IBM Data Science Capstone Project – Car Accident Severity

1. Introduction & Busines Understanding:

Car collision are quite common around the world. They cost countries billions of dollars in healthcare cost and property damage and result on thousands of deaths and injuries. To reduce the number of car collision and its severity, a model will be developed to predict the severity of an accident given attributes like weather conditions, car speed, light conditions, and road conditions and several other attributes.

For the data source of this project, I will be using the data from the City of Seattle's police department showing all the car collisions from 2004 to 2020. This model can be used in an application to warn drivers to be more cautious when driving in bad weather and bad road conditions. This model can also be used by city planners to better design roads and to install warning signs and speed bumps in dangerous road intersections.

2. Data Understanding:

The dataset provided by the City of Seattle's police department had a total of 194,673 collisions from 2004 to 2020. The dataset has 38 attributes describing the details of each car collision including the severity, coordinates, the number of vehicles, date & time, weather & road conditions for each collision. The figures below highlight the initial analysis made on the used dataset.

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

	SEVERITYCODE	X	Y	OBJECTID	INCKEY	COLDETKEY	REPORTNO	STATUS	ADDRTYPE	INTKEY	...	ROADCOND	LIGHTCOND
0	2	-122.323148	47.703140	1	1307	1307	3502005	Matched	Intersection	37475.0	...	Wet	Daylight
1	1	-122.347294	47.647172	2	52200	52200	2607959	Matched	Block	NaN	...	Wet	Dark - Street Lights On
2	1	-122.334540	47.607871	3	26700	26700	1482393	Matched	Block	NaN	...	Dry	Daylight
3	1	-122.334803	47.604803	4	1144	1144	3503937	Matched	Block	NaN	...	Dry	Daylight
4	2	-122.306426	47.545739	5	17700	17700	1807429	Matched	Intersection	34387.0	...	Wet	Daylight

5 rows × 38 columns

In [21]:
```
df.describe()
```
Out[21]:

	SEVERITYCODE	X	Y	OBJECTID	INCKEY	COLDETKEY	INTKEY	SEVERITYCODE.1	PERSONCOUNT
count	194673.000000	189339.000000	189339.000000	194673.000000	194673.000000	194673.000000	65070.000000	194673.000000	194673.000000
mean	1.298901	-122.330518	47.619543	108479.364930	141091.456350	141298.811381	37558.450576	1.298901	2.444427
std	0.457778	0.029976	0.056157	62649.722558	86634.402737	86986.542110	51745.990273	0.457778	1.345929
min	1.000000	-122.419091	47.495573	1.000000	1001.000000	1001.000000	23807.000000	1.000000	0.000000
25%	1.000000	-122.348673	47.575956	54267.000000	70383.000000	70383.000000	28667.000000	1.000000	2.000000
50%	1.000000	-122.330224	47.615369	106912.000000	123363.000000	123363.000000	29973.000000	1.000000	2.000000
75%	2.000000	-122.311937	47.663664	162272.000000	203319.000000	203459.000000	33973.000000	2.000000	3.000000
max	2.000000	-122.238949	47.734142	219547.000000	331454.000000	332954.000000	757580.000000	2.000000	81.000000

```
In [29]:  df.dtypes

Out[29]: 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
```

In our model, I will be using (SEVERITYCODE: a code that corresponds to the severity of the collision) as our dependent variable (Y), which will be the target attribute to be predicted. As for the independent variables (X), I will be using the following attributes:

- ADDTYPE: Collision address type (Alley, Block, Intersection)
- WEATHER: A description of the weather conditions during the time of the collision
- ROADCOND: The condition of the road during the collision.
- VEHCOUNT: The number of vehicles involved in the collision. This is entered by the state.
- PERSONCOUNT: The total number of people involved in the collision
- COLLISIONTYPE: Collision type
- LIGHTCOND: The light conditions during the collision.

3. Methodology:

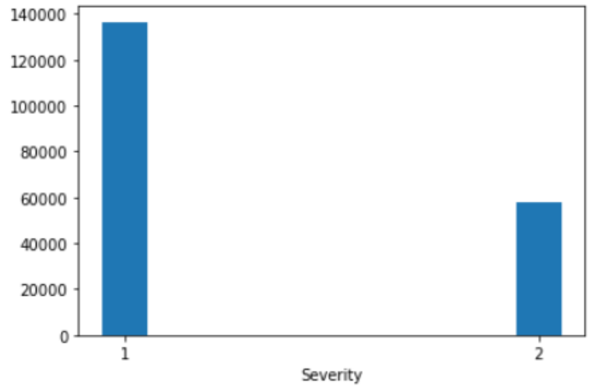
I used Jupyter Notebook and Python code to do the data analysis. To generate the table and graph for the dataset, we imported the following Python libraries (Numpy, Pandas, Matplotlib, and Seaborn). For further analysis of the data I will check if the data is balanced or not for SEVERITYCODE, to have a reliable prediction model.

In [43]:
```
fig, ax = plt.subplots()

im = df['SEVERITYCODE'].hist()

ax.set_xticks([1.05,1.95])
ax.set_xticklabels([1,2])
ax.grid(False)
ax.set_xlabel('Severity')

plt.show()
```



As seen by the above bar chart, the SEVERITYCODE is not balanced, as there are more than double the numbers of Severity 1 (Property Damage) compared to Severity 2 (Injury).

3.1 Exploratory Data Analysis

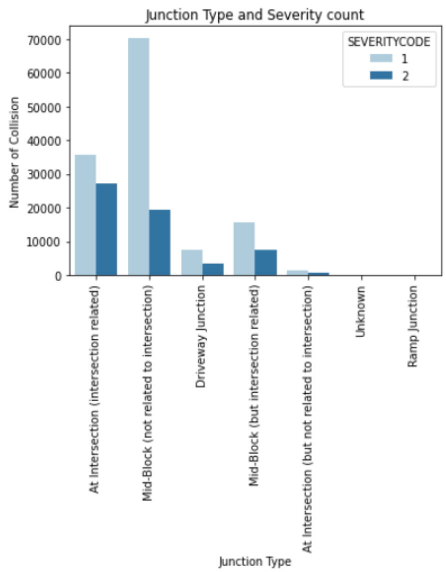
For exploratory data analysis, I will check the distribution of collisions in relation to junction types, to compare the severity count of each junction type.

```
In [50]: import seaborn as sns

         sns.countplot(x="JUNCTIONTYPE", hue="SEVERITYCODE", data=df, palette="Paired")

         plt.title('Junction Type and Severity count')
         plt.xlabel('Junction Type')
         plt.xticks(rotation = 90)
         plt.ylabel('Number of Collision')

Out[50]: Text(0, 0.5, 'Number of Collision')
```



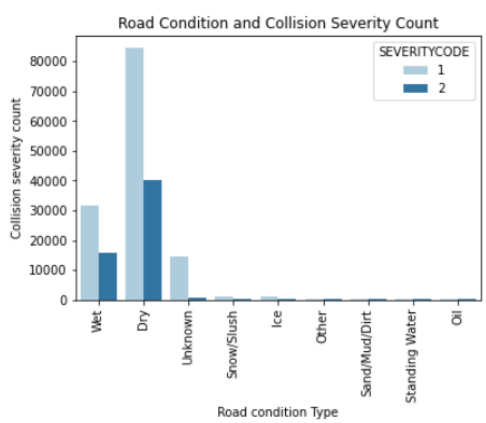
As shown in the above figure, the most property damage collision (SEVERITYCODE 1) happen in Mid-Block junction types. Whereas, the most injury collisions (SEVERITYCODE 2) happen at intersections.

Next, we will check the relationship between road condition and collision severity count.

```
In [53]: sns.countplot(x="ROADCOND", hue="SEVERITYCODE", data=df, palette="Paired")

         plt.title('Road Condition and Collision Severity Count')
         plt.xlabel('Road condition Type')
         plt.xticks(rotation = 90)
         plt.ylabel('Collision severity count')

Out[53]: Text(0, 0.5, 'Collision severity count')
```



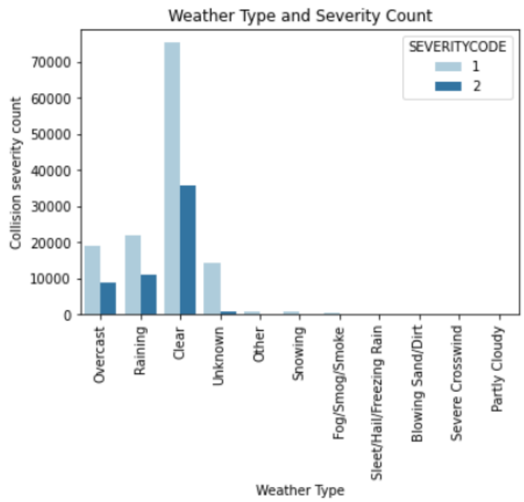
As shown in the above figure, most collisions (SEVERITYCODE 1&2) happen on dry road condition, not on wet road conditions.

Next, we will check the relationship between Weather type and collision severity count.

```
In [54]: sns.countplot(x="WEATHER", hue="SEVERITYCODE", data=df, palette="Paired")

         plt.title('Weather Type and Severity Count')
         plt.xlabel('Weather Type')
         plt.xticks(rotation = 90)
         plt.ylabel('Collision severity count')

Out[54]: Text(0, 0.5, 'Collision severity count')
```



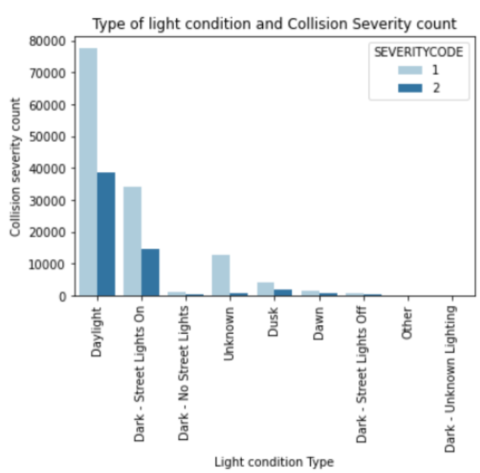
As shown in the above figure, most collisions (SEVERITYCODE 1&2) happen on clear weather, and not when raining or overcast.

Next, we will check the relationship between Type of light condition and collision severity count

```
In [56]: sns.countplot(x="LIGHTCOND", hue="SEVERITYCODE", data=df, palette="Paired")

         plt.title('Type of light condition and Collision Severity count')
         plt.xlabel('Light condition Type')
         plt.xticks(rotation = 90)
         plt.ylabel('Collision severity count')

Out[56]: Text(0, 0.5, 'Collision severity count')
```



As shown in the above figure, most collisions (SEVERITYCODE 1&2) happen in daylight.

3.2 Machine Learning:

In our model, I will be using (SEVERITYCODE) as our dependent variable (Y), which will be the target attribute to be predicted**.** As for the independent variables (X), I will be using the following attributes (WEATHER, JUNCTIONTYPE, ROADCOND)

To process the independent variables in our model I need to convert the categorical values to numeric values.

In [15]:

```
# Convert Categorical Values to Numeric Value for WEATHER
df[pd.get_dummies(df['WEATHER']).columns] = pd.get_dummies(df['WEATHER'])
df.drop(['WEATHER'], axis=1, inplace=True)

# Convert Categorical Values to Numeric Value for JUNCTIONTYPE
df[pd.get_dummies(df['JUNCTIONTYPE']).columns] = pd.get_dummies(df['JUNCTIONTYPE'])
df.drop(['JUNCTIONTYPE'], axis=1, inplace=True)


# Convert Categorical Values to Numeric Value for ROADCOND
df[pd.get_dummies(df['ROADCOND']).columns] = pd.get_dummies(df['ROADCOND'])
df.drop(['ROADCOND'], axis=1, inplace=True)
```

In [21]:

```
df.drop(['LIGHTCOND'],axis=1, inplace=True)
```

In [22]:

```
df.head()
```

Out[22]:

	VEHCOUNT	SEVERITYCODE	Blowing Sand/Dirt	Clear	Fog/Smog/Smoke	Overcast	Partly Cloudy	Raining	Severe Crosswind	Sleet/Hail/Freezing Rain	...	Mid-Block (but intersection related)	Mid-Blc (not relat intersecti
0	2	2	0	0	0	1	0	0	0	0	...	0	
1	2	1	0	0	0	0	0	1	0	0	...	0	
2	3	1	0	0	0	1	0	0	0	0	...	0	
3	3	1	0	1	0	0	0	0	0	0	...	0	
4	2	2	0	0	0	0	0	1	0	0	...	0	

5 rows × 24 columns

To build our model we will be using **Decisions Tree** and **Logistic Regression** algorithm. We will split the data set into 70% training set and 30% testing set

Building The Model

In [17]:
```
import sklearn.model_selection as model_selection
from sklearn.metrics import accuracy_score
from sklearn.tree import DecisionTreeClassifier
from sklearn.linear_model import LogisticRegression
from sklearn.preprocessing import StandardScaler
```

In [23]:
```
# Splitting the Training Set 70% and Testing Set 30%
df_model = df
X = df_model.drop(['SEVERITYCODE'],axis=1)
y = df_model['SEVERITYCODE']
X_train, X_test, y_train, y_test = model_selection.train_test_split(X, y, train_size=0.7,test_size=0.3)

print("done")
```

```
done
```

In [25]:
```
lr = LogisticRegression()
lr.fit(X_train,y_train)
yhat_lr=lr.predict(X_test)

print("done")
```

```
done
```

In [26]:
```
dt = DecisionTreeClassifier(criterion = "entropy", max_depth=10)
dt.fit(X_train, y_train)
yhat_dt = dt.predict(X_test)

print("done")
```

```
done
```

4. Results:

In this step we will evaluate each ML model used

Model Evaluation

In [27]:
```
from sklearn.metrics import f1_score, precision_score, recall_score
f1lr = f1_score(y_test, yhat_lr)
f1dt = f1_score(y_test, yhat_dt)
print('F1 Score for Log Reg=', f1lr)
print('F1 Score for Descision Tree', f1dt)

acclr = accuracy_score(y_test, yhat_lr)
accdt = accuracy_score(y_test, yhat_dt)
print('Accuracy Score for Log Reg=', acclr)
print('Accuracy for Descision Tree', accdt)

plr = precision_score(y_test, yhat_lr)
pdt = precision_score(y_test, yhat_dt)
print('Precision Score for Log Reg=', plr)
print('Precision for Descision Tree', pdt)



rlr = recall_score(y_test, yhat_lr)
rdt = recall_score(y_test, yhat_dt)
print('Recall Score for Log Reg=', rlr)
print('Recall for Descision Tree', rdt)
```

```
F1 Score for Log Reg= 0.8041569051480747
F1 Score for Descision Tree 0.8114587412057446
Accuracy Score for Log Reg= 0.6755567939812511
Accuracy for Descision Tree 0.7029038871086519
Precision Score for Log Reg= 0.6745067214865873
Precision for Descision Tree 0.7051566307378219
Recall Score for Log Reg= 0.9955083586174074
Recall for Descision Tree 0.9555000297459694
```

ML Model	F1 Score	Accuracy Score	Precision Score	Recall Score
Logistic Regression	0.8041	0.6755	0.6745	0.9955
Decision Tree	0.8114	0.7029	0.7051	0.9555

As we can see from the above table, the best ML model to choose from these two is Decision Tree.

5. Discussion & Conclusion:

Before doing the analysis on the data, I had a misconception that weather & road condition and light condition had a big correlation with the severity of the car collision. However, as we can see from the data there was no correlation. We can see from the data that the majority of the car collisions happen on dry roads and in clear weather and in day light.

From the data we can see that the majority of accidents happen in JUNCTIONTYPE Mid-Block, that is not related to intersections, therefore city planner can further analyze Mid-Block locations and either provide speed bumps or reduce the speed limits on those streets.