למידה סטטיסטית מבוססת נתונים - חורף - 096411

HW1

מגישים:

איתי ברקוביץ 039632732

אילן פרנק 043493386

שאלה ב

נתון מודל הרגרסיה הפשוטה:

$$y_i = \beta_0 + \beta_1 x_i + \varepsilon_i$$

כאשר, בין x-y:

β_0 - פרמטר החותך

β_1 - שיפוע קו הרגרסיה

הגדרה: סכום ריבועי השגיאות:

$$S(\beta_0,\beta_1)=\sum_{i=1}^{n}\varepsilon_i^2=\sum_{i=1}^{n}(y_i-\beta_0-\beta_1x_i)^2$$

הגדרה: אומד הריבועים הפחותים יסומן: $(\hat\beta_0,\hat\beta_1)=\arg\min_{\beta_0,\beta_1} S(\beta_0,\beta_1)$

* ערך ה-β שואף למינימום שגיאות

- S פונקציה רציפה, תנאי הכרחי ומספיק למינימום, הינו איפוס הנגזרות החלקיות:

$$\frac{\partial}{\partial\beta_0}S(\beta_0,\beta_1)=0$$

$$\frac{\partial}{\partial\beta_1}S(\beta_0,\beta_1)=0$$

גזירה לפי כלל השרשרת:

(1) $$\frac{\partial}{\partial\beta_0}S(\beta_0,\beta_1)=\frac{\partial}{\partial\beta_0}\sum_{i=1}^{n}(y_i-\beta_0-\beta_1x_i)^2=2\sum_{1}(y_i-\beta_0-\beta_1x_i)=0$$

$$\Rightarrow \sum(y_i-\hat\beta_0-\hat\beta_1x_i)=0$$

(2) $$\frac{\partial}{\partial\beta_1}S(\beta_0,\beta_1)=\frac{\partial}{\partial\beta_1}\sum(y_i-\beta_0-\beta_1x_i)^2=\frac{\partial}{\partial\beta_1}2\sum_{i=1}^{n}(y_i-\beta_0-\beta_1x_i)x_i$$

$$\Rightarrow\sum(y_i-\beta_0-\beta_1x_i)\cdot x_i=0$$

קיבלנו 2 משוואות לינאריות ב-2 נעלמים.
ממשוואה ① נקבל

$$① \sum y_i - n\hat{\beta}_0 - \hat{\beta}_1 \sum x_i = 0$$

$$\boxed{\hat{\beta}_0 = \bar{y} - \hat{\beta}_1 \bar{x}}$$

נחלץ את β_0 ונקבל

נציב משוואה 1 במשוואה 2 ונקבל:

$$② \sum \left(y_i - (\bar{y} - \hat{\beta}_1 \bar{x}) - \hat{\beta}_1 x_i\right) \cdot x_i = 0$$

$$= \sum \left[(y_i - \bar{y}) - \hat{\beta}_1 (x_i - \bar{x})\right] x_i = \sum x_i (y_i - \bar{y}) - \hat{\beta}_1 \sum (x_i - \bar{x})$$

$$\Rightarrow \boxed{\hat{\beta}_1 = \frac{\sum_{i=1}^{n} (y_i - \bar{y}) x_i}{\sum_{i=1}^{n} (x_i - \bar{x}) x_i}}$$

* ניתן לראות:

$$① \sum (y_i - \bar{y}) \cdot x_i = \sum (y_i - \bar{y})(x_i - \bar{x}) = \sum x_i y_i - n\bar{x}\bar{y}$$

$$② \sum (x_i - \bar{x}) \cdot x = \sum (x_i - \bar{x})^2 = \sum x_i^2 - n\bar{x}^2$$

$\Leftarrow$

$$\hat{\beta}_1 = \frac{\sum x_i y_i - n\bar{x} \cdot \bar{y}}{\sum x_i^2 - n\bar{x}^2}$$

אם נציב

$$\bar{y} = \frac{1}{n} \sum y_i \quad , \quad \bar{x} = \frac{1}{n} \sum x_i$$

$$\hat{\beta}_1 = \frac{\sum x_i y_i - \frac{1}{n} n \sum y_i \cdot \sum x_i}{\sum x_i^2 - \frac{1}{n} n \left(\sum x_i\right)^2}$$

$$\hat{\beta}_0 = \frac{1}{n} \sum y_i - \frac{\sum x_i y_i - \frac{1}{n} n \sum y_i \sum x_i}{\sum x_i^2 - \frac{1}{n} n \left(\sum x_i\right)^2} \times \frac{1}{n} \sum x_i$$

ב. טען: $\hat{W}_1, \hat{W}_0$ הינם אומד חסר הטיה:

$$E\{\hat{\beta}_1\} = \beta_1$$

$$E\{\hat{\beta}_0\} = \beta_0$$

הוכחה:

$$\hat{\beta}_1 = \underbrace{\frac{\sum (x_i - \bar{x}) y_i}{S_{xx}}}_{b_i} = \sum b_i y_i$$

$$\hat{\beta}_0 = \bar{y} - \hat{\beta}_1 \bar{x} = \frac{\sum y_i}{n} - \bar{x} \sum b_i y_i = \underline{\sum a_i y_i}$$

$$\underline{a_i} = \frac{1}{n} - b_i \bar{x} = \frac{1}{n} - \frac{(x_i - \bar{x}) \bar{x}}{S_{xx}}$$

$$E(\hat{\beta}_1) = E\left[\sum b_i y_i\right] = \sum b_i E(y_i) = \sum b_i (\beta_0 + \beta_1 x_i) = \beta_0 \sum b_i + \beta_1 \sum x b_i$$

$$* \quad \sum b_i = \sum \frac{(x_i - \bar{x})}{S_{xx}} = \frac{1}{S_{xx}} \sum (x_i - \bar{x}) = 0$$

$$* \quad \sum b_i x_i = \frac{\sum (x_i - \bar{x}) x_i}{S_{xx}} = \frac{1}{S_{xx}} \sum (x_i - \bar{x})^2 = \frac{S_{xx}}{S_{xx}} = 1.0$$

$$\Rightarrow E(\hat{\beta}_1) = \beta_1$$

מ.ש.ל

$$E(\hat{\beta}_0) = E\left\{\sum a_i y_i\right\} = \sum a_i E(y_i) = \sum a_i (\beta_0 + \beta_1 x_i) = \beta_0 \sum a_i + \beta_1 \sum x_i a_i$$

$$* \quad \sum a_i = \sum \frac{1}{n} - \frac{(x_i - \bar{x}) \cdot \bar{x}}{S_{xx}} = 1 + x \sum \frac{(x_i - \bar{x})}{S_{xx}} = \underline{\underline{1.0}}$$

$$* \quad \sum x_i a_i = \sum \left(\frac{1}{n} - \frac{(x_i - \bar{x})}{S_{xx}} \cdot \bar{x} \right) \cdot x_i = \sum \frac{x_i}{n} - \bar{x} \sum \frac{(x_i - \bar{x}) \cdot x_i}{S_{xx}}$$

$$= \bar{x} - \bar{x} = 0$$

$$\Rightarrow E(\hat{\beta}_0) = \beta_0$$

מ.ש.ל

In [ ]:

```

```

In [ ]:

```
# Question 2
#A.
import pandas as pd
import numpy as np

df = pd.read_csv('parkinsons_updrs_data.csv')

df['first'] = 1

df.head()
```

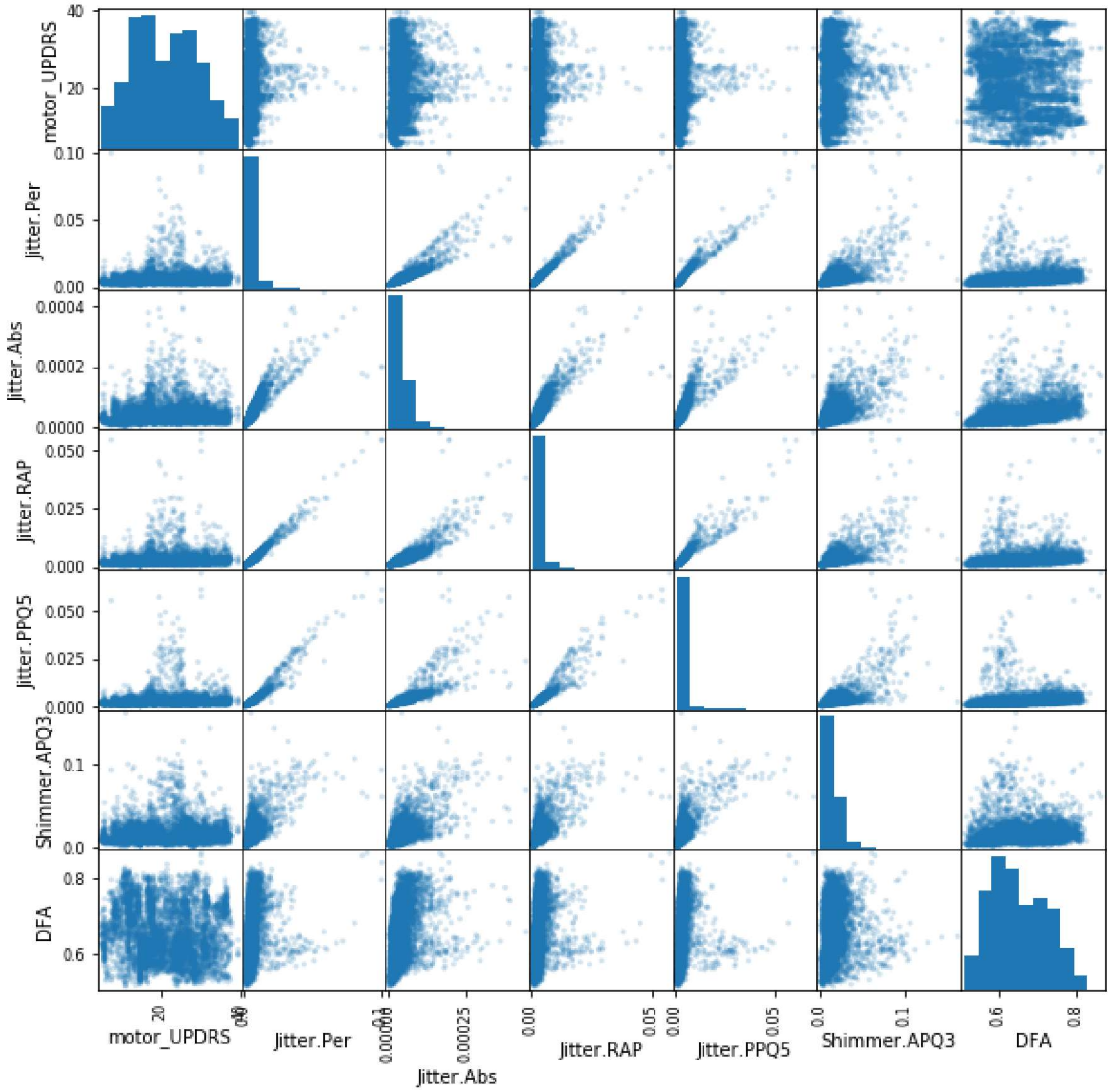
In [7]:

```
#B.
#Age -36 הערך מצייג את גיל הנבדקים. ממוצע גלאי הנבדקים הינו 65 , הנבדק המבוגר ביותר בן 85, הנבדק הצעיר ביותר
בן
#sex-הערך מייצג את מין הנבדק. הערך הינו ערך בוליאני. 0 מייצג גבר ו-1 מייצג אישה. 32% מהנבדקים הינם גברים
```

In [10]:

```
#C:

import numpy as np
import matplotlib.pyplot as plt
import pandas as pd
from sklearn import datasets
%matplotlib inline


df1 = df[['motor_UPDRS', 'Jitter.Per', 'Jitter.Abs','Jitter.RAP','Jitter.PPQ5', 'Shimme
r.APQ3', 'DFA']]
pd.scatter_matrix(df1, alpha=0.2, figsize=(10, 10))
plt.show()
```

```
C:\Users\lenovo\Anaconda3\lib\site-packages\ipykernel_launcher.py:11: Futu
reWarning: pandas.scatter_matrix is deprecated, use pandas.plotting.scatte
r_matrix instead
  # This is added back by InteractiveShellApp.init_path()
```



In [11]:

```
#D:

def answer_D(x,y):
    first_step = np.linalg.inv(np.dot(x.T,x))
    sec_step =  np.dot(first_step,x.T)
    third_step = np.dot(sec_step,y)
    return third_step
```

In [12]:

```
#E:

x = np.array(df[['first','Jitter.Per', 'Jitter.Abs','Jitter.RAP','Jitter.PPQ5', 'Shimme
r.APQ3', 'DFA']])
y = np.array(df[['motor_UPDRS']])
w = answer_D(x,y)
w
```

