Data Processing and Understanding

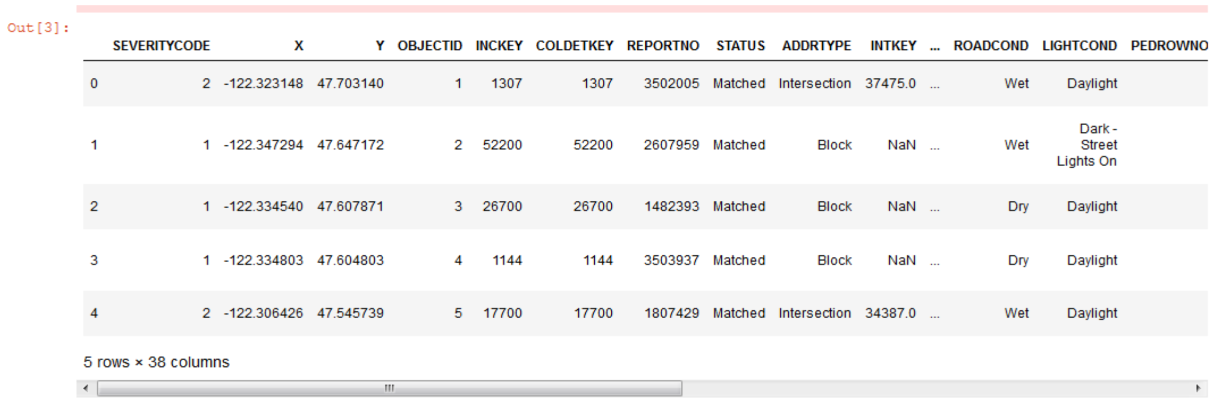
Data

This is an extensive data set from the Seattle Police Department with almost 35 parameters like collision type, Person Count, Vehicle count, weather , Speed etc. To accurately build a model to prevent future accidents and/or reduce their severity, I have decided to use few parameters and drop the remaining parameters depend on the data

Methodology

I used Jupyter Notebook to conduct that analysis and imported all the necessary Python libraries like Pandas, Numpy, Matplotlib and sklearn.

I started by downloading the csv file as given in week 1 details and saved as accident_data.csv.



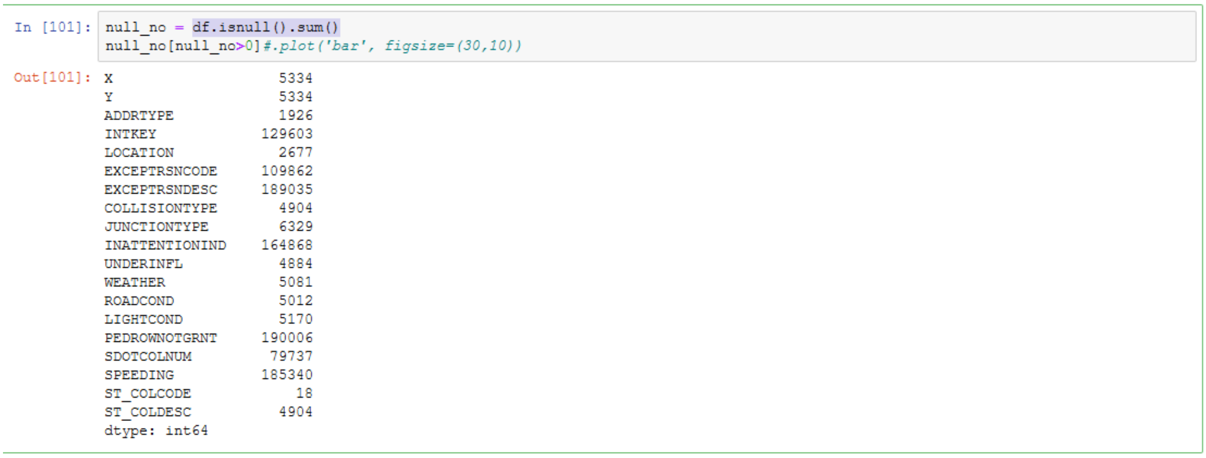
Out[3]:

	SEVERITYCODE	X	Y	OBJECTID	INCKEY	COLDETKEY	REPORTNO	STATUS	ADDRTYPE	INTKEY	...	ROADCOND	LIGHTCOND	PEDROWNO
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

The database considered around 35 parameters and the output label was severity code based on these parameters

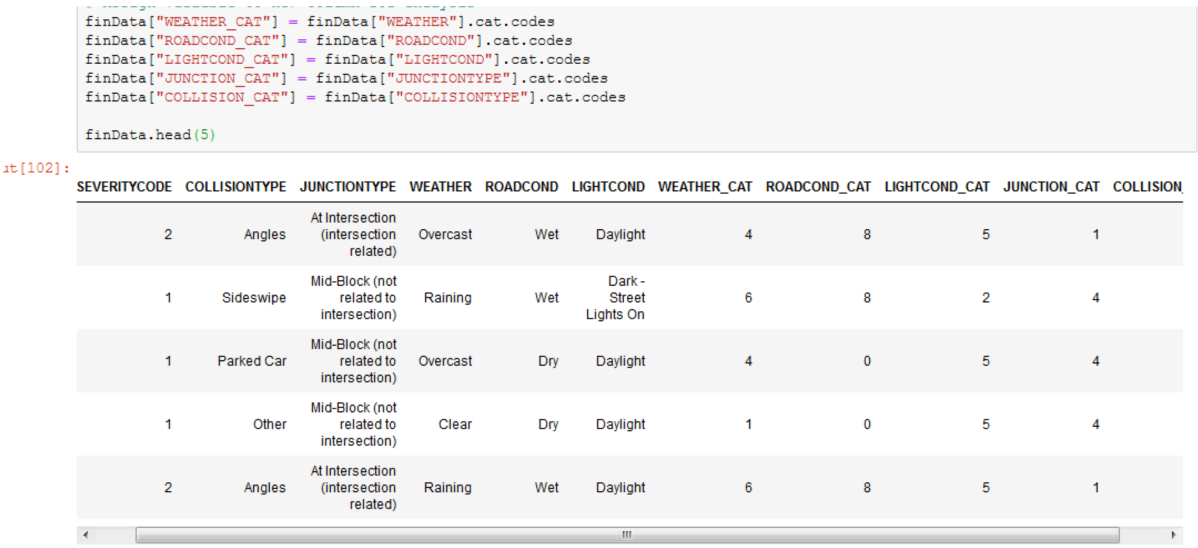
To prepare the data, so that I can drop columns we do not need from the dataset, i.e., columns that do not have values or where the values are unknown I used the commands df.info( ) , df.describe( )and df.isnull( ).sum( ) .



```
In [101]: null_no = df.isnull().sum()
          null_no[null_no>0]#.plot('bar', figsize=(30,10))

Out[101]: X                   5334
          Y                   5334
          ADDRTYPE            1926
          INTKEY            129603
          LOCATION            2677
          EXCEPTRSNCODE     109862
          EXCEPTRSNDESC     189035
          COLLISIONTYPE       4904
          JUNCTIONTYPE        6329
          INATTENTIONIND    164868
          UNDERINFL           4884
          WEATHER             5081
          ROADCOND            5012
          LIGHTCOND           5170
          PEDROWNOTGRNT     190006
          SDOTCOLNUM         79737
          SPEEDING          185340
          ST_COLCODE            18
          ST_COLDESC          4904
          dtype: int64
```

By looking at the table, I decided to incorporate 5 parameters weather, road condition, light condition , Junction type and collision category. Even though speeding is an important parameter, we have to drop speeding entirely because it is missing over 180,000 values and this can hamper the results.

From the main dataframe df, I dropped the unwanted parameters and kept only the necessary parameters and changed their data type to int64

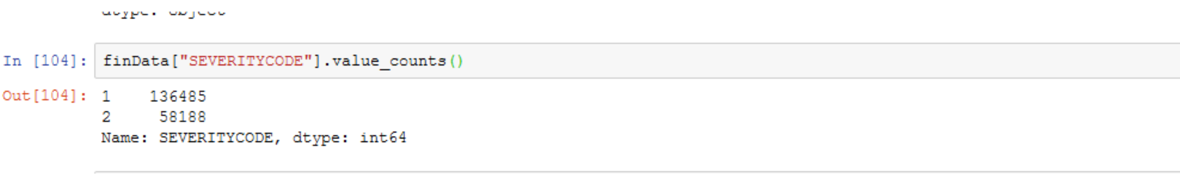


```
finData["WEATHER_CAT"] = finData["WEATHER"].cat.codes
finData["ROADCOND_CAT"] = finData["ROADCOND"].cat.codes
finData["LIGHTCOND_CAT"] = finData["LIGHTCOND"].cat.codes
finData["JUNCTION_CAT"] = finData["JUNCTIONTYPE"].cat.codes
finData["COLLISION_CAT"] = finData["COLLISIONTYPE"].cat.codes

finData.head(5)
```

ut[102]:

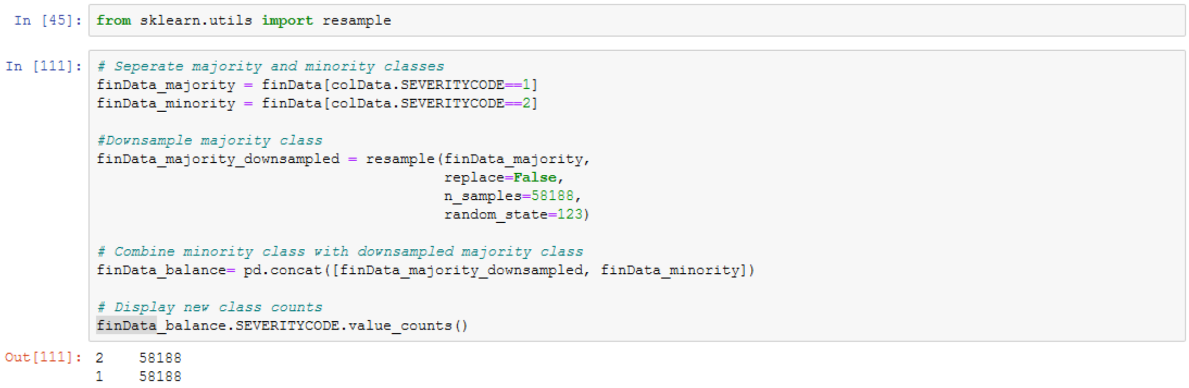
SEVERITYCODE	COLLISIONTYPE	JUNCTIONTYPE	WEATHER	ROADCOND	LIGHTCOND	WEATHER_CAT	ROADCOND_CAT	LIGHTCOND_CAT	JUNCTION_CAT	COLLISION
2	Angles	At Intersection (intersection related)	Overcast	Wet	Daylight	4	8	5	1	
1	Sideswipe	Mid-Block (not related to intersection)	Raining	Wet	Dark - Street Lights On	6	8	2	4	
1	Parked Car	Mid-Block (not related to intersection)	Overcast	Dry	Daylight	4	0	5	4	
1	Other	Mid-Block (not related to intersection)	Clear	Dry	Daylight	1	0	5	4	
2	Angles	At Intersection (intersection related)	Raining	Wet	Daylight	6	8	5	1	

When we analysed the severity code we found out that data is unbalanced as Severity code 1 was approximately three times larger than Severity code 2.



```
          dtype: object

In [104]: finData["SEVERITYCODE"].value_counts()

Out[104]: 1    136485
          2     58188
          Name: SEVERITYCODE, dtype: int64
```

So we have done down sampling for severity code 1 class with sklearn's resample tool. We down sampled to match the severity code 2 exactly with 58188 values each.

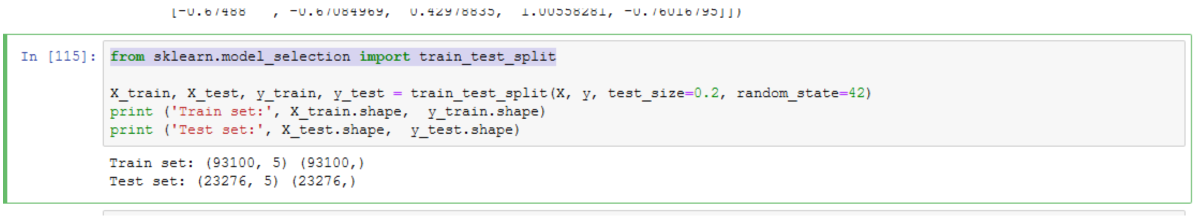


```
In [45]:  from sklearn.utils import resample

In [111]: # Seperate majority and minority classes
          finData_majority = finData[colData.SEVERITYCODE==1]
          finData_minority = finData[colData.SEVERITYCODE==2]

          #Downsample majority class
          finData_majority_downsampled = resample(finData_majority,
                                                  replace=False,
                                                  n_samples=58188,
                                                  random_state=123)

          # Combine minority class with downsampled majority class
          finData_balance= pd.concat([finData_majority_downsampled, finData_minority])

          # Display new class counts
          finData_balance.SEVERITYCODE.value_counts()

Out[111]: 2    58188
          1    58188
```

With this I have completed data cleaning and balancing work.

Now I have to split the data into training data and testing data with a ratio of 80:20 which I have completed using the command

from sklearn.model_selection import train_test_split



```
          [-0.67488   , -0.67084969,  0.42978835,  1.00558281, -0.76016795]])

In [115]: from sklearn.model_selection import train_test_split

          X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)
          print ('Train set:', X_train.shape,  y_train.shape)
          print ('Test set:', X_test.shape,  y_test.shape)

          Train set: (93100, 5) (93100,)
          Test set: (23276, 5) (23276,)
```

So the train data set consists of 93100 samples with 5 parameters and 93100 output labels and the test data consists of 23276 samples with 5 parameters and 23276 output labels.

Since we now we are ready with all necessary data , our next step is to build the models and analyse the performance of the system.