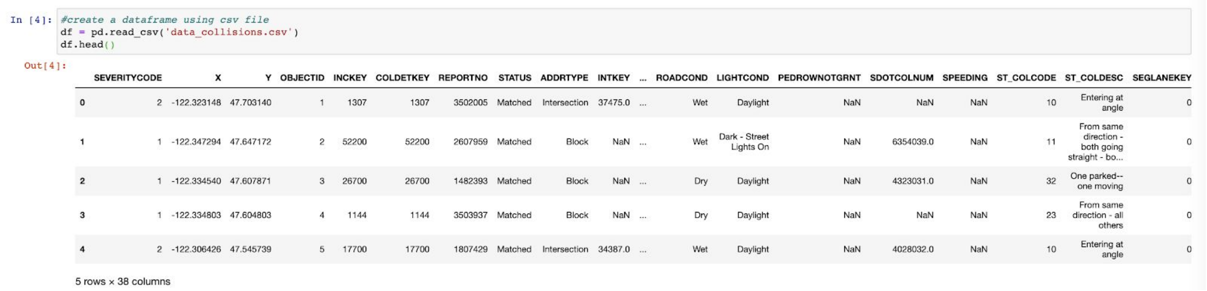
Prediction of Road Accident Severity in Rainy Weather

Data Understanding

Data used in this project is provided by Coursera-IBM Data Science course. This is raw data which requires analysis and transformation to apply machine learning models.
Initial few rows of raw data:

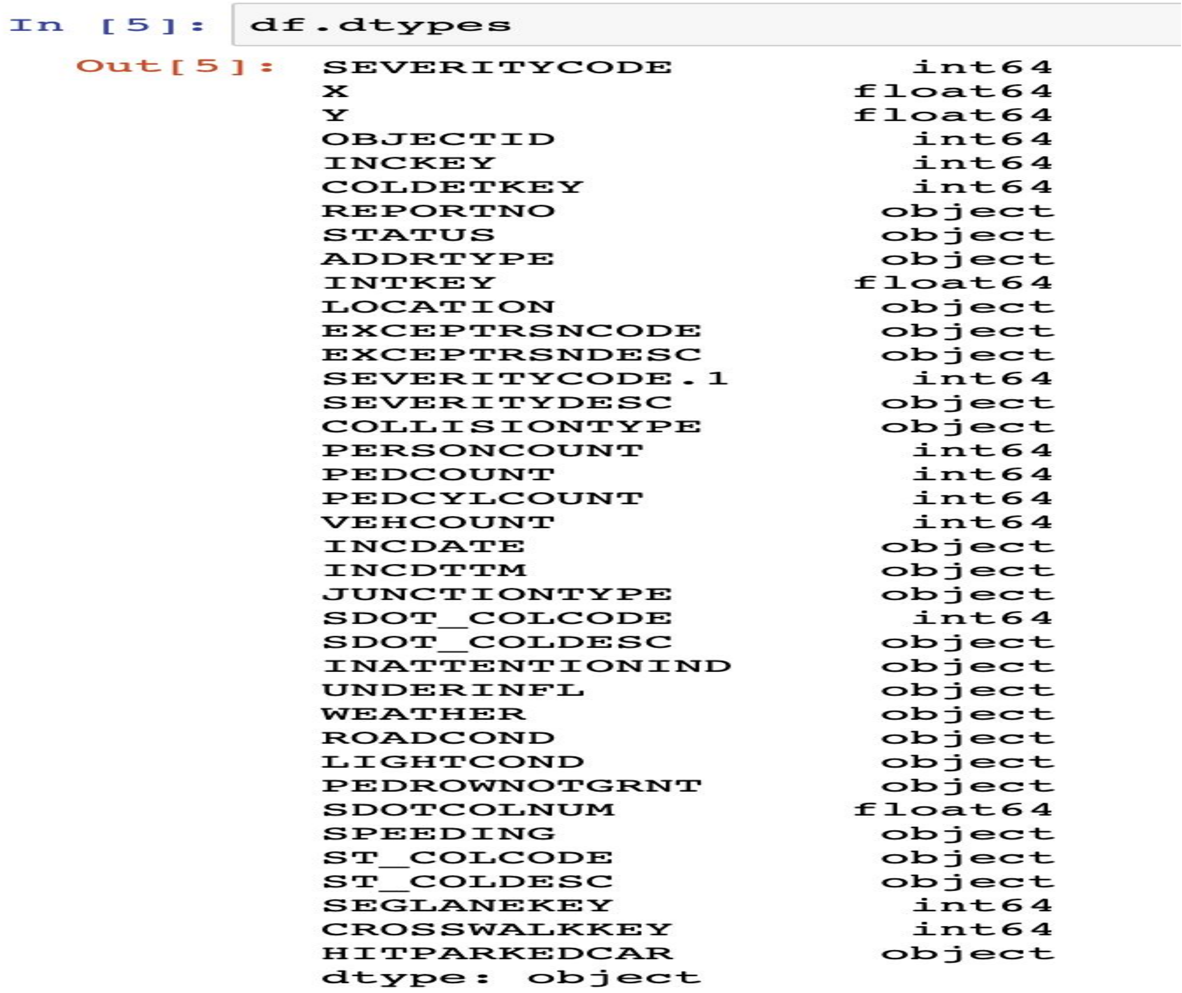


```
In [4]: #create a dataframe using csv file
        df = pd.read_csv('data_collisions.csv')
        df.head()
Out[4]:
```