Out[12]:

```
array([[ 3.01179617e+01],
       [ 1.35025189e+03],
       [-2.05976226e+04],
       [-1.19055662e+03],
       [-7.51684784e+02],
       [ 4.34744789e+01],
       [-1.67630228e+01]])
```

In [18]:

```
#F:
import os
import pandas as pd
from statsmodels.formula.api  import ols


df2 = df[['motor_UPDRS','first', 'Jitter.Per', 'Jitter.Abs','Jitter.RAP','Jitter.PPQ5',
 'Shimmer.APQ3', 'DFA']]
df2['motor_UPDRS'].values.reshape(-1, 1)
df2 = df2.rename(index=str, columns={'Jitter.Per': 'Jitter_Per','Jitter.Abs': 'Jitter_Abs','Jitter.RAP': 'Jitter_RAP','Jitter.PPQ5':'Jitter_PPQ5',
                                    'Shimmer.APQ3':'Shimmer_APQ3'})
est = ols(formula = ' motor_UPDRS ~ first + Jitter_Per + Jitter_Abs + Jitter_RAP + Jitter_PPQ5 + Shimmer_APQ3 + Shimmer_APQ3 + DFA', data = df2).fit()
est.summary()
```

Out[18]:

OLS Regression Results

Dep. Variable:	motor_UPDRS	**R-squared:**	0.038
Model:	OLS	**Adj. R-squared:**	0.037
Method:	Least Squares	**F-statistic:**	38.61
Date:	Wed, 14 Nov 2018	**Prob (F-statistic):**	2.94e-46
Time:	08:47:22	**Log-Likelihood:**	-20533.
No. Observations:	5875	**AIC:**	4.108e+04
Df Residuals:	5868	**BIC:**	4.113e+04
Df Model:	6		
Covariance Type:	nonrobust		

	coef	std err	t	P>|t|	[0.025	0.975]
Intercept	15.0590	0.509	29.600	0.000	14.062	16.056
first	15.0590	0.509	29.600	0.000	14.062	16.056
Jitter_Per	1350.2519	158.081	8.542	0.000	1040.355	1660.148
Jitter_Abs	-2.06e+04	6726.924	-3.062	0.002	-3.38e+04	-7410.373
Jitter_RAP	-1190.5566	191.788	-6.208	0.000	-1566.532	-814.582
Jitter_PPQ5	-751.6848	126.651	-5.935	0.000	-999.967	-503.403
Shimmer_APQ3	43.4745	10.981	3.959	0.000	21.947	65.002
DFA	-16.7630	1.598	-10.489	0.000	-19.896	-13.630

Omnibus:	476.923	**Durbin-Watson:**	0.082
Prob(Omnibus):	0.000	**Jarque-Bera (JB):**	171.811
Skew:	0.150	**Prob(JB):**	4.92e-38
Kurtosis:	2.218	**Cond. No.**	2.36e+17

Warnings:
[1] Standard Errors assume that the covariance matrix of the errors is correctly specified.
[2] The smallest eigenvalue is 2.55e-31. This might indicate that there are
strong multicollinearity problems or that the design matrix is singular.

In [ ]:

```
#f: בשני המקרים קיבלנו את אותם הערכים לאמד
```

סעיף ז:

עבור כללי ערכי האמד, קיבלנו ערכי p-value נמוכים מ-0.01, לכן נוכל לדחות את השערת האפס על β_j ברמת מובהקות של $\alpha = 0.01$.

ע"מ לדחות השערה זו נשתמש במבחן t דו צדדי בעל הסטטיסטי הבא:

$$\frac{\hat{\beta}_1}{\sqrt{\widehat{Var}(\hat{\beta}_1)}}$$

עבור רמת מובהקות $\alpha = 0.001$, נוכל לדחות את השערת האפס על β_j רק עבור המשתנה Jitter abs , מכיוון שערך ה-p-value הינו 0.002.

3 א. $y\in\{\pm1\}$

$$Pr(y_i=1|x_i)=\frac{e^{<w,x_i>}}{1+e^{<w,x_i>}}=\frac{1}{e^{-<w,x_i>}+1}$$

$$Pr(y_i=-1|x_i)=1-\frac{1}{e^{-<w,x_i>}+1}=\frac{e^{-<w,x_i>}}{e^{-<w,x_i>}+1}=\frac{1}{1+e^{<w,x_i>}}$$

הנראות המירבית

$$L(w)=\prod_{i=1}^{m}\Big[\underbrace{Pr(y_i=1|x_i)^{\frac{y_i+1}{2}}}_{y_i=1 \text{ עבור } =1}\cdot\underbrace{Pr(y_i=-1|x_i)^{-\frac{(y_i-1)}{2}}}_{y_i=1 \text{ עבור } =1}\Big]$$

ועבור לוג הנראות המירבית:

$$l(w)=\sum_{i=1}^{m}\Big[\frac{y_i+1}{2}\log\frac{1}{1+e^{-<w,x_i>}}-\frac{y_i-1}{2}\log\frac{1}{1+e^{<w,x_i>}}\Big]$$

נחפש w שממקסם ערך זה $\underset{w}{\arg\max}\ l(w)$

ב. נתבונן בביטוי בתוך הסכום עבור ערכי y_i השונים:

עבור $y_i=1$: $\frac{2}{2}\log\frac{1}{1+e^{-<w,x_i>}}+\cancel{0\cdot\log\frac{1}{1+e^{<w,x_i>}}}$ המחובר הי' הופך ל:

$$=\log(1+e^{-<w,x_i>})^{-1}$$

$$=-\log(1+e^{-y_i<w,x_i>})\quad y_i=1$$

עבור $y_i=-1$: $0\cdot\log\frac{1}{1+e^{-<w,x_i>}}-\frac{-2}{2}\log\frac{1}{1+e^{<w,x_i>}}$

$$=\log(1+e^{<w,x_i>})^{-1}\quad y_i=-1$$

$$=-\log(1+e^{-y_i<w,x_i>})$$

כלומר ניתן לכתוב את $l(w)$ כצורה

$$l(w)=\sum_{i=1}^{m}-\log(1+e^{-y_i<w,x_i>})$$

ונקבל כי

$$\underset{w}{\arg\max}\ l(w)=\underset{w}{\arg\max}-\sum\log(1+e^{-y_i<w,x_i>})$$

$$=\underset{w}{\arg\min}\sum\log(1+e^{-y_i<w,x_i>})$$

כנדרש

In [ ]:

```
# Question 4


from sklearn import datasets
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LogisticRegression
import pandas as pd
import numpy as np
```

In [2]:

```
# (a)
iris = datasets.load_iris()
X = iris.data
y = iris.target
```

In [3]:

```
# (b)
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.25, random_state=
1000)
```

In [4]:

```
# (c+d)
y_one_vs_all_train = {}
y_one_vs_all_test = {}
lrs = {}
print(y_test)
for i in range(3):
    y_one_vs_all_train[i] = np.ones(y_train.shape)*-1
    y_one_vs_all_train[i][y_train==i] = 1

    y_one_vs_all_test[i] = np.ones(y_test.shape)*-1
    y_one_vs_all_test[i][y_test==i] = 1
#     print(y_one_vs_all_test[i])

    lrs[i] = LogisticRegression()
    lrs[i].fit(X_train,y_one_vs_all_train[i])
#     print(lrs[i].predict_proba(X_test))

    print(f'{i} score: {lrs[i].score(X_test, y_one_vs_all_test[i])}')
# y_one_vs_all_train, y_train
```

```
[1 0 2 2 0 0 1 1 0 2 2 1 0 0 2 1 2 1 0 0 1 1 2 2 1 0 1 1 1 2 0 1 2 1 0 2 2
 1]
0 score: 1.0
1 score: 0.6842105263157895
2 score: 0.9473684210526315
```

In [7]:

```
# (e)

def get_one_vs_all(lrs, test):
    predictions = [max(lrs.keys(), key=lambda i: lrs[i].predict_proba([x])[0, 1]) for x
 in test]
#     for sample_idx, sample in enumerate(test):
#         print([lrs[i].predict_proba([sample]) for i in lrs])
#         max_lr = max(lrs.keys(), key=lambda i: lrs[i].predict_proba([sample])[0, 1])
#         print(max_lr)
    return predictions



y = get_one_vs_all(lrs, X_test)

y == y_test
```

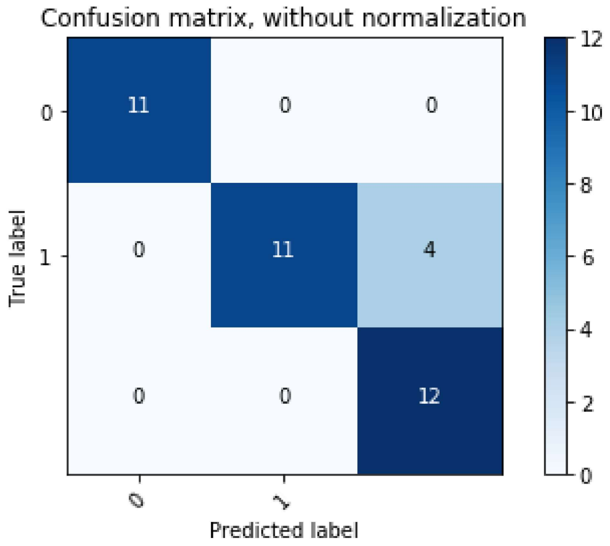
Out[7]:

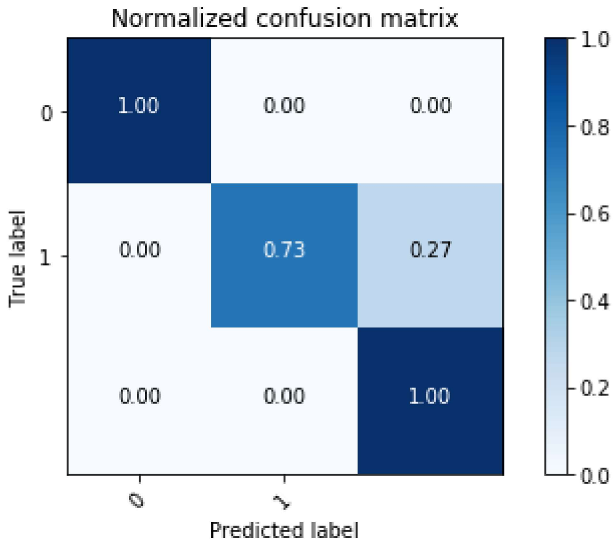
```
array([ True,  True,  True,  True,  True,  True, False,  True,  True,
        True,  True,  True,  True,  True,  True,  True,  True,  True,
        True,  True,  True, False,  True,  True,  True,  True, False,
        True,  True,  True,  True,  True,  True,  True,  True,  True,
        True, False])
```

In [397]:

```
# (f)

from sklearn.metrics import confusion_matrix
import numpy as np
import matplotlib.pyplot as plt
from sklearn.metrics import confusion_matrix
import itertools

def plot_confusion_matrix(cm, classes,
                          normalize=False,
                          title='Confusion matrix',
                          cmap=plt.cm.Blues):
    """
    This function prints and plots the confusion matrix.
    Normalization can be applied by setting `normalize=True`.
    """
    if normalize:
        cm = cm.astype('float') / cm.sum(axis=1)[:, np.newaxis]

    plt.imshow(cm, interpolation='nearest', cmap=cmap)
    plt.title(title)
    plt.colorbar()
    tick_marks = np.arange(len(classes))
    plt.xticks(tick_marks, classes, rotation=45)
    plt.yticks(tick_marks, classes)

    fmt = '.2f' if normalize else 'd'
    thresh = cm.max() / 2.
    for i, j in itertools.product(range(cm.shape[0]), range(cm.shape[1])):
        plt.text(j, i, format(cm[i, j], fmt),
                 horizontalalignment="center",
                 color="white" if cm[i, j] > thresh else "black")

    plt.ylabel('True label')
    plt.xlabel('Predicted label')
    plt.tight_layout()


y_pred = get_one_vs_all(lrs, X_test)
# print(y_test)
# print(y_pred)
print(list(zip(y_test, y_pred)))
cnf_matrix = confusion_matrix(y_test, y_pred)
np.set_printoptions(precision=2)

# Plot non-normalized confusion matrix
plt.figure()
plot_confusion_matrix(cnf_matrix, classes=[0,1],
                      title='Confusion matrix, without normalization')

# Plot normalized confusion matrix
plt.figure()
plot_confusion_matrix(cnf_matrix, classes=[0,1], normalize=True,
                      title='Normalized confusion matrix')

plt.show()
```

```
[(1, 1), (0, 0), (2, 2), (2, 2), (0, 0), (0, 0), (1, 2), (1, 1), (0, 0),
(2, 2), (2, 2), (1, 1), (0, 0), (0, 0), (2, 2), (1, 1), (2, 2), (1, 1),
(0, 0), (0, 0), (1, 1), (1, 2), (2, 2), (2, 2), (1, 1), (0, 0), (1, 2),
(1, 1), (1, 1), (2, 2), (0, 0), (1, 1), (2, 2), (1, 1), (0, 0), (2, 2),
(2, 2), (1, 2)]
```





