Capstone Project – Car accident severity

Elida Axundzada

November 11, 2020

Introduction

As a driving school, we would like to create a mobile application for young drivers. The purpose of this application is to increase the safety on the road based on various criteria. The young drivers could verify the risk of accident before hitting the road. To create our application, we should develop an accurate prediction model.

We will use various variables to train our machine learning model. Indeed, we should verify the correlation between the car accident severity and the chosen variables before making our application available.

Data acquisition

Data source

The city of Seattle has a dataset on all collisions collected since 2004 to present, as provided by the Seattle Police District and recorded by Traffic Records. It has 194,673 observations with 37 attributes. It has been downloaded as a csv file – DataCollisions.csv from this link: https://s3.us.cloud-object-storage.appdomain.cloud/cf-courses-data/CognitiveClass/DP0701EN/version-2/Data-Collisions.csv

The description of the attributes is below:

Attribute	**Data Type**	**Data Length**	**Description**
ADDRTYPE	Text	12	Collision address type: Alley, Block, Intersection
COLDETKEY	Long		Secondary key for the incident.
COLLISIONTYPE	Text	300	Collision type.
CROSSWALKKEY	Long		A key for the crosswalk at which the collision occurred.
EXCEPTRSNCODE	Text	10	
EXCEPTRSNDESC	Text	300	
FATALITIES	Double		The number of fatalities in the collision. This is entered by the state.
HITPARKEDCAR	Text	1	Whether or not the collision involved hitting a parked car. (Y/N)
INATTENTIONIND	Text	1	Whether or not collision was due to inattention. (Y/N)
INCDATE	Date		The date of the incident.

Attribute	**Data Type**	**Data Length**	**Description**
INCDTTM	Text	30	The date and time of the incident.
INCKEY	Long		A unique key for the incident.
INJURIES	Double		The number of total injuries in the collision. This is entered by the state.
INTKEY	Double		Key that corresponds to the intersection associated with a collision.
JUNCTIONTYPE	Text	300	Category of junction at which collision took place.
LIGHTCOND	Text	300	The light conditions during the collision.
LOCATION	Text	255	Description of the general location of the collision.
OBJECTID	ObjectID		ESRI unique identifier.
PEDCOUNT	Double		The number of pedestrians involved in the collision. This is entered by the state.
PEDCYLCOUNT	Double		The number of bicycles involved in the collision. This is entered by the state.
PEDROWNOTGRNT	Text	1	Whether or not the pedestrian right of way was not granted. (Y/N)
PERSONCOUNT	Double		The total number of people involved in the collision.
ROADCOND	Text	300	The condition of the road during the collision.
SDOT_COLCODE	Text	10	A code given to the collision by SDOT.
SDOT_COLDESC	Text	300	A description of the collision corresponding to the collision code.
SDOTCOLNUM	Text	10	A number given to the collision by SDOT.
SEGLANEKEY	Long		A key for the lane segment in which the collision occurred.
SERIOUSINJURIES	Double		The number of serious injuries in the collision. This is entered by the state.

Attribute	**Data Type**	**Data Length**	**Description**
SEVERITYCODE	Text	100	A code that corresponds to the severity of the collision: [3 – fatality, 2b - serious injury, 2 – injury, 1 – prop, damage, 0 – unknown]
SEVERITYDESC	Text		A detailed description of the severity of the collision.
SHAPE	Geometry		ESRI geometry field.
SPEEDING	Text	1	Whether or not speeding was a factor in the collision. (Y/N)
ST_COLCODE	Text	10	A code provided by the state that describes the collision. For more information about these codes, please see the State Collision Code Dictionary.
ST_COLDESC	Text	300	A description that corresponds to the state's coding designation.
UNDERINFL	Text	10	Whether or not a driver involved was under the influence of drugs or alcohol.
VEHCOUNT	Double		The number of vehicles involved in the collision. This is entered by the state.
WEATHER	Text	300	A description of the weather conditions during the time of the collision.

Feature selection

For our analyse, to help in predicting the possibility and severity of an accident/collision {SEVERITYCODE}, the following attributes will be used:

- {ADDRTYPE}: collision address type as source of location information
- {WEATHER}: weather during the time of collision
- {LIGHTCOND}: light conditions during the collision
- {ROADCOND}: road condition during the collision

