Week 3 Capstone Project

Severity of Car Accidents in Seattle

Problem Understanding and Target Audience

- 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 a car 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.
- This project will be beneficial for, people who travel on a regular bases by car., truck drivers, police officers who want to reduce the accident rate and severity.

Data

- 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.
- 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.

Important Abbreviations

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

Data Processing

- **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	20 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

Data Processing

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

Data Processing

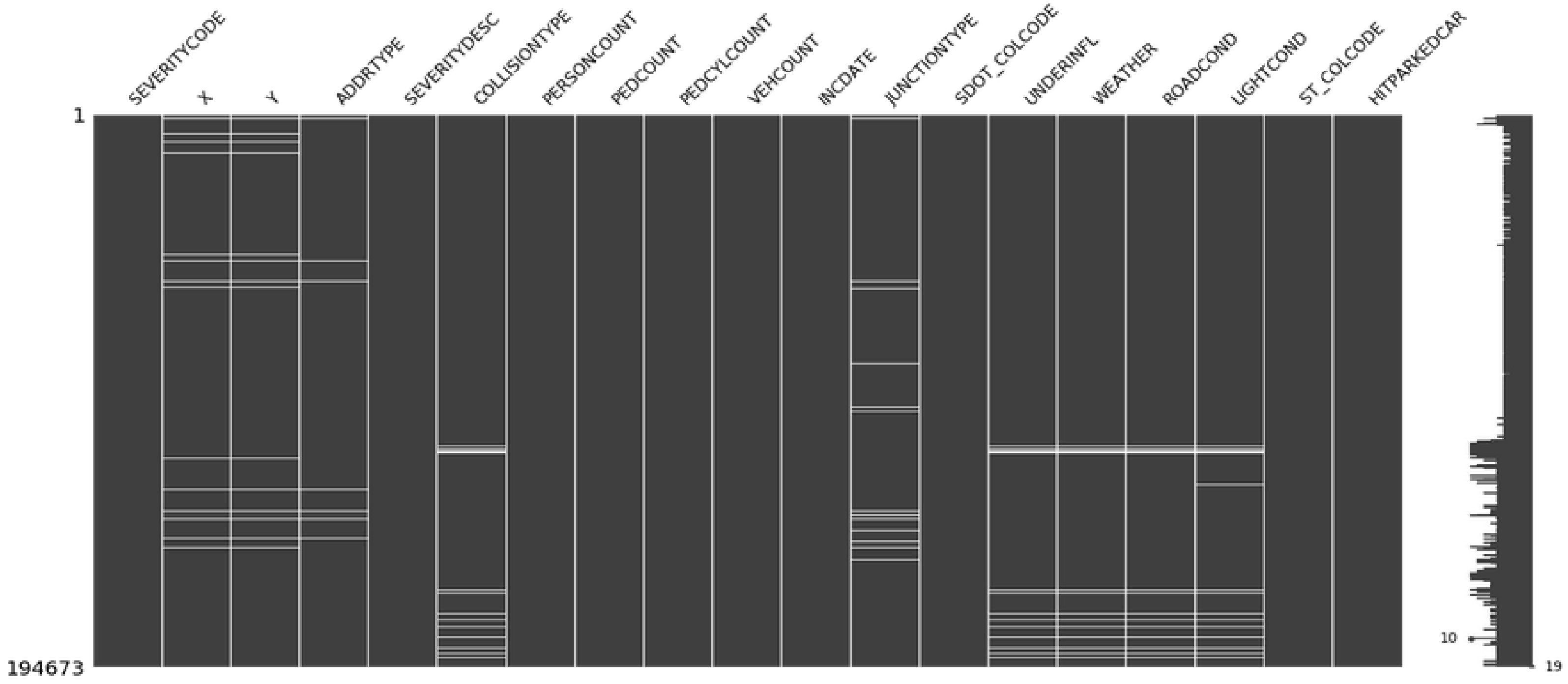
▶ Check Missing Values

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>
```



Data Processing

- **Drop Missing Values**

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]:

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

Data Processing

- **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]
```

Data Processing

- **Correct Data Format**

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

Data Processing

- 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
```

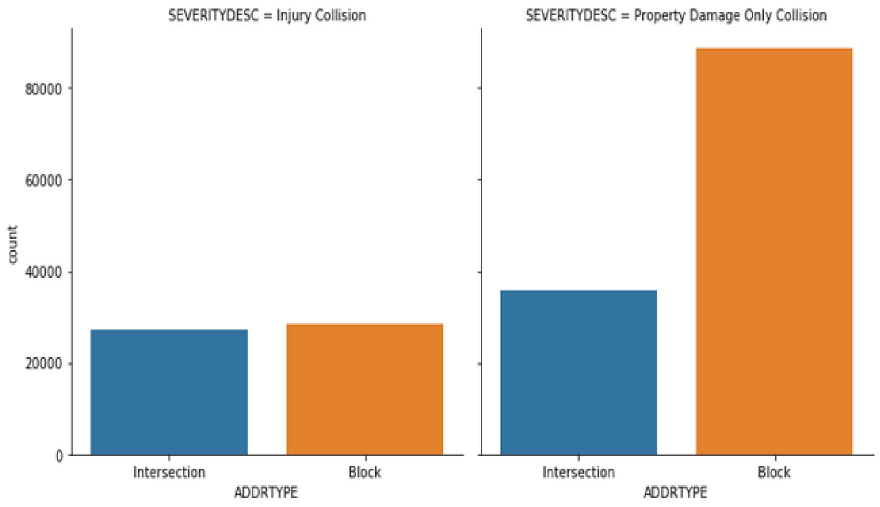
Methodology – Exploratory Data Analysis – Catplots for categorical data

1. **ADDRTYPE vs SEVERITYDESC** – mostly property damage only collision happened in the Block while few of those happened in the intersection. For injury collision the accidents are the same for both block and intersection.

Catplots for Categorical data

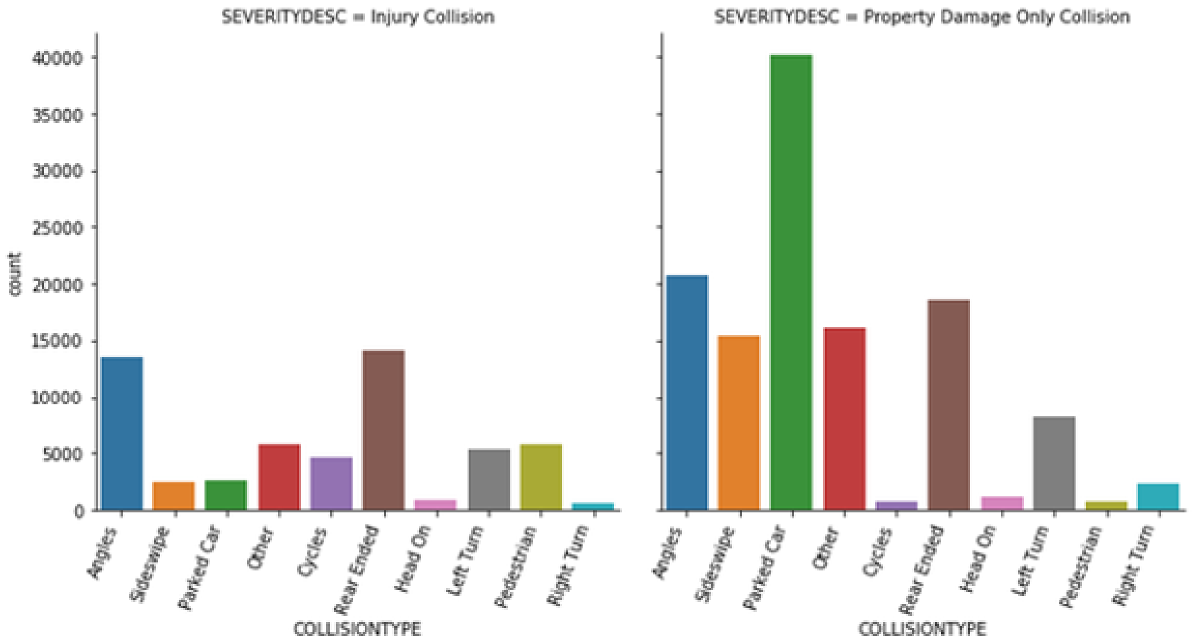
In [27]:
```
import seaborn as sns
```

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

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



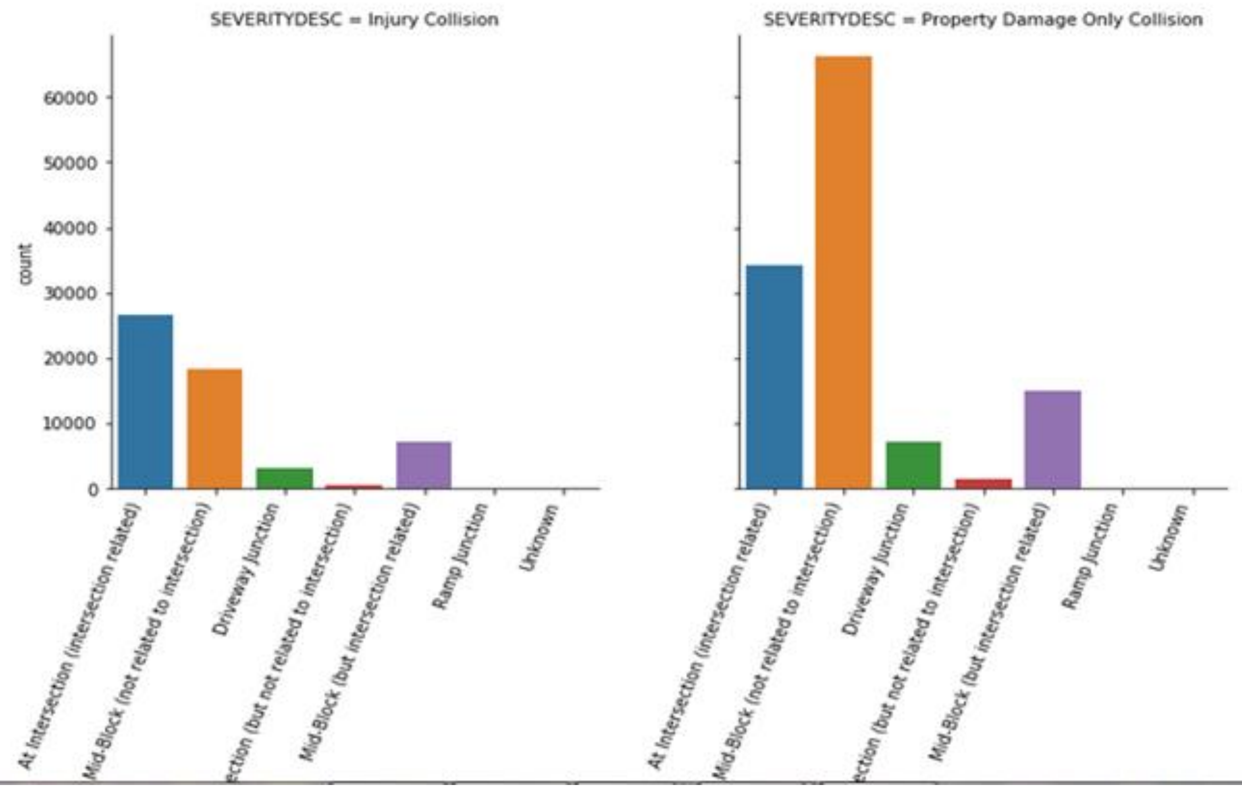
2. COLLISIONTYPE vs SEVERITYDESC – mostly property damage only collisions happened with a parked car. Rear-ended and Angles collisions also occurred frequently.

```
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>
```



3. JUNCTIONTYPE vs SEVERITYDESC – Most property damage only collisions with Mid-Block Junction type. At Intersection Junction type collisions were also frequent.

```
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>
```



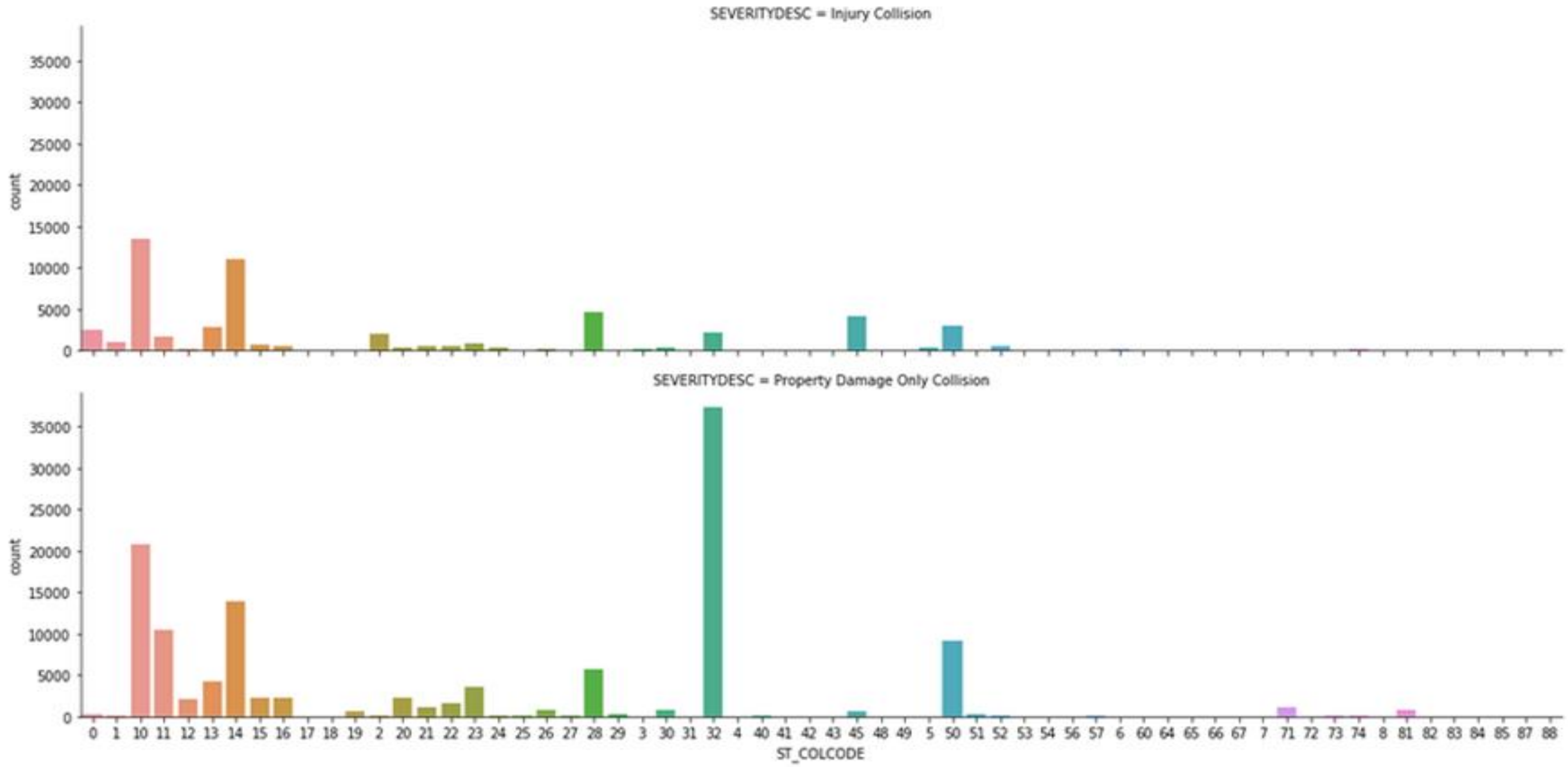
4. ST_COLCODE vs SEVERITYDESC - the most common collision type are:

10-Entering at an angle,

32-One parked--one moving

14-From same direction - both going straight - one stopped - rear-end

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



5. SDOT_COLCODE vs SEVERITYDESC - the most common collisions type are:

11-motor vehicle struck motor vehicle, front end at angle,

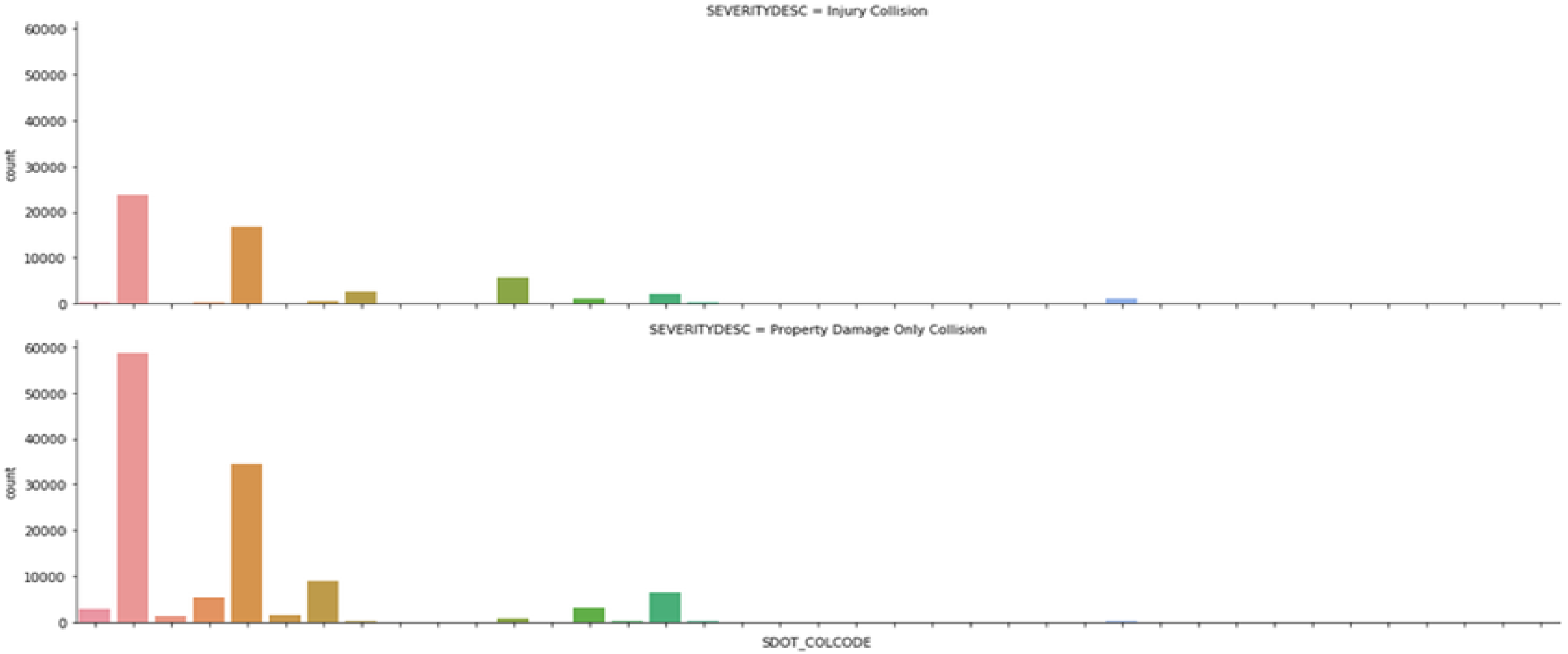
14-motor vehicle struck motor vehicle, rear end,

16-motor vehicle struck motor vehicle, left side sideswipe,

24-motor vehicle struck pedestrian

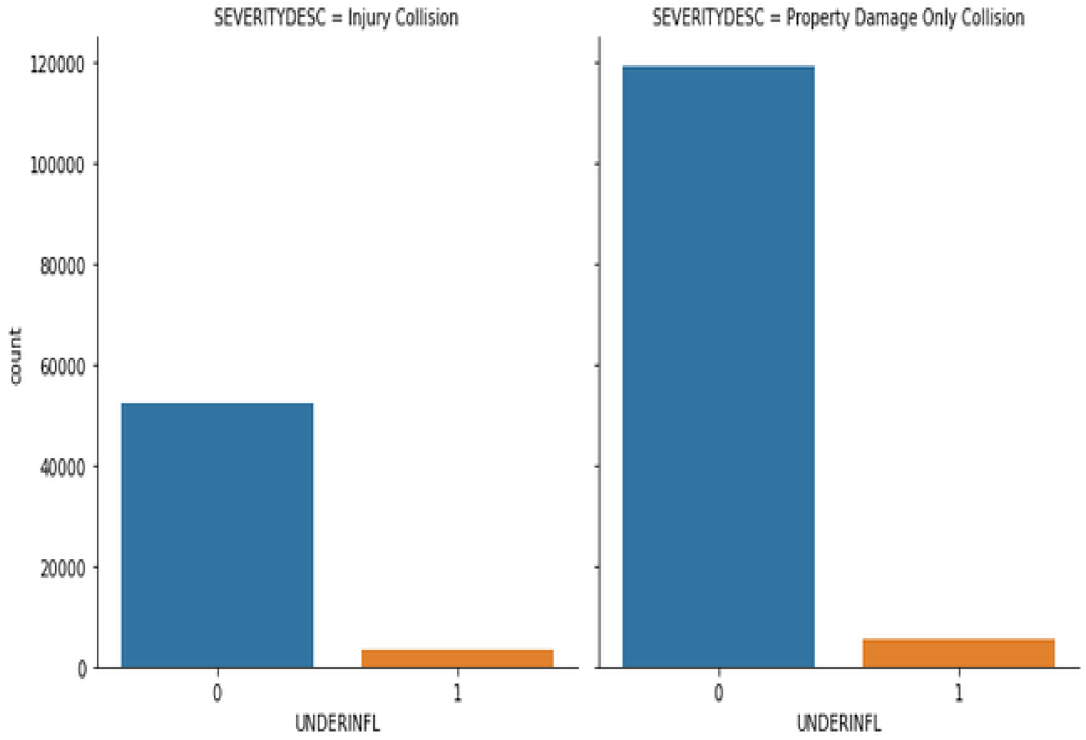
In [31]:
```
# plot catplot of frequency of SDOT_COLCODE

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>



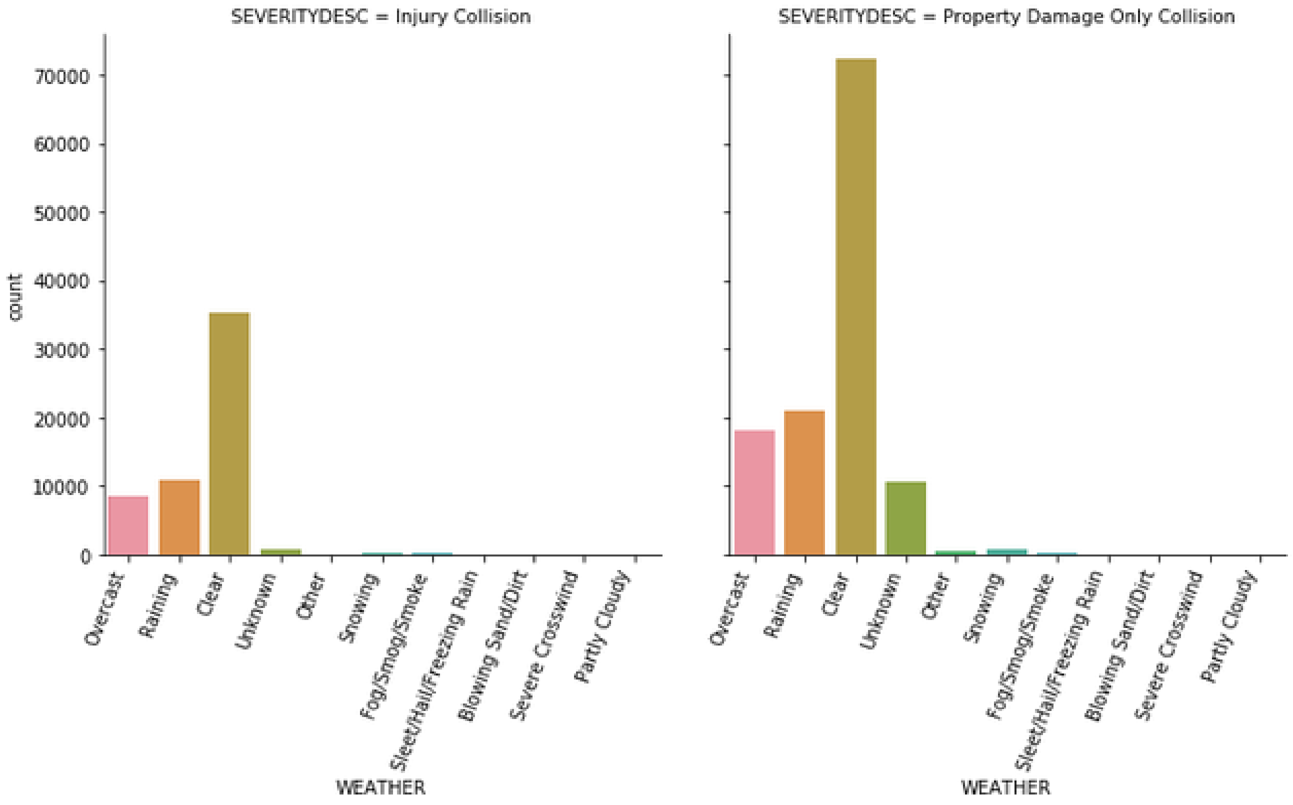
6. UNDERINFL vs SEVERITYDESC – we can see that most people that are involved in car accidents don’t use drug or alcohol while driving

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

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



7. WEATHER vs SEVERITYDESC – we can see that more accidents happened when the weather was clear.

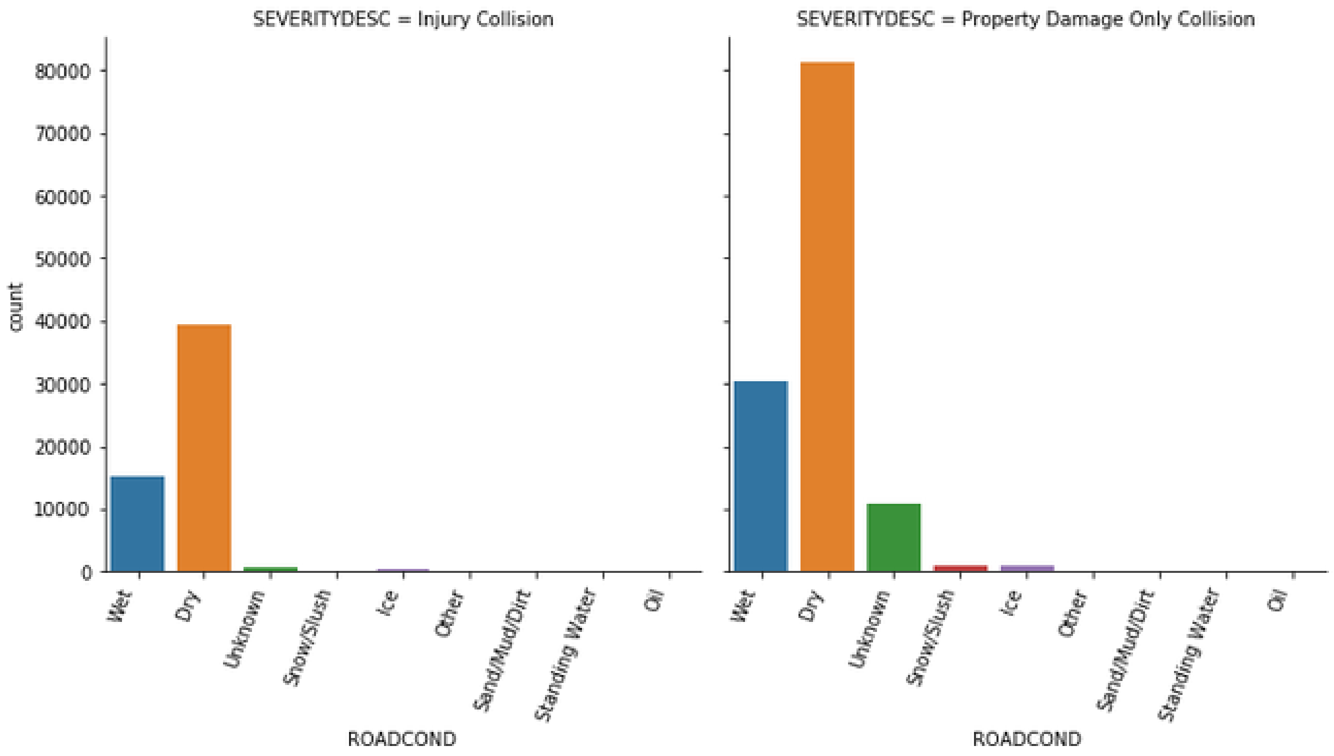
```
In [34]: # plot catplot of frequency of WEATHER

         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>
```



8. ROADCOND vs SEVERITYDESC – we can see that more accidents occurred when the road was dry. A lot accidents also occurred when the road was wet.

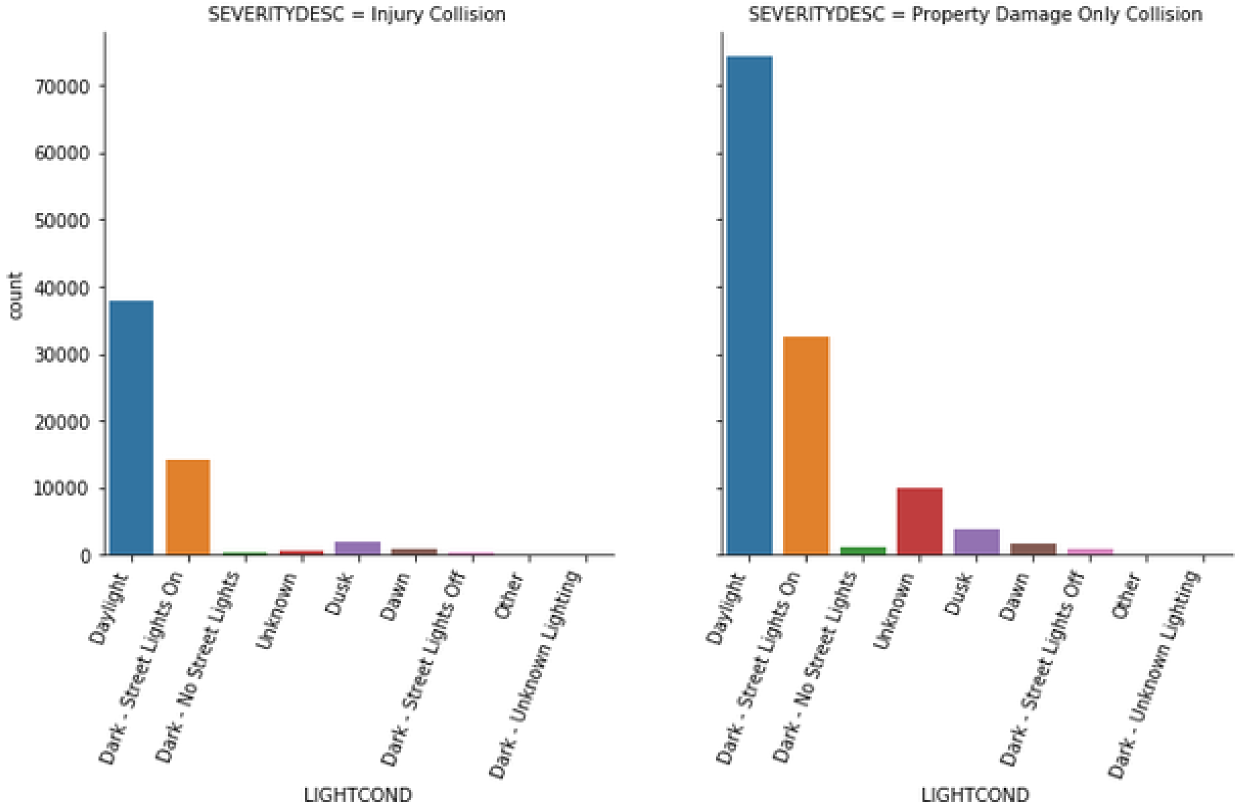
In [35]:
```
# plot catplot of frequency of ROADCOND

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>



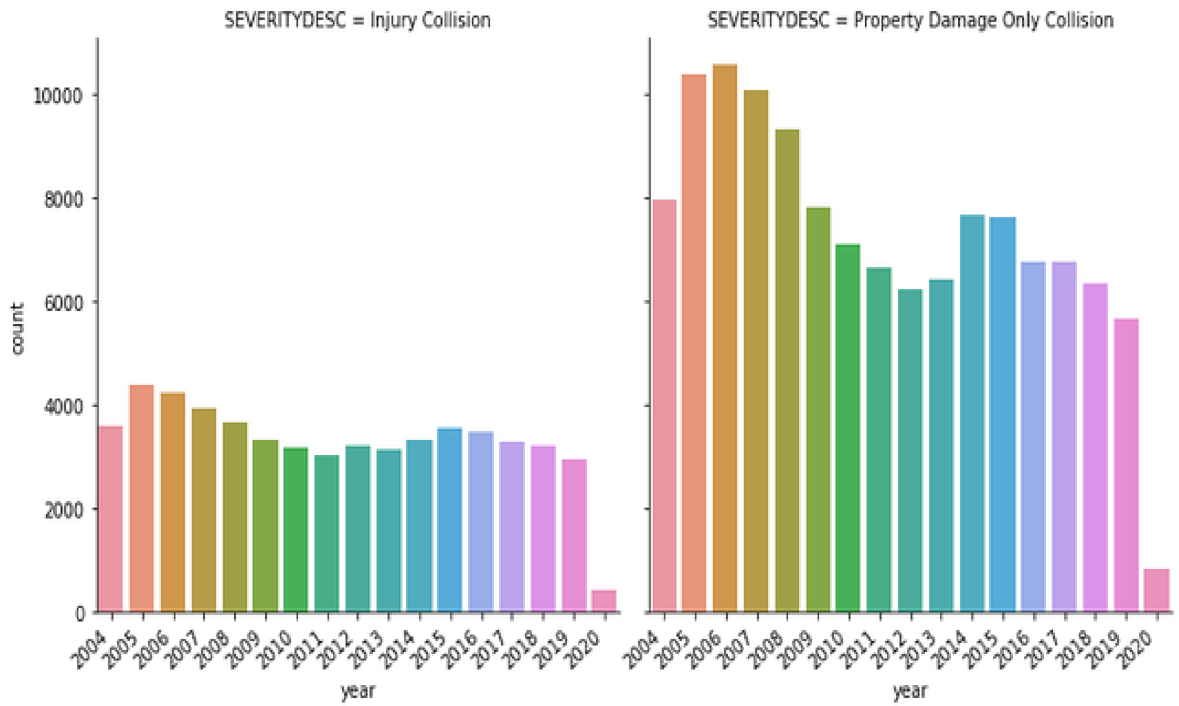
9. LIGHTCOND vs SEVERITYDESC – Most accidents occurred during daylight. A lot of accidents also occurred in the dark when the street lights were on.

```
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>
```



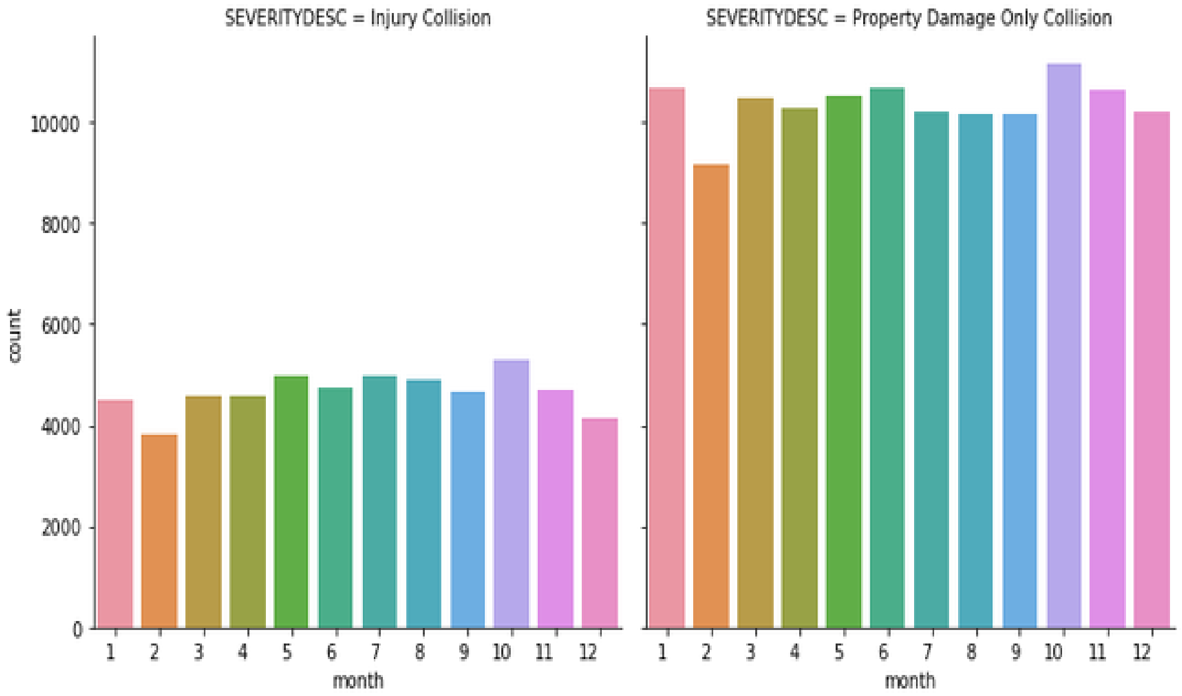
10. year vs SEVERITYDESC – most accidents occurred from 2004-2009 and since then have fallen will 2020.

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>



11. month vs SEVERITYDESC – a lot of property damage only collisions occurred throughout the year. Injury collisions also occurred.

```
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>
```

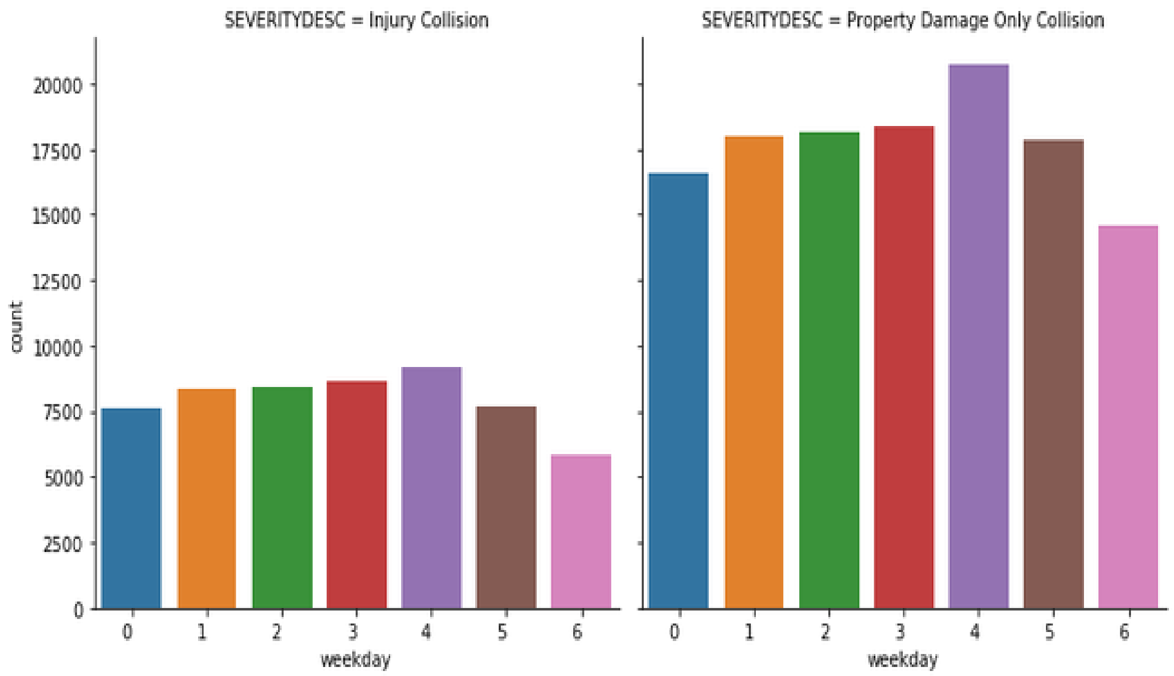


12. weekday vs SEVERITYDESC – most accidents occurred on Thursdays.

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>
```



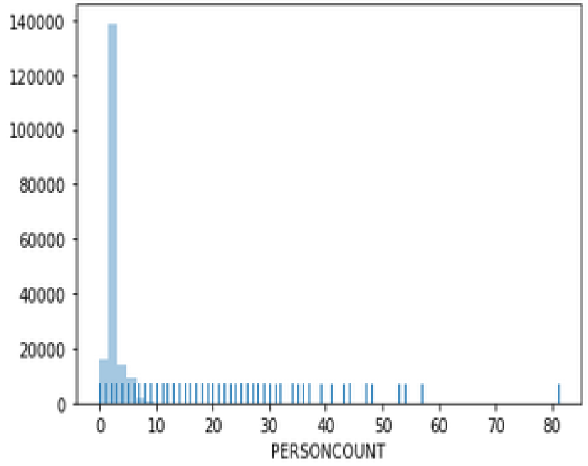
Methodology – Exploratory Data Analysis – Displots for numerical data

- 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>

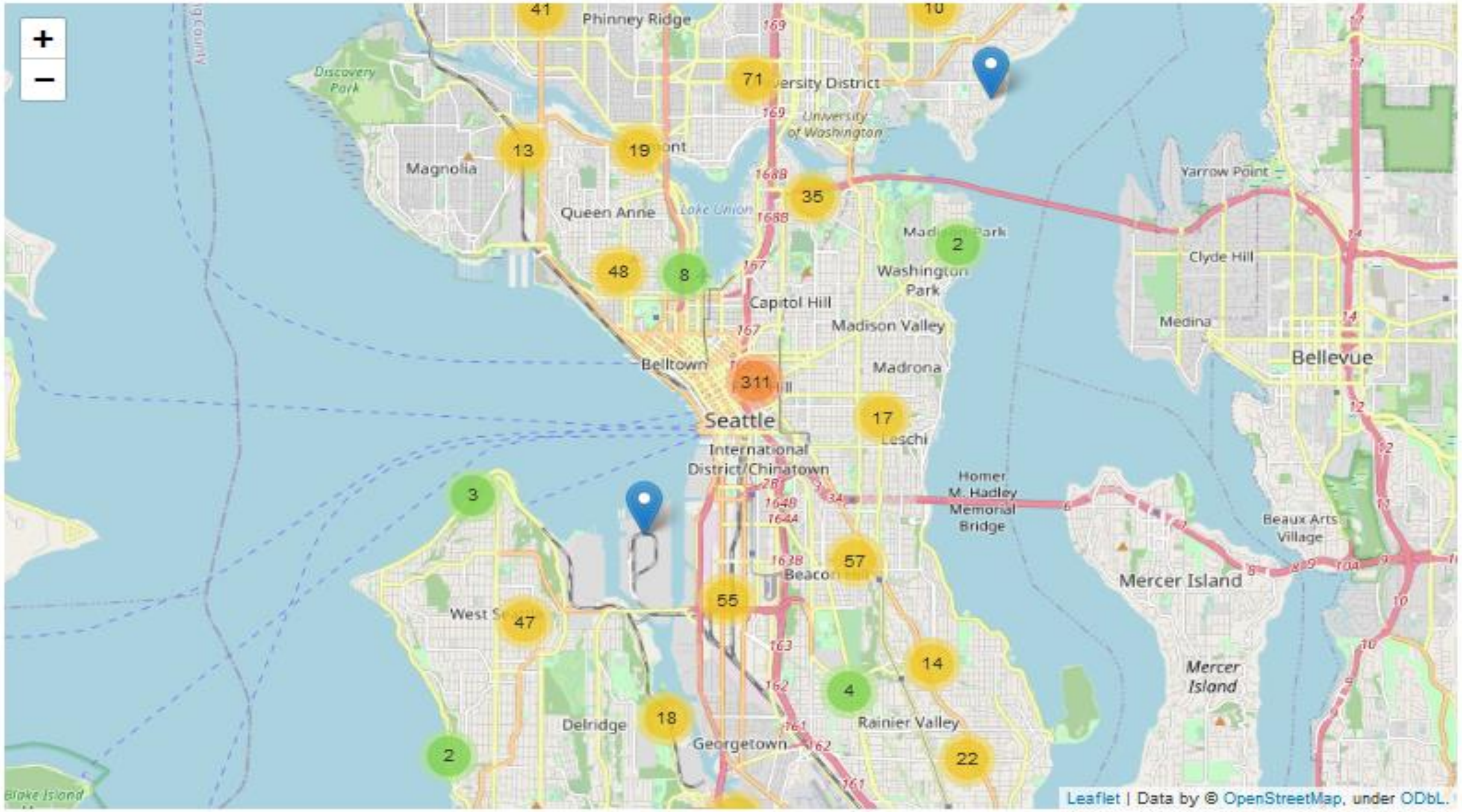


Methodology – Spatial Analysis

most accidents occurred in the downtown of Seattle and state highway.

```
    ).add_to(incidents)

# display map
seattle_map
```

Out[47]:



Methodology – Data Preparation

1. Drop columns with attributes that are not needed anymore.

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)	

2. 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]:

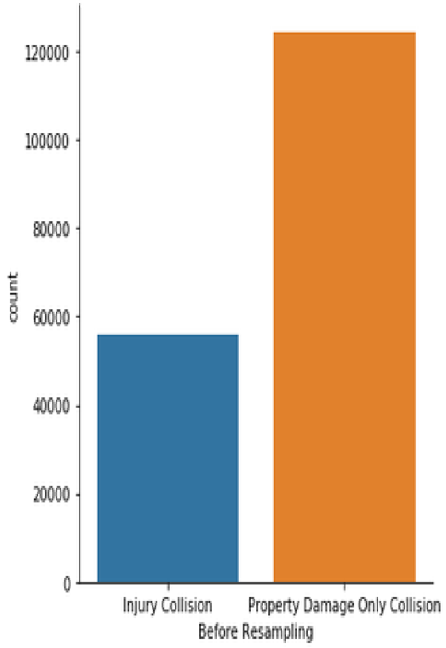
.	LIGHTCOND_Dark - Street Lights Off	LIGHTCOND_Dark - Street Lights On	LIGHTCOND_Dark - Unknown Lighting	LIGHTCOND_Dawn	LIGHTCOND_Daylight	LIGHTCOND_Dusk	LIGHTCOND_Other	LIGHTCOND_Unknown
.	0	0	0	0	1	0	0	0
.	0	1	0	0	0	0	0	0
.	0	0	0	0	1	0	0	0
.	0	0	0	0	1	0	0	0
.	0	0	0	0	1	0	0	0

3. **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

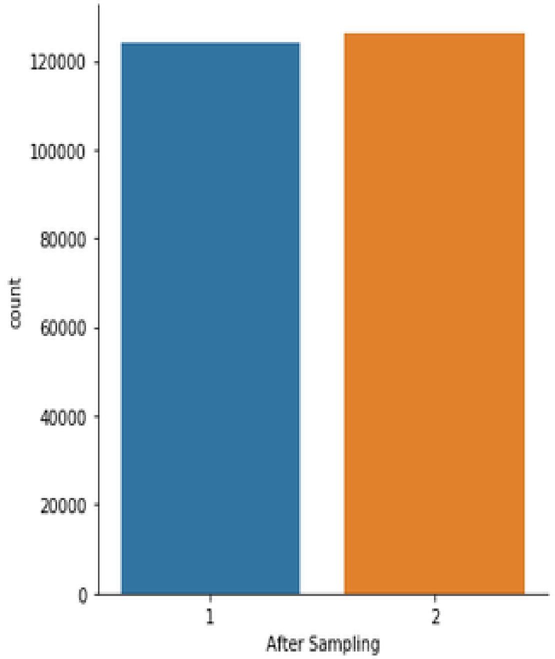
In [52]:
```
ax = sns.catplot(x='SEVERITYDESC', data=df3, kind='count')
ax.set(xlabel='Before Resampling')
```

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>



4. Set x and y variables

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

5. Split data into training and testing sets. Split data into training (70%) and testing (30%)

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,)
```

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.

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.

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, jaccard similarity scores and log loss were calculated.

Results

1. K-Nearest Neighbors model

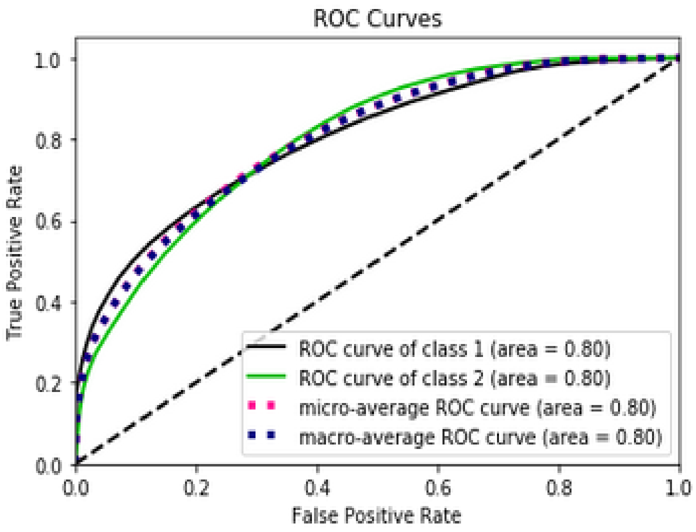
Accuracy score – 0.717

Jaccard Similarity score – 0.717

F1 score – 0.715

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()
```



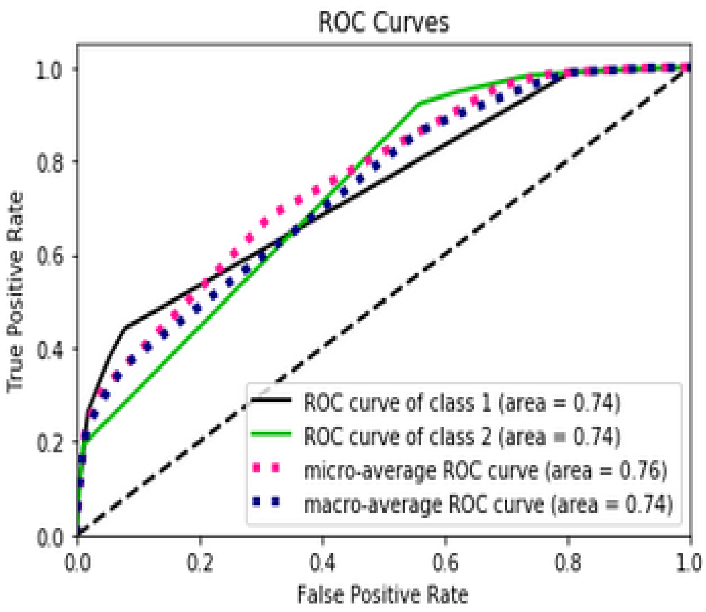
2. Decision Tree model

Accuracy score – 0.683

Jaccard Similarity score – 0.683

F1 score – 0.664

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()
```



3. Logistic Regression Model

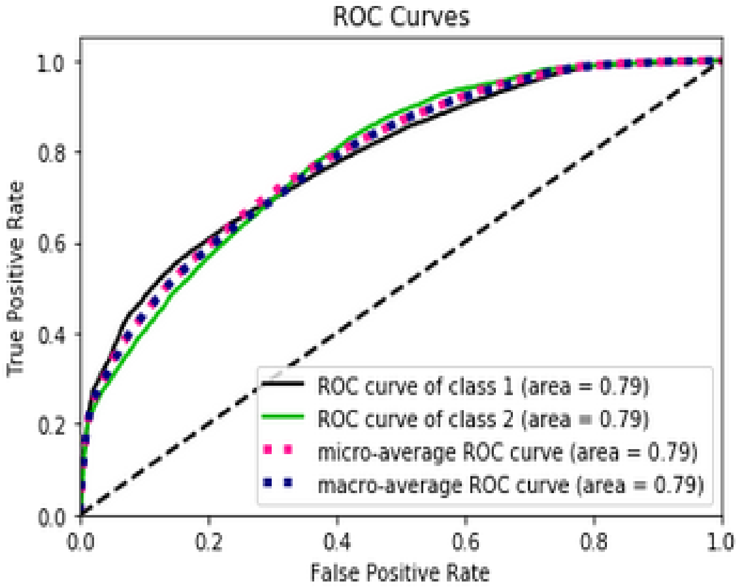
Accuracy score – 0.704

Jaccard Similarity score – 0.704

F1 score – 0.700

Log Loss = 0.5445

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



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%.

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 police can open more signals on Thursday in order to avoid accidents because of more traffic.
- There should be more warning and speed limit signs
- 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.
- Overall, this project can be useful to minimize the number of car accidents by taking strict actions in the future.