**Prediction of Traffic Accident Severity**

**Abstract**

In this report, we are going to show you how we manipulate the data acquired from Seattle government to build model for predicting severity of traffic accidents. Also, we will evaluate each algorithm to decide which the best one for this case is.

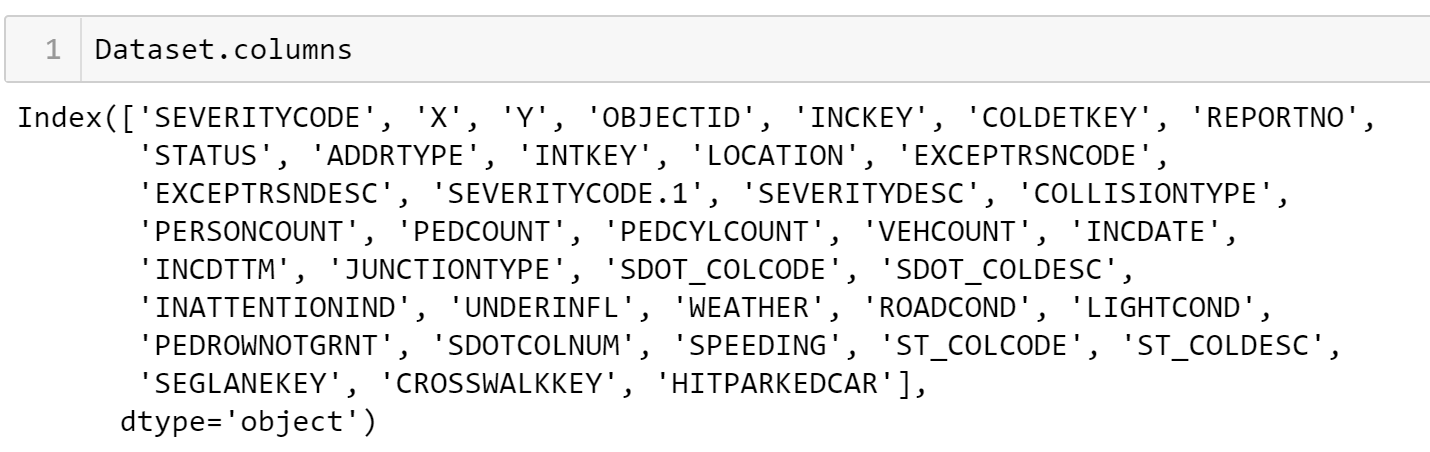
**Introduction**

1. **Background:**

Machine learning is now a universal tool in data science that can be applied to any kind of field. In this work, we would like to predict if a certain traffic accident event would cause injury or simple property damage. By applying classification machine learning algorithm with well-defined features and analysis, it is expected that we can have a fine model for the prediction.

