Predicting vehicle collisions for the area of Seattle

Rodrigo Galdino

September 06, 2020

Abstract: This is a machine learning project to predict the probability of vehicle collisions on the roads of the city of Seattle. it uses classification model to learn to predict the type of the collision based on accident reports provided by the Seattle Police Department and provided by Coursera via a download link.

1. Introduction: Business Problem

1.1 Background

The Seattle Department of Transportation’s annual traffic report poses a significant challenge to governments to review their road safety policies and take new initiatives that make drivers' lives safer.

1.2 Problem/Opportunity Statement

Given this scenario, the objective of this project is to provide the best classification model to predict the type of collision (an auto accident involving injury or property damage) based on the weather and road conditions during the collision, the geographical area where the collision took place, the type of the collision and the day of the week that the collision occurred.

1.3 Interest

The information produced through this model may serve as an instrument for decision support in the prioritization of actions, such as changing road infrastructure, the procurement and installation of the road signs to reflect the weather conditions. Moreover, these informations could alert drivers, given the weather and road conditions about the possibility of a car accident and the severity of it, which would allow the driver to drive more carefully or, even change his itinerary.

2. Data

2.1 Data sources

The data[1] used corresponds to the Seattle Department of Transportation's annual Traffic Report Dataset.

This data was collected by the during the period of 2004-2020 and contains such information as severity, location, collision type, weather conditions, road conditions, and light conditions, among others.

2.2 Data cleaning

As part of the cleaning process the data was downloaded and extracted, the whole dataset was loaded and converted into a pandas dataframe, allowing for a better use of the data set to get a sense of the data structure, missing and bad data, and what kind of data processing cleaning should take place.

So, the data set consists of 194,673 observations and 38 features.

The method "seaborn.heatmap" was used to find the missing values. The columns with more than 90% of missing values were removed from dataset.

After data cleaning, there were 194,673 samples and 35 features in the data.

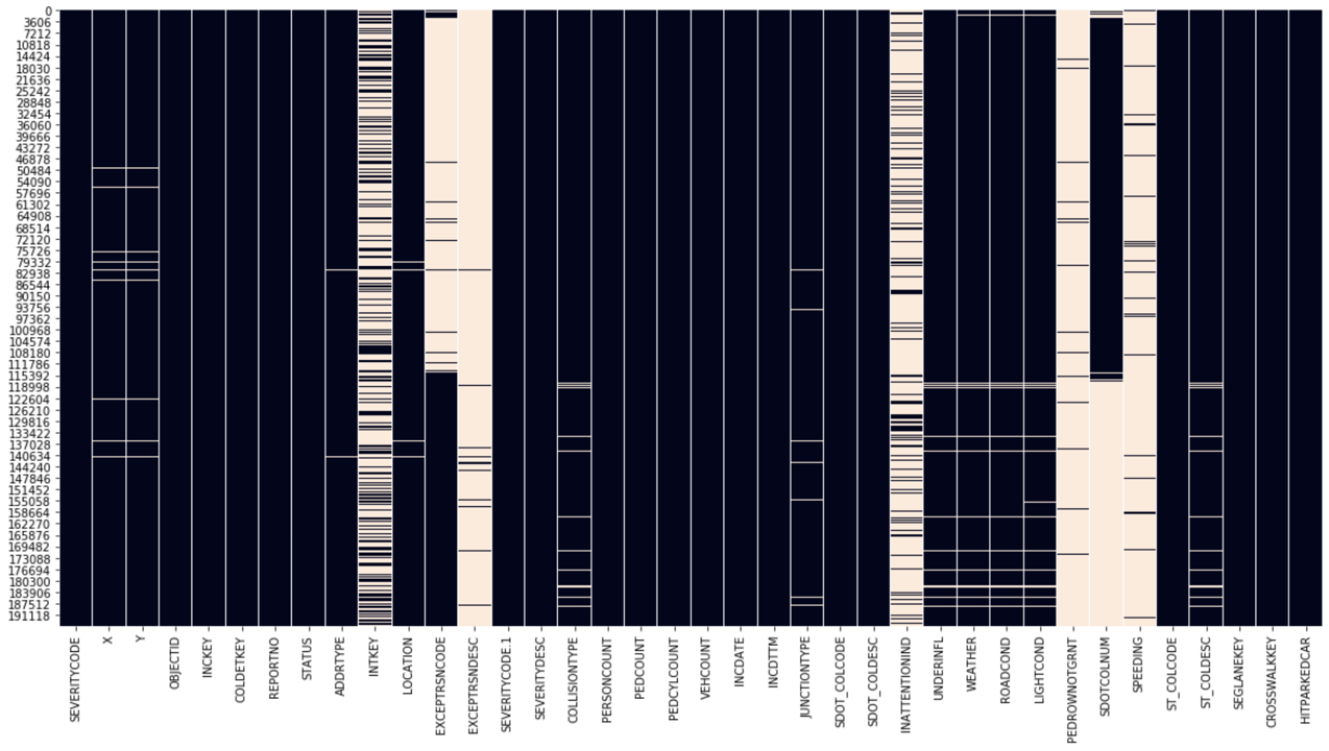


Fig. 1 Missing values before cleaning

[1] For further details about the data set used, see https://www.seattle.gov/Documents/Departments/SDOT/GIS/Collisions_OD.pdf

2.3 Feature selection

At this stage, the features of the dataset were reviewed, with the aim of verifying those that may or may not have a significant association with the dependent variable.

Then a correlation matrix was created to understand how features are correlated with target: SEVERITYDESC

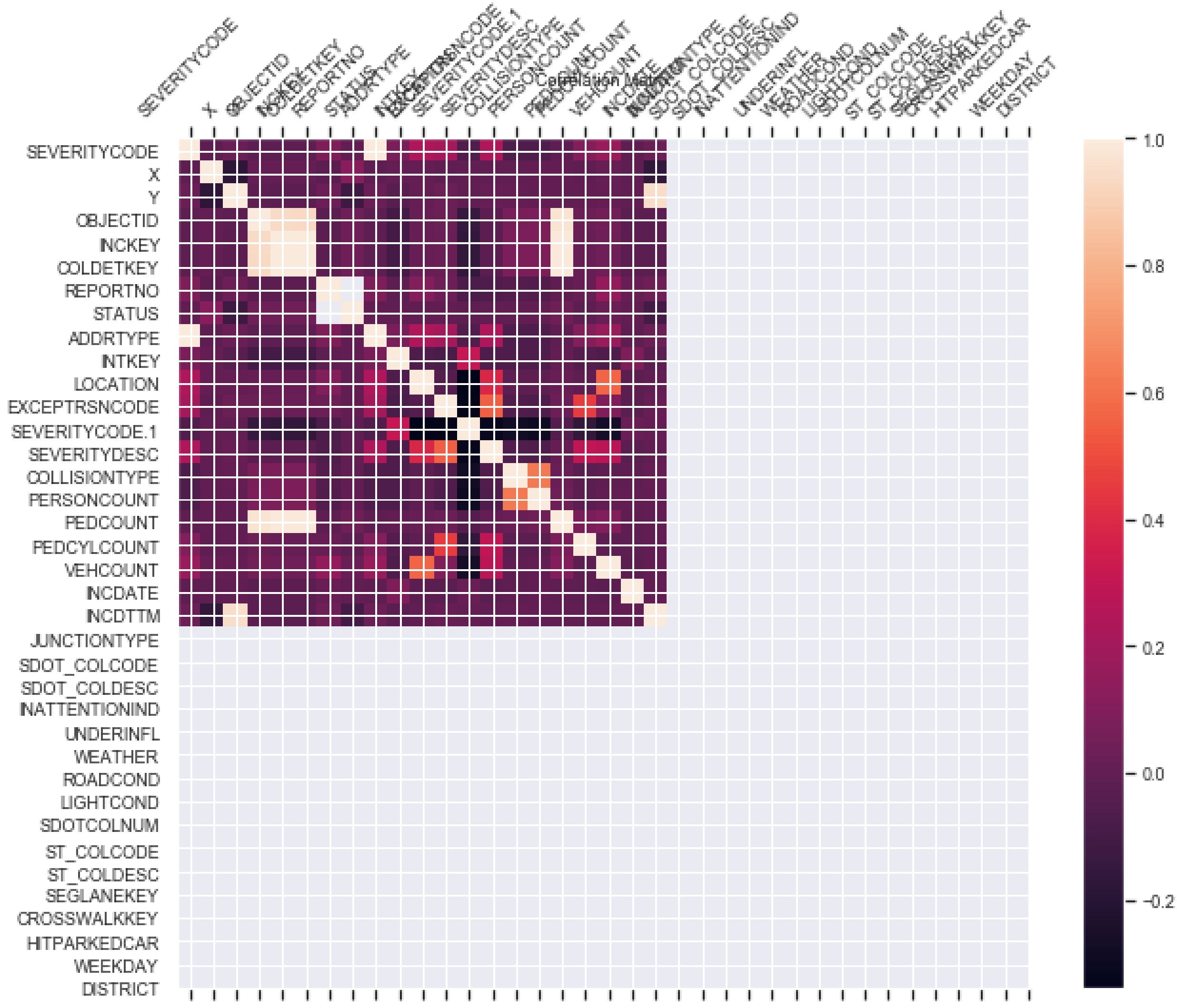


Fig. 1 Correlation matrix.

After this verification, the variables given below were used for machine learning purposes, where the target variable was the column SEVERITYCODE:

Data Set Basics	
SEVERITYCODE	A code that corresponds collision: • 2—injury, • 1—prop damage

SEVERITYDESC	Description collision: Injury Collision, Property Damage Only Collision.
WEATHER	A description of the weather conditions during the time of the collision: Clear, Severe Crosswind, Sleet/Hail/Freezing Rain, Snowing, Overcast, Raining.
ROADCOND	The condition of the road during the collision: Wet, Dry, Ice, Standing Water, Sand/Mud/Dirt, Oil, Snow/Slush.
LOCATION	The geographical area where the collision took place.
INCDATE	The date of the incident.
ADDRTYPE	Collision address type: • Block • Intersection

In short, the main changes were as follows:

(1) Convert the column INCDATE (the date of the incident) from type "str" to "datetime"

(2) Remove rows with missing values on columns "latitude" and "longitude"

(3) Create a new column 'WEEKDAY' that contains the days of the week, from the column INCDATE. This new column serves as an indicator the days of the week that are more dangerous for drivers.

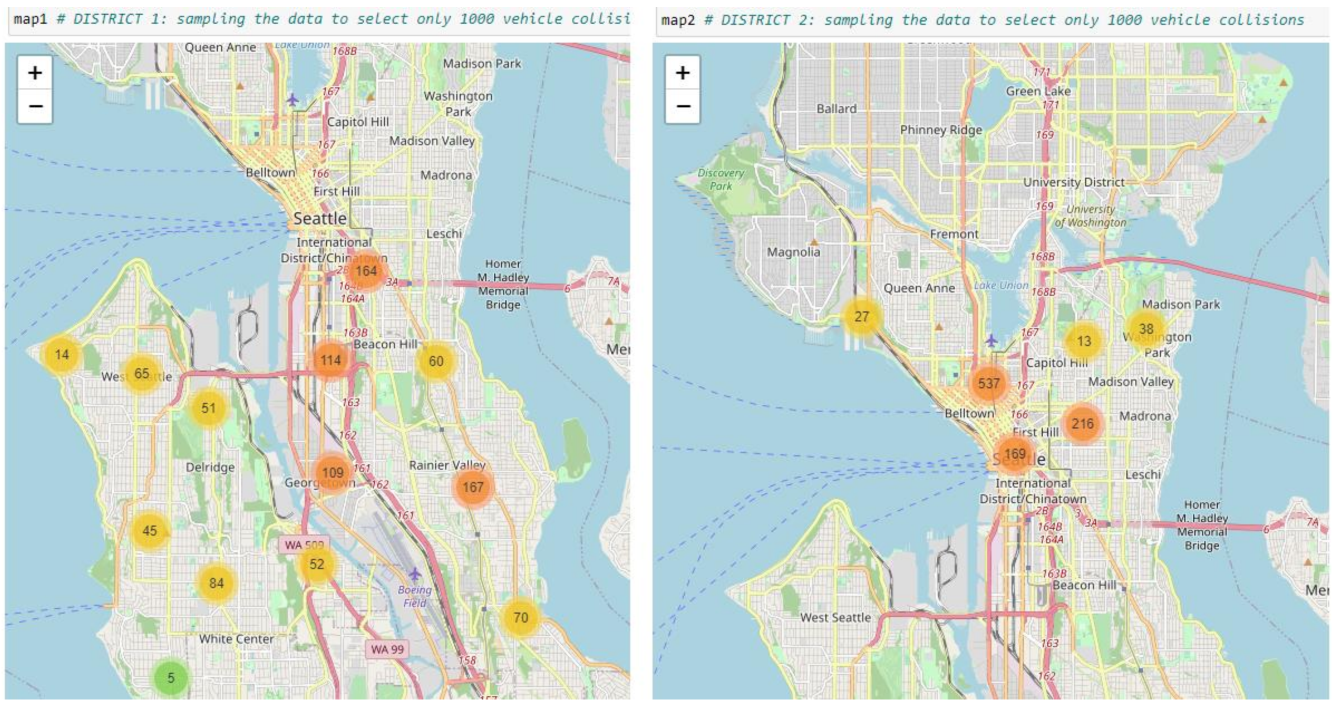
(4) In the Column 'COLLISIONTYPE', the rows whose values correspond 'Parked Car' and ‘Other' were dropped. Thus, the values in this column were considered, which correspond to: Angles, Rear Ended, Left Turn, Sideswipe, Pedestrian, Cycles, Right Turn and Head On.

In addition to these, a new column was created “DISTRICT”, grouping car accidents by area. For this, columns "longitude" and "latitude" were used, dividing Seattle into four districts.

- District 1: from latitude 47.4 to latitude 47.5.
- District 2: the latitudes 47.60, 47.61, 47.62 and 47.63.
- District 3: the latitudes 47.64, 47.65, 47.66 and 47.67.
- District 4: the latitudes 47.68, 47.69 and 47.7.

After feature selection, there were 120,681 rows and 3 columns.

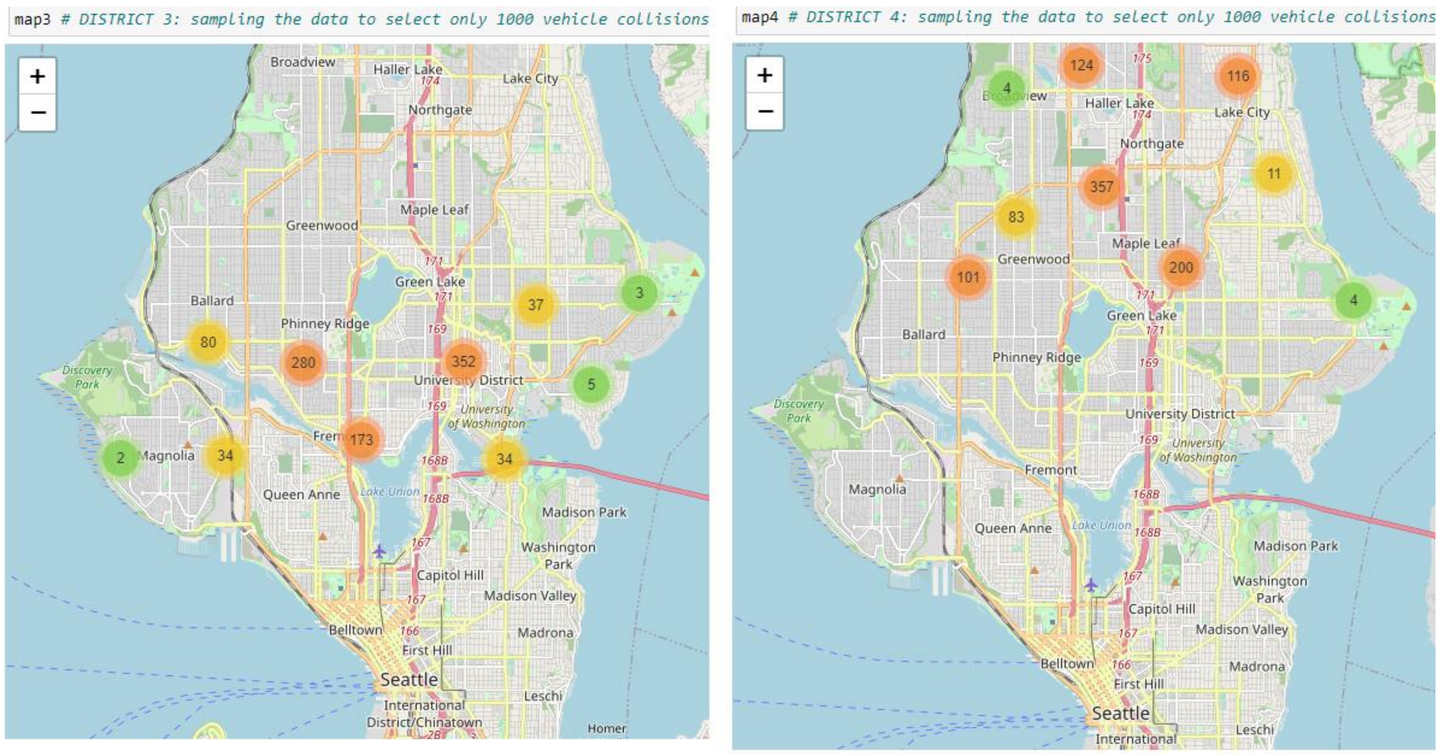
By using plugins in the folium, with a random sample of 1,000 rows, the figures below were generated to provide illustrations of the four areas / districts.



```
map1 # DISTRICT 1: sampling the data to select only 1000 vehicle collisi
```

```
map2 # DISTRICT 2: sampling the data to select only 1000 vehicle collisions
```

Fig. 2 Districts 1 and 2 representing column ‘District’ of the dataframe.



```
map3 # DISTRICT 3: sampling the data to select only 1000 vehicle collisions
```

```
map4 # DISTRICT 4: sampling the data to select only 1000 vehicle collisions
```

Fig. 3 Districts 3 and 4 representing column ‘District’ of the dataframe.

3. Methodology

The methodology used was based on the CRISP-DM (Cross Industry Standard Process for Data Mining) model.

In first step, the analytical approach was identified, based on the question: "In a given condition of road, climate, day of the week and an area of the city of Seattle, if an accident occurs, what would be the severity: an auto accident involving injury or property damage?"

The second step was to identify the analytical approach, that is, the type of model needed to address the question (defined in the first step) more effectively, from which a classification model was used to identify the combination of conditions that lead to the result of each accident.

In third step, the phases "data requirements", "data collection", "data understanding" and "data preparation" were performed in a way that addresses missing or invalid values and removes duplicates, toward ensuring that everything was properly formatted.

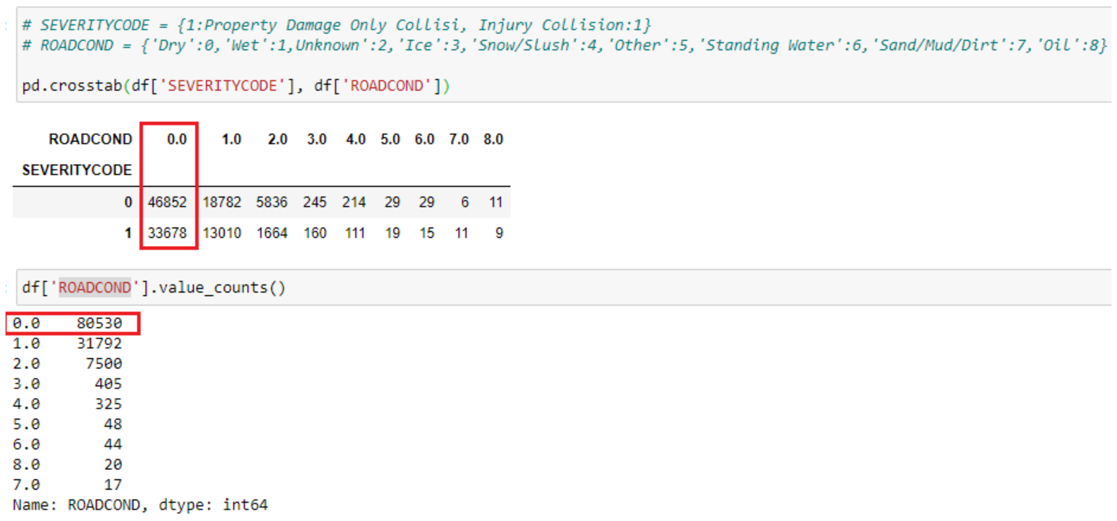
In fourth and final step, the modeling and evaluation phases were applied to the model building as such, but it was also to see if the model meets the initial question.

4. Analysis

4.1 Exploratory Data Analysis

At this stage it was possible to identify the relationships between variables, assess them and do some analysis.

The crosstabs were used to quantitatively analyze the relationship between feature and target variable.



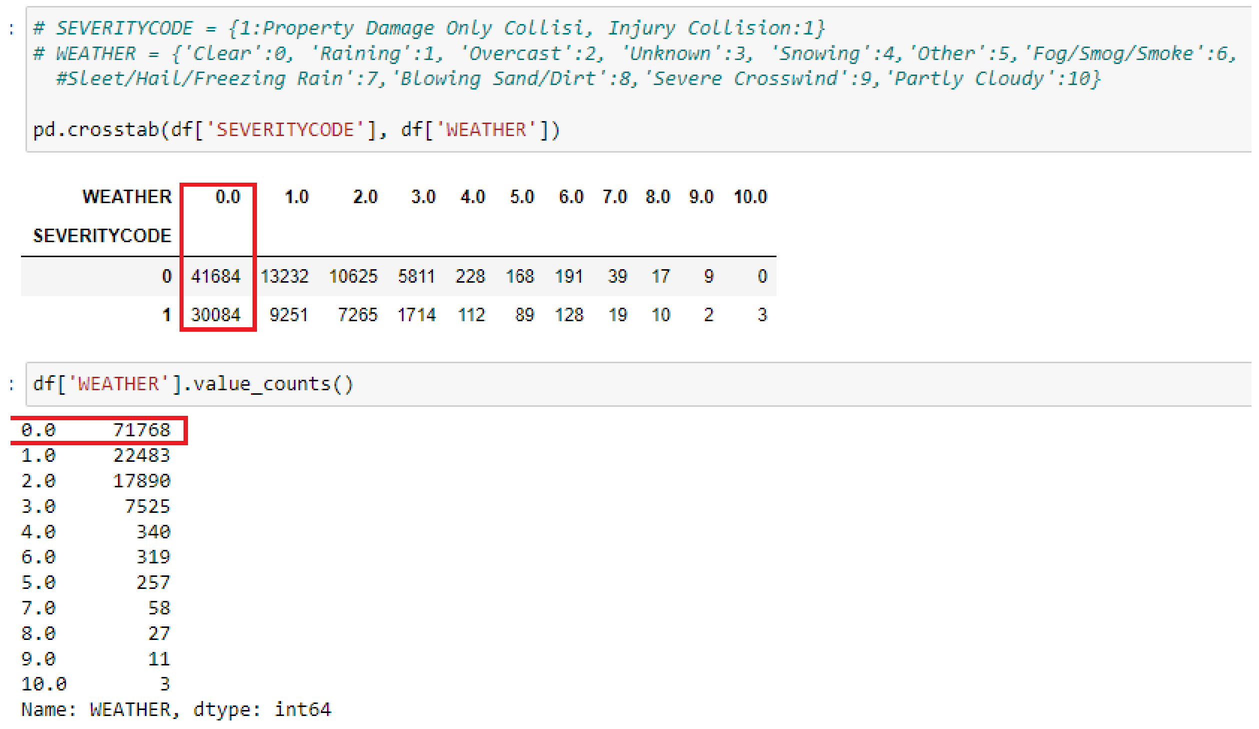
```
# SEVERITYCODE = {1:Property Damage Only Collisi, Injury Collision:1}
# ROADCOND = {'Dry':0,'Wet':1,Unknown':2,'Ice':3,'Snow/Slush':4,'Other':5,'Standing Water':6,'Sand/Mud/Dirt':7,'Oil':8}

pd.crosstab(df['SEVERITYCODE'], df['ROADCOND'])
```

ROADCOND	0.0	1.0	2.0	3.0	4.0	5.0	6.0	7.0	8.0
SEVERITYCODE									
0	46852	18782	5836	245	214	29	29	6	11
1	33678	13010	1664	160	111	19	15	11	9

```
df['ROADCOND'].value_counts()
```

```
0.0    80530
1.0    31792
2.0     7500
3.0      405
4.0      325
5.0       48
6.0       44
8.0       20
7.0       17
Name: ROADCOND, dtype: int64
```

Fig. 4 Crosstab: Relationship between 'SEVERITYDESC' and ' 'ROADCOND'



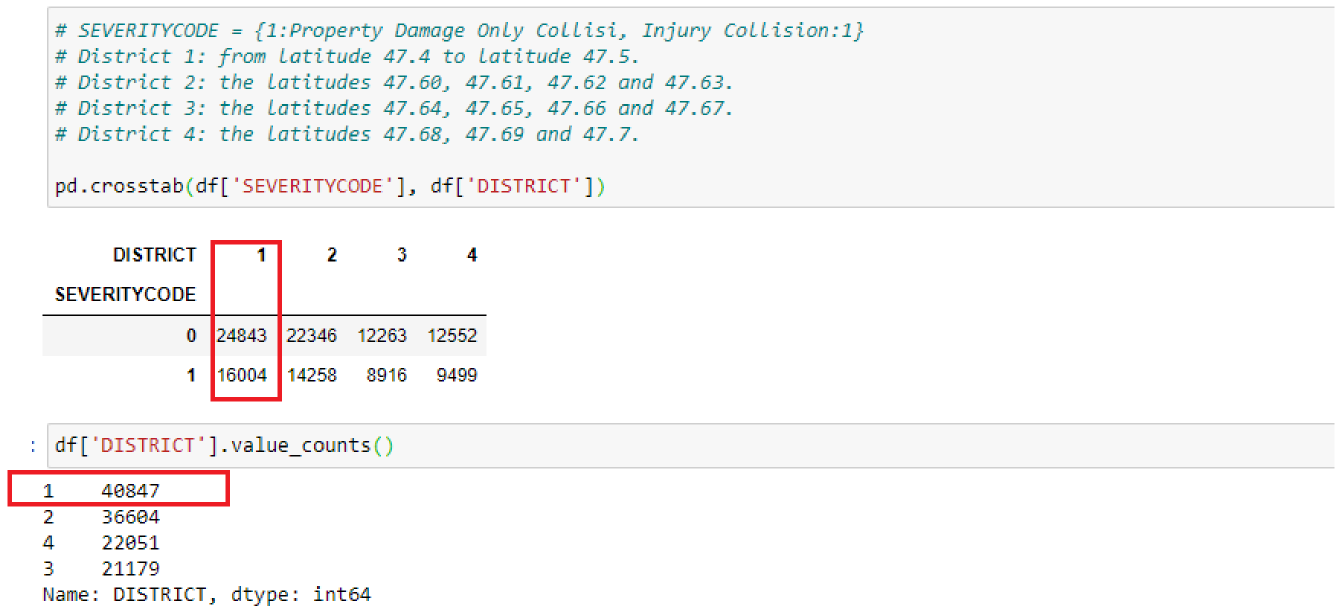
```
# SEVERITYCODE = {1:Property Damage Only Collisi, Injury Collision:1}
# WEATHER = {'Clear':0, 'Raining':1, 'Overcast':2, 'Unknown':3, 'Snowing':4,'Other':5,'Fog/Smog/Smoke':6,
  #Sleet/Hail/Freezing Rain':7,'Blowing Sand/Dirt':8,'Severe Crosswind':9,'Partly Cloudy':10}

pd.crosstab(df['SEVERITYCODE'], df['WEATHER'])
```

WEATHER	0.0	1.0	2.0	3.0	4.0	5.0	6.0	7.0	8.0	9.0	10.0
SEVERITYCODE											
0	41684	13232	10625	5811	228	168	191	39	17	9	0
1	30084	9251	7265	1714	112	89	128	19	10	2	3

```
df['WEATHER'].value_counts()
```

```
0.0     71768
1.0     22483
2.0     17890
3.0      7525
4.0       340
6.0       319
5.0       257
7.0        58
8.0        27
9.0        11
10.0        3
Name: WEATHER, dtype: int64
```

Fig. 5 Crosstab: Relationship between 'SEVERITYDESC' and ' 'WEATHER'



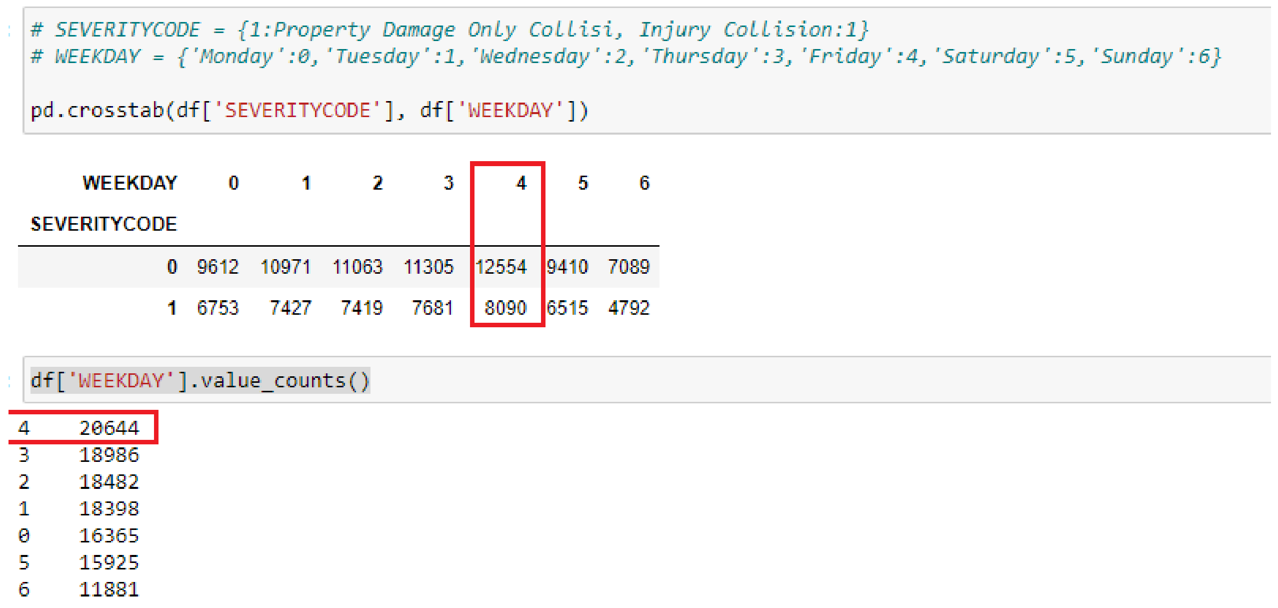
```
# SEVERITYCODE = {1:Property Damage Only Collisi, Injury Collision:1}
# District 1: from latitude 47.4 to latitude 47.5.
# District 2: the latitudes 47.60, 47.61, 47.62 and 47.63.
# District 3: the latitudes 47.64, 47.65, 47.66 and 47.67.
# District 4: the latitudes 47.68, 47.69 and 47.7.

pd.crosstab(df['SEVERITYCODE'], df['DISTRICT'])
```

DISTRICT	1	2	3	4
SEVERITYCODE				
0	24843	22346	12263	12552
1	16004	14258	8916	9499

```
df['DISTRICT'].value_counts()
```

```
1    40847
2    36604
4    22051
3    21179
Name: DISTRICT, dtype: int64
```

Fig. 6 Crosstab: Relationship between 'SEVERITYDESC' and ' 'DISTRICT'



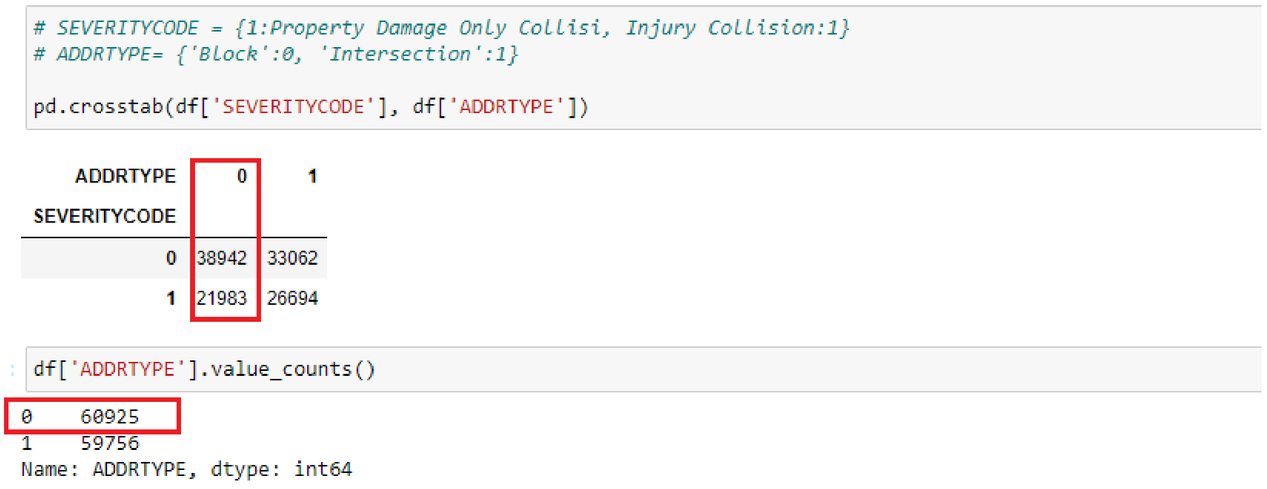
```
# SEVERITYCODE = {1:Property Damage Only Collisi, Injury Collision:1}
# WEEKDAY = {'Monday':0,'Tuesday':1,'Wednesday':2,'Thursday':3,'Friday':4,'Saturday':5,'Sunday':6}

pd.crosstab(df['SEVERITYCODE'], df['WEEKDAY'])
```

WEEKDAY	0	1	2	3	4	5	6
SEVERITYCODE							
0	9612	10971	11063	11305	12554	9410	7089
1	6753	7427	7419	7681	8090	6515	4792

```
df['WEEKDAY'].value_counts()
```

```
4     20644
3     18986
2     18482
1     18398
0     16365
5     15925
6     11881
```

Fig. 7 Crosstab: Relationship between 'SEVERITYDESC' and 'WEEKDAY'



```
# SEVERITYCODE = {1:Property Damage Only Collisi, Injury Collision:1}
# ADDRTYPE= {'Block':0, 'Intersection':1}

pd.crosstab(df['SEVERITYCODE'], df['ADDRTYPE'])
```

ADDRTYPE	0	1
SEVERITYCODE		
0	38942	33062
1	21983	26694

```
df['ADDRTYPE'].value_counts()
```

```
0     60925
1     59756
Name: ADDRTYPE, dtype: int64
```

Fig. 8 Crosstab: Relationship between 'SEVERITYDESC' and ADDRTYPE

Therefore, the following conclusions can be given:

(1) Friday is the day of the week with the highest risk of car crashes in Seattle.

