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->Classification:

We have two datasets to work on like the first problem (Regression) but this problem is a classification problem.

In this problem we should to detect the class terms of the other features.

Probably that all features is not important to us so in that problem we will see what features are important to get higher accuracy.

Reading Data:

First, taxi rides dataset, we have an extra feature (classification feature) its name “Ride Category”, and it contains five classes -> {unknown, cheap, moderate, expensive, very expensive}

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-classification.csv')

taxi_data.head()
```

Second, weather dataset, we used the same techniques which we used in taxi drives dataset, in that problem the data set of the weather doesn't change.

```
weather_data=pd.read_csv('weather.csv')
weather_data.head()
```

Pre-Processing of the Data:

In the taxi_data, we see the count of the rows then we see the summation of the null values, as you can see that there is no null values in the taxi_data.

```
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
RideCategory        0
dtype: int64
```

Also, we see if there duplicated values and there is no duplicated values.

```
taxi_data.duplicated().sum()
```

```
0
```

In the weather_data, we see the count of the rows then we see the summation of the null values, so from the output we see that we have many nulls in the rain feature.

```
weather_data.count()
```

```
temp          6276
location      6276
clouds        6276
pressure      6276
rain           894
time_stamp    6276
humidity      6276
wind          6276
dtype: int64
```

```
weather_data.isnull().sum()
```

```
temp             0
location         0
clouds           0
pressure         0
rain          5382
time_stamp       0
humidity         0
wind             0
dtype: int64
```

So we drop the rain feature because it includes many nulls.

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

After cleaning the weather_data

```
weather_data.isnull().sum()
```

```
temp          0
location      0
clouds        0
pressure      0
time_stamp    0
humidity      0
wind          0
dtype: int64
```

Also, we see if there duplicated values like the taxi_data and there is no duplicated values.

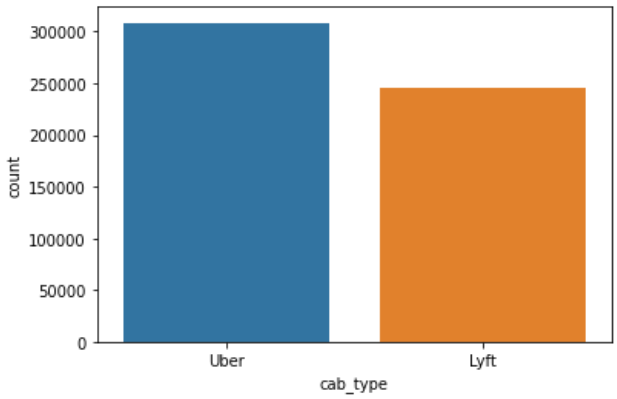
```
weather_data.duplicated().sum()
```

```
0
```

Visualizing of the Data:

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)
```



Feature Engineering:

We convert time stamp to data in each datasets (taxi data, weather data)

```
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'])
```

Encoding:

We see that data set need some encoding.

So in **taxi_data** we see that [cap_type] needs encoding so we make label encoding to that feature -> we set the **UBER TO 0** and **LYFT TO 1**

```
Data['cab_type']=Data.cab_type.map(dict(Uber=0,Lyft=1))
```

And we use one hot encoder to the **name** feature, and give every name a column with its name

```
encoder = OneHotEncoder(handle_unknown='ignore')
encoder_name = pd.DataFrame(encoder.fit_transform(Data[['name']]).toarray(
))



Data = Data.join(encoder_name)
Data=Data.rename(columns={0: "name0", 1: "name1", 2: "name2" , 3 :'name3'
, 4: "name4", 5: "name5", 6: "name6" , 7 :'name7' ,8: "name8", 9: "name9",
 10: "name10" , 11 :'name11',12:'name12'})
Data.drop('name',axis=1,inplace=True)
```

And we save the model of the encoder after making that.

```
with open('encoder', 'wb') as files:
    pickle.dump(encoder, files)
```

And we also making a label encoder to the **RIDECATEGORY** feature

```
le =LabelEncoder()
Data['RideCategory']=le.fit_transform(Data['RideCategory'])
with open('le_encode', 'wb') as files:
    pickle.dump(le, files)
```

Feature Selection:

We see the correlation between the data (features) and the **RIDECATEGORY** feature so we drop the features that are less correlated with it.

```
Data=Data.drop(['id','product_id','time_stamp'],axis=1,inplace=False)
```

After dropping some features we make a feature scaling to all the features except the classification feature and we split the data to the train and the test-> 70% to 30%.

```
X=Data.drop(['RideCategory'],axis=1,inplace=False)
X=FeatureScalling(X)
y=Data['RideCategory']#label
```

Splitting:

```
X_train,X_test,y_train,y_test=train_test_split(X,y,test_size=0.3,shuffle=False,random_state=42)
```

Models:

The first model we use a random forest classifier

```
#Random forest classfier
rf=RandomForestClassifier()
rf.fit(X_train,y_train)
preds=rf.predict(X_test)
print('Accuracy :', metrics.accuracy_score(y_test, preds))
```

```
Accuracy : 0.8780006853556335
```

Random forest classifier:

It is an ensemble method-> {use multiple learning algorithms to obtain better predictive performance}

For classification tasks, the output of the random forest is the class selected by most trees

The second model we use a logistic regression

```

```

```
def LogisticRegressionModel(X_train,y_train,X_test,y_test):
    model =LogisticRegression(max_iter=1000)
    model.fit(X_train, y_train)
    prediction = model.predict(X_test)
    print('Mean Square Error:', metrics.mean_squared_error(y_test, predict
ion))
    print('Accuracy :', metrics.accuracy_score(y_test, prediction))
```

```
Mean Square Error: 0.4991553292412392
Accuracy : 0.8752772984964259
```

Logistic Regression:

Is a process of modeling the probability of a discrete outcome given an input variable. The most common logistic regression models a binary outcome; something that can take two values such as true/false, yes/no, and so on. Multinomial logistic regression can model scenarios where there are more than two possible discrete outcomes. Logistic regression is a useful analysis method for classification problems, where you are trying to determine if a new sample fits best into a category. As aspects of cyber security are classification problems, such as attack detection, logistic regression is a useful analytic technique.

The third model we use cat boosting

```
from catboost import CatBoostClassifier
cat_clf=CatBoostClassifier()
cat_clf.fit(X_train,y_train)
preds=cat_clf.predict(X_test)
print('Accuracy :', metrics.accuracy_score(y_test, preds))
```

```
Accuracy : 0.8935594606130927
```

Cat boost classifier:

CatBoost is an algorithm for gradient boosting on decision trees and

CatBoost provides state of the art results and it is competitive with any leading machine learning algorithm on the performance front, CatBoost can use categorical features directly and is scalable in nature

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.