1. Постановка задачи

Осуществить визуализацию двух любых признаков и посчитать коэффициент корреляции между ними. Выполнить разбиение классов набора данных с помощью LDA (LinearDiscriminantAnalysis). Осуществить визуализацию разбиения. Осуществить классификацию с помощью методов LDA и QDA (LinearDiscriminantAnalysis и QuadraticDiscriminantAnalysis). Сравнить полученные результаты

2. Исходные данные

- Датасет: http://archive.ics.uci.edu/ml/datasets/seeds
- Предметная область: семена пшениц
- Задача: определить, к какому из 3х типов относится каждое семя (Kama, Rosa and Canadian)
- Количество записей: 210
- Количество атрибутов: 7
- Атрибуты:

1. area A,
2. perimeter P,
3. compactness C = 4*pi*A/P^2,
4. length of kernel,
5. width of kernel,
6. asymmetry coefficient
7. length of kernel groove.

3. Ход работы

```
from __future__ import division
import numpy
import pandas
from sklearn.model_selection import train_test_split
from matplotlib import pyplot as plt
from scipy.stats import pearsonr
from sklearn import preprocessing
from mpl_toolkits.mplot3d import Axes3D
from sklearn.discriminant_analysis import LinearDiscriminantAnalysis as LDA
from sklearn.discriminant_analysis import QuadraticDiscriminantAnalysis as QDA
from sklearn import metrics

def parse_dataset():
      ds = pandas.read_csv('seeds_dataset.txt', sep='\t', lineterminator='\n',
header=None).values
      ds_attributes = ds[:, :-1]
      ds_class = ds[:, -1].astype(numpy.int64, copy=False)
      return ds_attributes, ds_class


def train_split_dataset(occ_attr, occ_class, test_size, rnd_state):
      data_train, data_test, class_train, class_test =
train_test_split(occ_attr, occ_class, test_size=test_size,
random_state=rnd_state)

      print_dataset_info(class_train, data_train)
      print_dataset_info(class_test, data_test)

      return data_train, data_test, class_train, class_test


def visualize_data(is2d, is3d, is2plots, seed_attr=None, seed_class=None,
data_train=None, data_test=None,
                   class_train=None, class_test=None):
    if is2d is True:
        data_2d_visualization(seed_attr, seed_class)
    if is3d is True:
        data_3d_visualization(seed_attr, seed_class)
    if is2plots is True:
        train_test_visualization(data_train, data_test, class_train, class_test)


def data_2d_visualization(seed_attr, seed_class):
    plt.figure(figsize=(6, 5))
    for label, marker, color in zip(range(1, 4), ('x', 'o', '^'), ('red',
'blue', 'yellow')):
        r = pearsonr(seed_attr[:, 3][seed_class == label], seed_attr[:,
4][seed_class == label])
        plt.scatter(x=seed_attr[:, 3][seed_class == label],
                    y=seed_attr[:, 4][seed_class == label],
                    marker=marker,
                    color=color,
                    alpha=0.7,
                    label='class {:}, R={:.2f}'.format(label, r[0])
                    )

    plt.title('Seeds dataset')
    plt.xlabel('Length of kernel')
    plt.ylabel('Width of kernel')
    plt.legend(loc='upper right')
```

