Projekt
Liniowy dyskryminator Fishera

Tomasz Dądela

2021

1 Treść zadania

Projekt polega na stworzeniu klasyfikatora, który na podstawie danych uczących będzie w stanie klasyfikować dane, których wcześniej nie widział. Student może wybrać do implementacji naiwny klasyfikator Bayesowski lub klasyfikator Fishera (liniowa analiza dyskryminacyjna Fishera, LDA) dla klasyfikacji binarnej (w przypadku klasyfikatora Fishera można zaimplementować klasyfikację do wielu klas). W rozwiązaniu nie wolno wykorzystywać gotowych bibliotek/frameworków realizujących działanie klasyfikatorów. Same klasyfikatory należy zaimplementować samodzielnie.

Klasyfikator powinien działać w dwóch trybach: nauki i klasyfikacji. W trybie nauki powinien na wejściu dostawać zbiór uczący z wyróżnioną zmienną oznaczającą poprawną klasę. W trybie klasyfikacji powinien dostawać na wejście zbiór a na wyjściu dla każdego elementu z tego zbioru powinien zwracać przewidzianą klasę.

W ramach eksperymentów należy zbadać skuteczność nauczonego na określonym zbiorze danych klasyfikatora przy pomocy takich miar jak dokładność, precyzja, miara F1, krzywe ROC itp. (zrobić wcześniej research dotyczący metod oceny klasyfikatorów, a także dotyczący metod kroswalidacji)

Do projektu można wykorzystać np. zbiory danych ze strony uci.edu[1]. Po wyborze zbioru (zbiorów) danych (powinny być „interesujące", nie trywialne) należy przeprowadzić jego (ich) analizę (opisać zmienne (parametry), zbadać ich zależność, korelacje, elementy odstające, zbadać czy w zbiorze są braki itp.), a następnie opracować procedurę uczenia klasyfikatora (np. z zastosowaniem kroswalidacji), przeprowadzić eksperymenty i ocenić jakość klasyfikatora.

2 Zbiór danych

Do przeprowadzenia analizy wybrałem zbiór danych "Banknote authentication Data Set"[2] Zbiór zawiera dane otrzymane za pomocą analizy zdjęć banknotów wykonanych na potrzeby sprawdzenia ich autentyczności.
Na w/w stronie dostępne są następujące informacje na temat danych:

[1] https://archive.ics.uci.edu/ml/index.php
[2] https://archive.ics.uci.edu/ml/datasets/banknote+authentication

Abstract

Data were extracted from images that were taken for the evaluation of an authentication procedure for banknotes.

Source

Owner of database: Volker Lohweg (University of Applied Sciences, Ostwestfalen-Lippe, volker.lohweg '@' hs-owl.de)
Donor of database: Helene Dörksen (University of Applied Sciences, Ostwestfalen-Lippe, helene.doerksen '@' hs-owl.de)
Date received: August, 2012

Data Set Information

Data were extracted from images that were taken from genuine and forged banknote-like specimens. For digitization, an industrial camera usually used for print inspection was used. The final images have 400x 400 pixels. Due to the object lens and distance to the investigated object gray-scale pictures with a resolution of about 660 dpi were gained. Wavelet Transform tool were used to extract features from images.

Attribute Information

1. variance of Wavelet Transformed image (continuous)
2. skewness of Wavelet Transformed image (continuous)
3. curtosis of Wavelet Transformed image (continuous)
4. entropy of image (continuous)
5. class (integer)

2.1 Wartości parametrów

```
import pandas as pd
df = pd.read_csv('data_banknote_authentication.txt')

pd.options.display.float_format = "{:.1f}".format
print(df.describe())
print('Licznosc poszczegolnych klas:\n', df['class'].value_counts())
```

	variance	skewness	curtosis	entropy	class
count	1372.0	1372.0	1372.0	1372.0	1372.0
mean	0.4	1.9	1.4	−1.2	0.4
std	2.8	5.9	4.3	2.1	0.5
min	−7.0	−13.8	−5.3	−8.5	0.0
25%	−1.8	−1.7	−1.6	−2.4	0.0
50%	0.5	2.3	0.6	−0.6	0.0
75%	2.8	6.8	3.2	0.4	1.0
max	6.8	13.0	17.9	2.4	1.0

```
Licznosc poszczegolnych klas:
 0     762
1     610
Name: class, dtype: int64
```

