Exercises6

December 4, 2020

1 Exercise Set 6

1.1 Answered to all problems (1-5)

2 Question 1

a) Proportion of success $\hat{p} = 0.95$

$$\hat{p} \pm z\sqrt{\frac{\hat{p}(1-\hat{p})}{100}}$$

For 90% CI, $z = 1.65$. Therefore

$$0.95 \pm 1.65\sqrt{\frac{0.95 * 0.05}{100}}$$

$\therefore$ 90% CI for accuracy $= [0.914, 0.986]$

b) $\hat{p}_1 = 0.95$, $\hat{p}_2 = 0.97$

$$H_0 : \hat{p}_1 = \hat{p}_2$$

$$H_1 : \hat{p}_1 \neq \hat{p}_2$$

Test statistic:

$$z = \frac{\hat{p}_1 - \hat{p}_2}{\sqrt{\hat{p}(1-\hat{p})(\frac{1}{100} + \frac{1}{100})}}$$

where $\hat{p} = \frac{95+97}{100+100} = 0.96$. Therefore

$$z = \frac{0.95 - 0.97}{\sqrt{0.96 * 0.04 * \frac{1}{50}}} = -0.72$$

Because of $-1.65 < -0.72 < 1.65$, there is no statistical difference between the models at 90% confidence level.

If we do a one-tailed test instead of two-tailed, we get similar result: no statistical difference ($-1.28 < -0.72$).

3 Question 2

a)

$$\begin{aligned}
\mathbf{w} &= \lambda(\Sigma_0 + \Sigma_1 + \lambda I)^{-1}(\mu_\mathbf{1} - \mu_\mathbf{0}) \\
&= 100 * (\begin{pmatrix} 4 & 1 \\ 1 & 2 \end{pmatrix} + \begin{pmatrix} 3 & -1 \\ -1 & 2 \end{pmatrix} + \begin{pmatrix} 100 & 0 \\ 0 & 100 \end{pmatrix})^{-1} * (\begin{pmatrix} 0 \\ 0 \end{pmatrix} - \begin{pmatrix} 1 \\ 1 \end{pmatrix}) \\
&= 100 * \frac{1}{107 * 104} \begin{pmatrix} 104 & 0 \\ 0 & 107 \end{pmatrix} \begin{pmatrix} -1 \\ -1 \end{pmatrix} \\
&= 100 * \begin{pmatrix} 1/107 & 0 \\ 0 & 1/104 \end{pmatrix} \begin{pmatrix} -1 \\ -1 \end{pmatrix} \\
&= -\begin{pmatrix} 100/107 \\ 25/26 \end{pmatrix}
\end{aligned}$$

b)

$$\Sigma_0 + \Sigma_1 = \begin{pmatrix} a & b \\ c & d \end{pmatrix}, \lambda I = \begin{pmatrix} \lambda & 0 \\ 0 & \lambda \end{pmatrix}, \mu_\mathbf{1} - \mu_\mathbf{0} = \begin{pmatrix} t_0 \\ t_1 \end{pmatrix}$$

$$\begin{aligned}
\mathbf{w} &= \lambda(\Sigma_0 + \Sigma_1 + \lambda I)^{-1}(\mu_\mathbf{1} - \mu_\mathbf{0}) \\
&= (\begin{pmatrix} a & b \\ c & d \end{pmatrix} + \begin{pmatrix} \lambda & 0 \\ 0 & \lambda \end{pmatrix})^{-1} \begin{pmatrix} t_0 \\ t_1 \end{pmatrix} \\
&= \frac{\lambda}{(a+\lambda)(d+\lambda) - cb} \begin{pmatrix} d+\lambda & -b \\ -c & a+\lambda \end{pmatrix} \begin{pmatrix} t_0 \\ t_1 \end{pmatrix}
\end{aligned}$$

Calculate few limits:

It can be easily seen that:

$$\lim_{\lambda \to \infty} \frac{\lambda(d+\lambda)}{(a+\lambda)(d+\lambda) - cb} = \lim_{\lambda \to \infty} \frac{d\lambda + \lambda^2}{ad + (a+d)\lambda + \lambda^2 - cb} = 1$$

Similarly:

$$\lim_{\lambda \to \infty} \frac{\lambda(a+\lambda)}{(a+\lambda)(d+\lambda) - cb} = 1$$

$$\lim_{\lambda \to \infty} \frac{\lambda}{(a+\lambda)(d+\lambda) - cb} = 0$$

Therefore:

$$
\begin{aligned}
\lim_{\lambda\to\infty} \mathbf{w} &= \left(\lim_{\lambda\to\infty} \frac{\lambda}{(a+\lambda)(d+\lambda)-cb}\begin{pmatrix} d+\lambda & -b \\ -c & a+\lambda \end{pmatrix}\right)\begin{pmatrix} t_0 \\ t_1 \end{pmatrix} \\
&= \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix}\begin{pmatrix} t_0 \\ t_1 \end{pmatrix} \\
&= \begin{pmatrix} t_0 \\ t_1 \end{pmatrix} \\
&= \mu_1 - \mu_0
\end{aligned}
$$

4 Question 3

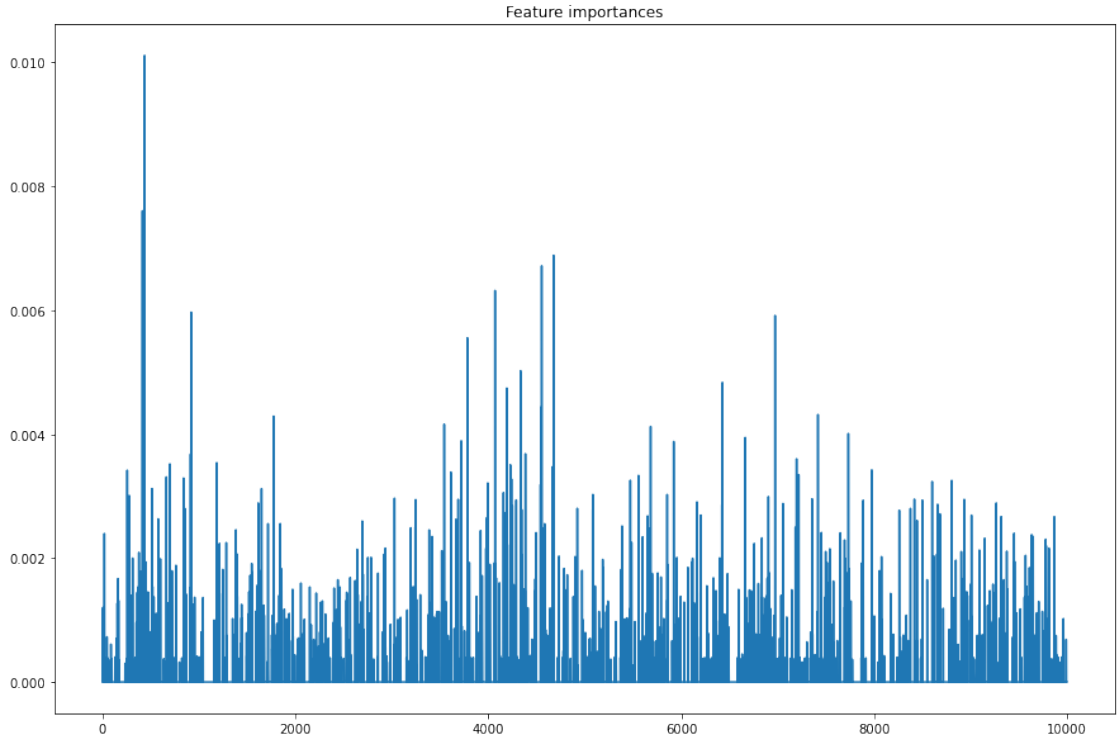
[1]:
```
from scipy.io import loadmat
from sklearn.ensemble import RandomForestClassifier
from sklearn.metrics import accuracy_score
import matplotlib.pyplot as plt
import os

#%% Load the data
os.chdir('/home/tuomas/Python/DATA.ML.200/Ex6')
data = loadmat('arcene.mat')

trainX = data['X_train']
trainY = data['y_train'].ravel()

testX = data['X_test']
testY = data['y_test'].ravel()
```

[2]:
```
# a) Train a random forest classifier with 100 trees.
model = RandomForestClassifier(n_estimators=100, n_jobs=-1)
model.fit(trainX, trainY)
```

[2]:
```
RandomForestClassifier(n_jobs=-1)
```

[3]:
```
# b) Plot a histogram of its feature importances.
feature_importances = model.feature_importances_
plt.figure(figsize=(15,10))
plt.plot(feature_importances, '-')
plt.title('Feature importances')
```

[3]:
```
Text(0.5, 1.0, 'Feature importances')
```



[4]:
```
# c) Compute the accuracy on X_test and y_test.
predY = model.predict(testX)
acc = accuracy_score(testY, predY)
print('Accuracy = {}'.format(acc))
```

```
Accuracy = 0.78
```

5 Question 4

[5]:
```
from sklearn.discriminant_analysis import LinearDiscriminantAnalysis
from sklearn.feature_selection import RFECV
# a) Instantiate an RFECV selector
lda = LinearDiscriminantAnalysis(solver='svd')
rfe = RFECV(estimator=lda, step=50, verbose=1, n_jobs=-1)
```

