🚖 Uber Trip Analysis

Internship Project Report

📌 Section 1: Project Overview

Project Title:

Uber Trip Analysis

Project Type:

Data Analysis & Machine Learning

Technologies Used:

✅ Python (Data Processing, Machine Learning, Web App)

✅ Flask (Web Application Development)

✅ Pandas & NumPy (Data Manipulation)

✅ Scikit-Learn (Machine Learning)

✅ Matplotlib, Seaborn (Data Visualization)

✅ Joblib (Model Saving & Deployment)

Project Difficulty Level:

✅ Advanced

📌 Section 2: Project Objective & Need

🎯 Objective

The aim of this project is to predict Uber trip demand based on historical ride-hailing data using machine learning models. This system helps:

🚕 Estimate demand for Uber trips at different times of the day

📊 Analyze ride-hailing trends using advanced data visualizations

📈 Assist businesses & drivers in making data-driven decisions on fleet distribution

🔍 Why is This Project Needed?

🚕 The demand for ride-hailing services fluctuates throughout the day.

📊 Accurate trip demand prediction helps Uber optimize its fleet for better efficiency.

📉 Prevents surge pricing & availability issues by forecasting peak ride demand.

🛣️ Helps city planners analyze urban mobility patterns for better infrastructure.

📌 Section 3: Dataset Information

📜 Dataset Source:

The dataset is obtained from Uber FOIL (January-February) trip records.

🗂️ Dataset Features (Columns Used for Prediction):

Feature Name	Description
Active Vehicles	Number of Uber vehicles available at a given time.
Hour	The hour of the day (0-23).
Day of the Week	The weekday (Monday-Sunday).
Month	The month of the year.
Weekend Flag	Indicates if the day is a weekend (1) or weekday (0).

📌 Target Variable (What We Predict):

✅ Number of Uber Trips

📌 Section 4: Project Execution - Steps & Implementation

Step 1: Data Collection

✓ Load the dataset using Pandas.

✓ Check for missing values, duplicates, and data types.

Step 2: Data Preprocessing

✓ Convert date column to datetime format.

✓ Extract hour, day of the week, and month from timestamps.

✓ Create a weekend flag (1 for weekends, 0 for weekdays).

✓ Handle missing values and normalize numerical features using StandardScaler.

Step 3: Exploratory Data Analysis (EDA)

✓ Trips per hour of the day (Bar Graph)

✓ Trips per day of the week (Grouped Bar Chart)

✓ Trips vs Active Vehicles (Scatter Plot)

✓ Hourly Trip Demand Heatmap

Step 4: Feature Engineering

✓ Select relevant features:

Active Vehicles

Hour

Day of the Week

Month

Weekend Flag

✓ Scale numerical features using StandardScaler for better ML performance.

Step 5: Model Building

✓ Train a Linear Regression model for trip prediction.

✓ Train a Random Forest model for better accuracy.

✓ Use GridSearchCV for hyperparameter tuning.

Step 6: Model Evaluation

✓ Use Mean Absolute Error (MAE), Mean Squared Error (MSE), and R^2 Score to measure accuracy.

✓ Compare actual vs. predicted values using a scatter plot.

Step 7: Data Visualization

✓ Pie Chart → Distribution of trips vs active vehicles.

✓ Bar Graph → Comparison of predicted trips & active vehicles.

✓ Line Chart → Trend analysis of trips over 24 hours.

✓ Scatter Plot → Relationship between active vehicles & trip demand.

✓ Heatmap → Visualizing hourly trip demand patterns.

Step 8: Flask Web Application

✓ Built a Flask web app for users to input date, time, and active vehicles.

✓ Predicts trip demand dynamically based on user input.

✓ Displays related visualizations (bar charts, pie charts, line charts, heatmaps).

✓ Fully responsive & modern UI (Dark Theme with Gold Accents).

📌 Section 5: Results & Conclusion

✅ Key Achievements

🔹 Successfully built a machine learning model to predict Uber trip demand.

🔹 Integrated real-time interactive visualizations (pie charts, bar graphs, line charts).

🔹 Developed a Flask-powered web app for seamless user interaction.

🔹 Created a modern, responsive UI/UX with dark mode & gold accents.

🔍 Future Enhancements

✓ Improve accuracy using XGBoost & Deep Learning models.

✓ Add real-time Uber API integration for dynamic trip predictions.

✓ Deploy on Cloud (AWS, Heroku, or Streamlit) for public access.

Uber-Trip-Analysis

```python
# Import necessary libraries
import pandas as pd  # For data handling
import numpy as np  # For numerical operations
import matplotlib.pyplot as plt  # For visualization
import seaborn as sns  # For better plots
from datetime import datetime  # For date-time processing

# Load the dataset
file_path = "Uber-Jan-Feb-FOIL.csv"  # Replace with actual file path
df = pd.read_csv(file_path)

# Display the first few rows
print("First 5 rows of dataset:")
print(df.head())

# Display dataset info
print("\nDataset Information:")
print(df.info())

# Check for missing values
print("\nMissing values in each column:")
print(df.isnull().sum())
```

```
First 5 rows of dataset:
  dispatching_base_number      date  active_vehicles  trips
0                  B02512  1/1/2015              190   1132
1                  B02765  1/1/2015              225   1765
2                  B02764  1/1/2015             3427  29421
3                  B02682  1/1/2015              945   7679
4                  B02617  1/1/2015             1228   9537

Dataset Information:
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 354 entries, 0 to 353
Data columns (total 4 columns):
 #   Column                  Non-Null Count  Dtype
---  ------                  --------------  -----
 0   dispatching_base_number  354 non-null    object
 1   date                    354 non-null    object
 2   active_vehicles         354 non-null    int64
 3   trips                   354 non-null    int64
dtypes: int64(2), object(2)
memory usage: 11.2+ KB
None

Missing values in each column:
```

```
dispatching_base_number    0
date                       0
active_vehicles            0
trips                      0
dtype: int64
```

```
print(df.columns)
```

```
Index(['dispatching_base_number', 'date', 'active_vehicles', 'trips'],
dtype='object')
```

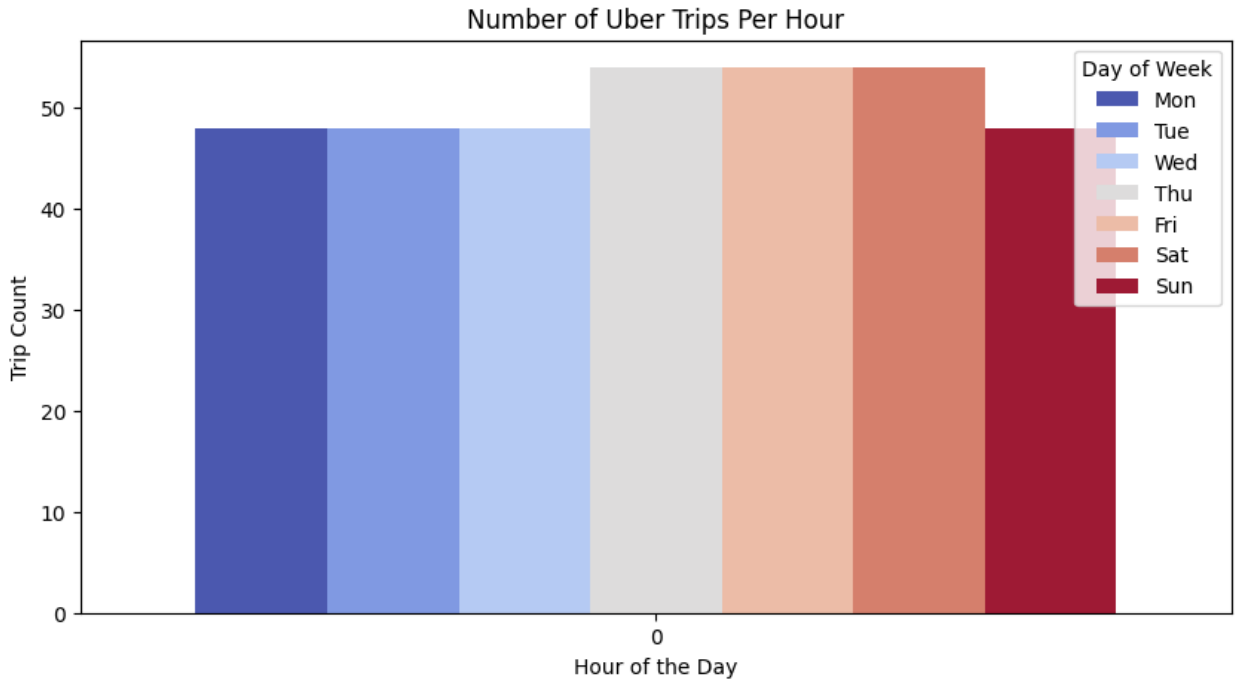
```
df['date'] = pd.to_datetime(df['date'])
print(df.info())  # Verify the new format
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 354 entries, 0 to 353
Data columns (total 4 columns):
 #   Column                  Non-Null Count  Dtype
---  ------                  --------------  -----
 0   dispatching_base_number  354 non-null    object
 1   date                    354 non-null    datetime64[ns]
 2   active_vehicles         354 non-null    int64
 3   trips                   354 non-null    int64
dtypes: datetime64[ns](1), int64(2), object(1)
memory usage: 11.2+ KB
None
```

```
df['Hour'] = df['date'].dt.hour
df['Day'] = df['date'].dt.day
df['DayOfWeek'] = df['date'].dt.dayofweek  # 0 = Monday, 6 = Sunday
df['Month'] = df['date'].dt.month
```

Plot number of trips per hour, colored by day of the week

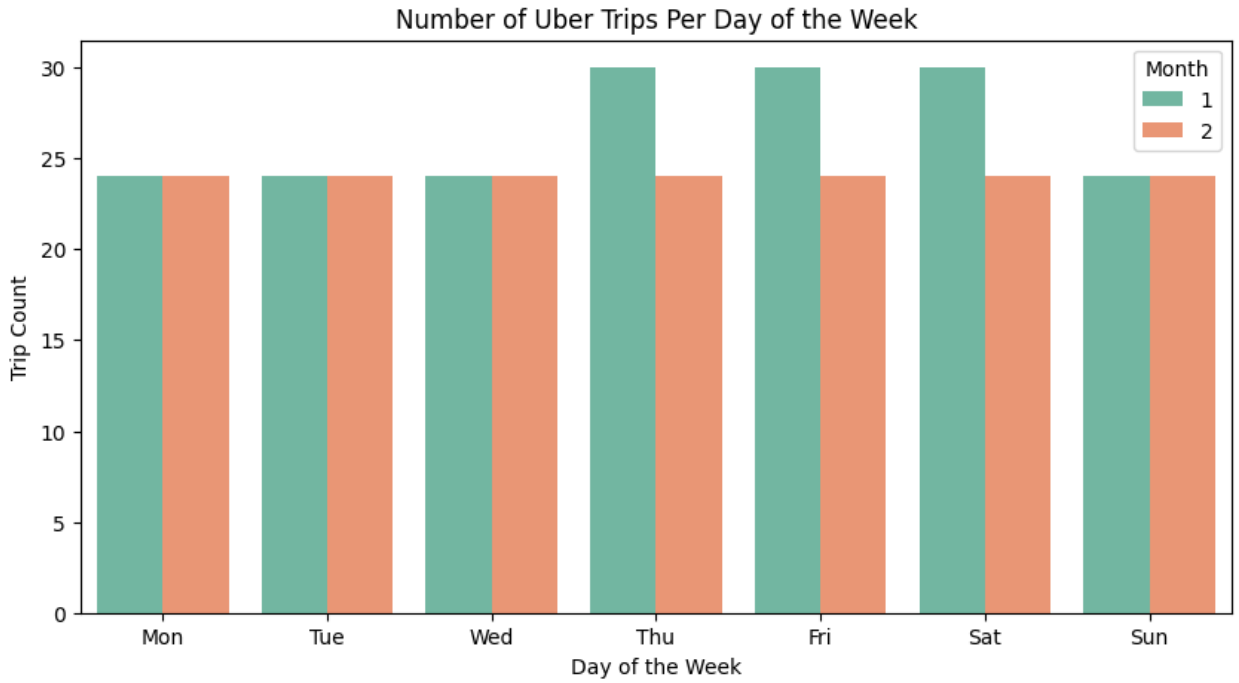
```
plt.figure(figsize=(10, 5))
sns.countplot(x=df['Hour'], hue=df['DayOfWeek'], palette="coolwarm")
plt.title("Number of Uber Trips Per Hour")
plt.xlabel("Hour of the Day")
plt.ylabel("Trip Count")
plt.legend(title="Day of Week", labels=["Mon", "Tue", "Wed", "Thu",
"Fri", "Sat", "Sun"])
plt.show()
```




```
# Check the column names in the dataset
print(df.columns)

Index(['dispatching_base_number', 'date', 'active_vehicles', 'trips',
'Hour',
       'Day', 'DayOfWeek', 'Month'],
      dtype='object')
```

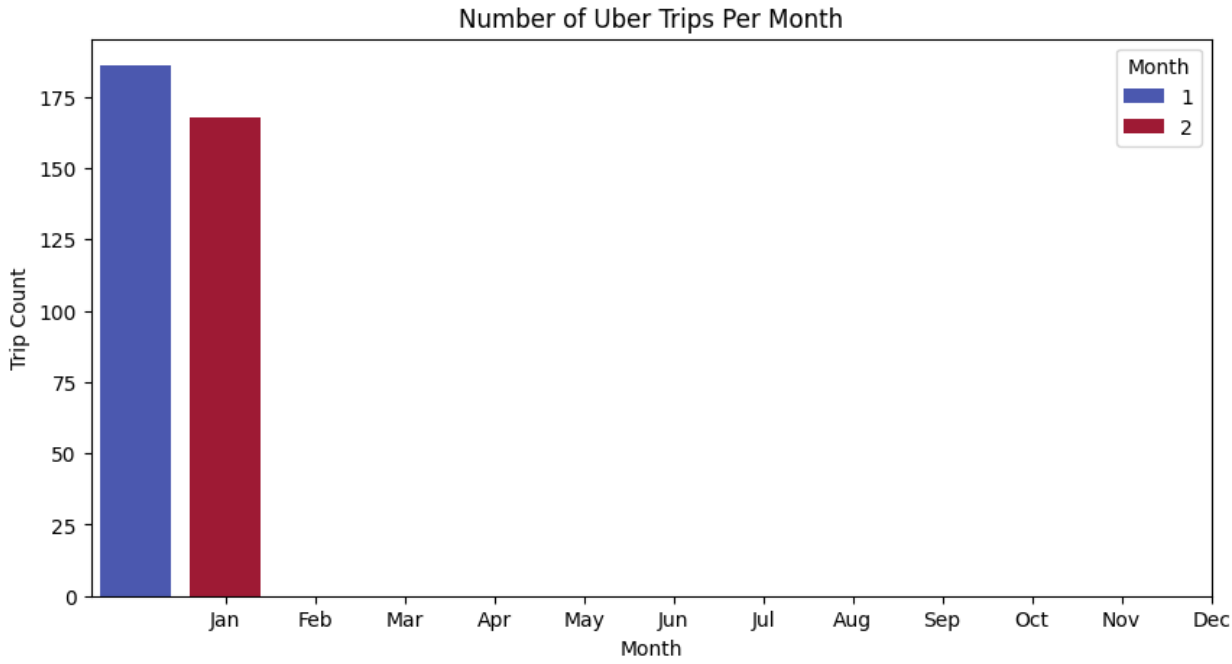
Plot number of trips per day of the week, colored by Month (for example)

```
plt.figure(figsize=(10, 5))
sns.countplot(x=df['DayOfWeek'], hue=df['Month'], palette="Set2")
plt.title("Number of Uber Trips Per Day of the Week")
plt.xlabel("Day of the Week")
plt.ylabel("Trip Count")
plt.xticks(ticks=np.arange(7), labels=["Mon", "Tue", "Wed", "Thu",
"Fri", "Sat", "Sun"])
plt.show()
```



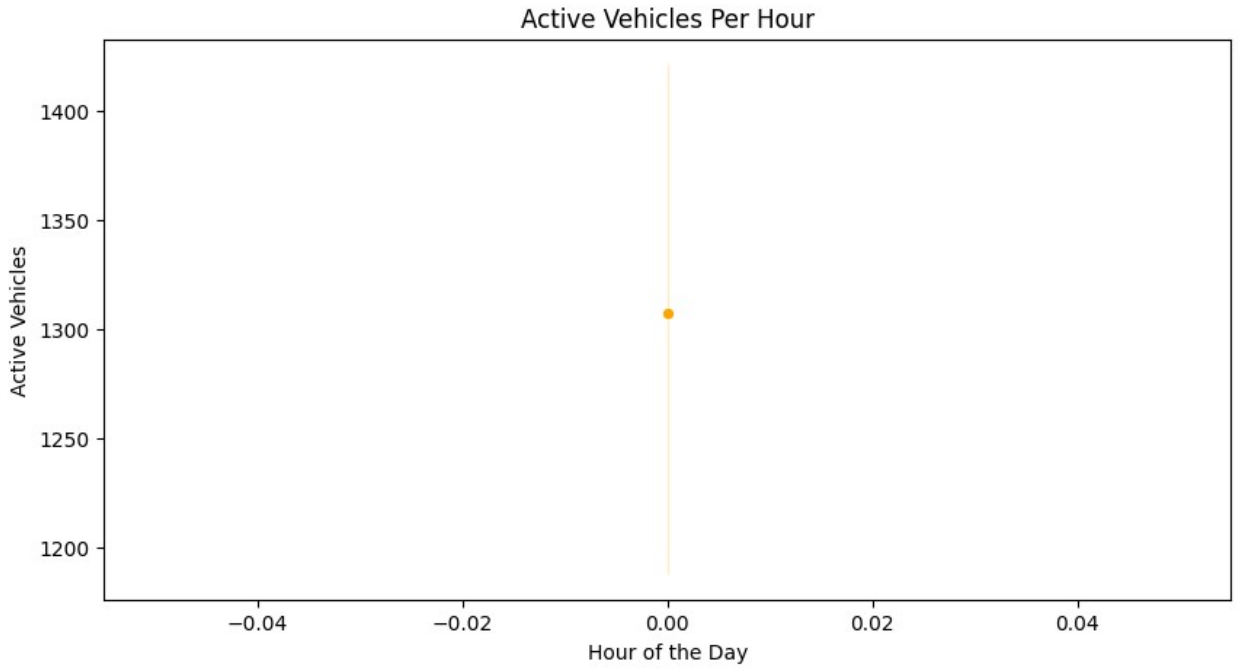
Plot number of trips per month with hue for coloring (if you want categories)

```
plt.figure(figsize=(10, 5))
sns.countplot(x=df['Month'], hue=df['Month'], palette="coolwarm")
plt.title("Number of Uber Trips Per Month")
plt.xlabel("Month")
plt.ylabel("Trip Count")
plt.xticks(ticks=np.arange(1, 13), labels=["Jan", "Feb", "Mar", "Apr",
"May", "Jun", "Jul", "Aug", "Sep", "Oct", "Nov", "Dec"])
plt.show()
```



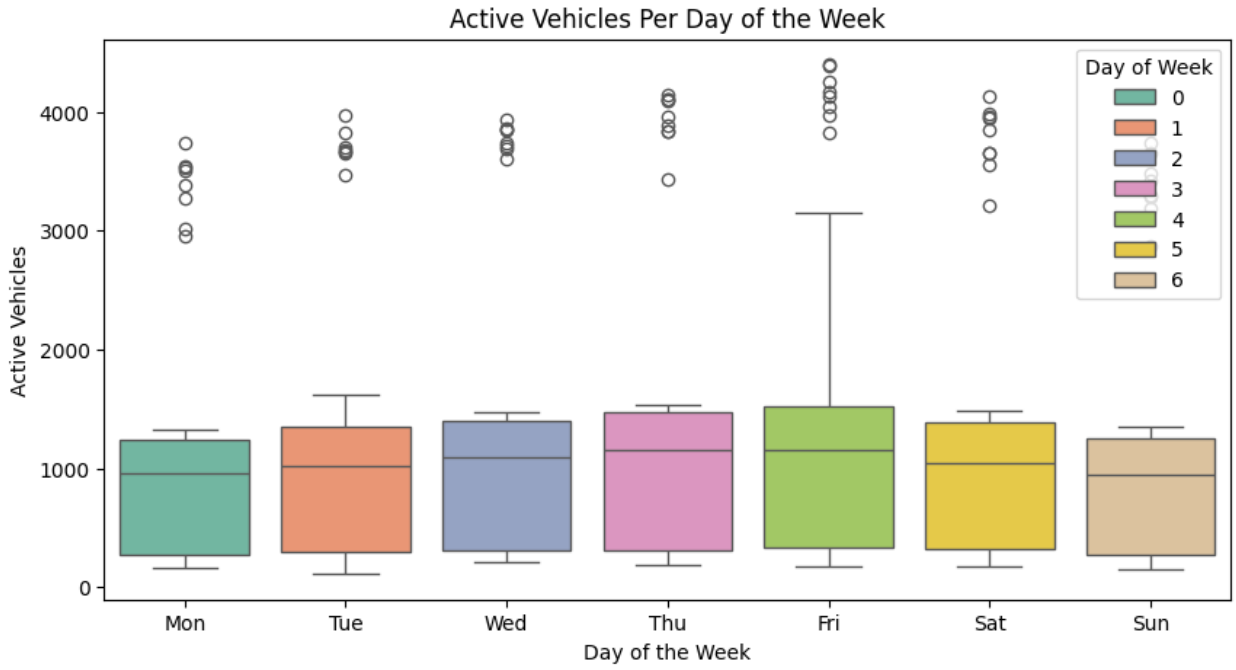
Plot active vehicles per hour

```
plt.figure(figsize=(10, 5))
sns.lineplot(x=df['Hour'], y=df['active_vehicles'], marker='o',
color='orange')
plt.title("Active Vehicles Per Hour")
plt.xlabel("Hour of the Day")
plt.ylabel("Active Vehicles")
plt.show()
```



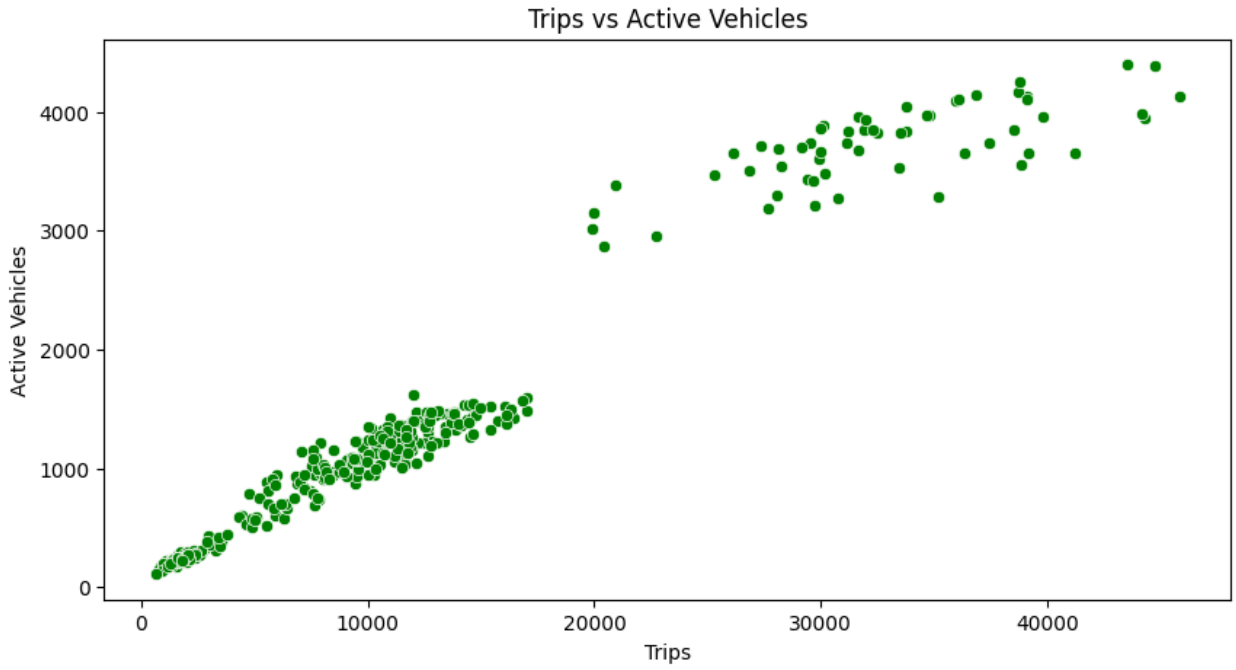
Plot active vehicles per day of the week with hue for coloring

```
plt.figure(figsize=(10, 5))
sns.boxplot(x=df['DayOfWeek'], y=df['active_vehicles'],
hue=df['DayOfWeek'], palette="Set2")
plt.title("Active Vehicles Per Day of the Week")
plt.xlabel("Day of the Week")
plt.ylabel("Active Vehicles")
plt.xticks(ticks=np.arange(7), labels=["Mon", "Tue", "Wed", "Thu",
"Fri", "Sat", "Sun"])
plt.legend(title="Day of Week")
plt.show()
```



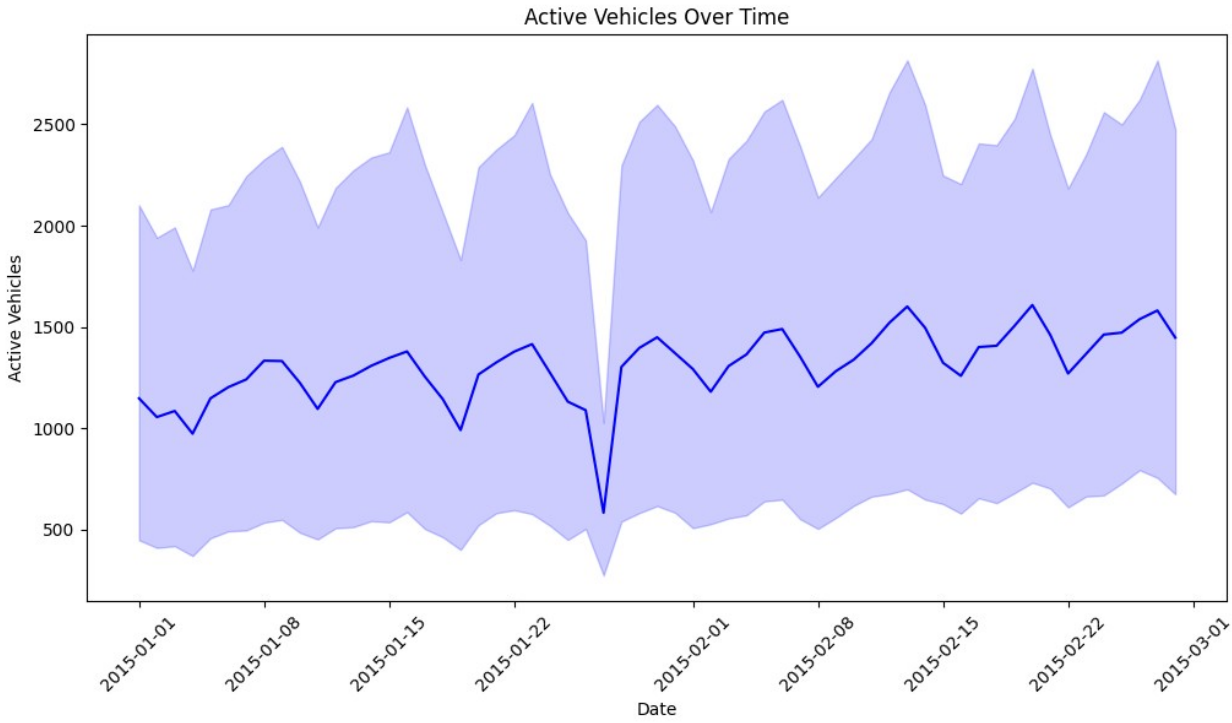
Plot trips vs active vehicles

```
plt.figure(figsize=(10, 5))
sns.scatterplot(x=df['trips'], y=df['active_vehicles'], color='green')
plt.title("Trips vs Active Vehicles")
plt.xlabel("Trips")
plt.ylabel("Active Vehicles")
plt.show()
```

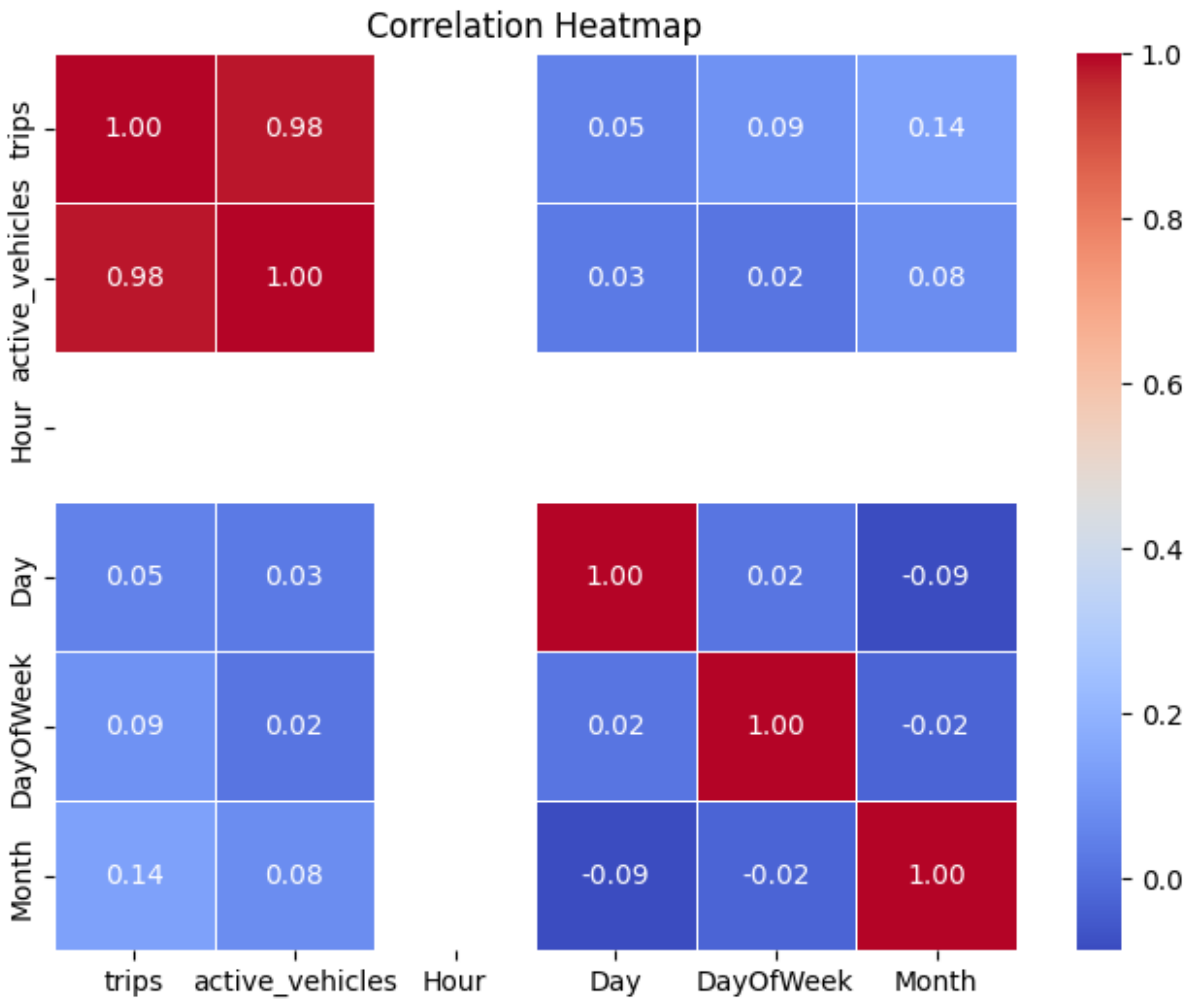


Plot active vehicles trend over time (by date)

```
plt.figure(figsize=(12, 6))
sns.lineplot(x=df['date'], y=df['active_vehicles'], color='blue')
plt.title("Active Vehicles Over Time")
plt.xlabel("Date")
plt.ylabel("Active Vehicles")
plt.xticks(rotation=45)
plt.show()
```




```
# Compute correlations between numerical features
correlation_matrix = df[['trips', 'active_vehicles', 'Hour', 'Day',
'DayOfWeek', 'Month']].corr()

# Plot the heatmap
plt.figure(figsize=(8, 6))
sns.heatmap(correlation_matrix, annot=True, cmap="coolwarm",
fmt=".2f", linewidths=0.5)
plt.title("Correlation Heatmap")
plt.show()
```




```
# Add weekday name
df['Weekday'] = df['date'].dt.strftime('%A')  # Converts DayOfWeek to
full weekday name (e.g., Monday, Tuesday)

# Add Weekend flag (1 for weekend, 0 for weekday)
df['Weekend'] = df['DayOfWeek'].apply(lambda x: 1 if x >= 5 else 0)

# Display the first few rows to verify
print(df[['DayOfWeek', 'Weekday', 'Weekend']].head())
```

```
   DayOfWeek   Weekday  Weekend
0          3  Thursday        0
1          3  Thursday        0
2          3  Thursday        0
3          3  Thursday        0
4          3  Thursday        0
```

```
from sklearn.preprocessing import StandardScaler
```

```
# Initialize the scaler
scaler = StandardScaler()

# Scale the 'trips' and 'active_vehicles' columns
df['scaled_trips'] = scaler.fit_transform(df[['trips']])
df['scaled_active_vehicles'] =
scaler.fit_transform(df[['active_vehicles']])

# Display the first few rows to verify
print(df[['trips', 'scaled_trips', 'active_vehicles',
'scaled_active_vehicles']].head())
```

```
   trips  scaled_trips  active_vehicles  scaled_active_vehicles
0   1132     -0.990791              190               -0.962586
1   1765     -0.931261              225               -0.932436
2  29421      1.669641             3427                1.825846
3   7679     -0.375080              945               -0.312211
4   9537     -0.200345             1228               -0.068427
```

Predictive Modeling

```
from sklearn.model_selection import train_test_split

# Select features and target
X = df[['active_vehicles', 'Hour', 'DayOfWeek', 'Month', 'Weekend']]
# Features
y = df['trips']  # Target (number of trips)

# Split the data into training and testing sets (80% train, 20% test)
X_train, X_test, y_train, y_test = train_test_split(X, y,
test_size=0.2, random_state=42)

# Display the shape of training and testing sets
print(f"Training Set Size: {X_train.shape[0]}")
print(f"Testing Set Size: {X_test.shape[0]}")
```

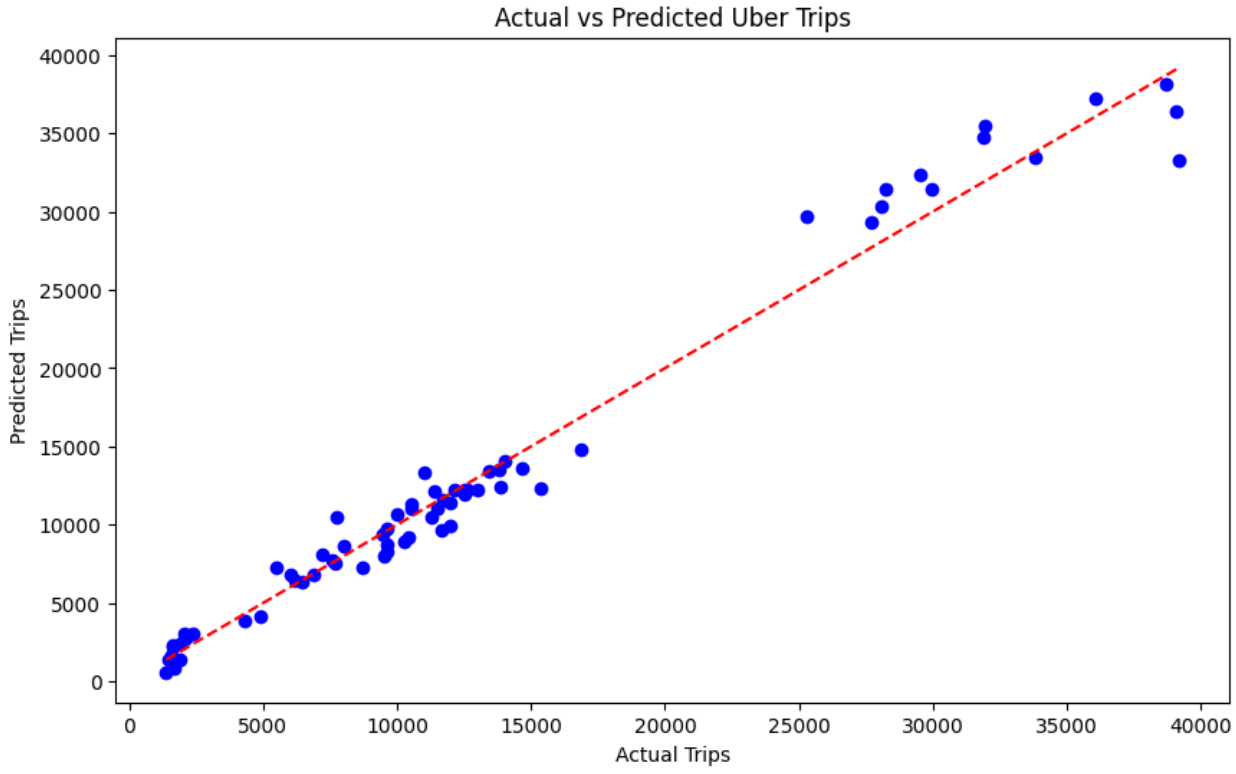
```
Training Set Size: 283
Testing Set Size: 71
```

