion\_Level'], c=df['Congestion\_Level'], cmap='viridis', alpha=0.7)

 plt.colorbar(label='Congestion Level')

 plt.xlabel('Time of Day')

 plt.ylabel('Congestion Level')

 plt.title('Traffic Congestion Level by Time of Day')

 plt.show()

 # Plot Congestion Level by Location and Weather

 plt.figure(figsize=(10, 6))

 plt.scatter(df['Location'], df['Weather'], c=df['Congestion\_Level'], cmap='viridis', alpha=0.7)

 plt.colorbar(label='Congestion Level')

 plt.xlabel('Location')

 plt.ylabel('Weather')

 plt.title('Traffic Congestion Level by Location and Weather')

 plt.show()

!pip install pandas openpyxl

import pandas as pd

# Load the dataset

file\_path = '/content/traffic.csv'  # Replace with your actual file path

df = pd.read\_csv(file\_path)

# Display the first few rows of the dataframe

print(df.head())

# Get a summary of the dataframe

print(df.info())

# Step 1: Import libraries

import pandas as pd

# Step 2: Load the dataset

file\_path = '/content/traffic.csv'  # Replace with your actual file path

df = pd.read\_csv(file\_path)

# Step 3: Explore the dataset

# Display the first few rows of the dataframe

print(df.head())

# Get a summary of the dataframe

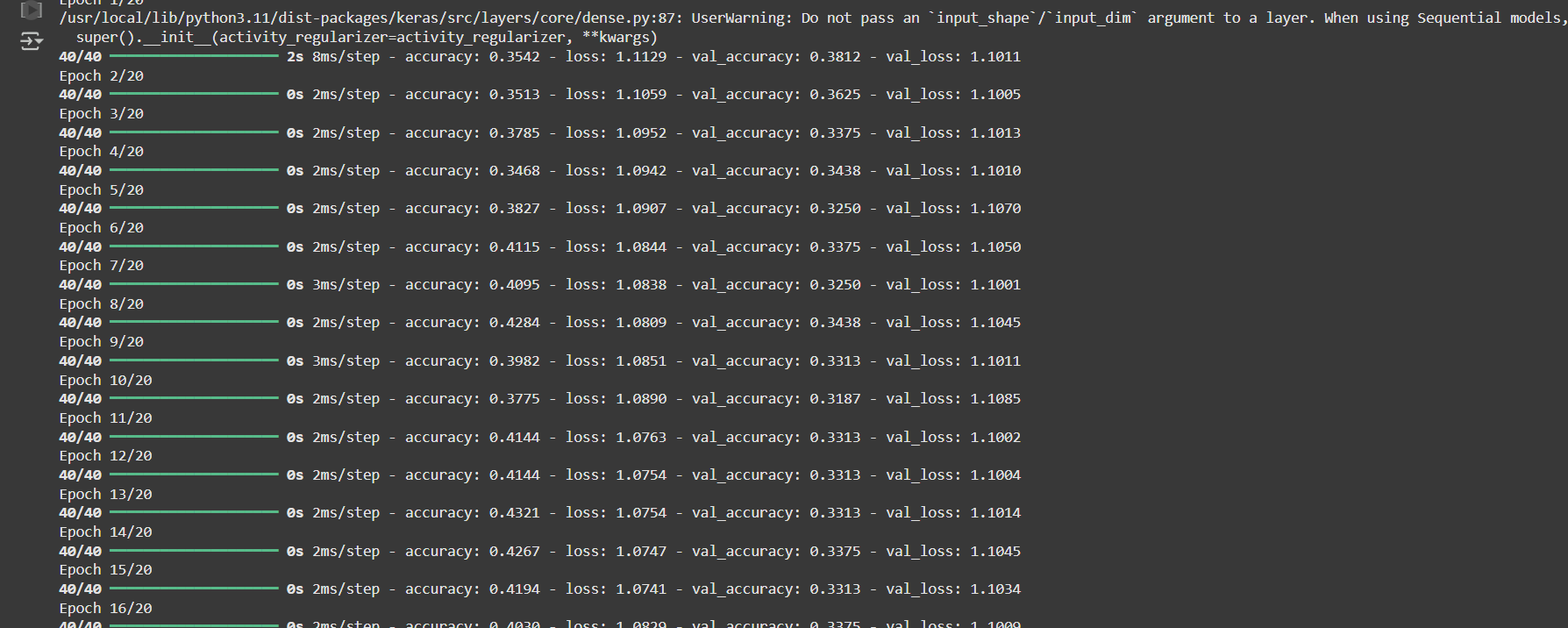
print(df.info())

