SEATTLE CITY – COLLISION LOG

CRIS-DM - REPORT

Erik Draganov Santos

1. Business Understanding:

I have decided to work on the example dataset provided by Seattle City Traffic Department, consisting of collision records registered since 2004. Collisions are classified by 37 features, including its severity degree, which can vary from 1 to 2. *The last degree is associated with injuries.*

Machine Learning algorithms can be extremely helpful in this scenario, which presents ***a usual classification problem***. By examining the variables associated with the incident and its respective severity code, the algorithm can indicate the contributing factors to more severe accidents.

The results may be useful to Seattle Authorities as a basis ***for developing public policies associated with road safet***y, or even identifying traffic areas in the city in ***need of maintenance or safety improvements***.

2. Data understanding:

The dataset is comprised of **194.673 records** in total. Besides the target variable, SEVERITYCODE, **there are 37 other columns**, which can be used as a feature to assess the severity of the accident. However, there are columns with missing or unregular data. Furthermore, the dataset is imbalanced, having **124.258 accidents associated with the code "01",** while **55.809 registers associated with the code "02".**

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

```
1    124258
2     55809
Name: SEVERITYCODE, dtype: int64
```

Regarding the features, there are columns specifying the conditions in which the accident occurred, as *weather, light and road conditions*. At first the model will be trained by using only external conditions variables as input, as intuitively they seem to contribute more to errors. Depending on the results, more features such as *collision type and location* will be included.

3. Data Preparation:

I will be conducting the following procedures in the data preparation stage:

a) **Balancing the dataset:** the first task to be completed in order to avoid biased models. An undersampling approach will be applied, reducing the

numbers of records in the majority class (01) instead of expanding the numbers of records in the minority class (02). Since the difference between each class is considerable, it is intuitively better to shuffle and reduce the data instead of expanding records based on faulty assumptions.

b) **Data cleaning:** the following task involves deleting columns without substantial data. If a column does not have at least 70% of useful data, it should be deleted. Rows with no information or data are to be deleted as well.

c) **Data transformation:** some columns require data transformation, since records are mixed, having for instance 1, 0, Y, N as records. A specific pattern should be applied, preferably 1,0.

```
Example:

df['UNDERINFL'] = df['UNDERINFL'].replace(['Y'],'1')
df['UNDERINFL'] = df['UNDERINFL'].replace(['N'],'0')
```

As indicated by the following summary, there are columns in the dataset with substantial lack of information, or even no information at all:

Missing Information Analysis (False = identifiable data | True = no data).

```
SEVERITYCODE
False    194673
Name: SEVERITYCODE, dtype: int64

X
False    189339
True       5334
Name: X, dtype: int64

Y
False    189339
True       5334
Name: Y, dtype: int64

OBJECTID
False    194673
Name: OBJECTID, dtype: int64

INCKEY
False    194673
Name: INCKEY, dtype: int64

COLDETKEY
False    194673
Name: COLDETKEY, dtype: int64

REPORTNO
False    194673
Name: REPORTNO, dtype: int64
```

```
STATUS
False    194673
Name: STATUS, dtype: int64


ADDRTYPE
False    192747
True       1926
Name: ADDRTYPE, dtype: int64

INTKEY
True     129603
False     65070
Name: INTKEY, dtype: int64

LOCATION
False    191996
True       2677
Name: LOCATION, dtype: int64

EXCEPTRSNCODE
True     109862
False     84811
Name: EXCEPTRSNCODE, dtype: int64

EXCEPTRSNDESC
True     189035
False      5638
Name: EXCEPTRSNDESC, dtype: int64

SEVERITYCODE.1
False    194673
Name: SEVERITYCODE.1, dtype: int64

SEVERITYDESC
False    194673
Name: SEVERITYDESC, dtype: int64

COLLISIONTYPE
False    189769
True       4904
Name: COLLISIONTYPE, dtype: int64

PERSONCOUNT
False    194673
Name: PERSONCOUNT, dtype: int64

PEDCOUNT
False    194673
Name: PEDCOUNT, dtype: int64

PEDCYLCOUNT
False    194673
Name: PEDCYLCOUNT, dtype: int64

VEHCOUNT
False    194673
```

```
Name: VEHCOUNT, dtype: int64

INCDATE
False    194673
Name: INCDATE, dtype: int64

INCDTTM
False    194673
Name: INCDTTM, dtype: int64

JUNCTIONTYPE
False    188344
True       6329
Name: JUNCTIONTYPE, dtype: int64

SDOT_COLCODE
False    194673
Name: SDOT_COLCODE, dtype: int64


SDOT_COLDESC
False    194673
Name: SDOT_COLDESC, dtype: int64

INATTENTIONIND
True     164868
False     29805
Name: INATTENTIONIND, dtype: int64

UNDERINFL
False    189789
True       4884
Name: UNDERINFL, dtype: int64

WEATHER
False    189592
True       5081
Name: WEATHER, dtype: int64

ROADCOND
False    189661
True       5012
Name: ROADCOND, dtype: int64

LIGHTCOND
False    189503
True       5170
Name: LIGHTCOND, dtype: int64

PEDROWNOTGRNT
True     190006
False      4667
Name: PEDROWNOTGRNT, dtype: int64

SDOTCOLNUM
False    114936
True      79737
Name: SDOTCOLNUM, dtype: int64
```

```
SPEEDING
True     185340
False      9333
Name: SPEEDING, dtype: int64

ST_COLCODE
False    194655
True         18
Name: ST_COLCODE, dtype: int64

ST_COLDESC
False    189769
True        4904
Name: ST_COLDESC, dtype: int64

SEGLANEKEY
False    194673
Name: SEGLANEKEY, dtype: int64

CROSSWALKKEY
False    194673
Name: CROSSWALKKEY, dtype: int64

HITPARKEDCAR
False    194673
Name: HITPARKEDCAR, dtype: int64
```

4. Modeling:

As mentioned, the problem under analysis herein is centered in classification. As such, the following algorithms and methods will be trained, assessed, and compared:

a) **KNeighborsClassifier:** initially the best “k” value should be identified, so the model can be trained accordingly.

b) **LogisticRegression:** basic algorithm.

c) **DecisionTreeClassifier:** preferably with a higher number of levels, due to the quantity of features (complexity).

d) **SVC – Support Vector Machines:** if possible, since the algorithm executes complex and hardware demanding calculations.

5. Evaluation:

The fours models will be compared in regard to their “Jaccard Index” and “F1-Scores”.

6. Deployment:

A final report will be developed, with the associated code published on GitHub.

7. Methodology and Discussion:

a) As specified above, the analysis started with cleaning related tasks, such as removing columns with irrelevant data or no information at all. Therefore, the following columns had empty rows deleted, or were completely dropped:

```
missing_data = df.isnull()
missing_data.head(5)

for column in missing_data.columns.values.tolist():
    print(column)
    print (missing_data[column].value_counts())
    print("")

# Simply drop whole row with NaN in "price" column
df.dropna(subset=["X"], axis=0, inplace=True)
df.dropna(subset=["COLLISIONTYPE"], axis=0, inplace=True)
df.dropna(subset=["JUNCTIONTYPE"], axis=0, inplace=True)
df.dropna(subset=["UNDERINFL"], axis=0, inplace=True)
df.dropna(subset=["WEATHER"], axis=0, inplace=True)
df.dropna(subset=["ROADCOND"], axis=0, inplace=True)
df.dropna(subset=["LIGHTCOND"], axis=0, inplace=True)

# Drop empty columns.
df.drop(['INATTENTIONIND', 'PEDROWNOTGRNT', 'SDOTCOLNUM', 'SPE
EDING', 'EXCEPTRSNDESC', 'INTKEY'], axis=1, inplace=True)
```

b) The data present in UNDERINFL had to be adjusted since there was a mix of 1, 0, Y and Ns. The value type of columns ST_COLCODE and UNDERINFL had to be adjusted as well.

```
df['UNDERINFL'] = df['UNDERINFL'].replace(['Y'],'1')
df['UNDERINFL'] = df['UNDERINFL'].replace(['N'],'0')
df.ST_COLCODE = df.ST_COLCODE.astype('int64')
df.UNDERINFL = df.UNDERINFL.astype('int64')
```

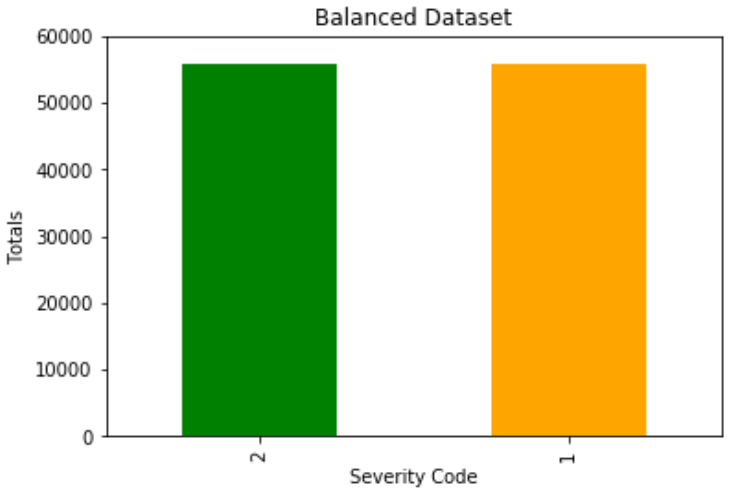
c) The dataset was extremely unbalanced, the majority class (SEVERITYCODE = 1) having more than twice the number of registers associated with the minority class (SEVERITYCODE = 2).

```
1    124258
2     55809
Name: SEVERITYCODE, dtype: int64
```

d) The dataset was then normalized, by applying an approach consisting of reducing the registers of the majority class.

```
# Shuffle the Dataset.
shuffled_df = df.sample(frac=1,random_state=4)


# SEVERITYCODE=2 in a separate dataset.
high_severity = shuffled_df.loc[shuffled_df['SEVERITYCODE'] ==
2]

#Randomly select 55809 observations from the majority class
low_severity = shuffled_df.loc[shuffled_df['SEVERITYCODE'] ==
1].sample(n=55809,random_state=42)

# Concatenate both dataframes again
normalized_df = pd.concat([high_severity, low_severity])

#plot the dataset after the undersampling
normalized_df['SEVERITYCODE'].value_counts(normalize=False).pl
ot(kind='bar', color=['green','orange'])
plt.ylim(0, 60000)
plt.title("Balanced Dataset")
plt.xlabel("Severity Code")
plt.ylabel("Totals")
```

```
Text(0, 0.5, 'Totals')
```




```
normalized_df['SEVERITYCODE'].value_counts()
2    55809
1    55809
Name: SEVERITYCODE, dtype: int64
```

e) I initially thought that external factors such as weather and light conditions were the most contributing factors to severe incidents. As such, I decided to use only features associated with a hypothetical difficult scenario confronting the driver:

```
Feature = normalized_df[['SEVERITYCODE']]
Feature = pd.concat([Feature,pd.get_dummies(normalized_df['WEA
THER'])], axis=1)
Feature = pd.concat([Feature,pd.get_dummies(normalized_df['ROA
DCOND'])], axis=1)
Feature = pd.concat([Feature,pd.get_dummies(normalized_df['LIG
HTCOND'])], axis=1)
Feature.drop(['SEVERITYCODE'], axis = 1,inplace=True)
Feature.head()
```

f) The whole dataset was divided in a train subset (20%) and a test subset (80%). Then, the following models were applied: logistic regression; K Neighbors and Decision Tree:

Logistic Regression:

```
LR Jaccard index:  0.5526625230975979
LR F1-score:  0.5233813041972027
LR log loss:  0.6728057120280866
```

K Neighbors Classifier:

```
KNN Jaccard index:  0.5509491012934655
KNN F1-score:  0.5257537777132462
```

Decision Tree:

```
DT Jaccard index:  0.5509267036228233
DT F1-score:  0.5184795246585846
```

g) The results were not satisfactory, the three algorithms reaching approximately 50% of accuracy. Those numbers indicated that selected variables were not as predictive as expected. Therefore, the variables **COLLISIONTYPE and SDOT_COLCODE** were added. This time, the algorithm was segmented on a different basis (20% test and 80% train).

Logistic Regression:

```
LR Jaccard index:  0.6958430388819208
LR F1-score:  0.6928812760928598
LR log loss:  0.5579978170956044
```

K Neighbors Classifier:

```
KNN Jaccard index:  0.688586274861136
KNN F1-score:  0.6856974458329252
```

Decision Tree:

```
DT Jaccard index:  0.6961118079197276
DT F1-score:  0.6940108307321592
```

h) The numbers presented a substantial increase in accuracy, reaching almost 70 percentage points. In order to verify how predictive those variables were, I decided to add more features to the model:

```
Feature = normalized_df[['SEVERITYCODE']]
Feature = pd.concat([Feature,normalized_df['PERSONCOUNT']], axis=1)
Feature = pd.concat([Feature,normalized_df['PEDCOUNT']], axis=1)
Feature = pd.concat([Feature,normalized_df['PEDCYLCOUNT']], axis=1)
Feature = pd.concat([Feature,normalized_df['VEHCOUNT']], axis=1)
Feature = pd.concat([Feature,pd.get_dummies(normalized_df['LOCATION'])], axis=1)
Feature = pd.concat([Feature,pd.get_dummies(normalized_df['ADDRTYPE'])], axis=1)
Feature = pd.concat([Feature,pd.get_dummies(normalized_df['COLLISIONTYPE'])], axis=1)
Feature = pd.concat([Feature,pd.get_dummies(normalized_df['WEATHER'])], axis=1)
Feature = pd.concat([Feature,pd.get_dummies(normalized_df['ROADCOND'])], axis=1)
Feature = pd.concat([Feature,pd.get_dummies(normalized_df['LIGHTCOND'])], axis=1)
Feature = pd.concat([Feature,pd.get_dummies(normalized_df['SDOT_COLCODE'])], axis=1)
Feature = pd.concat([Feature,pd.get_dummies(normalized_df['ST_COLCODE'])], axis=1)
Feature = pd.concat([Feature,pd.get_dummies(normalized_df['COLLISIONTYPE'])], axis=1)
Feature = pd.concat([Feature,pd.get_dummies(normalized_df['JUNCTIONTYPE'])], axis=1)
Feature = pd.concat([Feature,pd.get_dummies(normalized_df['HITPARKEDCAR'])], axis=1)
Feature = pd.concat([Feature,normalized_df['UNDERINFL']], axis=1)
Feature.drop(['SEVERITYCODE'], axis = 1,inplace=True)
Feature.head()
```

Logistic Regression:

```
LR Jaccard index:  0.7102413349011703
LR F1-score:  0.7081211777691742
LR log loss:  0.5398568538465066
```

SVC (Used this time instead of K Nearest Neighbors:

```
SVM Jaccard index:  0.708617503779607
SVM F1-score:  0.70662254199849
```

Decision Tree:

```
DT Jaccard index:  0.7136713850564415
DT F1-score:  0.7099078986561742
```

i) There was a slightly increase of accuracy, not substantial when comparing to the second feature group, consisting of climate conditions associated with the type collision. However, I believe the type of collision was the feature which contributed more to the results, since the most severe code is associated with rear ended crashes, or accidents involving pedestrian and cycles, categories with less incidence in accidents with no injuries involved:

```
normalized_df.groupby(['SEVERITYCODE'])['COLLISIONTYPE'].value
_counts(normalize=True)
```

```
              SEVERITYCODE  COLLISIONTYPE
1               Parked Car       0.321525
                Angles           0.168414
                Rear Ended       0.151983
                Other            0.130230
                Sideswipe        0.124854
                Left Turn        0.065043
                Right Turn       0.018151
                Head On          0.009102
                Cycles           0.005429
                Pedestrian       0.005268
2               Rear Ended       0.253830
                Angles           0.242649
                Pedestrian       0.104302
                Other            0.102761
                Left Turn        0.096275
                Cycles           0.083822
                Parked Car       0.047018
                Sideswipe        0.043273
                Head On          0.015338
                Right Turn       0.010733
Name: COLLISIONTYPE, dtype: float64
```

j) Furthermore, all categories of crashes seem to have happened on very similar conditions:

```
COLLISIONTYPE  WEATHER
Angles         Clear                       0.613618
               Raining                     0.204917
               Overcast                    0.160411
               Unknown                     0.012292
               Snowing                     0.003095
               Fog/Smog/Smoke              0.002964
               Other                       0.001744
               Sleet/Hail/Freezing Rain    0.000567
               Blowing Sand/Dirt           0.000218
               Severe Crosswind            0.000131
               Partly Cloudy               0.000044
Cycles         Clear                       0.725758
               Overcast                    0.147360
               Raining                     0.106605
               Unknown                     0.017466
               Other                       0.001205
               Fog/Smog/Smoke              0.001004
               Snowing                     0.000402
```

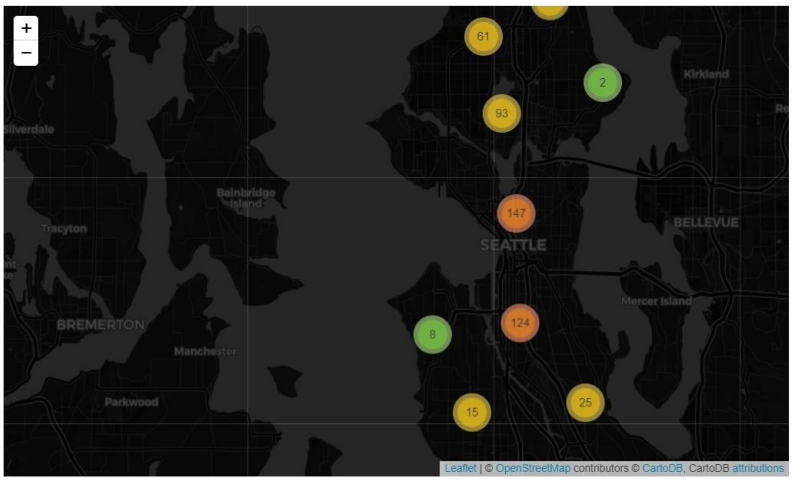
	Blowing Sand/Dirt	0.000201
Head On	Clear	0.522727
	Raining	0.272727
	Overcast	0.162757
	Snowing	0.016862
	Unknown	0.016129
	Fog/Smog/Smoke	0.003666
	Sleet/Hail/Freezing Rain	0.002933
	Other	0.002199
Left Turn	Clear	0.632012
	Raining	0.197157
	Overcast	0.152616
	Unknown	0.011885
	Fog/Smog/Smoke	0.002444
	Snowing	0.001666
	Other	0.001555
	Sleet/Hail/Freezing Rain	0.000333
	Blowing Sand/Dirt	0.000222
	Partly Cloudy	0.000111
Other	Clear	0.576559
	Raining	0.214873
	Overcast	0.165423
	Unknown	0.023687
	Snowing	0.008690
	Fog/Smog/Smoke	0.006229
	Other	0.002307
	Sleet/Hail/Freezing Rain	0.001461
	Severe Crosswind	0.000538
	Blowing Sand/Dirt	0.000154
	Partly Cloudy	0.000077
Parked Car	Clear	0.556690
	Unknown	0.186989
	Overcast	0.126070
	Raining	0.110268
	Other	0.009286
	Snowing	0.006709
	Fog/Smog/Smoke	0.002820
	Blowing Sand/Dirt	0.000583
	Sleet/Hail/Freezing Rain	0.000535
	Partly Cloudy	0.000049
Pedestrian	Clear	0.577269
	Raining	0.243009
	Overcast	0.144235
	Unknown	0.025348
	Fog/Smog/Smoke	0.003107
	Other	0.002944
	Snowing	0.002944
	Sleet/Hail/Freezing Rain	0.000981
	Blowing Sand/Dirt	0.000164
Rear Ended	Clear	0.614447
	Raining	0.201386

```
                Overcast                    0.150521
                Unknown                     0.024329
                Fog/Smog/Smoke              0.003223
                Snowing                     0.002605
                Other                       0.002517
                Sleet/Hail/Freezing Rain    0.000530
                Blowing Sand/Dirt           0.000309
                Severe Crosswind            0.000132
Right Turn      Clear                       0.650744
                Raining                     0.161290
                Overcast                    0.147643
                Unknown                     0.030397
                Fog/Smog/Smoke              0.003722
                Snowing                     0.003102
                Other                       0.001861
                Blowing Sand/Dirt           0.001241
Sideswipe       Clear                       0.647234
                Raining                     0.159650
                Overcast                    0.152190
                Unknown                     0.032719
                Other                       0.002984
                Snowing                     0.002345
                Fog/Smog/Smoke              0.002132
                Blowing Sand/Dirt           0.000320
                Sleet/Hail/Freezing Rain    0.000320
                Partly Cloudy               0.000107
Name: WEATHER, dtype: float64
```

j) In order to verify whether more severe accidents were happening in a specific neighborhood in Seattle, I decided to add markers to the city map on folium, indicating Low Severity accidents (01) and High Severity accidents (02). From the evidence, we may conclude that both categories are evenly distributed, being the neighborhood not a crucial factor for the severity of an accident.

Low Severity Accidents:

High Severity Accidents:



8. Discussion and Conclusion:

From the data above we may conclude that the type of collision is more important to the severity of the accident than other variables such as weather, road or light conditions.

Rear Ended	0.253830
Angles	0.242649
Pedestrian	0.104302
Other	0.102761
Left Turn	0.096275
Cycles	0.083822

Based in the proportion of collision types among the high severity ones, it seems the that driver inattention, especially driver distraction, could be an extremely influential factor contributing to serious accidents. Maybe the city traffic authority can work on policies to address the issue. Accidents involving pedestrians and cycles are also representative when comparison to lower severity incidents. Factors contributing to these differences are not clear, but some issues may be implied, such as: deteriorated sidewalks, force the pedestrians to walk on the streets; bad or inexistent signaling; no cycle tracks. Those items must be verified in details on order to assess their contribution; however the results offers good evidence to the cause of the problem.