(2) The clear weather and the dry road increase the risk of car crashes in Seattle.

(3) Areas / districts 1 (from latitude 47.4 to latitude 47.5) and 2 (latitudes 47.60, 47.61, 47.62 and 47.63.), are the places in Seattle with the highest risk of car crashes.

(4) The injury intersections and blocks have practically the same risk.

In addition, a heat map was created to identify the concentration of the vehicle collisions happened in 2004 to 2020.

Based on this heat map, we can clearly see the areas with the highest risk of car accidents, as well as the areas with the lowest accident rate. Thus, the blue areas indicate a low volume of collisions and the red areas a high volume.

The map shown in the figure below represents this.

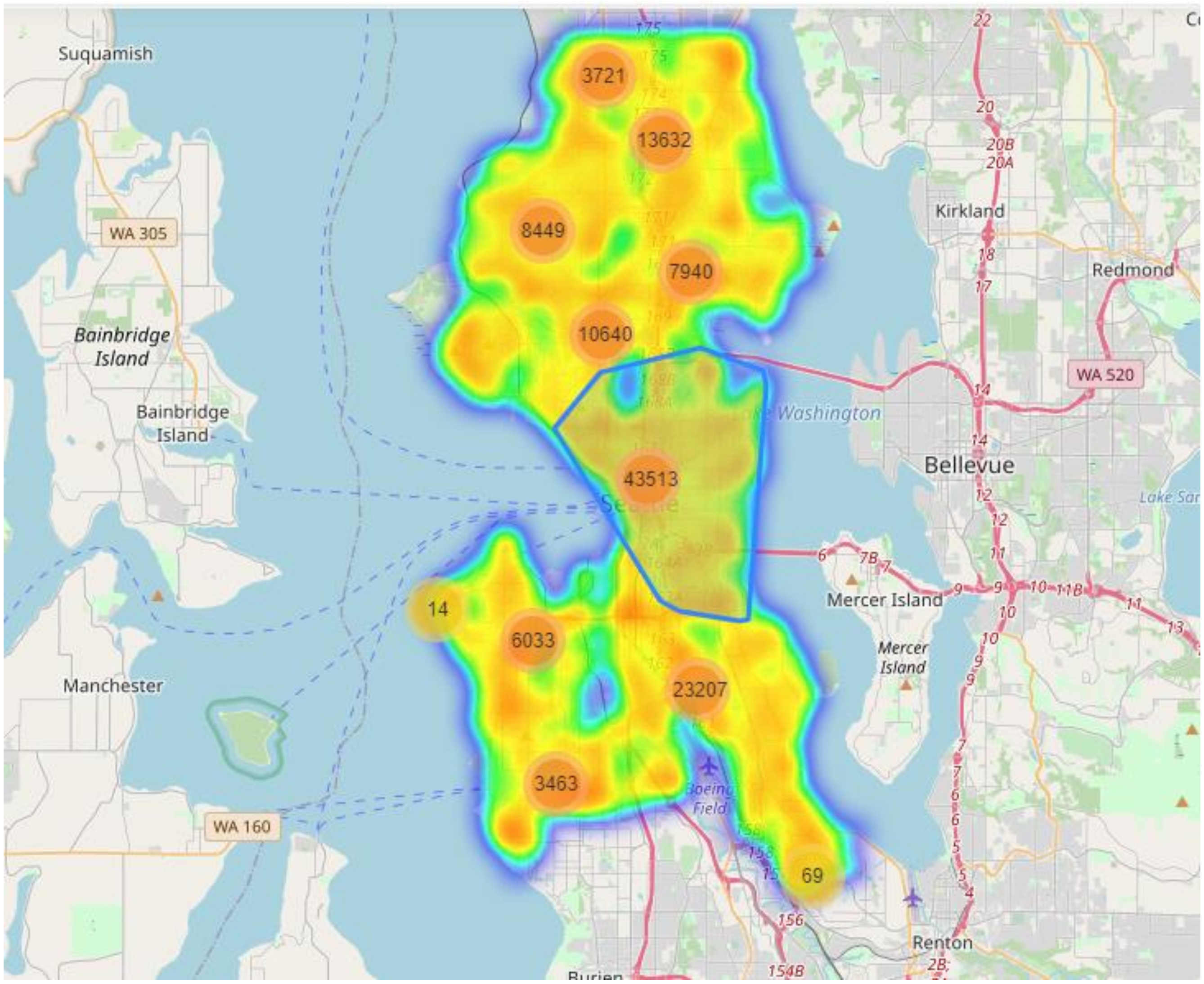


Fig. 9 Heat maps: the areas where the vehicle collisions happened in 2004 to 2020.

The scatter plot was used to show the relationship and correlation between the target variable “SEVERITYCODE” and predictors/ features.

