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

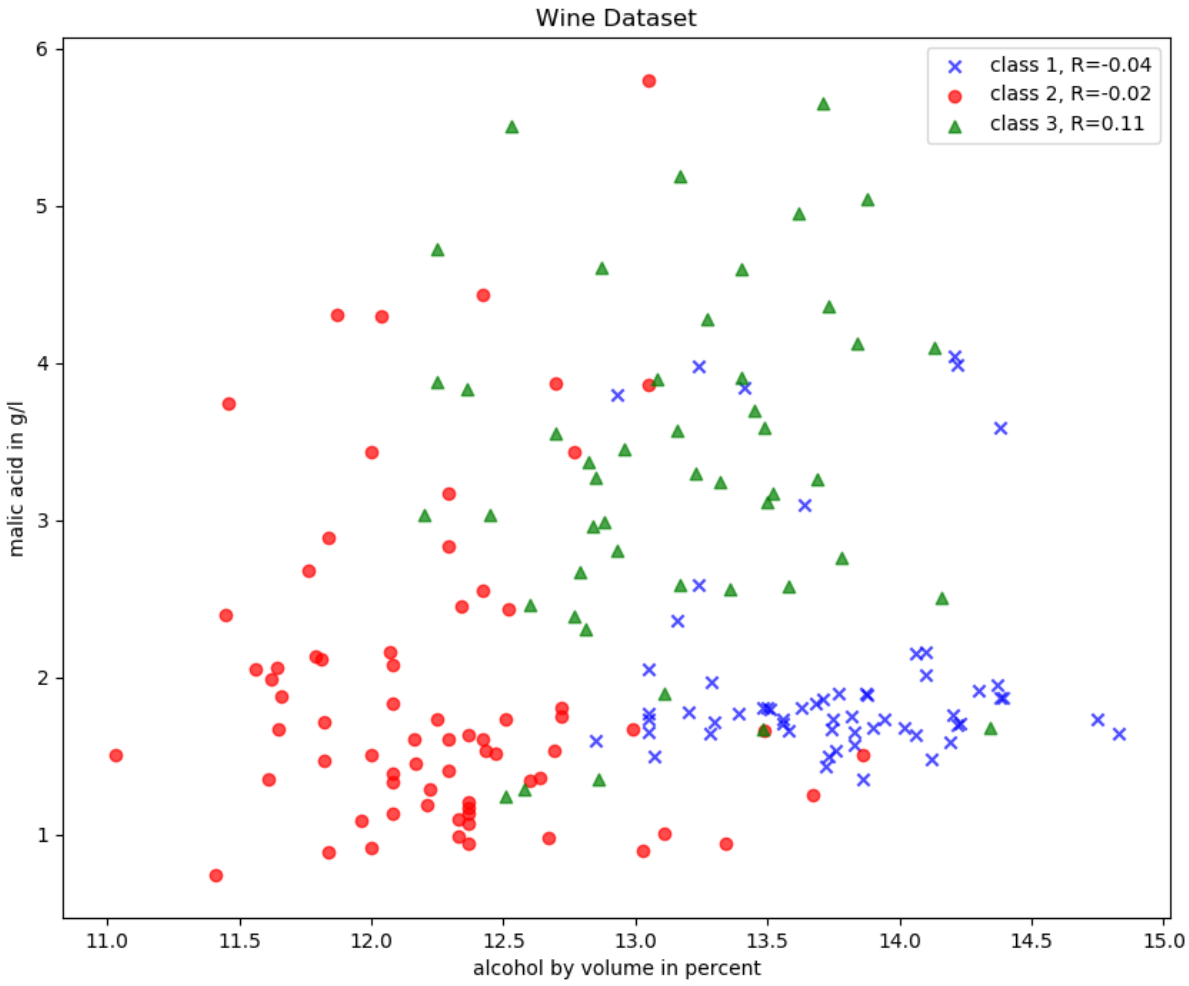
2. Исходные данные

- Датасет: https://archive.ics.uci.edu/ml/datasets/Wine
- Предметная область: Состав вина разного географического происхождения
- Задача: определить, в какой из 3 областей произведено вино
- Количество записей: 178
- Количество атрибутов: 13
- Атрибуты:
  1. Алкоголь
  2. Малиновая кислота
  3. Зола
  4. Алкалиния золы
  5. Магний
  6. Всего фенолов
  7. Флаваноиды
  8. Нефлаваноидные фенолы
  9. Проантоцианы
  10. Интенсивность цвета
  11. Оттенок
  12. OD280 / OD315 разведенных вин
  13. пролин

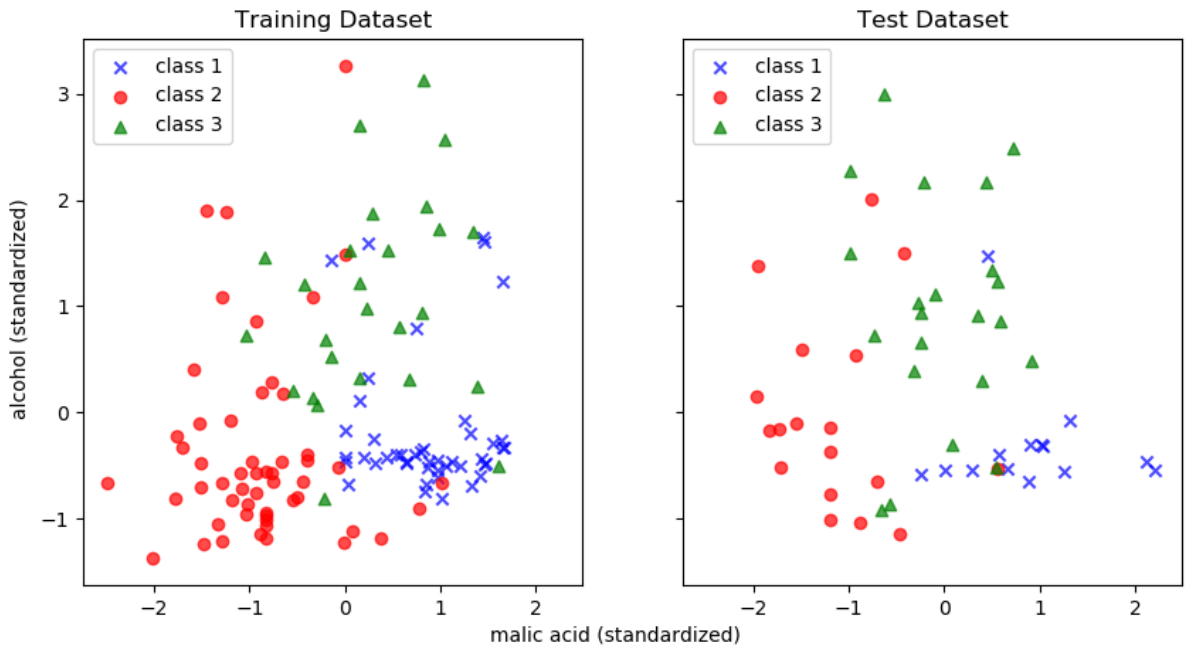
2. Ход работы

```
from __future__ import division
import numpy as np
import pandas as pd
from matplotlib import pyplot as plt
from scipy.stats import pearsonr
from sklearn.cross_validation import train_test_split
from sklearn import preprocessing
from sklearn.discriminant_analysis import LinearDiscriminantAnalysis as LDA
from sklearn.discriminant_analysis import QuadraticDiscriminantAnalysis as QDA
from sklearn import metrics


all_data = np.loadtxt(open("data.csv","r"),
        delimiter=",",
        skiprows=0,
        dtype=np.float64
        )

y_wine = all_data[:,0]
y_wine = y_wine.astype(np.int64, copy=False)
X_wine = all_data[:,1:]
```

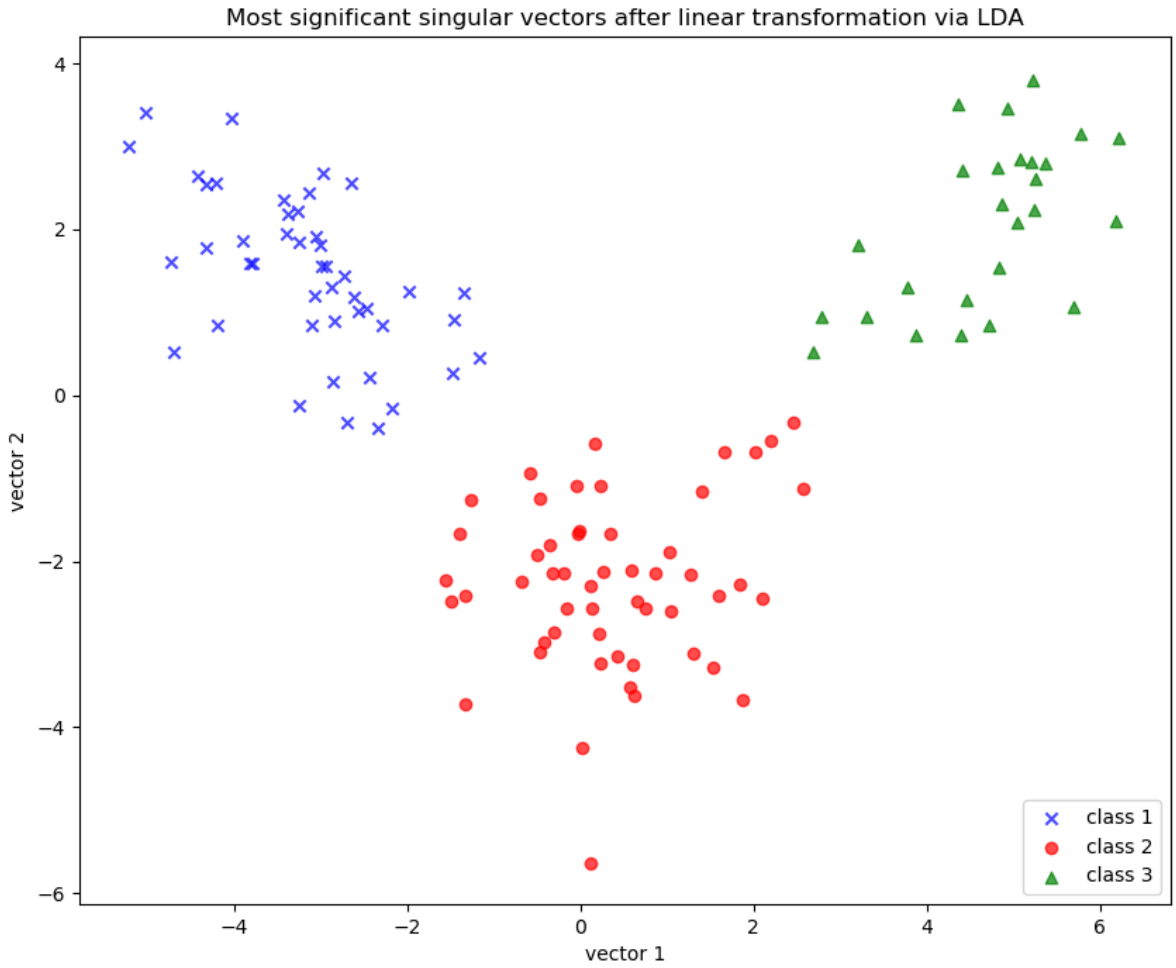
```
print('Rows count: ' + format(X_wine.shape[0]))
print('Params count: ' + format(X_wine.shape[1]))
print('\nPercent of each class:')
print('1 Class: {:.2%}'.format(list(y_wine).count(1)/X_wine.shape[0]))
print('2 Class: {:.2%}'.format(list(y_wine).count(2)/y_wine.shape[0]))
print('3 Class: {:.2%}\n'.format(list(y_wine).count(3)/y_wine.shape[0]))

plt.figure(figsize=(10,8))

for label,marker,color in zip(
        range(1,4),('x', 'o', '^'),('blue', 'red', 'green')):

    # Вычисление коэффициента корреляции Пирсона
    R = pearsonr(X_wine[:,0][y_wine == label], X_wine[:,1][y_wine == label])
    plt.scatter(x=X_wine[:,0][y_wine == label],
                y=X_wine[:,1][y_wine == label],
                marker=marker,
                color=color,
                alpha=0.7,
                label='class {:}, R={:.2f}'.format(label, R[0]) # label for the
legend
                )

plt.title('Wine Dataset')
plt.xlabel('alcohol by volume in percent')
plt.ylabel('malic acid in g/l')
plt.legend(loc='upper right')
plt.show()
```



```
# Random split
# Тестовый набор - 30%
# Обучающий набор - 70%
X_train, X_test, y_train, y_test = train_test_split(X_wine, y_wine,
     test_size=0.30, random_state=123)

print('Percent of each class')
print('\nTraining set:')
for l in range(1,4):
    print('Class {:}: {:.2%}'.format(l, list(y_train).count(l)/y_train.shape[0]))
print('\nTest set:')
for l in range(1,4):
    print('Class {:}: {:.2%}'.format(l, list(y_test).count(l)/y_test.shape[0]))

std_scale = preprocessing.StandardScaler().fit(X_train)
X_train = std_scale.transform(X_train)
X_test = std_scale.transform(X_test)

f, ax = plt.subplots(1, 2, sharex=True, sharey=True, figsize=(10,5))

for a,x_dat, y_lab in zip(ax, (X_train, X_test), (y_train, y_test)):
    for label,marker,color in zip(
        range(1,4),('x', 'o', '^'),('blue','red','green')):
        a.scatter(x=x_dat[:,0][y_lab == label],
                y=x_dat[:,1][y_lab == label],
                marker=marker,
                color=color,
                alpha=0.7,
                label='class {}'.format(label)
                )
    a.legend(loc='upper left')

ax[0].set_title('Training Dataset')
ax[1].set_title('Test Dataset')
f.text(0.5, 0.04, 'malic acid (standardized)', ha='center', va='center')
f.text(0.08, 0.5, 'alcohol (standardized)', ha='center', va='center',
rotation='vertical')
plt.show()
```



```
sklearn_lda = LDA(n_components=2)
sklearn_transf = sklearn_lda.fit_transform(X_train, y_train)
plt.figure(figsize=(10,8))
for label,marker,color in zip(
        range(1,4),('x', 'o', '^'),('blue', 'red', 'green')):
    plt.scatter(x=sklearn_transf[:,0][y_train == label],
                y=sklearn_transf[:,1][y_train == label],
                marker=marker,
                color=color,
                alpha=0.7,
                label='class {}'.format(label)
                )

plt.xlabel('vector 1')
plt.ylabel('vector 2')
plt.legend(loc='lower right')

# Визуализация разбиения классов после линейного преобразования LDA
plt.title('Most significant singular vectors after linear transformation via LDA')
plt.show()
```




```
# Обучение модели
lda_clf = LDA()
lda_clf.fit(X_train, y_train)
LDA(n_components=None, priors=None)
pred_train = lda_clf.predict(X_train)
print('LDA')
print('Accurancy of classification on training set ' +
'{:.2%}'.format(metrics.accuracy_score(y_train, pred_train)))
```

```
pred_test = lda_clf.predict(X_test)

print('Accurancy of classification on test set ' +
'{:.2%}'.format(metrics.accuracy_score(y_test, pred_test)))

lda_clf = QDA()
lda_clf.fit(X_train, y_train)
LDA(n_components=None, priors=None)
pred_train = lda_clf.predict(X_train)
print('QDA')

print('Accurancy of classification on training set ' +
'{:.2%}'.format(metrics.accuracy_score(y_train, pred_train)))

pred_test = lda_clf.predict(X_test)

print('Accurancy of classification on test set ' +
'{:.2%}'.format(metrics.accuracy_score(y_test, pred_test)))
```

Результаты:

Rows count: 178
Params count: 13

Percent of each class:
1 Class: 33.15%
2 Class: 39.89%
3 Class: 26.97%

Percent of each class:

Training set:
Class 1: 36.29%
Class 2: 42.74%
Class 3: 20.97%

Test set:
Class 1: 25.93%
Class 2: 33.33%
Class 3: 40.74%

LDA
Accurancy of classification on training set 100.00%
Accurancy of classification on test set 98.15%

QDA
Accurancy of classification on training set 99.19%
Accurancy of classification on test set 96.30%

Согласно полученным результатам, на обучающем наборе данных 100% точность показал алгоритм линейного дискриминантного анализа. Наивысшую точность на тестовом наборе показал также алгоритм LDA – 98.15%, в то время как точность алгоритма квадратичного дискриминантного анализа на тестовом наборе равна 96.3%.