Car Accident Severity Prediction

By: Yassine Fahim
26th September, 202

Introduction

Car accidents are terrible events that can take someone's life or cause significant delays. The severity is a factor that measures the impact of a car accident on the traffic.

Car accidents can be reduced by predicting their severity.

While driving in dangerous areas under dangerous weather conditions, car drivers can be alerted with the degree of severity of a potential car accident so that they can reduce the risk by applying safety measures.

Interested parties: Local authorities, Google maps or Waze...

Data

The dataset used in this project is a subset (120 thousand accidents) of the dataset of 3.5 million traffic accidents that took place in the United States, from February 2016 to June 2020.

The dataset was retrieved from Kaggle.

It includes 49 different features about the details of accidents such as the location, time, weather…

Dropping duplicates, missing values, irrelevant features, and outliers.

Ending with a dataset of 60 thousand accidents.

Feature Selection

The model will predict the severity according to the weather conditions, time features, and POI features.

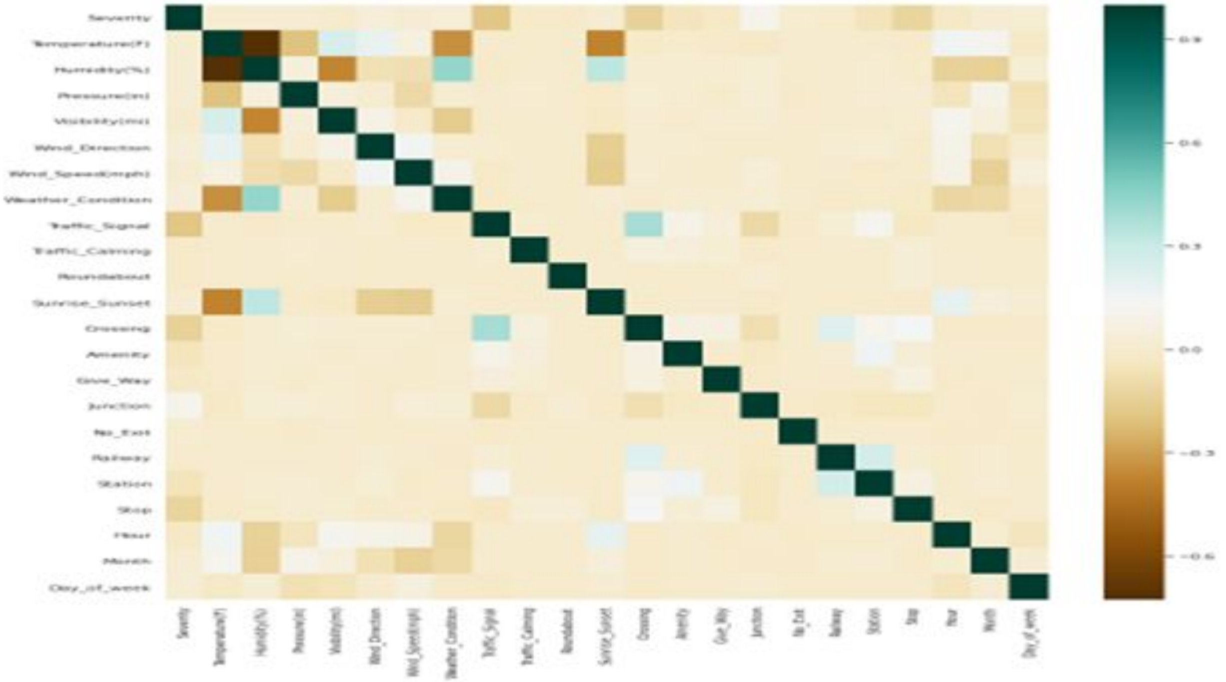
The selected features: 'Severity', `Hour’, `Day of week’, `Month’, 'Temperature(F)', 'Humidity(%)', 'Pressure(in)', 'Visibility(mi)', 'Wind_Direction', 'Wind_Speed(mph)', 'Weather_Condition', 'Traffic_Signal', 'Traffic_Calming', 'Roundabout', 'Sunrise_Sunset', 'Crossing','Amenity', 'Give_Way', 'Junction', 'No_Exit', 'Railway', 'Station', 'Stop'.