In [398]:

```
# (g)
# print(list(zip(X_train, y_train)))
for i, sample in enumerate(X_test):
    print(f'X: {sample}, true label: {y_test[i]}, classifier 1: {lrs[1].predict_proba
([sample])[0, 1]}, '
          f'classifier 2: {lrs[2].predict_proba([sample])[0, 1]}')
print()
for i in range(3):
    print(f'classifier {i}:')
    print(lrs[i].coef_)
print()
print(f'average train sample for classifier 1: {np.average(X_train[y_train==1], axis=
0)}')
print(f'average train sample or classifier 2: {np.average(X_train[y_train==2], axis=
0)}')
print()
print(f'average test for classifier 1: {np.average(X_test[y_test==1], axis=0)}')
print(f'average test for classifier 2: {np.average(X_test[y_test==2], axis=0)}')
print()
print("""The last test sample is labeled as 1, but tagged as 2 by the one-vs-all classi
fer.
We can tell by looking at the classifiers' weights that classifier 1 relies mostly on f
eatures 0,2, while 1,3 negate it,
and classifier 2 relies mostly on features 2,3, while 0,1 negate it.

We look at the data for the problematic sample: [ 5.6  3.   4.5  1.5].
Comparing to other samples of label 1, this has a relatively high value for feature 1 a
nd 2.
This causes the score for classifier 1 be relatively low (0.27) and for classifier 2 be
 a little high (0.44),
which causes the sample to be labeled as 2.""")
```

```
X: [ 5.7  3.   4.2  1.2], true label: 1, classifier 1: 0.3184256754872217,
classifier 2: 0.15081079174784146
X: [ 4.9  3.1  1.5  0.1], true label: 0, classifier 1: 0.2319246715255044
7, classifier 2: 0.00011700991022690092
X: [ 6.1  3.   4.9  1.8], true label: 2, classifier 1: 0.2796000816713362
6, classifier 2: 0.6287402302847547
X: [ 6.9  3.2  5.7  2.3], true label: 2, classifier 1: 0.2543561079263063,
classifier 2: 0.8682121895240641
X: [ 4.8  3.4  1.9  0.2], true label: 0, classifier 1: 0.1485946800469467,
classifier 2: 0.00030902795364945776
X: [ 5.2  3.4  1.4  0.2], true label: 0, classifier 1: 0.1551910236648960
4, classifier 2: 5.076884717174568e-05
X: [ 5.4  3.   4.5  1.5], true label: 1, classifier 1: 0.2342521975914924,
classifier 2: 0.5336967442627625
X: [ 6.2  2.2  4.5  1.5], true label: 1, classifier 1: 0.671426481741066,
classifier 2: 0.41278844180816826
X: [ 4.4  3.   1.3  0.2], true label: 0, classifier 1: 0.1778857122143906,
classifier 2: 0.00024327253363154723
X: [ 6.3  2.7  4.9  1.8], true label: 2, classifier 1: 0.4264187619574401
6, classifier 2: 0.6256782301284715
X: [ 6.4  2.8  5.6  2.2], true label: 2, classifier 1: 0.352800488421773,
classifier 2: 0.9380449220273951
X: [ 6.4  2.9  4.3  1.3], true label: 1, classifier 1: 0.4428950199101907,
classifier 2: 0.08659259870469917
X: [ 5.1  3.8  1.9  0.4], true label: 0, classifier 1: 0.0760834421512799
6, classifier 2: 0.00018888398064920722
X: [ 5.   3.6  1.4  0.2], true label: 0, classifier 1: 0.1025480243170745
5, classifier 2: 5.7347928824759955e-05
X: [ 6.7  3.3  5.7  2.5], true label: 2, classifier 1: 0.1663078557579183
4, classifier 2: 0.9287465297500211
X: [ 5.8  2.7  4.1  1. ], true label: 1, classifier 1: 0.5060438308245141,
classifier 2: 0.09544381343775583
X: [ 6.   2.2  5.   1.5], true label: 2, classifier 1: 0.6873627491159601,
classifier 2: 0.752979646735354
X: [ 5.8  2.7  3.9  1.2], true label: 1, classifier 1: 0.4275686128343456
6, classifier 2: 0.09550194938571142
X: [ 4.9  3.1  1.5  0.1], true label: 0, classifier 1: 0.2319246715255044
7, classifier 2: 0.00011700991022690092
X: [ 5.   3.3  1.4  0.2], true label: 0, classifier 1: 0.161969454587434,
classifier 2: 7.947230466700494e-05
X: [ 5.5  2.4  3.7  1. ], true label: 1, classifier 1: 0.5511030657944215,
classifier 2: 0.08983671376181318
X: [ 5.9  3.2  4.8  1.8], true label: 1, classifier 1: 0.1883552201054344
6, classifier 2: 0.6042370158206046
X: [ 7.2  3.2  6.   1.8], true label: 2, classifier 1: 0.4555996065243735,
classifier 2: 0.7157393298750881
X: [ 6.9  3.1  5.1  2.3], true label: 2, classifier 1: 0.2427993572338048
9, classifier 2: 0.6549465817014978
X: [ 4.9  2.4  3.3  1. ], true label: 1, classifier 1: 0.4191943886070788
6, classifier 2: 0.09978971101872118
X: [ 5.7  4.4  1.5  0.4], true label: 0, classifier 1: 0.03446151006438176
5, classifier 2: 1.4417578310348812e-05
X: [ 6.   3.4  4.5  1.6], true label: 1, classifier 1: 0.1637571686968637
1, classifier 2: 0.2511714922251177
X: [ 6.2  2.9  4.3  1.3], true label: 1, classifier 1: 0.4124593417288993
6, classifier 2: 0.11747330256713163
X: [ 5.   2.   3.5  1. ], true label: 1, classifier 1: 0.6261913982852128,
classifier 2: 0.18495233800212407
X: [ 6.4  3.1  5.5  1.8], true label: 2, classifier 1: 0.3322589397841805
3, classifier 2: 0.7793908693880235
X: [ 5.1  3.3  1.7  0.5], true label: 0, classifier 1: 0.1396366180413344
```

```
5, classifier 2: 0.00025974398363687003
X: [ 6.   2.2  4.   1. ], true label: 1, classifier 1: 0.728103239958982,
classifier 2: 0.09363832901161794
X: [ 6.8  3.2  5.9  2.3], true label: 2, classifier 1: 0.2575614717406986,
classifier 2: 0.9245606022456869
X: [ 5.6  2.9  3.6  1.3], true label: 1, classifier 1: 0.2682431071545994,
classifier 2: 0.07062766125176291
X: [ 5.1  3.4  1.5  0.2], true label: 0, classifier 1: 0.1522405750784626
4, classifier 2: 7.537523022263318e-05
X: [ 6.7  2.5  5.8  1.8], true label: 2, classifier 1: 0.6588771943476974,
classifier 2: 0.8891610578909227
X: [ 6.4  2.7  5.3  1.9], true label: 2, classifier 1: 0.4514781835353867
7, classifier 2: 0.8133596518339681
X: [ 5.6  3.   4.5  1.5], true label: 1, classifier 1: 0.2572965193943098
6, classifier 2: 0.44907763659831124

classifier 0:
[[ 0.4   1.34 -2.1  -0.95]]
classifier 1:
[[ 0.62 -1.75  0.4  -1.18]]
classifier 2:
[[-1.7  -1.09  2.26  2.26]]

average train sample for classifier 1: [ 6.02  2.79  4.32  1.35]
average train sample or classifier 2: [ 6.59  2.99  5.58  2.03]

average test for classifier 1: [ 5.73  2.73  4.11  1.28]
average test for classifier 2: [ 6.57  2.92  5.45  2.  ]

The last test sample is labeled as 1, but tagged as 2 by the one-vs-all cl
assifer.
We can tell by looking at the classifiers' weights that classifier 1 relie
s mostly on features 0,2, while 1,3 negate it,
and classifier 2 relies mostly on features 2,3, while 0,1 negate it.

We look at the data for the problematic sample: [ 5.6  3.   4.5  1.5].
Comparing to other samples of label 1, this has a relatively high value fo
r feature 1 and 2.
This causes the score for classifier 1 be relatively low (0.27) and for cl
assifier 2 be a little high (0.44),
which causes the sample to be labeled as 2.
```