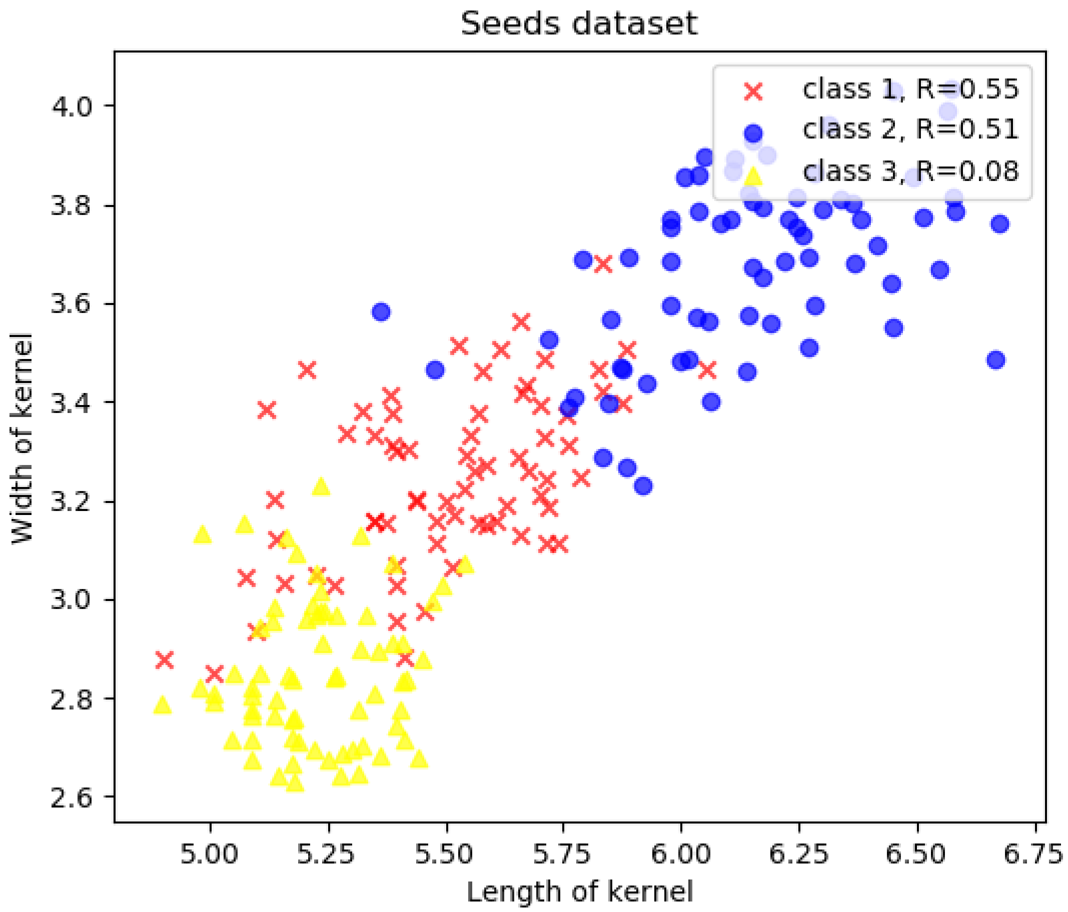
```
    plt.show()


def data_3d_visualization(seed_attr, seed_class):
    fig = plt.figure(figsize=(8, 8))
    ax = fig.add_subplot(111, projection='3d')
    for label, marker, color in zip(range(1, 4), ('x', 'o', '^'), ('red',
'blue', 'yellow')):
        ax.scatter(seed_attr[:, 2][seed_class == label],
                   seed_attr[:, 3][seed_class == label],
                   seed_attr[:, 4][seed_class == label],
                   marker=marker,
                   color=color,
                   s=40,
                   alpha=0.7,
                   label='class {:}'.format(label)
                   )

    ax.set_xlabel('Compactness')
    ax.set_ylabel('Length of kernel')
    ax.set_zlabel('Width of kernel')

    plt.title('Seeds dataset')
    plt.legend(loc='upper right')

    plt.show()


def train_test_visualization(data_train, data_test, class_train, class_test):
    std_scale = preprocessing.StandardScaler().fit(data_train)
    data_train = std_scale.transform(data_train)
    data_test = std_scale.transform(data_test)
    f, ax = plt.subplots(1, 2, sharex=True, sharey=True, figsize=(10, 5))

    for a, x_dat, y_lab in zip(ax, (data_train, data_test), (class_train,
class_test)):

        for label, marker, color in zip(
                range(1, 4), ('x', 'o', '^'), ('red', 'blue', 'yellow')):
            a.scatter(x=x_dat[:, 3][y_lab == label],
                      y=x_dat[:, 4][y_lab == label],
                      marker=marker,
                      color=color,
                      alpha=0.7,
                      label='class {}'.format(label)
                      )

        a.legend(loc='upper right')

    ax[0].set_title('Seeds training Dataset')
    ax[1].set_title('Seeds test Dataset')
    f.text(0.5, 0.04, 'Length of kernel (standardized)', ha='center',
va='center')
    f.text(0.08, 0.5, 'Width of kernel (standardized)', ha='center',
va='center', rotation='vertical')
    plt.show()


def linear_discriminant_analysis(data_train, class_train):
    sklearn_lda = LDA()
    sklearn_transf = sklearn_lda.fit(data_train,
class_train).transform(data_train)
```

```
    plt.figure(figsize=(8, 8))
    for label, marker, color in zip(
            range(1, 4), ('x', 'o', '^'), ('red', 'blue', 'yellow')):
        plt.scatter(x=sklearn_transf[class_train == label],
                    y=sklearn_transf[class_train == label],
                    marker=marker,
                    color=color,
                    alpha=0.7,
                    label='class {}'.format(label))

    plt.xlabel('vector 1')
    plt.ylabel('vector 2')

    plt.legend()
    plt.title('Most significant singular vectors after linear transformation via
LDA')

    plt.show()


def discriminant_analysis(fanalysis, data_train, data_test, class_train,
class_test, label):
    fanalysis.fit(data_train, class_train)
    pred_train = fanalysis.predict(data_train)

    print(label)
    print('The accuracy of the classification on the training set of data')
    print('{:.2%}'.format(metrics.accuracy_score(class_train, pred_train)))

    pred_test = fanalysis.predict(data_test)

    print('The accuracy of classification on the test data set')
    print('{:.2%}'.format(metrics.accuracy_score(class_test, pred_test)))


def print_dataset_info(seed_class, seed_attr):
    print('Number of records:', seed_class.shape[0])
    print('Number of characters:', seed_attr.shape[1])

    print('Class 0 (Kama): {:.2%}'.format(list(seed_class).count(1) /
seed_class.shape[0]))
    print('Class 1 (Rosa): {:.2%}'.format(list(seed_class).count(2) /
seed_class.shape[0]))
    print('Class 2 (Canadian): {:.2%}'.format(list(seed_class).count(3) /
seed_class.shape[0]))


def main():
    seed_attr, seed_class = parse_dataset()
    print_dataset_info(seed_class, seed_attr)

    data_train, data_test, class_train, class_test =
train_split_dataset(seed_attr, seed_class, 0.3, 55)

    visualize_data(is2d=True, is3d=True, is2plots=True, seed_attr=seed_attr,
seed_class=seed_class,
                   data_train=data_train, data_test=data_test,
class_train=class_train, class_test=class_test)

    linear_discriminant_analysis(data_train, class_train)
```

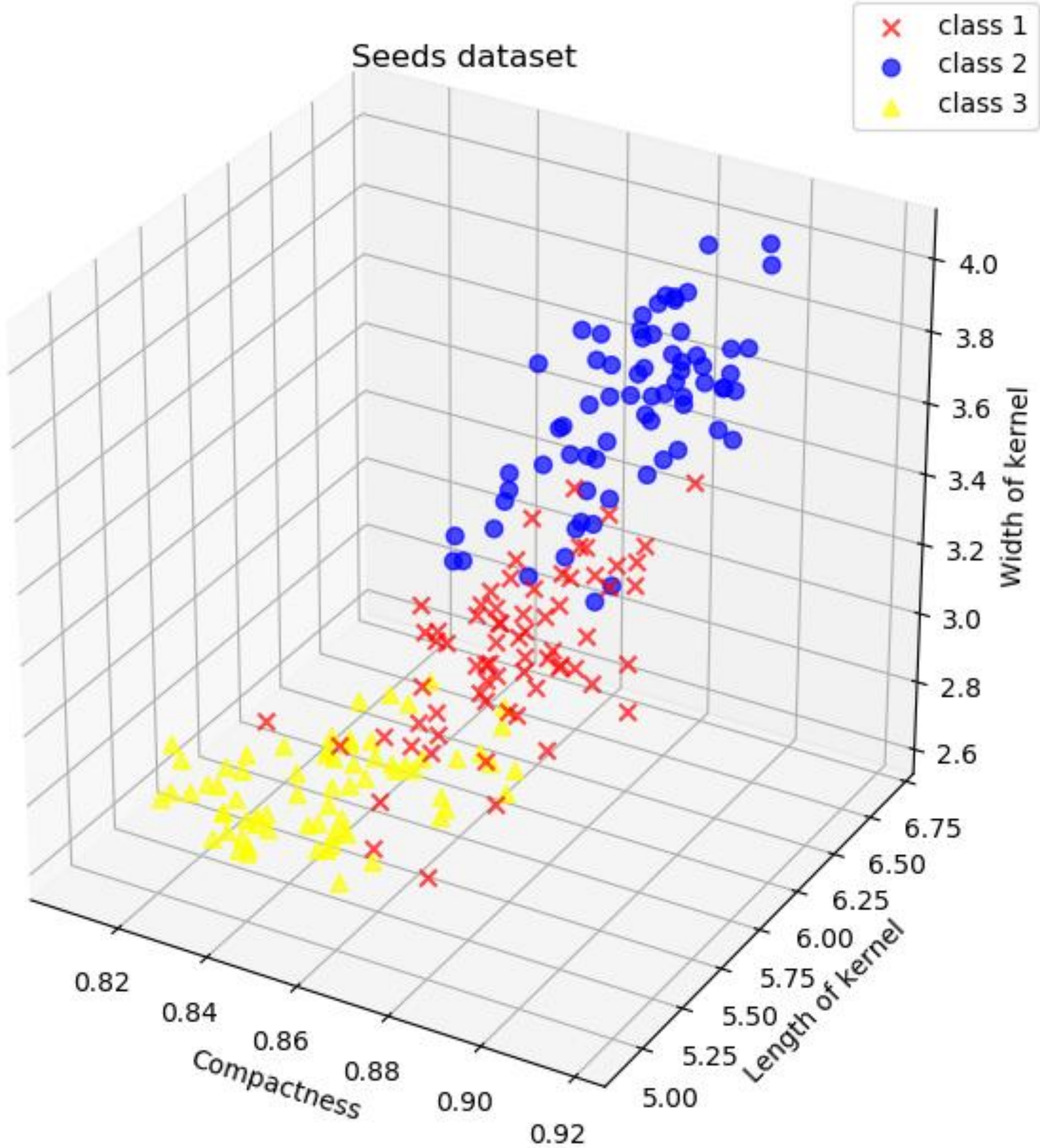
```
    discriminant_analysis(LDA(), data_train, data_test, class_train, class_test,
'LDA')
    discriminant_analysis(QDA(), data_train, data_test, class_train, class_test,
'QDA')


if __name__ == '__main__':
    main()
```

4. Результаты

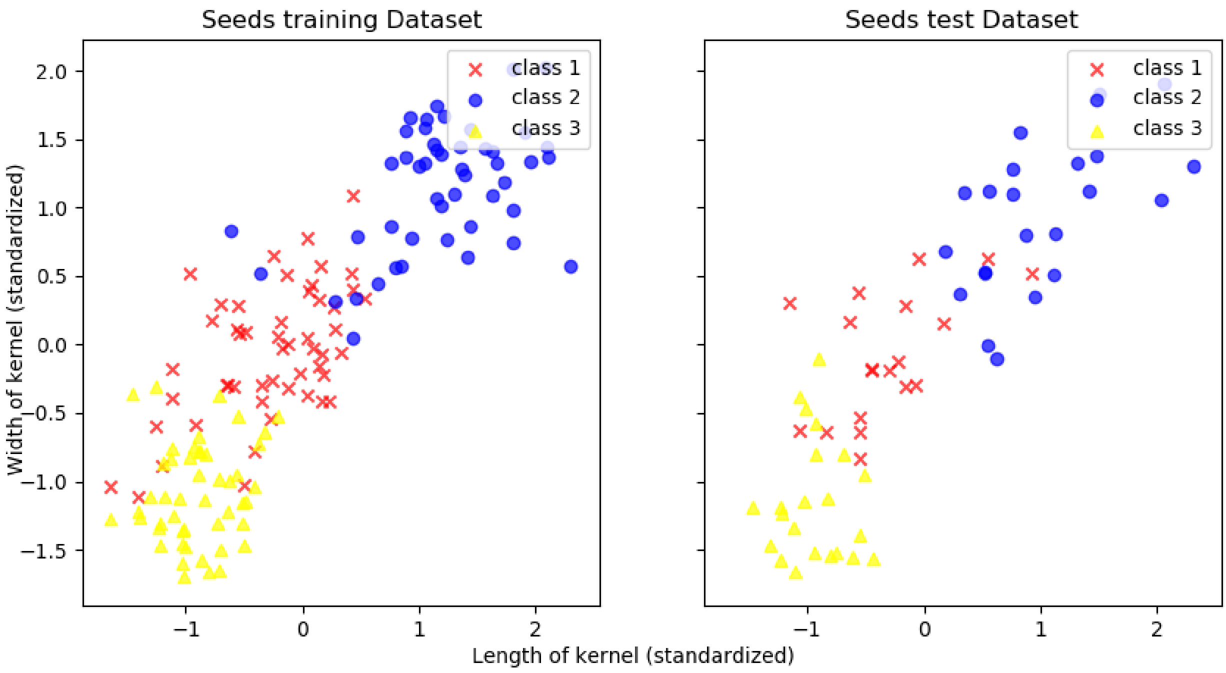
Визуализация параметров “4. length of kernel” и “5. width of kernel” :



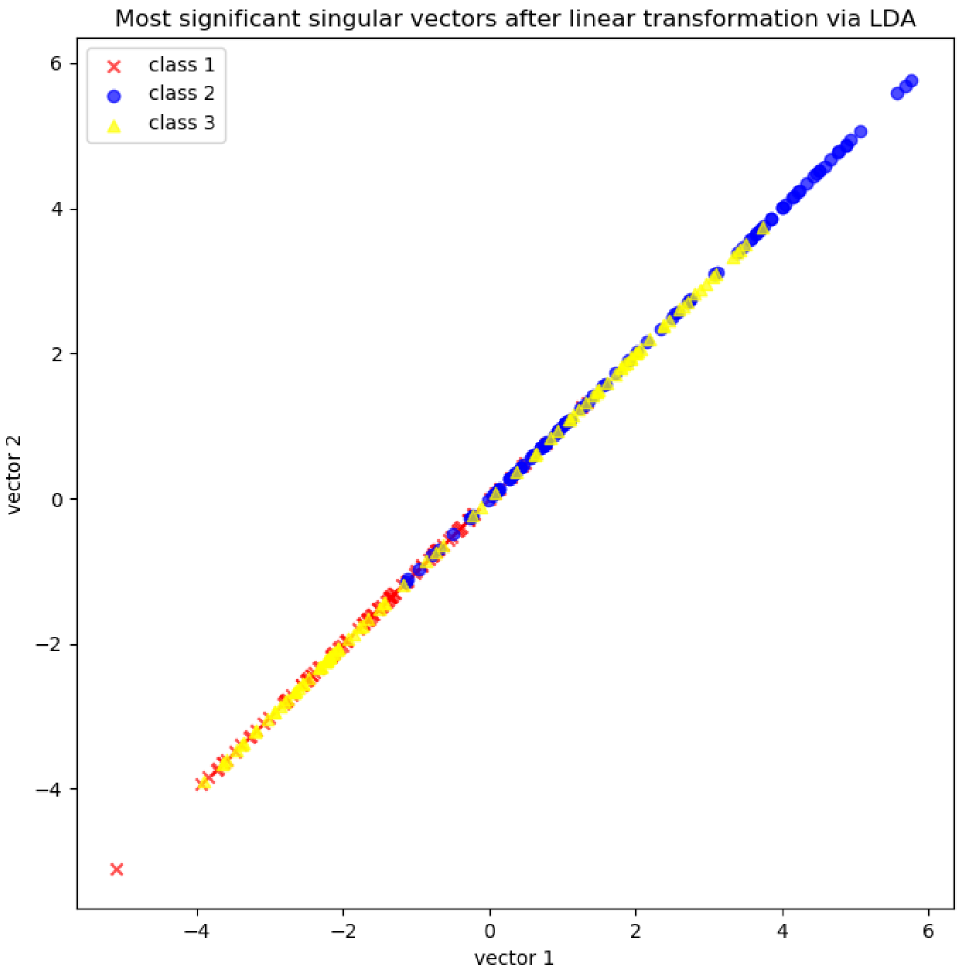
Добавление третьего параметра “3. compactness”:



Визуализация обучающего и тестового набора данных:



Визуализация разбиения классов после линейного преобразования LDA:



Number of records: 210
Number of characters: 7
Class 0 (Kama): 33.33%
Class 1 (Rosa): 33.33%
Class 2 (Canadian): 33.33%

Number of records: 147
Number of characters: 7
Class 0 (Kama): 34.69%
Class 1 (Rosa): 32.65%
Class 2 (Canadian): 32.65%

Number of records: 63
Number of characters: 7
Class 0 (Kama): 30.16%
Class 1 (Rosa): 34.92%
Class 2 (Canadian): 34.92%

Linear discriminant analysis
The accuracy of the classification on the training set of data 96.60%
The accuracy of classification on the test data set 96.83%

Quadratic discriminant analysis
The accuracy of the classification on the training set of data 95.24%
The accuracy of classification on the test data set 95.24%

Наивысшую точность на обучающем наборе данных показал линейный дискриминантный анализ. На тестовом наборе данных наивысшую точность показал так же линейный дискриминантный анализ.