DESCRIPTIONS

50/50

1. **BASERATE →**
   Here we calculated base-rate of our data sentiment feature 233/1344 which comes out to be 17% approximately. Our positive values are 233 and total records we have are 1344. As our data had less positive (1) values overall. So, our base rate is low.

```
In [175]: #baserate

          print(sum(df['sentiment']) / len(df['sentiment'])*100)

          17.336309523809522
```

2. Few measures of **CENTRAL TENDANCY.** We have 3 features and 1 is "Reviews" as text. So, we cannot get any stats from that, but we got mean, median, std, min, max etc. of rest of the two features.

```
In [108]: df.describe()
```

Out[108]:

	ride_rating	sentiment
count	1344.000000	1344.000000
mean	1.627232	0.173363
std	1.296149	0.378702
min	1.000000	0.000000

IT WOULD BE COOL TO SEE WHAT WORDS FROM THE TEXT ARE MOST/LEAST COMMON.

3. **Plots –** We plotted "Ratings" feature to see how many customers rated the ride as 1,2,3,4,5 stars, where 5 being the best and 1 being the worst. We were shocked to see that most of the customers rated the ride as 1. We want to try the logistic regression first and later decision trees for our project, we wanted to convert these 5 ratings categories into binary classes (1 and 0). A reason for this was, logistic regression only handles "either/or" target variables. For this purpose, we tried setting up ratings below 3 stars as class 0 (negative sentiment) and above 3 as class 1 (positive sentiment). If we consider 3-star ratings, they can be considered neutral and offer no sentiment insight. We created a new feature as "Binary-class" from the upper discussion.

```
In [221]: df['binary_class'] = np.where(df['ride_rating'] > 3, 1, 0)
          df
```

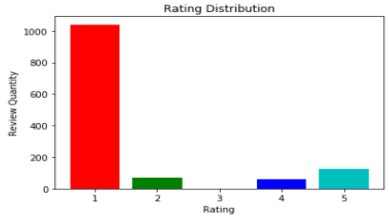
Out[221]:

	ride_review	ride_rating	sentiment	binary_class
0	I completed running New York Marathon requeste...	1.0	0	0
1	My appointment time auto repairs required earl...	1.0	0	0
2	Whether I using Uber ride service Uber Eats or...	1.0	0	0

THIS IS VERY SUPRISING. DID THE DATA COME WITH ANY DETAILS THAT MIGHT EXPLAIN WHY THERE ARE SO MANY 1'S?

Graph after removing all the 3 ratings

```
In [177]: #deleting all instances with ride_rating = 3
          df = df[df.ride_rating != 3]
          #separating by groups
          groups = df.groupby('ride_rating').count()
          Values = groups.ride_review
          colors = ['r', 'g', 'b', 'c']
          #making bar plot
          plt.bar(([1,2,4,5]), Values, color= colors)
          plt.title('Rating Distribution')
          plt.xlabel('Rating')
          plt.ylabel('Review Quantity')
          plt.show()
```



MODEL BUILDING (LOGISTIC REGRESSION)

- In our model building phase, we want to try out both Logistic Regression as well as Decision Trees. For the initial phase we tried Logistic regression. For some basic to start with we separated our data into training and testing sets. We're using 'sklearn' to shuffle and split. Without messing with the any parameters, it splits data into 75% training and 25% testing.
- This first part gives us a random review and shows how the data is split 970/1294 = 75%. As we have words in 'reviews', we want to convert those words to number features and then count the frequency of those numbers/words in-order to analyze (We found that it is called Bag of Words).

- We wanted to see the total numbers of words in all the reviews and see those words separately. We used the 'sklearn' 'CountVectorizer'. It shows all the words in a list/vector and in this case, it does not consider the order of those words but just the count of frequencies per instance.
- After this step we constructed a matrix which shows instances as rows and the count of new features (6,607 numerical features we converted from words) as columns. For example, our first word (0) as seen above is "abandoned". That will be the first column in the matrix. Whichever Uber reviews contain 'abandoned' will tally up how many times it was used and add it to that column. Someone using "abandoned", will likely have a negative effect on rating.
- After this step we tried running the Logistic Regression model and calculated the ROC/AUC to be 75% correctly classified instances. After the regression we used the coefficients we got back and converted numbers back to words to see their impacts on class separation.
- Tested few new inputs from our side so as to check if our model predicts 'good' or 'bad' as '1' or '0'.

```
In [223]: #splitting into train and test
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(df['ride_review'], df['binary_class'], random_state = 0)
#setting random number between 1 and 1000
number = random.randint(1,1000)
#printing random training text and X_train shape
print ('Random Review:')
print(' ')
print(X_train[number])
print(' ')
print('X_train shape: ' + str(X_train.shape))

Random Review:

I wanted write let Uber I amazing driver day As soon I got car handed bowl delicious treats I thought sweet Then handed bottle
water This may big deal however actually made day He polite helpful generous around awesome Please please let driver Jasur Naza
rou know awesome He deserves sort praise Uber All drivers take notes guy All drivers need like WONDEEFUL EXPERIENCE Stephanie A
ngelini Angelinistephanie

X_train shape: (970,)
```

```
In [93]: #importing countvectorizer
from sklearn.feature_extraction.text import CountVectorizer
#creating variable which assigns X_train to numbers
vect = CountVectorizer().fit(X_train)
#translates numbers back to text
vect.get_feature_names()[1:100]

Out[93]: ['abandoned',
 'ability',
 'able',
 'about',
 'abrupt',
 'absence',
 'absolute',
 'absolutely',
 'absorb',
 'absorbers',
```

MAYBE TRY A tf-idf VECTORIZER?

```
In [98]: #creating array variable of all the words
feature_names = np.array(vect.get_feature_names())
#creating array of all the regression coefficients per word
coef_index = model.coef_[0]
#creating df with both arrays in it
df = pd.DataFrame({'Word':feature_names, 'Coef': coef_index})
#sorting by coefficient
df.sort_values('Coef')

Out[98]:
```

	Word	Coef
913	charged	-1.025996
3820	never	-0.959084

```
In [94]: #creating matrix array for logistic regression
X_train_vectorized = vect.transform(X_train)
print (X_train_vectorized.toarray())

[[0 0 0 ... 0 0 0]
 [0 0 0 ... 0 0 0]
 [0 0 0 ... 0 0 0]
 ...
 [0 0 0 ... 0 0 0]
 [0 0 0 ... 0 0 0]
 [0 0 0 ... 0 0 0]]
```

```
In [96]: #creating Log regression
from sklearn.linear_model import LogisticRegression
model = LogisticRegression()
model.fit(X_train_vectorized, y_train)

Out[96]: LogisticRegression(C=1.0, class_weight=None, dual=False, fit_intercept=True,
                            intercept_scaling=1, l1_ratio=None, max_iter=100,
                            multi_class='warn', n_jobs=None, penalty='l2',
                            random_state=None, solver='warn', tol=0.0001, verbose=0,
                            warm_start=False)
```

```
In [97]: #calculating AUC
from sklearn.metrics import roc_auc_score
predictions = model.predict(vect.transform(X_test))
print('AUC: ', roc_auc_score(y_test, predictions))

AUC:  0.7567657038814863
```

```
In [101]: ##Test inputs reviews

print(model.predict(vect.transform(['abandoned great'])))
print(model.predict(vect.transform(['great she the best'])))
print(model.predict(vect.transform(['charged slow horrible'])))
print(model.predict(vect.transform(['it was as average as a trip could be'])))
print(model.predict(vect.transform(['my family felt safe we got to our dest

[1]
[1]
[0]
```

YOU REALLY ONLY NEED TO TRY THESE TWO.

Note → This was just some basic regression we did. For further updates we want to run detailed logistic regression as well as decision trees. We will do k-fold cross-validation and use many different type of matrices(Accuracy, Roc, Confusion Matrix, try and plot Roc/Auc etc) to evaluate our model we build in future for the final submission. We want to build a solid model that we can predict the review category and could be used to minimise loss by looking at all the classified reviews and increase the overall profits.

LOOKING AT MANY OF THESE FOR ALL MODELS MIGHT BE CONFUSING. MAYBE JUST SHOW ONE FOR ONE MODEL