1. **Dataset:**

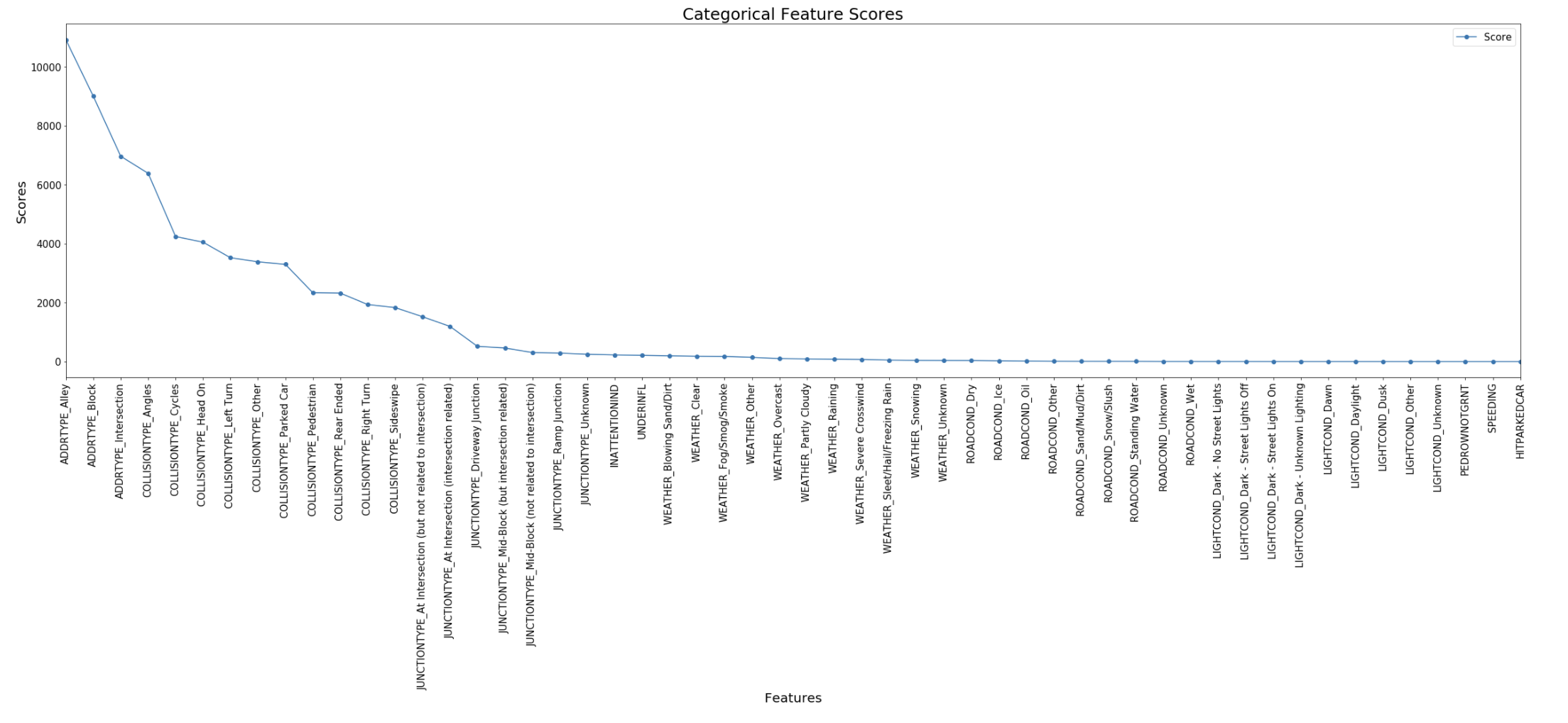
Our original dataset consists of 38 columns (columns shown in picture below) and 194673 rows. This dataset includes traffic accidents with several useful features recorded such as weather, road condition and so on.



1. **Data Cleaning, Feature Selection and Preprocessing**

In order to make good use of our dataset, we first split the dataset into numerical and categorical subsets. In the numerical dataset, we check and make sure there are no Nan or null values in the dataset after deleting meaningless code like feature. From the correlation coefficient inspection, we know that there are no features with correlation coefficient lower than 0.05. Thus, we keep all features in numerical dataset.

As for categorical dataset, the first step is also deleting non-relevant columns by intuition and second step is turning them into dummy variables. Then, we use SelectKBest method to pick features with highest score. The result is shown in the plot below. Eventually, we pick feature before the apparent turning point at feature number 15.



After selecting all features, we normalize the numerical dataset to fix the scale of feature between 0 to 1. Just to mention, in our prediction, 0 means injury traffic accident whereas 1 means property damage only traffic accident.

**Methodology**

In this work, we will apply four commonly used classification algorithm, that is , K-Nearest Neighbors, Support Vector Machine, Decision Tree and Logistic Regression

.

**K-Nearest Neighbors**

K-Nearest-Neighbors algorithm compares distance between the current data point and other data points, and we choose the data points (dependent on K) that is closest to current data point. Finally, we determine the classification of this data point by the majority of chosen K data points.

**Support Vector Machine**

Support vector machine algorithm considers data as multidimensional point in multidimensional space. The algorithm tries to find hyper planes that could divide data points with greatest margin which can be seen as nearest distance to the separation plane.

**Decision Tree**

The Decision Tree algorithm creates tree like split on the basis of our features. The performance could be dependent on the max depth of this tree.

**Logistic Regression**

Logistic regression is a non-linear transformation of linear combinations of our features. Due to the non-linear transformation method we apply, the results after computation can be seen as probability belonging to certain group (0 or 1).

