Университет ИТМО

Машинное обучение
Лабораторная работа №3

Студент: Маскайкин А.В.
Группа: P4117

Санкт-Петербург
2017г

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/Website+Phishing
- Предметная область: Фишинговые сайты
- Задача: определить, фишинговый, подозрительный или нормальный сайт
- Количество записей: 1353
- Количество атрибутов: 9
- Атрибуты:
  1. SFH {1,-1,0}
  2. Pop-up Window {1,-1,0}
  3. SSL final state {1,-1,0}
  4. Request URL {1,-1,0}
  5. URL of Anchor {1,-1,0}
  6. Web traffic {1,-1,0}
  7. URL Length {1,-1,0}
  8. Age of domain {1,-1}
  9. Having IP Address {1,-1}

Во всех характеристиках значение «-1» означает «фишинговый», «0» - подозрительный, «1» - нормальный.

2.1 Описание параметров

- SFH (Server from handler) — Представление пользовательской информации, которая передается из веб страницы на сервер. Если оно пустое — сайт фишинговый, если передача идет на другой домен — подозрительный.
- Pop-up Window — Наличие всплывающего окна. Если при окне не доступен правый клик, то сайт фишинговый.
- SSL final state — Подлинность SSL сертификата.
- Request URL — Количество запросов к веб странице. Если их много, то, вероятно, сайт подвергся атаке, которая заменяет содержимое (текст/картинки). Если количество запросов велико — сайт фишинговый.
- URL of Anchor — привязка к URL. Если при вводе адреса сайта в браузере происходит редирект на другой домен, то привязки нет. И если процент редиректов большой — сайт фишинговый.

- Web traffic — объем веб трафиика сайта. У нормальных сайтов объем высокий, у фишинговых — низкий.
- URL Length — Длина адреса сайта. Чем больше длина, тем выше вероятность, что в адрес встроен вредоносный код.
- Age of domain — Возраст сайта. Если сайт существует менее полугода, то его можно заподозрить как фишинговый.
- Having IP Address — Наличие IP адреса. Если адреса нет — сайт фишинговый.

3. Реализация алгоритма.

```
# coding=utf-8
from __future__ import division
import numpy as np
import pandas as pd
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 load_data(filename):
    return pd.read_csv(filename, header=None).values

# разделение датасета на тестовую и обучающую выборку
def process_dataset(name):
    dataset = load_data(name)
    site_attr = dataset[:, :-1]  # список атрибутов для каждого сайта
    site_class = dataset[:, -1]  # класс (результат) сайта (норм,
подозрительный, фишинговый)
    site_class = site_class.astype(np.int64, copy=False)
    return site_attr, site_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)
    # train_test_visualization(data_train, data_test, class_train, class_test)
    return data_train, data_test, class_train, class_test

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

```
def data_2d_visualization(site_attr, site_class):
    plt.figure(figsize=(6, 5))
    for label, marker, color in zip(
            range(-1, 2), ('x', 'o', '^'), ('blue', 'red', 'green')):
        # Вычисление коэффициента корреляции Пирсона
        r = pearsonr(site_attr[:, 5][site_class == label], site_attr[:, 6]
[site_class == label])
        plt.scatter(x=site_attr[:, 5][site_class == label],
                    y=site_attr[:, 6][site_class == label],
                    marker=marker,
                    color=color,
                    alpha=0.7,
                    label='class {:}, R={:.2f}'.format(label, r[0])
                    )
    plt.title('Phishing Website Data Set')
    plt.xlabel('Web Traffic')
    plt.ylabel('URL Length')
    plt.legend(loc='upper right')
    plt.show()

def data_3d_visualization(site_attr, site_class):
    fig = plt.figure(figsize=(8, 8))
    ax = fig.add_subplot(111, projection='3d')
    for label, marker, color in zip(
            range(-1, 2), ('x', 'o', '^'), ('blue', 'red', 'green')):
        # Вычисление коэффициента корреляции Пирсона
        ax.scatter(site_attr[:, 1][site_class == label],
                   site_attr[:, 5][site_class == label],
                   site_attr[:, 6][site_class == label],
                   marker=marker,
                   color=color,
                   s=40,
                   alpha=0.7,
                   label='class {:}'.format(label)
                   )
    ax.set_xlabel('popUpWidnow')
    ax.set_ylabel('Web Traffic')
    ax.set_zlabel('URL Length')
    plt.title('Phishing Website Data Set')
    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, 2), ('x', 'o', '^'), ('blue', 'red', 'green')):
            a.scatter(x=x_dat[:, 5][y_lab == label],
                      y=x_dat[:, 6][y_lab == label],
                      marker=marker,
                      color=color,
                      alpha=0.7,
                      label='class {}'.format(label)
```