Exploratory Data Analysis

Severity grouped by the mean

Severity	Temperature(F)	Humidity(%)	Pressure(in)	Visibility(mi)	Wind_Direction	Wind_Speed(mph)
1.0	63.596078	64.058824	30.006078	9.411765	15.313725	8.347059
2.0	66.213265	58.586086	29.956070	9.371615	14.448234	8.124448
3.0	66.421100	59.476312	29.962552	9.389343	14.930213	8.298948
4.0	61.845000	68.600000	29.956500	8.800000	15.100000	8.195000

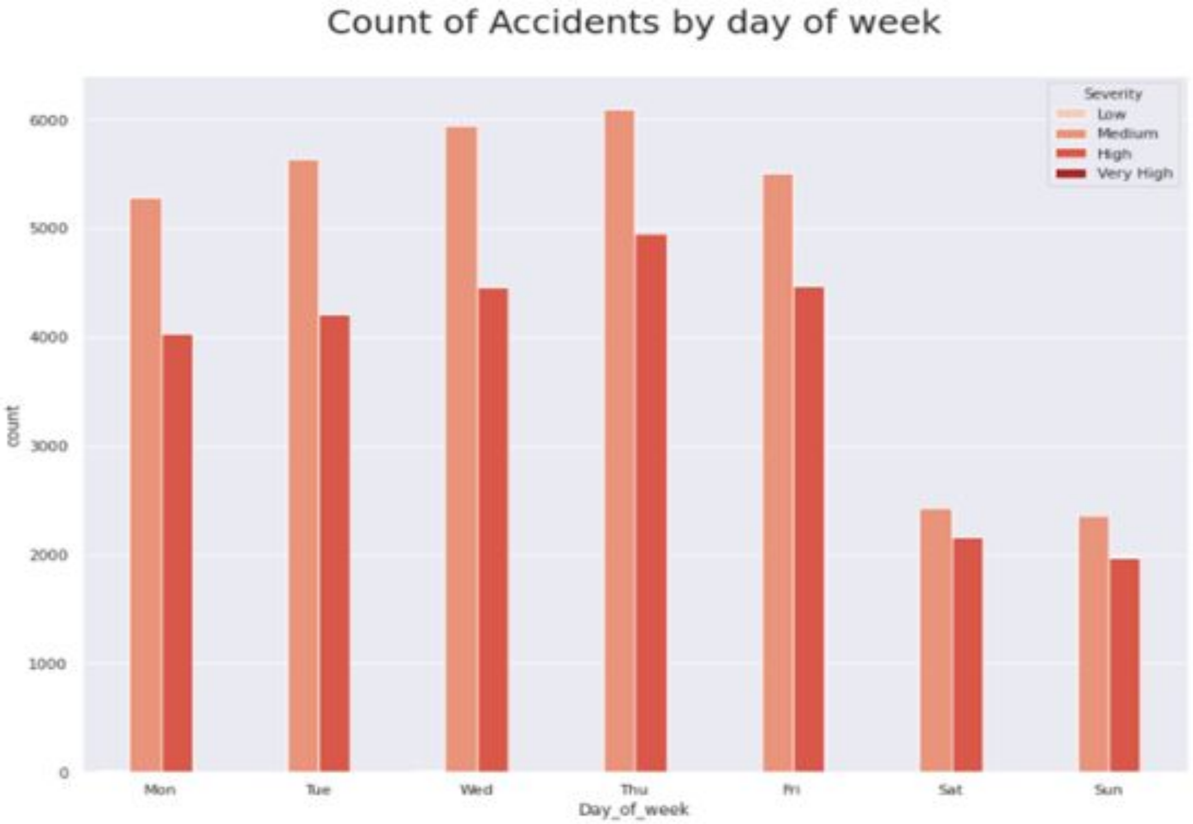
Correlation

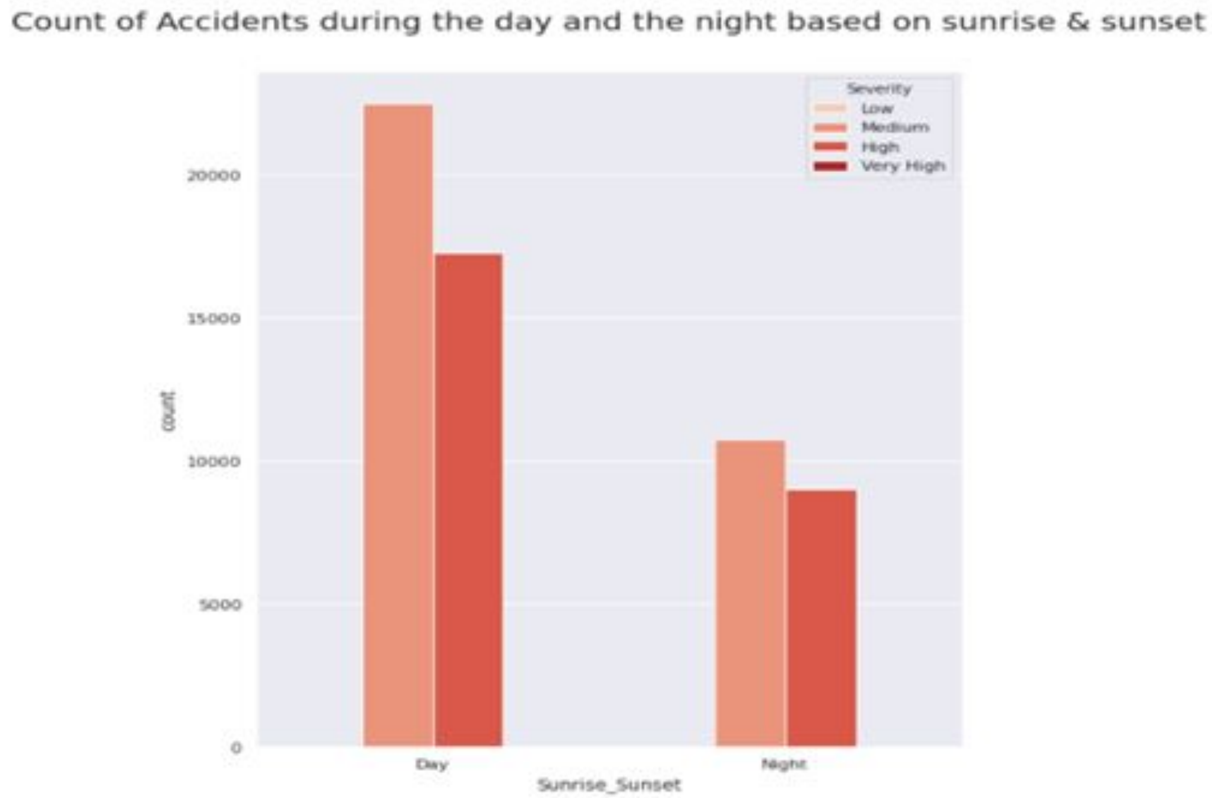


```
Traffic_Signal        -0.186074
Crossing              -0.136795
Stop                  -0.122309
Station               -0.053948
Amenity               -0.048250
Give_Way              -0.019850
Hour                  -0.019750
Traffic_Calming       -0.014488
Roundabout            -0.006266
Railway               -0.003294
Visibility(mi)         0.004249
Temperature(F)         0.008104
No_Exit                0.008733
Pressure(in)           0.015864
Humidity(%)            0.018906
Month                  0.018953
Sunrise_Sunset         0.019692
Wind_Speed(mph)        0.019730
Day_of_week            0.021675
Weather_Condition      0.039891
Wind_Direction         0.040140
Junction               0.103243
Severity               1.000000
Name: Severity, dtype: float64
```