Methodology

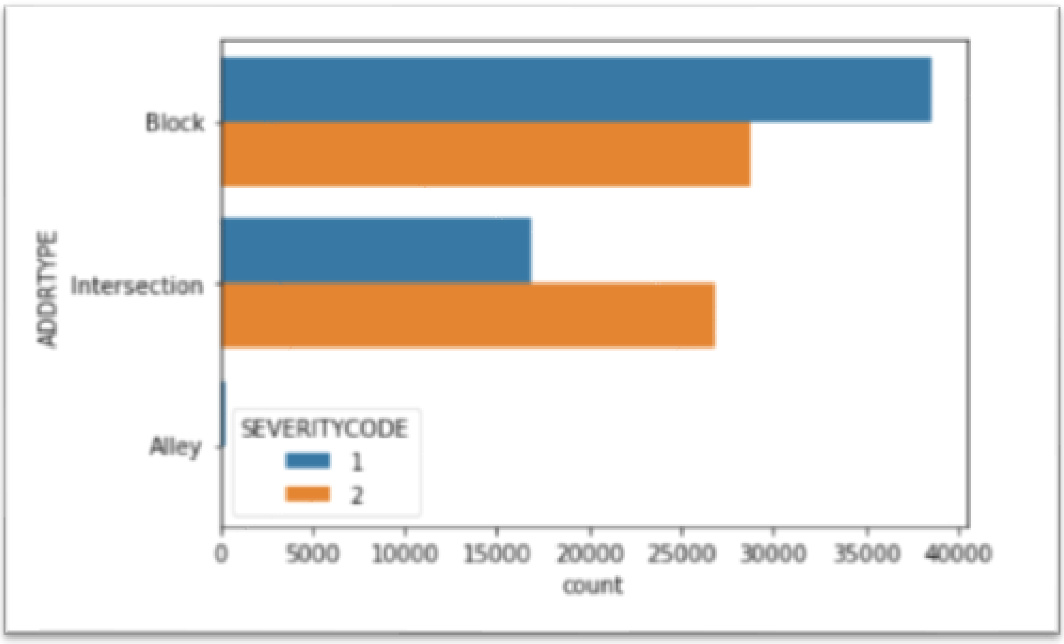
Data cleaning

For the chosen features, there are NaN values, they were dropped for better processing of sklearn. Moreover, there are unknown data on WEATHER, LIGHTCOND and ROADCOND, and these were dropped as well.

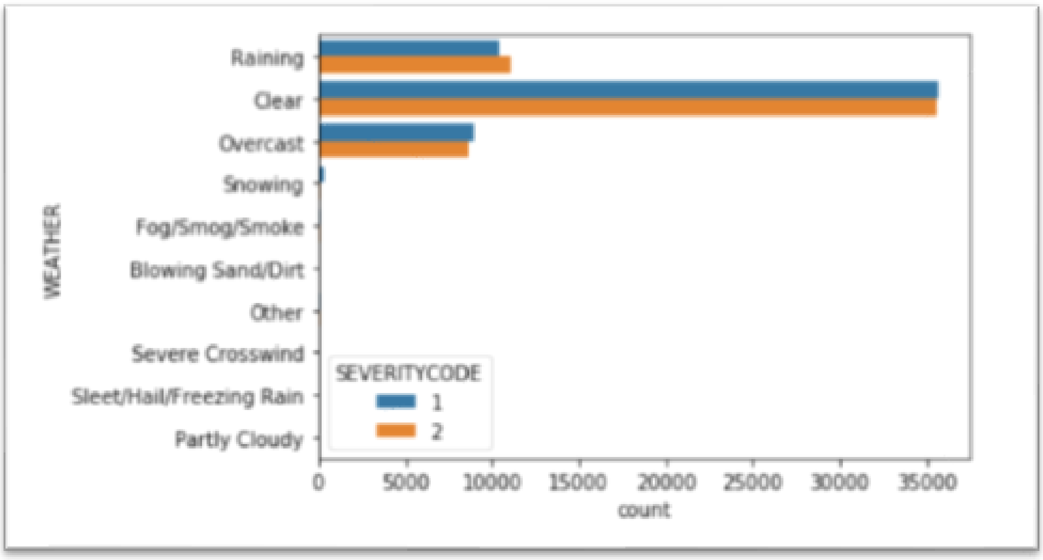
Exploratory analysis

Separate exploratory analysis is done on the independent variables.

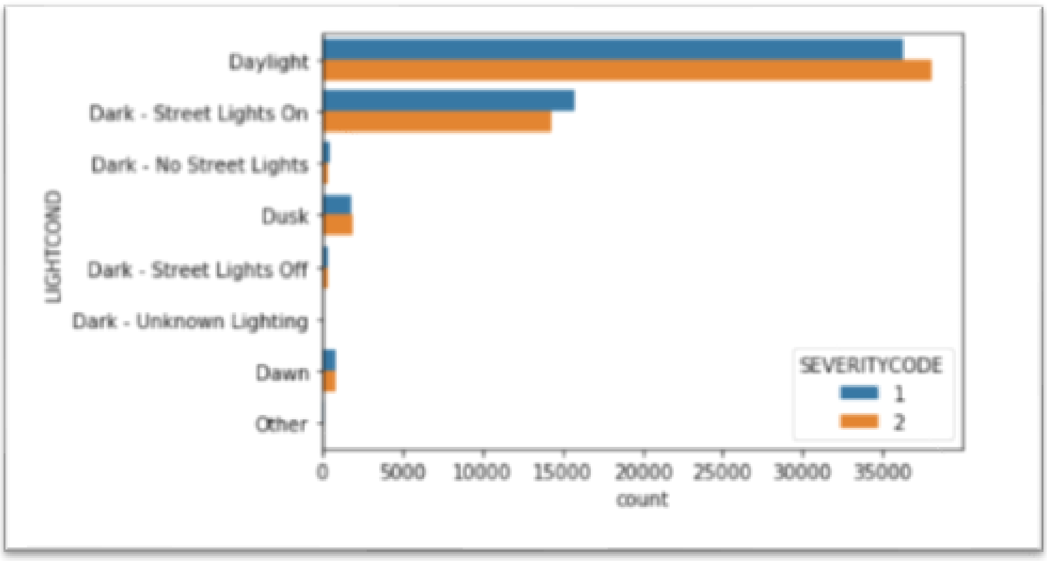
- Severity Code and Address Type: Severity Code 1 collisions along Blocks are more than on Intersections. The collisions of Severity Code 2 are almost the same on both Blocks and Intersections.