```
                    )
        a.legend(loc='upper right')
    ax[0].set_title('Training Dataset')
    ax[1].set_title('Test Dataset')
    f.text(0.5, 0.04, 'Web Traffic (standardized)', ha='center', va='center')
    f.text(0.08, 0.5, 'URL Length (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, 2), ('x', 'o', '^'), ('blue', 'red', 'green')):
        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()
    # Визуализация разбиения классов после линейного преобразования LDA
    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(site_class, site_attr):
    print('Number of records:', site_class.shape[0])
    print('Number of characters:', site_attr.shape[1])
    print('Class 0 (Normal): {:.2%}'.format(list(site_class).count(-1) /
site_class.shape[0]))
    print('Class 1 (Suspicious): {:.2%}'.format(list(site_class).count(0) /
site_class.shape[0]))
    print('Class 2 (Phishing): {:.2%}'.format(list(site_class).count(1) /
site_class.shape[0]))

def init(name):
    site_attr, site_class = process_dataset(name)
    print_dataset_info(site_class, site_attr)
    return site_attr, site_class

def main():
    site_attr, site_class = init('fs.dataset.csv')
```

```
    data_train, data_test, class_train, class_test =
train_split_dataset(site_attr, site_class, 0.3, 55)
    visualize_data(is2d=True, is3d=True, is2plots=True, site_attr=site_attr,
site_class=site_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,
'Linear discriminant analysis')
    discriminant_analysis(QDA(), data_train, data_test, class_train, class_test,
'Quadratic discriminant analysis')

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

4. Результаты работы.

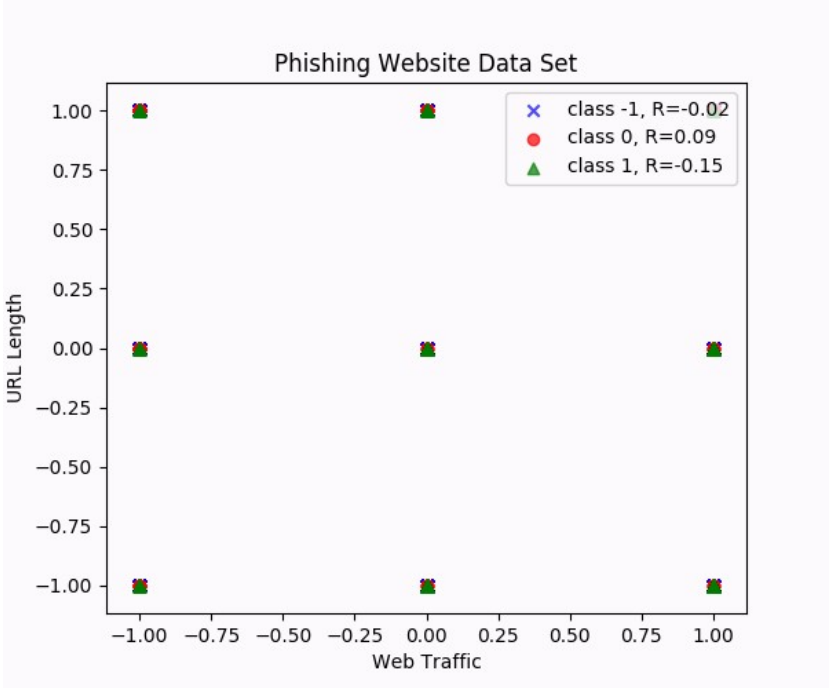
Распределение по классам.

