# 657A_assgn2_CM8

July 18, 2021

## 1 CM[8] Kaggle Competition Group38

[5]:

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
import sklearn
from sklearn.tree import DecisionTreeClassifier
from sklearn.model_selection import train_test_split
from sklearn import metrics
from sklearn.model_selection import KFold
from sklearn.ensemble import RandomForestClassifier
from sklearn.ensemble import GradientBoostingClassifier
from sklearn import tree
import scipy
```

[6]:

```
df = pd.read_csv("dkmacovid_kaggletest_features.csv")
df_train = pd.read_csv('cleaned_normalized_coviddata.csv')
X_train = df_train.iloc[:,2:-3]
y_train = df_train.iloc[:,-3:].astype('int')
df.head(5)
```

[6]:

```
   Id  Day  State ID     State      Lat    Long_  Active  Incident_Rate  \
0   0    2        14  Illinois  40.3495 -88.9861  957138    7697.015291
1   5    3        14  Illinois  40.3495 -88.9861  961499    7732.282519
2  10    4        14  Illinois  40.3495 -88.9861  966468    7772.205747
3  15    5        14  Illinois  40.3495 -88.9861  973157    7826.175891
4  20    6        14  Illinois  40.3495 -88.9861  980553    7885.906848

   Total_Test_Results  Case_Fatality_Ratio  Testing_Rate  \
0            13436652             1.867428   106035.6834
1            13482117             1.869933   106394.4716
2            13530371             1.869466   106775.2693
3            13617454             1.871700   107462.4870
4            13698428             1.874835   108101.4954

  Resident Population 2020 Census  Population Density 2020 Census  \
0                      12,812,508                           230.8
1                      12,812,508                           230.8
2                      12,812,508                           230.8
3                      12,812,508                           230.8
4                      12,812,508                           230.8

   Density Rank 2020 Census  SexRatio
0                        14        97
1                        14        97
2                        14        97
3                        14        97
4                        14        97
```

```
[7]: df.dtypes
```

```
[7]: Id                                     int64
     Day                                    int64
     State ID                               int64
     State                                 object
     Lat                                  float64
     Long_                                float64
     Active                                 int64
     Incident_Rate                        float64
     Total_Test_Results                     int64
     Case_Fatality_Ratio                  float64
     Testing_Rate                         float64
     Resident Population 2020 Census       object
     Population Density 2020 Census       float64
     Density Rank 2020 Census               int64
     SexRatio                               int64
     dtype: object
```

```
[8]: df['Resident Population 2020 Census'] = df['Resident Population 2020 Census'].
      ↪str.replace(',','').astype(int)
```

```
[9]: df.isna().sum()
```

```
[9]: Id                                   0
     Day                                  0
     State ID                             0
     State                                0
     Lat                                  0
     Long_                                0
     Active                               0
     Incident_Rate                        0
     Total_Test_Results                   0
     Case_Fatality_Ratio                  0
```

```
Testing_Rate                      0
Resident Population 2020 Census   0
Population Density 2020 Census    0
Density Rank 2020 Census          0
SexRatio                          0
dtype: int64
```

[10]:

```
(df.iloc[:,4:]<0).sum()
```

[10]:

```
Lat                                 0
Long_                             150
Active                              0
Incident_Rate                       0
Total_Test_Results                  0
Case_Fatality_Ratio                 0
Testing_Rate                        0
Resident Population 2020 Census     0
Population Density 2020 Census      0
Density Rank 2020 Census            0
SexRatio                            0
dtype: int64
```

[11]:

```
df_gstate = df.groupby('State')
```

[33]:

```
for key,value in df_gstate:
    groups = df_gstate.get_group(key)
    temp = groups.iloc[:,4:]
    for columns in temp:
        Q1 = np.percentile(temp[columns],25)
        Q3 = np.percentile(temp[columns],75)
        IQR = Q3 - Q1
        right_limit = Q3 + 1.5*IQR
        left_limit = Q1 - 1.5*IQR
        outlier_right_index = groups[groups[columns] > right_limit][columns].
 ↪index
        outlier_left_index = groups[groups[columns] < left_limit][columns].index
        n_outliers = len(outlier_right_index) + len(outlier_left_index)
        if(n_outliers > 0):
            print(key,columns,n_outliers)
            df.loc[outlier_right_index,columns] = right_limit
            df.loc[outlier_left_index,columns] = left_limit
```

[13]:

```
X_test = df.iloc[:,4:]
```

[14]:

```
X_test = (X_test - X_test.mean())/ X_test.std()
```

[15]:

```
X_test
```

```
[15]:       Lat     Long_    Active  Incident_Rate  Total_Test_Results  \
     0    0.185758  0.685045  1.624921      -0.034555            0.871610
     1    0.185758  0.685045  1.636807      -0.019067            0.879419
     2    0.185758  0.685045  1.650350      -0.001535            0.887708
     3    0.185758  0.685045  1.668580       0.022167            0.902667
     4    0.185758  0.685045  1.688737       0.048398            0.916576
     ..        …       …       …            …                 …
     145  1.449287 -1.903393 -0.163129      -1.654112           -0.672606
     146  1.449287 -1.903393 -0.156348      -1.639579           -0.668592
     147  1.449287 -1.903393 -0.151034      -1.628091           -0.664255
     148  1.449287 -1.903393 -0.146139      -1.617734           -0.659315
     149  1.449287 -1.903393 -0.146139      -1.617734           -0.655189

          Case_Fatality_Ratio  Testing_Rate  Resident Population 2020 Census  \
     0               1.714293      1.248503                         0.093862
     1               1.725587      1.261108                         0.093862
     2               1.723482      1.274487                         0.093862
     3               1.733551      1.298632                         0.093862
     4               1.747687      1.321083                         0.093862
     ..                   …           …                              …
     145            -0.486315     -0.425243                        -0.463803
     146            -0.490357     -0.414460                        -0.463803
     147            -0.469195     -0.402811                        -0.463803
     148            -0.505139     -0.389542                        -0.463803
     149            -0.505139     -0.378461                        -0.463803

          Population Density 2020 Census  Density Rank 2020 Census  SexRatio
     0                          1.600344                 -1.387770 -0.557150
     1                          1.600344                 -1.387770 -0.557150
     2                          1.600344                 -1.387770 -0.557150
     3                          1.600344                 -1.387770 -0.557150
     4                          1.600344                 -1.387770 -0.557150
     ..                              …                       …       …
     145                       -0.351668                 -0.102798  1.114301
     146                       -0.351668                 -0.102798  1.114301
     147                       -0.351668                 -0.102798  1.114301
     148                       -0.351668                 -0.102798  1.114301
     149                       -0.351668                 -0.102798  1.114301

     [150 rows x 11 columns]
```

### 1.1 Preprocessing Completed

## 2 Label Recovered

```
[46]: classifier_RandomForest_recovered = RandomForestClassifier(max_depth =␣
       ↪3,n_estimators = 150, random_state=0)
```

```
[47]: classifier_RandomForest_recovered.fit(X_train,y_train.loc[:,'Recovered'])
```

```
[47]: RandomForestClassifier(max_depth=3, n_estimators=150, random_state=0)
```

```
[48]: y_pred_recovered = classifier_RandomForest_recovered.predict(X_test)
```

```
[49]: y_pred_recovered
```

```
[49]: array([1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
             1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
             1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
             1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
             1, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
             0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
             0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0])
```

### 2.1 Label: Deaths

```
[50]: classifier_RandomForest_deaths = RandomForestClassifier(max_depth =␣
       ↪5,n_estimators = 150, random_state=0)
```

```
[51]: classifier_RandomForest_deaths.fit(X_train,y_train.loc[:,'Deaths'])
```

```
[51]: RandomForestClassifier(max_depth=5, n_estimators=150, random_state=0)
```

```
[52]: y_pred_deaths = classifier_RandomForest_deaths.predict(X_test)
```

```
[53]: y_pred_deaths
```

```
[53]: array([1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
             1, 1, 1, 1, 1, 1, 1, 1, 0, 0, 0, 0, 1, 1, 0, 1, 1, 0, 0, 0, 0, 0,
             0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 1, 1, 1, 1, 1,
             1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
             1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
             1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
             1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1])
```

## 3 Label : Confirmed

[54]:
```
classifier_RandomForest_confirmed = RandomForestClassifier(max_depth = 5,n_estimators = 150, random_state=0)
```

[55]:
```
classifier_RandomForest_confirmed.fit(X_train,y_train.loc[:,'Confirmed'])
```

[55]:
```
RandomForestClassifier(max_depth=5, n_estimators=150, random_state=0)
```

[56]:
```
y_pred_confirmed = classifier_RandomForest_confirmed.predict(X_test)
```

[57]:
```
y_pred_confirmed.shape
```

[57]:
```
(150,)
```

[58]:
```
new_df = pd.DataFrame()
```

[59]:
```
new_df['Id'] = df.loc[:,'Id']
new_df['Confirmed'] = y_pred_confirmed
new_df['Deaths'] = y_pred_deaths
new_df['Recovered'] = y_pred_recovered
```

[60]:
```
new_df
```

[60]:
```
      Id  Confirmed  Deaths  Recovered
0      0          1       1          1
1      5          1       1          1
2     10          1       1          1
3     15          1       1          1
4     20          1       1          1
..   ...        ...     ...        ...
145  129          1       1          0
146  134          1       1          0
147  139          1       1          0
148  144          1       1          0
149  149          1       1          0

[150 rows x 4 columns]
```

[61]:
```
new_df.to_csv("Kagglepred_new.csv",index = False)
```

[ ]: