Data Understanding

The data used for this project can be found [here] (https://s3.us.cloud-object-storage.appdomain.cloud/cf-courses-data/CognitiveClass/DP0701EN/version-2/Data-Collisions.csv). It consists of all type of collisions on the city of Seattle from 2004 to present. The raw data has 194673 observations (collisions) and the following columns:

```
Index(['SEVERITYCODE', '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', 'WEATHER', 'ROADCOND', 'LIGHTCOND',
       'PEDROWNOTGRNT', 'SDOTCOLNUM', 'SPEEDING', 'ST_COLCODE', 'ST_COLDESC',
       'SEGLANEKEY', 'CROSSWALKKEY', 'HITPARKEDCAR'],
      dtype='object')
```

The target variable is 'SEVERITYCODE', being a code that corresponds to the severity of the collision; 1 for property damage and 2 for injuries.

1. Predictor variables

The predictor variables are 'WEATHER', 'ROADCOND' and 'LIGHTCOND'.

Weather		**ROADCOND**		**LIGHTCOND**	
Variable	Count	Variable	Count	Variable	Count
Raining	111135	Dry	124510	Daylight	116137
Overcast	33145	Wet	47474	Dark – ST ON	48507
Unknown	27714	Unknown	15078	Unknown	13473
Snowing	15091	Ice	1209	Dusk	5902
Other	907	Snow/Slush	1004	Dawn	2502
Fog/Smog/Smoke	832	Other	132	Dark – NO ST	1537
Sleet/Hail/Freezing Rain	113	Standing Water	115	Dark – ST OFF	1119
Blowing Sand/ Dirt	56	Sand / Mud / Dirt	75	Other	235
Severe Crosswind	25	Oil	64	Dark – Unknown Lighting	11
Partly Cloudy	5				

*ST: Street Lights

Using this data, we cannot predict the weather of Seattle as we do not have the time of the collisions. Therefore, we will remove the red-coloured values on our table, corresponding to unknown values, or values that do not have any relationship with the weather (e.g. road oil).

We will remove the variables as well that had really low observations: Blowing Sand/ Dirt and Severe Crosswind.

Moreover, we will join several categorical groups of data that means the same thing (e.g. Dark No Street Lights, Dark Street Lights off). We have coloured using the same colour.

Another option will be use clustering all these variables and doing an important dimension reduction.

2. Null values

Being this a preliminary project, we will just remove any null/NaN values for the predictor and target variables. If we had more time, we should try to estimate any of these values (e.g. using data we can estimate the probability that on a rainy day the road will be wet).

3. Get Dummies

Being all our data categorical, we need to split it into different features:

```
df = df[['SEVERITYCODE', 'WEATHER', 'ROADCOND', 'LIGHTCOND']]
```

```
df = pd.concat([df,pd.get_dummies(df['WEATHER'])], axis=1)
df = pd.concat([df,pd.get_dummies(df['ROADCOND'])], axis=1)
df = pd.concat([df,pd.get_dummies(df['LIGHTCOND'])], axis=1)
```

```
df = df.drop([['ROADCOND', 'LIGHTCOND','WEATHER'], axis = 1)
df.head()
```

The preconditioned data looks this way:

	SEVERITYCODE	Clear	Fog/Smog/Smoke	Overcast	Raining	Snowing	Dry	Ice	Snow/Slush	Snow/slush	Wet	Dark - No Street Lights	Dark - Street Lights On	Dawn	Daylight	Dusk
0	2	0	0	1	0	0	0	0	0	0	1	0	0	0	1	0
1	1	0	0	0	1	0	0	0	0	0	1	0	1	0	0	0
2	1	0	0	1	0	0	1	0	0	0	0	0	0	0	1	0
3	1	1	0	0	0	0	1	0	0	0	0	0	0	0	1	0
4	2	0	0	0	1	0	0	0	0	0	1	0	0	0	1	0

This method can be inefficient and can carry multiple problems such as overfitting. This is because we are introducing multiple features in our model. A better method might be having only 3 features (weather, road and light conditions) and class the different features into new integer variables such as: weather_condition, road_condition and light_condition.

4. Data Imbalance

Our target variable 'SEVERITYCODE' is imbalanced. Imbalanced data typically refers to a problem with classification problems where the classes are not represented equally. In our case we have more than two times more cases with value 1 than with value 2. If we were to fit and train our data we will probably have an Accuracy Paradox, where our code will probably

estimate than having all or almost all our data predict target value equal to 2 for all our data will have a good accuracy score.

```
1    114188
2     55638
Name: SEVERITYCODE, dtype: int64
```

Here we have several options:

a) Use an algorithm that performs well on imbalanced datasets, such as Decision Trees.
b) Modify our dataset deleting instances from the over-represented class (y=1), under-sampling.
c) Add copies of instances from the under-represented class (y =2), over-sampling.
d) Divide the over-represented class into two distinct clusters, then train two predictors, where each predictor is trained on only one of the distinct clusters, but on all the data from the rare class. The we use model averaging for the two learned predictors as my final predictor.

We will try to do the (d) method, splitting randomly our database into two clusters and predict both models.

```
# y bewteen 0 and 1.
df['SEVERITYCODE'] = df['SEVERITYCODE']-1
```

```
code_zero = df[df['SEVERITYCODE']==0]
code_one = df[df['SEVERITYCODE'] ==1]
```

```
from sklearn.model_selection import train_test_split

code01, code02 = train_test_split(code_zero, test_size=0.5)
```

```
model1 = code_one.append(code01)
model2 = code_one.append(code02)
```

```
y1 = model1['SEVERITYCODE'].values
y2 = model2['SEVERITYCODE'].values
```

```
model1 = model1.drop(['SEVERITYCODE'], axis = 1)
model2 = model2.drop(['SEVERITYCODE'], axis = 1)
model2.head()
```

Figure 1: Split of the model into two different models for unbalanced data

5. Data Normalisation

The data does not need to be normalised, as all the values are between 0 and 1.