Accidents_MoreFeatures

January 7, 2020

1 Attempt in usage of Random Forest algorithm to classify severity of road congestion caused by an accident in the US

The work utilises data that can be found here: https://www.kaggle.com/sobhanmoosavi/us-accidents/data#, from which lebales are selected:

- Severity
- State
- Weather_Condition
- Precipitation(in)
- Crossing
- No_Exit
- Railway
- Roundabout
- Station
- Stop
- Traffic_Calming
- Traffic_Signal
- Turning_Loop
- Civil_Twilight
- Temperature(F)
- Visibility(mi)
- Distance(mi)
- Wind_Speed(mph)

to show the severity of the accident, a number between 1 and 4, where 1 indicates the least impact on traffic (i.e., short delay as a result of the accident) and 4 indicates a significant impact on traffic (i.e., long delay). The above has been attempted with use of Random Forest classification algorithm and described below.

1.1 First, we load original data with use of Pandas, which supports us with data structures, visualisation, manipulation and more:

[1]:
```
import pandas as pd
pd.set_option('display.max_columns', None)

dfOriginal = pd.read_csv('US_Accidents_May19.csv')
```

```
features = ['State', 'Weather_Condition', 'Civil_Twilight', 'Precipitation(in)',
 ↪, 'Crossing',\
                'No_Exit', 'Railway', 'Roundabout', 'Station', 'Stop',
 ↪'Traffic_Calming', 'Traffic_Signal', \
                'Turning_Loop', 'Temperature(F)', 'Visibility(mi)', \
                'Distance(mi)', 'Wind_Speed(mph)']

print('Original dataset shape:')
print(dfOriginal.shape)

dfOriginal.head()
```

```
Original dataset shape:
(2243939, 49)
```

[1]:

	ID	Source	TMC	Severity	Start_Time	End_Time	\
0	A-1	MapQuest	201.0	3	2016-02-08 05:46:00	2016-02-08 11:00:00	
1	A-2	MapQuest	201.0	2	2016-02-08 06:07:59	2016-02-08 06:37:59	
2	A-3	MapQuest	201.0	2	2016-02-08 06:49:27	2016-02-08 07:19:27	
3	A-4	MapQuest	201.0	3	2016-02-08 07:23:34	2016-02-08 07:53:34	
4	A-5	MapQuest	201.0	2	2016-02-08 07:39:07	2016-02-08 08:09:07	

	Start_Lat	Start_Lng	End_Lat	End_Lng	Distance(mi)	\
0	39.865147	-84.058723	NaN	NaN	0.01	
1	39.928059	-82.831184	NaN	NaN	0.01	
2	39.063148	-84.032608	NaN	NaN	0.01	
3	39.747753	-84.205582	NaN	NaN	0.01	
4	39.627781	-84.188354	NaN	NaN	0.01	

	Description	Number	\
0	Right lane blocked due to accident on I-70 Eas…	NaN	
1	Accident on Brice Rd at Tussing Rd. Expect del…	2584.0	
2	Accident on OH-32 State Route 32 Westbound at …	NaN	
3	Accident on I-75 Southbound at Exits 52 52B US…	NaN	
4	Accident on McEwen Rd at OH-725 Miamisburg Cen…	NaN	

	Street	Side	City	County	State	Zipcode	\
0	I-70 E	R	Dayton	Montgomery	OH	45424	
1	Brice Rd	L	Reynoldsburg	Franklin	OH	43068-3402	
2	State Route 32	R	Williamsburg	Clermont	OH	45176	
3	I-75 S	R	Dayton	Montgomery	OH	45417	
4	Miamisburg Centerville Rd	R	Dayton	Montgomery	OH	45459	