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

# Initialize the model
model = LinearRegression()

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

```python
# Predict on the test set
y_pred = model.predict(X_test)

# Display the model's performance metrics
print("Model Performance:")
print(f"Mean Absolute Error (MAE): {mean_absolute_error(y_test,
y_pred)}")
print(f"Mean Squared Error (MSE): {mean_squared_error(y_test,
y_pred)}")
print(f"R-squared: {r2_score(y_test, y_pred)}")
```

```
Model Performance:
Mean Absolute Error (MAE): 1111.8851978507662
Mean Squared Error (MSE): 2487470.123943445
R-squared: 0.9769314775935857
```

```python
# Plot actual vs predicted trips
plt.figure(figsize=(10, 6))
plt.scatter(y_test, y_pred, color='blue')
plt.plot([min(y_test), max(y_test)], [min(y_test), max(y_test)],
color='red', linestyle='--')  # 45-degree line
plt.title("Actual vs Predicted Uber Trips")
plt.xlabel("Actual Trips")
plt.ylabel("Predicted Trips")
plt.show()
```



```python
from sklearn.ensemble import RandomForestRegressor
from sklearn.model_selection import GridSearchCV

# Initialize the model
rf_model = RandomForestRegressor(random_state=42)

# Define hyperparameters for tuning
param_grid = {
    'n_estimators': [100, 200, 300],
    'max_depth': [10, 20, None],
    'min_samples_split': [2, 5],
    'min_samples_leaf': [1, 2],
}

# Initialize GridSearchCV
grid_search = GridSearchCV(estimator=rf_model, param_grid=param_grid,
cv=5, scoring='neg_mean_absolute_error', n_jobs=-1, verbose=2)

# Fit the model
grid_search.fit(X_train, y_train)

# Best hyperparameters
print("Best hyperparameters:", grid_search.best_params_)

# Predict on the test set using the best model
best_rf_model = grid_search.best_estimator_
y_pred_rf = best_rf_model.predict(X_test)

# Evaluate the model
print("Random Forest Model Performance:")
print(f"MAE: {mean_absolute_error(y_test, y_pred_rf)}")
print(f"MSE: {mean_squared_error(y_test, y_pred_rf)}")
print(f"R-squared: {r2_score(y_test, y_pred_rf)}")
```

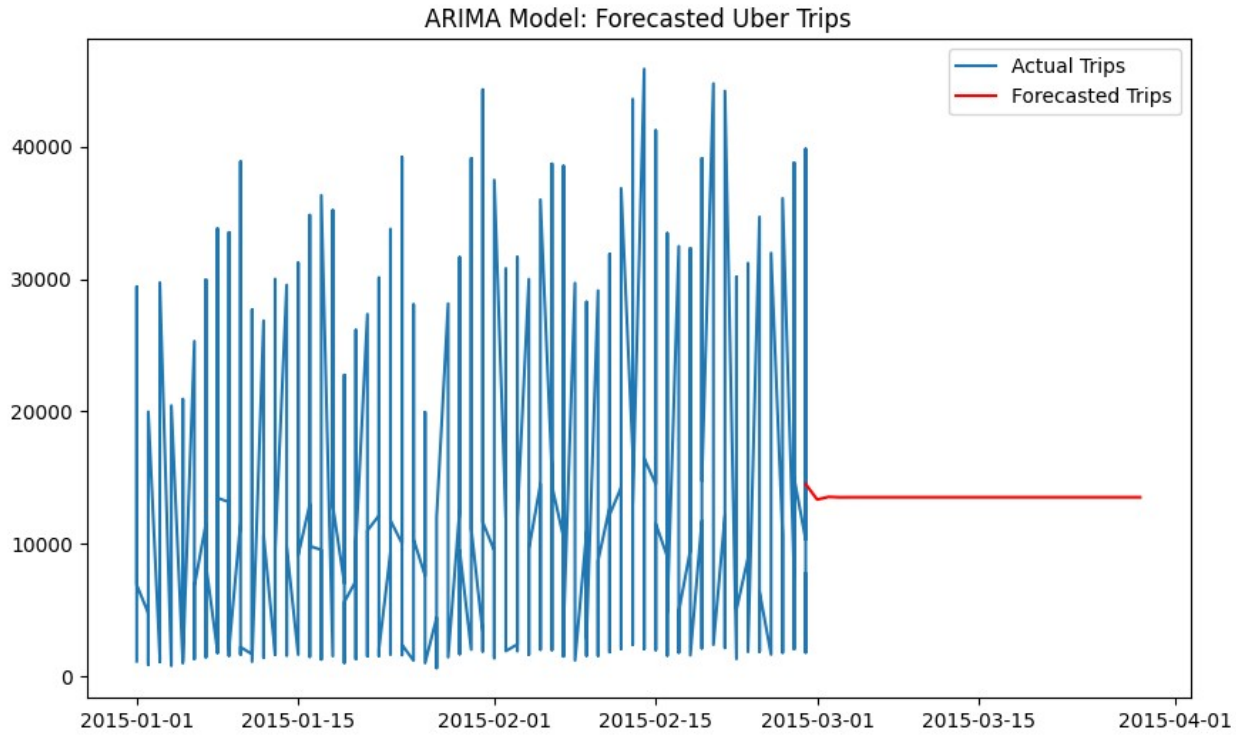
```
Fitting 5 folds for each of 36 candidates, totalling 180 fits
Best hyperparameters: {'max_depth': 10, 'min_samples_leaf': 1,
'min_samples_split': 2, 'n_estimators': 300}
Random Forest Model Performance:
MAE: 925.6039423517923
MSE: 1762135.0763385603
R-squared: 0.9836581544838008
```

```python
from sklearn.model_selection import cross_val_score

# Evaluate model using cross-validation
cv_scores = cross_val_score(model, X, y, cv=5,
scoring='neg_mean_absolute_error')
print(f"Mean MAE from cross-validation: {-cv_scores.mean()}")
```

```
Mean MAE from cross-validation: 1243.8186067973807
```

```python
from statsmodels.tsa.arima.model import ARIMA
import matplotlib.pyplot as plt

# Prepare time series data (using 'date' as the index)
df.set_index('date', inplace=True)

# Fit ARIMA model (adjust parameters based on your data)
arima_model = ARIMA(df['trips'], order=(1,1,1))  # (p,d,q) parameters
arima_model_fit = arima_model.fit()

# Make predictions
forecast = arima_model_fit.forecast(steps=30)  # Forecast the next 30
days

# Plot the forecasted trips
plt.figure(figsize=(10,6))
plt.plot(df.index, df['trips'], label='Actual Trips')
plt.plot(pd.date_range(df.index[-1], periods=30, freq='D'), forecast,
label='Forecasted Trips', color='red')
plt.legend()
plt.title('ARIMA Model: Forecasted Uber Trips')
plt.show()
```

```
C:\Users\91834\AppData\Local\Programs\Python\Python310\lib\site-
packages\statsmodels\tsa\base\tsa_model.py:473: ValueWarning: A date
index has been provided, but it has no associated frequency
information and so will be ignored when e.g. forecasting.
  self._init_dates(dates, freq)
C:\Users\91834\AppData\Local\Programs\Python\Python310\lib\site-
packages\statsmodels\tsa\base\tsa_model.py:473: ValueWarning: A date
index has been provided, but it has no associated frequency
information and so will be ignored when e.g. forecasting.
  self._init_dates(dates, freq)
C:\Users\91834\AppData\Local\Programs\Python\Python310\lib\site-
packages\statsmodels\tsa\base\tsa_model.py:473: ValueWarning: A date
index has been provided, but it has no associated frequency
information and so will be ignored when e.g. forecasting.
  self._init_dates(dates, freq)
C:\Users\91834\AppData\Local\Programs\Python\Python310\lib\site-
packages\statsmodels\tsa\base\tsa_model.py:837: ValueWarning: No
supported index is available. Prediction results will be given with an
integer index beginning at `start`.
  return get_prediction_index(
C:\Users\91834\AppData\Local\Programs\Python\Python310\lib\site-
packages\statsmodels\tsa\base\tsa_model.py:837: FutureWarning: No
supported index is available. In the next version, calling this method
in a model without a supported index will result in an exception.
  return get_prediction_index(
```




```
import joblib

# Save the trained Random Forest model (or any model)
joblib.dump(best_rf_model, 'uber_trip_predictor_model.pkl')
```

```
['uber_trip_predictor_model.pkl']
```