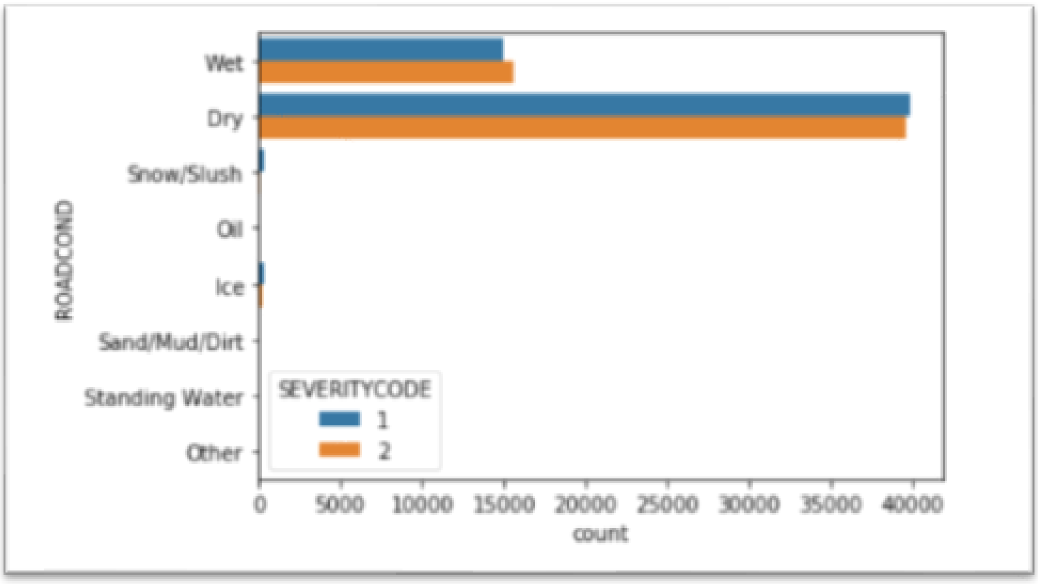
- Severity Code and Weather: most number of collisions happen on Clear weather.



➢ Severity Code and Light Conditions: most number of collisions happen on daylight.

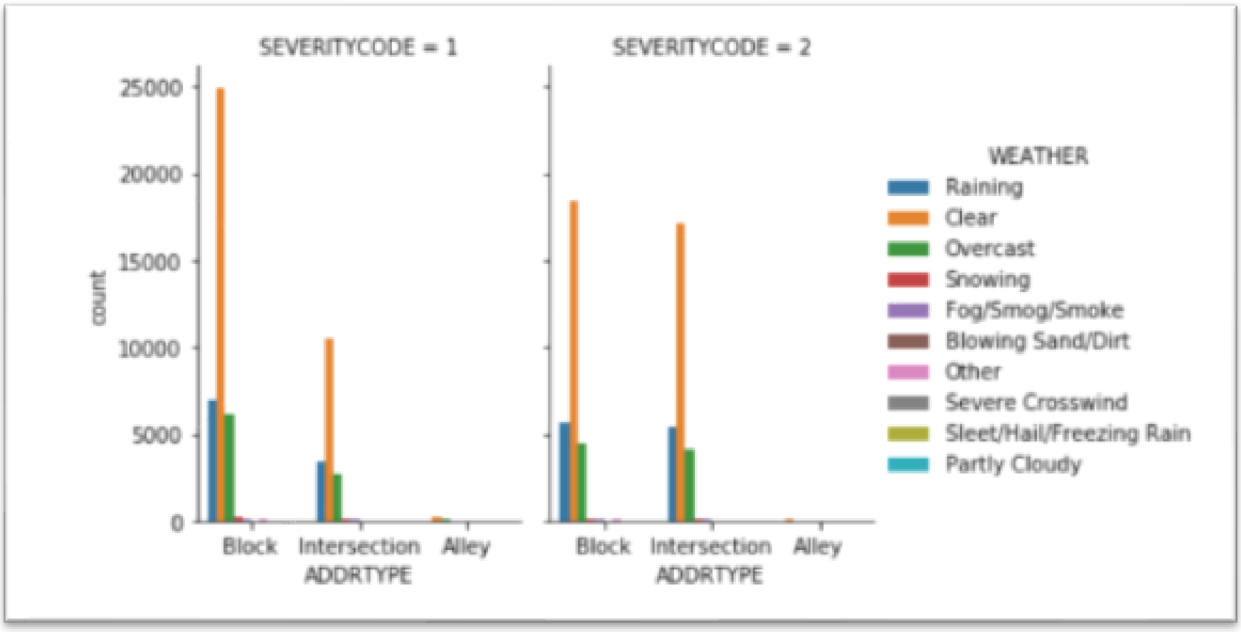


➢ Severity Code and Road Conditions: most number of collisions happen on dry roads.