[6]:
```
# b) Fit the RFECV to X_train and y_train.
rfe.fit(trainX, trainY)
```

```
Fitting estimator with 10000 features.
Fitting estimator with 9950 features.
Fitting estimator with 9900 features.
Fitting estimator with 9850 features.
Fitting estimator with 9800 features.
```

```
Fitting estimator with 9750 features.
Fitting estimator with 9700 features.
Fitting estimator with 9650 features.
Fitting estimator with 9600 features.
Fitting estimator with 9550 features.
Fitting estimator with 9500 features.
Fitting estimator with 9450 features.
Fitting estimator with 9400 features.
Fitting estimator with 9350 features.
Fitting estimator with 9300 features.
Fitting estimator with 9250 features.
Fitting estimator with 9200 features.
Fitting estimator with 9150 features.
Fitting estimator with 9100 features.
Fitting estimator with 9050 features.
Fitting estimator with 9000 features.
Fitting estimator with 8950 features.
Fitting estimator with 8900 features.
Fitting estimator with 8850 features.
Fitting estimator with 8800 features.
Fitting estimator with 8750 features.
Fitting estimator with 8700 features.
Fitting estimator with 8650 features.
Fitting estimator with 8600 features.
Fitting estimator with 8550 features.
Fitting estimator with 8500 features.
Fitting estimator with 8450 features.
Fitting estimator with 8400 features.
Fitting estimator with 8350 features.
Fitting estimator with 8300 features.
Fitting estimator with 8250 features.
Fitting estimator with 8200 features.
Fitting estimator with 8150 features.
Fitting estimator with 8100 features.
Fitting estimator with 8050 features.
Fitting estimator with 8000 features.
Fitting estimator with 7950 features.
Fitting estimator with 7900 features.
Fitting estimator with 7850 features.
Fitting estimator with 7800 features.
Fitting estimator with 7750 features.
Fitting estimator with 7700 features.
Fitting estimator with 7650 features.
Fitting estimator with 7600 features.
Fitting estimator with 7550 features.
Fitting estimator with 7500 features.
Fitting estimator with 7450 features.
Fitting estimator with 7400 features.
```

```
Fitting estimator with 7350 features.
Fitting estimator with 7300 features.
Fitting estimator with 7250 features.
Fitting estimator with 7200 features.
Fitting estimator with 7150 features.
Fitting estimator with 7100 features.
Fitting estimator with 7050 features.
Fitting estimator with 7000 features.
Fitting estimator with 6950 features.
Fitting estimator with 6900 features.
Fitting estimator with 6850 features.
Fitting estimator with 6800 features.
Fitting estimator with 6750 features.
Fitting estimator with 6700 features.
Fitting estimator with 6650 features.
Fitting estimator with 6600 features.
Fitting estimator with 6550 features.
Fitting estimator with 6500 features.
Fitting estimator with 6450 features.
Fitting estimator with 6400 features.
Fitting estimator with 6350 features.
Fitting estimator with 6300 features.
Fitting estimator with 6250 features.
Fitting estimator with 6200 features.
Fitting estimator with 6150 features.
Fitting estimator with 6100 features.
Fitting estimator with 6050 features.
Fitting estimator with 6000 features.
Fitting estimator with 5950 features.
Fitting estimator with 5900 features.
Fitting estimator with 5850 features.
Fitting estimator with 5800 features.
Fitting estimator with 5750 features.
Fitting estimator with 5700 features.
Fitting estimator with 5650 features.
Fitting estimator with 5600 features.
Fitting estimator with 5550 features.
Fitting estimator with 5500 features.
Fitting estimator with 5450 features.
Fitting estimator with 5400 features.
Fitting estimator with 5350 features.
Fitting estimator with 5300 features.
Fitting estimator with 5250 features.
Fitting estimator with 5200 features.
Fitting estimator with 5150 features.
Fitting estimator with 5100 features.
Fitting estimator with 5050 features.
Fitting estimator with 5000 features.
```

```
Fitting estimator with 4950 features.
Fitting estimator with 4900 features.
Fitting estimator with 4850 features.
Fitting estimator with 4800 features.
Fitting estimator with 4750 features.
Fitting estimator with 4700 features.
Fitting estimator with 4650 features.
Fitting estimator with 4600 features.
Fitting estimator with 4550 features.
Fitting estimator with 4500 features.
Fitting estimator with 4450 features.
Fitting estimator with 4400 features.
Fitting estimator with 4350 features.
Fitting estimator with 4300 features.
Fitting estimator with 4250 features.
Fitting estimator with 4200 features.
Fitting estimator with 4150 features.
Fitting estimator with 4100 features.
Fitting estimator with 4050 features.
Fitting estimator with 4000 features.
Fitting estimator with 3950 features.
Fitting estimator with 3900 features.
Fitting estimator with 3850 features.
Fitting estimator with 3800 features.
```