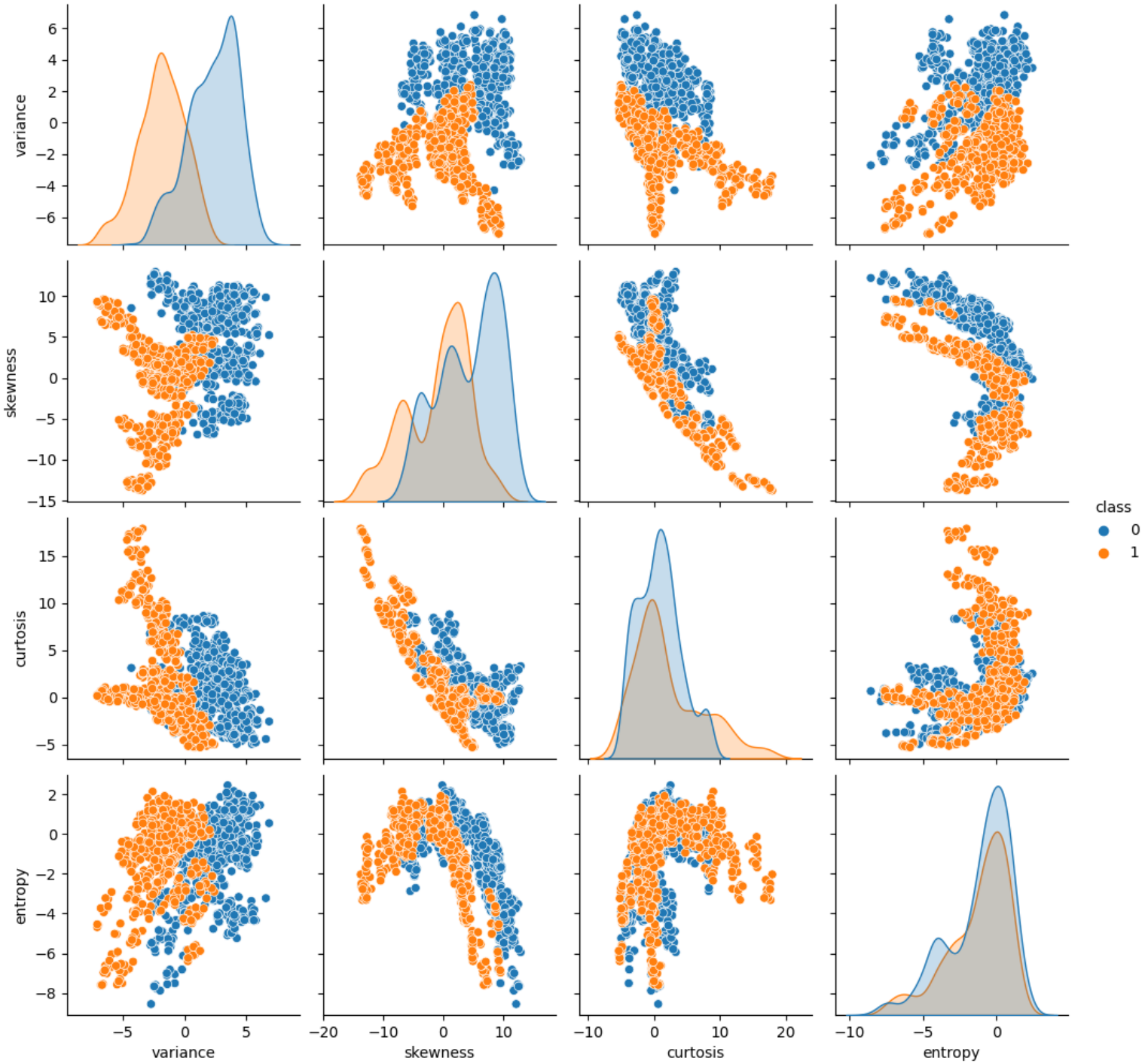
Zbiór danych zawiera 762 banknoty klasy 0 (banknoty autentyczne[3]) oraz 610 banknotów klasy 1 (banknoty sfałszowane).
Zbiór danych nie zawiera pustych parametrów.