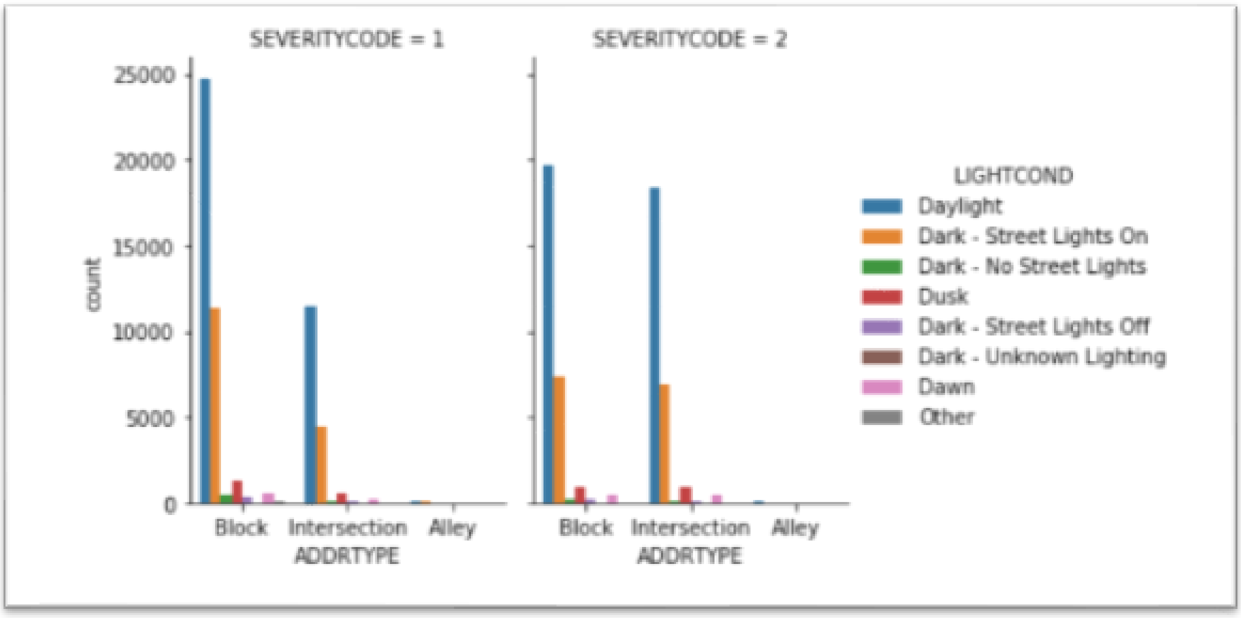


Correlations between variables are observed.

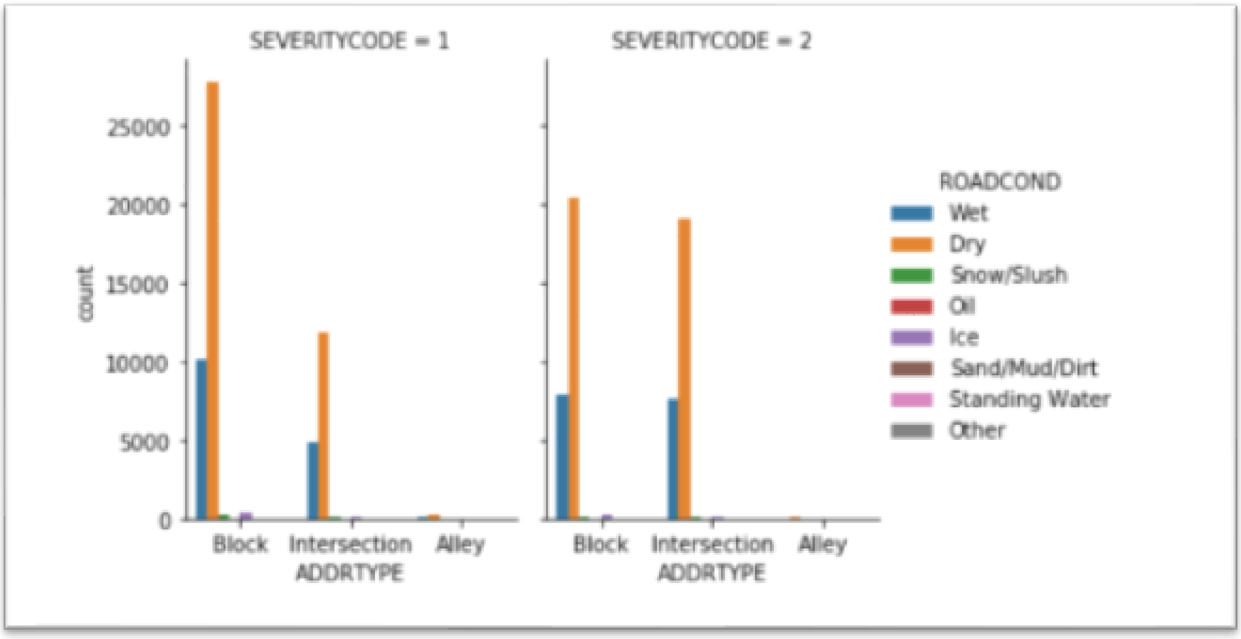
- ➢ Severity Code, Address type and weather: most number of collisions happen on blocks and on a clear weather.



- ➢ Severity Code, Address type and light conditions: most number of collisions happen daylight, mostly on blocks.



- Severity Code, Address type and road conditions: most number of collisions happen on dry roads, mostly on blocks.

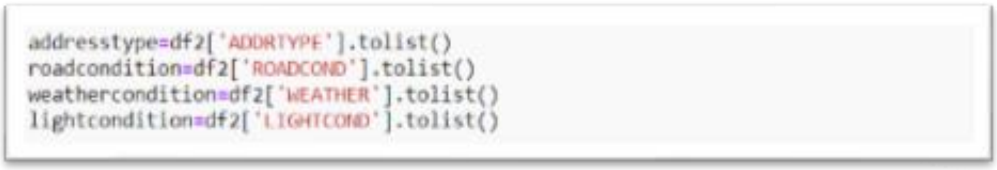


Machine learning model: decision tree

3.3.1 One-hot encoding

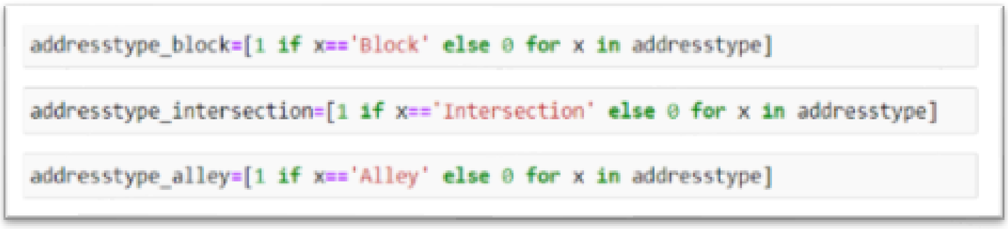
All selected independent variables are transformed to the format that can be fit to the machine learning model by using one-hot encoding. Indeed, the one-hot encoding help us to convert categorial features into numerical, that works better with classification and regressions algorithms. It is done by "creating one Boolean column for each of our given categories, where only one of these columns could take on the 1 for each sample".

- Each independent variable is transformed to a list

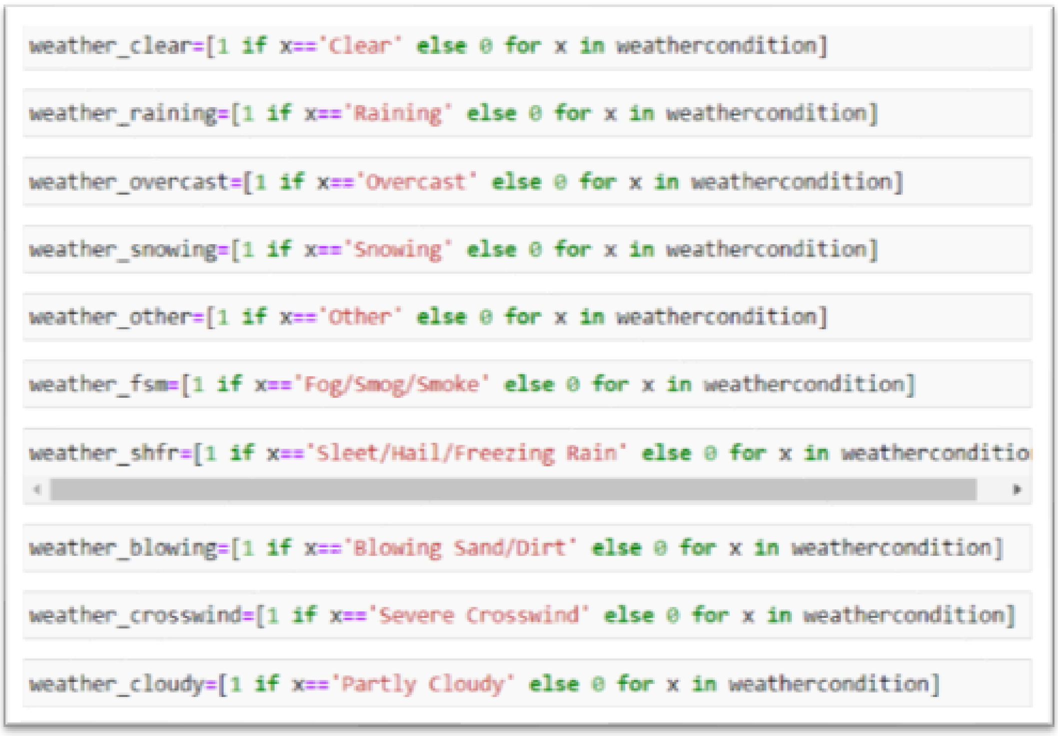


```
addresstype=df2['ADDRTYPE'].tolist()
roadcondition=df2['ROADCOND'].tolist()
weathercondition=df2['WEATHER'].tolist()
lightcondition=df2['LIGHTCOND'].tolist()
```

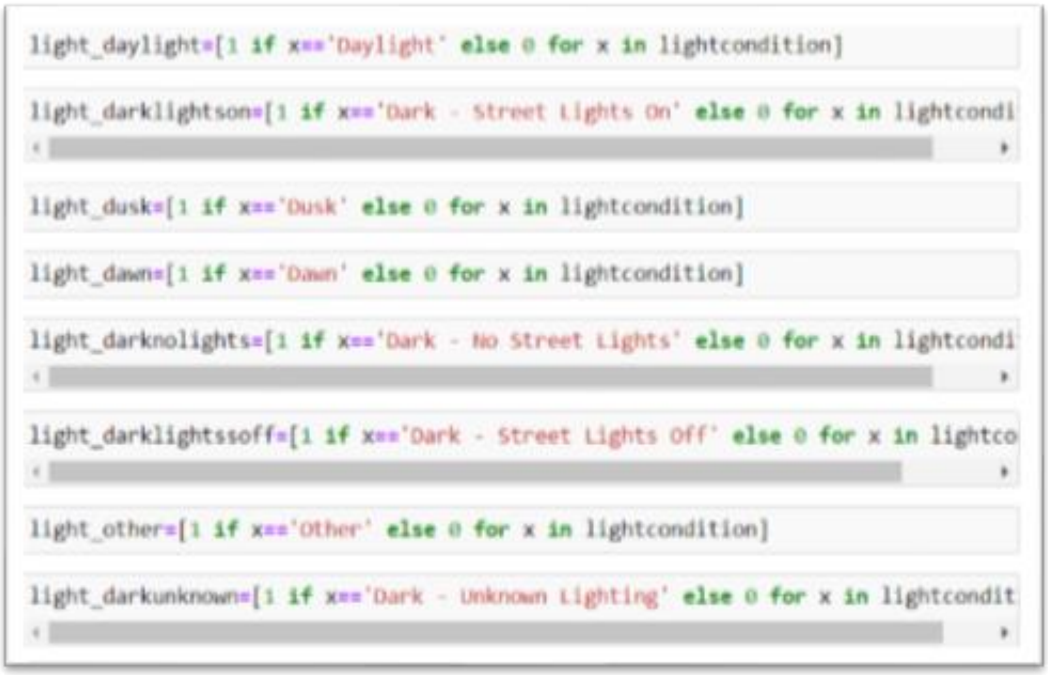
- One-hot encoding for each categorical feature



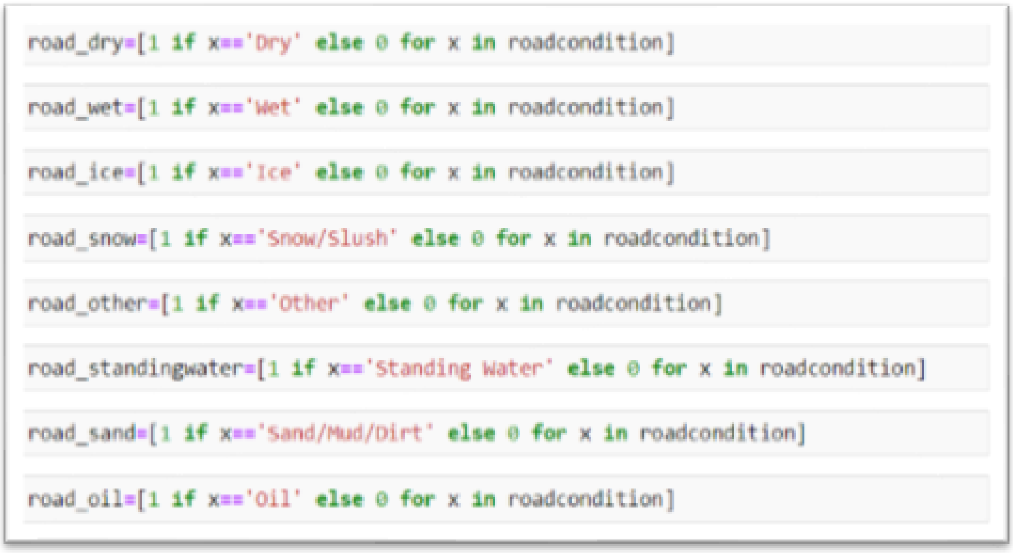
```
addresstype_block=[1 if x=='Block' else 0 for x in addresstype]
```
```
addresstype_intersection=[1 if x=='Intersection' else 0 for x in addresstype]
```
```
addresstype_alley=[1 if x=='Alley' else 0 for x in addresstype]
```



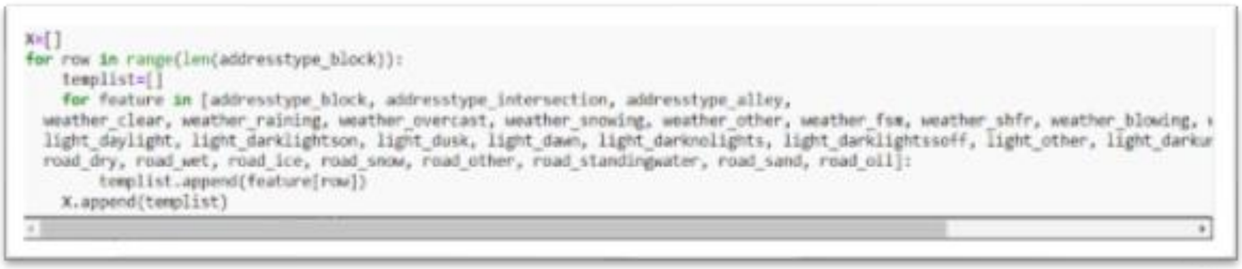
```
weather_clear=[1 if x=='Clear' else 0 for x in weathercondition]
```
```
weather_raining=[1 if x=='Raining' else 0 for x in weathercondition]
```
```
weather_overcast=[1 if x=='Overcast' else 0 for x in weathercondition]
```
```
weather_snowing=[1 if x=='Snowing' else 0 for x in weathercondition]
```
```
weather_other=[1 if x=='Other' else 0 for x in weathercondition]
```
```
weather_fsm=[1 if x=='Fog/Smog/Smoke' else 0 for x in weathercondition]
```
```
weather_shfr=[1 if x=='Sleet/Hail/Freezing Rain' else 0 for x in weatherconditio
```
```
weather_blowing=[1 if x=='Blowing Sand/Dirt' else 0 for x in weathercondition]
```
```
weather_crosswind=[1 if x=='Severe Crosswind' else 0 for x in weathercondition]
```
```
weather_cloudy=[1 if x=='Partly Cloudy' else 0 for x in weathercondition]
```



```
light_daylight=[1 if x=='Daylight' else 0 for x in lightcondition]

light_darklightson=[1 if x=='Dark - Street Lights On' else 0 for x in lightcondi

light_dusk=[1 if x=='Dusk' else 0 for x in lightcondition]

light_dawn=[1 if x=='Dawn' else 0 for x in lightcondition]

light_darknolights=[1 if x=='Dark - No Street Lights' else 0 for x in lightcondi

light_darklightssoff=[1 if x=='Dark - Street Lights Off' else 0 for x in lightco

light_other=[1 if x=='Other' else 0 for x in lightcondition]

light_darkunknown=[1 if x=='Dark - Unknown Lighting' else 0 for x in lightcondit
```



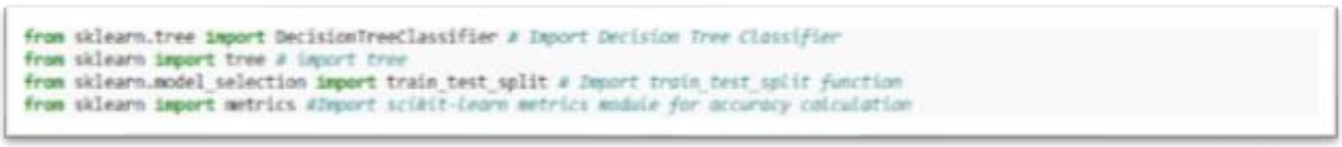
```
road_dry=[1 if x=='Dry' else 0 for x in roadcondition]

road_wet=[1 if x=='Wet' else 0 for x in roadcondition]

road_ice=[1 if x=='Ice' else 0 for x in roadcondition]

road_snow=[1 if x=='Snow/Slush' else 0 for x in roadcondition]

road_other=[1 if x=='Other' else 0 for x in roadcondition]

road_standingwater=[1 if x=='Standing Water' else 0 for x in roadcondition]

road_sand=[1 if x=='Sand/Mud/Dirt' else 0 for x in roadcondition]

road_oil=[1 if x=='Oil' else 0 for x in roadcondition]
```



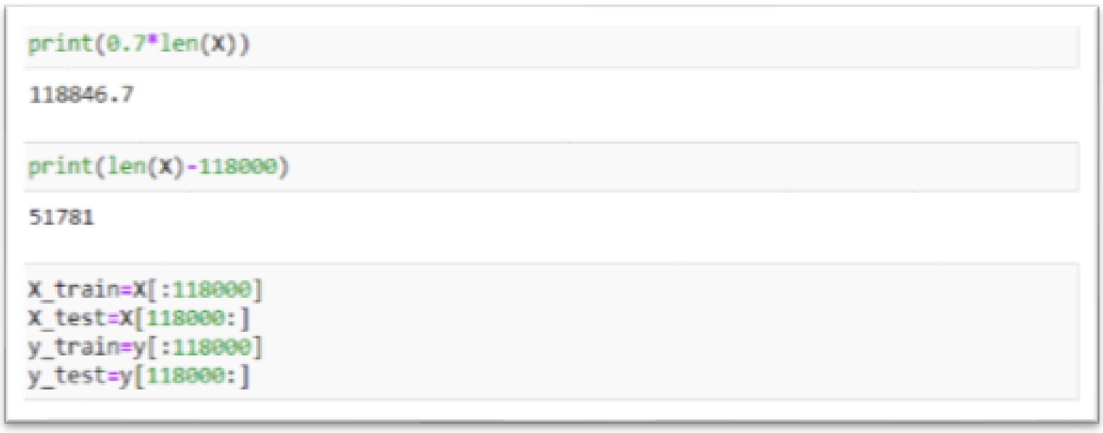
```
X=[]
for row in range(len(addresstype_block)):
    templist=[]
    for feature in [addresstype_block, addresstype_intersection, addresstype_alley,
  weather_clear, weather_raining, weather_overcast, weather_snowing, weather_other, weather_fsm, weather_shfr, weather_blowing, 
  light_daylight, light_darklightson, light_dusk, light_dawn, light_darknolights, light_darklightssoff, light_other, light_darku
  road_dry, road_wet, road_ice, road_snow, road_other, road_standingwater, road_sand, road_oil]:
        templist.append(feature[row])
    X.append(templist)
```

3.3.2 Data split

We are divided the dataset into a training set and a test set for better understanding.



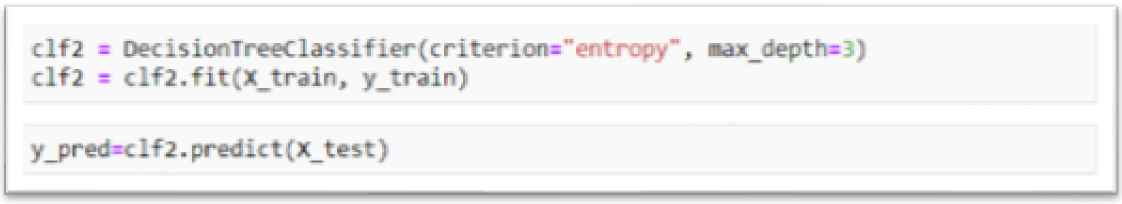
```
from sklearn.tree import DecisionTreeClassifier # Import Decision Tree Classifier
from sklearn import tree # import tree
from sklearn.model_selection import train_test_split # Import train_test_split function
from sklearn import metrics #Import scikit-learn metrics module for accuracy calculation
```



```
print(0.7*len(X))

118846.7

print(len(X)-118000)

51781

X_train=X[:118000]
X_test=X[118000:]
y_train=y[:118000]
y_test=y[118000:]
```

3.3.3 Building Decision Tree Model

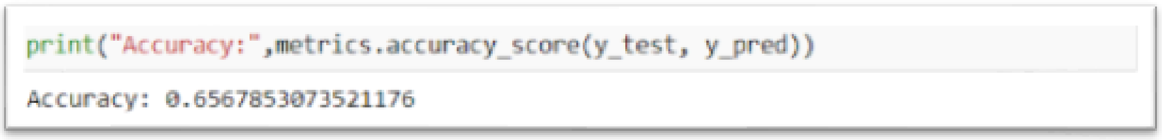
Then, we are created the Decision Tree Model using Scikit-learn. We have used the criterion *entropy* and *max_depth* equals to 3.



```
clf2 = DecisionTreeClassifier(criterion="entropy", max_depth=3)
clf2 = clf2.fit(X_train, y_train)

y_pred=clf2.predict(X_test)
```

3.3.4 Model evaluation

We are compared the actual test set values and the predicted values to evaluate the model.

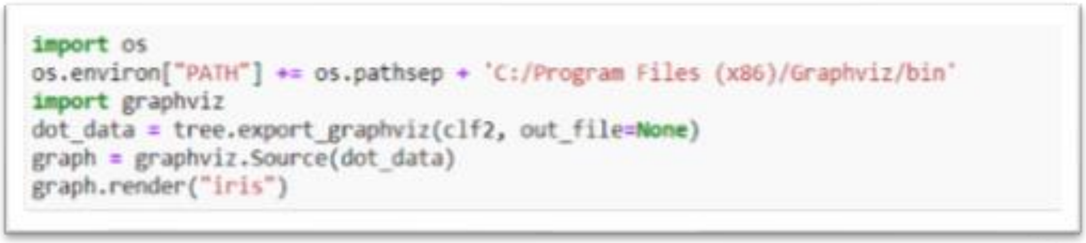


```
print("Accuracy:",metrics.accuracy_score(y_test, y_pred))

Accuracy: 0.6567853073521176
```

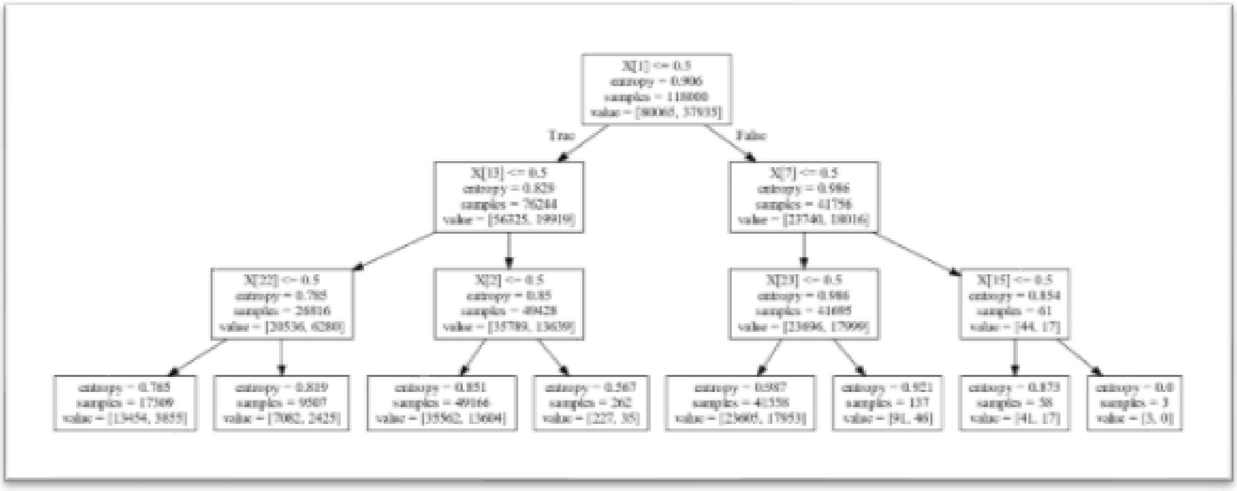
The accuracy of our model has a rate of 65,68%. The result is not good as expected because the accuracy is not very high.

3.3.5 Visualizing Decision Tree

We are used *export_graphviz* function to display the decision tree.



```python
import os
os.environ["PATH"] += os.pathsep + 'C:/Program Files (x86)/Graphviz/bin'
import graphviz
dot_data = tree.export_graphviz(clf2, out_file=None)
graph = graphviz.Source(dot_data)
graph.render("iris")
```



Conclusion

We are chosen the collision address type, weather condition, road condition, light condition as our independent variables to predict the severity of a collision. Our exploratory analysis shows us that most of the collisions happen on the following conditions: clear weather, dry road and daylight.

Our decision tree shows us when the collision happens in intersection, it is in daylight and road wet. While when the collision happens on alley, we do not have exact information on the weather. Furthermore, we are faced to a high number of *false negatives* to predict SEVERITYCLASS=2 and our results are biased because of imbalanced data issue.

Futures directions

For futures research, it could be interesting to increase the accuracy of our model by considering other features for analysis.

The model accuracy could maybe be improved by grouping independent variables into groups to reduce model bias (for example: good-moderate-bad driving conditions, which is influenced by LIGHTCOND, WEATHER, ROADCOND). To have more accurate results, it is also recommended to balance data.

It could be interesting to analyse our prediction with different algorithms than only Decision Tree as K-Nearest Neighbours (KNN) or Logistic Regression.