**Results and Discussion**

1. **Primitive Algorithm Exploration**

**Logistic Regression**

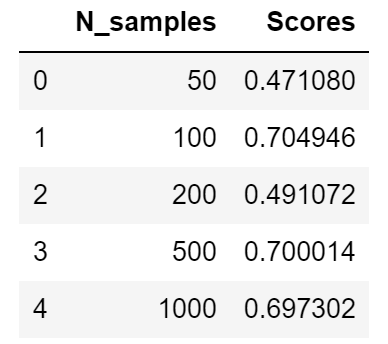
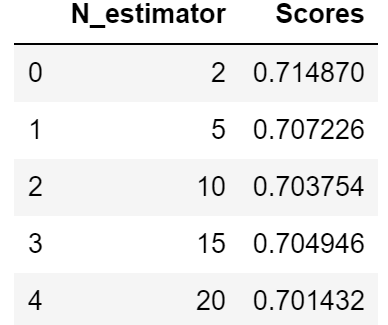
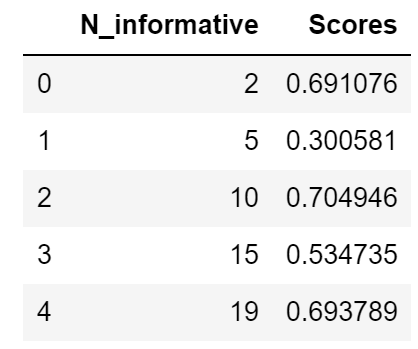
In the logistic regression part, we adjust regularization term as a variable to see the best option for logistic regression in our case. The result table is below, and it seems that this term does not really affect much on our model. Hence, in the final comparison part, we will use regularization term 1 as our choice.



**Support Vector Machine**

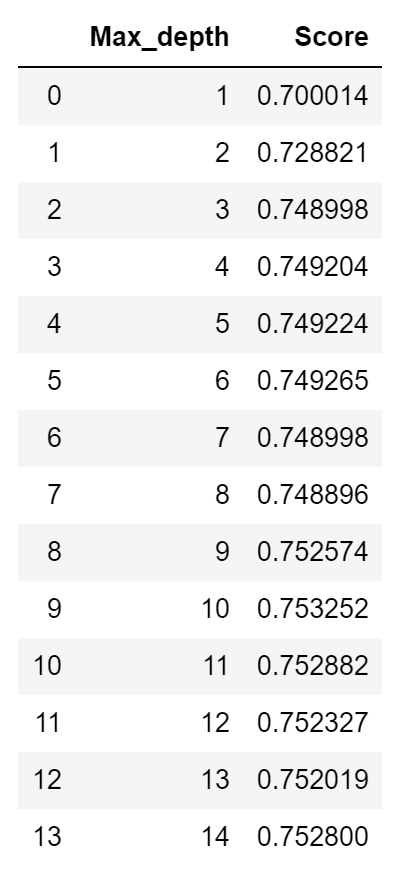
In support vector machine part, we change our way of training process due to the low execution efficiency of SVM in scikit learn. Instead, we use bagging classifier with SVM as basic algorithm to do our training. This classifier will randomly pick several samples and informative feature to reduce the run time of training process. We also want to see the influence of number of samples, number of informative and number of estimators.

The number of informative and samples does not seem to have clear trend and number of estimators does not seem to affect the result much. Thus, we just pick n\_sample 100, n\_informative 10 and n\_estimators 15 as our parameters.



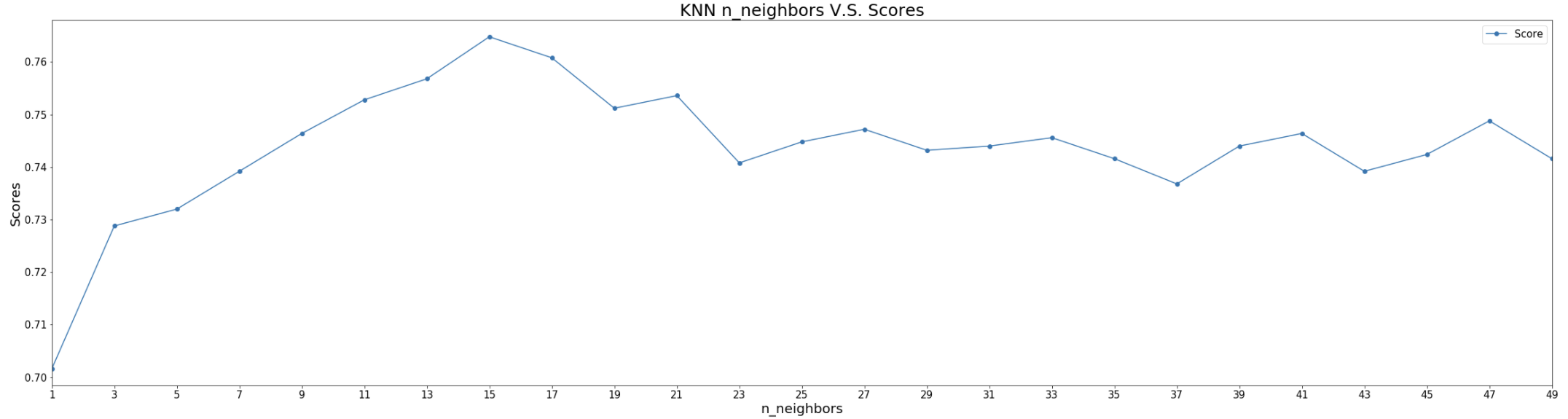
**Decision Tree**

In the part of decision tree, we adjust max\_depth to see the influence of it. From the table below, we can infer that as long as the max depth is set to a large enough value, the performance will somehow converge. Here we pick 10 as our final parameter.



**K-Nearest-Neighbors**

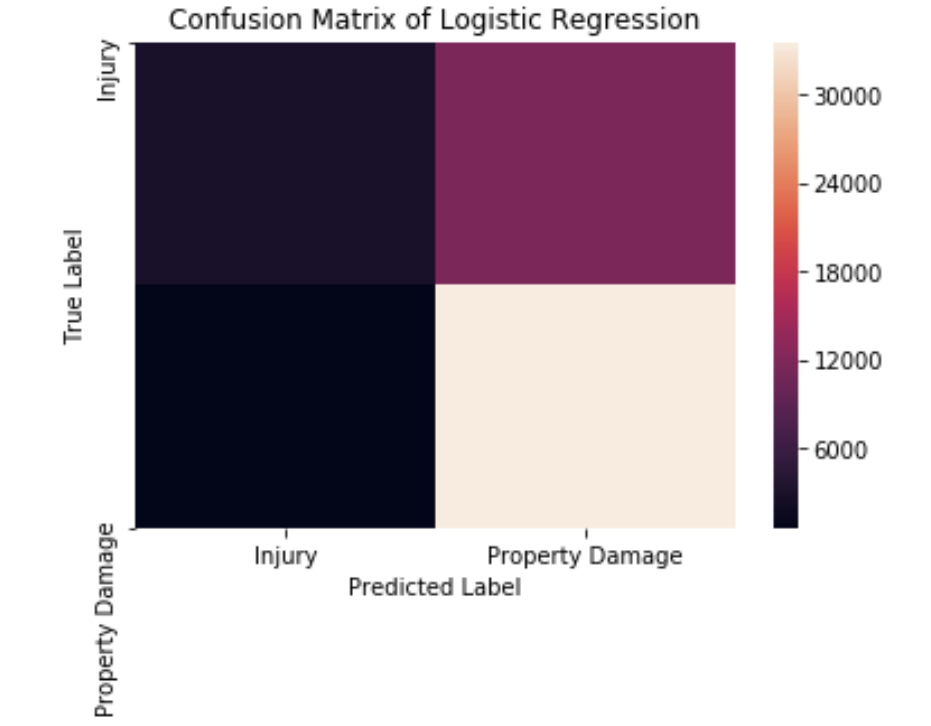
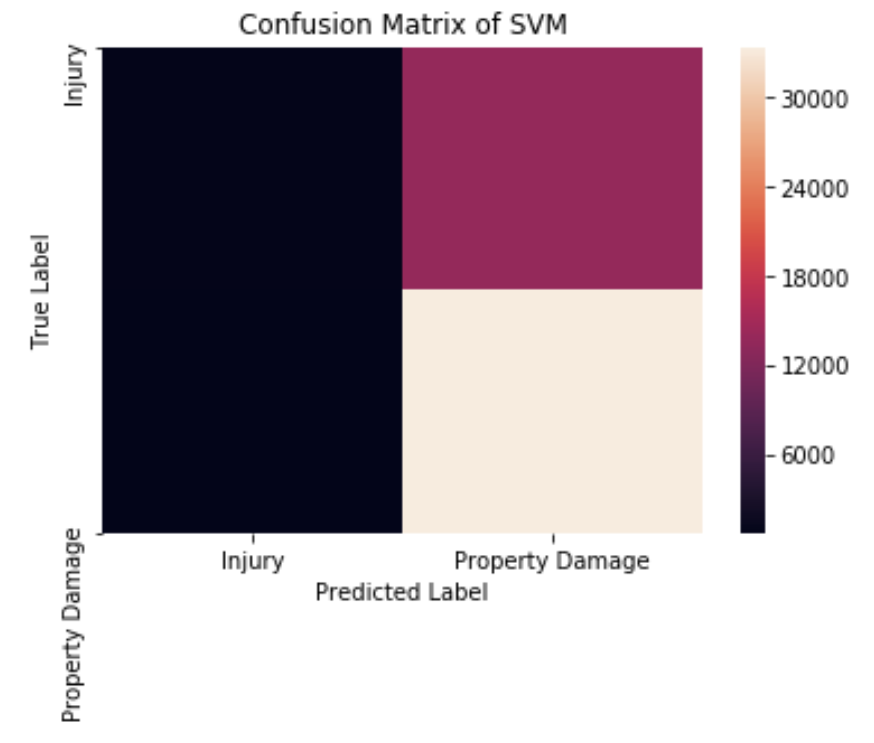
In the KNN part, we adjust the k value which stands for the number of closest neighbors referenced. Because the running efficiency is not good for KNN in scikit learn, I tend to use sample() method to randomly pick samples and do several times to evaluate the performance. According to result shown in plot below, we can observe that there exists a rough maximum score at around K=15. Thus, we choose K=15 as our final parameter.