	Country	Timezone	Airport_Code	Weather_Timestamp	Temperature(F)	\
0	US	US/Eastern	KFFO	2016-02-08 05:58:00	36.9	
1	US	US/Eastern	KCMH	2016-02-08 05:51:00	37.9	
2	US	US/Eastern	KI69	2016-02-08 06:56:00	36.0	

```
3      US  US/Eastern         KDAY  2016-02-08 07:38:00            35.1
4      US  US/Eastern         KMGY  2016-02-08 07:53:00            36.0

   Wind_Chill(F)  Humidity(%)  Pressure(in)  Visibility(mi) Wind_Direction  \
0            NaN         91.0         29.68           10.0           Calm
1            NaN        100.0         29.65           10.0           Calm
2           33.3        100.0         29.67           10.0             SW
3           31.0         96.0         29.64            9.0             SW
4           33.3         89.0         29.65            6.0             SW

   Wind_Speed(mph)  Precipitation(in) Weather_Condition  Amenity   Bump  \
0              NaN               0.02        Light Rain    False  False
1              NaN               0.00        Light Rain    False  False
2              3.5                NaN          Overcast    False  False
3              4.6                NaN     Mostly Cloudy    False  False
4              3.5                NaN     Mostly Cloudy    False  False

   Crossing  Give_Way  Junction  No_Exit  Railway  Roundabout  Station   Stop  \
0     False     False     False    False    False       False    False  False
1     False     False     False    False    False       False    False  False
2     False     False     False    False    False       False    False  False
3     False     False     False    False    False       False    False  False
4     False     False     False    False    False       False    False  False

   Traffic_Calming  Traffic_Signal  Turning_Loop Sunrise_Sunset  \
0            False           False         False          Night
1            False           False         False          Night
2            False            True         False          Night
3            False           False         False          Night
4            False            True         False            Day

  Civil_Twilight Nautical_Twilight Astronomical_Twilight
0          Night             Night                 Night
1          Night             Night                   Day
2          Night               Day                   Day
3            Day               Day                   Day
4            Day               Day                   Day
```

1.2 Preparation of the data is crucial for performance and usability of the algorithm.

Selection of features with potentially more importance occurs below.

[2]:
```
df = dfOriginal[['Severity', 'State', 'Weather_Condition', 'Precipitation(in)',␣
↪'Crossing',\
                 'No_Exit', 'Railway', 'Roundabout', 'Station', 'Stop',␣
↪'Traffic_Calming', 'Traffic_Signal', \
```

```
                'Turning_Loop', 'Civil_Twilight', 'Temperature(F)',␣
↪'Visibility(mi)', \
                'Distance(mi)', 'Wind_Speed(mph)']]
```

The newly selected data frame, data with over 2million lines and 18 columns:

[3]:
```
print(df.shape)

df.head()
```

```
(2243939, 18)
```

[3]:

	Severity	State	Weather_Condition	Precipitation(in)	Crossing	No_Exit	\
0	3	OH	Light Rain	0.02	False	False	
1	2	OH	Light Rain	0.00	False	False	
2	2	OH	Overcast	NaN	False	False	
3	3	OH	Mostly Cloudy	NaN	False	False	
4	2	OH	Mostly Cloudy	NaN	False	False	

	Railway	Roundabout	Station	Stop	Traffic_Calming	Traffic_Signal	\
0	False	False	False	False	False	False	
1	False	False	False	False	False	False	
2	False	False	False	False	False	True	
3	False	False	False	False	False	False	
4	False	False	False	False	False	True	

	Turning_Loop	Civil_Twilight	Temperature(F)	Visibility(mi)	Distance(mi)	\
0	False	Night	36.9	10.0	0.01	
1	False	Night	37.9	10.0	0.01	
2	False	Night	36.0	10.0	0.01	
3	False	Day	35.1	9.0	0.01	
4	False	Day	36.0	6.0	0.01	

	Wind_Speed(mph)
0	NaN
1	NaN
2	3.5
3	4.6
4	3.5

The next step is to clean the dataset by removing rows with missing values, which comes with a penalty of data reduction to around 200000 rows.

[4]:
```
df = df.dropna()
df = df[df.Severity != 0]

print(df.shape)
df.head()
```

(236070, 18)

[4]:

	Severity	State	Weather_Condition	Precipitation(in)	Crossing	No_Exit	\
5	3	OH	Light Rain	0.03	False	False	
9	3	OH	Light Rain	0.02	False	False	
11	3	OH	Light Rain	0.02	False	False	
14	2	OH	Light Rain	0.02	False	False	
20	2	OH	Light Snow	0.01	False	False	

	Railway	Roundabout	Station	Stop	Traffic_Calming	Traffic_Signal	\
5	False	False	False	False	False	False	
9	False	False	False	False	False	False	
11	False	False	False	False	False	False	
14	False	False	False	False	False	True	
20	False	False	False	False	False	False	

	Turning_Loop	Civil_Twilight	Temperature(F)	Visibility(mi)	Distance(mi)	\
5	False	Day	37.9	7.0	0.01	
9	False	Day	37.4	3.0	0.01	
11	False	Day	37.4	3.0	0.01	
14	False	Day	37.4	3.0	0.01	
20	False	Day	33.8	2.0	0.00	

	Wind_Speed(mph)
5	3.5
9	4.6
11	4.6
14	4.6
20	4.6

We further prepare the data by encoding features described by strings.

[5]:
```
import numpy as np

X = df[features].values
print('Features in the dataframe before encoding:')
print(X)
print('\n')

#label encoding to integers
from sklearn.preprocessing import LabelEncoder

state_le = LabelEncoder()
X[:, 0] = state_le.fit_transform(X[:, 0])

weather_le = LabelEncoder()
X[:, 1] = weather_le.fit_transform(X[:, 1])
```

```
twilight_le = LabelEncoder()
X[:, 2] = twilight_le.fit_transform(X[:, 2])

print('Encoded features in the dataframe: ')
print(X)
print('\n')

y = df.iloc[:, 0].values

print('Class labels:', np.unique(y))
print(X.shape)
```

```
Features in the dataframe before encoding:
[['OH' 'Light Rain' 'Day' … 7.0 0.01 3.5]
 ['OH' 'Light Rain' 'Day' … 3.0 0.01 4.6]
 ['OH' 'Light Rain' 'Day' … 3.0 0.01 4.6]
 …
 ['FL' 'Light Rain' 'Day' … 5.0 0.147 5.8]
 ['FL' 'Light Rain' 'Day' … 5.0 0.146 5.8]
 ['TN' 'Rain' 'Day' … 4.0 0.065 9.2]]


Encoded features in the dataframe:
[[33 24 0 … 7.0 0.01 3.5]
 [33 24 0 … 3.0 0.01 4.6]
 [33 24 0 … 3.0 0.01 4.6]
 …
 [8 24 0 … 5.0 0.147 5.8]
 [8 24 0 … 5.0 0.146 5.8]
 [40 34 0 … 4.0 0.065 9.2]]


Class labels: [1 2 3 4]
(236070, 17)
```

1.3 Selecting arrays for training and testing

[6]:
```
from sklearn.model_selection import train_test_split

X_train, X_test, y_train, y_test = train_test_split(X, y,
                     test_size=0.3,
                     random_state=42,
                     stratify=y)
```

1.4 Preparation of the Random Forest Classifier and training data fitting.

[7]:
```
from sklearn.ensemble import RandomForestClassifier
feat_labels = df.columns[1:]

forest = RandomForestClassifier(n_estimators=500,
                                random_state=42)

forest.fit(X_train, y_train)

print('Done')
```