Доли каждого класса по отношению ко всему объему датасета:

Class 0 (Normal): 51.88%
Class 1 (Suspicious): 7.61%
Class 2 (Phishing): 40.50%

Обучающий набор:
('Number of records:', 947)
('Number of characters:', 9)
Class 0 (Normal): 52.27%
Class 1 (Suspicious): 6.76%
Class 2 (Phishing): 40.97%

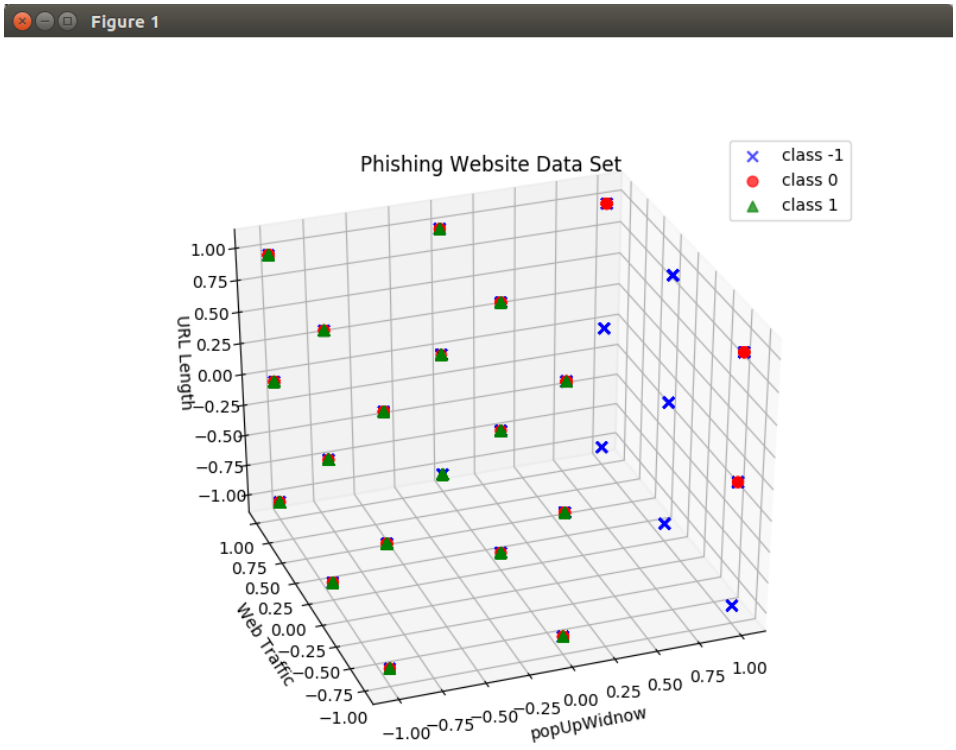
Тестовый набор:
('Number of records:', 406)
('Number of characters:', 9)
Class 0 (Normal): 50.99%
Class 1 (Suspicious): 9.61%
Class 2 (Phishing): 39.41%



Визуализация *Иллюстрация 1: Визуализация*

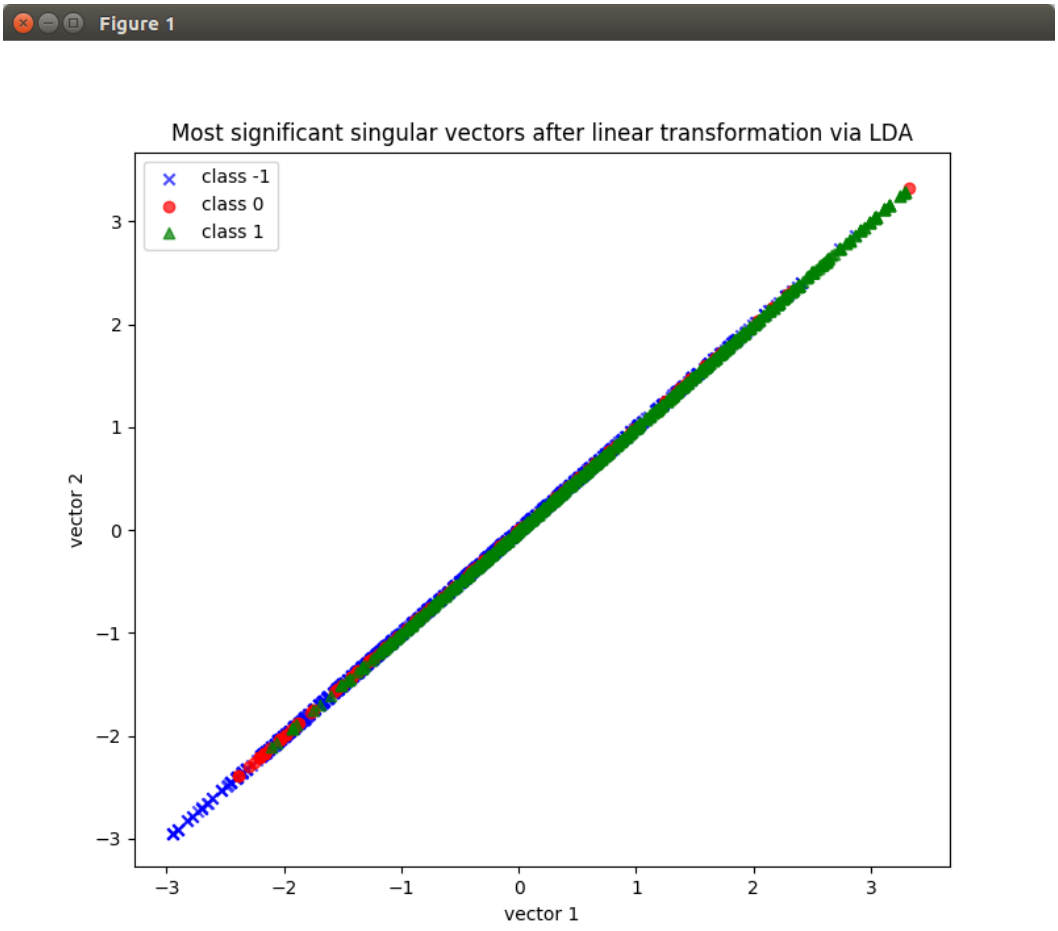
Визуализация двух признаков: "Web Traffic" и "URL Length". R - значение коэффициента корреляции Пирсона между этими двумя признаками. (Иллюстрация 1).

Визуализация трех признаков: "Web Traffic", "URL Length" и "Pop Up window".



Линейный дискриминантный анализ

Визуализация разбиения классов после линейного преобразования LDA:



Точность классификации на обучающем наборе данных

84.16%

Точность классификации на тестовом наборе данных

77.83%

Квадратичный дискриминантный анализ

Точность классификации на обучающем наборе данных

87.96%

Точность классификации на тестовом наборе данных

80.30%

По полученным результатам видно, что классификация с использованием квадратичного дискриминантного анализа незначительно точнее линейного — на 3%. Точность на обучающем наборе данных оказалась выше, чем на тестовом в обоих алгоритмах. Сравнивая точность классификации дискриминантных анализов с результатами алгоритмов классификации в предыдущих работах, можно сделать вывод, что точность дискриминантных алгоритмов схожи с результатами алгоритмов деревьев решений для данного датасета.

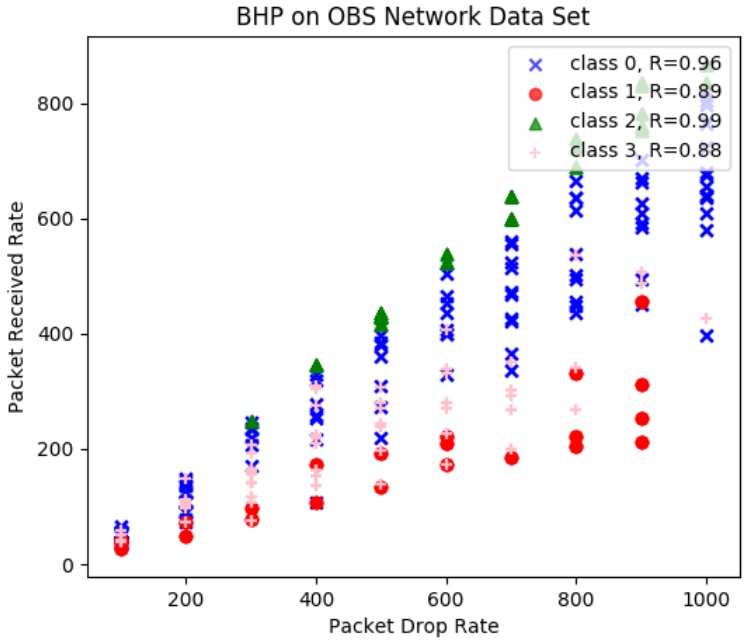
Дополнительная визуализация нового датасета

Полученные визуализации не совсем удачные, в виду ограниченности вариаций значений характеристик датасета: все параметры могут принимать три значения: -1, 0 и 1.

