Title: - Ola Analysis Project

Name: Roopa Saradar

Regno:P18BR23S126001

Department: MCA (BCU)

Semester:Third

✞ Objective: -

To analyze Ola's ride data to uncover insights into demand patterns, driver performance, customer satisfaction, and operational bottlenecks. The findings will support data-driven strategies to improve efficiency, enhance user experience, and drive profitability.

✞ Problem statement: -

Ola faces challenges in understanding ride demand patterns, optimizing driver allocation, enhancing customer satisfaction, and maximizing operational efficiency. This analysis aims to identify high-demand areas, assess driver and customer behaviour, evaluate

the impact of surge pricing, and detect operational inefficiencies to provide actionable insights for strategic decision-making.

✞ Solution: -

Solution for Ola Data Analysis
To address the identified problems, a comprehensive data analysis strategy can be implemented. Below is a structured approach to provide actionable solutions for each sub-problem:

- **Demand Analysis**: Identify peak demand hours, days, and locations to optimize driver allocation and reduce customer wait times.
- **Geographical Insights**: Analyze pickup and drop-off locations to discover high-demand zones.
- **Pricing Optimization**: Evaluate fare trends concerning trip distance and time to recommend fair and competitive pricing. .
  Operational Efficiency: Provide recommendations for better resource management and service delivery.

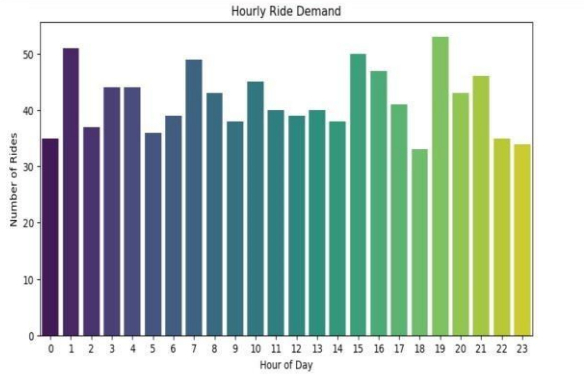
✞ Implemention: -

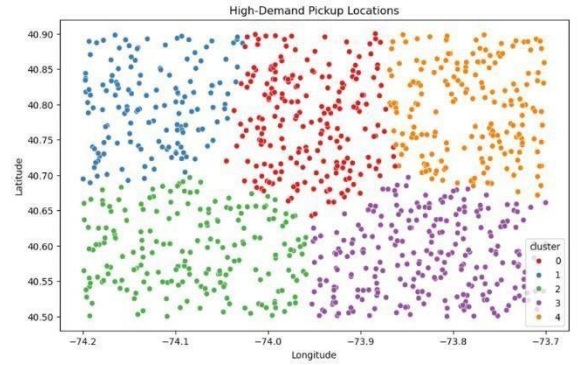
import pandas as pd import numpy as np import matplotlib.pyplot as plt import seaborn as sns from sklearn.cluster import KMeans from sklearn.ensemble import RandomForestRegressor from sklearn.metrics

```
import mean_squared_error from
sklearn.model_selection import train_test_split data =
pd.read_csv("E:/Ola_data.csv")  print(data.head())
print(data.isnull().sum())  data['rating'] =
data['rating'].fillna(data['rating'].mean())
data['pickup_datetime'] =
pd.to_datetime(data['pickup_datetime'])
data['dropoff_datetime'] =
pd.to_datetime(data['dropoff_datetime']) data['hour'] =
data['pickup_datetime'].dt.hour data['day_of_week'] =
data['pickup_datetime'].dt.day_name() data['month'] =
data['pickup_datetime'].dt.month hourly_demand =
data.groupby('hour').size() plt.figure(figsize=(10, 5))
sns.barplot(x=hourly_demand.index,
y=hourly_demand.values, palette='viridis')
plt.title('Hourly Ride Demand') plt.xlabel('Hour of
Day') plt.ylabel('Number of Rides') plt.show()
pickup_coords = data[['pickup_latitude',
'pickup_longitude']].dropna()
kmeans = KMeans(n_clusters=5, random_state=42)
pickup_coords['cluster'] =
kmeans.fit_predict(pickup_coords)
plt.figure(figsize=(10, 6))
sns.scatterplot(x='pickup_longitude', y='pickup_latitude',
hue='cluster', data=pickup_coords, palette='Set1')
```

```
plt.title('High-Demand Pickup Locations')
plt.xlabel('Longitude') plt.ylabel('Latitude') plt.show()
cancellation_rates = data[data['status'] ==
'Cancelled'].groupby('day_of_week').size() total_rides
= data.groupby('day_of_week').size()
cancellation_percentage = (cancellation_rates /
total_rides) * 100 plt.figure(figsize=(10, 5))
sns.barplot(x=cancellation_percentage.index,
y=cancellation_percentage.values, palette='coolwarm')
plt.title('Cancellation Rates by Day of the Week')
plt.xlabel('Day of the Week') plt.ylabel('Cancellation
Rate (%)') plt.show()
```

✞ **Output: -**






```
sns.barplot(x=cancellation_percentage.index, y=cancellation_percentage.values, palette='coolwarm')
```