Flask App.py

```
from flask import Flask, render_template, request
import pandas as pd
import joblib
import matplotlib.pyplot as plt
import seaborn as sns
import io
import base64
import numpy as np
from datetime import datetime

# Initialize Flask app
app = Flask(__name__)

# Load the trained model
model = joblib.load("uber_trip_predictor_model.pkl")

def preprocess_input(date_time, active_vehicles):
    """Extracts features from date and time input."""
    date_time = pd.to_datetime(date_time)
    hour = date_time.hour
    day_of_week = date_time.weekday()
    month = date_time.month
    weekend = 1 if day_of_week >= 5 else 0  # 1 for weekend, 0 for
weekdays
    return [[active_vehicles, hour, day_of_week, month, weekend]]

def generate_visualizations(active_vehicles, prediction, date_time):
    """Generate multiple visualizations including pie chart, bar
graph, line chart, scatter plot, and heatmap."""
    img_pie, img_bar, img_line, img_scatter, img_heatmap =
io.BytesIO(), io.BytesIO(), io.BytesIO(), io.BytesIO(), io.BytesIO()

    # Pie chart
    labels = ['Predicted Trips', 'Remaining Capacity']
    sizes = [prediction, max(0, active_vehicles - prediction)]
    colors = ['#ff9999','#66b3ff']
    plt.figure(figsize=(5, 5))
    plt.pie(sizes, labels=labels, autopct='%1.1f%%', colors=colors,
startangle=90)
    plt.axis('equal')
    plt.savefig(img_pie, format='png')
    img_pie.seek(0)
    pie_url = base64.b64encode(img_pie.getvalue()).decode('utf8')

    # Bar graph
    plt.figure(figsize=(6, 4))
```

```
    sns.barplot(x=['Predicted Trips', 'Active Vehicles'],
y=[prediction, active_vehicles], palette=['#ff9999', '#66b3ff'])
    plt.ylabel('Count')
    plt.title('Trips vs Active Vehicles')
    plt.savefig(img_bar, format='png')
    img_bar.seek(0)
    bar_url = base64.b64encode(img_bar.getvalue()).decode('utf8')

    # Line Chart (Trip Trends Over Time)
    hours = np.arange(24)
    trip_predictions = [model.predict([[active_vehicles, h,
date_time.weekday(), date_time.month, 1 if date_time.weekday() >= 5
else 0]])[0] for h in hours]
    plt.figure(figsize=(6, 4))
    plt.plot(hours, trip_predictions, marker='o', linestyle='-',
color='#FFA500')
    plt.xlabel('Hour of the Day')
    plt.ylabel('Predicted Trips')
    plt.title('Predicted Trip Trend Over 24 Hours')
    plt.savefig(img_line, format='png')
    img_line.seek(0)
    line_url = base64.b64encode(img_line.getvalue()).decode('utf8')

    # Scatter Plot (Active Vehicles vs. Predicted Trips)
    plt.figure(figsize=(6, 4))
    sns.scatterplot(x=[active_vehicles]*24, y=trip_predictions,
color='#FF4500')
    plt.xlabel('Active Vehicles')
    plt.ylabel('Predicted Trips')
    plt.title('Active Vehicles vs. Predicted Trips')
    plt.savefig(img_scatter, format='png')
    img_scatter.seek(0)
    scatter_url =
base64.b64encode(img_scatter.getvalue()).decode('utf8')

    # Heatmap (Hourly Trip Distribution)
    data = np.array(trip_predictions).reshape(1, -1)
    plt.figure(figsize=(8, 2))
    sns.heatmap(data, annot=True, fmt='.0f', cmap='coolwarm',
xticklabels=hours, yticklabels=['Predicted Trips'])
    plt.xlabel('Hour of the Day')
    plt.title('Heatmap of Predicted Trips')
    plt.savefig(img_heatmap, format='png')
    img_heatmap.seek(0)
    heatmap_url =
base64.b64encode(img_heatmap.getvalue()).decode('utf8')

    return pie_url, bar_url, line_url, scatter_url, heatmap_url
```

```
@app.route("/", methods=["GET", "POST"])
def index():
    if request.method == "POST":
        date_time = request.form["date_time"]
        active_vehicles = request.form.get("active_vehicles", 1000)  # 
Default value

        try:
            active_vehicles = int(active_vehicles)
        except ValueError:
            active_vehicles = 1000

        # Convert input to model-ready format
        input_features = preprocess_input(date_time, active_vehicles)
        prediction = model.predict(input_features)[0]

        # Generate visualizations
        pie_chart, bar_chart, line_chart, scatter_plot, heatmap = 
generate_visualizations(active_vehicles, int(prediction), 
pd.to_datetime(date_time))

        return render_template("result.html", date_time=date_time, 
active_vehicles=active_vehicles, prediction=int(prediction), 
pie_chart=pie_chart, bar_chart=bar_chart, line_chart=line_chart, 
scatter_plot=scatter_plot, heatmap=heatmap)

    return render_template("index.html")

if __name__ == "__main__":
    app.run(debug=True, host="0.0.0.0", port=8080, use_reloader=False)
```

```
 * Serving Flask app '__main__'
 * Debug mode: on
WARNING: This is a development server. Do not use it in a production 
deployment. Use a production WSGI server instead.
 * Running on all addresses (0.0.0.0)
 * Running on http://127.0.0.1:8080
 * Running on http://192.168.1.6:8080
Press CTRL+C to quit
192.168.1.6 - - [05/Feb/2025 21:41:18] "GET / HTTP/1.1" 200 -
192.168.1.6 - - [05/Feb/2025 21:41:20] "GET /favicon.ico HTTP/1.1" 404 
-
C:\Users\91834\AppData\Local\Programs\Python\Python310\lib\site-
packages\sklearn\base.py:493: UserWarning: X does not have valid 
feature names, but RandomForestRegressor was fitted with feature names
  warnings.warn(
C:\Users\91834\AppData\Local\Temp\ipykernel_10968\3605251356.py:43: 
FutureWarning:
```

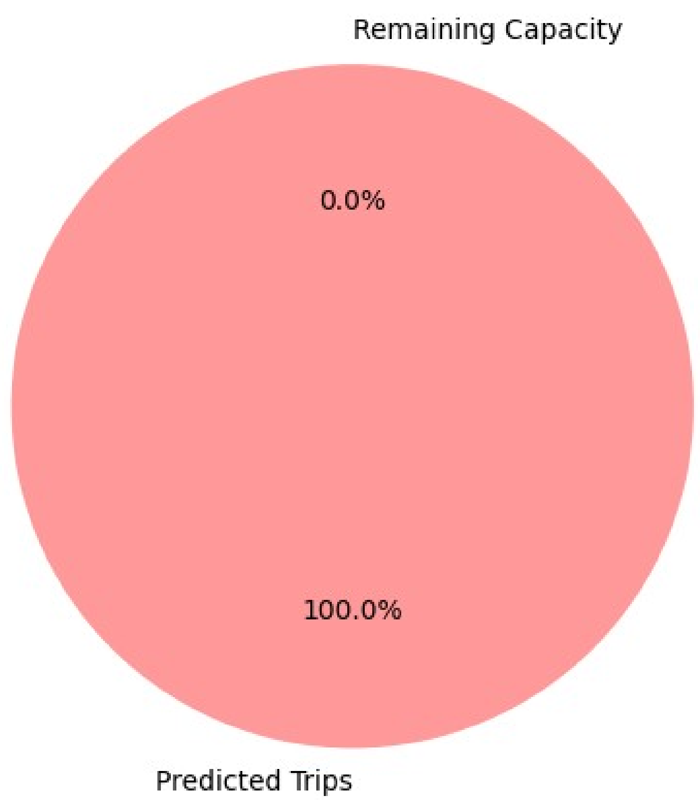
```
Passing `palette` without assigning `hue` is deprecated and will be
removed in v0.14.0. Assign the `x` variable to `hue` and set
`legend=False` for the same effect.

  sns.barplot(x=['Predicted Trips', 'Active Vehicles'], y=[prediction,
active_vehicles], palette=['#ff9999', '#66b3ff'])
C:\Users\91834\AppData\Local\Programs\Python\Python310\lib\site-
packages\sklearn\base.py:493: UserWarning: X does not have valid
feature names, but RandomForestRegressor was fitted with feature names
  warnings.warn(
C:\Users\91834\AppData\Local\Programs\Python\Python310\lib\site-
packages\sklearn\base.py:493: UserWarning: X does not have valid
feature names, but RandomForestRegressor was fitted with feature names
  warnings.warn(
C:\Users\91834\AppData\Local\Programs\Python\Python310\lib\site-
packages\sklearn\base.py:493: UserWarning: X does not have valid
feature names, but RandomForestRegressor was fitted with feature names
  warnings.warn(
C:\Users\91834\AppData\Local\Programs\Python\Python310\lib\site-
packages\sklearn\base.py:493: UserWarning: X does not have valid
feature names, but RandomForestRegressor was fitted with feature names
  warnings.warn(
C:\Users\91834\AppData\Local\Programs\Python\Python310\lib\site-
packages\sklearn\base.py:493: UserWarning: X does not have valid
feature names, but RandomForestRegressor was fitted with feature names
  warnings.warn(
C:\Users\91834\AppData\Local\Programs\Python\Python310\lib\site-
packages\sklearn\base.py:493: UserWarning: X does not have valid
feature names, but RandomForestRegressor was fitted with feature names
  warnings.warn(
C:\Users\91834\AppData\Local\Programs\Python\Python310\lib\site-
packages\sklearn\base.py:493: UserWarning: X does not have valid
feature names, but RandomForestRegressor was fitted with feature names
  warnings.warn(
C:\Users\91834\AppData\Local\Programs\Python\Python310\lib\site-
packages\sklearn\base.py:493: UserWarning: X does not have valid
feature names, but RandomForestRegressor was fitted with feature names
  warnings.warn(
C:\Users\91834\AppData\Local\Programs\Python\Python310\lib\site-
packages\sklearn\base.py:493: UserWarning: X does not have valid
feature names, but RandomForestRegressor was fitted with feature names
  warnings.warn(
C:\Users\91834\AppData\Local\Programs\Python\Python310\lib\site-
packages\sklearn\base.py:493: UserWarning: X does not have valid
feature names, but RandomForestRegressor was fitted with feature names
  warnings.warn(
C:\Users\91834\AppData\Local\Programs\Python\Python310\lib\site-
packages\sklearn\base.py:493: UserWarning: X does not have valid
feature names, but RandomForestRegressor was fitted with feature names
```

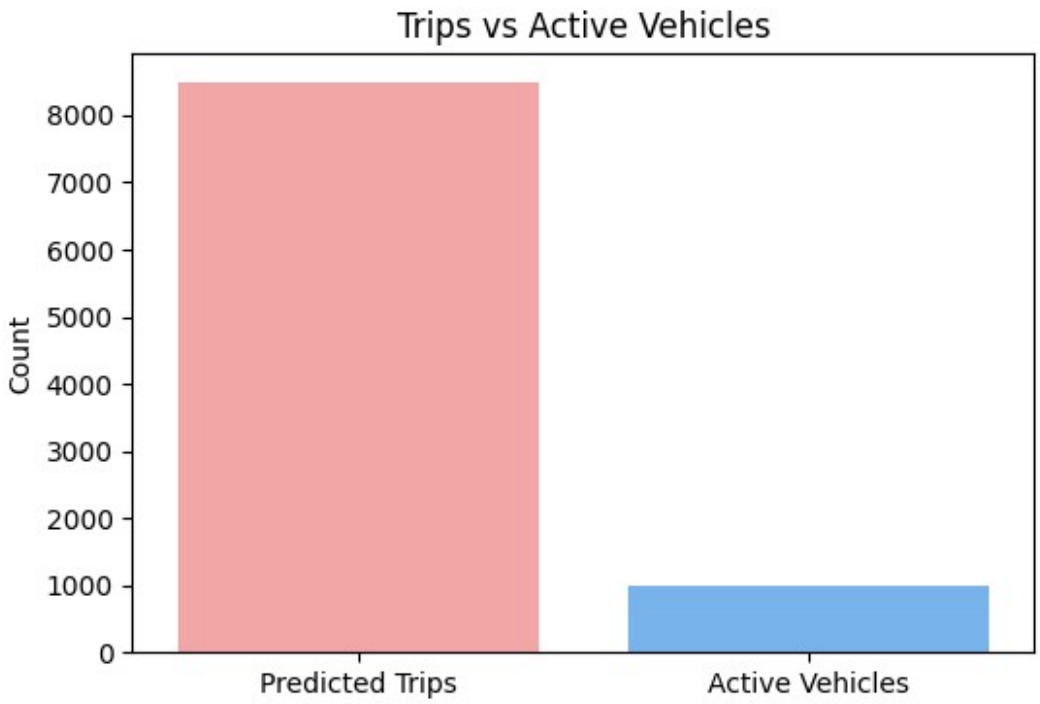
```
  warnings.warn(
C:\Users\91834\AppData\Local\Programs\Python\Python310\lib\site-packages\sklearn\base.py:493: UserWarning: X does not have valid feature names, but RandomForestRegressor was fitted with feature names
  warnings.warn(
C:\Users\91834\AppData\Local\Programs\Python\Python310\lib\site-packages\sklearn\base.py:493: UserWarning: X does not have valid feature names, but RandomForestRegressor was fitted with feature names
  warnings.warn(
C:\Users\91834\AppData\Local\Programs\Python\Python310\lib\site-packages\sklearn\base.py:493: UserWarning: X does not have valid feature names, but RandomForestRegressor was fitted with feature names
  warnings.warn(
C:\Users\91834\AppData\Local\Programs\Python\Python310\lib\site-packages\sklearn\base.py:493: UserWarning: X does not have valid feature names, but RandomForestRegressor was fitted with feature names
  warnings.warn(
C:\Users\91834\AppData\Local\Programs\Python\Python310\lib\site-packages\sklearn\base.py:493: UserWarning: X does not have valid feature names, but RandomForestRegressor was fitted with feature names
  warnings.warn(
C:\Users\91834\AppData\Local\Programs\Python\Python310\lib\site-packages\sklearn\base.py:493: UserWarning: X does not have valid feature names, but RandomForestRegressor was fitted with feature names
  warnings.warn(
C:\Users\91834\AppData\Local\Programs\Python\Python310\lib\site-packages\sklearn\base.py:493: UserWarning: X does not have valid feature names, but RandomForestRegressor was fitted with feature names
  warnings.warn(
C:\Users\91834\AppData\Local\Programs\Python\Python310\lib\site-packages\sklearn\base.py:493: UserWarning: X does not have valid feature names, but RandomForestRegressor was fitted with feature names
  warnings.warn(
C:\Users\91834\AppData\Local\Programs\Python\Python310\lib\site-packages\sklearn\base.py:493: UserWarning: X does not have valid feature names, but RandomForestRegressor was fitted with feature names
  warnings.warn(
C:\Users\91834\AppData\Local\Programs\Python\Python310\lib\site-packages\sklearn\base.py:493: UserWarning: X does not have valid feature names, but RandomForestRegressor was fitted with feature names
  warnings.warn(
C:\Users\91834\AppData\Local\Programs\Python\Python310\lib\site-packages\sklearn\base.py:493: UserWarning: X does not have valid feature names, but RandomForestRegressor was fitted with feature names
  warnings.warn(
C:\Users\91834\AppData\Local\Programs\Python\Python310\lib\site-packages\sklearn\base.py:493: UserWarning: X does not have valid feature names, but RandomForestRegressor was fitted with feature names
  warnings.warn(
```

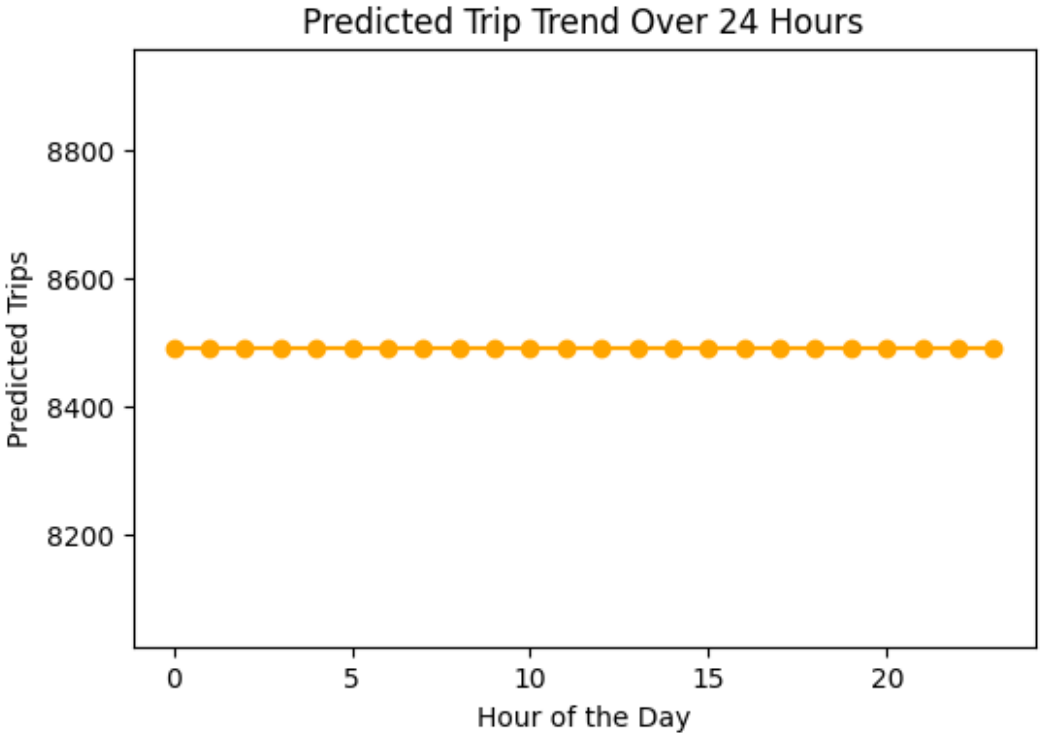
```
C:\Users\91834\AppData\Local\Programs\Python\Python310\lib\site-
packages\sklearn\base.py:493: UserWarning: X does not have valid
feature names, but RandomForestRegressor was fitted with feature names
  warnings.warn(
192.168.1.6 - - [05/Feb/2025 21:41:38] "POST / HTTP/1.1" 200 -
192.168.1.6 - - [05/Feb/2025 21:41:44] "GET / HTTP/1.1" 200 -
C:\Users\91834\AppData\Local\Programs\Python\Python310\lib\site-
packages\sklearn\base.py:493: UserWarning: X does not have valid
feature names, but RandomForestRegressor was fitted with feature names
  warnings.warn(
C:\Users\91834\AppData\Local\Temp\ipykernel_10968\3605251356.py:43:
FutureWarning:

Passing `palette` without assigning `hue` is deprecated and will be
removed in v0.14.0. Assign the `x` variable to `hue` and set
`legend=False` for the same effect.

  sns.barplot(x=['Predicted Trips', 'Active Vehicles'], y=[prediction,
active_vehicles], palette=['#ff9999', '#66b3ff'])
C:\Users\91834\AppData\Local\Programs\Python\Python310\lib\site-
packages\sklearn\base.py:493: UserWarning: X does not have valid
feature names, but RandomForestRegressor was fitted with feature names
  warnings.warn(
C:\Users\91834\AppData\Local\Programs\Python\Python310\lib\site-
packages\sklearn\base.py:493: UserWarning: X does not have valid
feature names, but RandomForestRegressor was fitted with feature names
  warnings.warn(
C:\Users\91834\AppData\Local\Programs\Python\Python310\lib\site-
packages\sklearn\base.py:493: UserWarning: X does not have valid
feature names, but RandomForestRegressor was fitted with feature names
  warnings.warn(
C:\Users\91834\AppData\Local\Programs\Python\Python310\lib\site-
packages\sklearn\base.py:493: UserWarning: X does not have valid
feature names, but RandomForestRegressor was fitted with feature names
  warnings.warn(
C:\Users\91834\AppData\Local\Programs\Python\Python310\lib\site-
packages\sklearn\base.py:493: UserWarning: X does not have valid
feature names, but RandomForestRegressor was fitted with feature names
  warnings.warn(
C:\Users\91834\AppData\Local\Programs\Python\Python310\lib\site-
packages\sklearn\base.py:493: UserWarning: X does not have valid
feature names, but RandomForestRegressor was fitted with feature names
  warnings.warn(
C:\Users\91834\AppData\Local\Programs\Python\Python310\lib\site-
packages\sklearn\base.py:493: UserWarning: X does not have valid
feature names, but RandomForestRegressor was fitted with feature names
  warnings.warn(
C:\Users\91834\AppData\Local\Programs\Python\Python310\lib\site-
packages\sklearn\base.py:493: UserWarning: X does not have valid
```

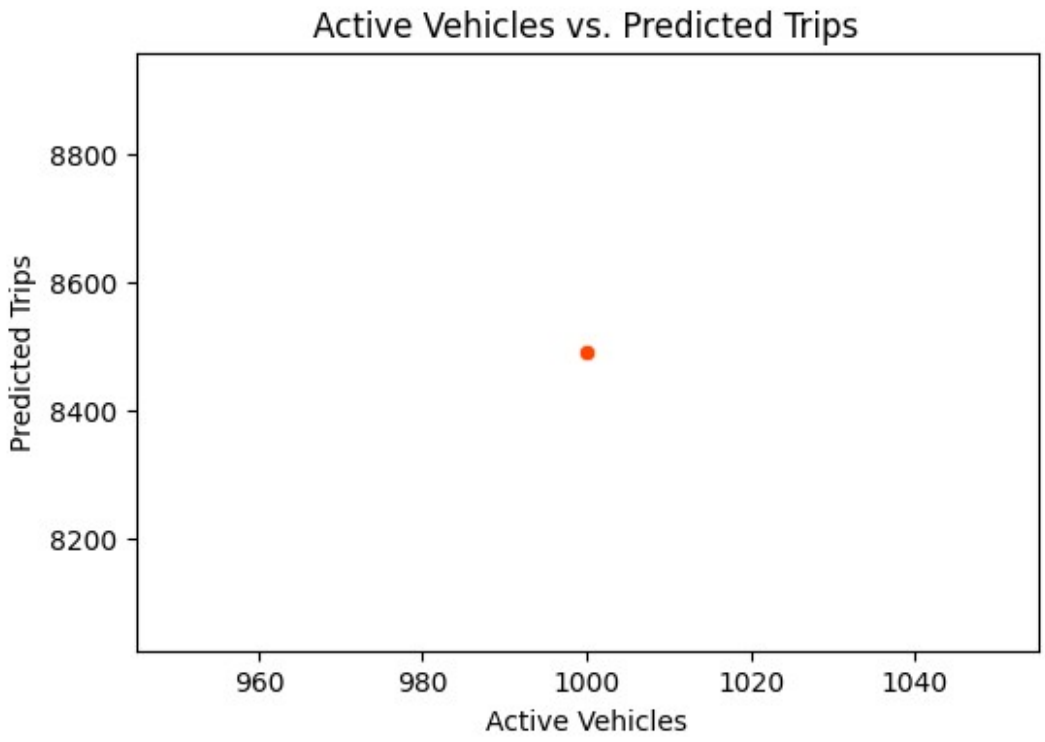
```
feature names, but RandomForestRegressor was fitted with feature names
  warnings.warn(
C:\Users\91834\AppData\Local\Programs\Python\Python310\lib\site-
packages\sklearn\base.py:493: UserWarning: X does not have valid
feature names, but RandomForestRegressor was fitted with feature names
  warnings.warn(
C:\Users\91834\AppData\Local\Programs\Python\Python310\lib\site-
packages\sklearn\base.py:493: UserWarning: X does not have valid
feature names, but RandomForestRegressor was fitted with feature names
  warnings.warn(
C:\Users\91834\AppData\Local\Programs\Python\Python310\lib\site-
packages\sklearn\base.py:493: UserWarning: X does not have valid
feature names, but RandomForestRegressor was fitted with feature names
  warnings.warn(
C:\Users\91834\AppData\Local\Programs\Python\Python310\lib\site-
packages\sklearn\base.py:493: UserWarning: X does not have valid
feature names, but RandomForestRegressor was fitted with feature names
  warnings.warn(
C:\Users\91834\AppData\Local\Programs\Python\Python310\lib\site-
packages\sklearn\base.py:493: UserWarning: X does not have valid
feature names, but RandomForestRegressor was fitted with feature names
  warnings.warn(
C:\Users\91834\AppData\Local\Programs\Python\Python310\lib\site-
packages\sklearn\base.py:493: UserWarning: X does not have valid
feature names, but RandomForestRegressor was fitted with feature names
  warnings.warn(
C:\Users\91834\AppData\Local\Programs\Python\Python310\lib\site-
packages\sklearn\base.py:493: UserWarning: X does not have valid
feature names, but RandomForestRegressor was fitted with feature names
  warnings.warn(
C:\Users\91834\AppData\Local\Programs\Python\Python310\lib\site-
packages\sklearn\base.py:493: UserWarning: X does not have valid
feature names, but RandomForestRegressor was fitted with feature names
  warnings.warn(
C:\Users\91834\AppData\Local\Programs\Python\Python310\lib\site-
packages\sklearn\base.py:493: UserWarning: X does not have valid
feature names, but RandomForestRegressor was fitted with feature names
  warnings.warn(
C:\Users\91834\AppData\Local\Programs\Python\Python310\lib\site-
packages\sklearn\base.py:493: UserWarning: X does not have valid
feature names, but RandomForestRegressor was fitted with feature names
  warnings.warn(
C:\Users\91834\AppData\Local\Programs\Python\Python310\lib\site-
packages\sklearn\base.py:493: UserWarning: X does not have valid
feature names, but RandomForestRegressor was fitted with feature names
```

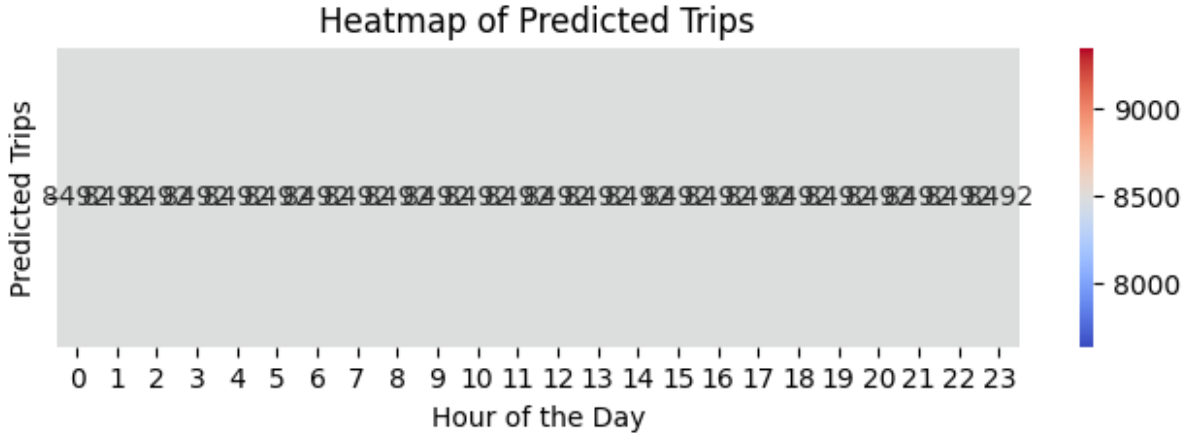
```
  warnings.warn(
C:\Users\91834\AppData\Local\Programs\Python\Python310\lib\site-
packages\sklearn\base.py:493: UserWarning: X does not have valid
feature names, but RandomForestRegressor was fitted with feature names
  warnings.warn(
C:\Users\91834\AppData\Local\Programs\Python\Python310\lib\site-
packages\sklearn\base.py:493: UserWarning: X does not have valid
feature names, but RandomForestRegressor was fitted with feature names
  warnings.warn(
C:\Users\91834\AppData\Local\Programs\Python\Python310\lib\site-
packages\sklearn\base.py:493: UserWarning: X does not have valid
feature names, but RandomForestRegressor was fitted with feature names
  warnings.warn(
C:\Users\91834\AppData\Local\Programs\Python\Python310\lib\site-
packages\sklearn\base.py:493: UserWarning: X does not have valid
feature names, but RandomForestRegressor was fitted with feature names
  warnings.warn(
192.168.1.6 - - [05/Feb/2025 21:42:19] "POST / HTTP/1.1" 200 -
```