	SEVERITYCODE	X	Y	OBJECTID	INCKEY	COLDETKEY	REPORTNO	STATUS	ADDRTYPE	INTKEY	...	ROADCOND	LIGHTCOND	PEDROWNOTGRNT	SDOTCOLNUM	SPEEDING	ST_COLCODE	ST_COLDESC	SEGLANEKEY
0	2	-122.323148	47.703140	1	1307	1307	3502005	Matched	Intersection	37475.0	...	Wet	Daylight	NaN	NaN	NaN	10	Entering at angle	0
1	1	-122.347294	47.647172	2	52200	52200	2607959	Matched	Block	NaN	...	Wet	Dark - Street Lights On	NaN	6354039.0	NaN	11	From same direction - both going straight - bo...	0
2	1	-122.334540	47.607871	3	26700	26700	1482393	Matched	Block	NaN	...	Dry	Daylight	NaN	4323031.0	NaN	32	One parked--one moving	0
3	1	-122.334803	47.604803	4	1144	1144	3503937	Matched	Block	NaN	...	Dry	Daylight	NaN	NaN	NaN	23	From same direction - all others	0
4	2	-122.306426	47.545739	5	17700	17700	1807429	Matched	Intersection	34387.0	...	Wet	Daylight	NaN	4028032.0	NaN	10	Entering at angle	0

5 rows × 38 columns

Data types of each column is obtained using **dtypes** of dataframe.



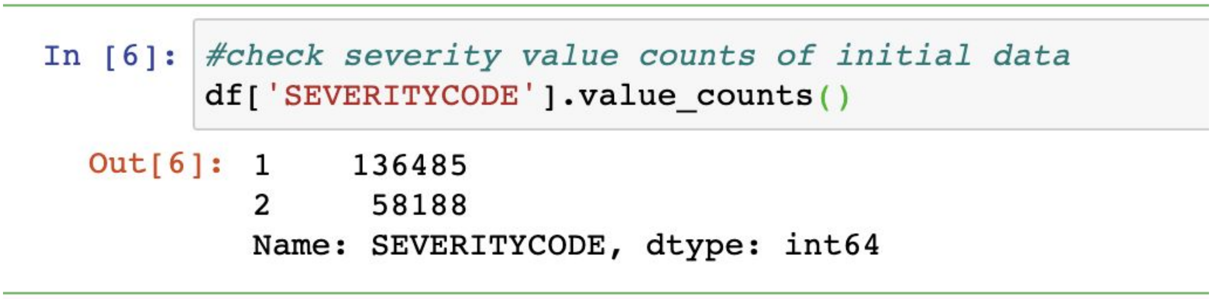
```
In [5]: df.dtypes

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
```

By analysing the raw data, the following conclusion can be done.

1. Severity code is the target parameter or predictor variable which as it shows the severity of the accidents.
2. Data clean up is required as few columns are not required for analysis. Require columns for analysis are as follows.
3. Since the goal is to get predictions for rainy weather, only rows corresponding to ‘Rainy’ Weather can be used.
4. ROADCOND and LIGHTCOND are different categories that can be derived from the Weather column.
5. Convert raw unbalanced data to balanced dataset.

Severity code

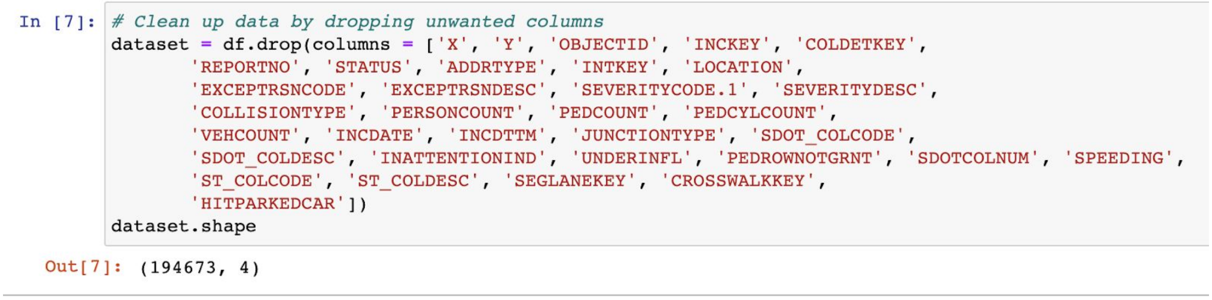
For Rainy weather, the SEVERITY CODE is either 1 or 2, where 1 indicates it is Safe to travel and 2 indicates damage to life or property. This can be used as a Target variable to derive a solution.



```
In [6]: #check severity value counts of initial data
        df['SEVERITYCODE'].value_counts()

Out[6]: 1     136485
        2      58188
        Name: SEVERITYCODE, dtype: int64
```

Data Clean Up

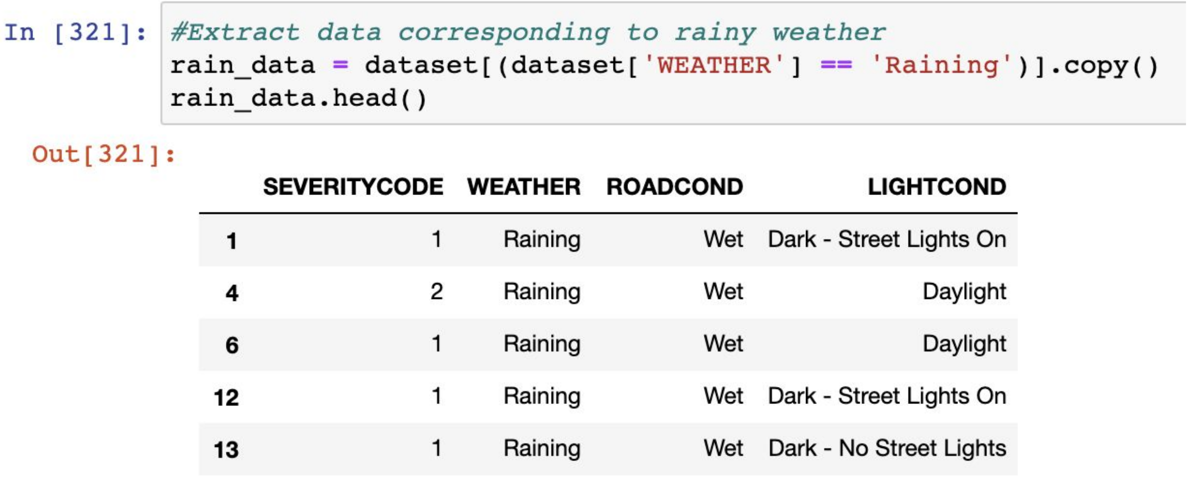
Post extracting csv data to the data frame, a clean up is required to remove unwanted data. Columns excluding SEVERITY CODE, WEATHER, ROADCOND, LIGHTCOND can be removed. This creates a clean data set with only required columns.



```
In [7]: # Clean up data by dropping unwanted columns
        dataset = df.drop(columns = ['X', 'Y', 'OBJECTID', 'INCKEY', 'COLDETKEY',
               'REPORTNO', 'STATUS', 'ADDRTYPE', 'INTKEY', 'LOCATION',
               'EXCEPTRSNCODE', 'EXCEPTRSNDESC', 'SEVERITYCODE.1', 'SEVERITYDESC',
               'COLLISIONTYPE', 'PERSONCOUNT', 'PEDCOUNT', 'PEDCYLCOUNT',
               'VEHCOUNT', 'INCDATE', 'INCDTTM', 'JUNCTIONTYPE', 'SDOT_COLCODE',
               'SDOT_COLDESC', 'INATTENTIONIND', 'UNDERINFL', 'PEDROWNOTGRNT', 'SDOTCOLNUM', 'SPEEDING',
               'ST_COLCODE', 'ST_COLDESC', 'SEGLANEKEY', 'CROSSWALKKEY',
               'HITPARKEDCAR'])
        dataset.shape

Out[7]: (194673, 4)
```

Extract Required Data

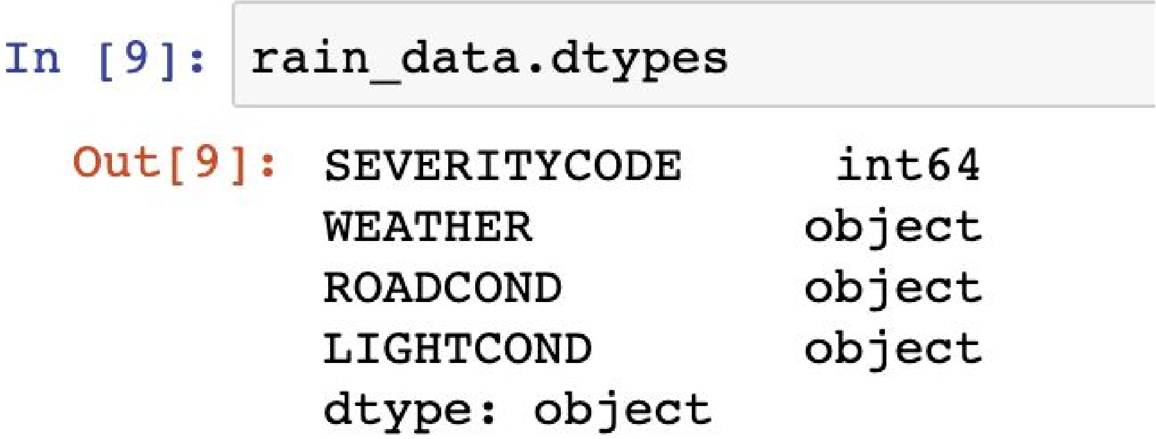
Since analysis is based on Rainy weather, rows including other weather conditions can be removed. This creates a clean data set with only required rows.



```
In [321]: #Extract data corresponding to rainy weather
          rain_data = dataset[(dataset['WEATHER'] == 'Raining')].copy()
          rain_data.head()
```

Out[321]:

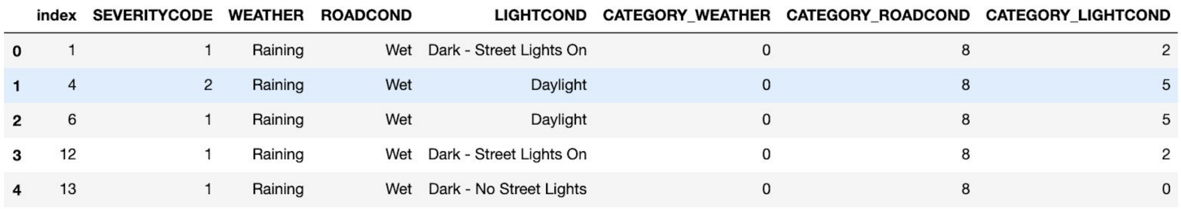
	SEVERITYCODE	WEATHER	ROADCOND	LIGHTCOND
1	1	Raining	Wet	Dark - Street Lights On
4	2	Raining	Wet	Daylight
6	1	Raining	Wet	Daylight
12	1	Raining	Wet	Dark - Street Lights On
13	1	Raining	Wet	Dark - No Street Lights



```
In [9]: rain_data.dtypes

Out[9]: SEVERITYCODE     int64
        WEATHER         object
        ROADCOND        object
        LIGHTCOND       object
        dtype: object
```

Categorise data

ROADCOND and LIGHTCOND are two columns which impact target variables along with Weather data. One of the major reasons why we convert categorical variables into factors i.e number because to make Analysis easy and effective.

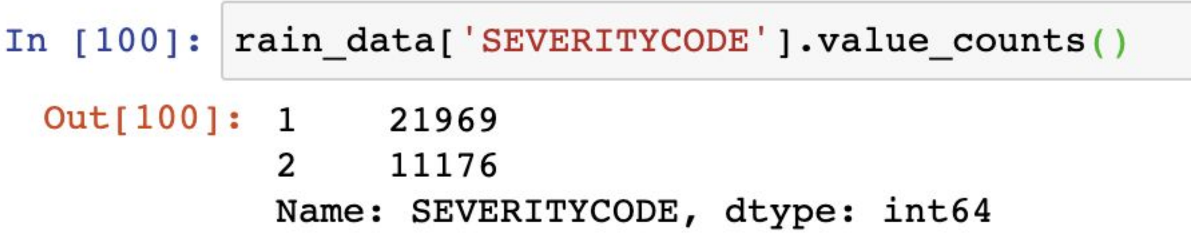


	index	SEVERITYCODE	WEATHER	ROADCOND	LIGHTCOND	CATEGORY_WEATHER	CATEGORY_ROADCOND	CATEGORY_LIGHTCOND
0	1	1	Raining	Wet	Dark - Street Lights On	0	8	2
1	4	2	Raining	Wet	Daylight	0	8	5
2	6	1	Raining	Wet	Daylight	0	8	5
3	12	1	Raining	Wet	Dark - Street Lights On	0	8	2
4	13	1	Raining	Wet	Dark - No Street Lights	0	8	0

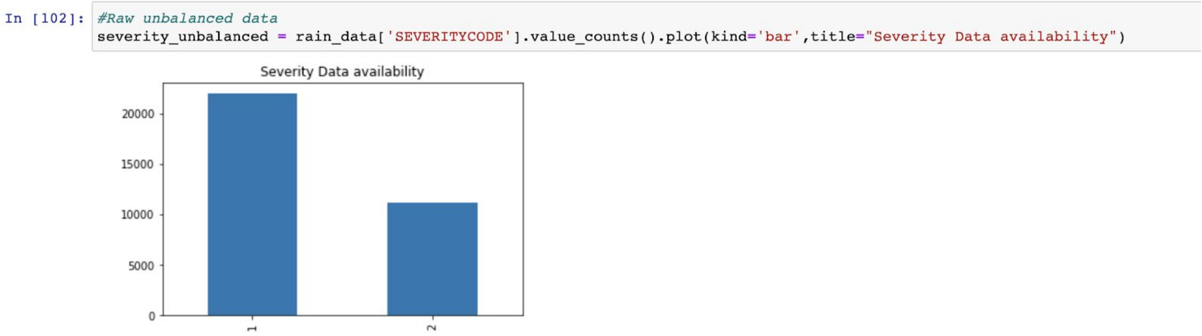
Convert Unbalanced data to balanced dataset

Unbalanced data refers to classification problems where we have unequal instances for different classes. Having unbalanced data is actually very common in general. Downsampling method is followed to achieve this.The main goal of downsampling (and upsampling) is to increase the discriminative power between the two classes.

Here is a plot of Unbalanced data:

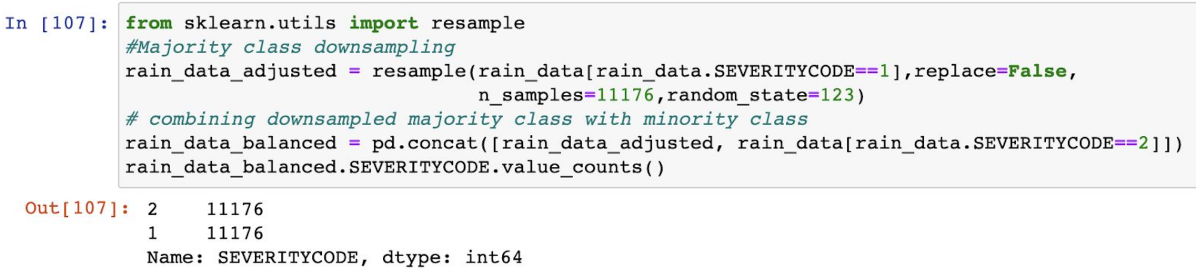


```
In [100]: rain_data['SEVERITYCODE'].value_counts()

Out[100]: 1    21969
          2    11176
          Name: SEVERITYCODE, dtype: int64
```

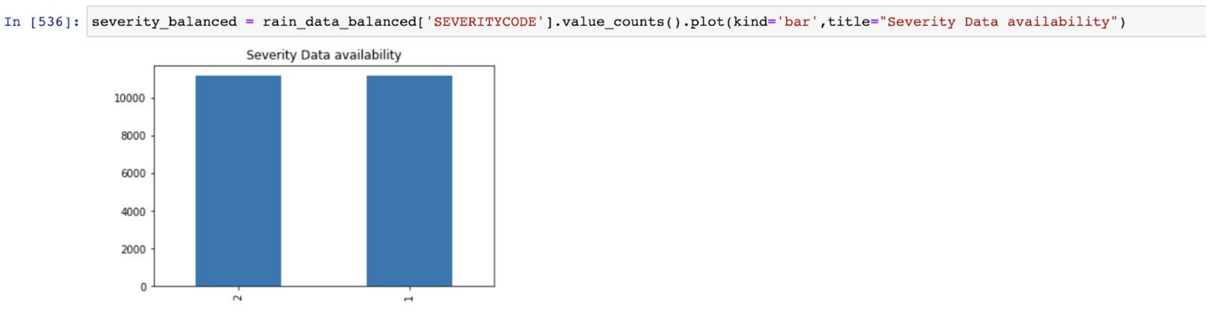


```
In [102]: #Raw unbalanced data
          severity_unbalanced = rain_data['SEVERITYCODE'].value_counts().plot(kind='bar',title="Severity Data availability")
```

Downsampling step



```
In [107]: from sklearn.utils import resample
          #Majority class downsampling
          rain_data_adjusted = resample(rain_data[rain_data.SEVERITYCODE==1],replace=False,
                                        n_samples=11176,random_state=123)
          # combining downsampled majority class with minority class
          rain_data_balanced = pd.concat([rain_data_adjusted, rain_data[rain_data.SEVERITYCODE==2]])
          rain_data_balanced.SEVERITYCODE.value_counts()

Out[107]: 2    11176
          1    11176
          Name: SEVERITYCODE, dtype: int64
```



```
In [536]: severity_balanced = rain_data_balanced['SEVERITYCODE'].value_counts().plot(kind='bar',title="Severity Data availability")
```

Following machine learning models are used for further analysis:

1. K Nearest Neighbor (KNN)
2. Decision Tree
3. Logical Regression

For further information, please refer to jupyter notebook provided in https://github.com/anushreeShenoy3/Coursera_Capstone/blob/master/Capstone_Project_Week1/CapstoneProject.ipynb. Code snippets for data understanding is provided.