Team Members
❖ **Name: محمد مجدي محمد حسين السيد**
➢ **Id: 20191700568**
❖ **Name:محمد عصام الدين ابراهيم الجبالي**
➢ **Id: 20191700559**
❖ **Name:سعيد عبدالناصر سعيد**
➢ **Id: 20191700286**

Report on taxi problem:

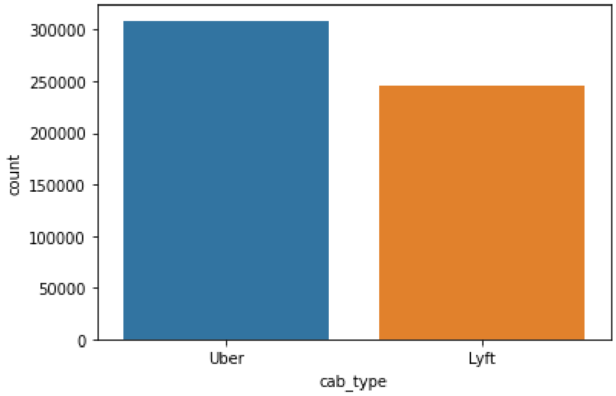
We have two datasets to work on.

First, taxi rides dataset, we read it using pandas library and using pandas again to show the first five rows to explore the data

```
taxi_data=pd.read_csv('taxi-rides.csv')
taxi_data.head()
```

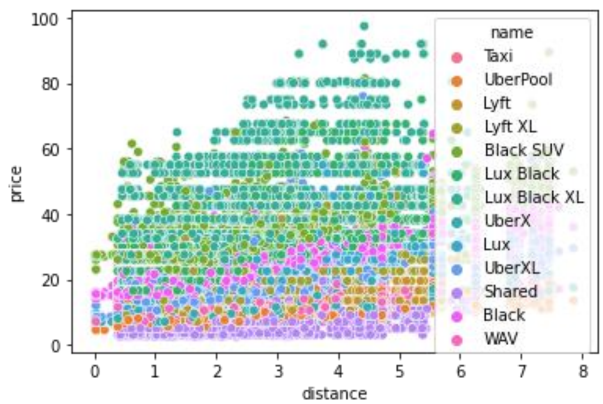
We visualize some features in taxi rides dataset like cap_type feature to see the ratio between UPER and LYFT

```
sns.countplot(x='cab_type', data=taxi_data)
```



We also visualize the relation between the distance, name of the cap and the price and similar with price, cap type and distance

```
sns.scatterplot(data=taxi_data, x="distance", y="price",hue='name')
```



We see the count of the rows then we see the summation of the null values

```
taxi_data.count()
distance            554456
cab_type            554456
time_stamp          554456
destination         554456
source              554456
surge_multiplier    554456
id                  554456
product_id          554456
name                554456
price               510321
      dtype: int64
```

```
taxi_data.isnull().sum()
```

distance	0
cab_type	0
time_stamp	0
destination	0
source	0
surge_multiplier	0
id	0
product_id	0
name	0
price	44135

dtype: int64

We drop the rows with the null values because we have 554456 record and 44135 null

We drop the duplicates

```
taxi_data.dropna(subset=['price'],inplace=True)
taxi_data.reset_index(drop=True, inplace=True)
```

Second, weather dataset, we used the same techniques which we used in taxi drives dataset

We see that feature rain have many missed data that has not been measured so we drop the column

```
weather_data.drop(['rain'],axis=1,inplace=True)
```

We convert time stamp to data in each datasets

```
taxi_data['key'] = pd.to_datetime(taxi_data['time_stamp'], unit='ms').appl
y(lambda x: x.strftime(('%Y/%m/%d')))
weather_data['key']=pd.to_datetime(weather_data['time_stamp'], unit='s').a
pply(lambda x: x.strftime(('%Y/%m/%d')))
```

We record the time of the trip

```
taxi_data['trip_hour'] = pd.to_datetime(taxi_data['time_stamp'], unit='ms'
).dt.hour
```

we used group by date and location and take the average

```
weather=weather_data.groupby(['key','location']).agg({'temp':'mean','cloud
s':'mean','pressure':'mean','humidity':'mean','wind':'mean'}).reset_index(
)
```

We merge the two datasets

```
Data=taxi_data.merge(weather,how='left',left_on=['source','key'], right_on
=['location','key'])
Data=Data.merge(weather,how='left',left_on=['destination','key'], right_on
=['location','key'])
```

We see the correlation between the data and drop the features that are less correlated with price

```
Data=Data.drop(['id','product_id','time_stamp','clouds_x','clouds_y','wind
_x','wind_y'],axis=1,inplace=False)
```

We use one hat encoder to the labeled data and we split the data to train and test with 70% and 30%

We use PCA to choose the best features and we make feature scaling

```
X=Data.drop(['price'],axis=1,inplace=False)
X=FeatureScalling(X)
pca=PCA(n_components=16)
X=pca.fit_transform(X)
y=Data['price']#label
X_train,X_test,y_train,y_test=train_test_split(X,y,test_size=0.3,shuffle=F
alse,random_state=42)
```

We make three models

First one: polynomial model and there is the values of the model

```
Mean Square Error: 3.344536955400812
r2_score : 0.9614382507773105
```

Second one: multivariable model and there is the values of the model

```
Mean Square Error: 9.144976902984913
r2_score : 0.894560499500317
```

Third one: we use cross validation on the model and there is the values of the model

```
model 1 cross validation score is 3.7626237499306323
```

Conclusion:

That problem we work on, we can say in the first days it was difficult to us and we faced some trouble with the data but in the end we worked hard and make many choices to achieve that result

The result satisfying us, and I see the problem is proved and we handle it.