Санкт-Петербургский политехнический университет Петра Великого

Физико-механический институт

Высшая школа прикладной математики и вычислительной физики

Отчет по лабораторной работе №2 по дисциплине «Многомерный статистический анализ»

Выполнила студентка группы 5030102/90401: О. А. Ковалёва

Преподаватель: Л. В. Павлова

Санкт-Петербург
2023

Оглавление

Цель

Построить и провести тестирование классификатора многомерных объектов на основе дискриминантного анализа при наличии обучающей выборки. Используемые данные:

a) Модельные данные – многомерное нормальное распределение с заданными параметрами;
b) Данные из репозитория.

Ход работы

a) Работа с модельными данными.

1) Генерация данных

Сгенерируем две обучающих выборки и тестовую из нормальных трехмерных распределений с заданными параметрами:

$$\boldsymbol{x} \sim N\left(\mu^{(1)}, \Sigma_1\right)$$

$$\boldsymbol{x} \sim N\left(\mu^{(2)}, \Sigma_2\right)$$

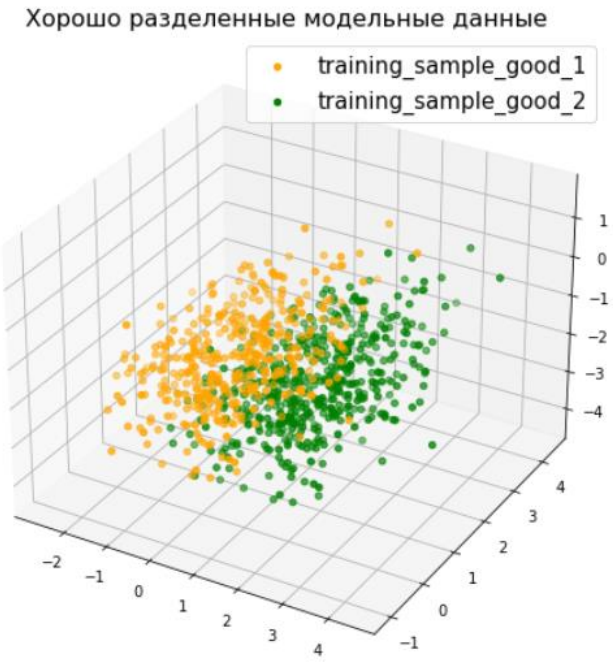
Рассмотрим «хорошо» и «плохо» разделенные данные. Для получения «плохо» разделенных данных увеличим зависимости между признаками, умножив ковариационную матрицу на 10.

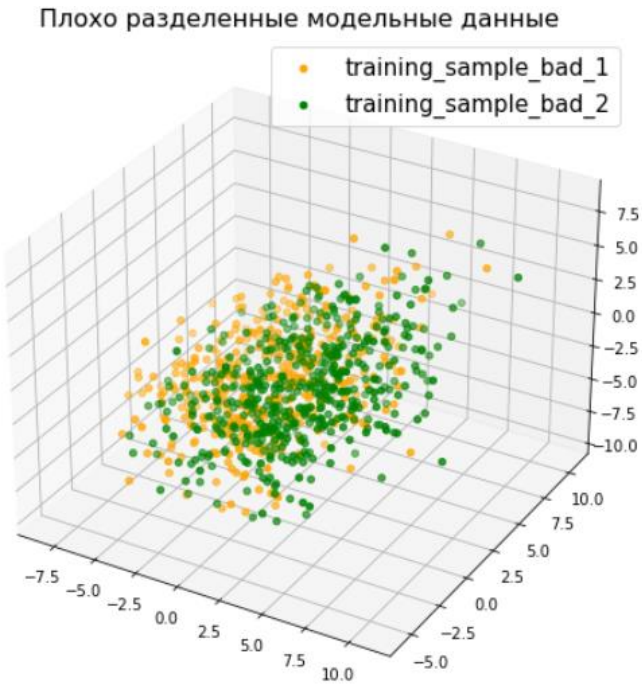
```
dim = 3 # число признаков
n1 = 400 # объем первой выборки
n2 = 500 # объем второй выборки
n = 1000 # объем тестовой выборки

q_training1 = n1 / (n1 + n2) # доля данных в обучающей выборке, взятых из первого распределения
q_training2 = n2 / (n1 + n2) # доля данных в обучающей выборке - из второго распределения
q1 = 0.4 # доля данных в тестовой выборке - из первого распределения
q2 = 1 - q1 # доля данных в тестовой выборке - из второго распределения

# параметры распредеелний
mu1 = np.array([0, 1, -1.5])
mu2 = np.array([1.5, 1.5, -2])
cov_matrix_good = np.array([[1.2, 0.2, 0.5], [0.2, 0.7, 0.3], [0.5, 0.3, 0.9]]) # "хорошо" разделенные данные
cov_matrix_bad = cov_matrix_good * 10 # "плохо" разделенные данные
```

Посмотрим на полученные обучающие выборки:





2) Вычисление выборочных оценок параметров распределений.

Оценка вектора средних значений:

$$\bar{\mu}_j^{(k)} = \frac{1}{n_k}\sum_{i=1}^{n_k} x_{ij}^{(k)}, k = 1,2, j = \overline{1,3}$$

Таким образом: $\bar{\mu}^{(k)} = \left(\bar{\mu}_1^{(k)}, \dots, \bar{\mu}_p^{(k)}\right)$, где $p -$ число признаков.

Оценка ковариационной матрицы по выборке:

$$S^{(k)} = \left(S_{lj}^{(k)}\right), l, j = \overline{1,p}, k = 1,2$$

где $S_{lj}^{(k)} = \frac{1}{n_k - 1}\sum_{i=1}^{n_k}(x_{il}^{(k)} - \bar{\mu}_l^{(k)})(x_{ij}^{(k)} - \bar{\mu}_j^{(k)}), k = 1,2.$

Так как $\Sigma_1 = \Sigma_2 = \Sigma$, усредним:

$$S = \frac{1}{n_1 + n_2 - 2}[(n_1 - 1)S^{(1)} + (n_2 - 1)S^{(2)}]$$

Реализуем функции, вычисляющие оценки:

```python
def multivariate_mean(sample, size, dim_): # многомерное выборочное среднее
    mu = np.array([])
    for j in range(dim_):
        mu = np.append(mu, np.sum(sample[:, j]) / size)
    return mu

def sample_S(sample, size, mu_est, dim_): # выборочная ковариационная матрица
    S = np.zeros(shape=(dim_, dim_))
    for l in range(dim_):
        for j in range(dim_):
            S[l, j] = sum((sample[i, l] - mu_est[l])*(sample[i, j] - mu_est[j]) for i in range(size)) / (size - 1)
    return S

# усредненная выборочная ковариационная матрица
def multivariate_cov_matrix(sample1, sample2, size1, size2, multivariate_mu_est, dim_):
    S1 = sample_S(sample1, size1, multivariate_mu_est[0], dim_)
    S2 = sample_S(sample2, size2, multivariate_mu_est[1], dim_)

    S = ((size1 - 1) * S1 + (size2 - 1) * S2) / (size1 + size2 - 2)
    return S
```

Полученные оценки для «хорошо» разделенных данных:

```
Выборочные оценки парметров распределений:
 Среднее: [ 0.03259111  1.04239009 -1.47577711],
          [ 1.50272217  1.51016601 -1.99153676]
 Ковариационная матрица:
[[1.01133908 0.10557364 0.37990203]
 [0.10557364 0.7285143  0.26631713]
 [0.37990203 0.26631713 0.82720077]]
```

```
Заданные значения парметров распределений:
 Среднее: [ 0.   1.  -1.5],
          [ 1.5  1.5 -2. ]
 Ковариационная матрица:
[[1.2 0.2 0.5]
 [0.2 0.7 0.3]
 [0.5 0.3 0.9]]
```

Рисунок 1. Оценки параметров распределения для "хорошо" разделенных данных.

Для «плохо» разделенных данных:

```
Выборочные оценки парметров распределений:
 Среднее: [ 0.10306214  1.13404925 -1.42340051],
          [ 1.50860827  1.53214776 -1.97323688]
 Ковариационная матрица:
[[10.11339075  1.05573639  3.7990203 ]
 [ 1.05573639  7.285143    2.66317129]
 [ 3.7990203   2.66317129  8.27200772]]
```

```
Заданные значения парметров распределений:
 Среднее: [ 0.   1.  -1.5],
          [ 1.5  1.5 -2. ]
 Ковариационная матрица:
[[12.  2.  5.]
 [ 2.  7.  3.]
 [ 5.  3.  9.]]
```

Рисунок 2. Оценки параметров распределения для "плохо" разделенных данных.

Заметим, что оценки по «хорошо» разделенным данным более близки к заданным параметрам. Так как «плохо» и «хорошо» разделенные данные отличаются только матрицей ковариаций, сравним оценки средних: по «хорошо» разделенным данным оценка немного ближе к истинному.

3) Построение классификатора

Используя полученные оценки, построим классификатор.

Дискриминантная функция:

$$z(\boldsymbol{x}) = \alpha_1 x_1 + \cdots + \alpha_p x_p,$$

вычисляя её значение на полученном на вход векторе, будем принимать решение, к какому классу его отнести.

Вычисляем коэффициенты:

$$\alpha \to \bar{\alpha} = S^{-1}(\bar{\mu}^{(1)} - \bar{\mu}^{(2)})$$

Далее по обучающим выборкам находим оценки:

$$\bar{z}^{(k)} = \frac{1}{n_k}\sum_{i=1}^{n_k} z_i^{(k)}, k = 1,2$$

$$\sigma_z^2 \to s_z^2 = \sum_{l=1}^{p}\sum_{j=1}^{p} \alpha_l s_{lj} \alpha_j$$

Тогда условие выбора класса:

$$\boldsymbol{x} \to \begin{cases} D_1 : z(\boldsymbol{x}) \geq \dfrac{\bar{z}^{(1)} + \bar{z}^{(2)}}{2} + ln\dfrac{\mathrm{q}_2 c(1|2)}{q_1 c(2|1)} \\ D_2 : z(\boldsymbol{x}) < \dfrac{\bar{z}^{(1)} + \bar{z}^{(2)}}{2} + ln\dfrac{\mathrm{q}_2 (1|2)}{q_1 c(2|1)} \end{cases}$$

где D_1, D_2 – классы, q_1, q_2 – доли данных в обучающей выборке, взятых из каждого распределения – априорные вероятности попадания в классы, а стоимости ошибочной классификации $c(1|2) = c(2|1) = 1$.

```python
def discriminant_fun(alpha, vect): # значение дискриминантной функции на векторе
    z = sum([a * x for a, x in zip(alpha, vect)])
    return z

# вычисление коэффициентов и среднего значения дискриминантной функции, порога принятия решения
def create_classifier(training1, training2, size1, size2, q1_, q2_, mu_est, cov_matrix_est):
    a = np.matmul(np.linalg.inv(cov_matrix_est), (mu_est[0] - mu_est[1]))
    mean_z1 = sum([discriminant_fun(a, training1[i, :]) for i in range(size1)]) / size1
    mean_z2 = sum([discriminant_fun(a, training2[i, :]) for i in range(size2)]) / size2
    mean_z = np.array([mean_z1, mean_z2])
    threshold = (mean_z1 + mean_z2) / 2.0 + np.log(q2_ / q1_)
    return a, mean_z, threshold

def classify(vect, alpha, threshold): # принятие решения о классе
    z = discriminant_fun(alpha, vect)
    if z >= threshold:
        return 0
    else:
        return 1
```

4) Оценка вероятности ошибочной классификации

Будем искать оценку вероятности ошибочной классификации на основе долей неправильно полученных классификатором меток обучающей выборки от её размера. Для этого классифицируем обучающие выборки и находим вероятности:

$$\bar{P}(2|1) = \frac{m_1}{n_1}, \quad \bar{P}(1|2) = \frac{m_2}{n_2},$$

где m_1, m_2 – число полученных неправильных меток для первой и второй выборки, соответственно.

А также оценим расстояние Махаланобиса:

$D_H^2 = \frac{n_1+n_2-p-3}{n_1+n_2-2} D^2 - p\left(\frac{1}{n_1}+\frac{1}{n_2}\right)$, где $D^2 = \frac{\left(\bar{z}^{(1)}+\bar{z}^{(2)}\right)^2}{s_z^2}$ – смещенная оценка, D_H^2 – несмещенная.

Тогда искомые оценки ($c(1|2) = c(2|1) = 1$):

$$\bar{P}(2|1) = \Phi\left(\frac{K - 0.5D_H^2}{\sqrt{D_H^2}}\right), \bar{P}(1|2) = \Phi\left(\frac{-K - 0.5D_H^2}{\sqrt{D_H^2}}\right),$$

где $K = \ln\left(\frac{q_2 c(1|2)}{q_1 c(2|1)}\right)$, Φ – функция Лапласа.

Таким образом оценка общей вероятности ошибочной классификации:

$$\bar{P} = q_1 c(2|1)\bar{P}(2|1) + q_2 c(1|2)\bar{P}(1|2)$$

```python
def err_prob_est(m1, m2, size1, size2): # оценка по доле неправильных меток от размера выборки
    p21 = m1 / size1
    p12 = m2 / size2
    print(f"        P(2|1) = {p21:.5}")
    print(f"        P(1|2) = {p12:.5}")
    return p21, p12

def mahalanobis_biased_est(alpha, mean_z, cov_matrix_est): # смещенная оценка расстояния Махаланобиса
    dim_ = cov_matrix_est.shape[1]
    var_z = 0
    for l in range(dim_):
        for j in range(dim_):
            var_z += alpha[l] * cov_matrix_est[l, j] * alpha[j]

    return (mean_z[0] - mean_z[1]) ** 2 / var_z

def mahalanobis_unbiased_est(alpha, mean_z, cov_matrix_est, size1, size2): # несмещенная оценка расстояния Махаланобиса
    dim_ = cov_matrix_est.shape[1]
    D = mahalanobis_biased_est(alpha, mean_z, cov_matrix_est)
    D_unbiased = (size1 + size2 - dim_ - 3) * D / (size1 + size2 - 2) - dim_ * (1 / size1 +  1 / size2)
    return D_unbiased
```

```python
def mahalanobis_err_prob_est(alpha, mean_z, cov_matrix_est, q1_, q2_): # оценка по расстоянию Махаланобиса
    K = np.log(q2_ / q1_)
    mah_biased_est = mahalanobis_biased_est(alpha, mean_z, cov_matrix_est)
    p21 = norm.cdf((K - 0.5 * mah_biased_est) / mah_biased_est ** 0.5)
    p12 = norm.cdf((-K - 0.5 * mah_biased_est) / mah_biased_est ** 0.5)
    print(f"        P(2|1) = {p21:.5}")
    print(f"        P(1|2) = {p12:.5}")
    return p21, p12

def overall_err_prob_est(p21, p12, q1_, q2_): # оценка общей вероятности ошибки
    return q1_ * p21 + q2_ * p12
```

Дальнейшую работу с данными и тестирование классификаторов будем выполнять по алгоритму:

1. *Предобработка данных (генерация/чтение/разбиение на выборки/PCA).*
2. *Вычисление выборочных оценок параметров распределения.*
3. *Построение классификатора по обучающим выборкам.*
4. *Вычисление оценок вероятности ошибочной классификации.*
5. *Применение классификатора к тестовой выборке*
6. *Анализ результатов: нахождение эмпирических вероятностей ошибочной классификации, построение таблицы сопряженности.*

Итак, был построен классификатор по обучающим выборкам модельных данных. Оценим вероятности ошибочной классификации, выполнив классификацию обучающих выборок:

```python
training_good_classification_res = np.array([])
training_bad_classification_res = np.array([])
training_sample_good = np.vstack((training_sample_good1, training_sample_good2))
training_sample_bad = np.vstack((training_sample_bad1, training_sample_bad2))

# Классифицируем обучающие выборки моделью, построенной по хорошо разделенным данным
for vect in training_sample_good:
    training_good_classification_res = np.append(training_good_classification_res,
                                                 classify(vect, alpha_good, threshold_good))
cm_training_good = confusion_matrix(training_sample_marks, training_good_classification_res)
tn_1, fp_1, fn_1, tp_1 = cm_training_good.ravel()
m1_good = fp_1
m2_good = fn_1

# Классифицируем обучающие выборки моделью, построенной по плохо разделенным данным
for vect in training_sample_bad:
    training_bad_classification_res = np.append(training_bad_classification_res,
                                                classify(vect, alpha_bad, threshold_bad))
cm_training_bad = confusion_matrix(training_sample_marks, training_bad_classification_res)
tn_2, fp_2, fn_2, tp_2 = cm_training_bad.ravel()
m1_bad = fp_2
m2_bad = fn_2
```

```
Классификтор, построенный по хорошо разделенным данным:
        Оценка вероятности ошибочной классификации - доля неправильных меток:
        P(2|1) = 0.165
        P(1|2) = 0.136
        Оценка общей вероятности ошибочной классификации: 0.14889

        Оценка вероятности ошибочной классификации по расстоянию Махаланобиса:
        P(2|1) = 0.16974
        P(1|2) = 0.12188
        Оценка общей вероятности ошибочной классификации: 0.14315

Классификатор, построенный по плохо разделенным данным:
        Оценка вероятности ошибочной классификации - доля неправильных меток:
        P(2|1) = 0.5125
        P(1|2) = 0.248
        Оценка общей вероятности ошибочной классификации: 0.36556

        Оценка вероятности ошибочной классификации по расстоянию Махаланобиса:
        P(2|1) = 0.50725
        P(1|2) = 0.25197
        Оценка общей вероятности ошибочной классификации: 0.36543
```

Рисунок 3. Результаты классификации обучающей выборки из модельных данных (оценки вероятности ошибки)

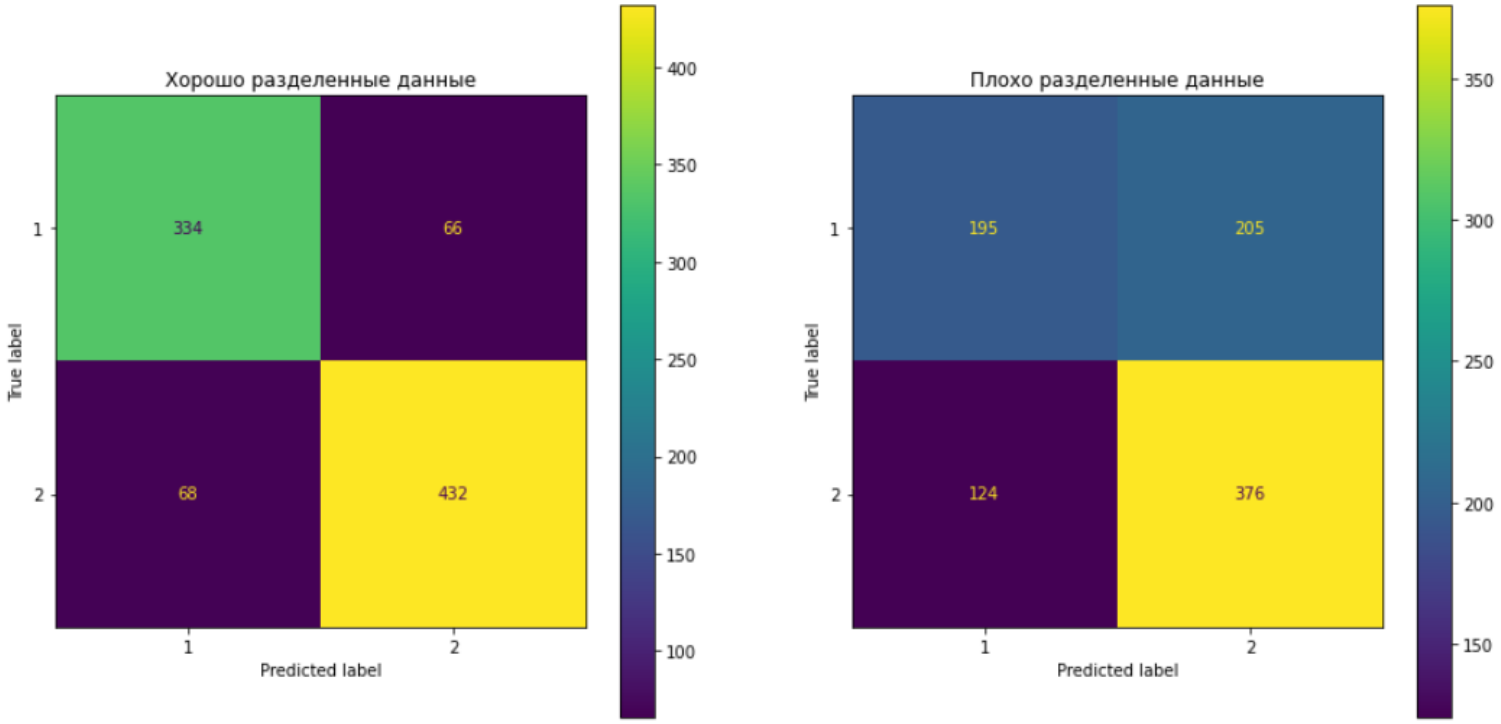


Рисунок 4. Результаты классификации обучающей выборки из модельных данных (таблица сопряженности)

Видим, что оценки, построенные по долям неправильных меток и по расстоянию Махаланобиса, достаточно близки между собой для соответствующих классификаторов. Вероятность ошибки классификатора, построенного на «хорошо» разделенных данных, гораздо ниже.

5) Применение классификатора к тестовой выборке

Классифицируем тестовые выборки, строим матрицы сопряженности и находим эмпирические вероятности ошибки.

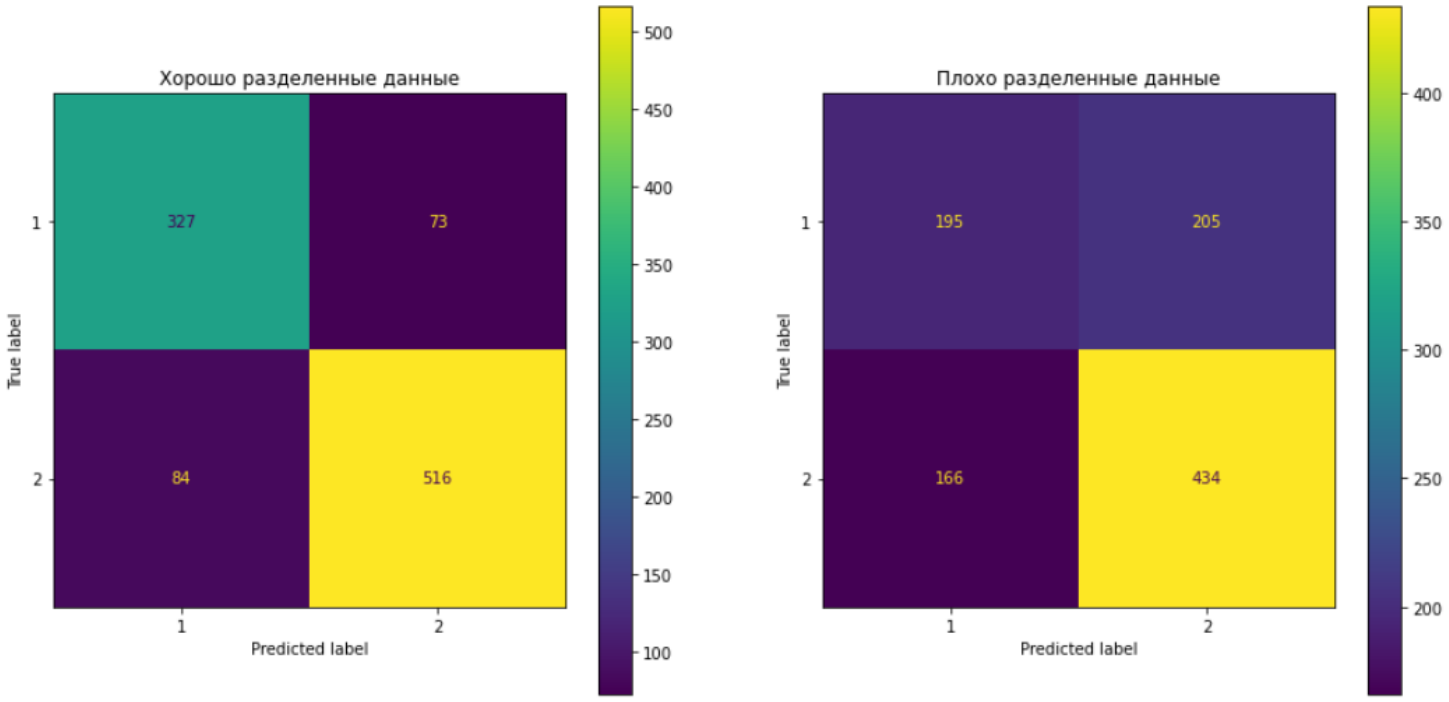


Рисунок 5. Результат классификации тестовых выборок из модельных данных (таблица сопряженности)

```
Эмпирические вероятности ошибочной классификации:
        Классификтор, построенный по хорошо разделенным данным:
        P(2|1) = 0.1825
        P(1|2) = 0.14
        Общая вероятность ошибочной классификации: 0.157

        Классификтор, построенный по плохо разделенным данным:
        P(2|1) = 0.5125
        P(1|2) = 0.27667
        Общая вероятность ошибочной классификации: 0.371
```

Рисунок 6. Результат классификации тестовых выборок из модельных данных (эмпирические вероятности ошибки)

Полученные вероятности близки к оценкам.

6) Применение PCA

Найдем такое преобразование L многомерных признаков, которое позволит понизить их размерность до заданного p', сохранив при этом

максимальную информативность данных. Данные предварительно центрируем.

С помощью преобразования получим новую матрицу данных: $z = Lx$

Считая, в соответствии с лекцией: $\boldsymbol{x} = \left(x_1, \dots, x_p\right)^T$, для применения преобразования к нашей матрице данных сначала необходимо ее транспонировать. Тогда строчками матрицы L являются собственные векторы матрицы ковариаций.

Найдем собственные числа и векторы выборочной оценки ковариационной матрицы. Отсортируем пары по убыванию собственных чисел и оставим только первые p'.

Таким образом, $z_i = l_i\boldsymbol{x},\ i = \overline{1, p'}$. Транспонируем матрицу z, чтобы перейти к прежнему представлению в коде программы.

```
def pca_reduce(sample, size, dim_, target_dim):
    mean = multivariate_mean(sample, size, dim_)
    sample_centered = sample - mean
    cov_matrix = sample_S(sample_centered, size, mean, dim_)

    eigen_val, eigen_vect = np.linalg.eig(cov_matrix)
    eigen = sorted(list(zip(eigen_val, eigen_vect.T)), key=lambda pair: pair[0], reverse=True)
    eigen_vals = np.array([eigen[i][0] for i in range(target_dim)])
    L = np.vstack(list([eigen[i][1] for i in range(target_dim)]))

    sample_pca = np.matmul(L, sample.T)
    sample_pca = sample_pca.T
    return sample_pca
```

Понизим размерность «хорошо» разделенных модельных данных до 2. Построим классификатор и оценим вероятности ошибочной классификации:

```
Оценка вероятности ошибочной классификации - доля неправильных меток:
        P(2|1) = 0.2775
        P(1|2) = 0.172
        Оценка общей вероятности ошибочной классификации: 0.21889

Оценка вероятности ошибочной классификации по расстоянию Махаланобиса:
        P(2|1) = 0.25859
        P(1|2) = 0.17606
        Оценка общей вероятности ошибочной классификации: 0.21274
```

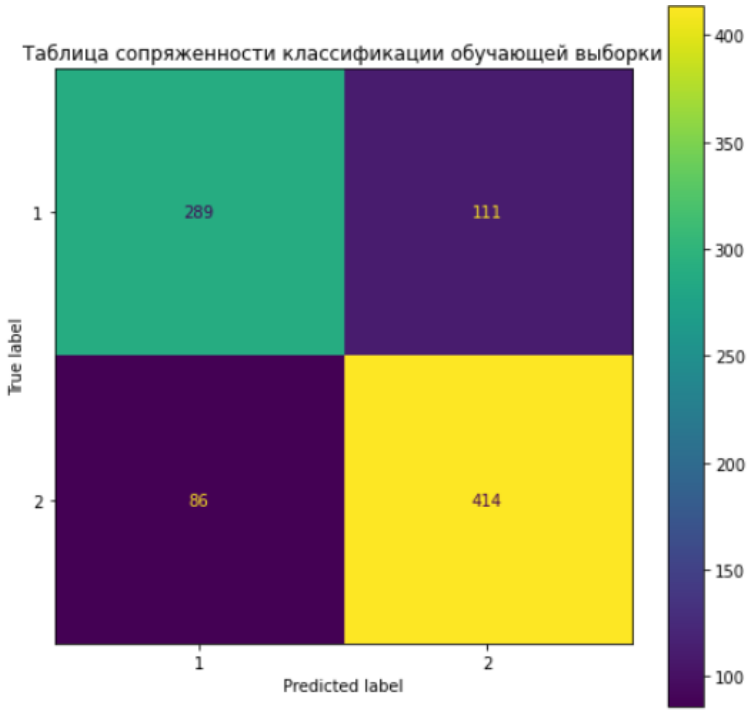
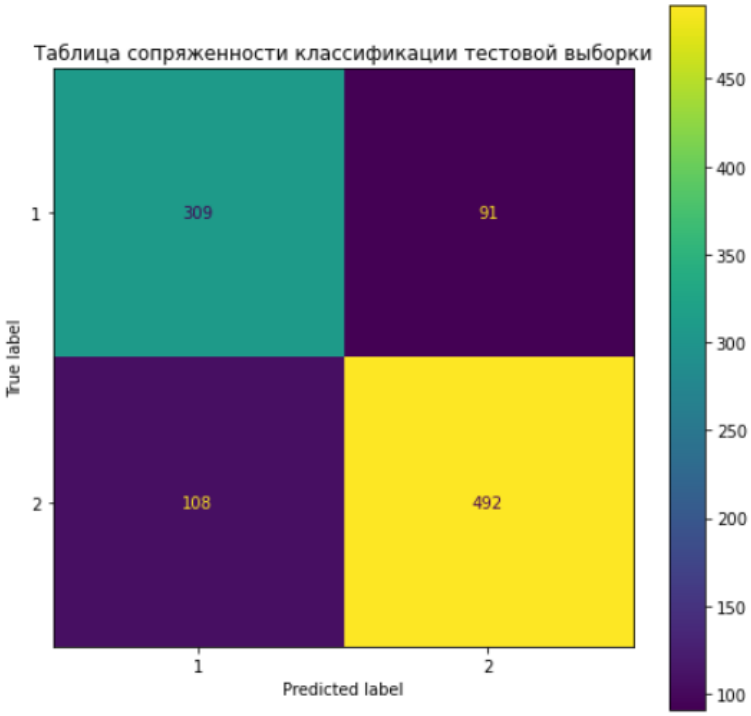


Рисунок 7. PCA: результаты классификации обучающей выборки из модельных данных

Применим к тестовой выборке:




```
Эмпирические вероятности ошибочной классификации:
        P(2|1) = 0.2275
        P(1|2) = 0.18
        Общая вероятность ошибочной классификации: 0.199
```

Рисунок 8. PCA: результаты классификации тестовой выборки из модельных данных.

Видим, что понижение размерности привело к увеличению вероятности ошибки.

b) Работа с данными из репозитория

1) Предобработка данных

Считаем данные из файла. Выберем соотношение между тестовыми и обучающими данными – 4:6. Для понимания, в каком соотношении должно быть число объектов разных классов в обучающей выборке, посмотрим на соотношение классов данных.

```
with open("german.data-numeric") as data:
    rep_objects = np.empty((0, rep_dim + 1))
    rep_marks = np.array([])
    for str in data:
        obj_str = str.split()
        rep_obj = list(map(float, obj_str))
        rep_objects = np.append(rep_objects, [rep_obj], axis=0)

# Посмотрим на соотношение классов
size_1 = len(rep_objects[rep_objects[:, -1] == 1])
size_2 = len(rep_objects[rep_objects[:, -1] == 2])

# Случайным образом разделяем данные на обучающую и тестовую выборку в заданном соотношении
rep_train_sample, rep_test_sample_ = train_test_split(rep_objects, train_size = rep_train_part, random_state = 12666)

# Разделяем обучающие выборки по классам
rep_train_sample1 = np.array(rep_train_sample[rep_train_sample[:, -1] == 1, :-1])
rep_train_sample2 = np.array(rep_train_sample[rep_train_sample[:, -1] == 2, :-1])
rep_train_sample_marks = np.hstack((np.ones(rep_train_sample1.shape[0]), np.full(rep_train_sample2.shape[0], 2)))

rep_n1 = rep_train_sample1.shape[0]
rep_n2 = rep_train_sample2.shape[0]
rep_q_training1 = rep_n1 / (rep_n1 + rep_n2)
rep_q_training2 = rep_n2 / (rep_n1 + rep_n2)

# Выносим в отдельный вектор метки для проверки, убираем лишний столбец из признаков
rep_test_sample_marks = rep_test_sample_[:, -1]
rep_test_sample = rep_test_sample_[:, :-1]
```

```
В первом классе: 700 объектов
Во втором классе: 300 объектов
Размер первой обучающей выборки: 408
Размер второй обучающей выборки: 192
```

Так как второй класс более чем в два раза меньше, необходимо обеспечить попадание в обучающую выборку достаточного числа объектов из этого класса. Случайным образом был подобран случай, когда в обучающую выборку вошло 64% объектов второго класса и чуть менее 60% объектов первого.

2) Построение классификатора, нахождение оценок вероятности ошибки

```
Оценка вероятности ошибочной классификации - доля неправильных меток:
        P(2|1) = 0.1348
        P(1|2) = 0.50521
        Оценка общей вероятности ошибочной классификации: 0.25333

Оценка вероятности ошибочной классификации по расстоянию Махаланобиса:
        P(2|1) = 0.10901
        P(1|2) = 0.53949
        Оценка общей вероятности ошибочной классификации: 0.24676
```

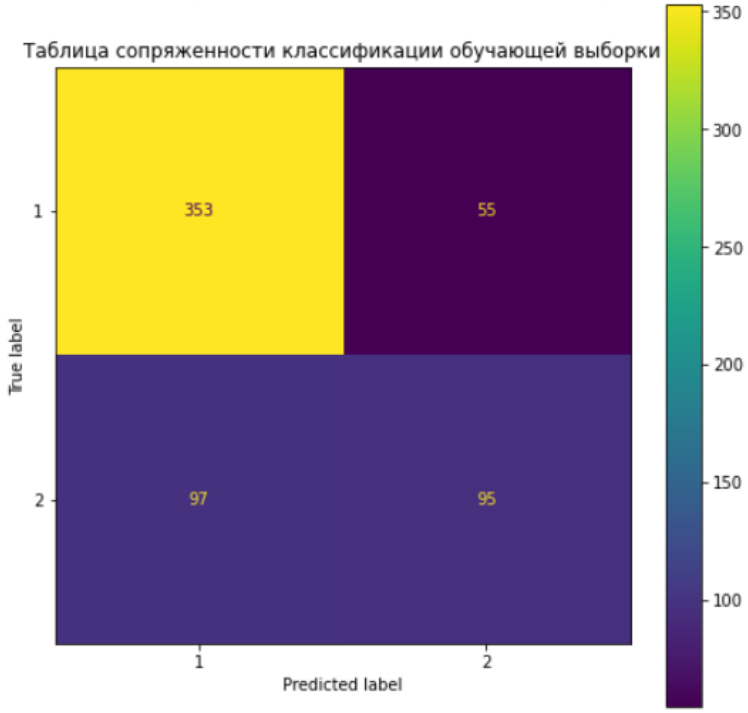
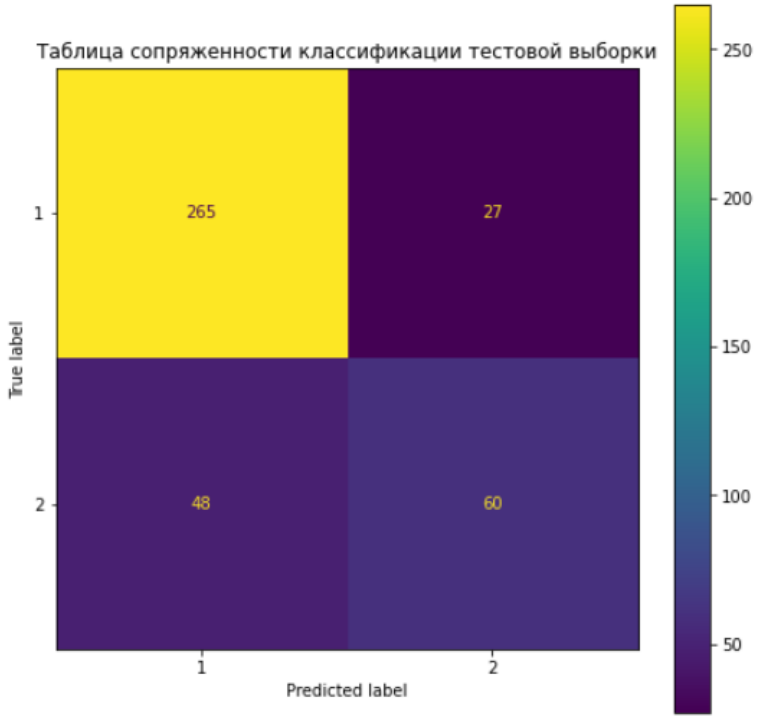


Рисунок 9. Результаты классификации обучающей выборки из данных из репозитория.

Видим, что ошибка отнесения объекта из второго класса к первому велика. Это связано с «перевесом» первого класса по объему. Хоть обучающая выборка из второго класса и включает больше половины её объёма, это не способствует уменьшению вероятности ошибки $\bar{P}(1|2)$.

Применим классификатор к тестовой выборке:



```
Эмпирические вероятности ошибочной классификации:
        P(2|1) = 0.092466
        P(1|2) = 0.44444
        Общая вероятность ошибочной классификации: 0.3651
```

Рисунок 10. Результаты классификации тестовой выборки из данных из репозитория.

На тестовой выборке вероятность ошибки оказалась больше, чем ее оценка.

3) Учет стоимости ошибочной классификации.

В файле, описывающем данные, содержится матрица стоимостей. Учтем рекомендованные значения: $c(2|1) = 1,\ c(1|2) = 5$ в классификации:

```
Оценка вероятности ошибочной классификации - доля неправильных меток:
        P(2|1) = 0.54412
        P(1|2) = 0.083333
        Оценка общей вероятности ошибочной классификации: 0.39667

Оценка вероятности ошибочной классификации по расстоянию Махаланобиса:
        P(2|1) = 0.575
        P(1|2) = 0.093121
        Оценка общей вероятности ошибочной классификации: 0.4208

Эмпирические вероятности ошибочной классификации:
        P(2|1) = 0.52055
        P(1|2) = 0.074074
        Общая вероятность ошибочной классификации: 0.40434
```

Оценки вероятности ошибки и эмпирическая вероятность ошибки при классификации тестовой выборки становятся больше, так как увеличилась вероятность $P(2|1)$. Можно предположить, что при соотношении объемов классов 7:3 значение $c(1|2)$ должно быть равно ≈ 2.33. В таком случае получим:

```
Оценка вероятности ошибочной классификации - доля неправильных меток:
        P(2|1) = 0.30637
        P(1|2) = 0.23438
        Оценка общей вероятности ошибочной классификации: 0.28333

Оценка вероятности ошибочной классификации по расстоянию Махаланобиса:
        P(2|1) = 0.31383
        P(1|2) = 0.25861
        Оценка общей вероятности ошибочной классификации: 0.29616

Эмпирические вероятности ошибочной классификации:
        P(2|1) = 0.28767
        P(1|2) = 0.16667
        Общая вероятность ошибочной классификации: 0.30895
```

Построенный таким образом классификатор даёт наилучший (из полученных) результатов на данных из репозитория.

4) Применение PCA.

Сократим размерность данных до 10:

```
Оценка вероятности ошибочной классификации:
        P(2|1) = 0.12381
        P(1|2) = 0.52222
        Общая вероятность ошибочной классификации: 0.45222

Оценка вероятности ошибочной классификации по расстоянию Махаланобиса:
        P(2|1) = 0.094707
        P(1|2) = 0.56595
        Общая вероятность ошибочной классификации: 0.46246
```

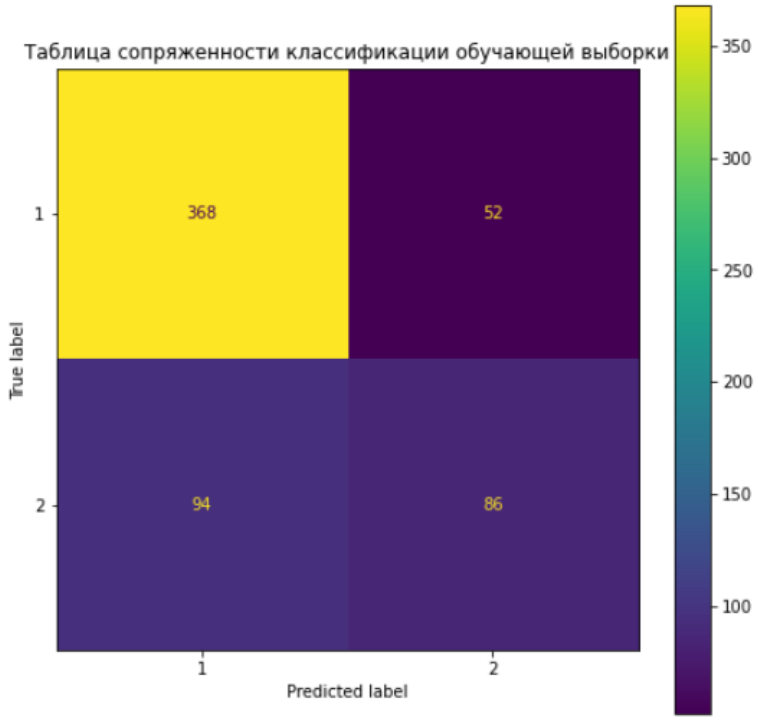
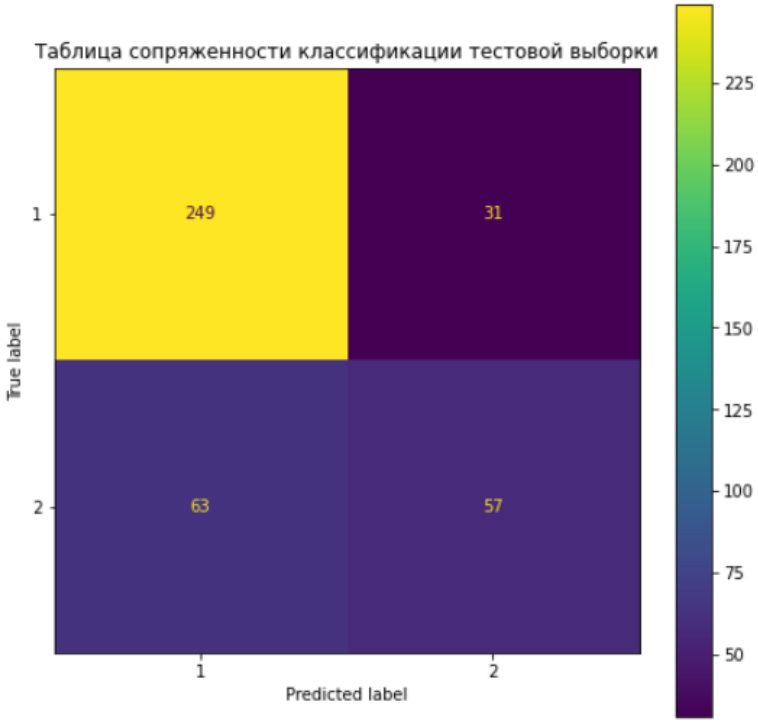


Рисунок 11. PCA: результаты классификации обучающей выборки из данных из репозитория.



```
Эмпирические вероятности ошибочной классификации:
        P(2|1) = 0.10616
        P(1|2) = 0.58333
        Общая вероятность ошибочной классификации: 0.48265
```

Эмпирическая вероятность близка к оценкам и велика.

Выводы

По проделанной работе можно сделать выводы о том, что результат классификации сильно зависит от качества данных.

Как и ожидалось, построенный по «плохо» разделённым данным классификатор имеет бОльшие вероятности ошибки, чем классификатор по «хорошо» разделенным. Применение метода главных компонент хоть и приводит к сокращению числа вычислений, что, безусловно, важно при данных большой размерности, но в то же время качество классификации ухудшается:

На модельных данных:

```
Доля сохранившейся дисперсии (для тестовой выборки): 0.766104
```

На данных из репозитория:

```
Доля сохранившейся дисперсии (для тестовой выборки): 0.997946
```

Размерность модельных данных и так можно считать маленькой – 3. Поэтому её понижение приводит только к потере информативности. В свою очередь, понижение размерности данных из репозитория на их информативность влияет слабо. В случае данных из репозитория главной проблемой для классификатора становится перевес одного из классов по объёму.

Таким образом, качество классификатора во многом зависит от работы исследователя: от предварительного изучения данных с целью назначения стоимостей для уравновешивания «перекоса» по объему, а также от выбора p'.