Поэтому ниже приведена визуализация другого датасета - Burst Header Packet (BHP) flooding attack on Optical Burst Switching (OBS) Network (https://archive.ics.uci.edu/ml/datasets/Burst+Header+Packet+%28BHP%29+flooding+attack+on+Optical+Burest+Switching+%28OBS%29+Network)
На данном датасете все разбиения на классы, обучающие и тестовые выборки более наглядны, т.к. параметры вариативны и принимают множество различных значений.

Задача данного датасета — определить вирусные ноды сети, которые и заблокировать их во избежание забивания канала данных, из-за чего остальные ноды сети не смогут использовать его. Результирующие классы - 'NB-No Block' (Нода ведет себя нормально, не заблокирована), Block (Нода заблокирована),'No Block' (Нода не заблокирована), NB-Wait (Нода ведет нормально, ждет).

Визуализация параметров «Количество полученных пакетов» и «Количество отброшенных пакетов».



Добавление третьего параметра «Процент потерянных пакетов» к предыдущей визуализации

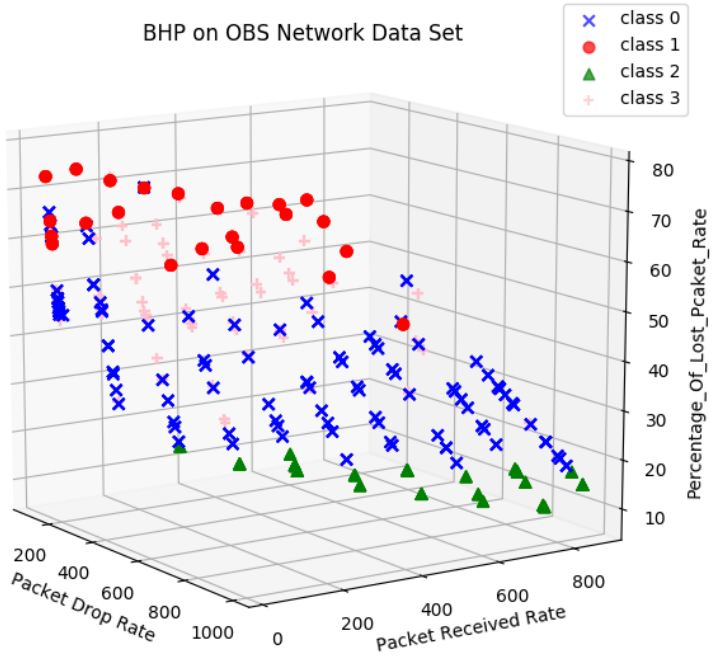
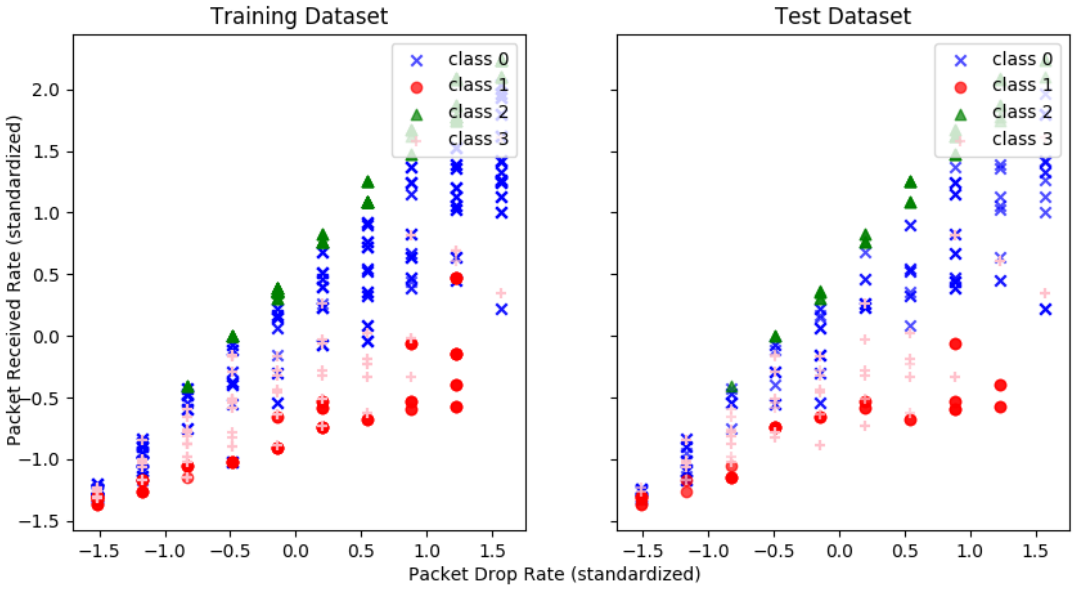
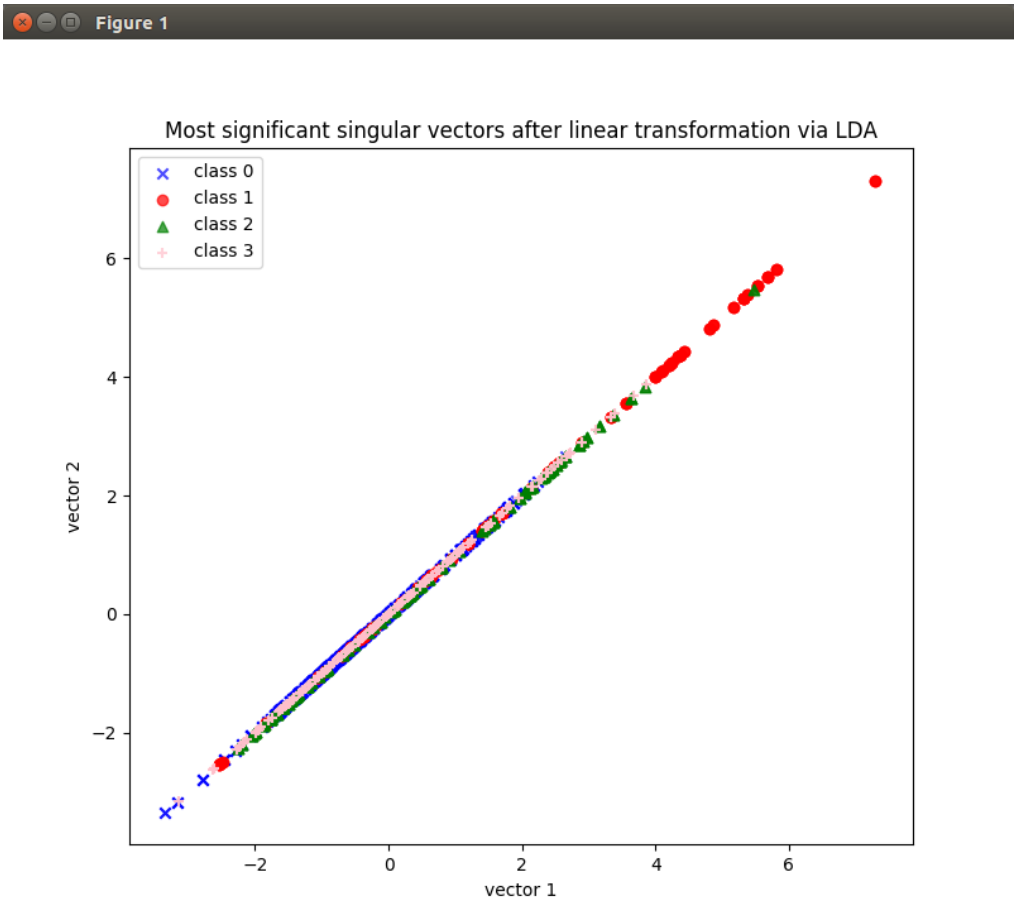


Иллюстрация обучающего и тестового наборов данных



Визуализация разбиения классов после линейного преобразования LDA:



На данном датасете разбиение классов по векторам после LDA нагляднее: в верхней части диагонали размещены точки класса1, в нижней половине — точки класса 0,2 и 3.