COMPLETE SOURCE CODE:

```
pip install pandas numpy scikit-learn matplotlib seaborn tensorflow keras


import pandas as pd


# Load the dataset
file_path = "Traffic.csv"  # Change this if your file is in another location
df = pd.read_csv(file_path)


# Display the first few rows
df.head()


# Convert 'Time' to minutes from midnight
def time_to_minutes(time_str):
    return pd.to_datetime(time_str, format='%I:%M:%S %p').hour * 60 + pd.to_datetime(time_str, 
format='%I:%M:%S %p').minute


df['Minutes'] = df['Time'].apply(time_to_minutes)


# One-hot encode 'Day of the week'
df = pd.get_dummies(df, columns=['Day of the week'], drop_first=True)


# Sort data by Date and Time (Minutes) to maintain time series order
df = df.sort_values(by=['Date', 'Minutes'])


# Create lag features (Previous 15, 30, 45 minutes' traffic count)
lag_intervals = [15, 30, 45]
for lag in lag_intervals:
    df[f'Total_Lag_{lag}'] = df['Total'].shift(lag // 15)  # Assuming data is recorded every 15 mins
# Drop rows with NaN values (caused by shifting)
df.dropna(inplace=True)


# Display processed data
df.head()
```

```
from sklearn.model_selection import train_test_split

# Select features and target variable
features = ['Minutes', 'Day of the week_Monday', 'Day of the week_Saturday', 'Day of the week_Sunday',
        'Day of the week_Thursday', 'Day of the week_Tuesday', 'Day of the week_Wednesday',
        'Total_Lag_15', 'Total_Lag_30', 'Total_Lag_45']

target = 'Total'

# Split into 80% training and 20% testing
X_train, X_test, y_train, y_test = train_test_split(df[features], df[target], test_size=0.2, random_state=42, shuffle=False)

# Check the shape of train and test sets
X_train.shape, X_test.shape

from sklearn.linear_model import LinearRegression
from sklearn.metrics import mean_absolute_error, mean_squared_error, r2_score
```

LINEAR REGRESSION MODEL

```
# Train Linear Regression model
lr_model = LinearRegression()
lr_model.fit(X_train, y_train)

# Make predictions
y_pred_lr = lr_model.predict(X_test)

# Evaluate performance
mae_lr = mean_absolute_error(y_test, y_pred_lr)
rmse_lr = mean_squared_error(y_test, y_pred_lr, squared=False)
r2_lr = r2_score(y_test, y_pred_lr)

print(f"Linear Regression Performance:")
print(f"MAE: {mae_lr:.2f}")
print(f"RMSE: {rmse_lr:.2f}")
print(f"R² Score: {r2_lr:.2f}")
from sklearn.ensemble import RandomForestRegressor
```

RANDOM FOREST MODEL

```
# Train Random Forest model
rf_model = RandomForestRegressor(n_estimators=100, random_state=42)
rf_model.fit(X_train, y_train)

# Make predictions
y_pred_rf = rf_model.predict(X_test)

# Evaluate performance
mae_rf = mean_absolute_error(y_test, y_pred_rf)
rmse_rf = mean_squared_error(y_test, y_pred_rf, squared=False)
r2_rf = r2_score(y_test, y_pred_rf)

print(f"Random Forest Performance:")
print(f"MAE: {mae_rf:.2f}")
print(f"RMSE: {rmse_rf:.2f}")
print(f"R² Score: {r2_rf:.2f}")
import numpy as np
from sklearn.preprocessing import MinMaxScaler

# Scale the features to [0,1] range
scaler = MinMaxScaler()
X_train_scaled = scaler.fit_transform(X_train)
X_test_scaled = scaler.transform(X_test)

# Reshape input for LSTM (samples, time steps, features)
X_train_reshaped = np.reshape(X_train_scaled, (X_train_scaled.shape[0], 1,
X_train_scaled.shape[1]))
X_test_reshaped = np.reshape(X_test_scaled, (X_test_scaled.shape[0], 1, X_test_scaled.shape[1]))
```

```
# Check new shapes

X_train_reshaped.shape, X_test_reshaped.shape
```

LSTM MODEL

```
import tensorflow as tf

from tensorflow.keras.models import Sequential

from tensorflow.keras.layers import LSTM, Dense


# Build LSTM model

lstm_model = Sequential([

  LSTM(50, activation='relu', input_shape=(1, X_train_reshaped.shape[2])),

  Dense(1)  # Output layer

])


# Compile the model

lstm_model.compile(optimizer='adam', loss='mse')


# Train the model

history = lstm_model.fit(X_train_reshaped, y_train, epochs=20, batch_size=32,
validation_data=(X_test_reshaped, y_test))
```