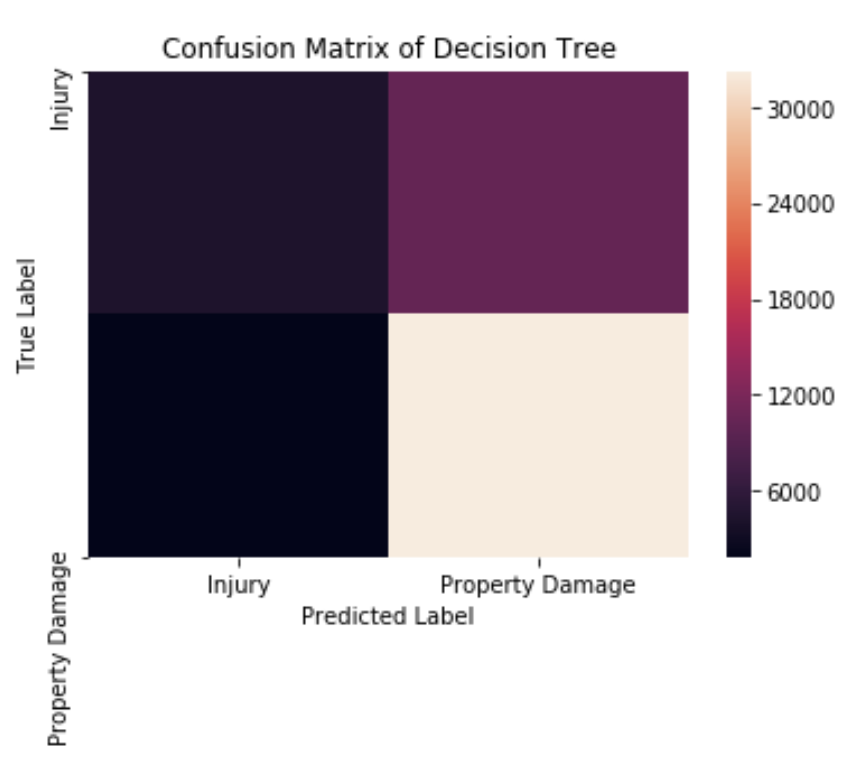
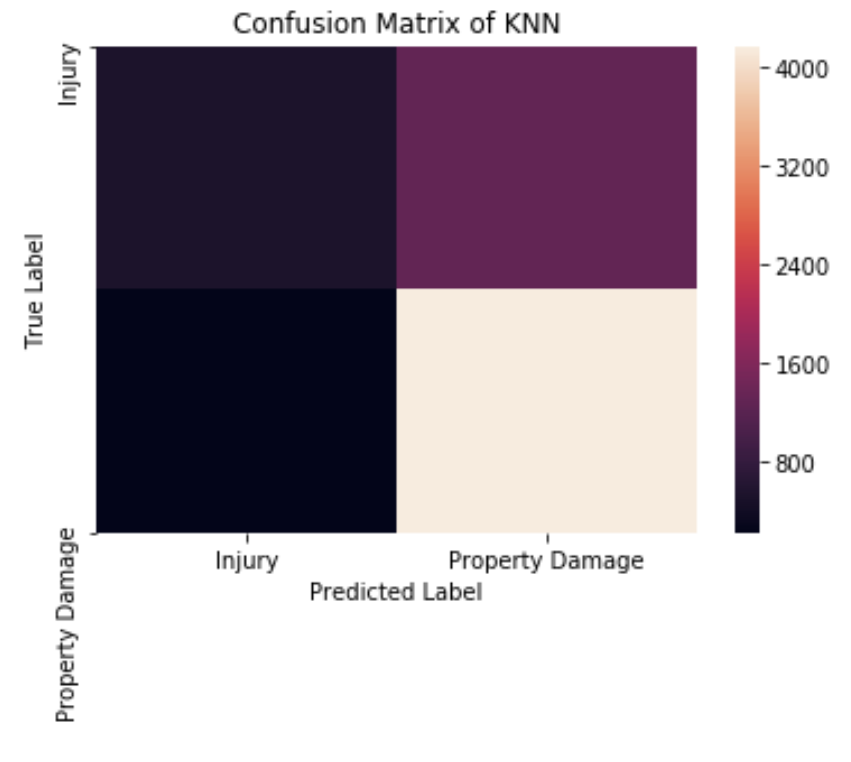