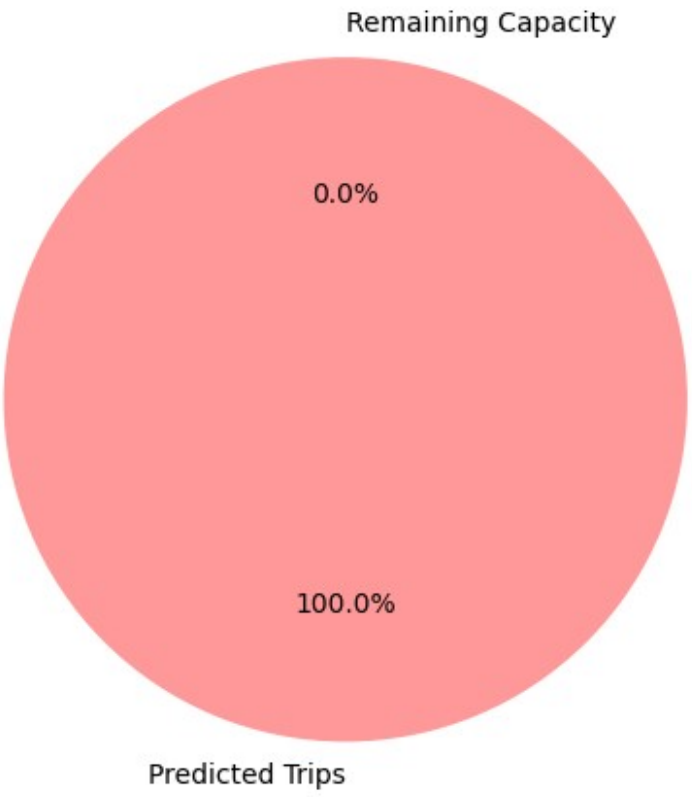


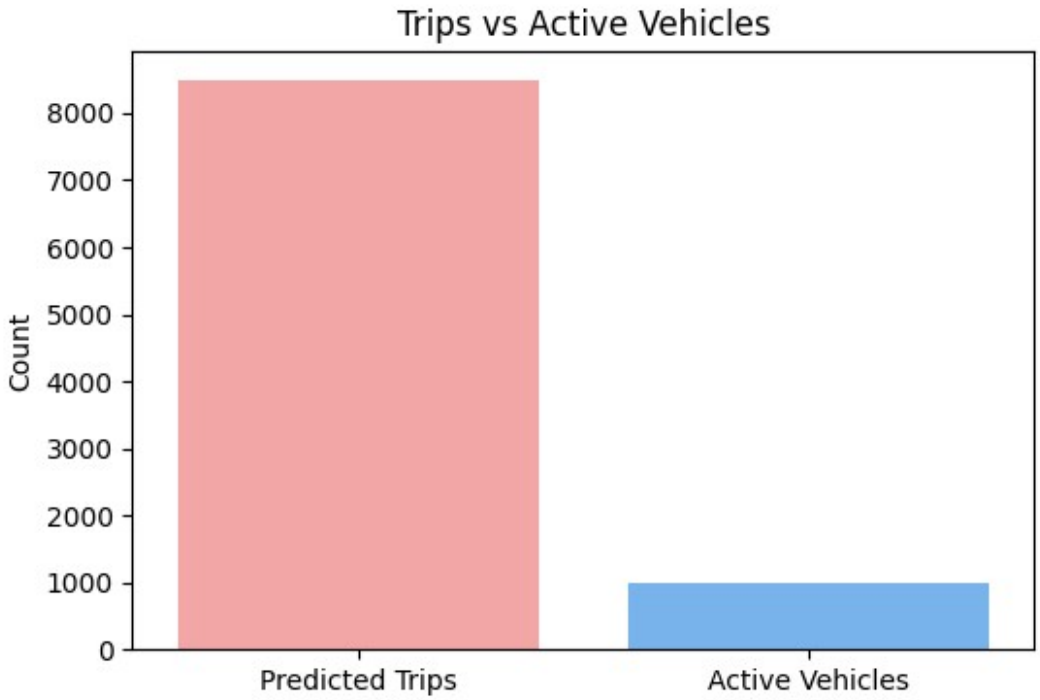

Trips vs Active Vehicles
Count
8000
7000
6000
5000
4000
3000
2000
1000
0
Predicted Trips
Active Vehicles

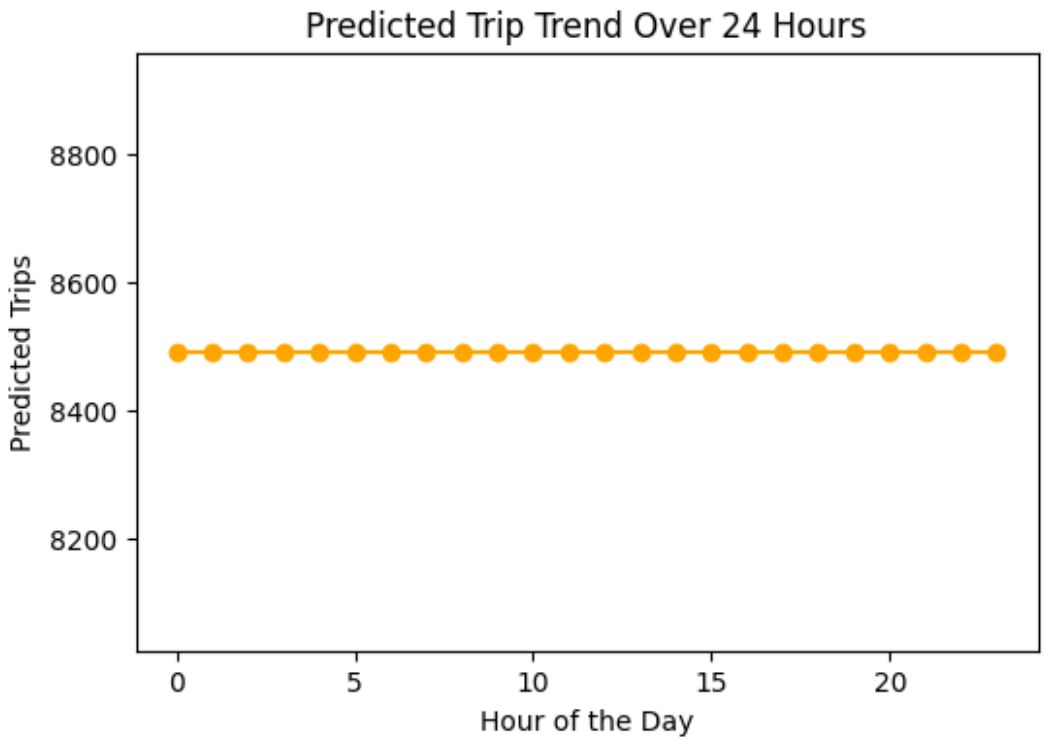

Predicted Trip Trend Over 24 Hours
Predicted Trips
8800
8600
8400
8200
0
5
10
15
20
Hour of the Day

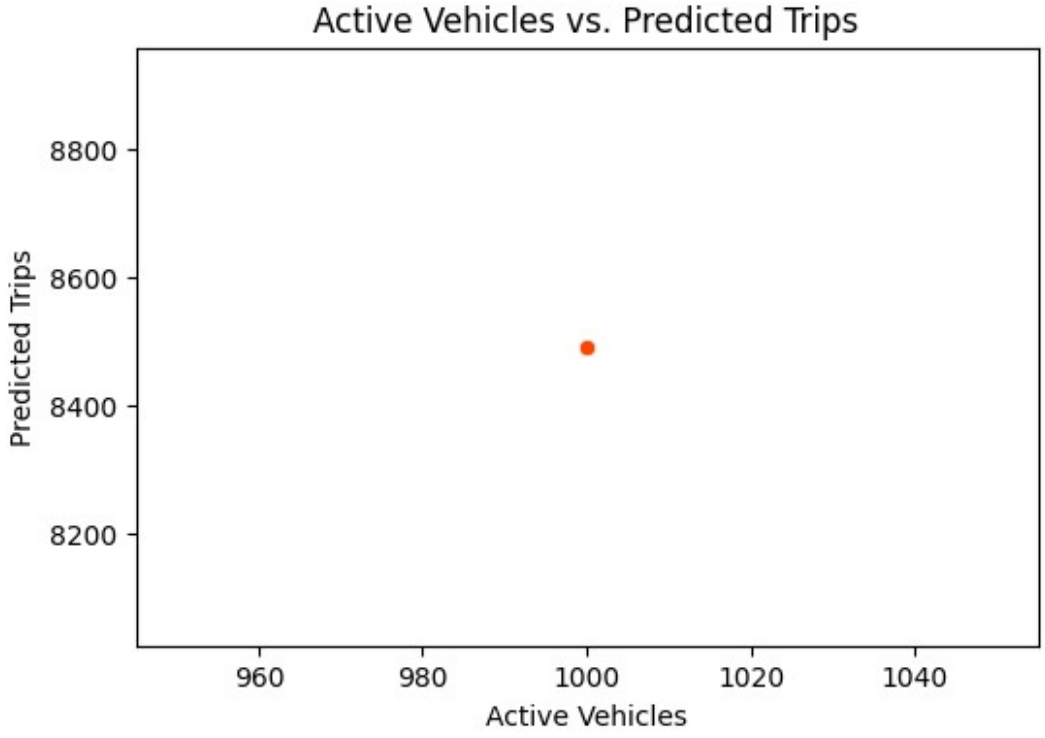

Active Vehicles vs. Predicted Trips
Predicted Trips
8800
8600
8400
8200
960
980
1000
1020
1040
Active Vehicles

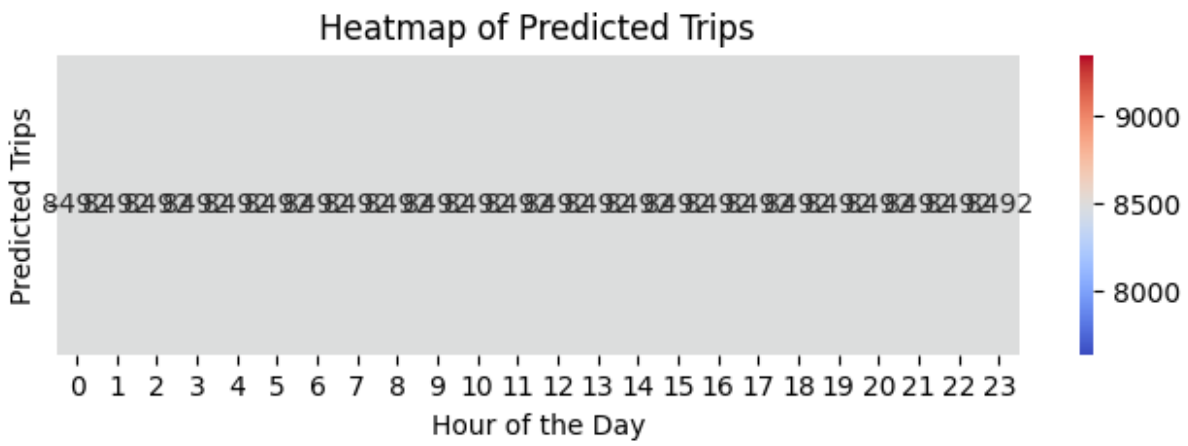

Heatmap of Predicted Trips
Predicted Trips
9000
8500
8000
0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23
Hour of the Day


