Week 3 – Capstone Project
Severity of Car Accidents in Seattle
Project Report

1. Problem Understanding and Target Audience

1.1 Background and Problem Understanding

Car accidents are one of the most common hazards we all face in daily life. From minor fender-benders to catastrophic, multi-car pileups, getting into an accident in a motor vehicle can end a life or change it forever. The Seattle Department of Transportation's annual traffic report illustrates the constant challenge to the city posed by car accidents.

Different tropic conditions, locations, weather conditions, road conditions, light conditions, day of the week, junction type, speed range and other types of factors are major attributes causing the car accidents.

In this project, we focus on the subject of predicting the severity of an accident in the city of Seattle. Some attributes will be evaluated like weather and road conditions which contribute in the severity of the car accidents.

1.2 Target Audience

This project will be most beneficial for:

- People who travel on a regular bases by car.
- Truck drivers.
- Police officers who want to reduce the accident rate and severity.

2. Data

2.1 Data resource

The data was retrieved from **https://s3.us.cloud-object-storage.appdomain.cloud/cf-courses-data/CognitiveClass/DP0701EN/version-2/Data-Collisions.csv**

This data for the capstone project was already provided by Applied data Science Capstone course by Coursera.

2.2 Data Description

The Collisions dataset includes records of collisions that happened on road from 2004 to Present. The dataset contains 194673 rows and 38 columns, each row is a record of the accident, and each column is an attribute. The first column "SEVERITYCODE" is the labeled data, which describes the fatality of an accident. The remaining 37 columns have different types of attributes. Some or all can be used to train the model.

There are some problems that need to be fixed in this dataset:

- There are missing values which needs to be removed
- Some columns need to be removed as they are not helpful in building our model
- The data type of some attributes is not correct. For e.g some data needs to be changed from object to integer
- The data has unbalanced labels which needs to be balanced

2.3 Data usage for the solution of the problem

2.3.1 Catplots and Distplots

Catplots and Distplots have been created for the severity of the accidents. The target variable “SEVERITYDESC” i.e a detailed description of the severity of the collision. Independent variables that have been chosen are “ADDRTYPE”, “COLLISIONTYPE”, “JUNCTIONTYPE”, “SDOT_COLCODE”, “ST_COLCODE”, “UNDERINFL”, “ROADCOND”, “LIGHTCOND”, “WEATHER”, and “PERSONCOUNT”.

1. **ADDRTYPE** - Collision address type:
   • Alley
   • Block
   • Intersection

2. **COLLISIONTYPE** – Collision type
3**. JUNCTIONTYPE** - Category of junction at which collision took
place
4. **SDOT_COLCODE** - A code given to the collision by SDOT.
5. **ST_COLCODE** - A code provided by the state that describes the
collision.
6. **UNDERINFL** - Whether or not a driver involved was under the
influence of drugs or alcohol.
7. **ROADCOND** - The condition of the road during the collision.
8. **LIGHTCOND** - The light conditions during the collision.
9. **WEATHER** - A description of the weather conditions during
the time of the collision.
10. **PERSONCOUNT** - The total number of people involved in the
Collision
11. **SEVERITYDESC** - A detailed description of the severity of the
Collision

2.3.2 Seattle map for the Severity of Car Accidents

Seattle map for the occurrence of car accidents was built.

2.3.3 K-Nearest Neighbors, Decision Tree, Logistic Regression and ROC Curves

K-Nearest neighbors, decision tree and Logistic Regression algorithms were used for the classification problem to predict the severity of accidents. "SEVERITYCODE" was used as the target variable

SEVERITYCODE - A code that corresponds to the severity of the
collision:
• **3**—fatality
• **2b**—serious injury
• **2**—injury
• **1**—prop damage
• **0**—unknown

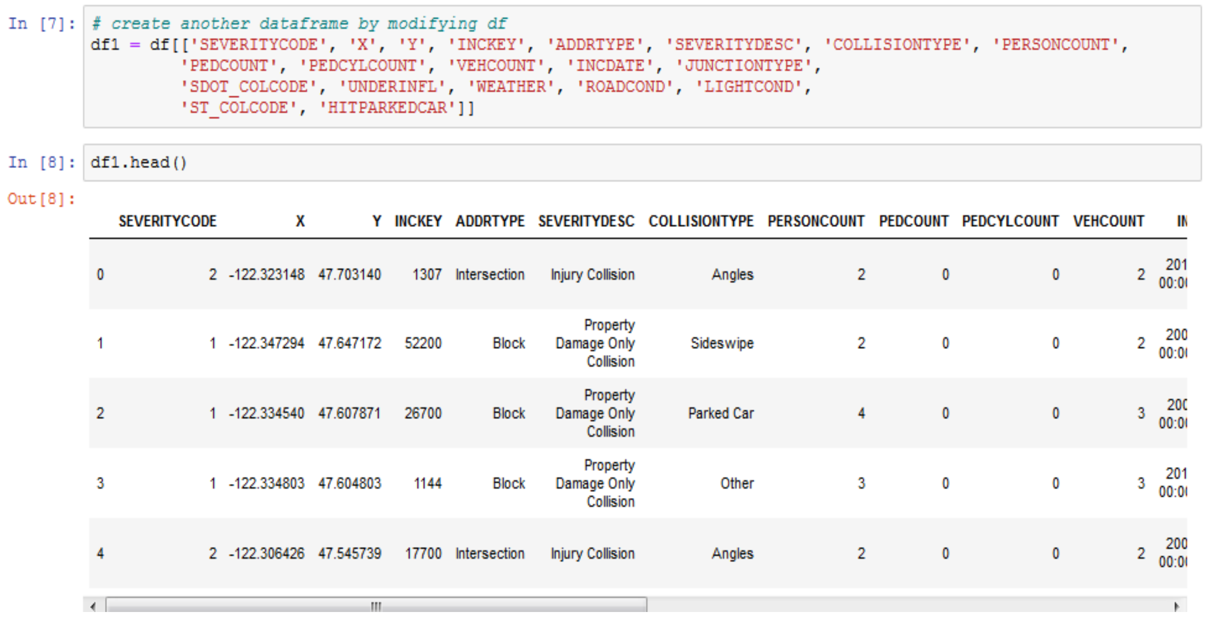
ROC curves were then build to interpret the above models.

2.3.4 Data Processing

We will fix a few problems with the data in order to work with it. We will perform data preprocessing procedures:

- Feature Selection

We will remove some of the attributes that are not needed in order to build our model. We will create a new data frame in and save the attributes that are needed in it.

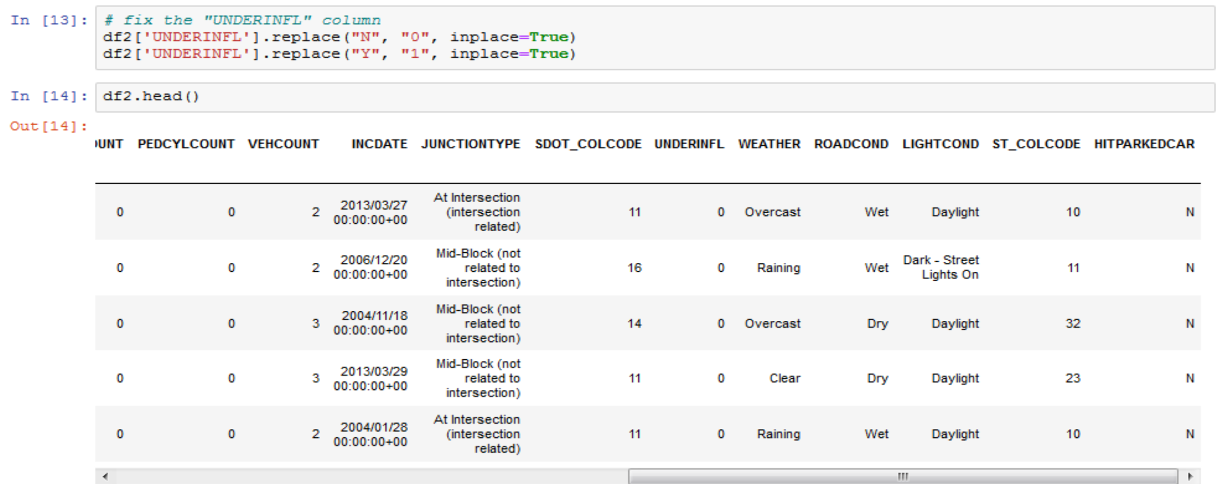


```
In [7]: # create another dataframe by modifying df
        df1 = df[['SEVERITYCODE', 'X', 'Y', 'INCKEY', 'ADDRTYPE', 'SEVERITYDESC', 'COLLISIONTYPE', 'PERSONCOUNT',
                 'PEDCOUNT', 'PEDCYLCOUNT', 'VEHCOUNT', 'INCDATE', 'JUNCTIONTYPE',
                 'SDOT_COLCODE', 'UNDERINFL', 'WEATHER', 'ROADCOND', 'LIGHTCOND',
                 'ST_COLCODE', 'HITPARKEDCAR']]

In [8]: df1.head()
Out[8]:
```

	SEVERITYCODE	X	Y	INCKEY	ADDRTYPE	SEVERITYDESC	COLLISIONTYPE	PERSONCOUNT	PEDCOUNT	PEDCYLCOUNT	VEHCOUNT	IN
0	2	-122.323148	47.703140	1307	Intersection	Injury Collision	Angles	2	0	0	2	201 00:0
1	1	-122.347294	47.647172	52200	Block	Property Damage Only Collision	Sideswipe	2	0	0	2	200 00:0
2	1	-122.334540	47.607871	26700	Block	Property Damage Only Collision	Parked Car	4	0	0	3	200 00:0
3	1	-122.334803	47.604803	1144	Block	Property Damage Only Collision	Other	3	0	0	3	201 00:0
4	2	-122.306426	47.545739	17700	Intersection	Injury Collision	Angles	2	0	0	2	200 00:0

- Make the data consistent

We will convert all “Y”/”N” to “1”/”0” of the attribute “UNDERINFL”



```
In [13]: # fix the "UNDERINFL" column
         df2['UNDERINFL'].replace("N", "0", inplace=True)
         df2['UNDERINFL'].replace("Y", "1", inplace=True)

In [14]: df2.head()
Out[14]:
```

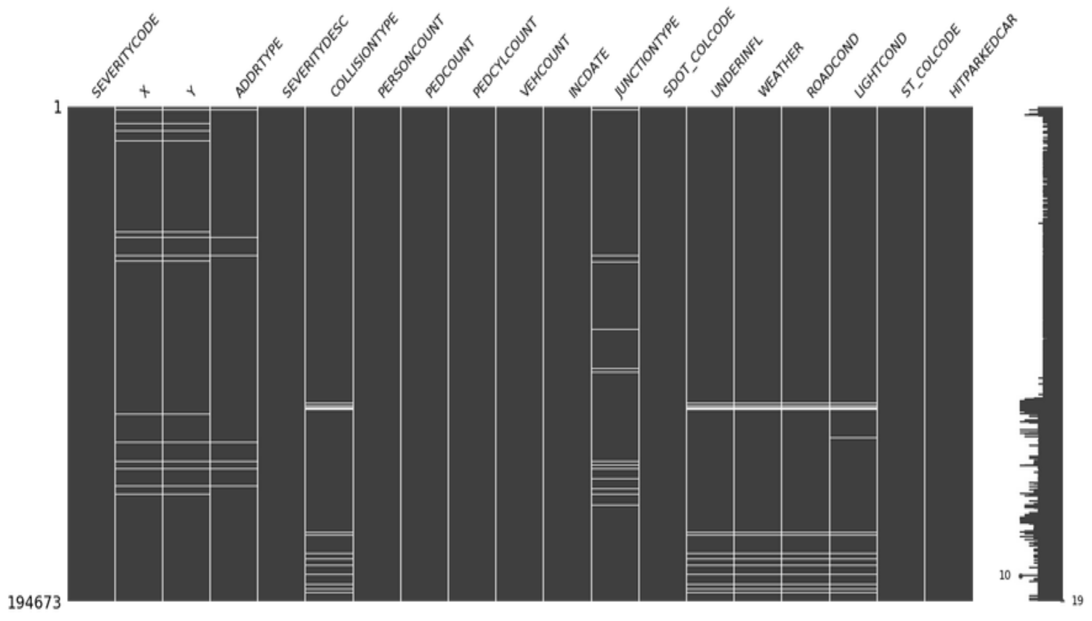
UNT	PEDCYLCOUNT	VEHCOUNT	INCDATE	JUNCTIONTYPE	SDOT_COLCODE	UNDERINFL	WEATHER	ROADCOND	LIGHTCOND	ST_COLCODE	HITPARKEDCAR
0	0	2	2013/03/27 00:00:00+00	At Intersection (intersection related)	11	0	Overcast	Wet	Daylight	10	N
0	0	2	2006/12/20 00:00:00+00	Mid-Block (not related to intersection)	16	0	Raining	Wet	Dark - Street Lights On	11	N
0	0	3	2004/11/18 00:00:00+00	Mid-Block (not related to intersection)	14	0	Overcast	Dry	Daylight	32	N
0	0	3	2013/03/29 00:00:00+00	Mid-Block (not related to intersection)	11	0	Clear	Dry	Daylight	23	N
0	0	2	2004/01/28 00:00:00+00	At Intersection (intersection related)	11	0	Raining	Wet	Daylight	10	N

- Missing Values

We will check the missing values through importing the library **“missingno”.**

We can see from the chart below that “X” and “Y” values miss at the same time. Also, the values of “SDOT_COLCODE”, “UNDERINFL”, “WEATHER”, “ROADCOND” and “LIGHTCOND” miss at the same time.

```
In [16]: # use the missingno library for the exploratory visualization of missing data
         import missingno as msno
         msno.matrix(df2)

Out[16]: <matplotlib.axes._subplots.AxesSubplot at 0x107e6088>
```



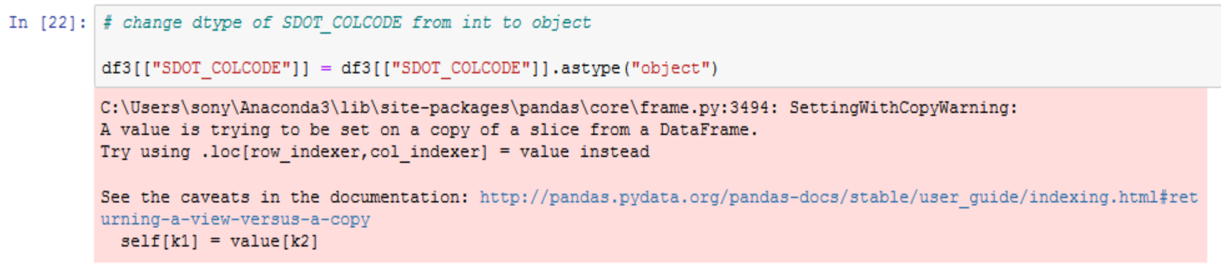
We then drop all rows with missing values by using the function **“dropna”** and axis=0 for rows.

```
In [17]: # let's modify df2 to create df3

         df3 = df2.dropna(axis = 0)     # drop rows with null values

In [18]: df3.head()
Out[18]:
```

INCKEY	SEVERITYCODE	X	Y	ADDRTYPE	SEVERITYDESC	COLLISIONTYPE	PERSONCOUNT	PEDCOUNT	PEDCYLCOUNT	VEHCOUNT	INCD
1307	2	-122.323148	47.703140	Intersection	Injury Collision	Angles	2	0	0	2	2013/0 00:00:0
52200	1	-122.347294	47.647172	Block	Property Damage Only Collision	Sideswipe	2	0	0	2	2006/1 00:00:0
26700	1	-122.334540	47.607871	Block	Property Damage Only Collision	Parked Car	4	0	0	3	2004/1 00:00:0
1144	1	-122.334803	47.604803	Block	Property Damage Only Collision	Other	3	0	0	3	2013/0 00:00:0
17700	2	-122.306426	47.545739	Intersection	Injury Collision	Angles	2	0	0	2	2004/0 00:00:0

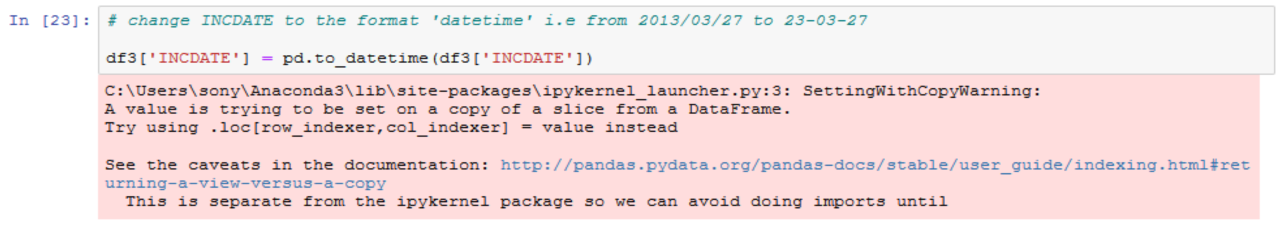
- Correct data format and change data type

We change the data type of SDOT_COLCODE from “**int**” to “**object”.**

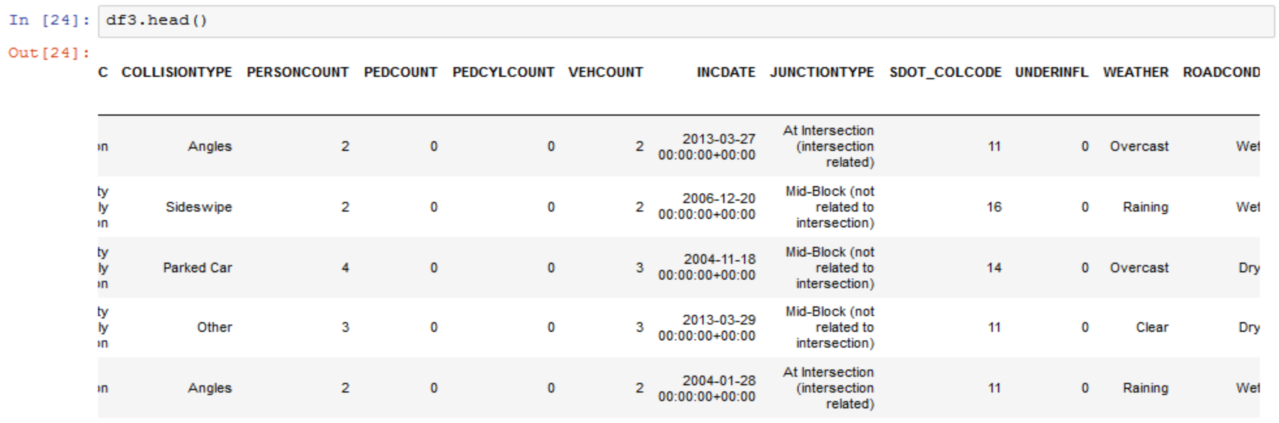


```
In [22]: # change dtype of SDOT_COLCODE from int to object

         df3[["SDOT_COLCODE"]] = df3[["SDOT_COLCODE"]].astype("object")

C:\Users\sony\Anaconda3\lib\site-packages\pandas\core\frame.py:3494: 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: http://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#ret
urning-a-view-versus-a-copy
  self[k1] = value[k2]
```

We also change the datetime format of “INDICATE” to i.e from **2013/03/27** to **23-03-27.**