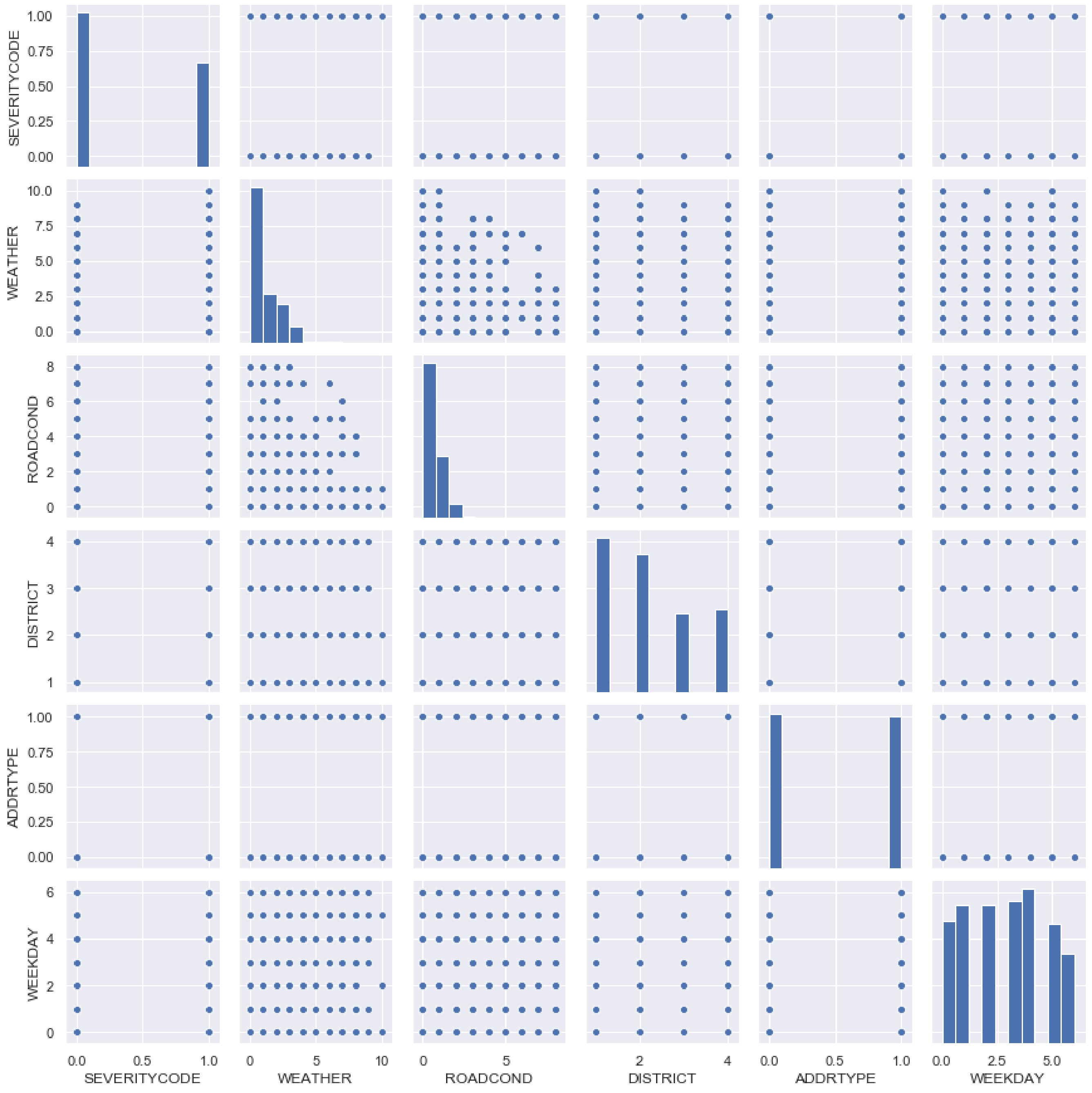


Fig. 10 Scatter plot

4. Predictive Modeling

In order to find the best model, the following algorithms were used:

- k-Nearest Neighbour
- Decision Tree
- Support Vector Machine
- Logistic Regression

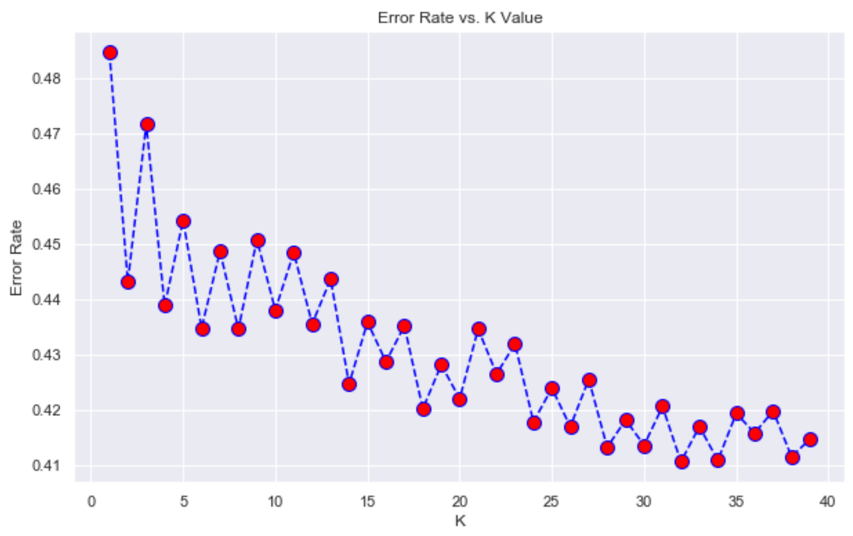
The results were reported as the accuracy of each classifier, using the following metrics when these are applicable:

- Jaccard index
- F1-score
- LogLoss.

4.1 K Nearest Neighbor (KNN)

```
plt.figure(figsize=(10,6))
plt.plot(range(1,40),error_rate,color='blue', linestyle='dashed', marker='o',
         markerfacecolor='red', markersize=10)
plt.title('Error Rate vs. K Value')
plt.xlabel('K')
plt.ylabel('Error Rate')
```

```
Text(0, 0.5, 'Error Rate')
```



```
CONFUSION_MATRIX :

[[9918 6317]
 [4563 3339]]


REPORT :

              precision    recall  f1-score   support

           0       0.68      0.61      0.65     16235
           1       0.35      0.42      0.38      7902

    accuracy                           0.55     24137
   macro avg       0.52      0.52      0.51     24137
weighted avg       0.57      0.55      0.56     24137
```

```
print('ACCURACY :')

metrics.accuracy_score(pred_i,y_test)
```

```
ACCURACY :

0.5492397563906036
```

4.2 Decision Tree

```
CONFUSION_MATRIX :

[[14013  9283]
 [  468   373]]
```

```
print('REPORT :\n')
print(classification_report(prediction,y_test))
```

```
REPORT :

              precision    recall  f1-score   support

           0       0.97      0.60      0.74     23296
           1       0.04      0.44      0.07       841

    accuracy                           0.60     24137
   macro avg       0.50      0.52      0.41     24137
weighted avg       0.94      0.60      0.72     24137
```

```
print('ACCURACY :\n')
metrics.accuracy_score(prediction,y_test)
```

```
ACCURACY :


0.5960144176989683
```

4.3 Support Vector Machine

```
print('CLASSIFICATION_REPORT :\n')
print(metrics.classification_report(y_pred,y_test))
```

```
CLASSIFICATION_REPORT :

              precision    recall  f1-score   support

           0       1.00      0.60      0.75     24134
           1       0.00      0.67      0.00         3

    accuracy                           0.60     24137
   macro avg       0.50      0.63      0.38     24137
weighted avg       1.00      0.60      0.75     24137
```

```
print('CONFUSION MATRIX :\n')
print(metrics.confusion_matrix(y_pred,y_test))
```

```
CONFUSION MATRIX :

[[14480  9654]
 [    1     2]]
```

```
print('ACCURACY :\n')
print(metrics.accuracy_score(y_pred,y_test))
```

```
ACCURACY :

0.5999917139661102
```

4.3 Logistic Regression

```
linear.fit(X_train,y_train)
```

```
LogisticRegression(C=1.0, class_weight=None, dual=False, fit_intercept=True,
                   intercept_scaling=1, l1_ratio=None, max_iter=100,
                   multi_class='auto', n_jobs=None, penalty='l2',
                   random_state=None, solver='lbfgs', tol=0.0001, verbose=0,
                   warm_start=False)
```

```
y_pred = linear.predict(X_test)
y_pred
```

```
array([0, 0, 0, ..., 0, 0, 0], dtype=int64)
```

4.4 Model Evaluation using Test set

```
print('CLASSIFICATION_REPORT :\n');
print(metrics.classification_report(y_pred,y_test))
```

```
CLASSIFICATION_REPORT :

              precision    recall  f1-score   support

           0       0.99      0.60      0.75     23968
           1       0.01      0.48      0.02       169

    accuracy                           0.60     24137
   macro avg       0.50      0.54      0.38     24137
weighted avg       0.99      0.60      0.74     24137
```

```
print('CONFUSION_MATRIX :\n');
print(metrics.confusion_matrix(y_pred,y_test))
```

```
CONFUSION_MATRIX :

[[14393  9575]
 [   88    81]]
```

```
print('ACCURACY_SCORE :\n');
print(metrics.accuracy_score(y_pred,y_test))
```

```
ACCURACY_SCORE :

0.599660272610515
```

4.5 Report

Accuracy of the built model using different evaluation metrics:

Algorithm	Jaccard	F1-score	LogLoss
KNN	0.549240	0.539595	NA
Decision Tree	0.596014	0.473522	NA
SVM	0.599992	0.450109	NA
Logistic Regression	0.599660	0.455767	0.666213

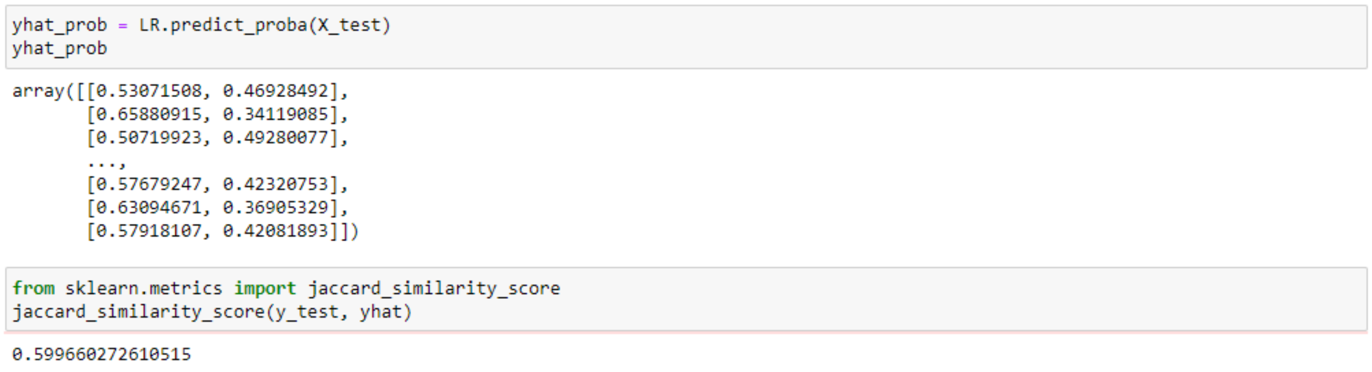
4. Results and Discussion

Based on the report, as seen before, the results show that the best model is logistic regression with log loss metric.

In addition, considering the size of the data set, + 100k, a logistic regression is more efficient in building a Machine Learning model.

4.1 Logistic Regression: Jaccard index

Lets try jaccard index for accuracy evaluation.



```
yhat_prob = LR.predict_proba(X_test)
yhat_prob
```

```
array([[0.53071508, 0.46928492],
       [0.65880915, 0.34119085],
       [0.50719923, 0.49280077],
       ...,
       [0.57679247, 0.42320753],
       [0.63094671, 0.36905329],
       [0.57918107, 0.42081893]])
```

```
from sklearn.metrics import jaccard_similarity_score
jaccard_similarity_score(y_test, yhat)
```

```
0.599660272610515
```

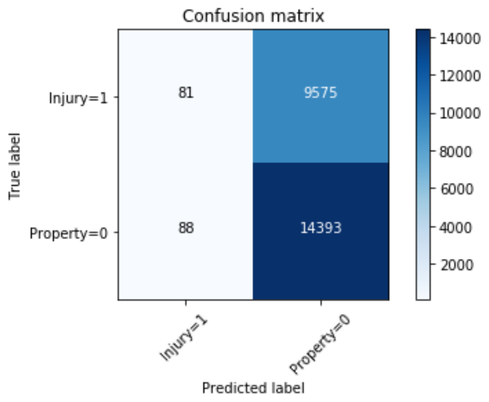
Now, let's take a look at confusion matrix structure.

```
print(confusion_matrix(y_test, yhat, labels=[1,0]))
```

```
[[   81  9575]
 [   88 14393]]
```

```
# Compute confusion matrix
cnf_matrix = confusion_matrix(y_test, yhat, labels=[1,0])
np.set_printoptions(precision=2)

# Plot non-normalized confusion matrix
plt.figure()
plot_confusion_matrix(cnf_matrix, classes=['Injury=1','Property=0'],normalize= False,  title='Confusion matrix')
```

```
Confusion matrix, without normalization
[[   81  9575]
 [   88 14393]]
```



4.2 Logistic Regression: log loss

Lets try log loss for accuracy evaluation.

```
from sklearn.metrics import log_loss
log_loss(y_test, yhat_prob)
```

```
0.6662216142597875
```

```
LR2 = LogisticRegression(C=0.01, solver='sag').fit(X_train,y_train)
yhat_prob2 = LR2.predict_proba(X_test)
print ("LogLoss: : %.2f" % log_loss(y_test, yhat_prob2))
```

```
LogLoss: : 0.67
```

5. Conclusion

Purpose of this project was to answer the question "In a given condition of road, climate, day of the week and an area of the city of Seattle, if an accident occurs, what would be the severity: an auto accident involving injury or property damage?", in order to aid authorities to reach decisions as regards the preventive safety measures to prevent accidents, such as changing road infrastructure, the procurement and installation of the road signs to reflect the weather conditions. Moreover, these informations could alert drivers, given the weather and road conditions about the possibility of a car accident and the severity of it, which would allow the driver to drive more carefully or, even change his itinerary.

By analyzing accidents distribution from data, the following conclusions can be made:

(1) Friday is the day of the week with the highest risk of car crashes in Seattle.

(2) The clear weather and the dry road increase the risk of car crashes in Seattle.

(3) Areas / districts 1 (from latitude 47.4 to latitude 47.5) and 2 (latitudes 47.60, 47.61, 47.62 and 47.63.), are the places in Seattle with the highest risk of car crashes.

(4) The injury intersections and blocks have practically the same risk.

By looking at confusion matrix, in the case of Logistic Regression with Jccard index, the first line refers to accidents whose value in the test set is 1, that is, "Injury". Thus, out of 24 137 accidents, the value (or type of collision) of 9 656 of them is 1. And of these 9656, the classifier correctly predicted 81 of them as 1 and 9575 of them as 0.

This means that, for 81 accidents, the real value was 1 in the test set, and the classifier also predicted those as 1. However, while the actual label of 9 656 customers was 1, the classifier predicted those as 0, which it's not very good. We can consider it an error of the model of the first line.

Accidents with a value of 0, ie "Property", correspond to the second line. It appears that there were 14 481 accidents with a value of 0. Where, the classifier correctly predicted 14 393 of them as 0 and 88 of them wrongly as 1. Therefore, he did a good job of predicting customers with a value of 0.

A good thing about the confusion matrix is that it shows the model's ability to correctly predict or separate classes. In the specific case of the binary classifier, as in this example, we can interpret these numbers as the count of true positives, false positives, true negatives and false negatives.