lab10

September 29, 2020

1 Lab 10

Manojkumar V K CSE17040

Kaggle link: https://www.kaggle.com/manojkumarvk/eda-ensemble-learning

[1]:
```
import numpy as np
import pandas as pd
import os
import seaborn as sns
import matplotlib.pyplot as plt
from collections import Counter
from plotly.offline import init_notebook_mode, iplot, plot
import plotly as py
import plotly.graph_objs as go
import plotly.figure_factory as ff

from mlxtend.classifier import EnsembleVoteClassifier
from sklearn.model_selection import train_test_split,cross_val_score
from sklearn.metrics import mean_squared_error, r2_score, recall_score, f1_score
from sklearn.metrics import confusion_matrix, accuracy_score, precision_score
from sklearn.ensemble import RandomForestRegressor, AdaBoostRegressor
from sklearn.ensemble import RandomForestClassifier, AdaBoostClassifier

import warnings
init_notebook_mode(connected=True)
warnings.filterwarnings("ignore")
plt.style.use('ggplot')

for dirname, _, filenames in os.walk('/kaggle/input'):
    for filename in filenames:
        print(os.path.join(dirname, filename))
```

```
/kaggle/input/world-happiness/2019.csv
/kaggle/input/world-happiness/2018.csv
/kaggle/input/world-happiness/2016.csv
/kaggle/input/world-happiness/2017.csv
/kaggle/input/world-happiness/2015.csv
```

[2]:
```
data1 = pd.read_csv("../input/world-happiness/2015.csv")
data1["year"]= 2015
data1.rename(columns={"Economy (GDP per Capita)":"Economy",
                      "Family":"Social support",
                      "Health (Life Expectancy)":"Health",
                      "Happiness Score":"Score"},inplace=True)

data2 = pd.read_csv("../input/world-happiness/2016.csv")
data2["year"]= 2016
data2.rename(columns={"Economy (GDP per Capita)":"Economy",
                      "Health (Life Expectancy)":"Health",
                      "Family":"Social support",
                      "Happiness Score":"Score"},inplace=True)

data3 = pd.read_csv("../input/world-happiness/2017.csv")
data3["year"] = 2017
data3.rename(columns={"Economy..GDP.per.Capita.":"Economy",
                      "Health..Life.Expectancy.":"Health",
                      "Family":"Social support",
                      "Happiness.Rank":"Happiness Rank",
                      "Happiness.Score":"Score"},inplace=True)

data4 = pd.read_csv("../input/world-happiness/2018.csv")
data4["year"] = 2018
data4.rename(columns={"Country or region":"Country",
                       "GDP per capita":"Economy",
                      "Healthy life expectancy":"Health",
                      "Freedom to make life choices":"Freedom",
                      "Overall rank":"Happiness Rank",
                      "Happiness.Score":"Score"},inplace=True)

data5 = pd.read_csv("../input/world-happiness/2019.csv")
data5["year"] = 2019
data5.rename(columns={"Country or region":"Country",
                       "GDP per capita":"Economy",
                      "Healthy life expectancy":"Health",
                      "Freedom to make life choices":"Freedom",
                      "Overall rank":"Happiness Rank",
                      "Happiness.Score":"Score"},inplace=True)
```

[3]:
```
happinessData = pd.concat([data1,data2,data3, data4, data5],join="inner")
happinessData.head()
```

[3]:

	Country	Happiness Rank	Score	Economy	Social support	Health	\
0	Switzerland	1	7.587	1.39651	1.34951	0.94143	
1	Iceland	2	7.561	1.30232	1.40223	0.94784	
2	Denmark	3	7.527	1.32548	1.36058	0.87464	

```
3       Norway                4  7.522  1.45900          1.33095  0.88521
4       Canada                5  7.427  1.32629          1.32261  0.90563

   Freedom  Generosity  year
0  0.66557     0.29678  2015
1  0.62877     0.43630  2015
2  0.64938     0.34139  2015
3  0.66973     0.34699  2015
4  0.63297     0.45811  2015
```

[4]:
```
happinessData.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 782 entries, 0 to 155
Data columns (total 9 columns):
 #   Column          Non-Null Count  Dtype
---  ------          --------------  -----
 0   Country         782 non-null    object
 1   Happiness Rank  782 non-null    int64
 2   Score           782 non-null    float64
 3   Economy         782 non-null    float64
 4   Social support  782 non-null    float64
 5   Health          782 non-null    float64
 6   Freedom         782 non-null    float64
 7   Generosity      782 non-null    float64
 8   year            782 non-null    int64
dtypes: float64(6), int64(2), object(1)
memory usage: 61.1+ KB
```

[5]:
```
df = happinessData.iloc[:100,:]
trace1 = go.Scatter(x = df['Happiness Rank'], y = df.Generosity, mode = "lines",
                    name = "Generosity", marker = dict(color = 'rgba(16, 112,␣
 ↪2, 0.8)'),
                    text= df.Country)
trace2 = go.Scatter(
                    x = df['Happiness Rank'],
                    y = df.Score,
                    mode = "lines+markers",
                    name = "Happiness_Score",
                    marker = dict(color = 'rgba(80, 26, 80, 0.8)'),    ␣
 ↪#isaretleme noktalama renkleri ve saydamligi 0.8 olacak
                    text= df.Country)                                  #grafikte␣
 ↪uzerinde gelince isim gorunsun diye kullaniliyor
data = [trace1, trace2]
layout = dict(title = 'Generosity and Happiness Score vs 100 countries\'␣
 ↪happiness rank',
```

```
            xaxis= dict(title= 'Happiness_Rank',ticklen= 5,zeroline= False)␣
↪   #zeroline sifirdan baslamak gosterilsin mi?
            )
fig = dict(data = data, layout = layout)
iplot(fig)
```

[6]:
```
df2015 = happinessData[happinessData.year == 2015].iloc[:100,:]           #␣
↪2014 2015 2016 ilk 100 al
df2016 = happinessData[happinessData.year == 2016].iloc[:100,:]
df2017 = happinessData[happinessData.year == 2017].iloc[:100,:]
trace1 =go.Scatter(
                    x = df2015['Happiness Rank'],
                    y = df2015.Generosity,
                    mode = "markers",
                    name = "2015",
                    marker = dict(color = 'rgba(255, 128, 255, 0.8)'),
                    text= df2015.Country)
trace2 =go.Scatter(
                    x = df2016['Happiness Rank'],
                    y = df2016.Generosity,
                    mode = "markers",
                    name = "2016",
                    marker = dict(color = 'rgba(255, 128, 2, 0.8)'),
                    text= df2016.Country)
trace3 =go.Scatter(
                    x = df2017['Happiness Rank'],
                    y = df2017.Generosity,
                    mode = "markers",
                    name = "2017",
                    marker = dict(color = 'rgba(0, 255, 200, 0.8)'),
                    text= df2017.Country)
data = [trace1, trace2, trace3]
layout = dict(title = 'Generosity vs Happiness_Rank of 100 Countries with 2015,␣
↪2016 and 2017 years',
              xaxis= dict(title= 'Happiness_Rank',ticklen= 5,zeroline= False),
              yaxis= dict(title= 'Generosity',ticklen= 5,zeroline= False)
             )
fig = dict(data = data, layout = layout)
iplot(fig)
```

[7]:
```
trace1 = go.Bar(
                x = df2015.Country,
                y = df2015.Generosity,
                name = "Generosity",
                marker = dict(color = 'rgba(255, 174, 255, 0.5)',
                              line=dict(color='rgb(0,0,0)',width=1.5)),
                text = df2015.Country)
```

```
trace2 = go.Bar(
                x = df2015.Country,
                y = df2015['Happiness Rank'],
                name = "Happiness_Rank",
                marker = dict(color = 'rgba(255, 255, 128, 0.5)',
                              line=dict(color='rgb(0,0,0)',width=1.5)),
                text = df2015.Country)
data = [trace1, trace2]
layout = go.Layout(barmode = "group")
fig = go.Figure(data = data, layout = layout)
iplot(fig)
```