2.2 Wizualizacja danych

```
import seaborn as sns
import matplotlib.pyplot as plt
import pandas as pd
df = pd.read_csv('data_banknote_authentication.txt')
sns.pairplot(df, hue='class')

plt.show()
```

[3] https://jamesmccaffrey.wordpress.com/2020/08/18/in-the-banknote-authentication-dataset-class-0-is-genuine-authentic/


variance
skewness
curtosis
entropy
class
0
1

2.3 Elementy odstające

```
import pandas as pd
df = pd.read_csv('data_banknote_authentication.txt')


def outliers(factor, df):
    print("factor: ", factor)
    for column in df:
        if column == "class":
            break
        columnSeriesObj = df[column]
        Q1 = columnSeriesObj.quantile(0.25)
        Q3 = columnSeriesObj.quantile(0.75)
        IQR = Q3 - Q1
        print('Colunm', column, "- ", end=" ")
        lower = Q1 - factor*IQR  # Standard value is 1.5
        upper = Q3 + factor*IQR
        outliers = ((columnSeriesObj < lower) |
                    (columnSeriesObj > upper)).sum()
        if column == "entropy":
            print(" ", end="")
        print("# outliers: ", outliers)


outliers(1.5, df)
outliers(2.5, df)
print()

authentic = df[df['class'] == 0]  # class 0
fake = df[df['class'] == 1]  # class 1

outliers(1.5, authentic)
outliers(2.0, authentic)
outliers(2.5, authentic)
print()

outliers(1.5, fake)
outliers(2.0, fake)
outliers(2.5, fake)
```

```
factor:  1.5
Columm variance -  # outliers:  0
Columm skewness -  # outliers:  0
Columm curtosis -  # outliers:  59
Columm entropy -   # outliers:  33
factor:  2.5
Columm variance -  # outliers:  0
Columm skewness -  # outliers:  0
Columm curtosis -  # outliers:  17
Columm entropy -   # outliers:  0

factor:  1.5
Columm variance -  # outliers:  1
Columm skewness -  # outliers:  0
Columm curtosis -  # outliers:  0
Columm entropy -   # outliers:  19
factor:  2.0
Columm variance -  # outliers:  0
Columm skewness -  # outliers:  0
Columm curtosis -  # outliers:  0
Columm entropy -   # outliers:  8
factor:  2.5
Columm variance -  # outliers:  0
Columm skewness -  # outliers:  0
Columm curtosis -  # outliers:  0
Columm entropy -   # outliers:  0

factor:  1.5
Columm variance -  # outliers:  3
Columm skewness -  # outliers:  0
Columm curtosis -  # outliers:  10
Columm entropy -   # outliers:  15
factor:  2.0
Columm variance -  # outliers:  0
Columm skewness -  # outliers:  0
Columm curtosis -  # outliers:  0
Columm entropy -   # outliers:  0
factor:  2.5
Columm variance -  # outliers:  0
Columm skewness -  # outliers:  0
Columm curtosis -  # outliers:  0
Columm entropy -   # outliers:  0
```

Zadecydowałem nie usuwać żadnego z rekordów.

