МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ

ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ АВТОНОМНОЕ
ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО
ОБРАЗОВАНИЯ
«Санкт-Петербургский национальный исследовательский
университет информационных технологий, механики и оптики»

ФАКУЛЬТЕТ ПРОГРАММНОЙ ИНЖЕНЕРИИ И
КОМПЬЮТЕРНОЙ ТЕХНИКИ

Отчёт по лабораторной работе №4

по дисциплине

«Системы искусственного интеллекта»

Выполнил ***:*** Студент группы
P3334 Баянов Равиль
Динарович
Преподаватель:
Авдюшина А. Е.

Санкт-Петербург, 2024

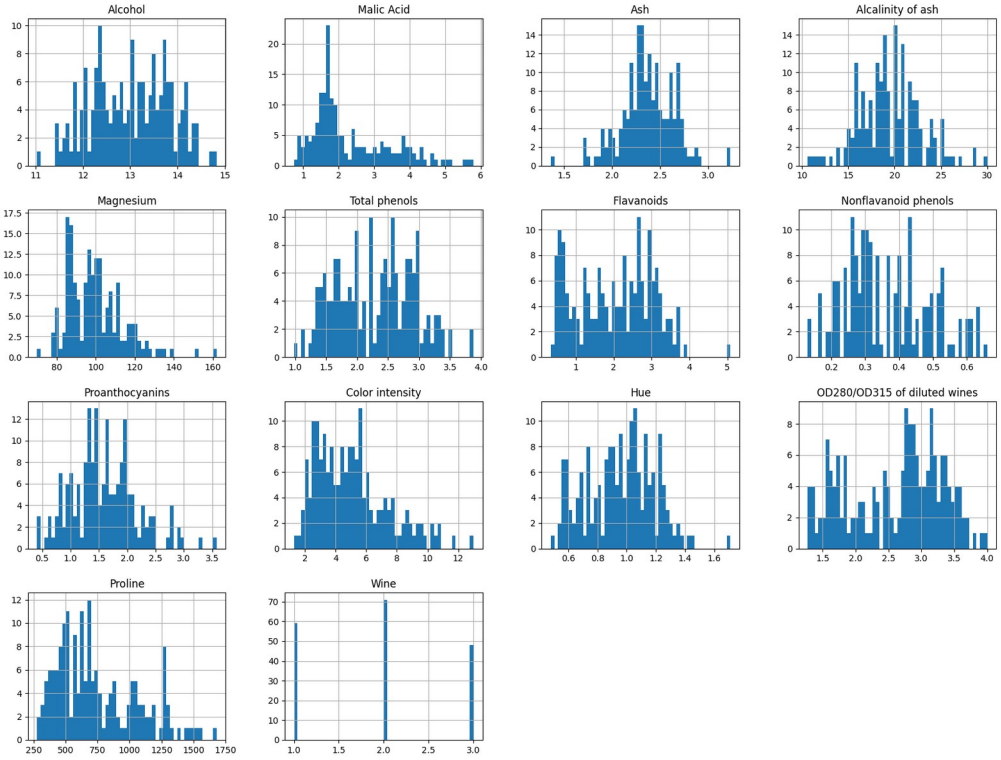
Оглавление

Задание

Вариант: Четный номер в группе - Датасет о вине

- Проведите предварительную обработку данных, включая обработку отсутствующих значений, кодирование категориальных признаков и масштабирование.
- Получите и визуализируйте (графически) статистику по датасету (включая количество, среднее значение, стандартное отклонение, минимум, максимум и различные квантили), постройте 3d-визуализацию признаков.
- Реализуйте метод k-ближайших соседей без использования сторонних библиотек, кроме NumPy и Pandas.
- Постройте две модели k-NN с различными наборами признаков:
    - Модель 1: Признаки случайно отбираются.
    - Модель 2: Фиксированный набор признаков, который выбирается заранее.
- Для каждой модели проведите оценку на тестовом наборе данных при разных значениях k. Выберите несколько различных значений k, например, k=3, k=5, k=10, и т. д. Постройте матрицу ошибок.

