tktbmogzd

August 2, 2023

[ ]:
```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.linear_model import LogisticRegression
from sklearn.preprocessing import StandardScaler
```

[ ]:
```
from google.colab import drive
drive.mount('/content/drive')
```

```
Drive already mounted at /content/drive; to attempt to forcibly remount, call
drive.mount("/content/drive", force_remount=True).
```

[ ]:
```
df=pd.read_csv("/content/drive/MyDrive/mydatasets/C1_ionosphere.csv")
df
```

[ ]:

	1	0	0.99539	-0.05889	0.85243	0.02306	0.83398	-0.37708	1.1	\
0	1	0	1.00000	-0.18829	0.93035	-0.36156	-0.10868	-0.93597	1.00000	
1	1	0	1.00000	-0.03365	1.00000	0.00485	1.00000	-0.12062	0.88965	
2	1	0	1.00000	-0.45161	1.00000	1.00000	0.71216	-1.00000	0.00000	
3	1	0	1.00000	-0.02401	0.94140	0.06531	0.92106	-0.23255	0.77152	
4	1	0	0.02337	-0.00592	-0.09924	-0.11949	-0.00763	-0.11824	0.14706	
..	..	..	…	…	…	…	…	…	…	
345	1	0	0.83508	0.08298	0.73739	-0.14706	0.84349	-0.05567	0.90441	
346	1	0	0.95113	0.00419	0.95183	-0.02723	0.93438	-0.01920	0.94590	
347	1	0	0.94701	-0.00034	0.93207	-0.03227	0.95177	-0.03431	0.95584	
348	1	0	0.90608	-0.01657	0.98122	-0.01989	0.95691	-0.03646	0.85746	
349	1	0	0.84710	0.13533	0.73638	-0.06151	0.87873	0.08260	0.88928	

	0.03760	…	-0.51171	0.41078	-0.46168	0.21266	-0.34090	0.42267	\
0	-0.04549	…	-0.26569	-0.20468	-0.18401	-0.19040	-0.11593	-0.16626	
1	0.01198	…	-0.40220	0.58984	-0.22145	0.43100	-0.17365	0.60436	
2	0.00000	…	0.90695	0.51613	1.00000	1.00000	-0.20099	0.25682	
3	-0.16399	…	-0.65158	0.13290	-0.53206	0.02431	-0.62197	-0.05707	
4	0.06637	…	-0.01535	-0.03240	0.09223	-0.07859	0.00732	0.00000	
..	…	…	…	…	…	…	…	…	
345	-0.04622	…	-0.04202	0.83479	0.00123	1.00000	0.12815	0.86660	
346	0.01606	…	0.01361	0.93522	0.04925	0.93159	0.08168	0.94066	

```
347  0.02446  …   0.03193  0.92489   0.02542  0.92120   0.02242  0.92459
348  0.00110  …  -0.02099  0.89147  -0.07760  0.82983  -0.17238  0.96022
349 -0.09139  …  -0.15114  0.81147  -0.04822  0.78207  -0.00703  0.75747

     -0.54487  0.18641  -0.45300  g
0    -0.06288 -0.13738  -0.02447  b
1    -0.24180  0.56045  -0.38238  g
2     1.00000 -0.32382   1.00000  b
3    -0.59573 -0.04608  -0.65697  g
4     0.00000 -0.00039   0.12011  b
..        …       …        … ..
345  -0.10714  0.90546  -0.04307  g
346  -0.00035  0.91483   0.04712  g
347   0.00442  0.92697  -0.00577  g
348  -0.03757  0.87403  -0.16243  g
349  -0.06678  0.85764  -0.06151  g

[350 rows x 35 columns]
```

[ ]:
```
df.head()
```

[ ]:
```
   1  0  0.99539  -0.05889  0.85243  0.02306  0.83398  -0.37708      1.1  \
0  1  0  1.00000  -0.18829  0.93035 -0.36156 -0.10868  -0.93597  1.00000
1  1  0  1.00000  -0.03365  1.00000  0.00485  1.00000  -0.12062  0.88965
2  1  0  1.00000  -0.45161  1.00000  1.00000  0.71216  -1.00000  0.00000
3  1  0  1.00000  -0.02401  0.94140  0.06531  0.92106  -0.23255  0.77152
4  1  0  0.02337  -0.00592 -0.09924 -0.11949 -0.00763  -0.11824  0.14706

   0.03760  …  -0.51171  0.41078  -0.46168  0.21266  -0.34090  0.42267  \
0 -0.04549  …  -0.26569 -0.20468  -0.18401 -0.19040  -0.11593 -0.16626
1  0.01198  …  -0.40220  0.58984  -0.22145  0.43100  -0.17365  0.60436
2  0.00000  …   0.90695  0.51613   1.00000  1.00000  -0.20099  0.25682
3 -0.16399  …  -0.65158  0.13290  -0.53206  0.02431  -0.62197 -0.05707
4  0.06637  …  -0.01535 -0.03240   0.09223 -0.07859   0.00732  0.00000

   -0.54487  0.18641  -0.45300  g
0  -0.06288 -0.13738  -0.02447  b
1  -0.24180  0.56045  -0.38238  g
2   1.00000 -0.32382   1.00000  b
3  -0.59573 -0.04608  -0.65697  g
4   0.00000 -0.00039   0.12011  b

[5 rows x 35 columns]
```

1 Data Cleaning and Data Preprocessing

[ ]:
```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 350 entries, 0 to 349
Data columns (total 35 columns):
 #   Column     Non-Null Count  Dtype
---  ------     --------------  -----
 0   1          350 non-null    int64
 1   0          350 non-null    int64
 2   0.99539    350 non-null    float64
 3   -0.05889   350 non-null    float64
 4   0.85243    350 non-null    float64
 5   0.02306    350 non-null    float64
 6   0.83398    350 non-null    float64
 7   -0.37708   350 non-null    float64
 8   1.1        350 non-null    float64
 9   0.03760    350 non-null    float64
 10  0.85243.1  350 non-null    float64
 11  -0.17755   350 non-null    float64
 12  0.59755    350 non-null    float64
 13  -0.44945   350 non-null    float64
 14  0.60536    350 non-null    float64
 15  -0.38223   350 non-null    float64
 16  0.84356    350 non-null    float64
 17  -0.38542   350 non-null    float64
 18  0.58212    350 non-null    float64
 19  -0.32192   350 non-null    float64
 20  0.56971    350 non-null    float64
 21  -0.29674   350 non-null    float64
 22  0.36946    350 non-null    float64
 23  -0.47357   350 non-null    float64
 24  0.56811    350 non-null    float64
 25  -0.51171   350 non-null    float64
 26  0.41078    350 non-null    float64
 27  -0.46168   350 non-null    float64
 28  0.21266    350 non-null    float64
 29  -0.34090   350 non-null    float64
 30  0.42267    350 non-null    float64
 31  -0.54487   350 non-null    float64
 32  0.18641    350 non-null    float64
 33  -0.45300   350 non-null    float64
 34  g          350 non-null    object
dtypes: float64(32), int64(2), object(1)
memory usage: 95.8+ KB
```

[ ]:
```
df.describe()
```

[ ]:

	1	0	0.99539	-0.05889	0.85243	0.02306	\
count	350.000000	350.0	350.000000	350.000000	350.000000	350.000000	
mean	0.891429	0.0	0.640330	0.044667	0.600350	0.116154	
std	0.311546	0.0	0.498059	0.442032	0.520431	0.461443	
min	0.000000	0.0	-1.000000	-1.000000	-1.000000	-1.000000	
25%	1.000000	0.0	0.471517	-0.065388	0.412555	-0.024868	
50%	1.000000	0.0	0.870795	0.016700	0.808620	0.021170	
75%	1.000000	0.0	1.000000	0.194727	1.000000	0.335317	
max	1.000000	0.0	1.000000	1.000000	1.000000	1.000000	

	0.83398	-0.37708	1.1	0.03760	…	0.56811	\
count	350.000000	350.000000	350.000000	350.000000	…	350.000000	
mean	0.549284	0.120779	0.510453	0.181756	…	0.395643	
std	0.493124	0.520816	0.507117	0.484482	…	0.579206	
min	-1.000000	-1.000000	-1.000000	-1.000000	…	-1.000000	
25%	0.209105	-0.053483	0.086785	-0.049003	…	0.000000	
50%	0.728000	0.015085	0.682430	0.017550	…	0.549175	
75%	0.970445	0.451572	0.950555	0.536192	…	0.907165	
max	1.000000	1.000000	1.000000	1.000000	…	1.000000	

	-0.51171	0.41078	-0.46168	0.21266	-0.34090	0.42267	\
count	350.000000	350.000000	350.000000	350.000000	350.000000	350.000000	
mean	-0.069928	0.542015	-0.068417	0.378919	-0.027013	0.352313	
std	0.508675	0.516896	0.550411	0.576642	0.508425	0.572289	
min	-1.000000	-1.000000	-1.000000	-1.000000	-1.000000	-1.000000	
25%	-0.323745	0.283612	-0.428992	0.000000	-0.234935	0.000000	
50%	-0.014915	0.708530	-0.017685	0.499215	0.000000	0.446875	
75%	0.157922	0.999972	0.154862	0.884572	0.154218	0.859490	
max	1.000000	1.000000	1.000000	1.000000	1.000000	1.000000	

	-0.54487	0.18641	-0.45300
count	350.000000	350.000000	350.000000
mean	-0.002248	0.349829	0.015816
std	0.513491	0.523339	0.468338
min	-1.000000	-1.000000	-1.000000
25%	-0.239347	0.000000	-0.161013
50%	0.000000	0.413115	0.000000
75%	0.200935	0.816778	0.172105
max	1.000000	1.000000	1.000000

[8 rows x 34 columns]

[ ]:
```
df.columns
```

[ ]:
```
Index(['1', '0', '0.99539', '-0.05889', '0.85243', '0.02306', '0.83398',
       '-0.37708', '1.1', '0.03760', '0.85243.1', '-0.17755', '0.59755',
       '-0.44945', '0.60536', '-0.38223', '0.84356', '-0.38542', '0.58212',
       '-0.32192', '0.56971', '-0.29674', '0.36946', '-0.47357', '0.56811',
       '-0.51171', '0.41078', '-0.46168', '0.21266', '-0.34090', '0.42267',
       '-0.54487', '0.18641', '-0.45300', 'g'],
      dtype='object')
```

[ ]:
```
feature_matrix = df.iloc[:,0:34]
target_vector = df.iloc[:,-1]
```

[ ]:
```
fs = StandardScaler().fit_transform(feature_matrix)
logr = LogisticRegression()
logr.fit(fs,target_vector)
```

[ ]:
```
LogisticRegression()
```

[ ]:
```
observation=[[1.0,0.0,1.0,-0.18829,0.93035,
 -0.36156,
 -0.10868,
 -0.93597,
 1.0,
 -0.04549,
 0.50874,
 -0.67743,
 0.34432,
 -0.69707,
 -0.51685,
 -0.97515,
 0.05499,
 -0.62237,
 0.33109,
 -1.0,
 -0.13151,
 -0.453,
 -0.18056,
 -0.35734,
 -0.20332,
 -0.26569,
 -0.20468,
 -0.18401,
 -0.1904,
 -0.11593,
 -0.16626,
 -0.06288,
 -0.13738,
 -0.02447]]
```

```
prediction = logr.predict(observation)
print(prediction)
```

```
['g']
```

[ ]:
```
logr.classes_
```

[ ]:
```
array(['b', 'g'], dtype=object)
```

[ ]:
```
logr.predict_proba(observation)
```

[ ]:
```
array([[0.07006552, 0.92993448]])
```

Random Forest

[ ]:
```
df['g'].value_counts()
```

[ ]:
```
g    224
b    126
Name: g, dtype: int64
```

[ ]:
```
x=df.drop('g', axis=1)
y=df['g']
```

[ ]:
```
g1={"g":{"g":1, "b":2}}
df=df.replace(g1)
df
```

[ ]:
```
      1  0  0.99539  -0.05889  0.85243  0.02306  0.83398  -0.37708      1.1  \
0     1  0  1.00000  -0.18829  0.93035 -0.36156 -0.10868  -0.93597  1.00000
1     1  0  1.00000  -0.03365  1.00000  0.00485  1.00000  -0.12062  0.88965
2     1  0  1.00000  -0.45161  1.00000  1.00000  0.71216  -1.00000  0.00000
3     1  0  1.00000  -0.02401  0.94140  0.06531  0.92106  -0.23255  0.77152
4     1  0  0.02337  -0.00592 -0.09924 -0.11949 -0.00763  -0.11824  0.14706
..   .. ..      ...       ...      ...      ...      ...       ...      ...
345   1  0  0.83508   0.08298  0.73739 -0.14706  0.84349  -0.05567  0.90441
346   1  0  0.95113   0.00419  0.95183 -0.02723  0.93438  -0.01920  0.94590
347   1  0  0.94701  -0.00034  0.93207 -0.03227  0.95177  -0.03431  0.95584
348   1  0  0.90608  -0.01657  0.98122 -0.01989  0.95691  -0.03646  0.85746
349   1  0  0.84710   0.13533  0.73638 -0.06151  0.87873   0.08260  0.88928

     0.03760  …  -0.51171  0.41078  -0.46168  0.21266  -0.34090  0.42267  \
0   -0.04549  …  -0.26569 -0.20468  -0.18401 -0.19040  -0.11593 -0.16626
1    0.01198  …  -0.40220  0.58984  -0.22145  0.43100  -0.17365  0.60436
2    0.00000  …   0.90695  0.51613   1.00000  1.00000  -0.20099  0.25682
3   -0.16399  …  -0.65158  0.13290  -0.53206  0.02431  -0.62197 -0.05707
4    0.06637  …  -0.01535 -0.03240   0.09223 -0.07859   0.00732  0.00000
..       …  …       …       …        …       …        …       …
```

```
345 -0.04622  …  -0.04202  0.83479   0.00123  1.00000   0.12815  0.86660
346  0.01606  …   0.01361  0.93522   0.04925  0.93159   0.08168  0.94066
347  0.02446  …   0.03193  0.92489   0.02542  0.92120   0.02242  0.92459
348  0.00110  …  -0.02099  0.89147  -0.07760  0.82983  -0.17238  0.96022
349 -0.09139  …  -0.15114  0.81147  -0.04822  0.78207  -0.00703  0.75747

     -0.54487  0.18641  -0.45300  g
0    -0.06288 -0.13738  -0.02447  2
1    -0.24180  0.56045  -0.38238  1
2     1.00000 -0.32382   1.00000  2
3    -0.59573 -0.04608  -0.65697  1
4     0.00000 -0.00039   0.12011  2
..        …       …        … ..
345  -0.10714  0.90546  -0.04307  1
346  -0.00035  0.91483   0.04712  1
347   0.00442  0.92697  -0.00577  1
348  -0.03757  0.87403  -0.16243  1
349  -0.06678  0.85764  -0.06151  1

[350 rows x 35 columns]
```

[ ]:
```
from sklearn.model_selection import train_test_split
x_train,x_test,y_train,y_test=train_test_split(x,y,train_size=0.70)
```

[ ]:
```
from sklearn.ensemble import RandomForestClassifier
rfc = RandomForestClassifier()
rfc.fit(x_train,y_train)
```

[ ]:
```
RandomForestClassifier()
```

[ ]:
```
parameters = {'max_depth':[1,2,3,4,5],'min_samples_leaf':[5,10,15,20,25],
              'n_estimators': [10,20,30,40,50]
              }
```

[ ]:
```
from sklearn.model_selection import GridSearchCV
grid_search =␣
 ↪GridSearchCV(estimator=rfc,param_grid=parameters,cv=2,scoring="accuracy")
grid_search.fit(x_train,y_train)
```

[ ]:
```
GridSearchCV(cv=2, estimator=RandomForestClassifier(),
             param_grid={'max_depth': [1, 2, 3, 4, 5],
                         'min_samples_leaf': [5, 10, 15, 20, 25],
                         'n_estimators': [10, 20, 30, 40, 50]},
             scoring='accuracy')
```

[ ]:
```
grid_search.best_score_
```

[ ]:
```
0.9344262295081966
```

[ ]:
```
rfc_best = grid_search.best_estimator_
```

[ ]:
```
from sklearn.tree import plot_tree
plt.figure(figsize=(89,40))
plot_tree(rfc_best.estimators_[5], feature_names=x.columns, class_names=['Yes',␣
↪'No'], filled=True)
```

[ ]:
```
[Text(0.42105263157894735, 0.9166666666666666, '0.42267 <= 0.12\ngini =
0.474\nsamples = 157\nvalue = [94, 150]\nclass = No'),
 Text(0.21052631578947367, 0.75, '0.42267 <= -0.018\ngini = 0.39\nsamples =
50\nvalue = [58, 21]\nclass = Yes'),
 Text(0.10526315789473684, 0.5833333333333334, '0.83398 <= 0.346\ngini =
0.5\nsamples = 29\nvalue = [19, 20]\nclass = No'),
 Text(0.05263157894736842, 0.4166666666666667, 'gini = 0.26\nsamples = 8\nvalue
= [11, 2]\nclass = Yes'),
 Text(0.15789473684210525, 0.4166666666666667, '0.42267 <= -0.732\ngini =
0.426\nsamples = 21\nvalue = [8, 18]\nclass = No'),
 Text(0.10526315789473684, 0.25, 'gini = 0.32\nsamples = 7\nvalue = [8,
2]\nclass = Yes'),
 Text(0.21052631578947367, 0.25, 'gini = 0.0\nsamples = 14\nvalue = [0,
16]\nclass = No'),
 Text(0.3157894736842105, 0.5833333333333334, '0.60536 <= -0.018\ngini =
0.049\nsamples = 21\nvalue = [39, 1]\nclass = Yes'),
 Text(0.2631578947368421, 0.4166666666666667, 'gini = 0.219\nsamples = 5\nvalue
= [7, 1]\nclass = Yes'),
 Text(0.3684210526315789, 0.4166666666666667, 'gini = 0.0\nsamples = 16\nvalue =
[32, 0]\nclass = Yes'),
 Text(0.631578947368421, 0.75, '-0.34090 <= -0.934\ngini = 0.341\nsamples =
107\nvalue = [36, 129]\nclass = No'),
 Text(0.5263157894736842, 0.5833333333333334, '-0.44945 <= -0.932\ngini =
0.278\nsamples = 11\nvalue = [15, 3]\nclass = Yes'),
 Text(0.47368421052631576, 0.4166666666666667, 'gini = 0.0\nsamples = 5\nvalue =
[7, 0]\nclass = Yes'),
 Text(0.5789473684210527, 0.4166666666666667, 'gini = 0.397\nsamples = 6\nvalue
= [8, 3]\nclass = Yes'),
 Text(0.7368421052631579, 0.5833333333333334, '0.99539 <= 0.133\ngini =
0.245\nsamples = 96\nvalue = [21, 126]\nclass = No'),
 Text(0.6842105263157895, 0.4166666666666667, 'gini = 0.0\nsamples = 5\nvalue =
[8, 0]\nclass = Yes'),
 Text(0.7894736842105263, 0.4166666666666667, '0.85243 <= 0.995\ngini =
0.17\nsamples = 91\nvalue = [13, 126]\nclass = No'),
 Text(0.6842105263157895, 0.25, '0.42267 <= 0.433\ngini = 0.042\nsamples =
61\nvalue = [2, 92]\nclass = No'),
 Text(0.631578947368421, 0.08333333333333333, 'gini = 0.147\nsamples = 19\nvalue
= [2, 23]\nclass = No'),
```

```
 Text(0.7368421052631579, 0.08333333333333333, 'gini = 0.0\nsamples = 42\nvalue
= [0, 69]\nclass = No'),
 Text(0.8947368421052632, 0.25, '-0.05889 <= 0.348\ngini = 0.369\nsamples =
30\nvalue = [11, 34]\nclass = No'),
 Text(0.8421052631578947, 0.08333333333333333, 'gini = 0.234\nsamples =
24\nvalue = [5, 32]\nclass = No'),
 Text(0.9473684210526315, 0.08333333333333333, 'gini = 0.375\nsamples = 6\nvalue
= [6, 2]\nclass = Yes')]
```