2.4 Korelacja danych

```
import pandas as pd
df = pd.read_csv('data_banknote_authentication.txt')

authentic = df[df['class'] == 0]  # class 0
fake = df[df['class'] == 1]  # class 1

print(df.corr("pearson"))
print()
print("Fake")
print(authentic.corr("pearson"))
print()
print("Authentic")
print(fake.corr("pearson"))
```

	variance	skewness	curtosis	entropy	class
variance	1.000000	0.264026	−0.380850	0.276817	−0.724843
skewness	0.264026	1.000000	−0.786895	−0.526321	−0.444688
curtosis	−0.380850	−0.786895	1.000000	0.318841	0.155883
entropy	0.276817	−0.526321	0.318841	1.000000	−0.023424
class	−0.724843	−0.444688	0.155883	−0.023424	1.000000

Fake

	variance	skewness	curtosis	entropy	class
variance	1.000000	−0.226662	−0.329991	0.416110	NaN
skewness	−0.226662	1.000000	−0.750740	−0.674411	NaN
curtosis	−0.329991	−0.750740	1.000000	0.414543	NaN
entropy	0.416110	−0.674411	0.414543	1.000000	NaN
class	NaN	NaN	NaN	NaN	NaN

Authentic

	variance	skewness	curtosis	entropy	class
variance	1.000000	0.073689	−0.473867	0.324043	NaN
skewness	0.073689	1.000000	−0.887664	−0.508938	NaN
curtosis	−0.473867	−0.887664	1.000000	0.276017	NaN
entropy	0.324043	−0.508938	0.276017	1.000000	NaN
class	NaN	NaN	NaN	NaN	NaN

2.5 Podział danych

```
import pandas as pd
df = pd.read_csv('data_banknote_authentication.txt')
df = df.sample(frac=1).reset_index(drop=True)

k = 5

for i in range(k):
    df_test = df.iloc[int(i*len(df)/k):int((i+1)*len(df)/k), :]
    df_learning_1 = df.iloc[:int(i*len(df)/k), :]
    df_learning_2 = df.iloc[int((i+1)*len(df)/k):, :]
    df_learning = pd.concat([df_learning_1, df_learning_2], axis=0)
    df_test.to_csv("data_test_set_" + str(i) + ".csv", index=False)
    df_learning.to_csv("data_learning_set_" + str(i) + ".csv", index=False)
```

