SVM & Decision Trees

November 25, 2022

0.1 Zjazd 4 - 19.11.2022 - Cezary Graban s21752, Paweł Iwiński s19771

Decision tree & SVM

Notebook ilustrujący klasyfikacje dla dwóch datasetów przy użyciu drzew dycyzyjnych oraz SVC

[ ]:
```
import pandas as pd
import seaborn as sns
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler, LabelEncoder
from sklearn.svm import SVC
from sklearn.metrics import accuracy_score
from sklearn.tree import DecisionTreeClassifier
```

0.1.1 Dataset 1 - prediction of whether or not an object is a mine or a rock given the strength of sonar returns at different angles.

Load and parse the data with features into dataframe object.

[ ]:
```
new_column_names = []
for i in range(60):
    new_column_names.append(f"Feature {i}")

new_column_names.append("Class")
df = pd.read_csv('sonar.all-data.csv', names=new_column_names)
train = df.replace({'Class': {'M': 0, 'R': 1}})
x = train.drop("Class",axis=1)
y = train['Class']
scaler = StandardScaler()
x = scaler.fit_transform(x)
df
```

[ ]:
```
     Feature 0  Feature 1  Feature 2  Feature 3  Feature 4  Feature 5  \
0       0.0200     0.0371     0.0428     0.0207     0.0954     0.0986
1       0.0453     0.0523     0.0843     0.0689     0.1183     0.2583
2       0.0262     0.0582     0.1099     0.1083     0.0974     0.2280
3       0.0100     0.0171     0.0623     0.0205     0.0205     0.0368
4       0.0762     0.0666     0.0481     0.0394     0.0590     0.0649
..         …          …          …          …          …          …
```

```
203     0.0187     0.0346     0.0168     0.0177     0.0393     0.1630
204     0.0323     0.0101     0.0298     0.0564     0.0760     0.0958
205     0.0522     0.0437     0.0180     0.0292     0.0351     0.1171
206     0.0303     0.0353     0.0490     0.0608     0.0167     0.1354
207     0.0260     0.0363     0.0136     0.0272     0.0214     0.0338

     Feature 6  Feature 7  Feature 8  Feature 9  …  Feature 51  Feature 52  \
0       0.1539     0.1601     0.3109     0.2111  …      0.0027      0.0065
1       0.2156     0.3481     0.3337     0.2872  …      0.0084      0.0089
2       0.2431     0.3771     0.5598     0.6194  …      0.0232      0.0166
3       0.1098     0.1276     0.0598     0.1264  …      0.0121      0.0036
4       0.1209     0.2467     0.3564     0.4459  …      0.0031      0.0054
..         …          …          …          …  …         …           …
203     0.2028     0.1694     0.2328     0.2684  …      0.0116      0.0098
204     0.0990     0.1018     0.1030     0.2154  …      0.0061      0.0093
205     0.1257     0.1178     0.1258     0.2529  …      0.0160      0.0029
206     0.1465     0.1123     0.1945     0.2354  …      0.0086      0.0046
207     0.0655     0.1400     0.1843     0.2354  …      0.0146      0.0129

     Feature 53  Feature 54  Feature 55  Feature 56  Feature 57  Feature 58  \
0        0.0159      0.0072      0.0167      0.0180      0.0084      0.0090
1        0.0048      0.0094      0.0191      0.0140      0.0049      0.0052
2        0.0095      0.0180      0.0244      0.0316      0.0164      0.0095
3        0.0150      0.0085      0.0073      0.0050      0.0044      0.0040
4        0.0105      0.0110      0.0015      0.0072      0.0048      0.0107
..          …           …           …           …           …           …
203      0.0199      0.0033      0.0101      0.0065      0.0115      0.0193
204      0.0135      0.0063      0.0063      0.0034      0.0032      0.0062
205      0.0051      0.0062      0.0089      0.0140      0.0138      0.0077
206      0.0126      0.0036      0.0035      0.0034      0.0079      0.0036
207      0.0047      0.0039      0.0061      0.0040      0.0036      0.0061

     Feature 59  Class
0        0.0032      R
1        0.0044      R
2        0.0078      R
3        0.0117      R
4        0.0094      R
..          …    …
203      0.0157      M
204      0.0067      M
205      0.0031      M
206      0.0048      M
207      0.0115      M

[208 rows x 61 columns]
```

Describe the data

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

```
[ ]:          Feature 0   Feature 1   Feature 2   Feature 3   Feature 4   Feature 5  \
count   208.000000  208.000000  208.000000  208.000000  208.000000  208.000000
mean      0.029164    0.038437    0.043832    0.053892    0.075202    0.104570
std       0.022991    0.032960    0.038428    0.046528    0.055552    0.059105
min       0.001500    0.000600    0.001500    0.005800    0.006700    0.010200
25%       0.013350    0.016450    0.018950    0.024375    0.038050    0.067025
50%       0.022800    0.030800    0.034300    0.044050    0.062500    0.092150
75%       0.035550    0.047950    0.057950    0.064500    0.100275    0.134125
max       0.137100    0.233900    0.305900    0.426400    0.401000    0.382300

          Feature 6   Feature 7   Feature 8   Feature 9  …  Feature 50  \
count   208.000000  208.000000  208.000000  208.000000  …  208.000000
mean      0.121747    0.134799    0.178003    0.208259  …    0.016069
std       0.061788    0.085152    0.118387    0.134416  …    0.012008
min       0.003300    0.005500    0.007500    0.011300  …    0.000000
25%       0.080900    0.080425    0.097025    0.111275  …    0.008425
50%       0.106950    0.112100    0.152250    0.182400  …    0.013900
75%       0.154000    0.169600    0.233425    0.268700  …    0.020825
max       0.372900    0.459000    0.682800    0.710600  …    0.100400

        Feature 51  Feature 52  Feature 53  Feature 54  Feature 55  Feature 56  \
count   208.000000  208.000000  208.000000  208.000000  208.000000  208.000000
mean      0.013420    0.010709    0.010941    0.009290    0.008222    0.007820
std       0.009634    0.007060    0.007301    0.007088    0.005736    0.005785
min       0.000800    0.000500    0.001000    0.000600    0.000400    0.000300
25%       0.007275    0.005075    0.005375    0.004150    0.004400    0.003700
50%       0.011400    0.009550    0.009300    0.007500    0.006850    0.005950
75%       0.016725    0.014900    0.014500    0.012100    0.010575    0.010425
max       0.070900    0.039000    0.035200    0.044700    0.039400    0.035500

        Feature 57  Feature 58  Feature 59
count   208.000000  208.000000  208.000000
mean      0.007949    0.007941    0.006507
std       0.006470    0.006181    0.005031
min       0.000300    0.000100    0.000600
25%       0.003600    0.003675    0.003100
50%       0.005800    0.006400    0.005300
75%       0.010350    0.010325    0.008525
max       0.044000    0.036400    0.043900

[8 rows x 60 columns]
```

Definition of the function for calculation of accuracy.

[ ]:
```
def calculate_accuracy(x, y, svm_kernel="linear"):
    """Calculate the accuracy for a given data and labels.

    Args:
        x (dataframe): Dataframe containing info.
        y (dataframe): Labels for classification
        svm_kernel (str, optional): Kernel used for SVM. Defaults to "linear".
    """
    x_train, x_test, y_train, y_test = train_test_split(x, y, test_size = 0.33,␣
 ↪stratify=y)
    svc = SVC(kernel=svm_kernel)
    svc.fit(x_train, y_train)
    y_pred_svc = svc.predict(x_test)
    svc_accuracy = round(accuracy_score(y_test, y_pred_svc)*100, 8)
    print(f"SVM Kernel used: {svm_kernel}")
    print('SVC accuracy = ', svc_accuracy,'%')

    clf = DecisionTreeClassifier()
    clf = clf.fit(x_train, y_train)
    y_pred = clf.predict(x_test)
    print("Decision tree accuracy:", round(accuracy_score(y_test, y_pred)*100,␣
 ↪8))
    print("\n")
```

Train the models with different kernels.

[ ]:
```
calculate_accuracy(x, y, svm_kernel="linear")
calculate_accuracy(x, y, svm_kernel="rbf")
calculate_accuracy(x, y, svm_kernel="sigmoid")
```

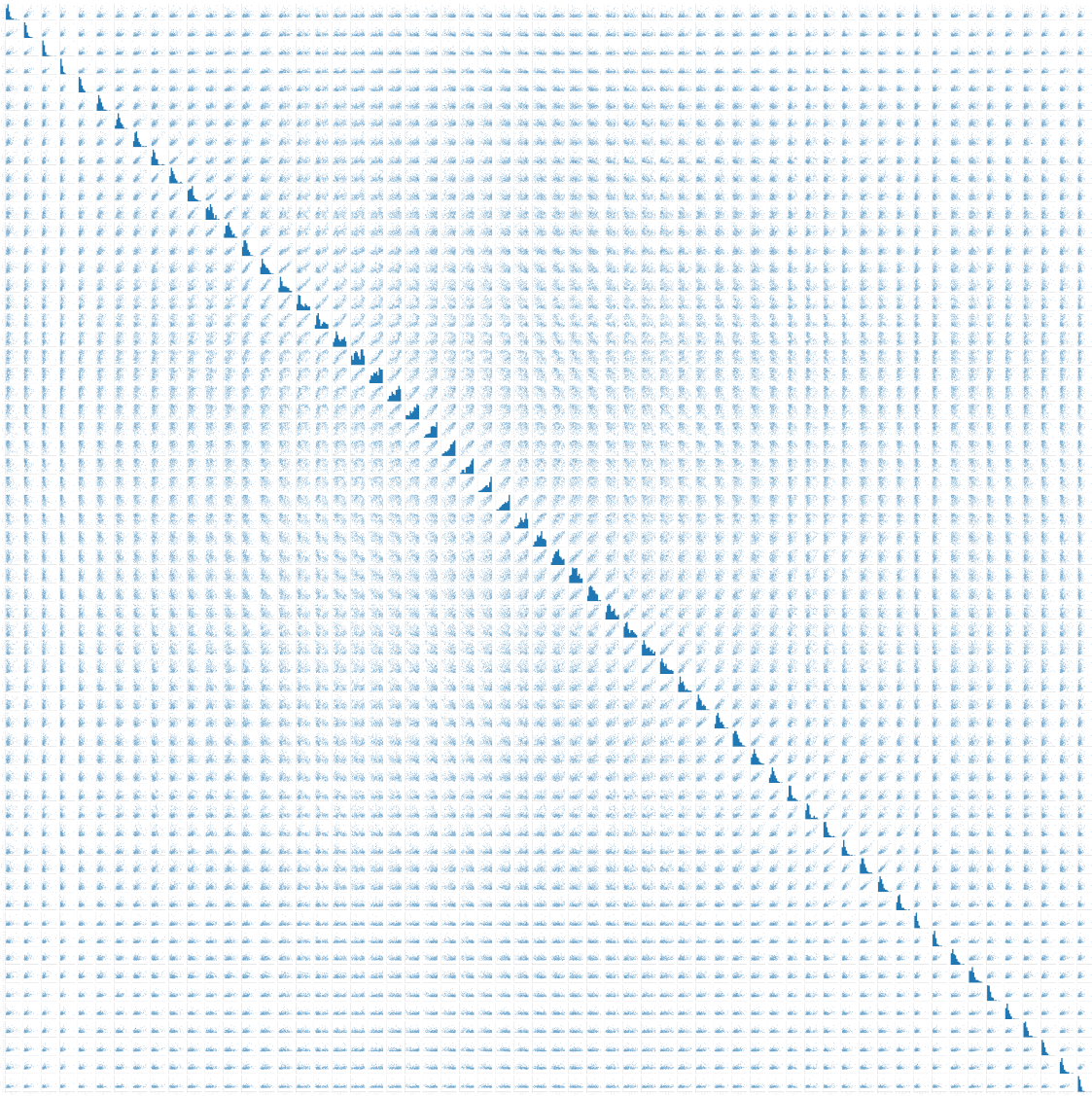
```
SVM Kernel used: linear
SVC accuracy =  76.8115942 %
Decision tree accuracy: 68.11594203


SVM Kernel used: rbf
SVC accuracy =  85.50724638 %
Decision tree accuracy: 73.91304348


SVM Kernel used: sigmoid
SVC accuracy =  79.71014493 %
Decision tree accuracy: 73.91304348


c:\Users\Pawel\Anaconda3\lib\site-packages\sklearn\svm\base.py:193:
FutureWarning: The default value of gamma will change from 'auto' to 'scale' in
```

```
version 0.22 to account better for unscaled features. Set gamma explicitly to
'auto' or 'scale' to avoid this warning.
  "avoid this warning.", FutureWarning)
c:\Users\Pawel\Anaconda3\lib\site-packages\sklearn\svm\base.py:193:
FutureWarning: The default value of gamma will change from 'auto' to 'scale' in
version 0.22 to account better for unscaled features. Set gamma explicitly to
'auto' or 'scale' to avoid this warning.
  "avoid this warning.", FutureWarning)
```

Create plots out of data.

[ ]:
```
# Wiem ze to glupie zeby robic kazdy mozliwy plot pokolei, ale nie chodzi o␣
 ↪sens, chodzi o pokazanie ze sie da.
# W problemach klasyfikacji, gdy danych jest duzo ficzerow robienie plotów␣
 ↪wszystko-od-wszystkiego mija sie z celem, lepiej
# jest liczyc korelacje miedzy danymi ficzerami i wtedy pokazywac korelacje na␣
 ↪wykresie.
sns.pairplot(df, size=3).savefig("plot_all_dataset_1.png")
```

```
c:\Users\Pawel\Anaconda3\lib\site-packages\seaborn\axisgrid.py:2065:
UserWarning: The `size` parameter has been renamed to `height`; pleaes update
your code.
  warnings.warn(msg, UserWarning)
```



0.1.2 Dataset 2 - information about people used to predict whether we should give them a loan or not depending on data.

Load and transform the dataset

[ ]:
```
df = pd.read_csv('loan_train_data.csv')
df['Credit_History'] = df['Credit_History'].astype('O')  # change type
df.drop('Loan_ID', axis=1, inplace=True)  # useless, not needed


df.isnull().sum().sort_values(ascending=False)

# remove + agg missing data
```

```
cat_data = []
num_data = []
for i,c in enumerate(df.dtypes):
    if c == object:
        cat_data.append(df.iloc[:, i])
    else :
        num_data.append(df.iloc[:, i])
cat_data = pd.DataFrame(cat_data).transpose()
num_data = pd.DataFrame(num_data).transpose()
cat_data = cat_data.apply(lambda x:x.fillna(x.value_counts().index[0]))
cat_data.isnull().sum().any() # no more missing data
num_data.fillna(method='bfill', inplace=True)
num_data.isnull().sum().any() # no more missing data

df.describe()
```

[ ]:

	ApplicantIncome	CoapplicantIncome	LoanAmount	Loan_Amount_Term
count	614.000000	614.000000	592.000000	600.00000
mean	5403.459283	1621.245798	146.412162	342.00000
std	6109.041673	2926.248369	85.587325	65.12041
min	150.000000	0.000000	9.000000	12.00000
25%	2877.500000	0.000000	100.000000	360.00000
50%	3812.500000	1188.500000	128.000000	360.00000
75%	5795.000000	2297.250000	168.000000	360.00000
max	81000.000000	41667.000000	700.000000	480.00000

Clean the data and create labels for classification to be a 0, 1 choice.

[ ]:
```
le = LabelEncoder()
target_values = {'Y': 0 , 'N' : 1}

target = cat_data['Loan_Status']
cat_data.drop('Loan_Status', axis=1, inplace=True)

target = target.map(target_values)
for i in cat_data:
    cat_data[i] = le.fit_transform(cat_data[i])
df = pd.concat([cat_data, num_data, target], axis=1)
df
```

[ ]:

	Gender	Married	Dependents	Education	Self_Employed	Credit_History	\
0	1	0	0	0	0	1	
1	1	1	1	0	0	1	
2	1	1	0	0	1	1	
3	1	1	0	1	0	1	
4	1	0	0	0	0	1	
..	…	…	…	…	…	…	

```
609       0        0           0          0              0               1
610       1        1           3          0              0               1
611       1        1           1          0              0               1
612       1        1           2          0              0               1
613       0        0           0          0              1               0

     Property_Area  ApplicantIncome  CoapplicantIncome  LoanAmount  \
0                2           5849.0                0.0       128.0
1                0           4583.0             1508.0       128.0
2                2           3000.0                0.0        66.0
3                2           2583.0             2358.0       120.0
4                2           6000.0                0.0       141.0
..             …              …                 …          …
609              0           2900.0                0.0        71.0
610              0           4106.0                0.0        40.0
611              2           8072.0              240.0       253.0
612              2           7583.0                0.0       187.0
613              1           4583.0                0.0       133.0

     Loan_Amount_Term  Loan_Status
0               360.0            0
1               360.0            1
2               360.0            0
3               360.0            0
4               360.0            0
..                …           …
609             360.0            0
610             180.0            0
611             360.0            0
612             360.0            0
613             360.0            1

[614 rows x 12 columns]
```

Calculate the accuracy for training data and labels with different SVC kernels

[ ]:
```
x = pd.concat([cat_data, num_data], axis=1)
y = target

calculate_accuracy(x, y, svm_kernel="linear")
calculate_accuracy(x, y, svm_kernel="rbf")
calculate_accuracy(x, y, svm_kernel="sigmoid")
```

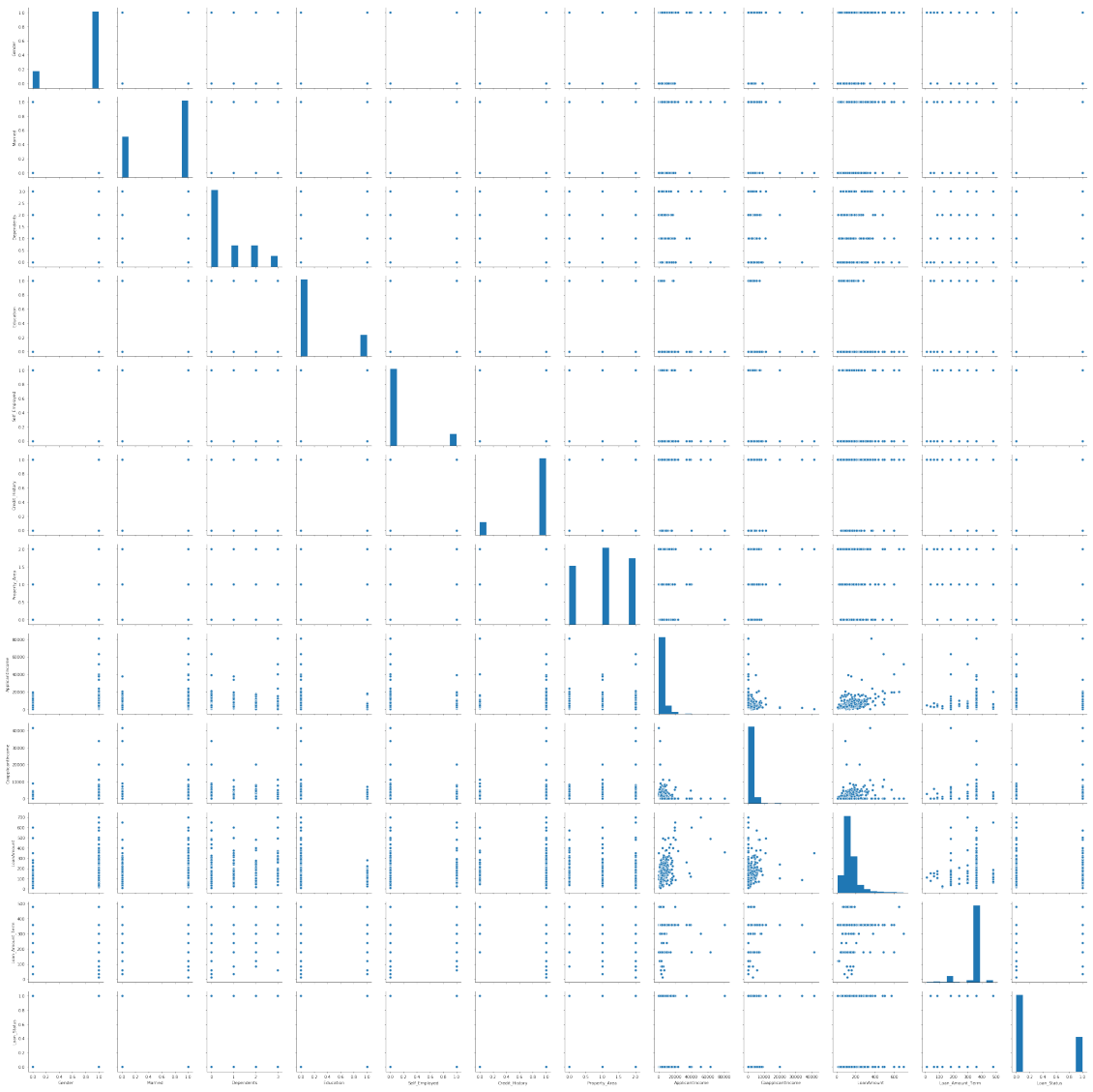
```
SVM Kernel used: linear
SVC accuracy =  75.86206897 %
Decision tree accuracy: 69.95073892
```

```
SVM Kernel used: rbf
SVC accuracy =  68.96551724 %
Decision tree accuracy: 69.95073892


SVM Kernel used: sigmoid
SVC accuracy =  68.96551724 %
Decision tree accuracy: 72.90640394



c:\Users\Pawel\Anaconda3\lib\site-packages\sklearn\svm\base.py:193:
FutureWarning: The default value of gamma will change from 'auto' to 'scale' in
version 0.22 to account better for unscaled features. Set gamma explicitly to
'auto' or 'scale' to avoid this warning.
  "avoid this warning.", FutureWarning)
c:\Users\Pawel\Anaconda3\lib\site-packages\sklearn\svm\base.py:193:
FutureWarning: The default value of gamma will change from 'auto' to 'scale' in
version 0.22 to account better for unscaled features. Set gamma explicitly to
'auto' or 'scale' to avoid this warning.
  "avoid this warning.", FutureWarning)
```

Create the plots out of data - everything against everything

[ ]:
```
sns.pairplot(df, size=3).savefig("plot_all_dataset_2.png")
```

```
c:\Users\Pawel\Anaconda3\lib\site-packages\seaborn\axisgrid.py:2065:
UserWarning: The `size` parameter has been renamed to `height`; pleaes update
your code.
  warnings.warn(msg, UserWarning)
```


Gender
Married
Dependents
Education
Self_Employed
Credit_History
Property_Area
ApplicantIncome
CoapplicantIncome
LoanAmount
Loan_Amount_Term
Loan_Status