Remaining Capacity
0.0%
100.0%
Predicted Trips


Trips vs Active Vehicles
Count
8000
7000
6000
5000
4000
3000
2000
1000
0
Predicted Trips
Active Vehicles


Predicted Trip Trend Over 24 Hours
Predicted Trips
8800
8600
8400
8200
0
5
10
15
20
Hour of the Day


Active Vehicles vs. Predicted Trips
Predicted Trips
8800
8600
8400
8200
960
980
1000
1020
1040
Active Vehicles


Heatmap of Predicted Trips
Predicted Trips
9000
8500
8000
0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23
Hour of the Day

Uber Trip Demand Prediction

Select Date & Time:

15-05-2023 15:18

Active Vehicles (Optional):

1000

Predict

Uber Trip Prediction Result

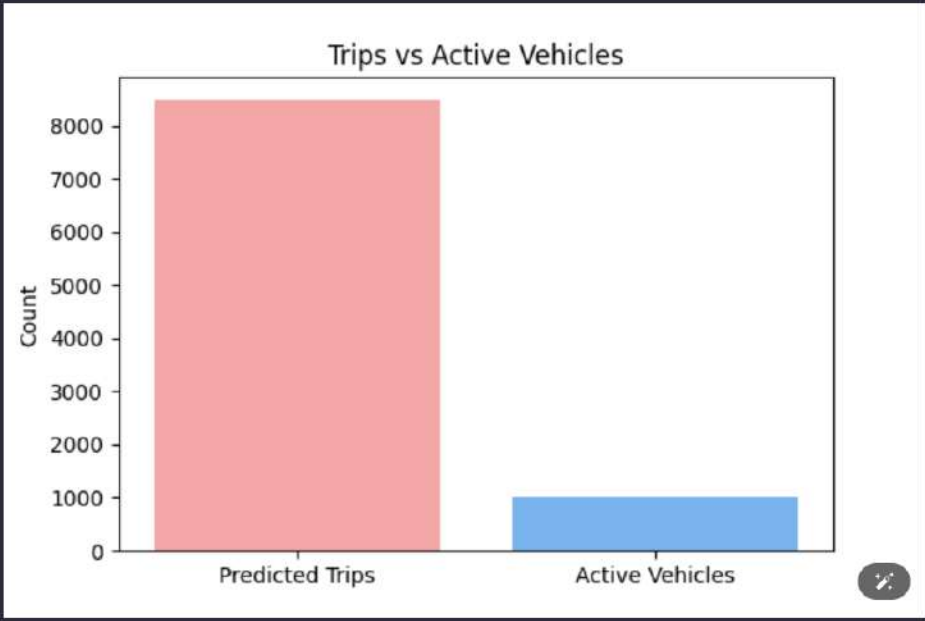
Predicted Uber Trips: 8491

Trip Details

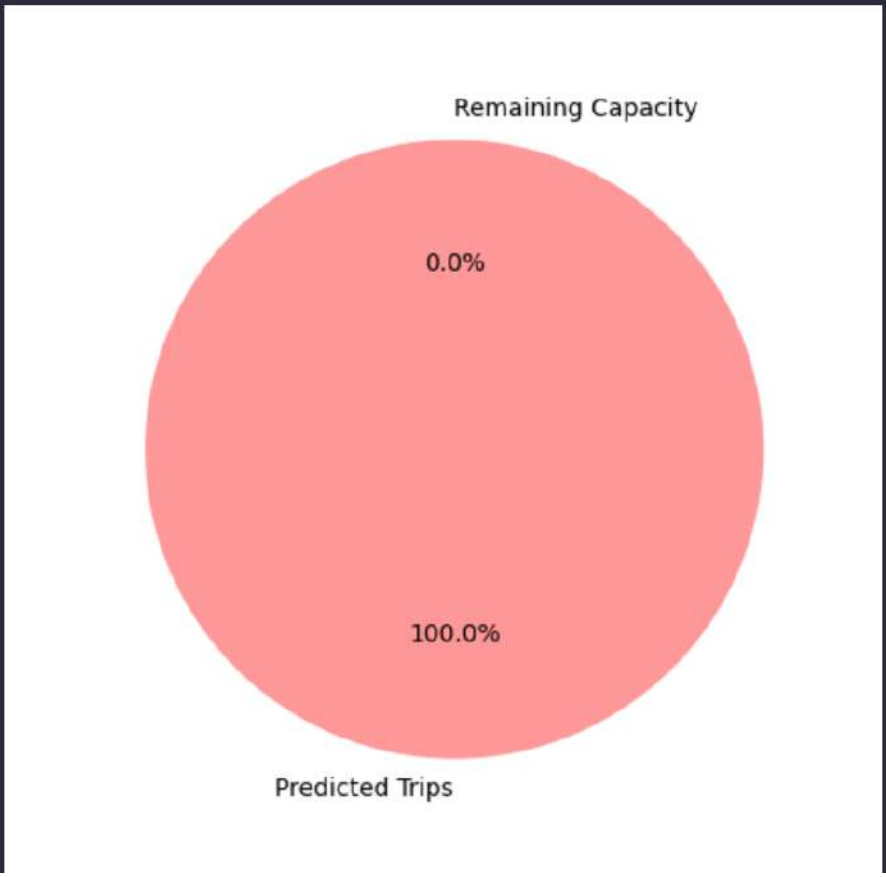
Date & Time: 2023-05-15T15:18

Active Vehicles: 1000

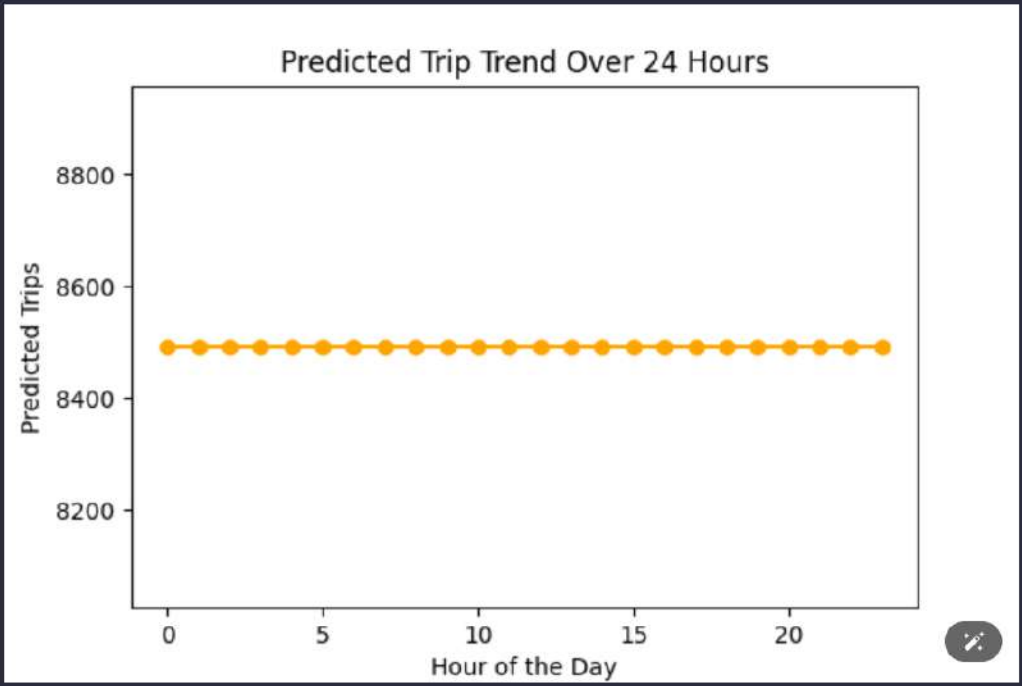
Trips vs Active Vehicles



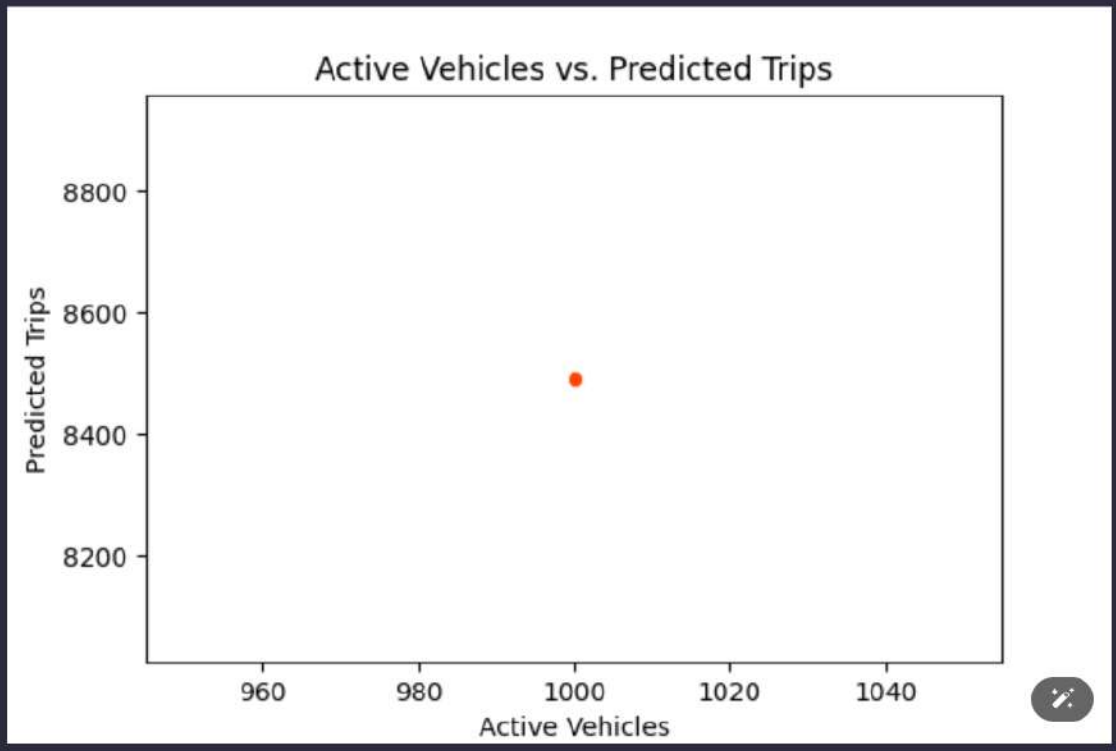
Trip Distribution



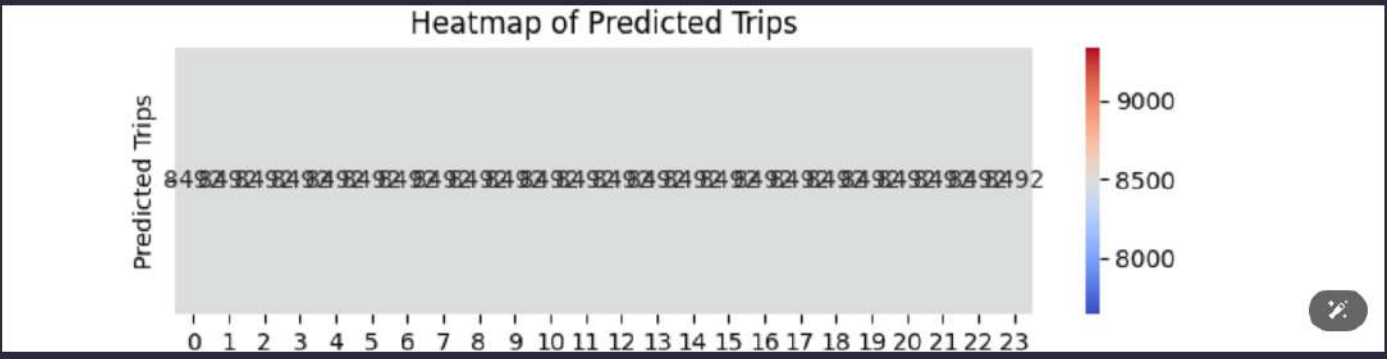
Trips Trend (Line Chart)



Scatter Plot (Trips vs Vehicles)



Heatmap of Trips



Go Back