3 Liniowy dyskryminator Fishera

```
from math import sqrt


def lda(learning_set_path, test_set_path):
    import numpy as np
    import pandas as pd
    from math import log
    df = pd.read_csv(learning_set_path)

    authentic = df[df['class'] == 0]  # class 0
    authentic = authentic.drop(columns=["class"])
    no_authentic = len(authentic)

    fake = df[df['class'] == 1]  # class 1
    fake = fake.drop(columns=["class"])
    no_fake = len(fake)

    def getMu(dataset: list):
        mu = []
        for i in range(len(fake.columns)):
            column = dataset.iloc[:, [i]]
            column_mean = column.sum() / len(column)
            mu.append(column_mean)
        return mu

    mu_authentic = getMu(authentic)
    mu_authentic = np.asarray(mu_authentic, dtype=np.float64)

    mu_fake = getMu(fake)
    mu_fake = np.asarray(mu_fake, dtype=np.float64)

    Sw = no_authentic*np.cov(authentic.to_numpy()
```

```
                                [1:].T) + no_fake*np.cov(fake.to_numpy()[1:].T)
        Sw /= (no_authentic + no_fake)
        inv_S = np.linalg.inv(Sw)

        res = inv_S.dot(mu_authentic-mu_fake).T  # vector of coefficients

        mahalanobis_distance = sqrt(res.dot(mu_authentic - mu_fake))

        mean_vector_dim = (mu_authentic * no_authentic + mu_fake *
                           no_fake) / (2*(no_authentic + no_fake))
        mean_vec = [mean_vector_dim[i][0] for i in range(len(mean_vector_dim))]

        ratio = log(no_fake/no_authentic)

        df = pd.read_csv(test_set_path)
        authentic_test = df[df['class'] == 0]
        authentic_test = authentic_test.drop(columns=["class"])

        roc_auth = []
        tp = 0
        fn = 0
        for i in range(len(authentic_test)):
            roc_auth.append(float(res.dot(np.asarray(authentic_test)[i]-mean_vec)))
            if(res.dot(np.asarray(authentic_test)[i]-mean_vec) > ratio):
                tp += 1
            else:
                fn += 1

        fake_test = df[df['class'] == 1]
        fake_test = fake_test.drop(columns=["class"])

        roc_fake = []
        tn = 0
        fp = 0
        for i in range(len(fake_test)):
            roc_fake.append(float(res.dot(np.asarray(authentic_test)[i]-mean_vec)))
            if(res.dot(np.asarray(fake_test)[i]-mean_vec) <= ratio):
                tn += 1
            else:
                fp += 1

        return {"tp": tp, "tn": tn, "fp": fp, "fn": fn, "dist": mahalanobis_distanc


k = 5
acc_sum = 0
ppv_sum = 0
npv_sum = 0
recall_sum = 0
distance_sum = 0
```

```
f1_sum = 0

roc_auth = []
roc_fake = []

for i in range(k):
    result, roc_auth, roc_fake = lda("data_learning_set_" + str(i) + ".csv",
                                     "data_test_set_" + str(i) + ".csv")
    print("Mahalanobis distance: %0.3f" % result["dist"])
    print("FP: ", result["tp"])
    print("FN: ", result["fn"])
    print("TN: ", result["tn"])
    print("FT: ", result["fp"])
    acc = (result["tp"]+result["tn"]) / \
        (result["tp"]+result["tn"]+result["fp"]+result["fn"])
    ppv = (result["tp"])/(result["tp"]+result["fp"])
    npv = (result["tn"])/(result["fn"]+result["tn"])
    recall = result["tp"]/(result["tp"]+result["fn"])
    f1 = 2 * ppv * recall / (ppv + recall)
    print("Accuracy: %0.3f" % acc)  # Dokladnosc
    print("PPV: %0.3f" % ppv)  # Precyzja
    print("NPV: %0.3f" % npv)
    print("Recall: %0.3f" % recall)
    print("F1: %0.3f" % f1)
    print()
    distance_sum += result["dist"]
    acc_sum += acc
    ppv_sum += ppv
    npv_sum += npv
    recall_sum += recall
    f1_sum += f1

print("Mahalanobis distance mean: %0.3f" % (distance_sum/k))
print("Accuracy mean: %0.3f" % (acc_sum/k))  # Dokladnosc
print("PPV mean: %0.3f" % (ppv_sum/k))  # Precyzja
print("NPV mean: %0.3f" % (npv_sum/k))
print("Recall mean: %0.3f" % (recall_sum/k))
print("F1 mean: %0.3f" % (f1_sum/k))
```

```
Mahalanobis distance: 5.090
FP:  140
FN:  3
TN:  131
FT:  0
Accuracy: 0.989
PPV: 1.000
NPV: 0.978
Recall: 0.979
F1: 0.989

Mahalanobis distance: 5.129
FP:  150
FN:  0
TN:  124
FT:  0
Accuracy: 1.000
PPV: 1.000
NPV: 1.000
Recall: 1.000
F1: 1.000

Mahalanobis distance: 4.939
FP:  153
FN:  1
TN:  118
FT:  3
Accuracy: 0.985
PPV: 0.981
NPV: 0.992
Recall: 0.994
F1: 0.987

Mahalanobis distance: 5.136
FP:  162
FN:  2
TN:  108
FT:  2
Accuracy: 0.985
PPV: 0.988
NPV: 0.982
Recall: 0.988
F1: 0.988

Mahalanobis distance: 5.138
FP:  147
FN:  4
TN:  123
FT:  1
Accuracy: 0.982
```

```
PPV: 0.993
NPV: 0.969
Recall: 0.974
F1: 0.983

Mahalanobis distance mean: 5.086
Accuracy mean: 0.988
PPV mean: 0.992
NPV mean: 0.984
Recall mean: 0.987
F1 mean: 0.990
```