Статистика по датасету



	Alcohol	Malic Acid	Ash	Alcalinity of ash	Magnesium
count	178.000000	178.000000	178.000000	178.000000	178.000000
mean	13.000618	2.336348	2.366517	19.494944	99.741573
std	0.811827	1.117146	0.274344	3.339564	14.282484
min	11.030000	0.740000	1.360000	10.600000	70.000000
25%	12.362500	1.602500	2.210000	17.200000	88.000000
50%	13.050000	1.865000	2.360000	19.500000	98.000000
75%	13.677500	3.082500	2.557500	21.500000	107.000000
max	14.830000	5.800000	3.230000	30.000000	162.000000

	Total phenols	Flavanoids	Nonflavanoid phenols	Proanthocyanins
count	178.000000	178.000000	178.000000	178.000000
mean	2.295112	2.029270	0.361854	1.590899
std	0.625851	0.998859	0.124453	0.572359
min	0.980000	0.340000	0.130000	0.410000
25%	1.742500	1.205000	0.270000	1.250000
50%	2.355000	2.135000	0.340000	1.555000
75%	2.800000	2.875000	0.437500	1.950000
max	3.880000	5.080000	0.660000	3.580000

	Color intensity	Hue	OD280/OD315 of diluted wines	Proline
count	178.000000	178.000000	178.000000	178.000000
mean	5.058090	0.957449	2.611685	746.893258
std	2.318286	0.228572	0.709990	314.907474
min	1.280000	0.480000	1.270000	278.000000
25%	3.220000	0.782500	1.937500	500.500000
50%	4.690000	0.965000	2.780000	673.500000
75%	6.200000	1.120000	3.170000	985.000000
max	13.000000	1.710000	4.000000	1680.000000

	Wine
count	178.000000
mean	1.938202
std	0.775035
min	1.000000
25%	1.000000
50%	2.000000
75%	3.000000
max	3.000000

Где:

- count – кол-во значений
- mean – среднее
- std – стандартное отклонение (среднеквадратическое отклонение от мат. ожидания)
- min – минимум

- max – максимум
- Значения процентов - квантили столбцов

Графики, описывающие датасет:

Код на Python реализации модели

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler

# Загрузка данных
df = pd.read_csv('data\WineDataset.csv')
pd.set_option('display.max_columns', None)

# Проверка данных и вычисление статистики
print(df.describe())

# Визуализация статистики
df.hist(bins=50, figsize=(20, 15))
plt.show()

# Обработка отсутствующих значений
df = df.dropna()

# Масштабирование данных (нормализация)
scaler = StandardScaler()
features = df.columns[:-1]  # Все столбцы, кроме Wine
df[features] = scaler.fit_transform(df[features])

# Разделение на обучающий и тестовый наборы
X = df[features].values
y = df['Wine'].values
X_train, X_test, y_train, y_test = train_test_split(X, y,
test_size=0.2, random_state=42)


# Функция для вычисления расстояния между двумя точками
def euclidean_distance(point1, point2):
    return np.sqrt(np.sum((point1 - point2) ** 2))


# Реализация k-NN
def k_nearest_neighbors(X_train, y_train, X_test, k):
    predictions = []
    for X_test_instance in X_test:
        distances = []
        for i, train_instance in enumerate(X_train):
            distance = euclidean_distance(X_test_instance,
train_instance)
            distances.append((distance, y_train[i]))

        # Сортировка по расстоянию и выбор k ближайших
        distances.sort(key=lambda x: x[0])
        neighbors = distances[:k]

        # Получение метки по большинству голосов
        classes = [neighbor[1] for neighbor in neighbors]
        majority_vote = max(set(classes), key=classes.count)
        predictions.append(majority_vote)
    return predictions


