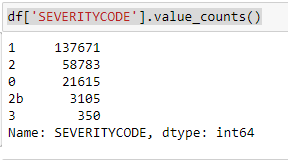
Capstone Project

Car Accident Severity

# About the dataset

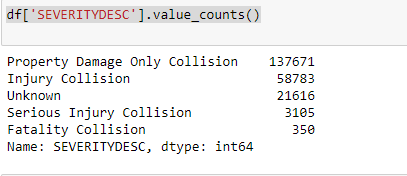
The data for this project is taken from an open source website - [**https://data-seattlecitygis.opendata.arcgis.com/datasets/5b5c745e0f1f48e7a53acec63a0022ab\_0/data**](https://data-seattlecitygis.opendata.arcgis.com/datasets/5b5c745e0f1f48e7a53acec63a0022ab_0/data)

The data consists of 39 independent variables and 221525 rows. The dependent variable, “**SEVERITYCODE**”, contains codes that correspond to different levels of severity caused by an accident.



Severity codes are as follows:

* 0 : Unknown
* 1 : Property Damage Only Collision
* 2 : Injury Collision
* 2b : Serious Injury Collision
* 3 : Fatality Collision

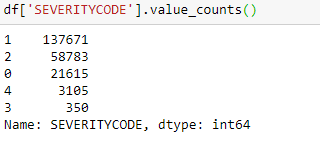


Furthermore, because of the existence of null values in some records, the data needs to be preprocessed before any further processing.

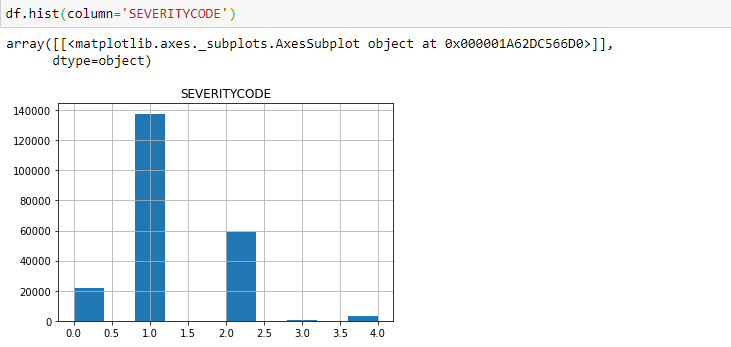
# Data Wrangling

The dataset in the original form is not ready for data analysis. In order to prepare the data, first, we need to drop the non-relevant columns. In addition, most of the features are of object data types that need to be converted into numerical data types. We have to convert the **SEVERITYCODE** our target variable into numerical data type too.

For this we have updated the value of **SEVERITYCODE** ‘2b’ to ‘4’ and then updated the column data type.

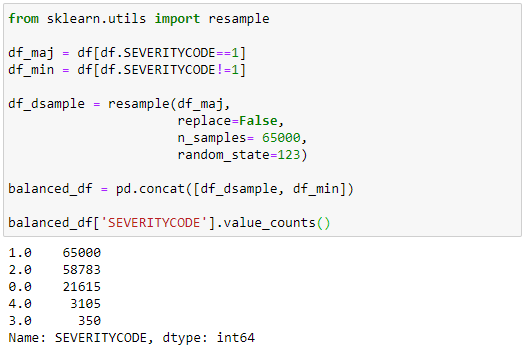


To get a good understanding of the dataset, I have checked different values in the features. The results show, the target feature is imbalance, so we use a simple statistical technique to balance it.



The number of rows in class 1 is almost three times bigger than the number of rows in class 2. It is possible to solve the issue by down sampling the class 1.

Post down sampling the data for class 1 we get a balanced target feature. Though the overall record size reduced but the model built would not be biased.



# Methodology

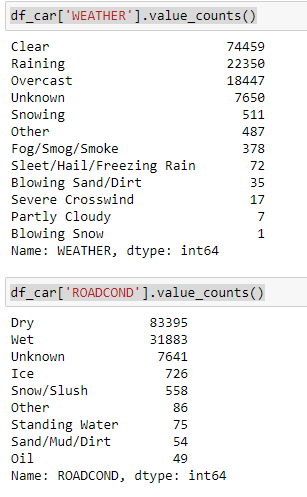
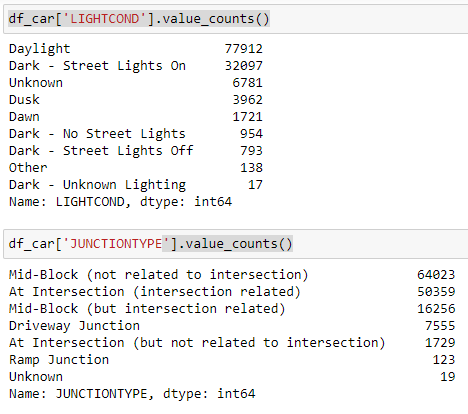
For implementing the solution, I have used GitHub as a repository and running Jupyter Notebook to preprocess data and build Machine Learning models. Regarding coding, I have used Python and its popular packages such as Pandas, NumPy and Sklearn.

Once I have load data into Pandas Dataframe, used ‘dtypes’ attribute to check the feature names and their data types. Then I have selected the most important features to predict the severity of accidents in Seattle. Among all the features, the following features have the most influence in the accuracy of the predictions:

* COLLISIONTYPE
* PERSONCOUNT
* PEDCOUNT
* PEDCYLCOUNT
* VEHCOUNT
* INJURIES
* SERIOUSINJURIES
* FATALITIES
* JUNCTIONTYPE
* WEATHER
* ROADCOND
* LIGHTCOND
* SPEEDING

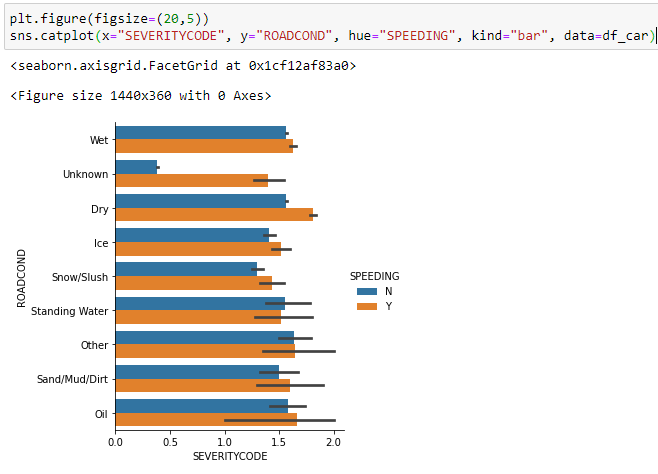
Also, as I mentioned earlier, “**SEVERITYCODE**” is the target variable.

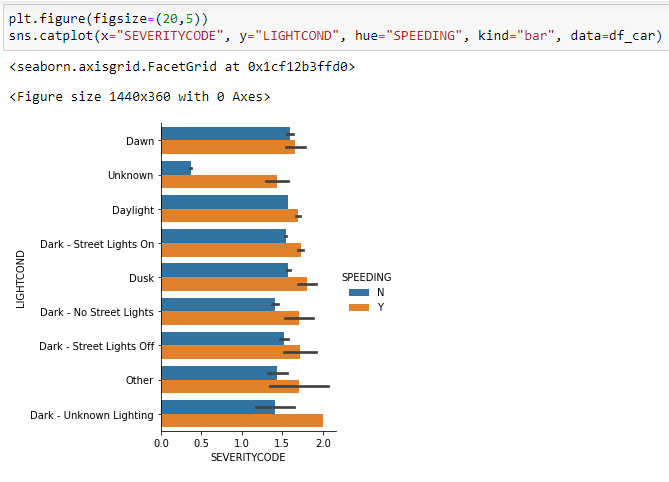
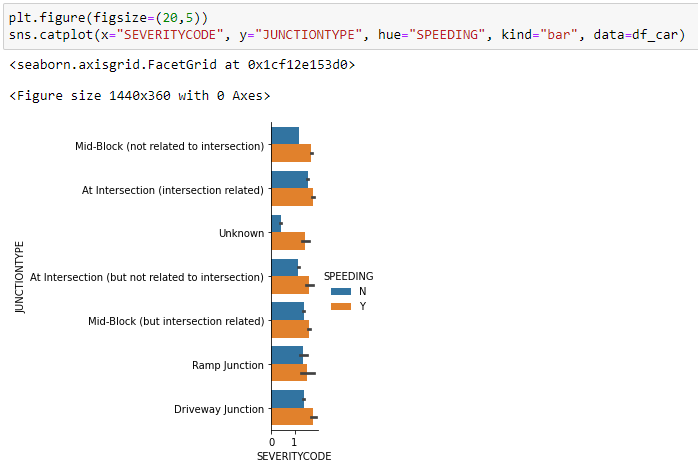
I have run a value count on road (‘**ROADCOND**’) and weather condition (‘**WEATHER**’) to get ideas of the different road and weather conditions. I also have run a value count on light condition (**’LIGHTCOND**’), to see the breakdowns of accidents occurring during the different light conditions. Similarly we have run value count for other attributes as well. The results of a few can be seen below:

# Exploratory Data Analysis

The below graphs shows severity levels of accidents basis the different attributes like WEATHER, ROADCOND, LIGHTCOND and JUNCTIONTYPE.