1. **Performance Evaluation and Reporting**

**Confusion Matrix**

In order to have a brief insight to the performance of each algorithm, we visualize the result of confusion matrix with heatmap.

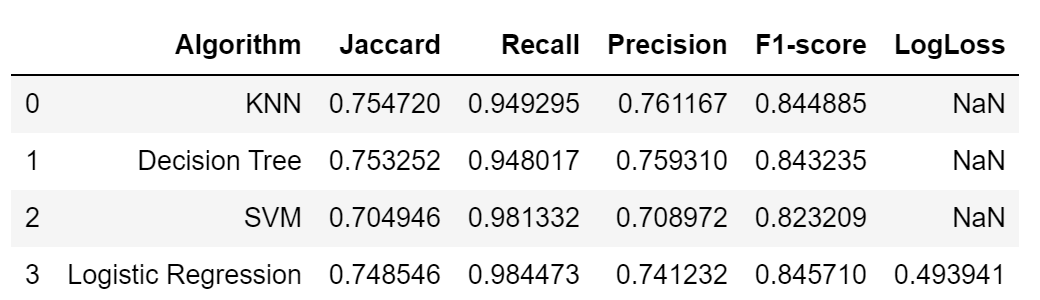
 

Roughly from the visualized confusion matrix, we can see that most algorithms do not bad job at predicting correctly. As we expect, true positive predictions is major. The false positive rate of SVM is a bit higher than other 3 algorithms.

**Classfication Report**

Below is some key scores used to evaluate machine learning algorithms including Jaccard score, recall, precision, f1-score and log loss specifically for logistic regression.



The first column lists the Jaccard scrore that reflects the overall accuracy. We can see that SVM has relatively low score in this term. The second one is recall which represents the fraction of true labels being correctly predicted. The SVM and logistic regression does good job at this. But when looking into precision, SVM does relatively bad as we mentioned in previous heatmap. F1 score is a balanced score considering both recall and precision. SVM does relatively not good at this whereas other three also remains decent.

**Conclusion**

According to our final results, we can say that SVM seems not a good choice for this application. KNN has best accuracy but somehow a bit lower recall. KNN could be a good choice for relatively small dataset. Logistic regression has decent accuracy as well as good recall. Besides, logistic regression is relatively computation efficient, it may be good for larger dataset. Decision tree has middle class performance at every term, thus it may be a safe choice if you do not know which algorithm to apply or you want an algorithm to compare with another. Also, decision tree may be good at larger dataset as well.

Performance of our models may be restricted by some factors. First is the limitation of computation power. For example, the efficiency of SVM computation in scikit learn is bit low which limit our training samples. Second is that the missing categorical values. In our case, we do not add any value in Nan or null value. Instead, we let it be Nan or null and this means 0s in every dummy variables. These factors could limit our learning performance.