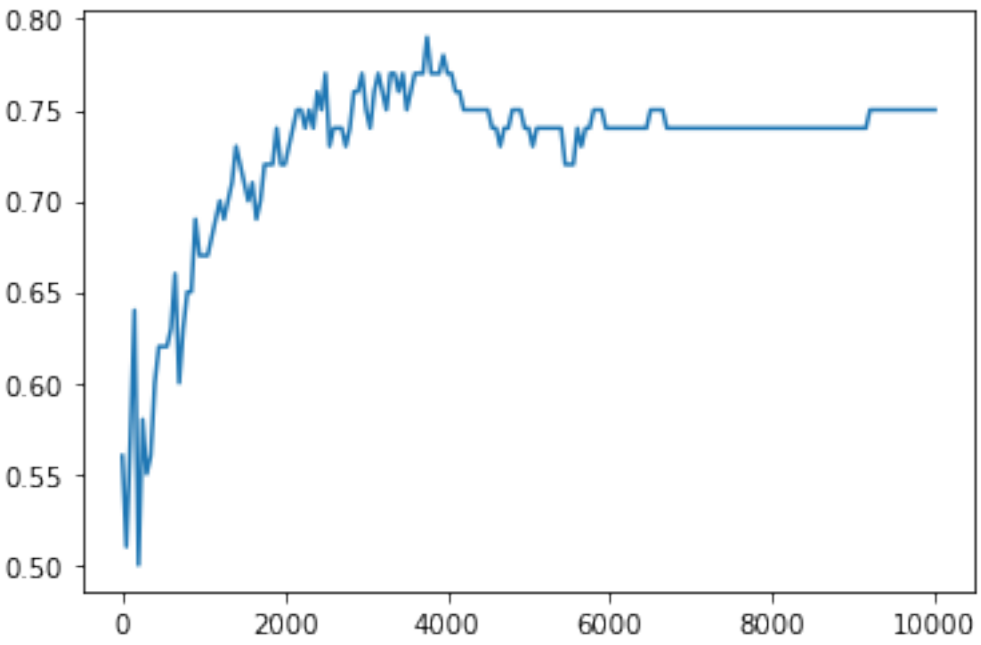
[6]:
```
RFECV(estimator=LinearDiscriminantAnalysis(), n_jobs=-1, step=50, verbose=1)
```

[7]:
```
import numpy as np
# c) Count the number of selected features from rfe.support_.
selected_features_count = np.sum(rfe.support_)
print('Number of selected features = {}'.format(selected_features_count))
```

```
Number of selected features = 3750
```

[8]:
```
# d) Plot the errors for different number of features:
plt.plot(range(0,10001,50), rfe.grid_scores_)
```

[8]:
```
[<matplotlib.lines.Line2D at 0x7f7399700e80>]
```



[9]:
```
# e) Compute the accuracy on X_test and y_test.
predY = rfe.predict(testX)
acc = accuracy_score(testY, predY)
print('Accuracy = {}'.format(acc))
```

```
Accuracy = 0.77
```

6 Question 5

[24]:
```
from sklearn.linear_model import LogisticRegression
# a) Instantiate a LogisticRegression classifier.
model = LogisticRegression(penalty='l1', max_iter=1000, solver='liblinear')
```

[25]:
```
from sklearn.model_selection import cross_val_score
# b) Estimate the accuracy of the classifier using 5-fold CV
cv_scores = []
for i in range(-4,3):
    c = 10**i
    model.C = c
    cvs = cross_val_score(model, trainX, trainY, cv=5, n_jobs=-1)
    # Evaluate classifiers by the mean of cv scores
    cv_scores.append(np.mean(cvs))
```

[26]:
```
# c) Fit the LogisticRegression to X_train and y_train with the best C.
i=np.argmax(cv_scores)-4
```

```
model.C = 10**i
print('Best C = 10^{}'.format(i))
model.fit(trainX, trainY)
```

```
Best C = 10^2
```

[26]:
```
LogisticRegression(C=100, max_iter=1000, penalty='l1', solver='liblinear')
```

[27]:
```
# d) Count the number of selected features from clf.coef_
selected_features_count = np.sum(model.coef_!=0.)
print('Number of selected features = {}'.format(selected_features_count))
```

```
Number of selected features = 2496
```

[28]:
```
# Compute the accuracy on X_test and y_test.
predY = model.predict(testX)
acc = accuracy_score(testY, predY)
print('Accuracy = {}'.format(acc))
```

```
Accuracy = 0.81
```