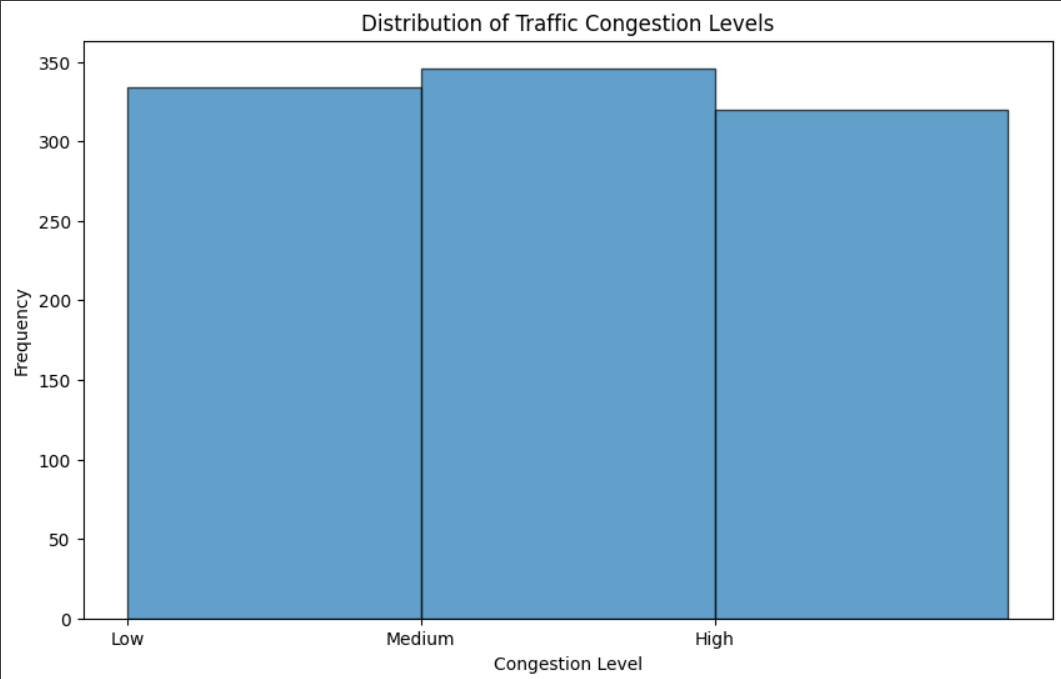
for chunk in pd.read\_csv(file\_path, chunksize=10000):

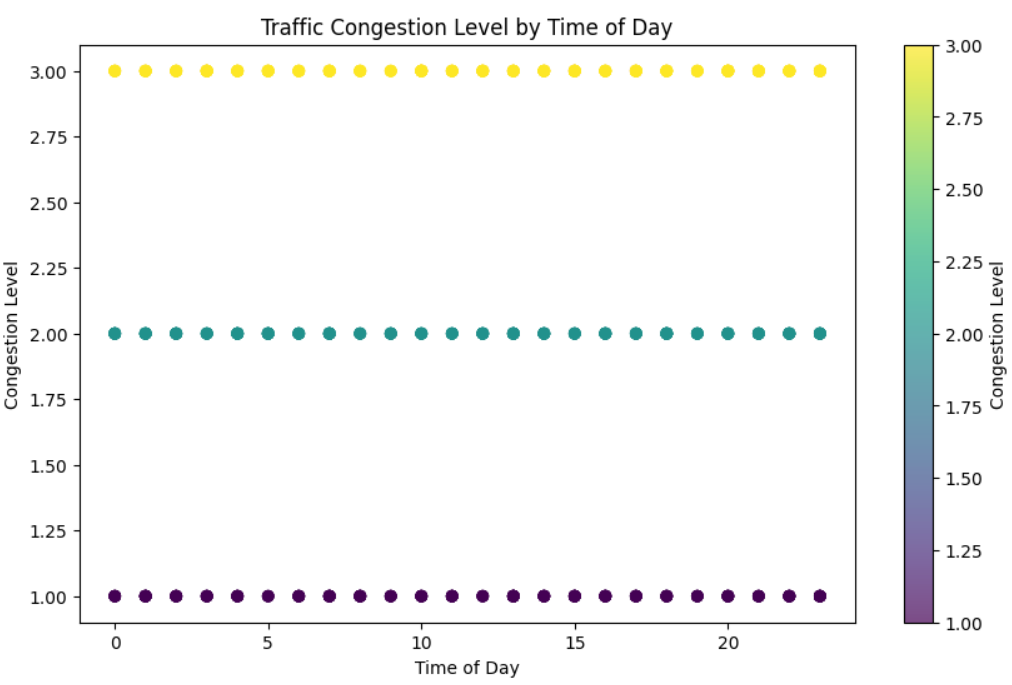
    # Process each chunk

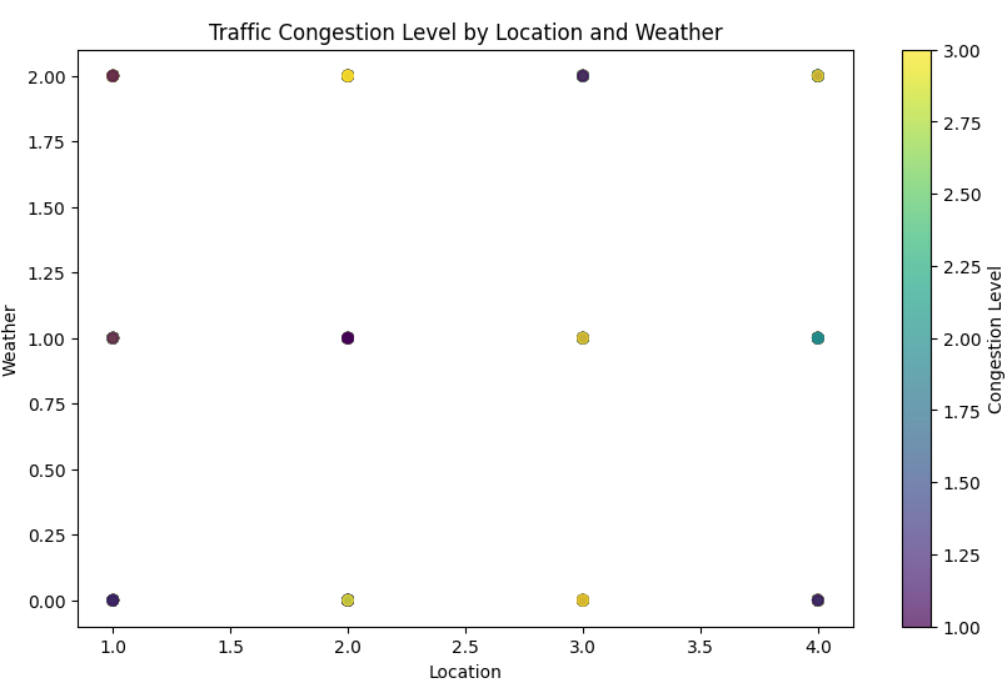
    print(chunk.head())

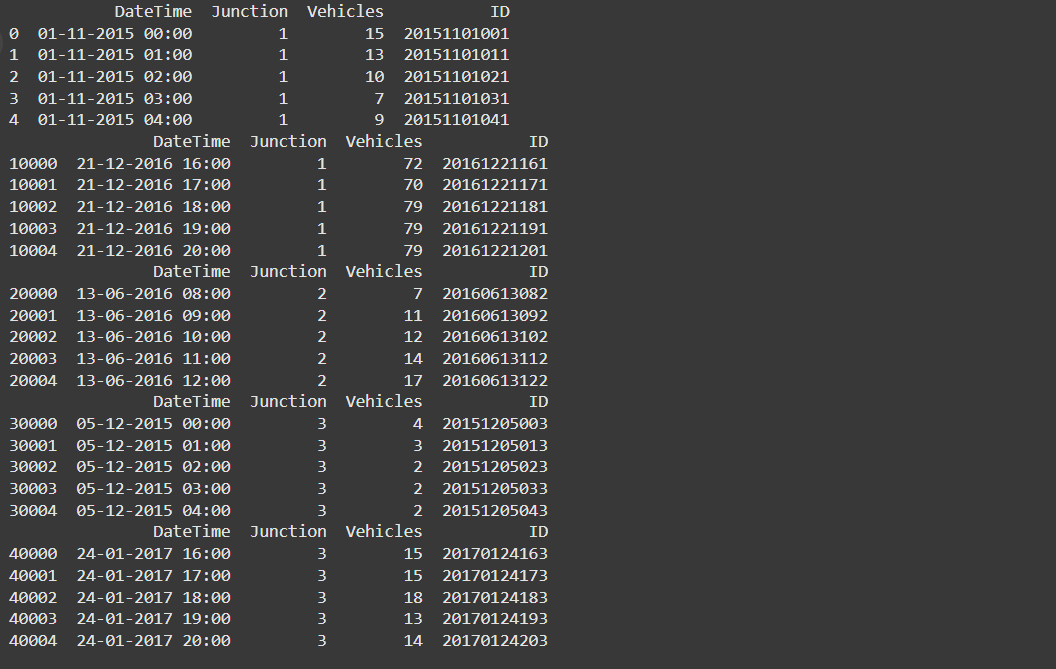
**OUTPUT**

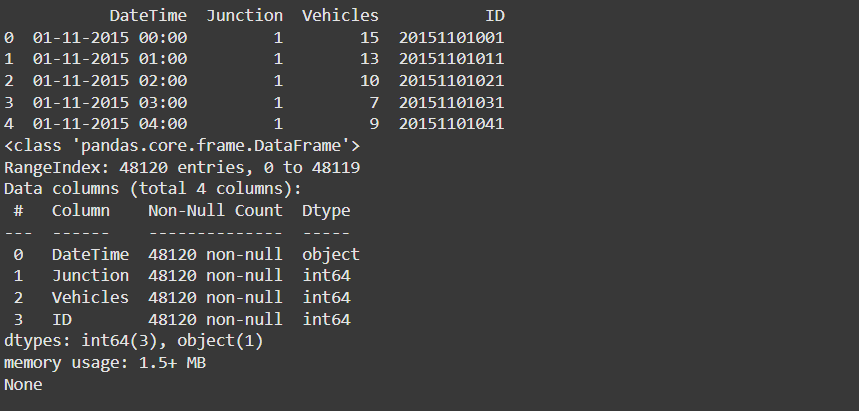
****

****

****

****

****

****

**GitHub Details**

Repository Link: [**https://github.com/Allen-Immanuel-David-R/Allen-Workspace/blob/12e42d7fe37877bd8fe8f63744b4ac2864f2f794/Traffic%20Prediction.ipynb**](https://github.com/Allen-Immanuel-David-R/Allen-Workspace/blob/12e42d7fe37877bd8fe8f63744b4ac2864f2f794/Traffic%20Prediction.ipynb)