# Оценка модели на тестовом наборе
```

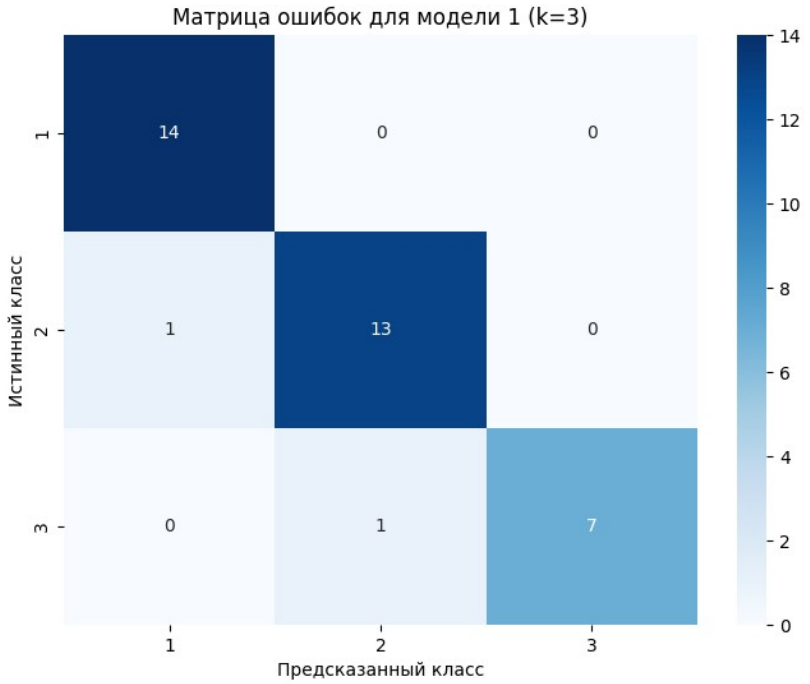
```
def evaluate_knn(X_train, y_train, X_test, y_test, k):
    y_pred = k_nearest_neighbors(X_train, y_train, X_test, k)
    accuracy = np.mean(y_pred == y_test)
    return accuracy, y_pred


# Функция для случайного выбора признаков
def select_random_features(X_train, X_test, num_features):
    feature_indices = np.random.choice(X_train.shape[1],
num_features, replace=False)
    return X_train[:, feature_indices], X_test[:, feature_indices]


# Модель 1: Случайные признаки
num_random_features = 3  # Число случайных признаков
X_train_1, X_test_1 = select_random_features(X_train, X_test,
num_random_features)

# Модель 2: Фиксированный набор признаков (Алкоголь, Малиновая
кислота, Пролин)
X_train_2 = X_train[:, [0, 1, 12]]
X_test_2 = X_test[:, [0, 1, 12]]


def confusion_matrix(y_true, y_pred):
    unique_labels = np.unique(y_true)
    matrix = np.zeros((len(unique_labels), len(unique_labels)),
dtype=int)
    for true, pred in zip(y_true, y_pred):
        matrix[int(true) - 1, int(pred) - 1] += 1
    return matrix


def plot_confusion_matrix(y_true, y_pred, model_num, k):
    conf_matrix = confusion_matrix(y_true, y_pred)
    print(f'Confusion Matrix для модели {model_num} (k={k}):')
    print(conf_matrix)

    # Визуализация матрицы ошибок
    plt.figure(figsize=(8, 6))
    sns.heatmap(conf_matrix, annot=True, fmt='d', cmap='Blues',
                xticklabels=np.unique(y), yticklabels=np.unique(y))
    plt.title(f'Матрица ошибок для модели {model_num} (k={k})')
    plt.xlabel('Предсказанный класс')
    plt.ylabel('Истинный класс')
    plt.show()


# Оценка модели 1 и визуализация матрицы ошибок для разных значений
k
for k in [3, 5, 10]:
    accuracy_1, y_pred_1 = evaluate_knn(X_train_1, y_train,
X_test_1, y_test, k)
    print(f'Точность для модели 1 при k={k}: {accuracy_1}')
    plot_confusion_matrix(y_test, y_pred_1, model_num=1, k=k)

# Оценка модели 2 и визуализация матрицы ошибок для разных значений
k
for k in [3, 5, 10]:
    accuracy_2, y_pred_2 = evaluate_knn(X_train_2, y_train,
X_test_2, y_test, k)
    print(f'Точность для модели 2 при k={k}: {accuracy_2}')
    plot_confusion_matrix(y_test, y_pred_2, model_num=2, k=k)