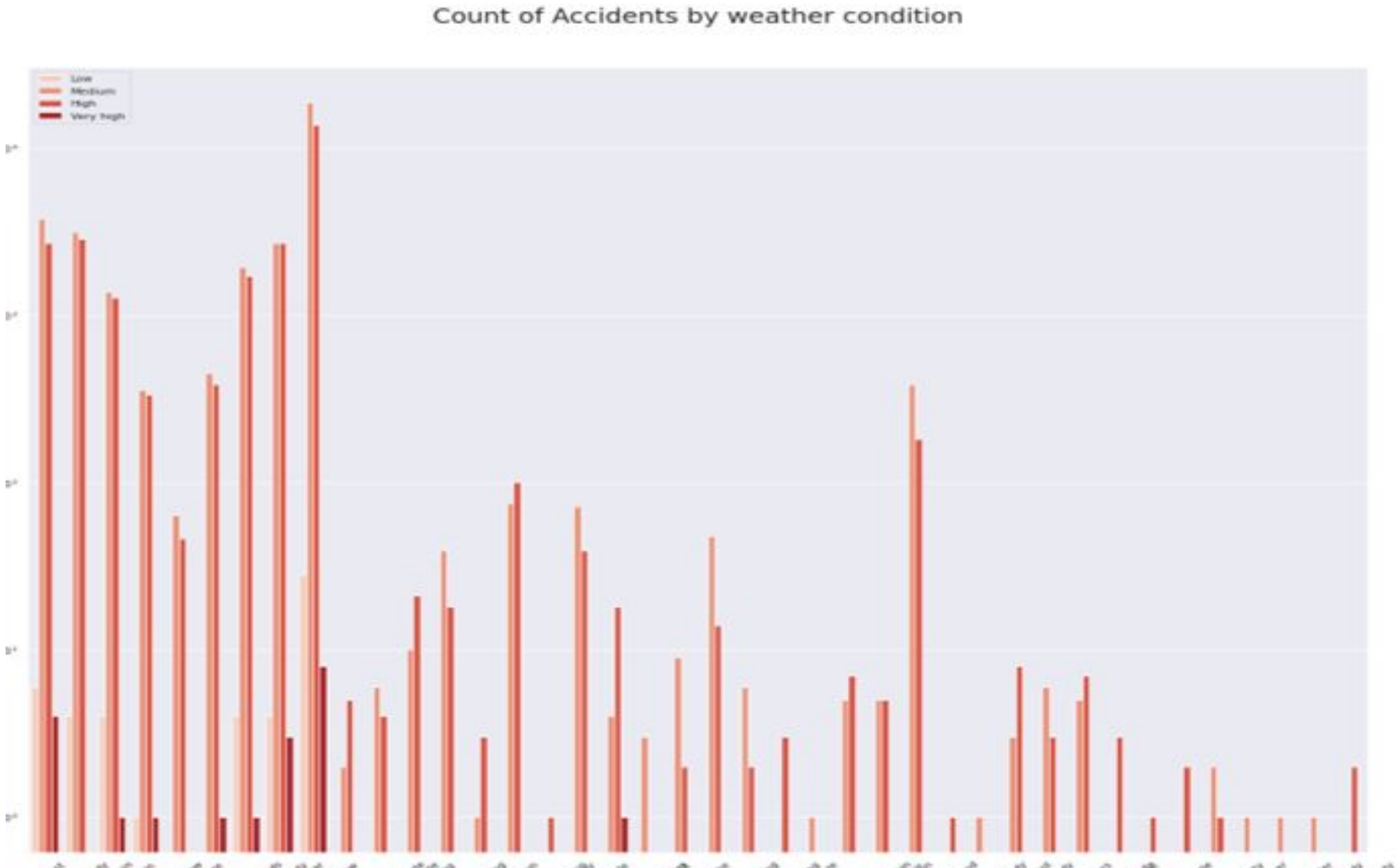
Time features and Severity


Count of Accidents by Month
Severity
Low
Medium
High
Very High
count
4000
3000
2000
1000
0
1.0
2.0
3.0
4.0
5.0
6.0
7.0
8.0
9.0
10.0
11.0
12.0
Month

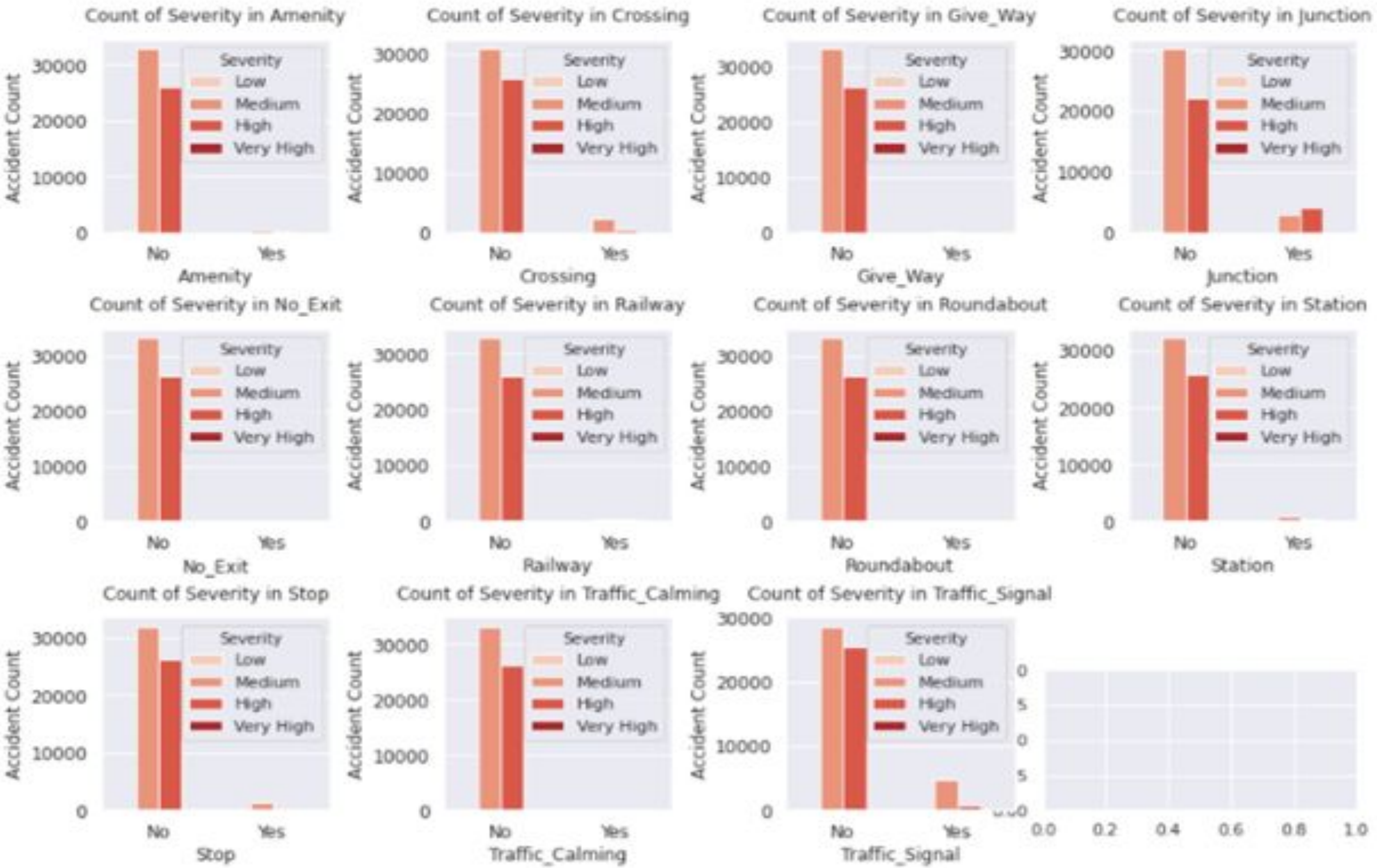

Count of Accidents by day of week
Severity
Low
Medium
High
Very High
6000
5000
4000
3000
2000
1000
0
count
Mon
Tue
Wed
Thu
Fri
Sat
Sun
Day_of_week


Count of Accidents during the day and the night based on sunrise & sunset
Severity
Low
Medium
High
Very High
count
0
5000
10000
15000
20000
Day
Night
Sunrise_Sunset

Severity and weather features



POI features and severity



Classification Models

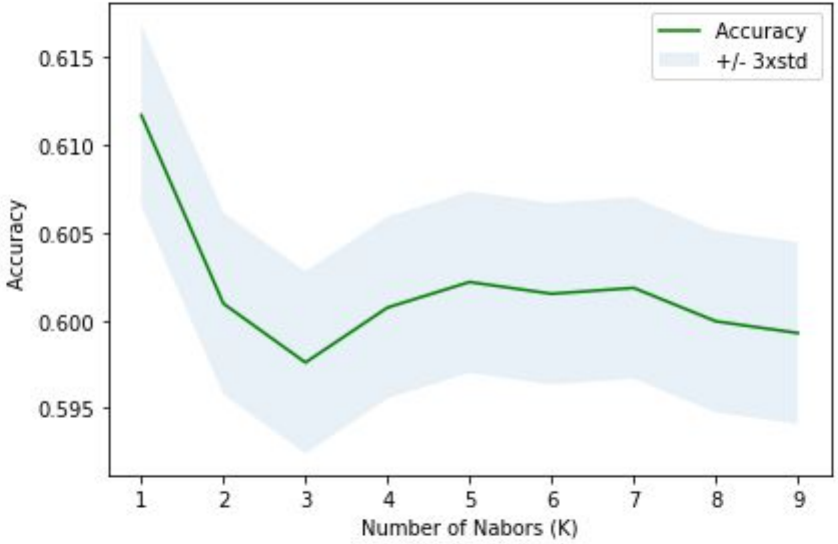
Import the clean dataset

Split the data to Train/Test sample (85%-15%)

```
# We split the X into train and test
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.15, random_state=4)
print ('Train set:', X_train.shape,  y_train.shape)
print ('Test set:', X_test.shape,  y_test.shape)
```

```
Train set: (50620, 22) (50620,)
Test set: (8934, 22) (8934,)
```

KNN




```
The best accuracy was with 0.6117080814864563 with k= 1
```

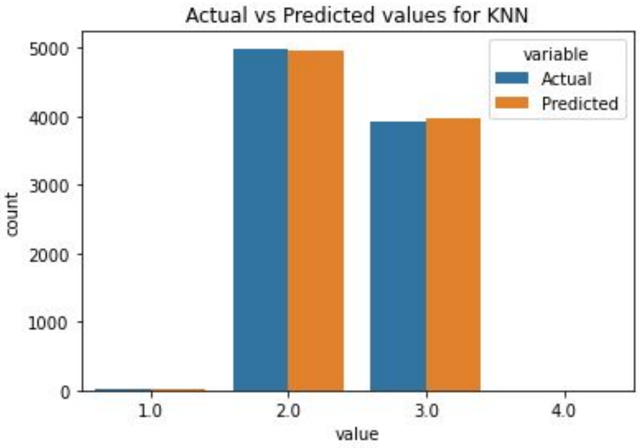
```python
from sklearn.neighbors import KNeighborsClassifier
k = 1
#Train Model and Predict
neigh = KNeighborsClassifier(n_neighbors = k).fit(X_train,y_train)
neigh
```

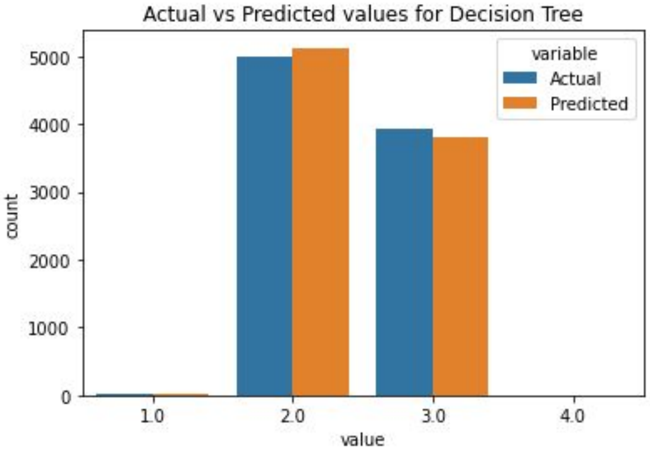
Decision Tree

```
from sklearn.tree import DecisionTreeClassifier
DT_model = DecisionTreeClassifier(criterion="entropy", splitter = "best", max_depth = 30)
DT_model.fit(X_train,y_train)
DT_model
```

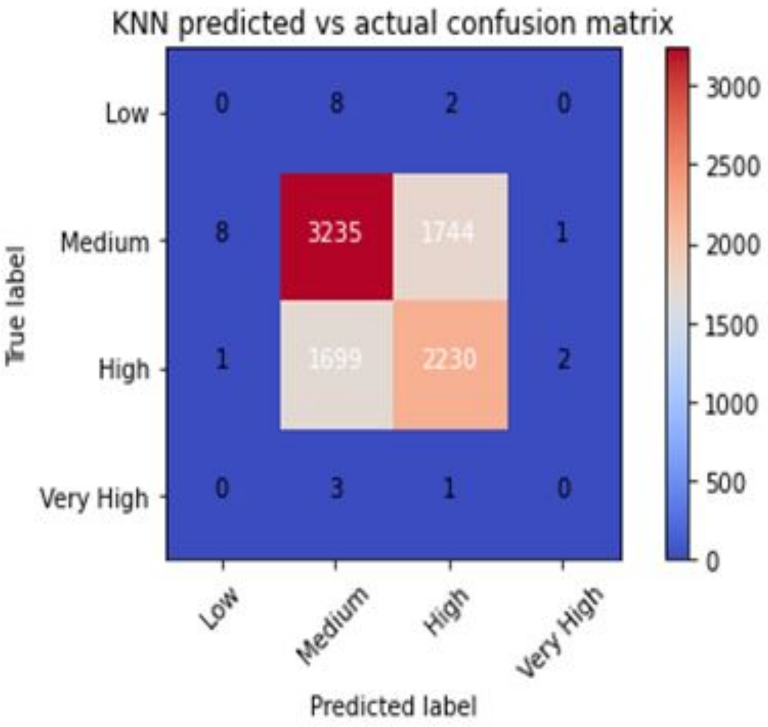
Results

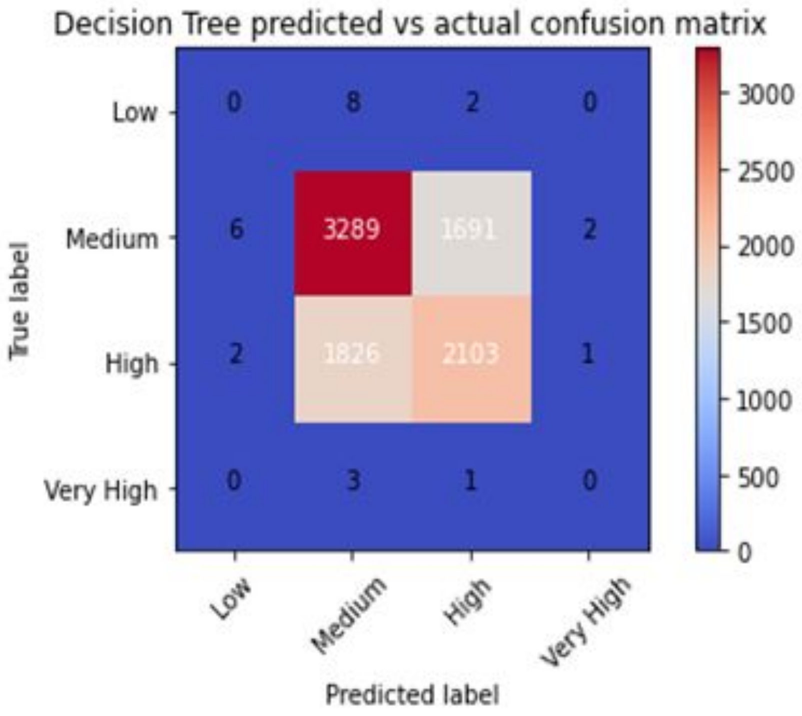
Models	Jaccard	F1s
KNN	0.611708	0.611855
DecisionTree	0.601746	0.600777





Confusion Matrix





Conclusion and Discussion

The accuracy of the two models is around 61%

The results obtained by these two models are not very satisfying.

To increase the accuracy, we can add more data to the models and more relevant features.

Low computational power