Introduction / Business Problem :

The dataset we are going to use in this project is about collisions in some years in Seattle. As car accidents are increasing day-by-day, there is a need to predict a model which can detect car accident severity based on parameters like weather, road condition, visibility condition, speed, etc., If an eddective model is built which can detect accident severity, it can alert drivers to be more careful and cautious and thus avoid accidents.

Data Description :

The Dataset we are going to use here is consists of 194673 rows as the incidents and 38 columns as the attributes. The "SEVERITYCODE" column has the values of 1 and 2 where 1 is prop damage and 2 is injury. Similarly, there are attributes like road condition (ROADCOND), light condition (LIGHTCOND), etc., which can help us in building a model to predict the accidents.

After understanding the business problem, we need to understand the data. Here, the libraries we are mainly going to use are Pandas, Seaborn, Numpy and Matplotlib. After we have imported the dataset, we need to check for the rows and columns and their values. In simple terms, we need to see the characteristics of the dataset to understand it better and make it perfectly balanced as there might me some rows having empty or null values which can hamper the test results.

We will be using supervised Machine Learning models to predict the values. The split is as follows:

-> We can use K-Nearest Neighbors as it can help predict the SEVERITYCODE value. -> We can use Descision Tree model when it comes to predicting the outcome of the weather as this model analyzes all the conditions and then comes up with a decision. -> We can use Logistic Regression as out dataset only has two values for the SEVERITYCODE and hence, it will be the best approach to use logistic Regression.

Now, as we have a brief understanding of the data and what we have to do, let's move to the code where each process is explained.

First, let's import the necessary libraries

In [2]:

```
import pandas as pd
import numpy as np
import seaborn as sns
import matplotlib.pyplot as plt
from sklearn import preprocessing
%matplotlib inline
```

Then we have to load the dataset

In [3]:

```
df = pd.read_csv("/Users/chirayusharma/Downloads/Data-Collisions.csv")
```

```
/Users/chirayusharma/opt/anaconda3/lib/python3.7/site-packages/IPython/co
re/interactiveshell.py:3063: DtypeWarning: Columns (33) have mixed types.
Specify dtype option on import or set low_memory=False.
  interactivity=interactivity, compiler=compiler, result=result)
```

In [4]:

```
df.head()
```

Out[4]:

	SEVERITYCODE	X	Y	OBJECTID	INCKEY	COLDETKEY	REPORTNO	STATUS
0	2	-122.323148	47.703140	1	1307	1307	3502005	Matchec
1	1	-122.347294	47.647172	2	52200	52200	2607959	Matchec
2	1	-122.334540	47.607871	3	26700	26700	1482393	Matchec
3	1	-122.334803	47.604803	4	1144	1144	3503937	Matchec
4	2	-122.306426	47.545739	5	17700	17700	1807429	Matchec

5 rows × 38 columns

In [5]:

```
df.dtypes
```

Out[5]:

```
SEVERITYCODE        int64
X                 float64
Y                 float64
OBJECTID            int64
INCKEY              int64
COLDETKEY           int64
REPORTNO           object
STATUS             object
ADDRTYPE           object
INTKEY            float64
LOCATION           object
EXCEPTRSNCODE      object
EXCEPTRSNDESC      object
SEVERITYCODE.1      int64
SEVERITYDESC       object
COLLISIONTYPE      object
PERSONCOUNT         int64
PEDCOUNT            int64
PEDCYLCOUNT         int64
VEHCOUNT            int64
INCDATE            object
INCDTTM            object
JUNCTIONTYPE       object
SDOT_COLCODE        int64
SDOT_COLDESC       object
INATTENTIONIND     object
UNDERINFL          object
WEATHER            object
ROADCOND           object
LIGHTCOND          object
PEDROWNOTGRNT      object
SDOTCOLNUM        float64
SPEEDING           object
ST_COLCODE         object
ST_COLDESC         object
SEGLANEKEY          int64
CROSSWALKKEY        int64
HITPARKEDCAR       object
dtype: object
```

We are only going to need to columns of SEVERITYCODE, WEATHER, ROADCOND and LIGHTCOND. So either we can drop rest of the columns or create a new dataframe.

In [6]:

```
df_2 = df[['SEVERITYCODE','WEATHER','ROADCOND','LIGHTCOND']]
```

In [7]:
```
df_2
```

Out[7]:

	SEVERITYCODE	WEATHER	ROADCOND	LIGHTCOND
0	2	Overcast	Wet	Daylight
1	1	Raining	Wet	Dark - Street Lights On
2	1	Overcast	Dry	Daylight
3	1	Clear	Dry	Daylight
4	2	Raining	Wet	Daylight
...	...	...	...	...
194668	2	Clear	Dry	Daylight
194669	1	Raining	Wet	Daylight
194670	2	Clear	Dry	Daylight
194671	2	Clear	Dry	Dusk
194672	1	Clear	Wet	Daylight

194673 rows × 4 columns

In [8]:
```
df_2.dtypes
```

Out[8]:
```
SEVERITYCODE     int64
WEATHER         object
ROADCOND        object
LIGHTCOND       object
dtype: object
```

As we have converted them into integers, let's check for the number of values and remove any NaN values if there are any.

In [9]:
```
df_2.count()
```

Out[9]:
```
SEVERITYCODE    194673
WEATHER         189592
ROADCOND        189661
LIGHTCOND       189503
dtype: int64
```

In [10]:
```
df_final = df_2.dropna()
```

In [11]:
```
df_final
```

Out[11]:

	SEVERITYCODE	WEATHER	ROADCOND	LIGHTCOND
0	2	Overcast	Wet	Daylight
1	1	Raining	Wet	Dark - Street Lights On
2	1	Overcast	Dry	Daylight
3	1	Clear	Dry	Daylight
4	2	Raining	Wet	Daylight
...	...	...	...	...
194668	2	Clear	Dry	Daylight
194669	1	Raining	Wet	Daylight
194670	2	Clear	Dry	Daylight
194671	2	Clear	Dry	Dusk
194672	1	Clear	Wet	Daylight

189337 rows × 4 columns

In [12]:
```
df_final.isnull().values.any()
```

Out[12]:
```
False
```

In [13]:
```
df_final.isnull().sum()
```

Out[13]:
```
SEVERITYCODE    0
WEATHER         0
ROADCOND        0
LIGHTCOND       0
dtype: int64
```

In [14]:
```
df_final.count()
```

Out[14]:
```
SEVERITYCODE    189337
WEATHER         189337
ROADCOND        189337
LIGHTCOND       189337
dtype: int64
```

Let's check for their data types

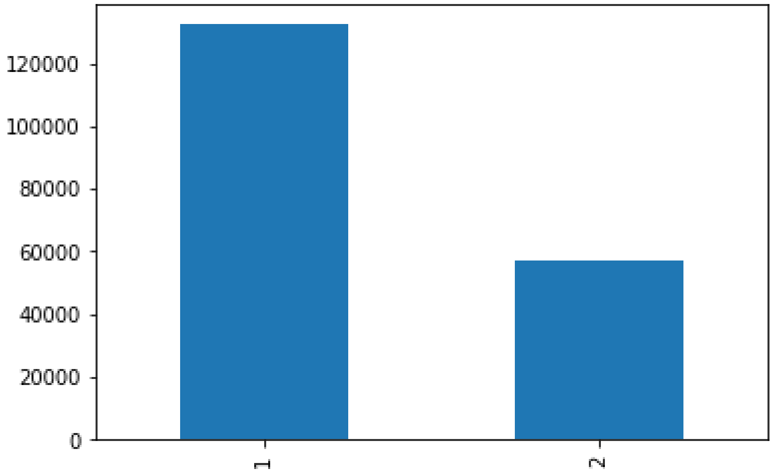
```
In [15]: df_final.dtypes
```

```
Out[15]: SEVERITYCODE      int64
         WEATHER          object
         ROADCOND         object
         LIGHTCOND        object
         dtype: object
```

Let's visualize the data.

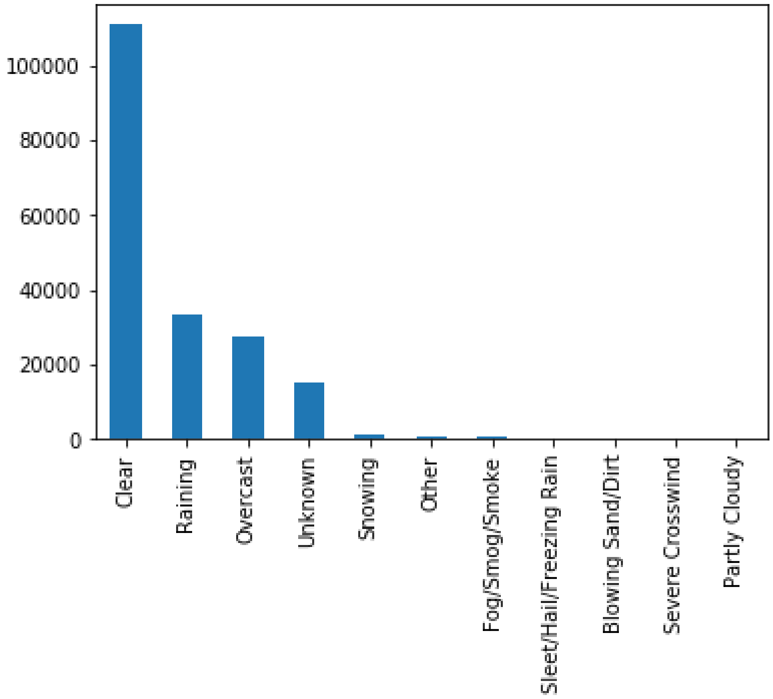
```
In [16]: df_final['SEVERITYCODE'].value_counts().plot(kind = "bar")
```

```
Out[16]: <matplotlib.axes._subplots.AxesSubplot at 0x7fa42e230a90>
```



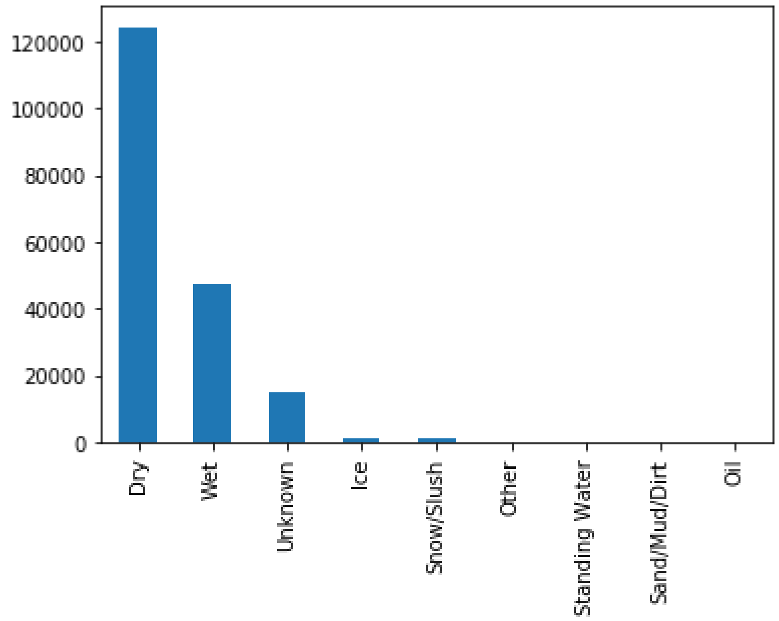
```
In [17]: df_final['WEATHER'].value_counts().plot(kind = "bar")
```

```
Out[17]: <matplotlib.axes._subplots.AxesSubplot at 0x7fa42e46ad50>
```



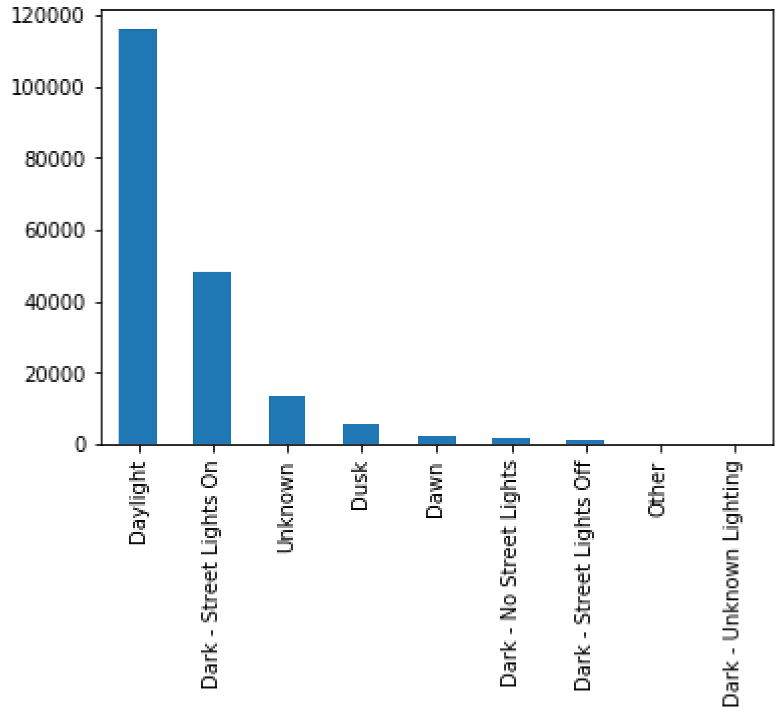
In [18]:
```
df_final['ROADCOND'].value_counts().plot(kind = "bar")
```

Out[18]:
```
<matplotlib.axes._subplots.AxesSubplot at 0x7fa42e7ab1d0>
```



In [20]:
```
df_final['LIGHTCOND'].value_counts().plot(kind = "bar")
```

Out[20]:
```
<matplotlib.axes._subplots.AxesSubplot at 0x7fa42e820810>
```



In [24]:
```
df_final['SEVERITYCODE'].value_counts()
```

Out[24]:
```
1    132285
2     57052
Name: SEVERITYCODE, dtype: int64
```

In [25]:
```
df_final['WEATHER'].value_counts()
```

Out[25]:
```
Clear                       111008
Raining                      33117
Overcast                     27681
Unknown                      15039
Snowing                        901
Other                          824
Fog/Smog/Smoke                 569
Sleet/Hail/Freezing Rain       113
Blowing Sand/Dirt               55
Severe Crosswind                25
Partly Cloudy                    5
Name: WEATHER, dtype: int64
```

In [26]:
```
df_final['ROADCOND'].value_counts()
```

Out[26]:
```
Dry               124300
Wet                47417
Unknown            15031
Ice                 1206
Snow/Slush           999
Other                131
Standing Water       115
Sand/Mud/Dirt         74
Oil                   64
Name: ROADCOND, dtype: int64
```

In [27]:
```
df_final['LIGHTCOND'].value_counts()
```

Out[27]:
```
Daylight                    116077
Dark - Street Lights On      48440
Unknown                      13456
Dusk                          5889
Dawn                          2502
Dark - No Street Lights       1535
Dark - Street Lights Off      1192
Other                          235
Dark - Unknown Lighting         11
Name: LIGHTCOND, dtype: int64
```

In [28]:
```
df_final.dtypes
```

Out[28]:
```
SEVERITYCODE      int64
WEATHER          object
ROADCOND         object
LIGHTCOND        object
dtype: object
```

As we can see the data types are not integers, we need to convert it to integer

In [29]:

```
df_final["WEATHER"] = df["WEATHER"].astype('category').cat.codes
df_final["ROADCOND"] = df["ROADCOND"].astype('category').cat.codes
df_final["LIGHTCOND"] = df["LIGHTCOND"].astype('category').cat.codes
```

```
/Users/chirayusharma/opt/anaconda3/lib/python3.7/site-packages/ipykernel_
launcher.py:1: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-do
cs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy (http
s://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returni
ng-a-view-versus-a-copy)
  """Entry point for launching an IPython kernel.
/Users/chirayusharma/opt/anaconda3/lib/python3.7/site-packages/ipykernel_
launcher.py:2: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-do
cs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy (http
s://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returni
ng-a-view-versus-a-copy)

/Users/chirayusharma/opt/anaconda3/lib/python3.7/site-packages/ipykernel_
launcher.py:3: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-do
cs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy (http
s://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returni
ng-a-view-versus-a-copy)
  This is separate from the ipykernel package so we can avoid doing impor
ts until
```

In [30]:

```
df_final.head()
```

Out[30]:

	SEVERITYCODE	WEATHER	ROADCOND	LIGHTCOND
0	2	4	8	5
1	1	6	8	2
2	1	4	0	5
3	1	1	0	5
4	2	6	8	5

df_final.dtypes

In [36]:
```
df_final.count()
```

Out[36]:
```
SEVERITYCODE    189337
WEATHER         189337
ROADCOND        189337
LIGHTCOND       189337
dtype: int64
```

Now, we can apply Machine Learning models to our data

We can use SEVERITYCODE as our target to predict the outcome. The models we can use here are KNN, Decision Trees and Logistic Regression.

(a) Using KNN, we can predict the SEVERITYCODE of an outcome as it finds the most similar data point within k distance.

(b) Using Decision trees, we can get a layout of all possible outcomes. In this, it will observe all possible outcomes of Road, Weather and Visibilty conditions.

(c) Finally, with the help of Logistic Regression we can predict one of the two values as there are only two values for the SEVERITYCODE.

In [38]:
```
df_final['SEVERITYCODE'].value_counts().to_frame()
```

Out[38]:

	SEVERITYCODE
1	132285
2	57052

In [39]:
```
## As we can see that there is no balance in the SEVERITYCODE. This can ham
## So, we need to balance it by downsampling the majority
```

In [40]:
```
# For this, we need to import the package

from sklearn.utils import resample
```

In [41]:
```
df_final_majority = df_final[df_final.SEVERITYCODE==1]
df_final_minority = df_final[df_final.SEVERITYCODE==2]
```

In [42]:
```
## Now, let's Downsample majority class
df_final_majority_downsampled = resample(df_final_majority,
                                         replace=False,
                                         n_samples=57052,
                                         random_state=123)
```

In [43]:

```
## Now, let's combine minority class with downsampled majority class
df_final_balanced = pd.concat([df_final_majority_downsampled, df_final_mino
```

In [48]:

```
df_final_balanced['SEVERITYCODE'].value_counts().to_frame()
```

Out[48]:

	SEVERITYCODE
2	57052
1	57052

Now, it's time to train and test the model

For that, let's normalize

In [63]:

```
df_final_balanced.count()
```

Out[63]:

```
SEVERITYCODE    114104
WEATHER         114104
ROADCOND        114104
LIGHTCOND       114104
dtype: int64
```

In [56]:

```
X = np.asarray(df_final_balanced[['WEATHER', 'ROADCOND', 'LIGHTCOND']])
X[0:5]
```

Out[56]:

```
array([[1, 0, 5],
       [1, 0, 5],
       [1, 0, 5],
       [4, 0, 5],
       [6, 8, 2]], dtype=int8)
```

In [58]:

```
y = np.asarray(df_final_balanced['SEVERITYCODE'])
y [0:5]
```

Out[58]:

```
array([1, 1, 1, 1, 1])
```

In [64]:

```
from sklearn import preprocessing
```

In [65]:

```
X = preprocessing.StandardScaler().fit(X).transform(X)
X[0:5]
```

Out[65]:

```
array([[-0.71907961, -0.69272349,  0.39316776],
       [-0.71907961, -0.69272349,  0.39316776],
       [-0.71907961, -0.69272349,  0.39316776],
       [ 0.39080216, -0.69272349,  0.39316776],
       [ 1.13072334,  1.5045195 , -1.43589428]])
```

Now, we have to train and test the data by dividing 80% for Training and 20% for Testing

In [66]:
```
## Let's import the package

from sklearn.model_selection import train_test_split
```

In [69]:
```
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, ra
print ('Train set:', X_train.shape,  y_train.shape)
print ('Test set:', X_test.shape,  y_test.shape)
```

```
Train set: (91283, 3) (91283,)
Test set: (22821, 3) (22821,)
```

Now we have to build the models :

KNN Model :

In [75]:
```
## let's import the package

from sklearn.neighbors import KNeighborsClassifier
```

In [76]:
```
## Lets consider k value to be 20

k = 20
```

In [77]:
```
## Now, we have to train the model and then predict

knn = KNeighborsClassifier(n_neighbors = k).fit(X_train,y_train)
knn
```

Out[77]:
```
KNeighborsClassifier(algorithm='auto', leaf_size=30, metric='minkowski',
                     metric_params=None, n_jobs=None, n_neighbors=20, p=
2,
                     weights='uniform')
```

In [78]:
```
Knn_yhat = knn.predict(X_test)
Knn_yhat[0:5]
```

Out[78]:
```
array([2, 2, 1, 1, 1])
```

Decision tree Model :

In [79]:
```
## Let's import the package

from sklearn.tree import DecisionTreeClassifier
```

In [80]:

```
Dec_Tree = DecisionTreeClassifier(criterion="entropy", max_depth = 5)
Dec_Tree
```

Out[80]:

```
DecisionTreeClassifier(ccp_alpha=0.0, class_weight=None, criterion='entro
py',
                       max_depth=5, max_features=None, max_leaf_nodes=Non
e,
                       min_impurity_decrease=0.0, min_impurity_split=Non
e,
                       min_samples_leaf=1, min_samples_split=2,
                       min_weight_fraction_leaf=0.0, presort='deprecate
d',
                       random_state=None, splitter='best')
```