# 3D-визуализация нескольких признаков
fig = plt.figure(figsize=(10, 7))
```

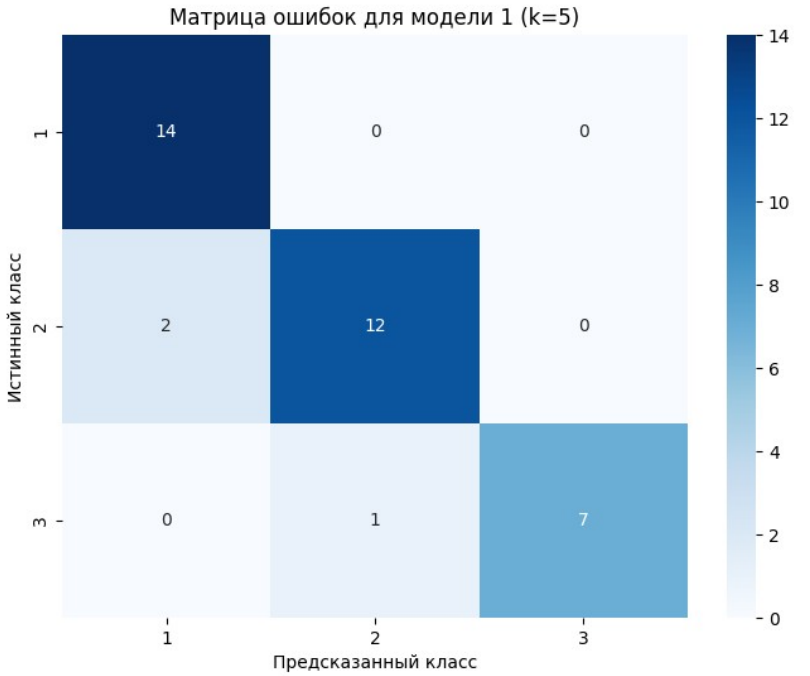
```
ax = fig.add_subplot(111, projection='3d')
ax.scatter(df['Alcohol'], df['Malic Acid'], df['Color intensity'],
c=df['Wine'])
ax.set_xlabel('Alcohol')
ax.set_ylabel('Malic Acid')
ax.set_zlabel('Color intensity')
plt.show()
```

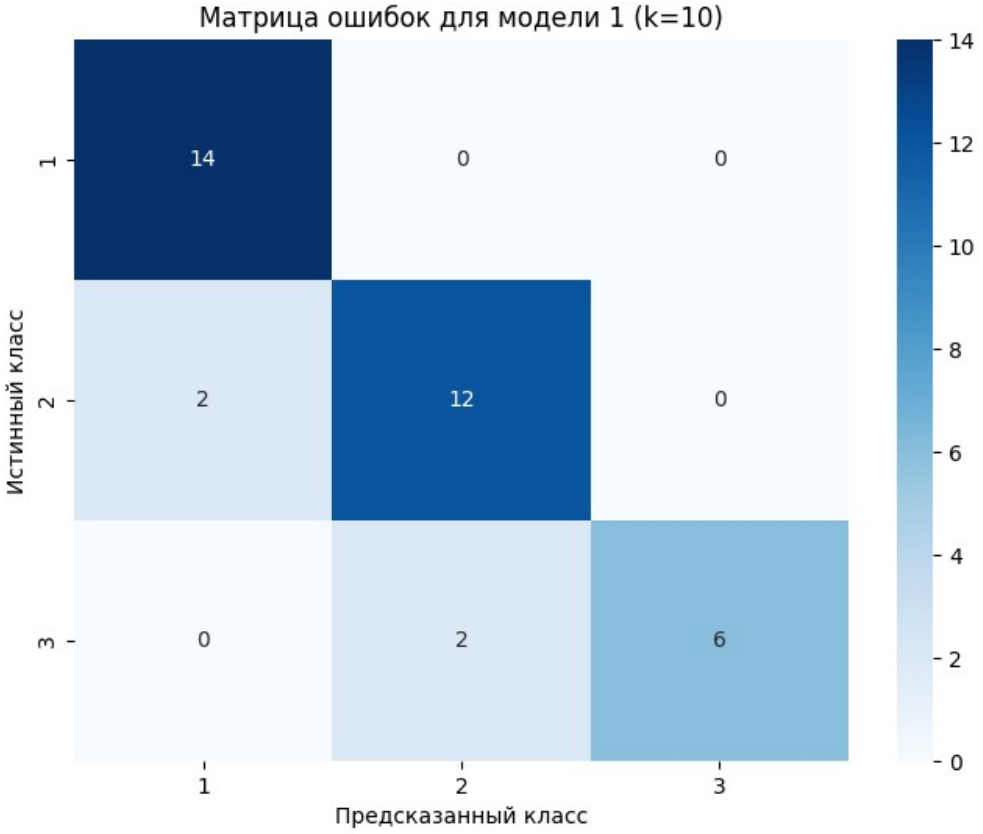
Показатели точности модели

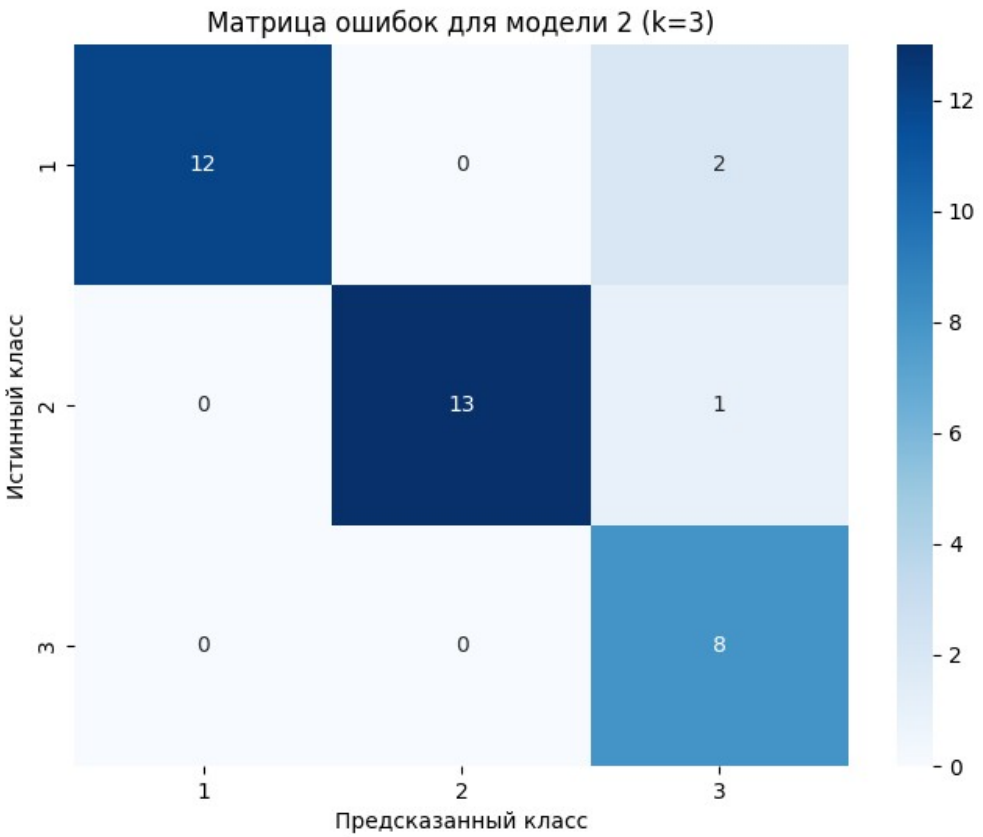
```
Точность для модели 1 при k=3: 0.7777777777777778
Confusion Matrix для модели 1 (k=3):
[[10  2  2]
 [ 3 11  0]
 [ 0  1  7]]
Точность для модели 1 при k=5: 0.75
Confusion Matrix для модели 1 (k=5):
[[10  2  2]
 [ 3 11  0]
 [ 0  2  6]]
Точность для модели 1 при k=10: 0.8055555555555556
Confusion Matrix для модели 1 (k=10):
[[11  2  1]
 [ 3 11  0]
 [ 0  1  7]]
Точность для модели 2 при k=3: 0.9166666666666666
Confusion Matrix для модели 2 (k=3):
[[12  0  2]
 [ 0 13  1]
 [ 0  0  8]]
Точность для модели 2 при k=5: 0.9444444444444444
Confusion Matrix для модели 2 (k=5):
[[12  0  2]
 [ 0 14  0]
 [ 0  0  8]]
Точность для модели 2 при k=10: 0.8888888888888888
Confusion Matrix для модели 2 (k=10):
[[12  0  2]
 [ 0 13  1]
 [ 1  0  7]]
```

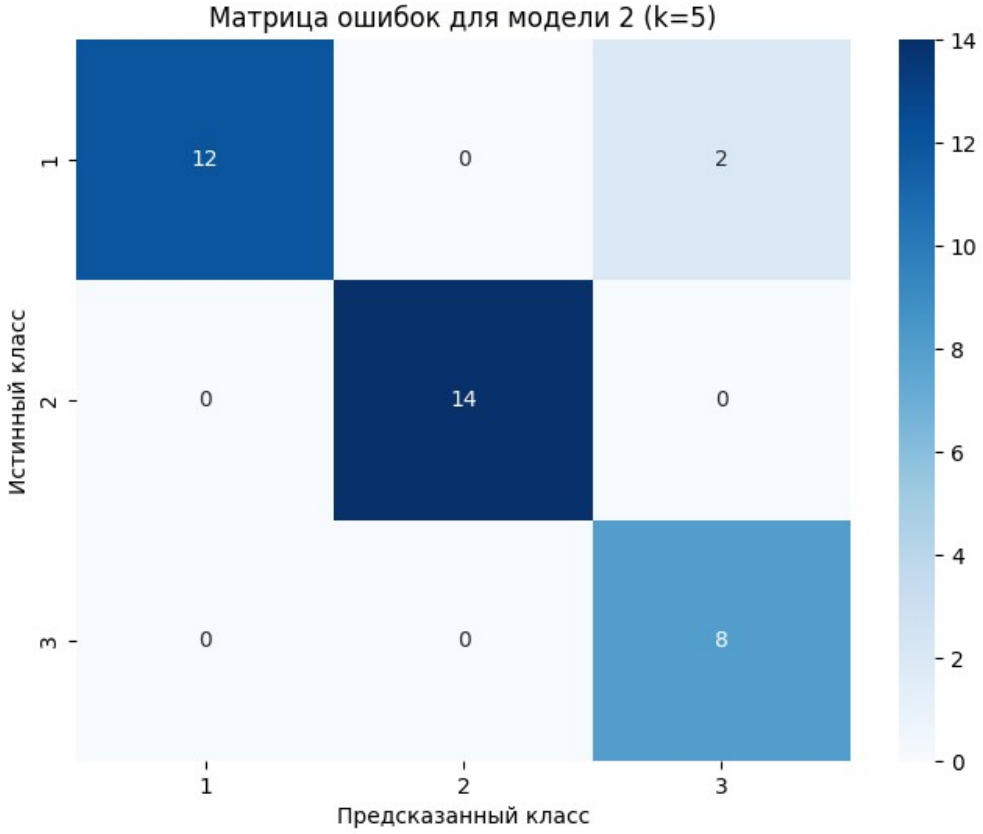
Визуализация матриц ошибок и визуализация нескольких признаков 3D

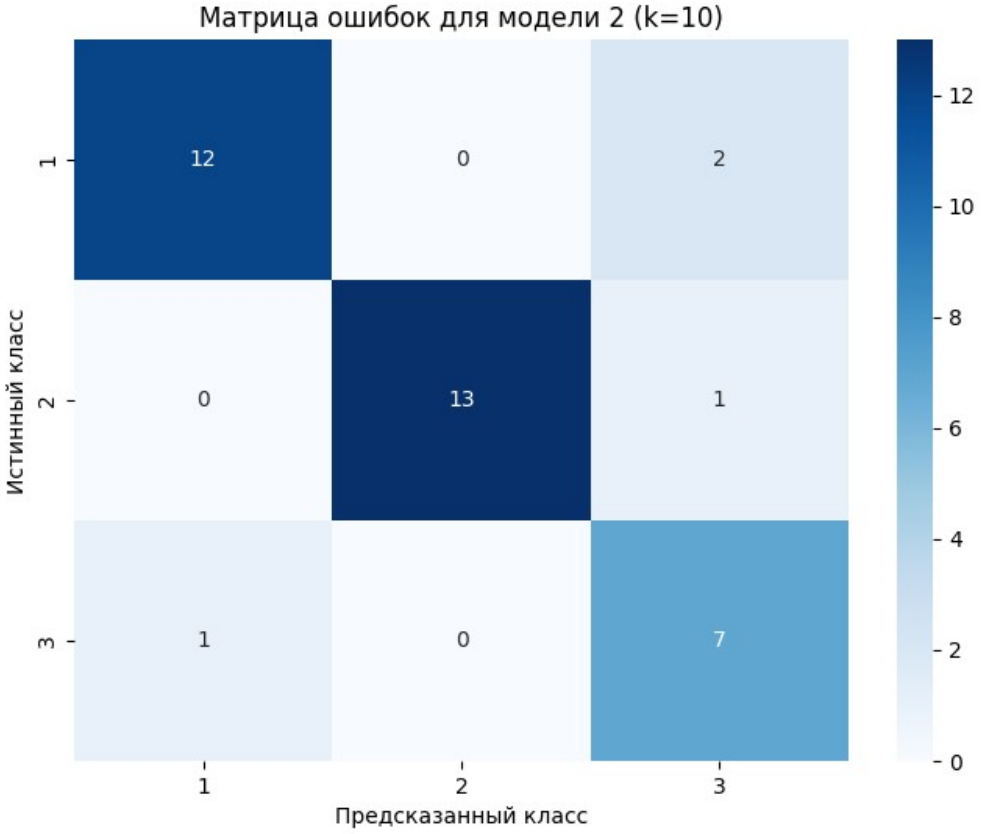


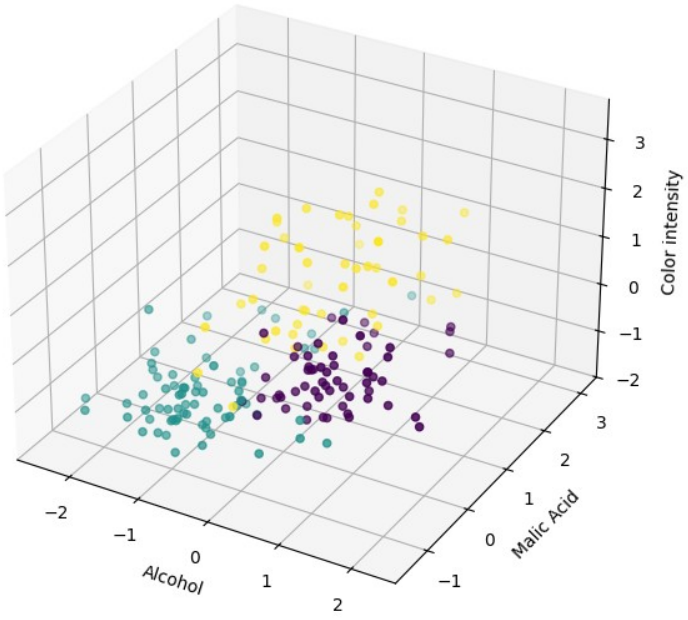



Матрица ошибок для модели 1 (k=10)
Истинный класс
Предсказанный класс
1
2
3
14
0
0
2
12
0
0
2
6
0
2
4
6
8
10
12
14


Матрица ошибок для модели 2 (k=3)
Истинный класс
Предсказанный класс
1
2
3
12
0
2
0
13
1
0
0
8
0
2
4
6
8
10
12


Матрица ошибок для модели 2 (k=5)
Истинный класс
Предсказанный класс
1
2
3
12
0
2
0
14
0
0
0
8
14
12
10
8
6
4
2
0


Матрица ошибок для модели 2 (k=10)
Истинный класс
Предсказанный класс
1
2
3
12
0
2
0
13
1
1
0
7
12
10
8
6
4
2
0


Alcohol
Malic Acid
Color intensity

Вывод

Выполнив данную лабораторную работу, мы научились пользоваться методом машинного обучения – k-ближайших соседей. Здесь мы видим, что в целом этот метод крайне просто реализуется и невероятно точен в своих предсказаниях. Матрицы ошибок показывают, что большинство случаев (в зависимости от параметров и значения k) удовлетворяют нашим целям. Заметим, также что крайне важно правильно подобрать значение k, так как по нашим данным можно заметить, что при чрезмерном увеличении k модель уже начинается ошибаться (например k = 10).