Quadratic Discriminant Analysis

December 22, 2022

[1]:
```
import pandas as pd
import matplotlib.pyplot as plt
import numpy as np

import sklearn.linear_model as skl_lm
import sklearn.discriminant_analysis as skl_da
from sklearn.model_selection import train_test_split

from sklearn.preprocessing import StandardScaler
```

0.0.1 QDA

[2]:
```
# creating the data set (initial columns + new ones that we used in the data␣
↪analysis)

d = pd.read_csv("train.csv")
d['Lead'].replace({'Male':1, 'Female':0}, inplace = True)


d["Number of words co-lead"] = d["Number of words lead"] - d["Difference in␣
↪words lead and co-lead"]
d = d.drop( ["Difference in words lead and co-lead"], axis = "columns")
```

[3]:
```
d
```

[3]:
```
      Number words female  Total words  Number of words lead  \
0                    1512         6394                2251.0
1                    1524         8780                2020.0
2                     155         4176                 942.0
3                    1073         9855                3440.0
4                    1317         7688                3835.0
…                       …            …                    …
1034                  303         2398                1334.0
1035                  632         8404                1952.0
1036                 1326         2750                 877.0
1037                  462         3994                 775.0
1038                 2735        11946                3410.0
```

```
      Number of male actors  Year  Number of female actors  Number words male  \
0                         2  1995                        5               2631
1                         9  2001                        4               5236
2                         7  1968                        1               3079
3                        12  2002                        2               5342
4                         8  1988                        4               2536
…                       …     …                        …                  …
1034                      5  1973                        2                761
1035                      6  1992                        2               5820
1036                      2  2000                        3                547
1037                      8  1996                        3               2757
1038                     13  2007                        4               5801

      Gross  Mean Age Male  Mean Age Female  Age Lead  Age Co-Lead  Lead  \
0     142.0      51.500000        42.333333      46.0         65.0     0
1      37.0      39.125000        29.333333      58.0         34.0     1
2     376.0      42.500000        37.000000      46.0         37.0     1
3      19.0      35.222222        21.500000      33.0         23.0     1
4      40.0      45.250000        45.000000      36.0         39.0     1
…     …             …              …        …        …   …
1034  174.0      43.200000        31.000000      46.0         24.0     1
1035  172.0      37.166667        24.000000      21.0         34.0     0
1036   53.0      27.500000        27.666667      28.0         25.0     1
1037   32.0      42.857143        38.500000      29.0         32.0     0
1038   32.0      44.090909        50.000000      38.0         48.0     1

      Number of words co-lead
0                      1908.0
1                       801.0
2                       155.0
3                       817.0
4                       686.0
…                       …
1034                    168.0
1035                   1765.0
1036                    521.0
1037                    723.0
1038                   1874.0

[1039 rows x 14 columns]
```

[4]:
```
lead = list()
colead = list()
femrest = list()
malerest = list()
```

```python
for i in range(1039):
    lead.append(d.iloc[i,2] / d.iloc[i,1])
    colead.append(d.iloc[i,13] / d.iloc[i,1])
    femrest.append( (d.iloc[i,0] / d.iloc[i,1]))
    malerest.append( (d.iloc[i,6] / d.iloc[i,1]))

d2 = pd.DataFrame( {"Lead words percentage":lead, "Colead words perentage":
 ↪colead, "Female word percentage":femrest, "Male word percentage":malerest} )

d["Lead words percentage"]=d2["Lead words percentage"]
d["Colead words perentage"]=d2["Colead words perentage"]
d["Female word percentage"]=d2["Female word percentage"]
d["Male word percentage"]=d2["Male word percentage"]

x = d[list(('Total words',
 'Number of male actors',
 'Number of female actors',
 'Mean Age Male',
 'Mean Age Female',
 'Age Lead',
 'Age Co-Lead',
 'Lead words percentage',
 'Colead words perentage',
 'Female word percentage'))]
y = d["Lead"]
```

[5]:
```python
from sklearn.model_selection import train_test_split

x_train, x_test, y_train, y_test = train_test_split(x, y, random_state = 4045)

scaler1 = StandardScaler()
scaler1.fit(x_train)

x_train = scaler1.transform(x_train)


x_test = scaler1.transform(x_test)

qda = skl_da.QuadraticDiscriminantAnalysis()
qda.fit(x_train, y_train)

qda = skl_da.QuadraticDiscriminantAnalysis()
qda.fit(x_train, y_train)
```

[5]: QuadraticDiscriminantAnalysis()

[6]:
```python
from sklearn.metrics import accuracy_score, precision_score, recall_score,␣
↪f1_score

print('Training set metrics:')
print('Accuracy:', accuracy_score(y_train, qda.predict(x_train)))
print('Precision:', precision_score(y_train, qda.predict(x_train)))
print('Recall:', recall_score(y_train, qda.predict(x_train)))
print('F1:', f1_score(y_train, qda.predict(x_train)))

print('\n')

print('Test set metrics:')
print('Accuracy:', accuracy_score(y_test, qda.predict(x_test)))
print('Precision:', precision_score(y_test, qda.predict(x_test)))
print('Recall:', recall_score(y_test, qda.predict(x_test)))
print('F1:', f1_score(y_test, qda.predict(x_test)))
```

```
Training set metrics:
Accuracy: 0.9409499358151476
Precision: 0.9548494983277592
Recall: 0.9677966101694915
F1: 0.9612794612794614


Test set metrics:
Accuracy: 0.9307692307692308
Precision: 0.9447236180904522
Recall: 0.9641025641025641
F1: 0.9543147208121828
```

[7]:
```python
from sklearn.inspection import permutation_importance
import seaborn as sns
from colour import Color
import eli5
from eli5.sklearn import PermutationImportance

perm = PermutationImportance(qda, random_state=1).fit(x_test, y_test)

perm_imp = permutation_importance(qda, x_test, y_test)

# View the feature scores as a dataframe to plot them:
feature_permutation_scores = pd.Series(perm_imp.importances_mean, index = x.
↪columns).sort_values(ascending = False)
feature_permutation_scores

# Normalise the feature scores to sum 1, so we can compare its relative␣
↪contribution to the model output change and compare it
```

```
# to the Gini importance scores.
normalized_feature_permutation_scores = feature_permutation_scores /␣
 ↪sum(feature_permutation_scores)

sns.set(font_scale = 6)

limegreen= Color("limegreen")
colors = list(limegreen.range_to(Color("red"),21))
colors = [color.rgb for color in colors]

f, ax = plt.subplots(figsize = (30, 24))
ax = sns.barplot(x = normalized_feature_permutation_scores, y =␣
 ↪normalized_feature_permutation_scores.index,palette = colors)
ax.set_title("Feature permutation importance",y = 1.03, fontsize = 95)
ax.set_xlabel("Feature importance score", fontsize = 95)
ax.xaxis.set_label_coords(0.5, -.07)

f.savefig('qda.svg', format = 'svg', dpi = 1200,  bbox_inches = 'tight',␣
 ↪transparent = True)
plt.show()
```