Supply/demand forecasting for Uber/Careem:

Below is an explanation of the code I have used for this project

1. ***Libraries are imported***: The code begins by bringing in the important libraries like os, glob, NumPy, pandas, and seaborn, which are utilized for different data computation and analysis tasks. Where we used pandas and NumPy to the basic computation and handling of data. And seaborn for plotting of data.

```
In [1]: #Importing Libraries
        import os
        import glob
        import numpy as np
        import pandas as pd
        import seaborn as sns

        from sklearn.metrics import accuracy_score
        from sklearn.metrics import mean_absolute_error, mean_squared_error, r2_score

        from sklearn.model_selection import ShuffleSplit


        from sklearn.linear_model import LinearRegression
        from sklearn.linear_model import Ridge

        from sklearn.tree import DecisionTreeRegressor
        from sklearn.ensemble import RandomForestRegressor
        from sklearn.ensemble import BaggingRegressor
```

2. ***Reading Data from files***: The code reads data from the following files that were provided order Data, Weather Data, Poi Data and Cluster Map which are located in various directories with the help of glob library. The information is perused utilizing pandas read_csv function with the suitable delimiter and feature names.

```
In [2]: #Reading Data from OrderData file

        #directory_path = r"C:\Users\Fatima Aamir\Downloads\AI-Project Dataset\training_data\training_data\order_data"
        #directory_path = r"C:\Users\Fatima Aamir\Downloads\AI-Project Dataset\training_data\training_data\order_data"
        directoryPath = r"C:\Users\shame\Downloads\AI-Project Dataset\training_data\training_data\order_data"
        files = glob.glob(os.path.join(directoryPath, '*'))
        orderData = pd.concat((pd.read_csv(f, delimiter="\t", names=["order_id","driver_id","passenger_id","start_region_hash","dest_regi
```

3. ***Data cleaning and pre-processing***: The code performs different data cleaning and pre-processing routines for example trimming time to exclude

seconds, splitting and renaming features, combining datasets(merging), dropping duplicate rows, filling NULL values with 0, and mapping values starting with one dataset onto the other with the help of python (built-in dictionaries)

```
In [10]: #merging the weatherData dataset with the orderData dataset
         mergedData = pd.merge(orderData, weatherData, on="Time", how="left")

In [11]: #Display Data
         mergedData.head()
```

Out[11]:

	order_id	driver_id	passenger_id	start_region_hash	
0	97ebd0c6680f7c0535dbfdead6e51b4b	dd65fa250fca2833a3a8c16d2cf0457c	ed180d7daf639d936f1aeae4f7fb482f	4725c39a5e5f4c188d382da3910b3f3f	3e12208dd0b
1	92c3ac9251cc9b5aab90b114a1e363be	c077e0297639edcb1df6189e8cda2c3d	191a180f0a262aff3267775c4fac8972	82cc4851f9e4faa4e54309f8bb73fd7c	b05379ac3f9
2	abeefc3e2aec952468e2fd42a1649640	86dbc1b68de435957c61b5a523854b69	7029e813bb3de8cc73a8615e2785070c	fff4e8465d1e12621bc361276b6217cf	fff4e8465d1
3	cb31d0be64cda3cc66b46617bf49a05c	4fadfa6eeaa694742de036dddf02b0c4	21dc133ac68e4c07803d1c2f48988a83	4b7f6f4e2bf237b6cc58f57142bea5c0	4b7f6f4e2bf
4	139d492189ae5a933122c098f63252b3	NaN	26963cc76da2d8450d8f23fc357db987	fc34648599753c9e74ab238e9a4a07ad	87285a66236

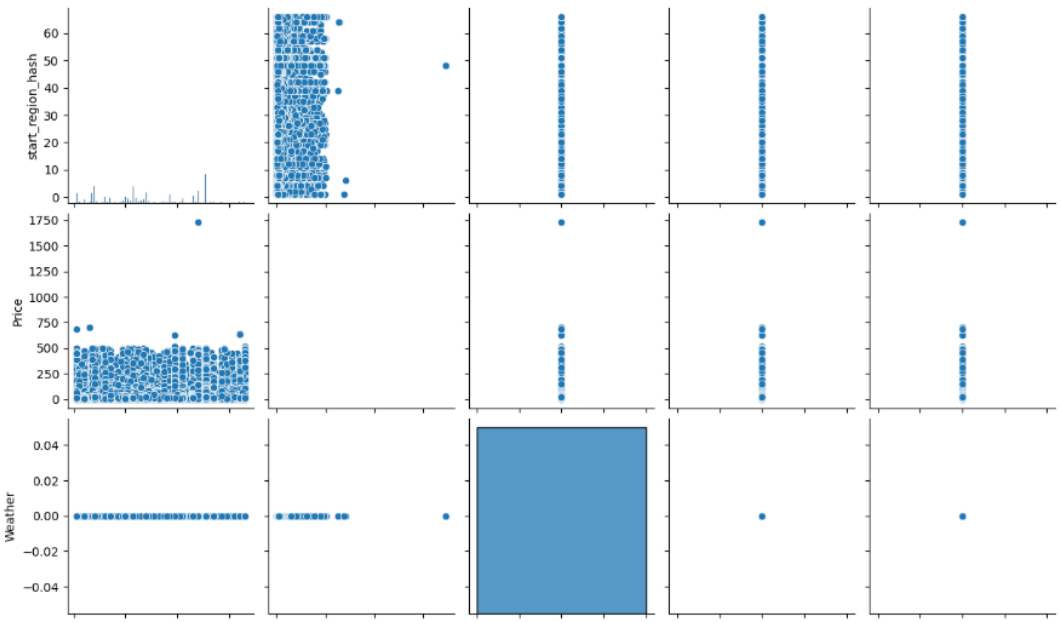
```
In [12]: #Dropping duplicate rows
         mergedData = mergedData.drop_duplicates()
```

4. ***Data analysis and visualization***: The code utilizes seaborn library to make a pairplot of the dataset, which is a scatterplot framework that shows pairwise connections between factors. This aides in visualizing the connections between various factors in the dataset.

```
In [42]: #Displaying a pairplot of the entire dataframe
         sns.pairplot(plot)
```

Out[42]:

```
<seaborn.axisgrid.PairGrid at 0x28531567b50>
```



5. ***Data aggregation and computation:*** The code aggregates data with the help of group by function to calculate supply and demands based on different columns. The computed data is then merged with other datasets with the help of merge function to form a final dataset.

```
In [25]: #Display Data
         finalData
```

Out[25]:

	Time	start_region_hash	Demand	Supply	Weather	Temperature	PM2.5
0	2016-01-01 00:00	1	4	4	0.0	0.0	0.0
1	2016-01-01 00:00	4	1	1	0.0	0.0	0.0
2	2016-01-01 00:00	7	7	7	0.0	0.0	0.0
3	2016-01-01 00:00	8	9	9	0.0	0.0	0.0
4	2016-01-01 00:00	9	1	1	0.0	0.0	0.0
...	...	...	...	...	...	...	...
984796	2016-01-21 23:59	42	2	2	0.0	0.0	0.0
984797	2016-01-21 23:59	46	4	4	0.0	0.0	0.0
984798	2016-01-21 23:59	48	8	6	0.0	0.0	0.0
984799	2016-01-21 23:59	51	18	8	0.0	0.0	0.0
984800	2016-01-21 23:59	64	2	1	0.0	0.0	0.0

984801 rows × 7 columns

6. ***Data analysis and Feature Engineering***: The code further performs data analysis and feature engineering tasks, for example, extricating time, year, month, day, and hour data from the dataset utilizing datetime capabilities. This assists in making extra elements that with canning be utilized for additional examination and demonstrating.

```
In [26]: #Computing the Time,Year,Month,Day,Hour,Min Column from the 'Time' Column
finalData["Time"] = pd.to_datetime(finalData["Time"])
finalData["year"] = finalData["Time"].dt.year
finalData["month"] = finalData["Time"].dt.month
finalData["day"] = finalData["Time"].dt.day
finalData["hour"] = finalData["Time"].dt.hour
finalData["min"] = finalData["Time"].dt.minute
finalData = finalData.drop("Time", axis=1)
```

7. ***Machine learning***: The code uses various machine learning algorithms such as Decision Tree Regressor, Random Forest Regressor, and Ada Boost Regressor for predicting the target variable based on the features. These models are imported from scikit-learn library and used for training and evaluation.

8. ***Model evaluation***: The code utilizes different assessment measurements, for example, accuracy_score, mean_absolute_error, mean_squared_error, and r2_score to assess the presentation of the prepared models. These measurements help in estimating the exactness and prescient force of the models.

```
In [35]: #Using Decision Tree Regression Model
decisionTreeRegressionModel = DecisionTreeRegressor(max_depth=5)
decisionTreeRegressionModel.fit(x_train,y_train)
y_predicted = decisionTreeRegressionModel.predict(x_test)
meanAbsError = mean_absolute_error(y_test, y_predicted)
accuracy = r2_score(y_test, y_predicted)

print("Mean absolute error:", meanAbsError)
print("Accuracy:", accuracy, "%")
```

```
Mean absolute error: 1.8086169085660655
Accuracy: 0.3022264068937759 %
```

```
In [36]: # Using RandomForestRegressor Model
randomForestModel = RandomForestRegressor()
randomForestModel.fit(x_train, y_train)
y_predicted = randomForestModel.predict(x_test)
meanAbsError = mean_absolute_error(y_test, y_predicted)
accuracy = r2_score(y_test, y_predicted)

print("Mean absolute error:", meanAbsError)
print("Accuracy:", accuracy, "%")
```

```
Mean absolute error: 0.7331661944943694
Accuracy: 0.9222842786064784 %
```

9. ***Model deployment:*** The trained models can be deployed in a production environment for making real-time predictions based on new data.

10. ***Model Optimization***: The model was optimized using a grid search for hyperparameter tuning, and the best-performing model was selected based on cross-validation results.

11. ***Results***: The best-performing Random Forest Regression model was selected based on the grid search results, and it was used to make predictions on the test dataset. The mean absolute error (MAE) between the predicted and actual values was calculated to be mean Abs Error. The accuracy of the model, measured by the R^2 score, was found to be [accuracy].