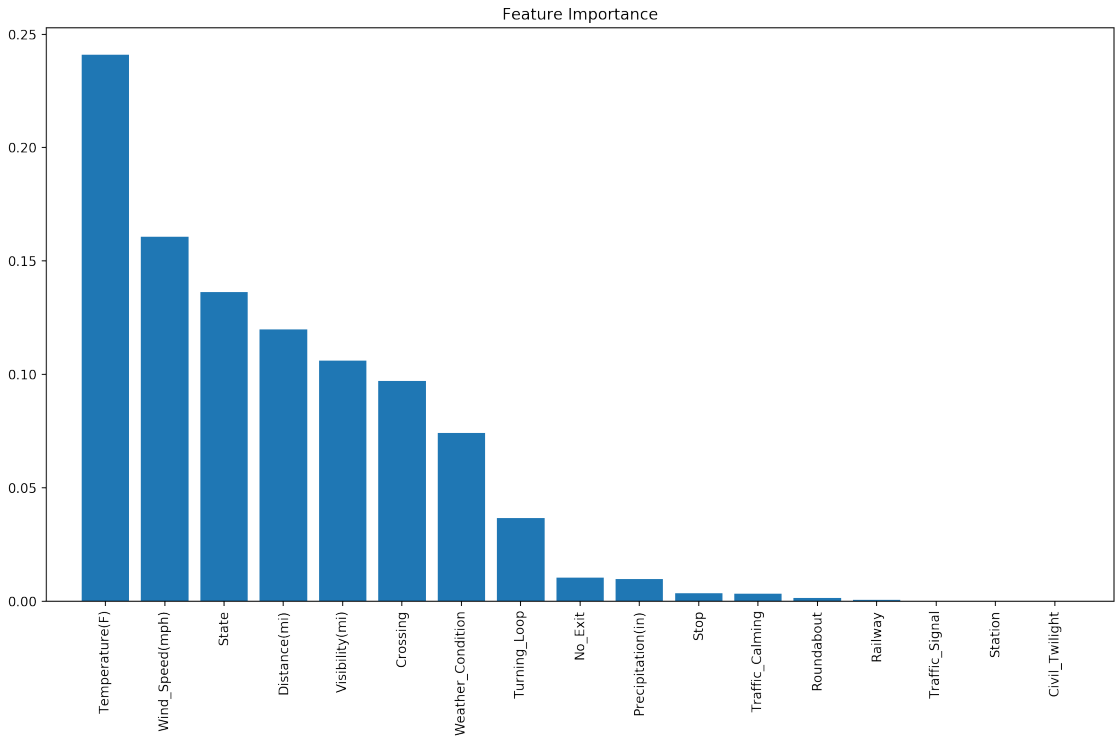
```
Done
```

1.5 Plotting feature importance

[16]:
```
importances = forest.feature_importances_

indices = np.argsort(importances)[::-1]

import matplotlib.pyplot as plt

for f in range(X_train.shape[1]):
    print("%2d) %-*s %f" % (f + 1, 30,
                            feat_labels[indices[f]],
                            importances[indices[f]]))

plt.figure(figsize=(12,8), dpi=200)
plt.title('Feature Importance')

plt.bar(range(X_train.shape[1]), importances[indices], align='center')

plt.xticks(range(X_train.shape[1]), feat_labels[indices], rotation=90)
plt.xlim([-1, X_train.shape[1]])
plt.tight_layout()

plt.savefig('US_Accidents_MoreFeatures_FeatureImportance.png', dpi=300)
```

```
 1) Temperature(F)                0.240843
 2) Wind_Speed(mph)               0.160665
 3) State                         0.136239
 4) Distance(mi)                  0.119821
 5) Visibility(mi)                0.106051
 6) Crossing                      0.096971
 7) Weather_Condition             0.074162
 8) Turning_Loop                  0.036510
 9) No_Exit                       0.010328
10) Precipitation(in)             0.009685
```

```
11) Stop                          0.003389
12) Traffic_Calming               0.003320
13) Roundabout                    0.001314
14) Railway                       0.000435
15) Traffic_Signal                0.000205
16) Station                       0.000062
17) Civil_Twilight                0.000000
```



1.6 Random Forest algorithm performance analysis

Random Forest training score:

[9]:
```
print(forest.score(X_test, y_test))
```

```
0.6859829711526242
```

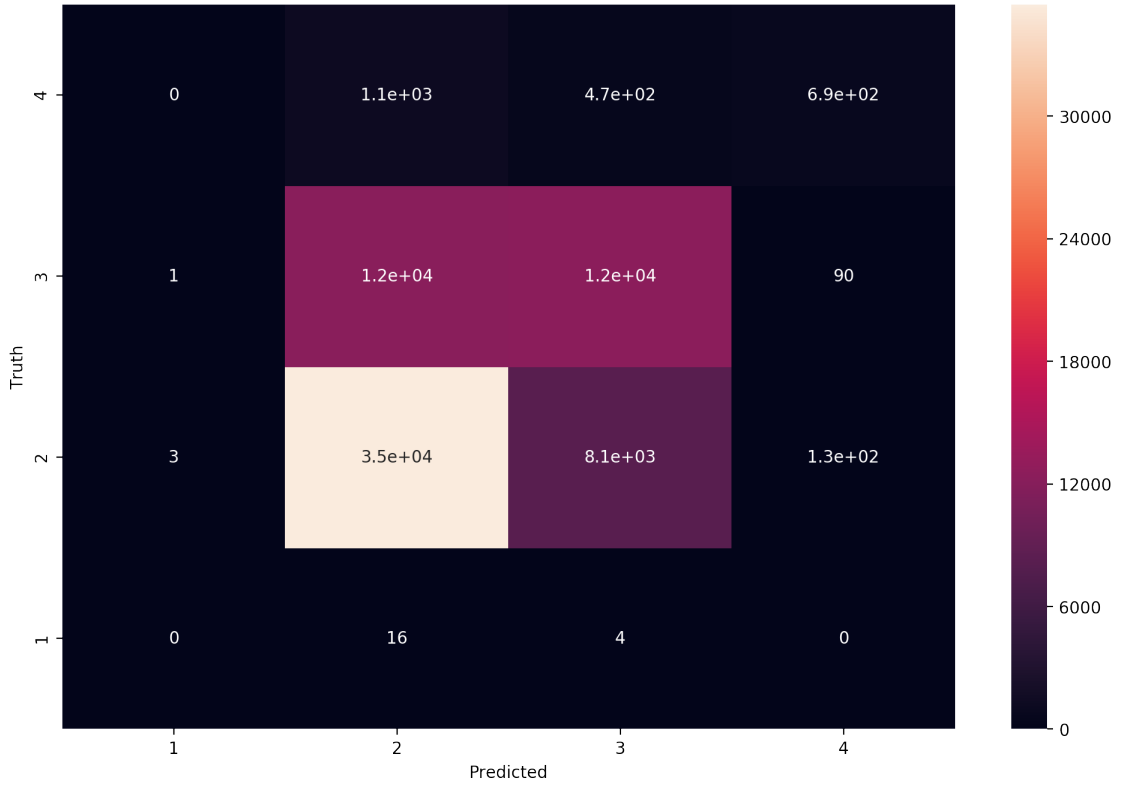
[10]:
```
forest.score(X_test, y_test)

y_predicted = forest.predict(X_test)

from sklearn.metrics import confusion_matrix
cm = confusion_matrix(y_test, y_predicted)
cm
```

[10]:
```
array([[    0,    16,     4,     0],
       [    3, 35456,  8133,   131],
       [    1, 12273, 12432,    90],
       [    0,  1118,   470,   694]], dtype=int64)
```

1.7 Plotting Confusion Matrix

[11]:
```
%matplotlib inline
import matplotlib.pyplot as plt
import seaborn as sn

plt.figure(figsize=(12,8), dpi=200)
ax = plt.axes()

sn.heatmap(cm, annot=True)

ax.xaxis.set_ticklabels(['1', '2', '3', '4']); ax.yaxis.set_ticklabels(['1',␣
↪'2', '3', '4']);
plt.xlabel('Predicted')
plt.ylabel('Truth')
plt.ylim(0, 4)
```

[11]:
```
(0, 4)
```



1.8 Selecting one of the trees nad tree display

[12]:
```
# Extracting single tree
estimator = forest.estimators_[5]
```

[13]:
```
from sklearn.tree import export_graphviz

# Export as dot file
treeFileName = 'Accidents_MoreFeatures_tree'
export_graphviz(estimator, out_file = treeFileName+'.dot',
                feature_names = features,
                class_names = ['1', '2', '3', '4'],
                rounded = True, proportion = False,
                precision = 2, filled = True)
```

The entire tree is huge and weights 11MB, which proved problematic for png conversion. The original .dot file has been trimmed manually, displayed with a tool online. A part of the resultant visualisation can be seen below:

[14]:
```
from IPython.display import Image
Image(filename = 'Accidents_MoreFeatures_tree_trimmed.jpg')
```

[14]: