ex7

May 31, 2022

0.1 Exercise sheet 7

Cüneyt Erem 3277992 s6cuerem@uni-bonn.de

Nkeh Victor Ndiwago 3504121 s0vinkeh@uni-bonn.de

Paula Romero Jiménez 3320220 s0parome@uni-bonn.de

0.1.1 Exercise 1

[ ]:
```
import pandas as pd
import numpy as np
```

[ ]:
```
df = pd.read_csv('schizophrenia_labels.tsv', sep='\t')
df.head()
```

[ ]:

	Unnamed: 0	FileName	Target
0	0	GSM317486.CEL	control
1	1	GSM317484.CEL	control
2	2	GSM317482.CEL	schizophrenia
3	3	GSM317481.CEL	schizophrenia
4	4	GSM317480.CEL	control

[ ]:
```
class_label=[]

for col in df["Target"]:
    if "control" in col:
        class_label.append(0)
    elif "schizophrenia" in col:
        class_label.append(1)
```

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

df2 = pd.read_csv('schizophrenia_data.tsv', sep='\t')
df2 = df2.T
df2.columns = class_label
```

```
X = df2.iloc[1:,:]
X = X.transpose()

y = class_label
X.head()
```

[ ]:

	DDR1	RFC2	HSPA6	PAX8	GUCA1A	UBA7	THRA	\
0	10.543733	5.385013	5.535427	7.265465	3.678816	6.733502	5.323099	
0	9.078255	5.974958	5.790271	7.896282	4.02971	6.62998	5.465493	
1	10.348714	5.697933	5.27822	7.667168	3.913032	6.658972	5.391923	
1	10.022377	5.859198	5.388661	7.753764	3.989167	6.34045	5.454207	
0	8.912777	6.028037	5.204214	7.83574	3.976735	6.344254	5.593455	

	PTPN21	CCL5	CYP2E1	…	ACTB.2	ACTB.3	ACTB.4	\
0	4.307191	3.951761	3.682025	…	12.048716	11.389709	12.106053	
0	4.323722	4.06232	4.139489	…	11.972652	12.214945	12.882387	
1	4.050861	3.471891	3.790555	…	12.312299	11.724446	12.49331	
1	3.978568	3.685096	3.94453	…	12.223028	11.653218	12.497715	
0	3.986807	3.668305	4.203255	…	12.179574	11.905463	12.551949	

	GAPDH.3	GAPDH.4	GAPDH.5	STAT1.2	STAT1.3	STAT1.4	STAT1.5
0	12.343496	11.103524	11.857728	7.351567	3.537313	5.817774	5.565078
0	12.478911	11.762979	12.602796	7.342315	3.882709	6.420142	6.006399
1	12.376871	11.282427	12.016803	8.444306	3.498873	6.347346	6.079468
1	12.450958	11.440124	12.213041	7.232103	3.37116	5.88041	5.481548
0	12.650134	11.741904	12.42236	7.78382	3.481569	6.253778	5.844453

[5 rows x 21074 columns]

---

part a and d)

```
from sklearn.linear_model import LogisticRegression

X_train, X_test, y_train, y_test = train_test_split(X, y, random_state=1,␣
 ↪test_size=0.3, train_size=0.7)

log_reg = LogisticRegression(random_state=1, penalty="none")
log_reg.fit(X_train, y_train)

test_score_a = log_reg.score(X_test, y_test)
train_score_a = log_reg.score(X_train, y_train)

print("test_score: ", test_score_a)
print("train_score: ", train_score_a)
```

[ ]:

```
test_score:  0.6190476190476191
train_score:  1.0
```

---

part b and d)

[ ]:
```
l1_train = []
l1_test = []

for i in [0.001, 0.01, 0.1, 1, 10, 100]:
    log_reg = LogisticRegression(random_state=1, penalty='l1', C=1/i,␣
 ↪solver='liblinear')
    log_reg.fit(X_train, y_train)

    test_score = log_reg.score(X_test, y_test)
    train_score = log_reg.score(X_train, y_train)

    l1_test.append(test_score)
    l1_train.append(train_score)

    print("test_score for ", i, " is: ", test_score)
    print("train_scorefor ", i, " is: ", train_score)
    print()
```

```
test_score for  0.001  is:  0.5714285714285714
train_scorefor  0.001  is:  1.0

test_score for  0.01  is:  0.5714285714285714
train_scorefor  0.01  is:  1.0

test_score for  0.1  is:  0.5714285714285714
train_scorefor  0.1  is:  1.0

test_score for  1  is:  0.5714285714285714
train_scorefor  1  is:  1.0

test_score for  10  is:  0.42857142857142855
train_scorefor  10  is:  0.5208333333333334

test_score for  100  is:  0.42857142857142855
train_scorefor  100  is:  0.5208333333333334
```

---

part c and d)

[ ]:
```
l2_train = []
l2_test = []

for i in [0.001, 0.01, 0.1, 1, 10, 100]:
    log_reg = LogisticRegression(max_iter=5000, random_state=1, penalty='l2',␣
 ↪C=1/i)
    log_reg.fit(X_train, y_train)

    test_score = log_reg.score(X_test, y_test)
    train_score = log_reg.score(X_train, y_train)

    l2_test.append(test_score)
    l2_train.append(train_score)

    print("test_score for ", i, " is: ", test_score)
    print("train_scorefor ", i, " is: ", train_score)
    print()
```

```
test_score for  0.001  is:  0.6190476190476191
train_scorefor  0.001  is:  1.0

test_score for  0.01  is:  0.5714285714285714
train_scorefor  0.01  is:  1.0

test_score for  0.1  is:  0.5714285714285714
train_scorefor  0.1  is:  1.0

test_score for  1  is:  0.5714285714285714
train_scorefor  1  is:  1.0

test_score for  10  is:  0.6666666666666666
train_scorefor  10  is:  1.0

test_score for  100  is:  0.6666666666666666
train_scorefor  100  is:  1.0
```

---

part d)

every score for training and test sets are written above after fit by logistic regression as given above

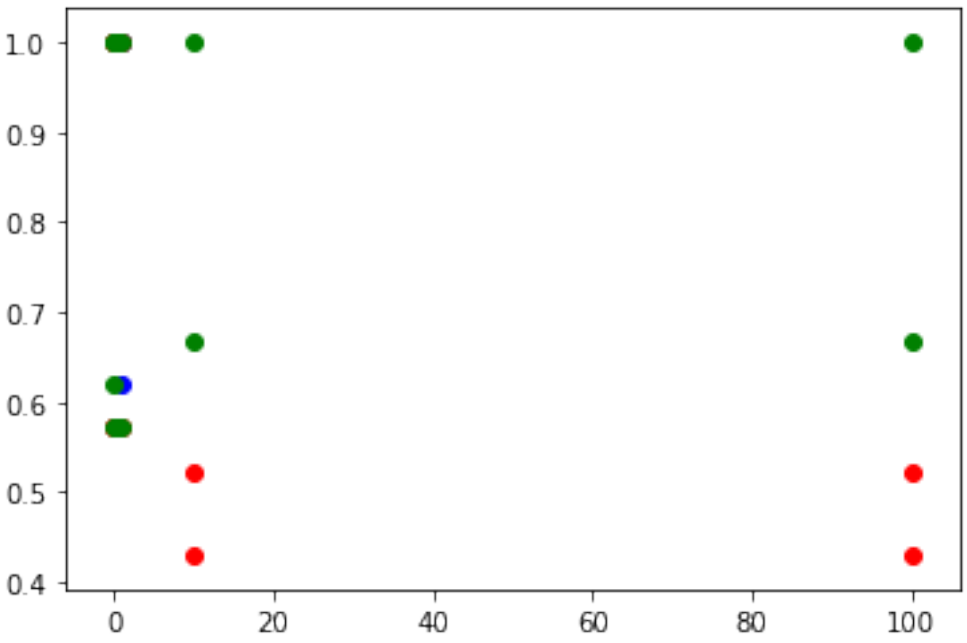
---

part e)

[ ]:
```
import matplotlib.pyplot as plt

plt.scatter(1, test_score_a, color="blue")
```

```
plt.scatter(1, train_score_a, color="blue")

plt.scatter([0.001, 0.01, 0.1, 1, 10, 100], l1_train, color="red", vmin=0.001,␣
↪vmax=100)
plt.scatter([0.001, 0.01, 0.1, 1, 10, 100], l1_test, color="red", vmin=0.001,␣
↪vmax=100)

plt.scatter([0.001, 0.01, 0.1, 1, 10, 100], l2_train, color="green", vmin=0.
↪001, vmax=100)
plt.scatter([0.001, 0.01, 0.1, 1, 10, 100], l2_test, color="green", vmin=0.001,␣
↪vmax=100)
```

[ ]: <matplotlib.collections.PathCollection at 0x7ff7bec5ad90>



---

part f)

for the test scores, for l2 = 10 and l2 = 100 is: 0.6666666666666666 gives best results

---

0.1.2 Exercise 2

1. Using the diabetes.csv dataset, train a logistic regression model with elastic net penalization to demonstrate the pros and cons of the different data splitting methods and give a short description on what you observe.

[ ]:
```
import pandas as pd
path= "diabetes.csv"
df=pd.read_csv(path)
#print(df.head())
df
```

[ ]:
```
     Pregnancies  Glucose  BloodPressure  SkinThickness  Insulin   BMI  \
0              6      148             72             35        0  33.6
1              1       85             66             29        0  26.6
2              8      183             64              0        0  23.3
3              1       89             66             23       94  28.1
4              0      137             40             35      168  43.1
..           ...      ...            ...            ...      ...  ...
763           10      101             76             48      180  32.9
764            2      122             70             27        0  36.8
765            5      121             72             23      112  26.2
766            1      126             60              0        0  30.1
767            1       93             70             31        0  30.4

     DiabetesPedigreeFunction  Age  Outcome
0                       0.627   50        1
1                       0.351   31        0
2                       0.672   32        1
3                       0.167   21        0
4                       2.288   33        1
..                        ...  ...      ...
763                     0.171   63        0
764                     0.340   27        0
765                     0.245   30        0
766                     0.349   47        1
767                     0.315   23        0

[768 rows x 9 columns]
```

[ ]:
```
from numpy import mean
from numpy import std
from numpy import absolute
#from pandas import read_csv
from sklearn.model_selection import cross_val_score
from sklearn.model_selection import RepeatedKFold
from sklearn.linear_model import ElasticNet
# load dataset
data = df.values
X, y = data[:, :-1], data[:, -1]
# define model
model = ElasticNet(alpha=1.0, l1_ratio=0.5)
# define model evaluation method
```

```
cv = RepeatedKFold(n_splits=5, n_repeats=3, random_state=1)
# evaluate model
scores = cross_val_score(model, X, y, scoring='neg_mean_absolute_error', cv=cv,␣
 ↪n_jobs=-1)
# force scores to be positive
scores = absolute(scores)
print('Mean MAE: %.3f (%.3f)' % (mean(scores), std(scores)))
```

```
Mean MAE: 0.361 (0.011)
```

0.1.3 Exercise 3 - SVM (8 points)

1. Inform yourself about SVM and briefly explain the working strategy of linear SVM and why maximizing the margin is a good strategy. (2 points)

SVM stands for Support Vector Machine algorithm. The basic idea of this algorithm is to find the optimal hyperplane, a decision boundary that differentiates classes. The margin is the distance between the support vectors (data points near the hyperplane) and the hyperplane.

The strategy of this algorith is to find the maximun margin, this way it is going to be more robust when applied to additional data in the future. If we have a bigger distance between the classes, more data is going to fit in this hyperplane separation.

2. Inform yourself about the non-linearity problem for classifiers. Briefly explain how SVM uses kernel trick to overcome this issue. (2 points)

When we only have 2 dimensions, the hyperplane is going to be linear. However, the data could be not linearly separable, and we would need to project the data in a higher dimension. To do this, there is a kernel transformation, in which another dimension Z is created. Plotting this feature Z with respect to x will give us linearly separable data.

In the kernel transformation we can choose between linear, gaussian / RBF or polynomial.

3. Using the diabetes dataset from Question 2.1 answer the following questions:

a. Carry out SVM on PCA reduced dataset (n_components=2) and report the accuracy of the model using 80% of the reduced data for the model training. (1 point)

[ ]:
```
import numpy as np
import pandas as pd
from sklearn.decomposition import PCA
from sklearn import svm
from sklearn.metrics import accuracy_score
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler
from seaborn import load_dataset, pairplot
from matplotlib import pyplot as plt
from sklearn.model_selection import cross_val_score
from sklearn.model_selection import KFold
from sklearn.inspection import DecisionBoundaryDisplay
```

[ ]:
```
# We load the dataset and divide it into features and labels to work with it
dataset = pd.read_csv('diabetes.csv')
dataset.head()
```

[ ]:

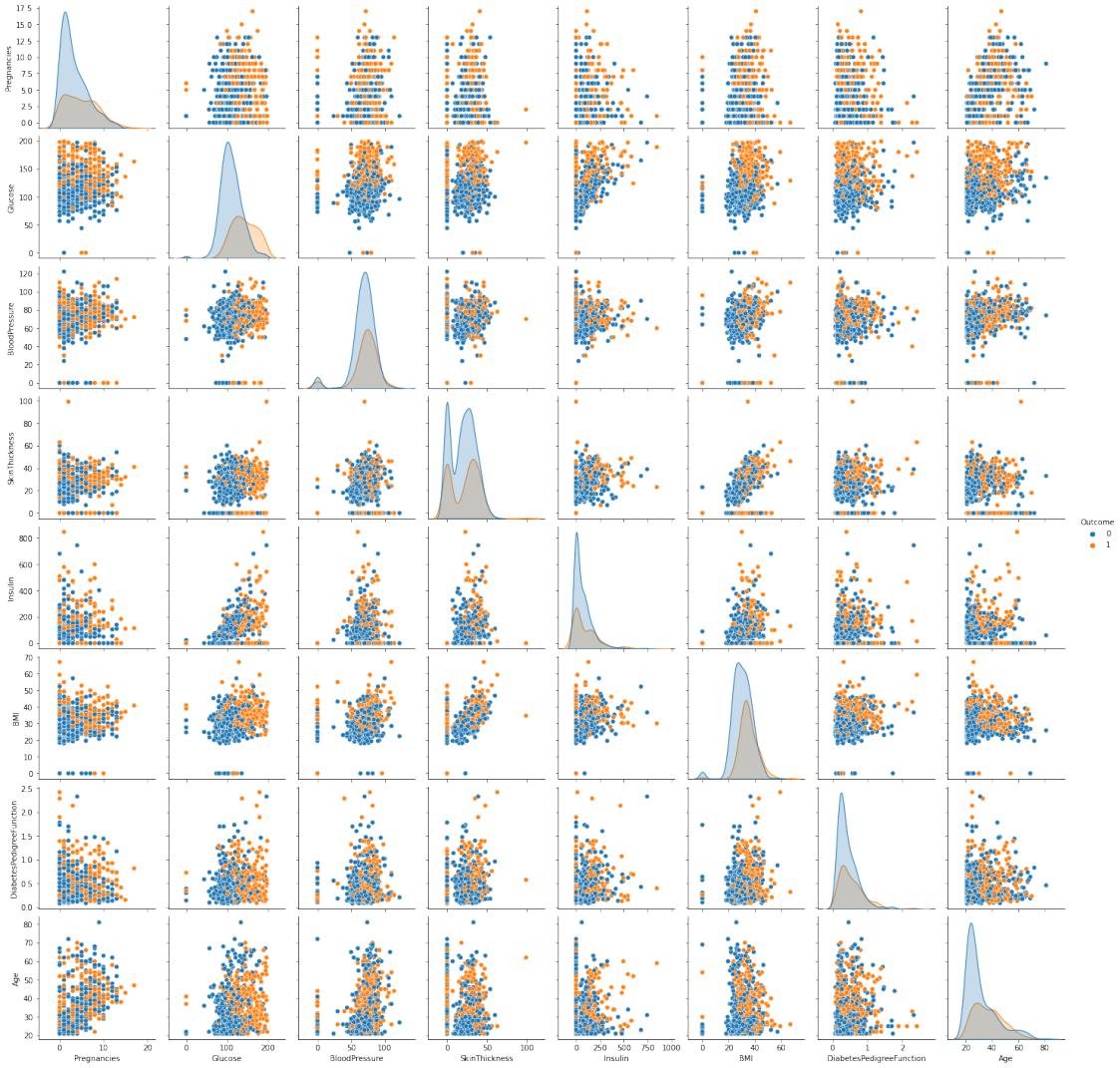
	Pregnancies	Glucose	BloodPressure	SkinThickness	Insulin	BMI	\
0	6	148	72	35	0	33.6	
1	1	85	66	29	0	26.6	
2	8	183	64	0	0	23.3	
3	1	89	66	23	94	28.1	
4	0	137	40	35	168	43.1	

	DiabetesPedigreeFunction	Age	Outcome
0	0.627	50	1
1	0.351	31	0
2	0.672	32	1
3	0.167	21	0
4	2.288	33	1

[ ]:
```
dataset.isnull().values.any()
```

[ ]: False

[ ]:
```
# We explore the data
pairplot(dataset, hue='Outcome')
plt.show()
```



[ ]:
```
features = dataset.drop(['Outcome'], axis=1) #everything but the Outcome column
labels = dataset['Outcome']
```

[ ]:
```
# Divide the dataset into 80% training and 20% test
X_train, X_test, y_train, y_test = train_test_split(features, labels,␣
 ↪test_size=0.20, random_state=0)

# We normalize the data
scaler = StandardScaler()
X_train = scaler.fit_transform(X_train)
X_test = scaler.transform (X_test)
```

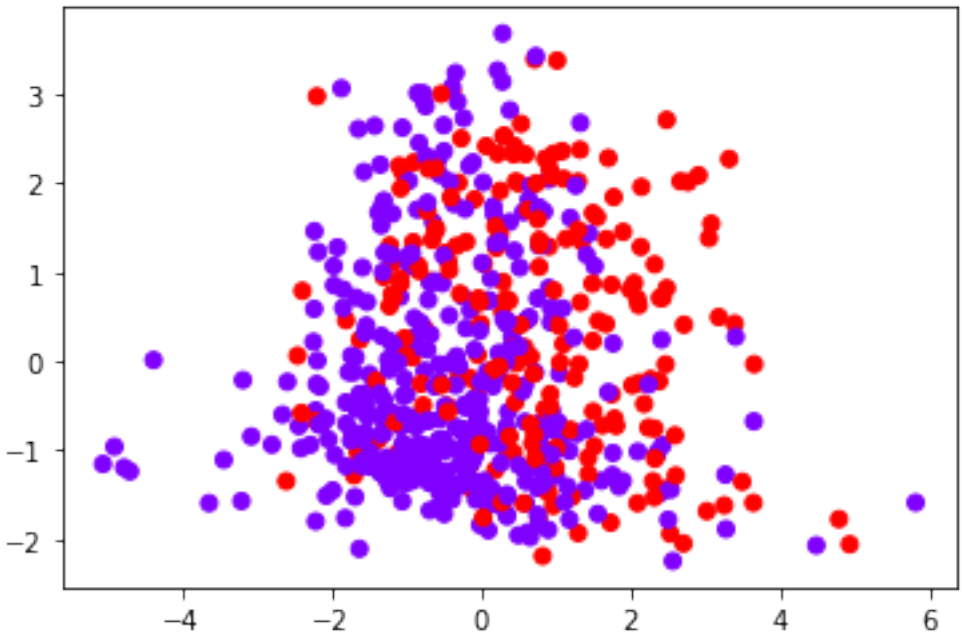
[ ]:
```
# We have to reduce the diabetes dataset into 2 PCAs
pca_model = PCA(n_components=2)
```

```
X_train_pca = pca_model.fit_transform(X_train)
X_test_pca = pca_model.transform(X_test)
```

[ ]:
```
# Perform SVM classification on this reduced dataset and we report the accuracy␣
  ↪for each type of kernel
for k in ('linear', 'poly', 'rbf'):
    model = svm.SVC(kernel=k)
    model.fit(X_train_pca, y_train)
    y_pred = model.predict(X_test_pca)
    print(k)
    print(accuracy_score(y_test, y_pred))
```

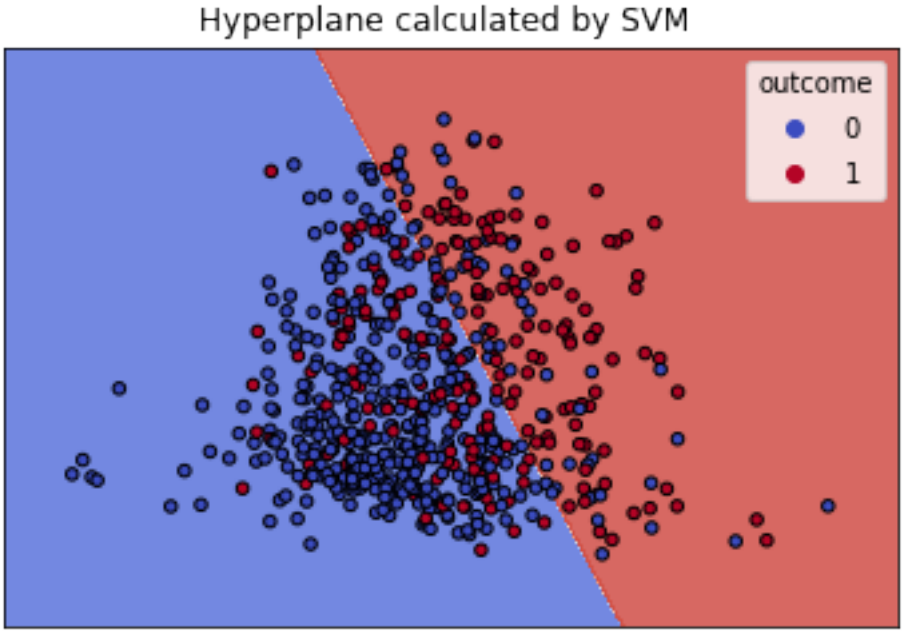
```
linear
0.7597402597402597
poly
0.7337662337662337
rbf
0.7532467532467533
```

[ ]:
```
plt.scatter(X_train_pca[:,0], X_train_pca[:,1], c= y_train, cmap='rainbow' )
plt.show()
```



b. Plot the training data along with their class labels and depict the hyperplane calculated by SVM in question 3.3a. (1 point) Hint: A similar plot can be found in scikit-learn's tutorials.

[ ]:
```
def make_meshgrid(x, y, h=.02):
    x_min, x_max = x.min() - 1, x.max() + 1
    y_min, y_max = y.min() - 1, y.max() + 1
    xx, yy = np.meshgrid(np.arange(x_min, x_max, h), np.arange(y_min, y_max, h))
    return xx, yy

def plot_contours(ax, clf, xx, yy, **params):
    Z = clf.predict(np.c_[xx.ravel(), yy.ravel()])
    Z = Z.reshape(xx.shape)
    out = ax.contourf(xx, yy, Z, **params)
    return out

model = svm.SVC(kernel='linear')
clf = model.fit(X_train_pca, y_train)

fig, ax = plt.subplots()
# title for the plots
title = ('Hyperplane calculated by SVM ')
# Set-up grid for plotting.
X0, X1 = X_train_pca[:, 0], X_train_pca[:, 1]
xx, yy = make_meshgrid(X0, X1)
plot_contours(ax, clf, xx, yy, cmap=plt.cm.coolwarm, alpha=0.8)
scat = ax.scatter(X0, X1, c=y_train, cmap=plt.cm.coolwarm, s=20, edgecolors='k')
legend1 = ax.legend(*scat.legend_elements(), loc="upper right", title="outcome")
ax.set_xticks(())
ax.set_yticks(())
ax.set_title(title)
plt.show()
```



c. Compare and report the difference in the performance of the SVM model and the logistic regression model created in Question 2.1b by carrying out KFold cross validation with 5 splits on the original diabetes dataset and plotting a boxplot of the accuracy. (2 points)

[ ]:
```
# prepare the cross-validation procedure
cv = KFold(n_splits=5, random_state=1, shuffle=True)
# create model
model = svm.SVC()
# evaluate model
scores = cross_val_score(model, features, labels, scoring='accuracy', cv=cv,␣
 ↪n_jobs=-1)
# report performance
print('Accuracy: %.3f (%.3f)' % (np.mean(scores), np.std(scores)))
```

```
Accuracy: 0.764 (0.013)
```

[ ]:
```

```