[8]:
```
df2017 = happinessData[happinessData.year == 2017].iloc[:7,:]
pie1 = df2017.Freedom
labels = df2017.Country
fig = {
  "data": [
    {
      "values": pie1,
      "labels": labels,
      "domain": {"x": [0, .5]},
      "name": "Freedom Of Countries",
      "hoverinfo":"label+percent+name",
      "hole": .3,
      "type": "pie"
    },],
  "layout": {
        "title":"Countries rate of Freedom (2017)",
        "annotations": [
            { "font": { "size": 20},
              "showarrow": False,
              "text": "Freedom rate",
                "x": 0.135,
                "y": 1.1
            },
        ]
    }
}
iplot(fig)
```

[9]:
```
df2015 = happinessData[happinessData.year == 2015].iloc[:7,:]
pie1 = df2015.Freedom
labels = df2015.Country
fig = {
  "data": [
    {
      "values": pie1,
```

```
      "labels": labels,
      "domain": {"x": [0, .5]},
      "name": "Freedom Of Countries",
      "hoverinfo":"label+percent+name",
      "hole": .3,
      "type": "pie"
    },],
  "layout": {
        "title":"Countries rate of Freedom (2015)",
        "annotations": [
            { "font": { "size": 20},
              "showarrow": False,
              "text": "Freedom rate",
                "x": 0.135,
                "y": 1.1
            },
        ]
    }
}
iplot(fig)
```

[10]:
```
x = happinessData.iloc[:,3:]
y = happinessData["Score"]
x_train,x_test,y_train,y_test=train_test_split(x,y,test_size=0.
 ↪20,random_state=42)
print(y_train.shape)
print(y_test.shape)
```

```
(625,)
(157,)
```

[11]:
```
model = RandomForestRegressor(random_state=45)
model.fit(x_train,y_train)
pred=model.predict(x_test)
```

[12]:
```
print("R2 Score                : %0.3f" % r2_score(y_test,pred))
print("Root Mean Squared Error : %0.3f" % np.
 ↪sqrt(mean_squared_error(y_test,pred)))
print("Train Accuracy          : %0.3f" % model.score(x_train,y_train))
print("Test Accuracy           : %0.3f" % model.score(x_test,y_test))
```

```
R2 Score                : 0.801
Root Mean Squared Error : 0.492
Train Accuracy          : 0.973
Test Accuracy           : 0.801
```

[13]:
```
model = AdaBoostRegressor(n_estimators=100)
model.fit(x_train,y_train)
pred=model.predict(x_test)
```

[14]:
```
print("R2 Score                : %0.3f" % r2_score(y_test,pred))
print("Root Mean Squared Error : %0.3f" % np.
 ↪sqrt(mean_squared_error(y_test,pred)))
print("Train Accuracy          : %0.3f" % model.score(x_train,y_train))
print("Test Accuracy           : %0.3f" % model.score(x_test,y_test))
```

```
R2 Score                : 0.759
Root Mean Squared Error : 0.542
Train Accuracy          : 0.816
Test Accuracy           : 0.759
```

[ ]:
```

```

[15]:
```
mean = happinessData['Score'].mean()
happinessData['Happy'] = [False for _ in range(len(happinessData))]
happinessData.loc[happinessData['Score'] > mean, 'Happy'] = True
happinessData.head()
```

[15]:

	Country	Happiness Rank	Score	Economy	Social support	Health	Freedom	Generosity	year	Happy
0	Switzerland	1	7.587	1.39651	1.34951	0.94143	0.66557	0.29678	2015	True
1	Iceland	2	7.561	1.30232	1.40223	0.94784	0.62877	0.43630	2015	True
2	Denmark	3	7.527	1.32548	1.36058	0.87464	0.64938	0.34139	2015	True
3	Norway	4	7.522	1.45900	1.33095	0.88521	0.66973	0.34699	2015	True
4	Canada	5	7.427	1.32629	1.32261	0.90563	0.63297	0.45811	2015	True

[16]:
```
x = happinessData.iloc[:,3:].drop(columns = ['Happy'])
y = happinessData["Happy"]
x_train,x_test,y_train,y_test=train_test_split(x,y,test_size=0.
 ↪20,random_state=42)
print(y_train.shape)
print(y_test.shape)
```

```
(625,)
(157,)
```

[17]:
```
model = RandomForestClassifier(random_state=45)
model.fit(x_train,y_train)
pred=model.predict(x_test)
```

[18]:
```
print("Train Accuracy          : %0.3f" % model.score(x_train,y_train))
print("Test Accuracy           : %0.3f" % model.score(x_test,y_test))
print("Precision               : %0.3f" % precision_score(y_test, pred))
print("Recall                  : %0.3f" % recall_score(y_test, pred))
print("F1 Score                : %0.3f" % f1_score(y_test, pred))
```

```
Train Accuracy          : 1.000
Test Accuracy           : 0.917
Precision               : 0.926
Recall                  : 0.915
F1 Score                : 0.920
```

[19]:
```
confusion_matrix(y_test, pred)
```

[19]:
```
array([[69,  6],
       [ 7, 75]])
```

[20]:
```
model = AdaBoostClassifier(n_estimators=100)
model.fit(x_train,y_train)
pred=model.predict(x_test)
```

[21]:
```
print("Train Accuracy          : %0.3f" % model.score(x_train,y_train))
print("Test Accuracy           : %0.3f" % model.score(x_test,y_test))
print("Precision               : %0.3f" % precision_score(y_test, pred))
print("Recall                  : %0.3f" % recall_score(y_test, pred))
print("F1 Score                : %0.3f" % f1_score(y_test, pred))
```

```
Train Accuracy          : 0.944
Test Accuracy           : 0.892
Precision               : 0.901
Recall                  : 0.890
F1 Score                : 0.896
```

[22]:
```
confusion_matrix(y_test, pred)
```

[22]:
```
array([[67,  8],
       [ 9, 73]])
```

For the given dataset, Random Forest performs better than AdaBoost in both regressing the scores as well as classifying.

[23]:
```
clf1 = RandomForestClassifier(random_state=4)
clf2 = AdaBoostClassifier(n_estimators=100)
eclf = EnsembleVoteClassifier(clfs=[clf1, clf2], weights=[1, 1], voting='soft')
```

[24]:
```
labels = ['Random Forest', 'Ada Boost']

for clf, label in zip([clf1, clf2], labels):
    scores = cross_val_score(clf, x, y, cv=5,scoring='accuracy')
    print("Accuracy: %0.2f (+/- %0.2f) [%s]"
          % (scores.mean(), scores.std(), label))
```

```
Accuracy: 0.89 (+/- 0.03) [Random Forest]
Accuracy: 0.88 (+/- 0.02) [Ada Boost]
```

[2]:
```
!pip install nbconvert
!apt install pandoc
!apt install texlive-xetex -y
!jupyter nbconvert --execute --to pdf __notebook_source__.ipynb
```

```
Reading package lists… Done
Building dependency tree
Reading state information… Done
pandoc is already the newest version (1.19.2.4~dfsg-1build4).
0 upgraded, 0 newly installed, 0 to remove and 89 not upgraded.
Reading package lists… Done
Building dependency tree
Reading state information… Done
texlive-xetex is already the newest version (2017.20180305-1).
0 upgraded, 0 newly installed, 0 to remove and 89 not upgraded.
```

[ ]:
```

```