In [81]:

```
## Now we have to train the model and predict


Dec_Tree.fit(X_train,y_train)

Dec_Tree_yhat = Dec_Tree.predict(X_test)
```

In [84]:

```
Dec_Tree_yhat [0:5]
```

Out[84]:

```
array([1, 2, 2, 2, 2])
```

In [ ]:

```

```

In [86]:

```
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import confusion_matrix
LR = LogisticRegression(C=5, solver='liblinear').fit(X_train,y_train)
LR
```

Out[86]:

```
LogisticRegression(C=5, class_weight=None, dual=False, fit_intercept=Tru
e,
                   intercept_scaling=1, l1_ratio=None, max_iter=100,
                   multi_class='auto', n_jobs=None, penalty='l2',
                   random_state=None, solver='liblinear', tol=0.0001, ver
bose=0,
                   warm_start=False)
```

In [88]:

```
LR_yhat = LR.predict(X_test)
LR_yhat
```

Out[88]:

```
array([2, 1, 1, ..., 2, 2, 2])
```

In [89]:

```
yhat_prob = LR.predict_proba(X_test)
yhat_prob
```

Out[89]:

```
array([[0.43407687, 0.56592313],
       [0.55117425, 0.44882575],
       [0.57834919, 0.42165081],
       ...,
       [0.49599534, 0.50400466],
       [0.4725284 , 0.5274716 ],
       [0.43407687, 0.56592313]])
```

Results and evaluation through metrics

In [133]:

```
from sklearn.metrics import jaccard_similarity_score
from sklearn.metrics import f1_score
from sklearn.metrics import log_loss
```

For KNN Model :

In [134]:

```
print("For KNN Model, below are the metrics: ")
print("Jaccard Similarity Score:" ,jaccard_similarity_score(y_test, Knn_yha
print("F1-Score : ", f1_score(y_test, Knn_yhat, average='macro'))
```

```
For KNN Model, below are the metrics: 
Jaccard Similarity Score: 0.5054993208010166
F1-Score :  0.4795258816494271
```

For Decision Tree Model :

In [146]:

```
print("For Decision Tree Model, below are the metrics: ")
print("Jaccard Similarity  Score: ", jaccard_similarity_score(y_test, Dec_T
print("F1-Score : ", f1_score(y_test, Dec_Tree_yhat, average='macro'))
```

```
For Decision Tree Model, below are the metrics: 
Jaccard Similarity  Score:  0.5604925288111827
F1-Score :  0.5257715304320439
```

For Logistic Regression :

In [152]:

```
print("For Logistic Regression, below are the metrics: ")
print("Jaccard Similarity score : ", jaccard_similarity_score(y_test, LR_yh
print("F1-Score : ",f1_score(y_test, LR_yhat, average='macro'))

yhat_prob = LR.predict_proba(X_test)
print("Log Loss: ", log_loss(y_test, yhat_prob))
```

```
For Logistic Regression, below are the metrics:
Jaccard Similarity score :  0.5360413654090531
F1-Score :  0.5233137197110393
Log Loss:  0.6819423921545464
```

Discussion :

The libraries I used were

In the beginning of the project, I imported the dataset, saw it's characteristics and picked certain columns I was going to use for further processes. The columns were SEVERITYCODE, WEATHER, ROADCOND and LIGHTCOND. I used EDA to visualize the data and get a better understanding of the dataset in the form of plots. I had object values for the columns I was about to use for analyzing the data("WEATHER","ROADCOND","LIGHTCOND"). I then converted it to integer values. But, I found that the ratio of values was 3:1 in the column SEVERITYCODE which is our target column.

To balance the dataset, I downsampled the majority class to the minority class. After the dataset was balanced, it was all set for building the models.

I then normalized the data for training and testing.

Here, I used KNN, Decision Tree and Logistic Regression to predict the outcome. KNN and DT's are ideal but Logistic Regression will be the best to predict because of it's binary nature because even the values of SEVERITYCODE are in terms of 1's and 2's.

For the mentioned models, the metrics I used were Jaccard Similarity Score, F1 Score and LogLoss.

Conclusion :

Based on the evaluation metrics, I found that the weather, road and visibility conditions makes an impact on the car accident severity i,e, property damage(1) or injury (2).

In [ ]: