Accident Severity Prediction

Kargil Thakur

Introduction

- In an effort to reduce the frequency of car collisions in a community, an algorithm must be developed to predict the severity of an accident given the current weather, road conditions , visibility conditions and also based on the day of week(weekday or weekend).
- When conditions are bad, this model will alert drivers to remind them to be more careful.

Data

- The data is the accident data for Great Britain , dated from 2005 to 2010 . This dataset was available on kaggle :- https://www.kaggle.com/pachriisk/great-britain-road-accidents?select=uk-traffic-accidents-columns.csv
- The different columns of the dataset like light conditions , road_surface_conditions , weather_conditions , day_of_week will help us to predict the feature - accident severity .

Contents of Dataset

1. Accident_Index
2. Accident_Severity
3. Date
4. Day_of_Week
5. Light_Conditions
6. Number_of_Casualties
7. Number_of_Vehicles
8. Pedestrian_Crossing-Human_Control
9. Pedestrian_Crossing-Physical_Facilities
10. Road_Surface_Conditions
11. Speed_limit
12. Time
13. Weather_Conditions
14. Year

Methodology

- I have used the KNN classification for predicting the severity of the accident using the features like road conditions , weather conditions , light conditions and day of week.

- Firstly we preprocess the data by dropping the irrelevant columns , dropping the null values and encoding the data.

- Then we use the train_test_split for splitting the data for training and testing.

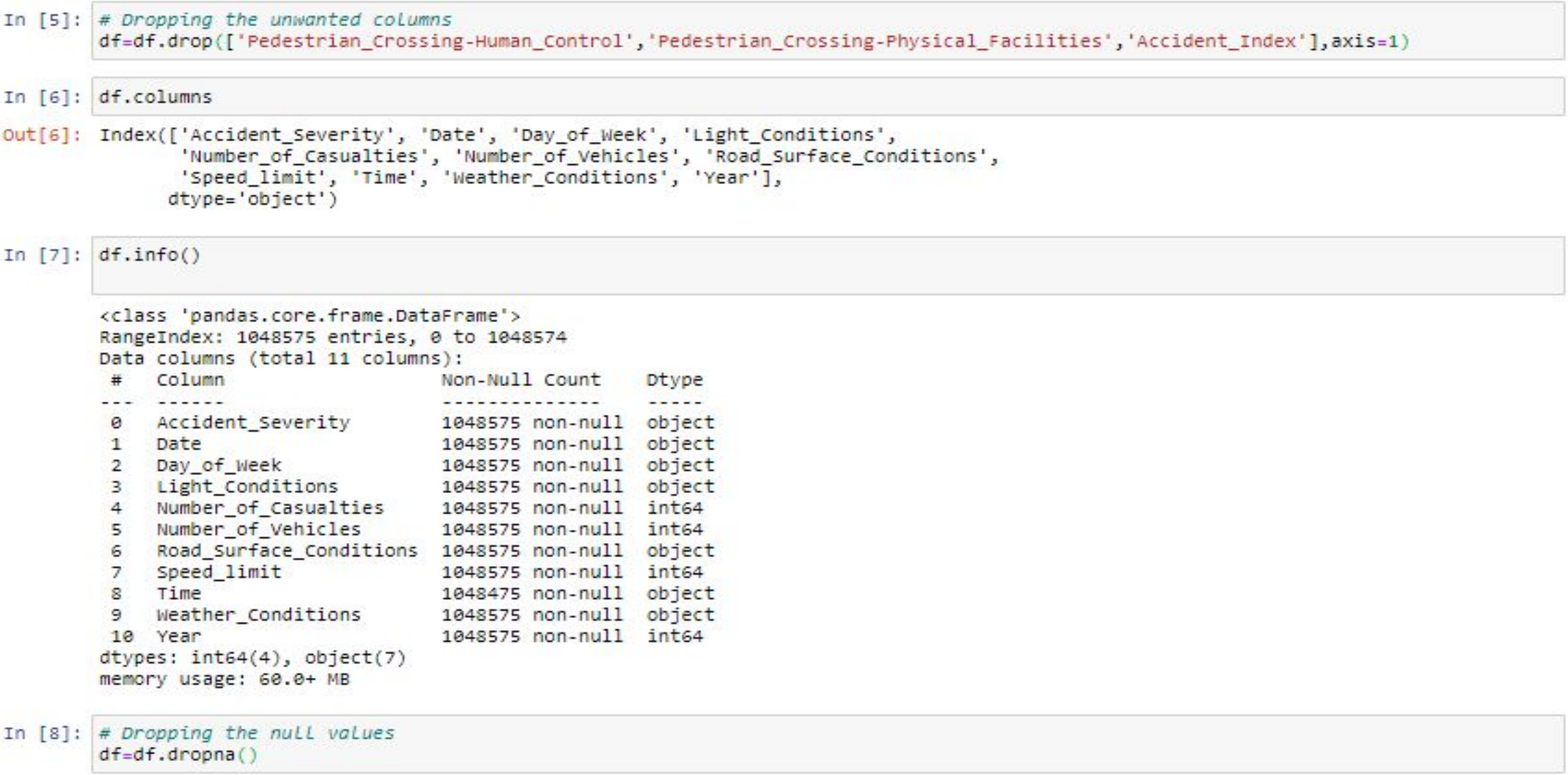
Methodology

- Creating the model:

  Firstly , we try to create the knn model using the k=6 , we train the model using fit_transform and predict some values from it.

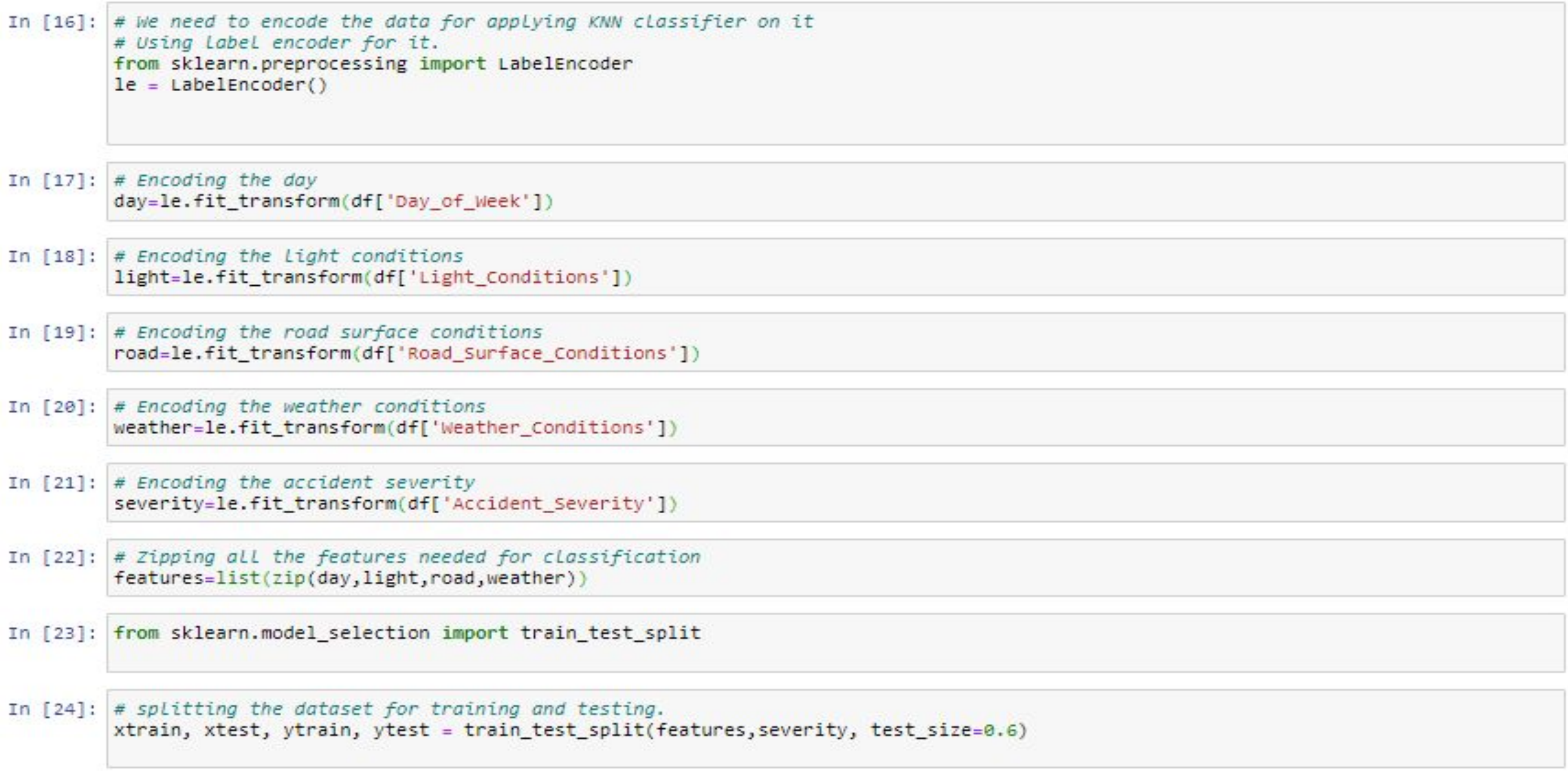
- Finding the accuracy:

  Now , we try to find the accuracy of the model by running it on the test data , and then finding the optimal K value for our classifier.



```
In [5]: # Dropping the unwanted columns
        df=df.drop(['Pedestrian_Crossing-Human_Control','Pedestrian_Crossing-Physical_Facilities','Accident_Index'],axis=1)

In [6]: df.columns

Out[6]: Index(['Accident_Severity', 'Date', 'Day_of_Week', 'Light_Conditions',
               'Number_of_Casualties', 'Number_of_Vehicles', 'Road_Surface_Conditions',
               'Speed_limit', 'Time', 'Weather_Conditions', 'Year'],
              dtype='object')

In [7]: df.info()

        <class 'pandas.core.frame.DataFrame'>
        RangeIndex: 1048575 entries, 0 to 1048574
        Data columns (total 11 columns):
         #   Column                   Non-Null Count    Dtype 
        ---  ------                   --------------    ----- 
         0   Accident_Severity        1048575 non-null  object
         1   Date                     1048575 non-null  object
         2   Day_of_Week              1048575 non-null  object
         3   Light_Conditions         1048575 non-null  object
         4   Number_of_Casualties     1048575 non-null  int64 
         5   Number_of_Vehicles       1048575 non-null  int64 
         6   Road_Surface_Conditions  1048575 non-null  object
         7   Speed_limit              1048575 non-null  int64 
         8   Time                     1048475 non-null  object
         9   Weather_Conditions       1048575 non-null  object
         10  Year                     1048575 non-null  int64 
        dtypes: int64(4), object(7)
        memory usage: 60.0+ MB

In [8]: # Dropping the null values
        df=df.dropna()
```

Methodology



```
In [16]: # We need to encode the data for applying KNN classifier on it
         # Using Label encoder for it.
         from sklearn.preprocessing import LabelEncoder
         le = LabelEncoder()
```

```
In [17]: # Encoding the day
         day=le.fit_transform(df['Day_of_Week'])
```

```
In [18]: # Encoding the light conditions
         light=le.fit_transform(df['Light_Conditions'])
```

```
In [19]: # Encoding the road surface conditions
         road=le.fit_transform(df['Road_Surface_Conditions'])
```

```
In [20]: # Encoding the weather conditions
         weather=le.fit_transform(df['Weather_Conditions'])
```

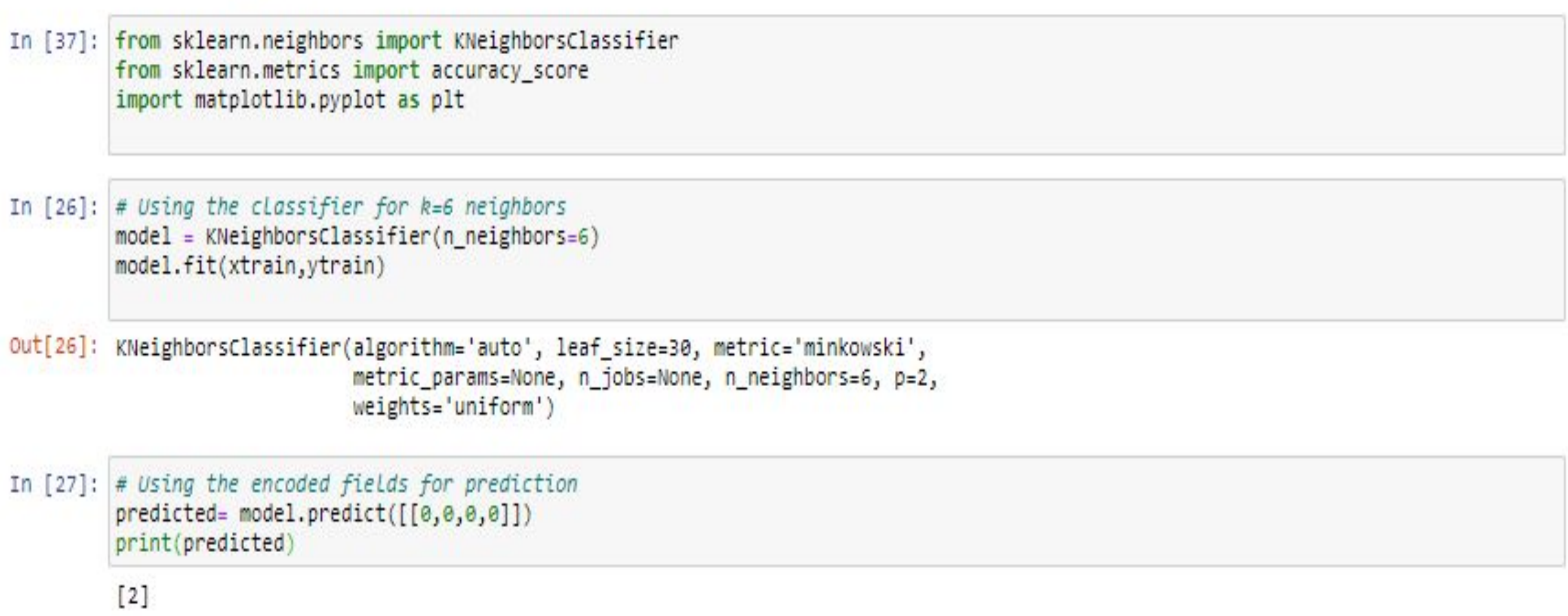
```
In [21]: # Encoding the accident severity
         severity=le.fit_transform(df['Accident_Severity'])
```

```
In [22]: # Zipping all the features needed for classification
         features=list(zip(day,light,road,weather))
```

```
In [23]: from sklearn.model_selection import train_test_split
```

```
In [24]: # splitting the dataset for training and testing.
         xtrain, xtest, ytrain, ytest = train_test_split(features,severity, test_size=0.6)
```

Methodology

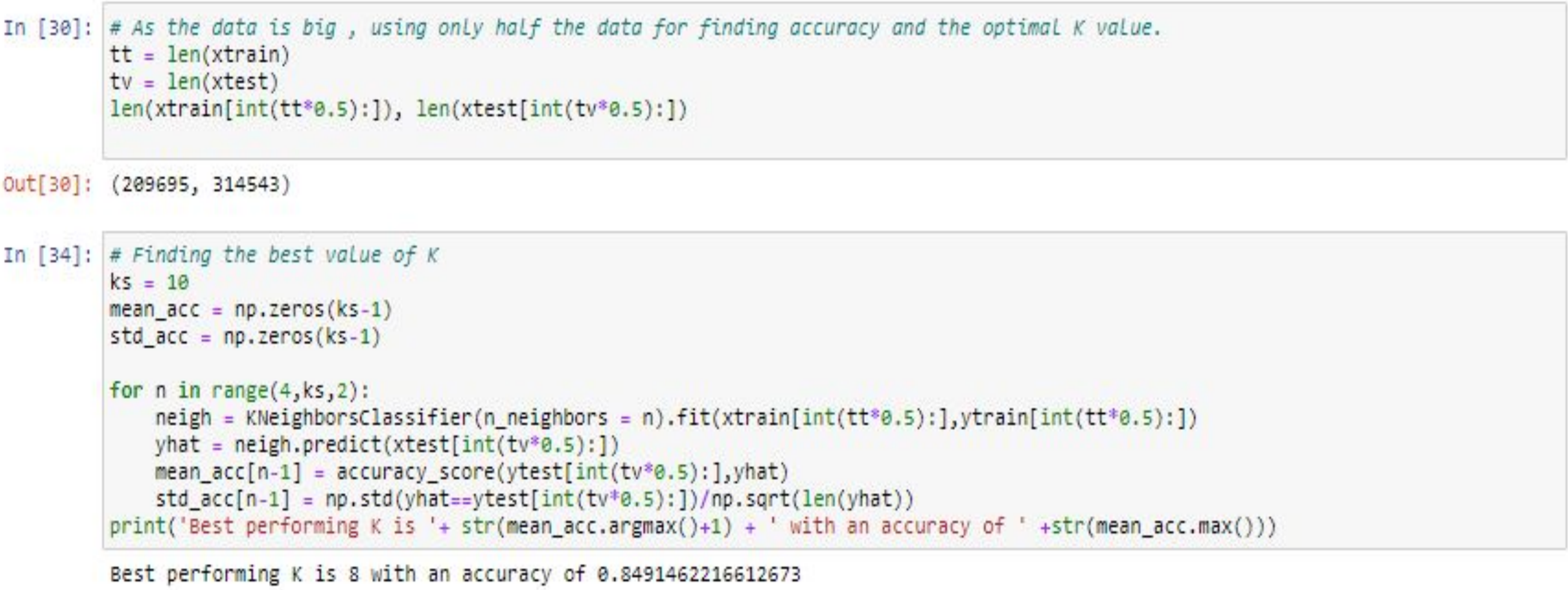


```
In [37]: from sklearn.neighbors import KNeighborsClassifier
         from sklearn.metrics import accuracy_score
         import matplotlib.pyplot as plt
```

```
In [26]: # Using the classifier for k=6 neighbors
         model = KNeighborsClassifier(n_neighbors=6)
         model.fit(xtrain,ytrain)

Out[26]: KNeighborsClassifier(algorithm='auto', leaf_size=30, metric='minkowski',
                              metric_params=None, n_jobs=None, n_neighbors=6, p=2,
                              weights='uniform')
```

```
In [27]: # Using the encoded fields for prediction
         predicted= model.predict([[0,0,0,0]])
         print(predicted)

         [2]
```

Methodology



```
In [30]: # As the data is big , using only half the data for finding accuracy and the optimal K value.
         tt = len(xtrain)
         tv = len(xtest)
         len(xtrain[int(tt*0.5):]), len(xtest[int(tv*0.5):])

Out[30]: (209695, 314543)

In [34]: # Finding the best value of K
         ks = 10
         mean_acc = np.zeros(ks-1)
         std_acc = np.zeros(ks-1)

         for n in range(4,ks,2):
             neigh = KNeighborsClassifier(n_neighbors = n).fit(xtrain[int(tt*0.5):],ytrain[int(tt*0.5):])
             yhat = neigh.predict(xtest[int(tv*0.5):])
             mean_acc[n-1] = accuracy_score(ytest[int(tv*0.5):],yhat)
             std_acc[n-1] = np.std(yhat==ytest[int(tv*0.5):])/np.sqrt(len(yhat))
         print('Best performing K is '+ str(mean_acc.argmax()+1) + ' with an accuracy of ' +str(mean_acc.max()))

         Best performing K is 8 with an accuracy of 0.8491462216612673
```

The best value of K is 8 and it gives an accuracy of 84.9%

Methodology

Results

- We have used KNN classifiers to predict the severity of an accident on the basis of various features . We have found the accuracy of the model which is around 85% and this accuracy is got when k=8.

- Secondly, we have found that the KNN classifier is best for this problem statement because:-
  Very simple implementation. Robust with regard to the search space; for instance, classes don't have to be linearly separable.

Conclusion

Hence , we can predict the severity of any accident on the basis of the various features like light conditions , weather conditions ,etc and hence we can predict how much time the traffic would be stuck on the basis of severity of the accident.

The End