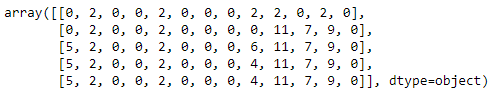
# Data Pre-processing

Using **df\_car** as the final data after removing extra fields, declare the following variables:

* **X**as the **Feature Matrix**(data of df\_car)
* **y**as the **response vector (SEVERITYCODE)**

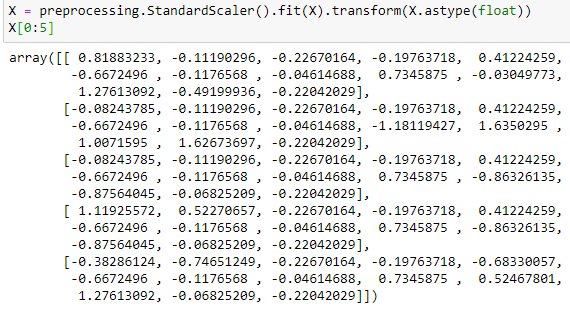
As you may figure out, some features in this dataset are categorical such as **WEATHER, ROADCOND,** **LIGHTCOND,** etc. Unfortunately; Sklearn Decision Trees do not handle categorical variables. But still we can convert these features to numerical values. **pandas.get\_dummies()** Convert categorical variable into dummy/indicator variables.





# Normalize Data

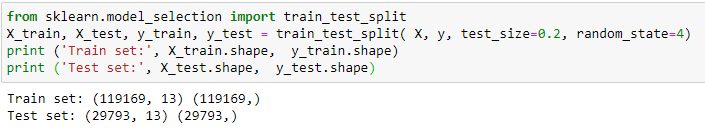
Data Standardization give data zero mean and unit variance, it is good practice, especially for algorithms such as KNN which is based on distance of cases:



# Train Test Split

Out of Sample Accuracy is the percentage of correct predictions that the model makes on data that that the model has NOT been trained on. Doing a train and test on the same dataset will most likely have low out-of-sample accuracy, due to the likelihood of being over-fit.

It is important that our models have a high, out-of-sample accuracy, because the purpose of any model, of course, is to make correct predictions on unknown data. So how can we improve out-of-sample accuracy? One way is to use an evaluation approach called Train/Test Split. Train/Test Split involves splitting the dataset into training and testing sets respectively, which are mutually exclusive. After which, you train with the training set and test with the testing set.



# MODELING

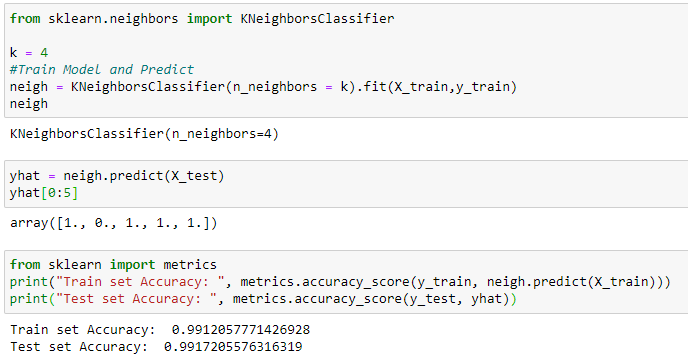
After balancing SEVERITYCODE feature, and standardizing the input feature, the data has been ready for building machine learning models.

I have employed three machine learning models:

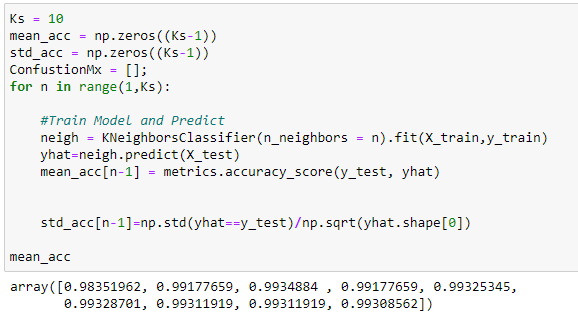
* K Nearest Neighbor (KNN)
* Decision Tree
* Logistic Regression

After importing necessary packages and splitting preprocessed data into test and train sets, for each machine learning model, I have built and evaluated the model and shown the results as follow:

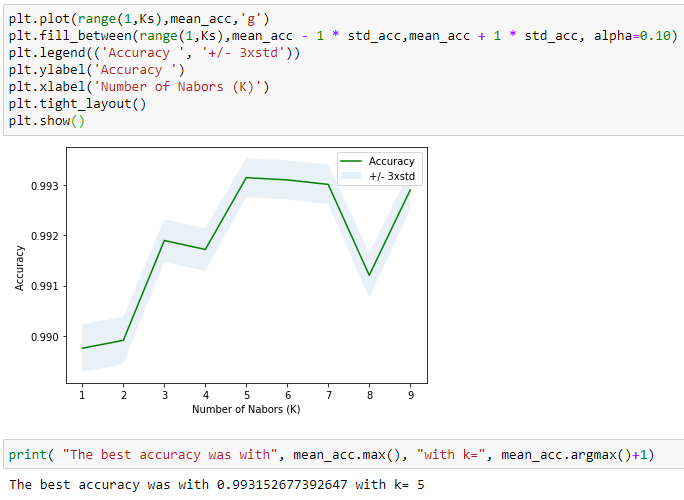
# K Nearest Neighbor (KNN)



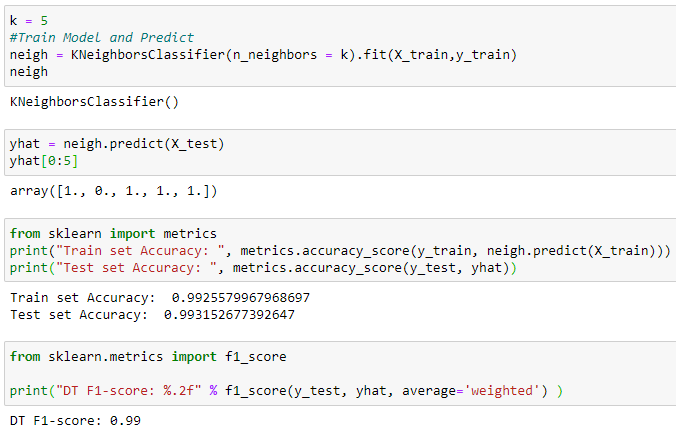
We can calculate the accuracy of KNN for different Ks.



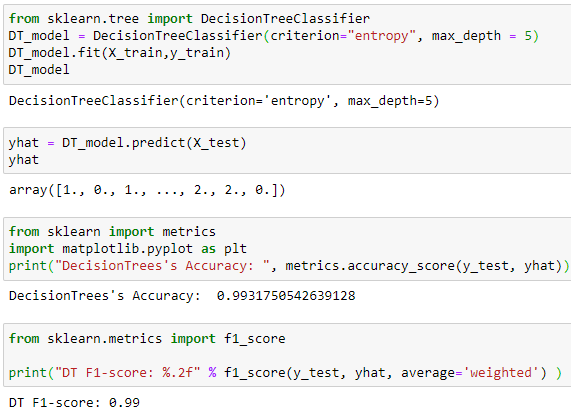
#### Plot model accuracy for Different number of Neighbors



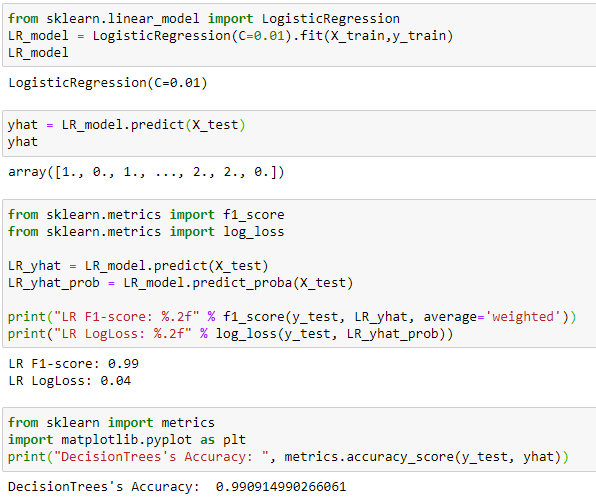
Then we rebuild the model with best value for K=5.



# Decision Tree



# Logistic Regression



# Results and Evaluations

The final results of the model evaluations are summarized in the following table:

|  |  |  |
| --- | --- | --- |
| ML Model | F1-Score | Accuracy |
| KNN | 0.99 | 0.993 |
| Decision Tree | 0.99 | 0.993 |
| Logistic Regression | 0.99 | 0.990 |

# Conclusion

Based on the dataset provided for this capstone from weather, road, light conditions, etc. pointing to certain classes, we can conclude that particular conditions have a somewhat impact on whether or not travel could result in property damage or injury.