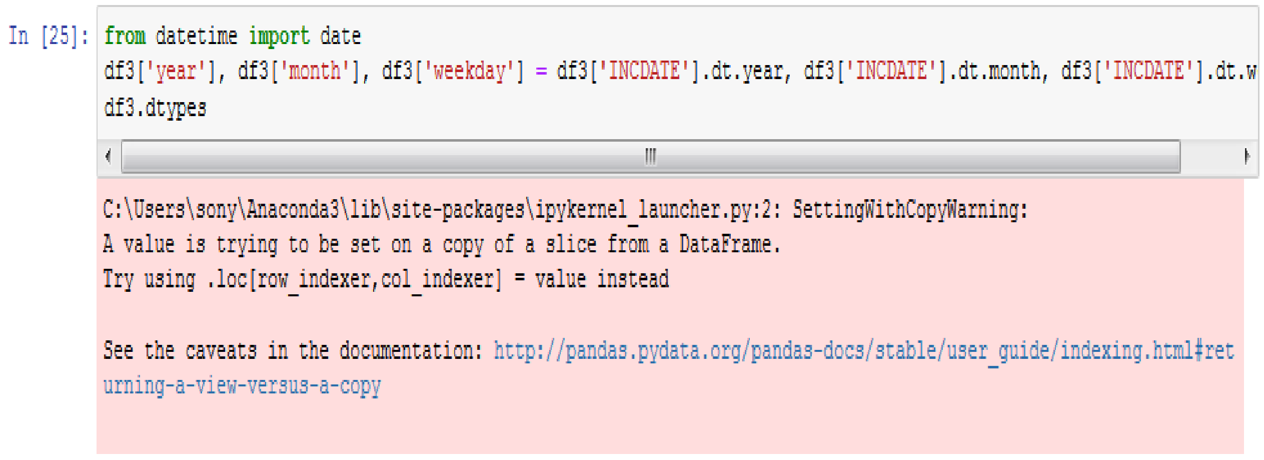
```
In [23]: # change INCDATE to the format 'datetime' i.e from 2013/03/27 to 23-03-27

         df3['INCDATE'] = pd.to_datetime(df3['INCDATE'])

C:\Users\sony\Anaconda3\lib\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: http://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#ret
urning-a-view-versus-a-copy
  This is separate from the ipykernel package so we can avoid doing imports until
```



```
In [24]: df3.head()
Out[24]:
```

C	COLLISIONTYPE	PERSONCOUNT	PEDCOUNT	PEDCYLCOUNT	VEHCOUNT	INCDATE	JUNCTIONTYPE	SDOT_COLCODE	UNDERINFL	WEATHER	ROADCOND
n	Angles	2	0	0	2	2013-03-27 00:00:00+00:00	At Intersection (intersection related)	11	0	Overcast	Wet
ty ly n	Sideswipe	2	0	0	2	2006-12-20 00:00:00+00:00	Mid-Block (not related to intersection)	16	0	Raining	Wet
ty ly n	Parked Car	4	0	0	3	2004-11-18 00:00:00+00:00	Mid-Block (not related to intersection)	14	0	Overcast	Dry
ty ly n	Other	3	0	0	3	2013-03-29 00:00:00+00:00	Mid-Block (not related to intersection)	11	0	Clear	Dry
n	Angles	2	0	0	2	2004-01-28 00:00:00+00:00	At Intersection (intersection related)	11	0	Raining	Wet

Create new variables **“year”**, **“month”**, and **“weekday”** from “INDICATE”.



```
In [25]: from datetime import date
         df3['year'], df3['month'], df3['weekday'] = df3['INCDATE'].dt.year, df3['INCDATE'].dt.month, df3['INCDATE'].dt.w
         df3.dtypes

C:\Users\sony\Anaconda3\lib\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: http://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#ret
urning-a-view-versus-a-copy
```

3. Methodology

3.1 Exploratory Data Analysis

3.1.1 Catplots for Categorical Data

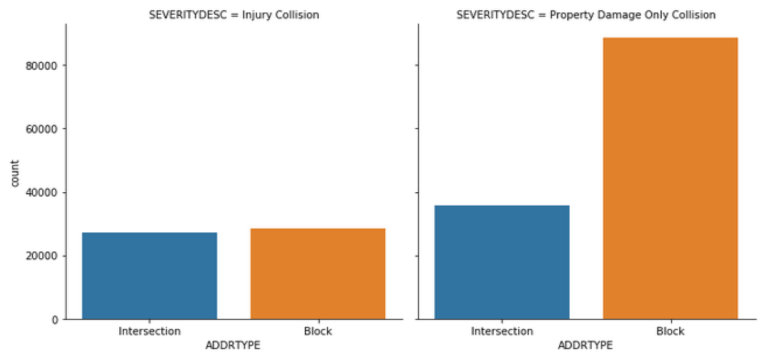
We will create Catplots in order to represent categorical data as most of the attributes in the data are categorical. The target variable is “SEVERITYDESC”.

- ADDRTYPE vs SEVERITYDESC -

Catplots for Categorical data

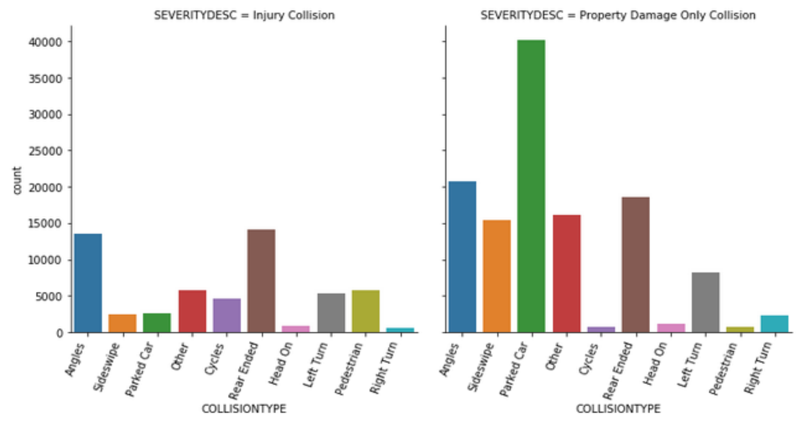
```
In [27]: import seaborn as sns
```

```
In [28]: # create catplot of frequencies of ADDRTYPE

         sns.catplot(x='ADDRTYPE', data=df3, kind='count', col='SEVERITYDESC')
         plt.show()
```



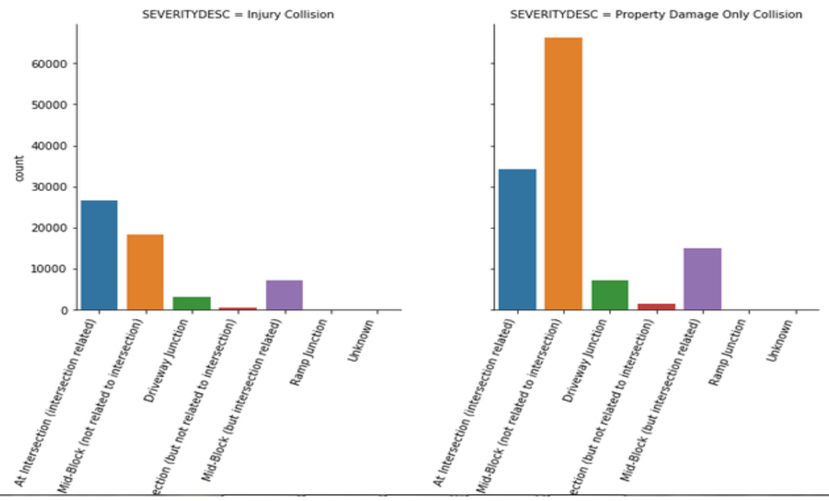
- COLLISIONTYPE vs SEVERITYDESC

```
In [29]: # create catplot of frequency of COLLISIONTYPE

         chart = sns.catplot(x="COLLISIONTYPE", data=df3, kind='count', col='SEVERITYDESC')
         chart.set_xticklabels(rotation=70, horizontalalignment='right')      # add labels of the COLLISIONTYPE

Out[29]: <seaborn.axisgrid.FacetGrid at 0x10980dc8>
```



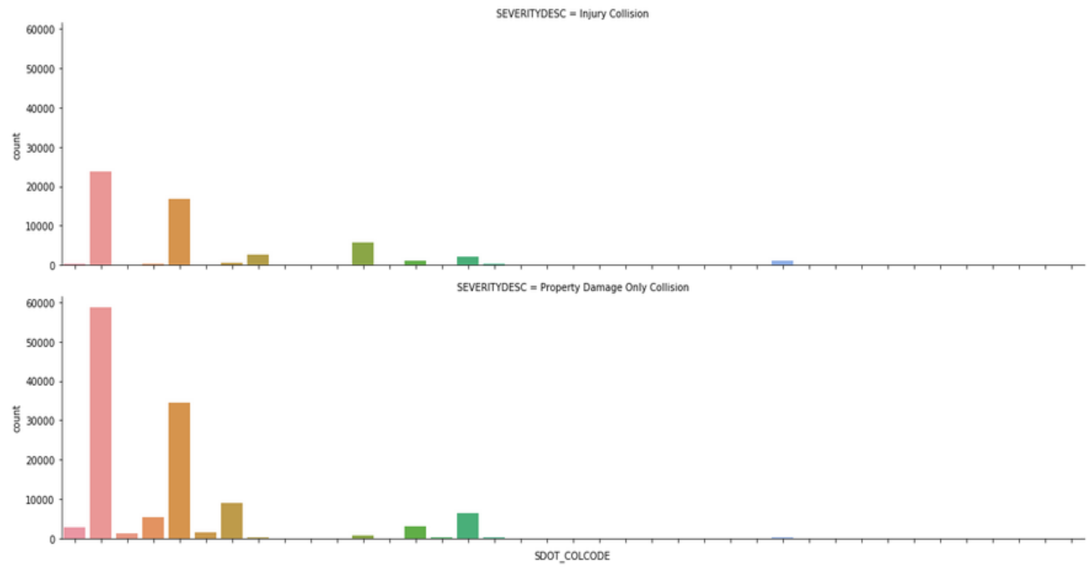
- JUNCTIONTYPE vs SEVERITYDESC

```
In [30]: # plot catplot of frequency of JUNCTIONTYPE

         chart1 = sns.catplot(x='JUNCTIONTYPE', data=df3, kind='count', col='SEVERITYDESC')
         chart1.set_xticklabels(rotation=70, horizontalalignment='right')

Out[30]: <seaborn.axisgrid.FacetGrid at 0x4eb9148>
```



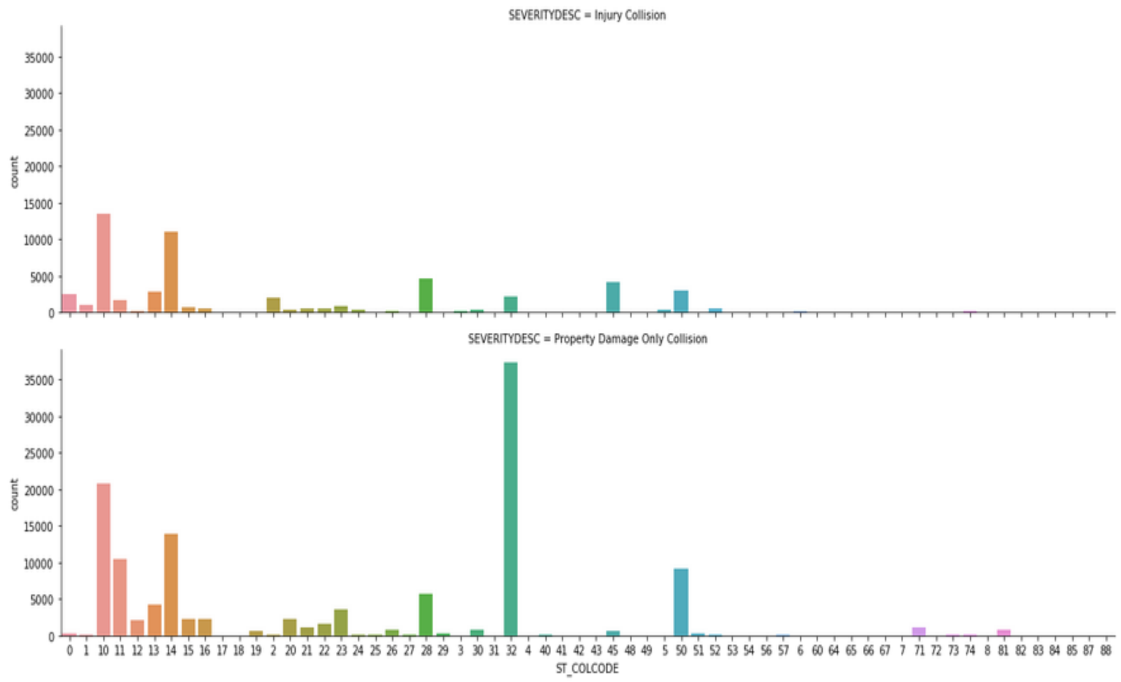
- SDOT_COLCODE vs SEVERITYDESC

```
In [31]: # plot catplot of frequency of SDOT_COLCODE

         chart2 = sns.catplot(x='SDOT_COLCODE', data=df3, kind='count', row='SEVERITYDESC', aspect=4, height=4)
         chart2.set_xticklabels(rotation=70)

Out[31]: <seaborn.axisgrid.FacetGrid at 0x1ae47408>
```



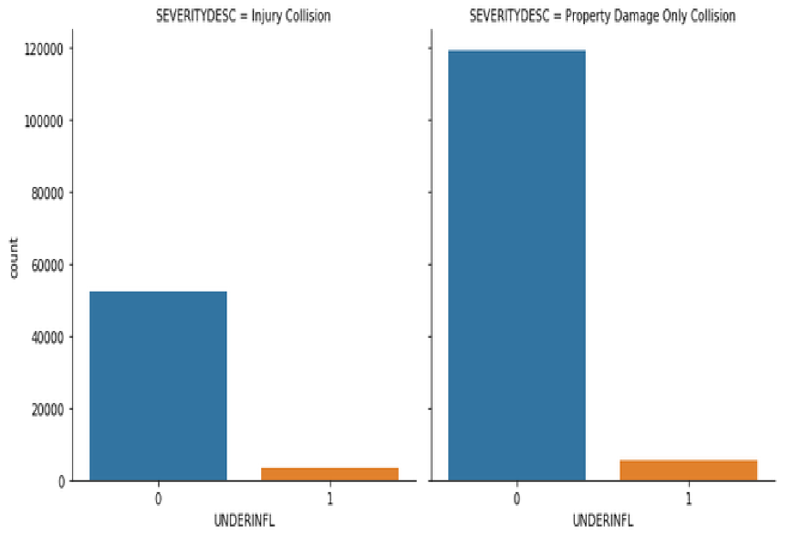
- ST_COLCODE vs SEVERITYDESC

```
In [32]: chart3 = sns.catplot(x = "ST_COLCODE", data = df3, kind = "count",  row= "SEVERITYDESC",aspect=4,height=4)
```



- UNDERINFL vs SEVERITYDESC

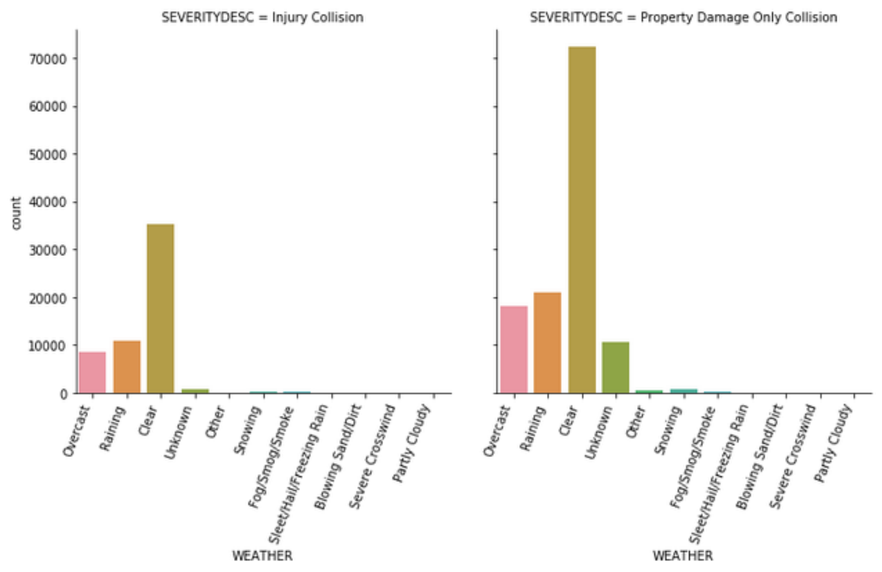
```
In [33]: # plot catplot of frequency of UNDERINFL

         chart4 = sns.catplot(x='UNDERINFL', data=df3, kind='count', col='SEVERITYDESC')
```



- WEATHER vs SEVERITYDESC

```
In [34]: # plot catplot of frequency of WEATHER

         chart5 = sns.catplot(x='WEATHER', data=df3, kind='count', col='SEVERITYDESC')
         chart5.set_xticklabels(rotation=70, horizontalalignment='right')
```

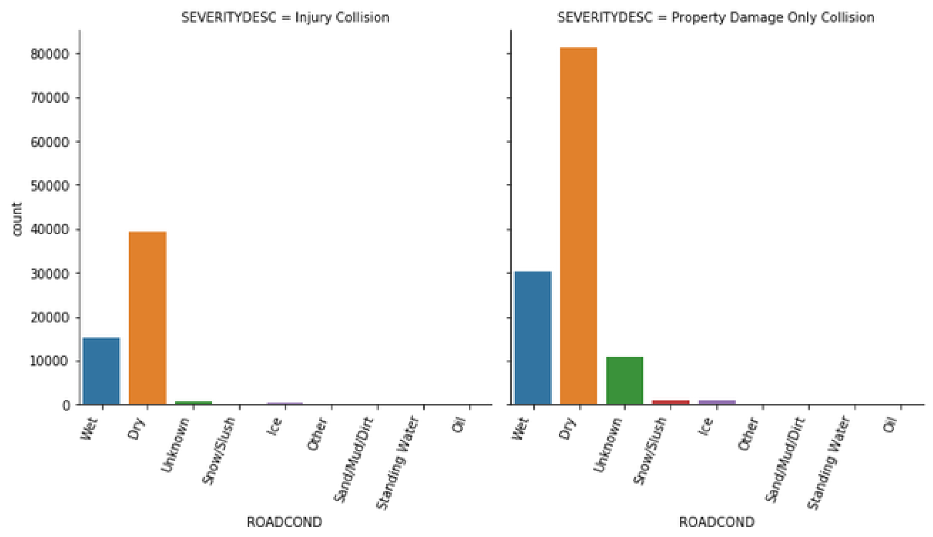
```
Out[34]: <seaborn.axisgrid.FacetGrid at 0x95e9b88>
```



- ROADCOND vs SEVERITYDESC

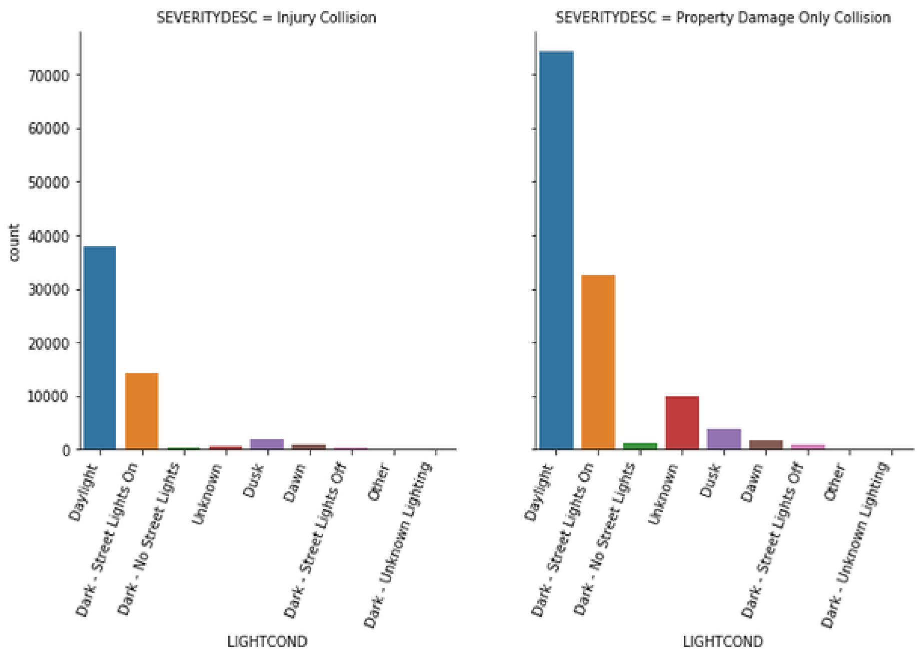
```
In [35]: # plot catplot of frequency of ROADCOND

         chart6 = sns.catplot(x='ROADCOND', data=df3, kind='count', col='SEVERITYDESC')
         chart6.set_xticklabels(rotation=70, horizontalalignment='right')
```

```
Out[35]: <seaborn.axisgrid.FacetGrid at 0x184c15c8>
```



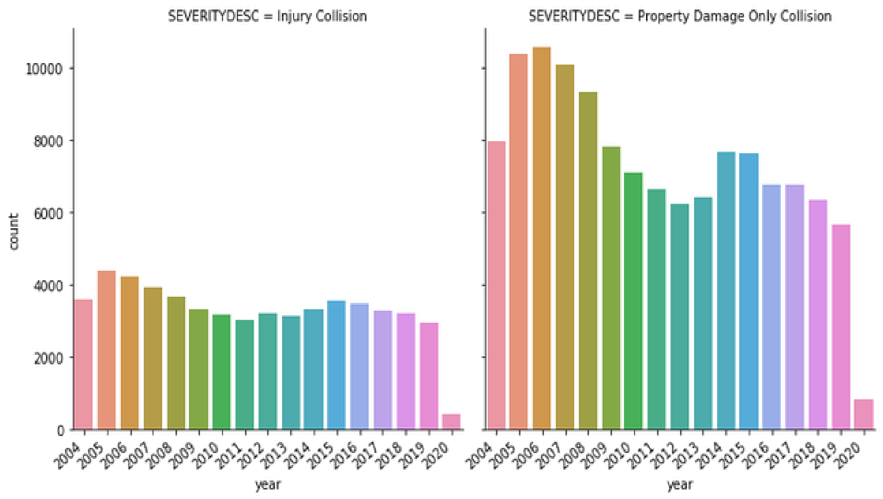
- LIGHTCOND vs SEVERITYDESC

```
In [36]: # plot catplot of frequency of LIGHTCOND

         chart7 = sns.catplot(x='LIGHTCOND', data=df3, kind='count', col='SEVERITYDESC')
         chart7.set_xticklabels(rotation=70, horizontalalignment='right')

Out[36]: <seaborn.axisgrid.FacetGrid at 0x9658d08>
```



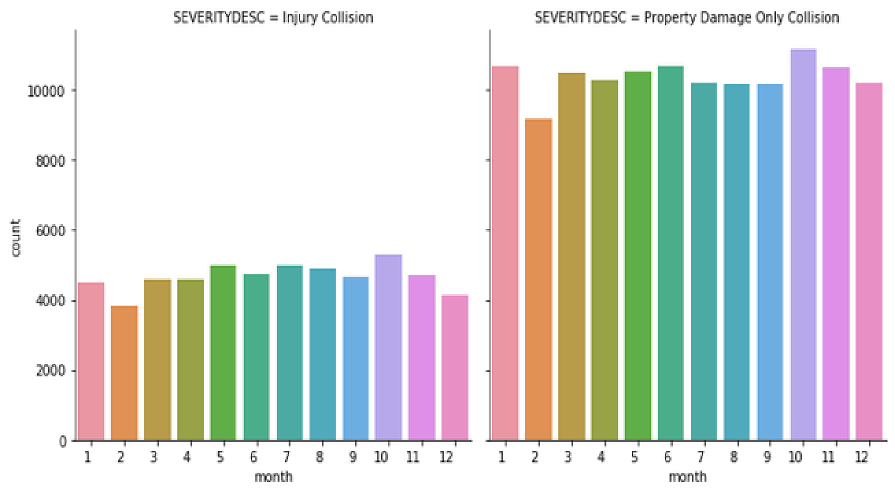
- year vs SEVERITYDESC

```
In [37]: # plot catplot of frequency of year

         chart8 = sns.catplot(x='year', data=df3, kind='count', col='SEVERITYDESC')
         chart8.set_xticklabels(rotation=40, horizontalalignment='right')

Out[37]: <seaborn.axisgrid.FacetGrid at 0x11819a48>
```



- month vs SEVERITYDESC

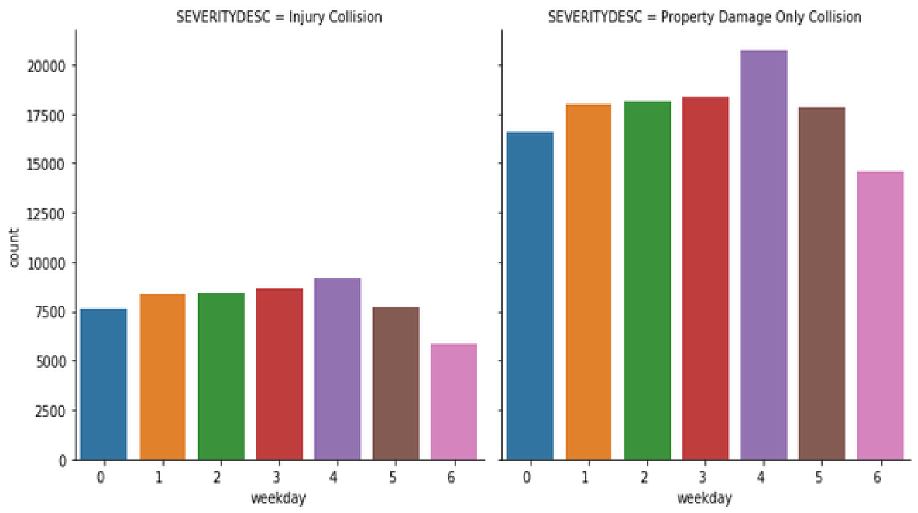
```
In [38]: # plot catplot of frequency of month

         chart9 = sns.catplot(x='month', data=df3, kind='count', col='SEVERITYDESC')
         chart9.set_xticklabels(rotation=0, horizontalalignment='right')
Out[38]: <seaborn.axisgrid.FacetGrid at 0x24758f88>
```



- weekday vs SEVERITYDESC

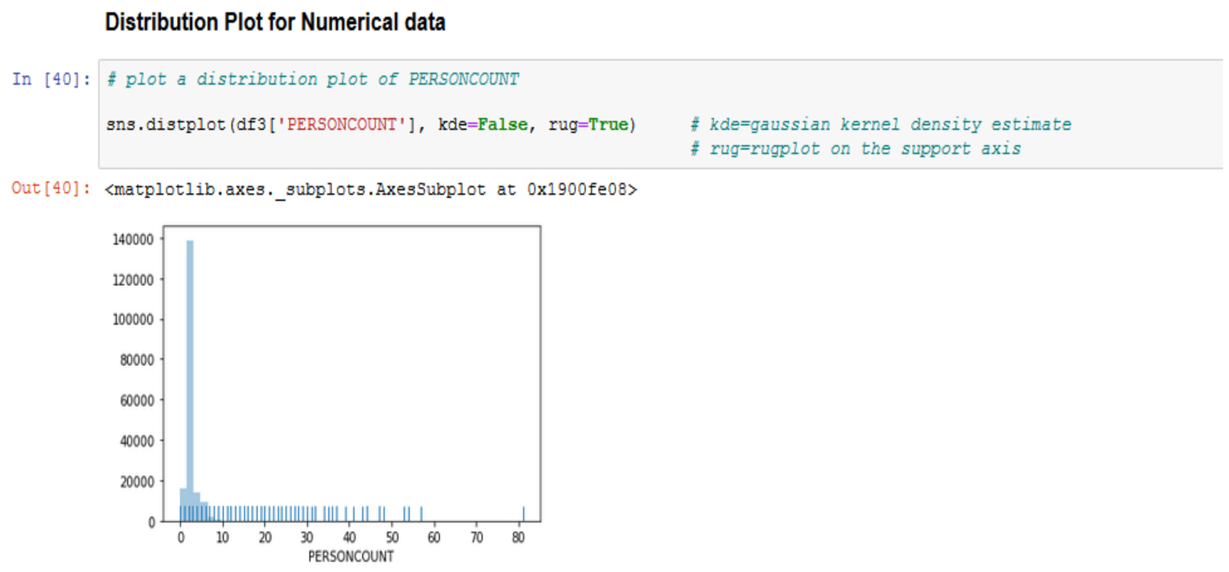
month month

```
In [39]: # plot catplot of frequency of weekday

         chart10 = sns.catplot(x='weekday', data=df3, kind='count', col='SEVERITYDESC')
         chart10.set_xticklabels(rotation=0, horizontalalignment='right')
Out[39]: <seaborn.axisgrid.FacetGrid at 0x133ac7c8>
```



3.1.2 Displots for numerical data

We created a distribution plot for the total number of people involved in the collision

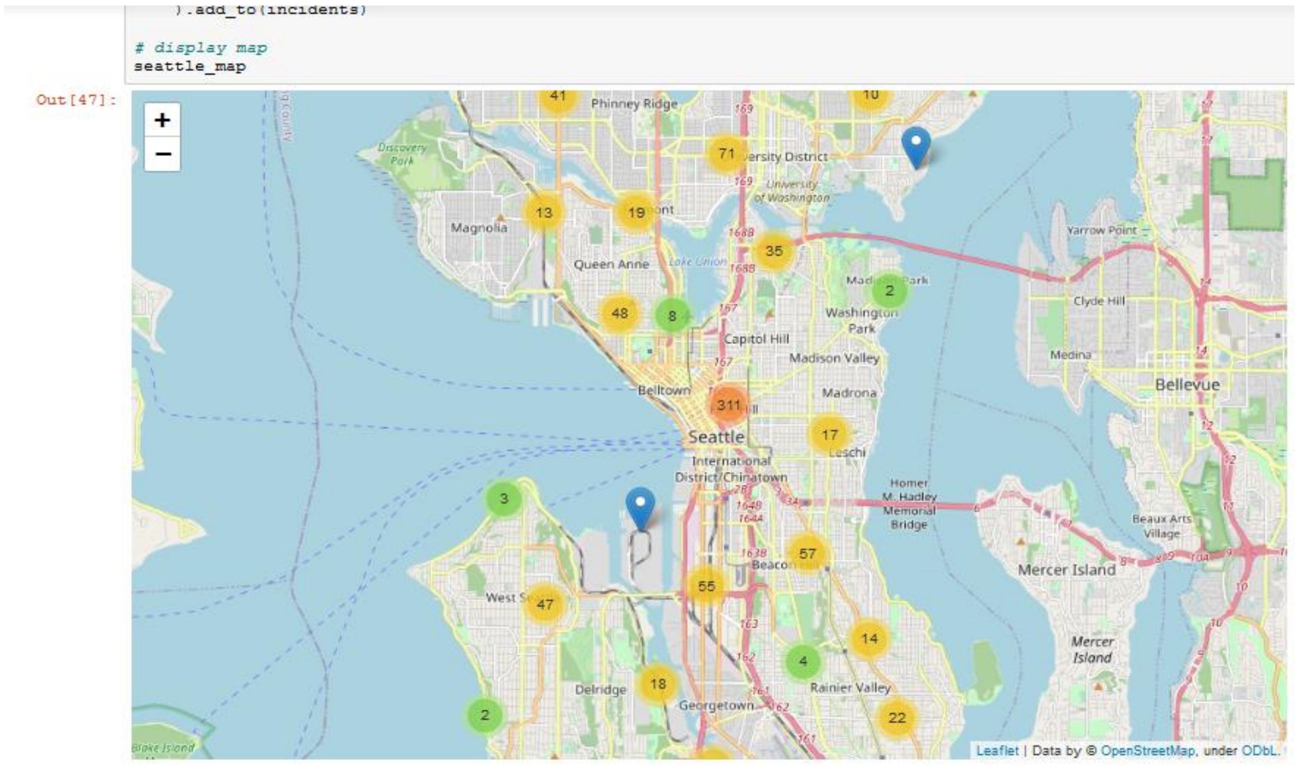


Distribution Plot for Numerical data

```
In [40]: # plot a distribution plot of PERSONCOUNT

         sns.distplot(df3['PERSONCOUNT'], kde=False, rug=True)      # kde=gaussian kernel density estimate
                                                                    # rug=rugplot on the support axis

Out[40]: <matplotlib.axes._subplots.AxesSubplot at 0x1900fe08>
```

3.2 Spatial Analysis

We plotted the map of car accidents in the city of Seattle for the year 2020

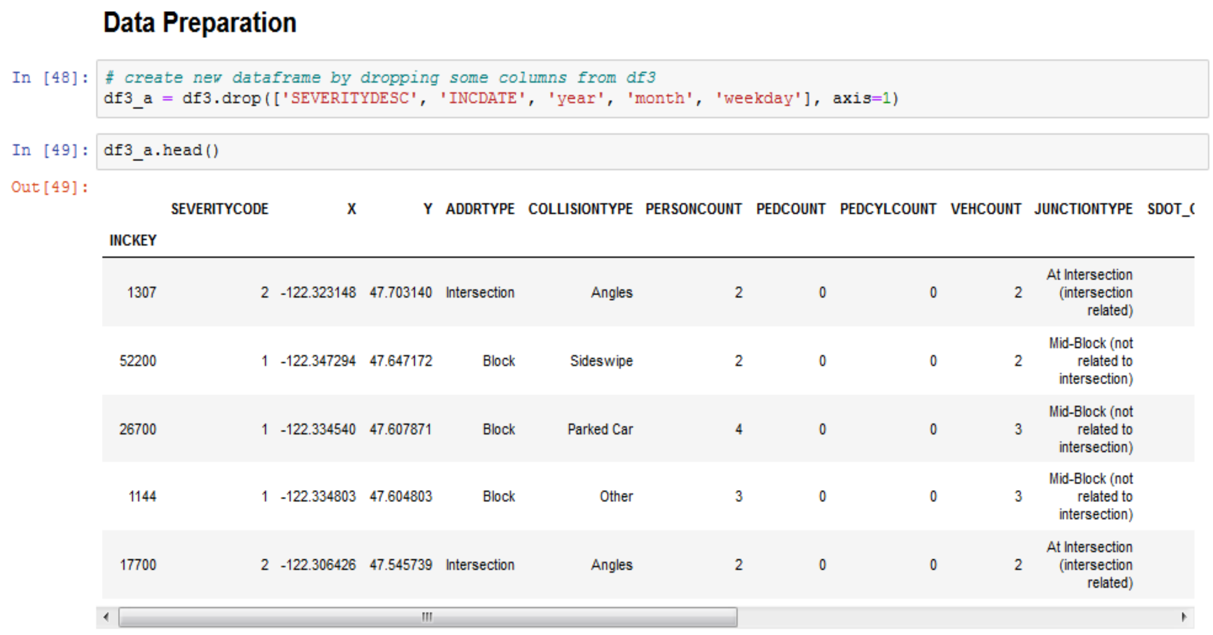


```
    ).add_to(incidents)

# display map
seattle_map

Out[47]:
```

From the data, we can see that most of the accidents were located in the state highway.

3.3 Data Preparation

- **Drop columns with attributes that are not needed.**

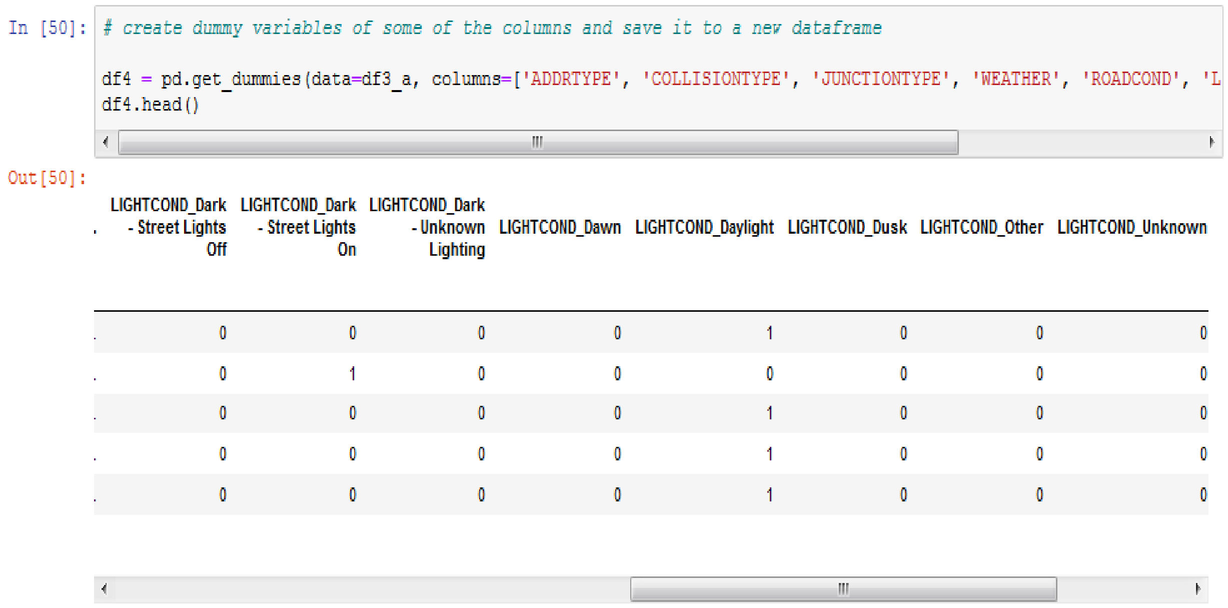


Data Preparation

```
In [48]: # create new dataframe by dropping some columns from df3
         df3_a = df3.drop(['SEVERITYDESC', 'INCDATE', 'year', 'month', 'weekday'], axis=1)

In [49]: df3_a.head()

Out[49]:
```

INCKEY	SEVERITYCODE	X	Y	ADDRTYPE	COLLISIONTYPE	PERSONCOUNT	PEDCOUNT	PEDCYLCOUNT	VEHCOUNT	JUNCTIONTYPE	SDOT_(
1307	2	-122.323148	47.703140	Intersection	Angles	2	0	0	2	At Intersection (intersection related)	
52200	1	-122.347294	47.647172	Block	Sideswipe	2	0	0	2	Mid-Block (not related to intersection)	
26700	1	-122.334540	47.607871	Block	Parked Car	4	0	0	3	Mid-Block (not related to intersection)	
1144	1	-122.334803	47.604803	Block	Other	3	0	0	3	Mid-Block (not related to intersection)	
17700	2	-122.306426	47.545739	Intersection	Angles	2	0	0	2	At Intersection (intersection related)	

- **Create dummies variable for building decision trees.**



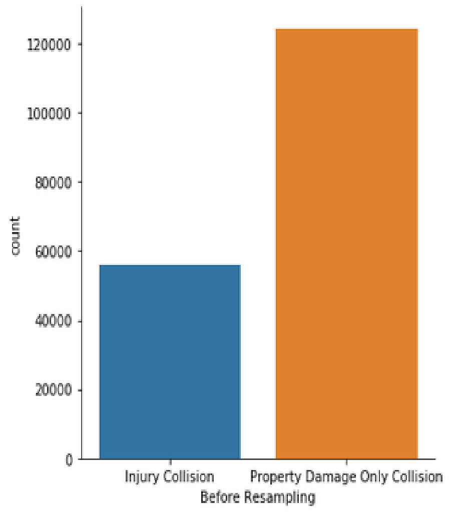
```
In [50]: # create dummy variables of some of the columns and save it to a new dataframe

         df4 = pd.get_dummies(data=df3_a, columns=['ADDRTYPE', 'COLLISIONTYPE', 'JUNCTIONTYPE', 'WEATHER', 'ROADCOND', 'L
         df4.head()

Out[50]:
```

.	LIGHTCOND_Dark - Street Lights Off	LIGHTCOND_Dark - Street Lights On	LIGHTCOND_Dark - Unknown Lighting	LIGHTCOND_Dawn	LIGHTCOND_Daylight	LIGHTCOND_Dusk	LIGHTCOND_Other	LIGHTCOND_Unknown
.	0	0	0	0	1	0	0	0
.	0	1	0	0	0	0	0	0
.	0	0	0	0	1	0	0	0
.	0	0	0	0	1	0	0	0
.	0	0	0	0	1	0	0	0

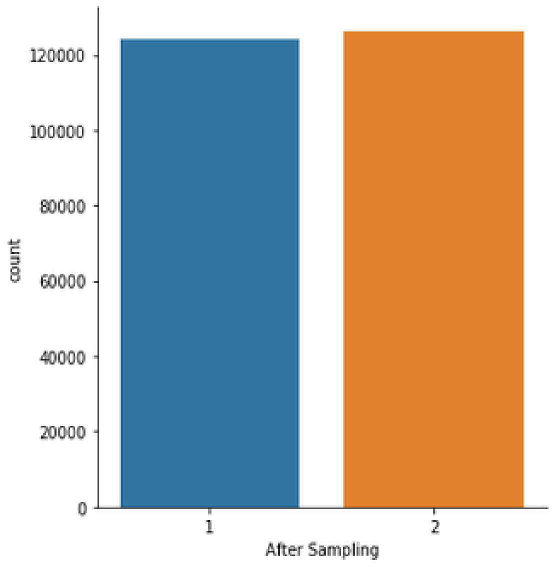
- **Balance unbalanced data**: In our dataset, the labeled response value is imbalanced. There are 136485 obs of label-1 and only 58188 obs of label-2. We need to resample the label-2 data and add more copies of the minority class.

Balanace Labels

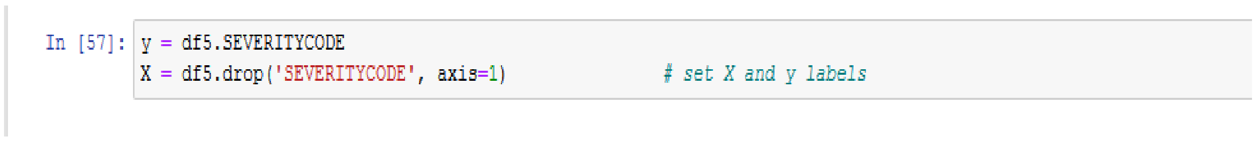
```
In [52]: ax = sns.catplot(x='SEVERITYDESC', data=df3, kind='count')
         ax.set(xlabel='Before Resampling')

Out[52]: <seaborn.axisgrid.FacetGrid at 0x2b2c7108>
```



```
In [55]: ax2 = sns.catplot(x='SEVERITYCODE', data=df5, kind='count')
         ax2.set(xlabel='After Sampling')

Out[55]: <seaborn.axisgrid.FacetGrid at 0x35ead508>
```

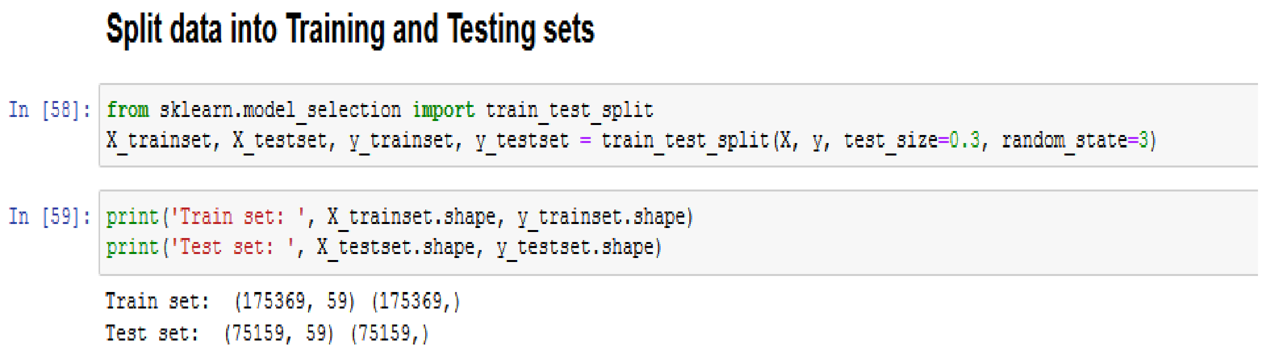


- Set x and y variables. Independent variable is “SEVERITYCODE”



```
In [57]: y = df5.SEVERITYCODE
         X = df5.drop('SEVERITYCODE', axis=1)              # set X and y labels
```

- Split data into training and testing sets: 70% training set and 30% testing set.



Split data into Training and Testing sets

```
In [58]: from sklearn.model_selection import train_test_split
         X_trainset, X_testset, y_trainset, y_testset = train_test_split(X, y, test_size=0.3, random_state=3)

In [59]: print('Train set: ', X_trainset.shape, y_trainset.shape)
         print('Test set: ', X_testset.shape, y_testset.shape)

         Train set:  (175369, 59) (175369,)
         Test set:  (75159, 59) (75159,)
```

3.4 K-Nearest Neighbors

K-nearest neighbors was applied to make predictions about the testing data. We compared the true vale with the testing value. Accuracy score, F1 score and jaccard similarity scores were calculated.

3.5 Decision Tree

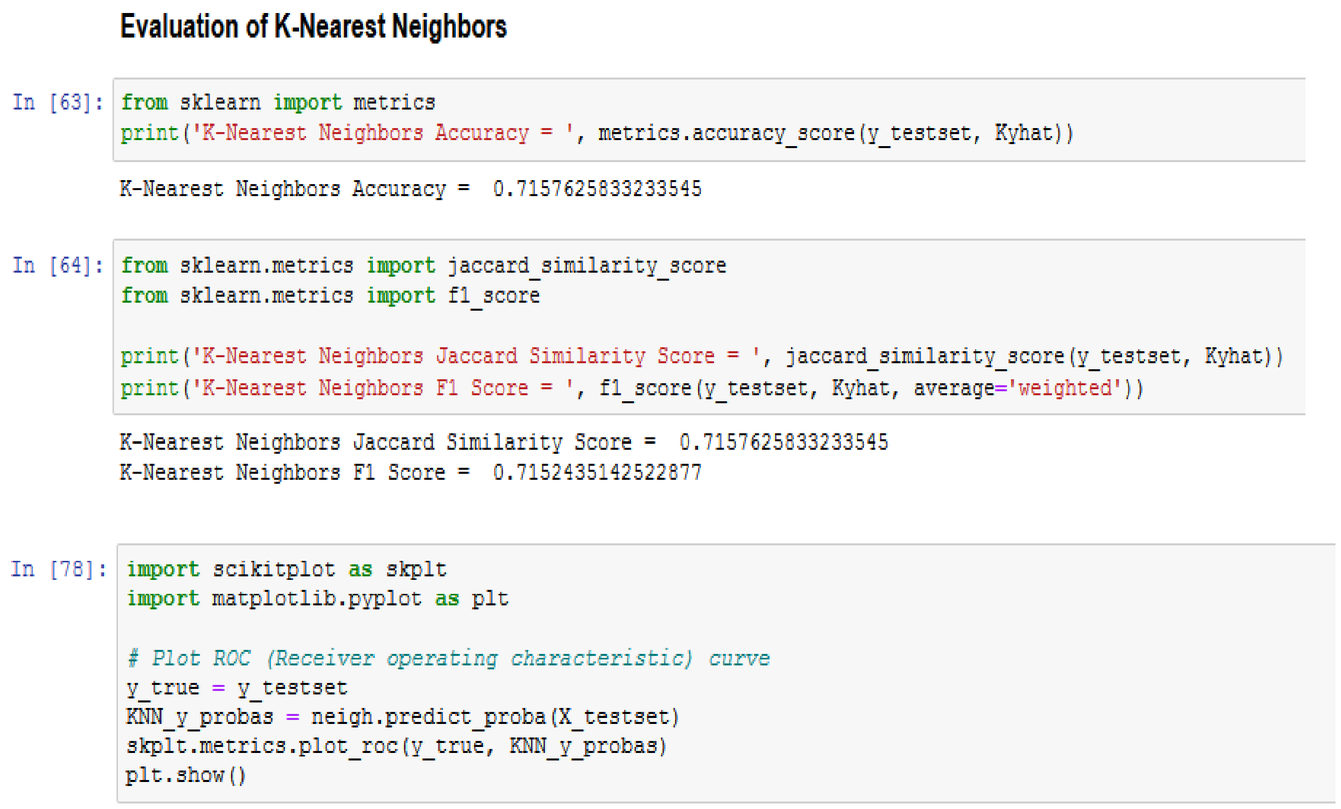
Decision Tree was applied to make predictions about the testing data. We compared the true vale with the testing value. Accuracy score, F1 score and jaccard similarity scores were calculated.

3.6 Logistic Regression

Logistic Regression was applied to make predictions about the testing data. We compared the true vale with the testing value. Accuracy score, F1 score and jaccard similarity scores were calculated.

4. Results

4.1 Performance of K-Nearest Neighbors Model



Evaluation of K-Nearest Neighbors

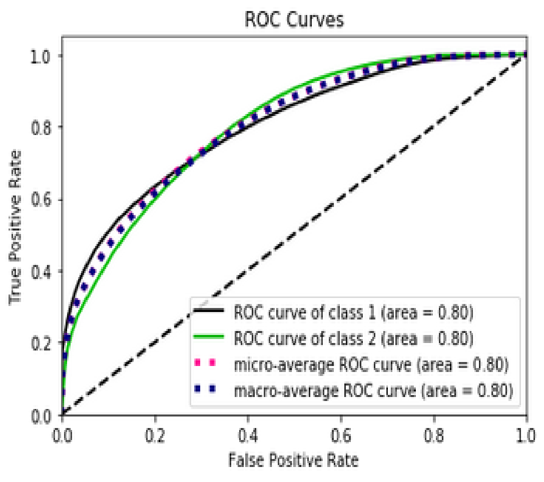
```
In [63]: from sklearn import metrics
         print('K-Nearest Neighbors Accuracy = ', metrics.accuracy_score(y_testset, Kyhat))
```

```
K-Nearest Neighbors Accuracy =  0.7157625833233545
```

```
In [64]: from sklearn.metrics import jaccard_similarity_score
         from sklearn.metrics import f1_score

         print('K-Nearest Neighbors Jaccard Similarity Score = ', jaccard_similarity_score(y_testset, Kyhat))
         print('K-Nearest Neighbors F1 Score = ', f1_score(y_testset, Kyhat, average='weighted'))
```

```
K-Nearest Neighbors Jaccard Similarity Score =  0.7157625833233545
K-Nearest Neighbors F1 Score =  0.7152435142522877
```

```
In [78]: import scikitplot as skplt
         import matplotlib.pyplot as plt

         # Plot ROC (Receiver operating characteristic) curve
         y_true = y_testset
         KNN_y_probas = neigh.predict_proba(X_testset)
         skplt.metrics.plot_roc(y_true, KNN_y_probas)
         plt.show()
```



4.2 Performance of Decision Tree Model

Evaluation of Decision Tree

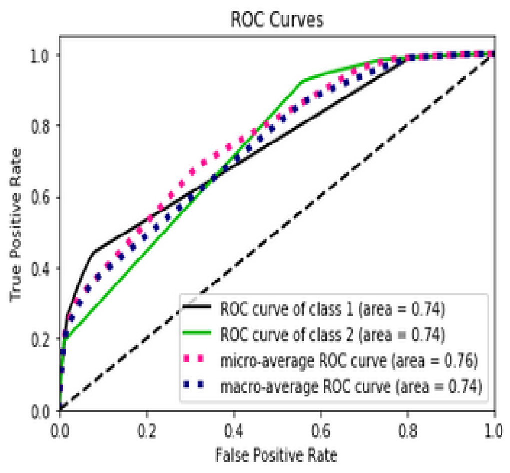
```
In [69]: # Accuracy score

         print("Decision Tree's Accuracy = ", metrics.accuracy_score(y_testset, DTyhat))
```

```
Decision Tree's Accuracy =  0.6833512952540614
```

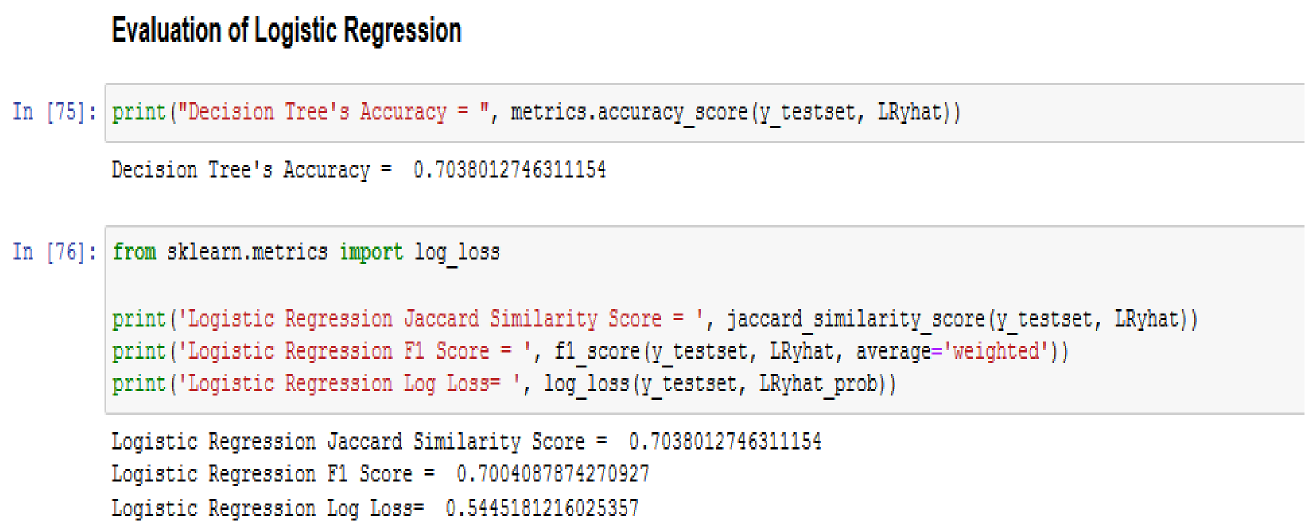
```
In [70]: print('Decision Tree Jaccard Similarity Score = ', jaccard_similarity_score(y_testset, DTyhat))
         print('Decision Tree F1 Score = ', f1_score(y_testset, DTyhat, average='weighted'))
```

```
Decision Tree Jaccard Similarity Score =  0.6833512952540614
Decision Tree F1 Score =  0.6643819514335526
```

```
In [79]: # Plot ROC curve
         y_true = y_testset
         DT_y_probas = DT.predict_proba(X_testset)
         skplt.metrics.plot_roc(y_true, DT_y_probas)
         plt.show()
```



4.3 Performance of Logistic Regression Model



Evaluation of Logistic Regression

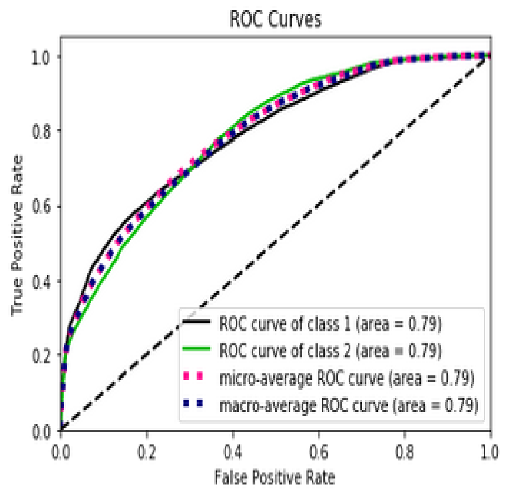
```
In [75]: print("Decision Tree's Accuracy = ", metrics.accuracy_score(y_testset, LRyhat))
```

```
Decision Tree's Accuracy =  0.7038012746311154
```

```
In [76]: from sklearn.metrics import log_loss

         print('Logistic Regression Jaccard Similarity Score = ', jaccard_similarity_score(y_testset, LRyhat))
         print('Logistic Regression F1 Score = ', f1_score(y_testset, LRyhat, average='weighted'))
         print('Logistic Regression Log Loss= ', log_loss(y_testset, LRyhat_prob))
```

```
Logistic Regression Jaccard Similarity Score =  0.7038012746311154
Logistic Regression F1 Score =  0.7004087874270927
Logistic Regression Log Loss=  0.5445181216025357
```

```
In [80]: # Plot ROC Curve
         y_true = y_testset
         skplt.metrics.plot_roc(y_true, LRyhat_prob)
         plt.show()
```



5. Discussion

From the results, we were able to find a lot of information regarding the severity of road accidents in the city of Seattle

- The attributes like road condition, weather, and light conditions did not really contribute in the occurrence of the accidents. We cannot say that these were the major reasons because most accidents happened when these conditions were normal.
- Most accidents occurred in the downtown area of Seattle because there is traffic there as there are more people and therefore, more cars on the road. Most accidents also occurred in the state highways as more people drive along these roads.
- A few accidents occurred on the weekend as it’s a holiday from work so less people drive during the weekend.
- More accidents occurred on Thursday.
- With the results we have, we can predict the severity of an accident with an accuracy of about 70%.

6. Conclusion

In this Capstone Project, the Severity of Accidents in the city of Seattle was analyzed. The data was from the year 2004-2020.

The independed variables that were taken into account were LIGHTCOND, ROADCOND, WEATHER, JUNCTIONTYPE, UNDERINFL, month, year, weekday.

The Target variable were SEVERITYDESC and SEVERITYCODE.

Catplots and Displots were plotted to analyze the data.

Three machine learning models were built to analyze the data i.e K-Nearest Neighbors, Decision Tree, and Logistic Regression.

These models can be very helpful for the target audience in order to minimize the number of accidents.

The traffic system can open more signals on Thursday in order to avoid accidents.

There should be more security even when the weather and road conditions are clear as we saw from the results that more accidents occurred in clear weather.

There should be more warning and speed limit signs.

Overall, this project can be useful to minimize the number of car accidents by taking strict actions in the future.