Introduction | Business Undertanding

As years go by, internet gets more and more into human lives and this fact has both positive and negative impact. A positive result of this fact is that people can have prior knowledge of the world outside their homes, before even going out. We can search for restaurant reviews and look into the menu before going there. We have weather forcasts for the days coming. We can online check in to our flights and choose our seat or we can book a ticket for a museum or a movie. All these privileges can prevent us from having a bad experience, wait to long queues and they save us a lot of time. Of course, all of the above could not be done without the data that are provided.

In this project, we are going to use a dataset regarding car accidents in Seattle city. Our purpose is to built a machine learning project, which can predict the severity of an accident given same independent variables, suc as weather conditions, road traffic and visibility conditions. This will help in reducing the frequency of car collisions in the specific community.

Data Understanding

Our dataset in hand cossists of 37 independent variables (columns) and 194,673 observations (rows). The sourse of our data is the Seattle Police Department and Accident Traffic Records Department and includes information from 2004 until now. "SEVERITYCODE" is going to be our dependent (target) variable, in other words the one to be predicted. Our target variable consists of numbers from 0 to 4 that correspond to different levels of severity caused by an accident and are coded as follows:

- **0: Little to no Probability - (Clear Conditions)**
- **1: Very Low Probability — Chance or Property Damage**
- **2: Low Probability — Chance of Injury**
- **3: Mild Probability — Chance of Serious Injury**
- **4: High Probability — Chance of Fatality**

The shape and form of our data is not appropriate to be included in the model, so before proceeding with the modeling phase, we will need to do some data preprocessing.

Data Preprocessing

Before proceeding with the modeling phase, we are going to make some modifications in our dataset, in order to be suitable for the model. Firsly, lets have a look into our dataset.

In [52]:

```
import pandas as pd

import types
import pandas as pd
from botocore.client import Config
import ibm_boto3

def __iter__(self): return 0

# @hidden_cell
# The following code accesses a file in your IBM Cloud Object Storage. It includes your c
redentials.
# You might want to remove those credentials before you share the notebook.
client_8a8aef6d2d0f44c584c8a8c616346ef5 = ibm_boto3.client(service_name='s3',
    ibm_api_key_id='P6dVk9ZBLbRdPShTs-XYLas_GK7r4HyhGT-OITkL8Cel',
    ibm_auth_endpoint="https://iam.cloud.ibm.com/oidc/token",
    config=Config(signature_version='oauth'),
    endpoint_url='https://s3.eu-geo.objectstorage.service.networklayer.com')

body = client_8a8aef6d2d0f44c584c8a8c616346ef5.get_object(Bucket='capstoneweek1-donotdele
te-pr-dgmhib76quulae',Key='Data-Collisions.csv')['Body']
```

```
# add missing __iter__ method, so pandas accepts body as file-like object
if not hasattr(body, "__iter__"): body.__iter__ = types.MethodType( __iter__, body )

coll = pd.read_csv(body)
coll.head()
```

Out[52]:

	SEVERITYCODE	X	Y	OBJECTID	INCKEY	COLDETKEY	REPORTNO	STATUS	ADDRTYPE	INTKEY	...
0	2	-122.323148	47.703140	1	1307	1307	3502005	Matched	Intersection	37475.0	...
1	1	-122.347294	47.647172	2	52200	52200	2607959	Matched	Block	NaN	...
2	1	-122.334540	47.607871	3	26700	26700	1482393	Matched	Block	NaN	...
3	1	-122.334803	47.604803	4	1144	1144	3503937	Matched	Block	NaN	...
4	2	-122.306426	47.545739	5	17700	17700	1807429	Matched	Intersection	34387.0	...

5 rows × 38 columns

We are not going to use all of the varibales into our machinr learning model, but only "ROADCOND","LIGHTCOND" and "WEATHER". So our dataset will look like the following, we "SEVERITYCODE" being our target variable and the rest three being our independent, explanatory variables.

In [36]:

```
coll1 = coll[["SEVERITYCODE","ROADCOND","LIGHTCOND","WEATHER"]]
coll1.head()
```

Out[36]:

	SEVERITYCODE	ROADCOND	LIGHTCOND	WEATHER
0	2	Wet	Daylight	Overcast
1	1	Wet	Dark - Street Lights On	Raining
2	1	Dry	Daylight	Overcast
3	1	Dry	Daylight	Clear
4	2	Wet	Daylight	Raining

As you can see from the below table, three out of variables of our dataset, are of type objcect. We need to transform these variables into categorical ones, in order to fit our models. We will use label encoding for this purpose and the resutls can be seen below.

In [37]:

```
coll1.dtypes
```

Out[37]:

```
SEVERITYCODE     int64
ROADCOND        object
LIGHTCOND       object
WEATHER         object
dtype: object
```

In [40]:

```
coll1["ROADCOND"].value_counts()
```

Out[40]:

```
Dry                124510
Wet                 47474
Unknown             15078
Ice                  1209
Snow/Slush           1004
Other                 132
Standing Water        115
Sand/Mud/Dirt          75
Oil                    64
Name: ROADCOND, dtype: int64
```

In [7]:

```
coll1["LIGHTCOND"].value_counts()
```

Out[7]:

```
Daylight                    116137
Dark - Street Lights On      48507
Unknown                      13473
Dusk                          5902
Dawn                          2502
Dark - No Street Lights       1537
Dark - Street Lights Off      1199
Other                          235
Dark - Unknown Lighting         11
Name: LIGHTCOND, dtype: int64
```

In [41]:

```
coll1["WEATHER"].value_counts()
```

Out[41]:

```
Clear                       111116
Raining                      33141
Overcast                     27702
Unknown                      15080
Snowing                        907
Other                          830
Fog/Smog/Smoke                 569
Sleet/Hail/Freezing Rain       113
Blowing Sand/Dirt               55
Severe Crosswind                25
Partly Cloudy                    5
Name: WEATHER, dtype: int64
```

In [61]:

```
from sklearn import preprocessing

label_encoder = preprocessing.LabelEncoder()

coll1['ROADCOND']= label_encoder.fit_transform(coll1['ROADCOND'])
coll1['LIGHTCOND'] = label_encoder.fit_transform(coll1['LIGHTCOND'].astype(str))
coll1['WEATHER'] = label_encoder.fit_transform(coll1['WEATHER'].astype(str))
coll1.head()
```

Out[61]:

	SEVERITYCODE	ROADCOND	LIGHTCOND	WEATHER
0	2	8	5	10
1	1	8	2	2
2	1	0	5	10
3	1	0	5	1

	SEVERITYCODE	ROADCOND	LIGHTCOND	WEATHER
4	2	8	5	2

If we check the different values of our target variable, we will see that is an inbalanced variable. In order to obtain better results from our models, we will have to balance the variable.

In [63]:

```
coll1["SEVERITYCODE"].value_counts()
```

Out[63]:

```
1    132533
2     57128
Name: SEVERITYCODE, dtype: int64
```

As you can see from the above output, the observations in class 1 are almost three times bigger than the observations in class 2. We will solve this discrepancy by downsampling class 1.

In [70]:

```
from sklearn.utils import resample

coll_max = coll1[coll1.SEVERITYCODE==1]
coll_min = coll1[coll1.SEVERITYCODE==2]

coll_sample_max = resample(coll_max, replace = False, n_samples=57128, random_state=123)

coll_bal = pd.concat([coll_sample_max, coll_min])

coll_bal["SEVERITYCODE"].value_counts()
```

Out[70]:

```
2    57128
1    57128
Name: SEVERITYCODE, dtype: int64
```

After keeping only the variables of interest, encoding them in order to have numerical values and balancing our target variable so to have only more unbased results, we are ready to run our models.

We are going to employ the following machine learing models:

- **K Nearest Neighbour (KNN)**
- **Decision Tree**
- **Linear Regression**

Before impementing the above models, we have to split our dataset into train and test.

In [72]:

```
from sklearn.model_selection import train_test_split

x = coll_bal[["ROADCOND","LIGHTCOND","WEATHER"]]
y = coll_bal["SEVERITYCODE"]

x_train, x_test, y_train, y_test = train_test_split (x, y, test_size = 0.2, random_state
= 0)
```

KNN

In [82]:

```
from sklearn.neighbors import KNeighborsClassifier

k = 17
```

```
knn = KNeighborsClassifier(n_neighbors = k).fit(x,y)
knn_y_pred = knn.predict(x_test)
knn_y_pred[0:5]
```

Out[82]:

```
array([1, 1, 1, 1, 1])
```

In [94]:

```
from sklearn.metrics import jaccard_similarity_score
from sklearn.metrics import f1_score
from sklearn.metrics import log_loss

knn1 = jaccard_similarity_score(y_test, knn_y_pred)
knn2 = f1_score(y_test, knn_y_pred, average='weighted')
```

Decision Tree

In [91]:

```
from sklearn.tree import DecisionTreeClassifier

dt = DecisionTreeClassifier(criterion='entropy', max_depth = 7)

dt.fit(x_train,y_train)

dt_y_pred = dt.predict(x_test)

dt_y_pred[0:5]
```

Out[91]:

```
array([2, 1, 2, 1, 1])
```

In [95]:

```
dt1 = jaccard_similarity_score(y_test, dt_y_pred)
dt2 = f1_score(y_test, dt_y_pred, average='weighted')
```

Linear Regression

In [99]:

```
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import confusion_matrix

lr = LogisticRegression(C=6, solver='liblinear').fit(x_train,y_train)

lr_y_pred = lr.predict(x_test)

lr_y_prob = lr.predict_proba(x_test)
```

In [102]:

```
lr1 = jaccard_similarity_score(y_test, lr_y_pred)
lr2 = f1_score(y_test, lr_y_pred, average='weighted')
lr3 = log_loss(y_test, lr_y_prob)

lr3
```

Out[102]:

```
0.6874765557345517
```

Model Evaluations

In [104]:

```
list_jc = [round(knn1,3), round(dt1,3), round(lr1,3)]
list_fsc = [round(knn2,3), round(dt2,3), round(lr2,3)]
list_ll = ['NA', 'NA',round(lr3,3)]

df = pd.DataFrame(list_jc, index=['KNN','Decision Tree','Logistic Regression'])
df.columns = ['Jaccard']
df.insert(loc=1, column='F1-score', value=list_fsc)
df.insert(loc=2, column='LogLoss', value=list_ll)
df.columns.name = 'Algorithm'
df
```

Out[104]:

Algorithm	Jaccard	F1-score	LogLoss
KNN	0.512	0.466	NA
Decision Tree	0.562	0.537	NA
Logistic Regression	0.539	0.526	0.687

Conclusion

From the above evaluation results we can conclude that Decission Tree model gives the most accuarate results. Moreover, we can say that, based on the dataset for car accident severity in Seattle, weather, light and road conditions have impact on the whether the travel is going to result in property damage or injury. So, it would be good to check the weather conditions and if the road is well constructed and sufficiently brigh in order to guarantee a safe travel.

In [ ]:

```
```