Факультет программной инженерии и

компьютерной техники

Системы искусственного интеллекта

Лабораторная работа №5

Метод к-ближайших соседей

Выполнила Громилова Мария Дмитриевна,

группа P33311

Преподаватель Кугаевских Александр Владимирович

г. Санкт-Петербург 2023 г

Задание

- Проведите предварительную обработку данных, включая обработку отсутствующих значений, кодирование категориальных признаков и масштабирование.
- 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.

Описание метода

В случае использования метода для классификации объект присваивается тому классу, который является наиболее распространённым среди **k** соседей данного элемента, классы которых уже известны.
В случае использования метода для регрессии, объекту присваивается среднее значение по **k** ближайшим к нему объектам, значения которых уже известны.

Решение

Предварительная обработка данных. Столбец Wine – целевая переменная

```
# Выделение числовых признаков для масштабирования
numerical_features = df.drop( labels: 'Wine', axis=1)

# Масштабирование числовых признаков
numerical_features = (numerical_features - numerical_features.mean()) / numerical_features.std()
df_scaled = pd.concat( objs: [wine_column, numerical_features], axis=1)

# Обработка отсутствующих значений
df.fillna(df.mean(), inplace=True)

# Выделение числовых признаков и целевой переменной
X = df.drop( labels: 'Wine', axis=1).values
y = df['Wine'].values
```

Функция определяющая Евклидово расстояние между двумя элементами

```
def euclidean_distance(x1, x2):
    return np.sqrt(np.sum((x1[selected_features] - x2[selected_features]) ** 2))
```

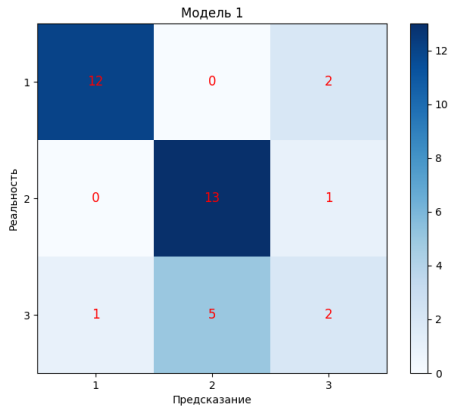
Метод k-ближайших соседей – возвращает вид вина

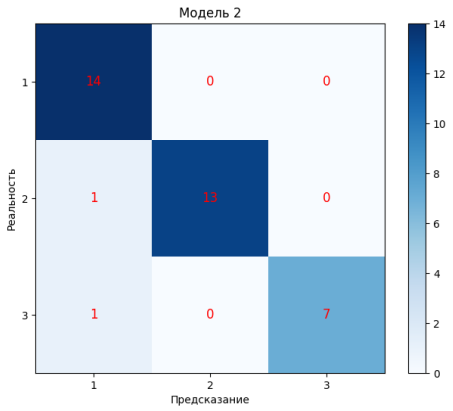
```
def k_nearest_neighbors(X_train, y_train, x_test, k, selected_features):
    def euclidean_distance(x1, x2):
        return np.sqrt(np.sum((x1[selected_features] - x2[selected_features]) ** 2))

    distances = [euclidean_distance(x_test, x_train) for x_train in X_train]
    k_indices = np.argsort(distances)[:k]
    k_nearest_labels = [y_train[i] for i in k_indices]
    result_class = np.bincount(k_nearest_labels).argmax()
    return result_class


# Создание и обучение Модели 1
random_features_model_1 = np.random.choice(X_train.shape[1], size=3, replace=False)
y_pred_model_1 = []
for x_test in X_test:
    result_class = k_nearest_neighbors(X_train, y_train, x_test, k=3, selected_features=random_features_model_1)
    y_pred_model_1.append(result_class)

# Создание и обучение Модели 2
y_pred_model_2 = []
fixed_features_model_2 = [0, 1, 2]
for x_test in X_test:
    result_class = k_nearest_neighbors(X_train, y_train, x_test, k=3, selected_features=fixed_features_model_2)
    y_pred_model_2.append(result_class)
```

Вывод матрицы ошибок

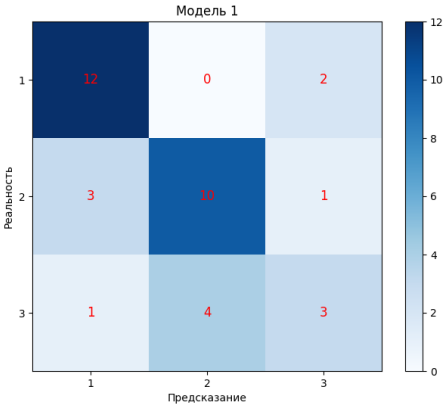
```
def plot_confusion_matrix(y_true, y_pred, model_name):
    cm = confusion_matrix(y_true, y_pred)
    plt.figure(figsize=(8, 6))
    plt.imshow(cm, interpolation='nearest', cmap=plt.cm.Blues)
    plt.title(model_name)
    plt.colorbar()
    tick_marks = np.arange(len(np.unique(y_true)))
    plt.xticks(tick_marks, np.unique(y_true))
    plt.yticks(tick_marks, np.unique(y_true))
    plt.xlabel('Предсказание')
    plt.ylabel('Реальность')
    plt.show()

plot_confusion_matrix(y_test, y_pred_model_1, model_name: 'Модель 1')
plot_confusion_matrix(y_test, y_pred_model_2, model_name: 'Модель 2')
```

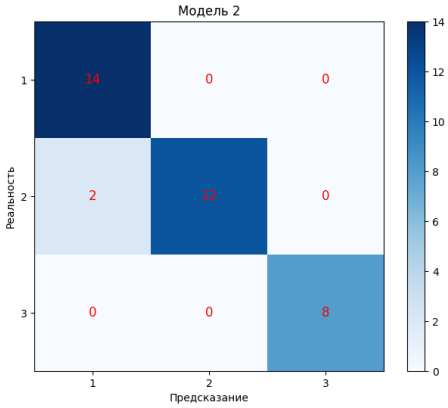
K = 3



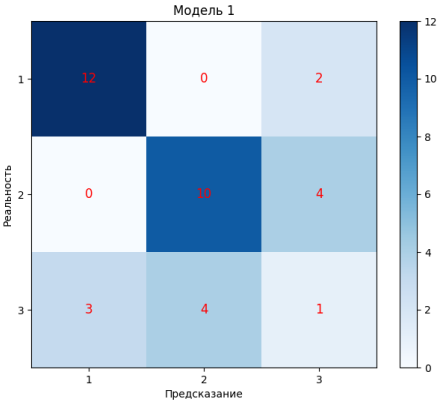


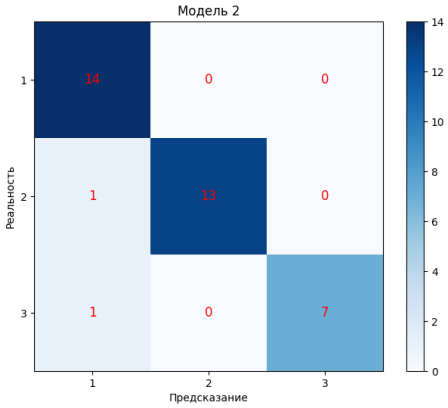
K = 5


Модель 1
Реальность
Предсказание
12 0 2
3 10 1
1 4 3

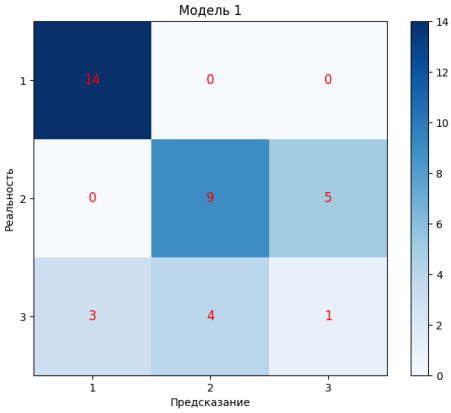

Модель 2
Реальность
Предсказание
14 0 0
2 12 0
0 0 8

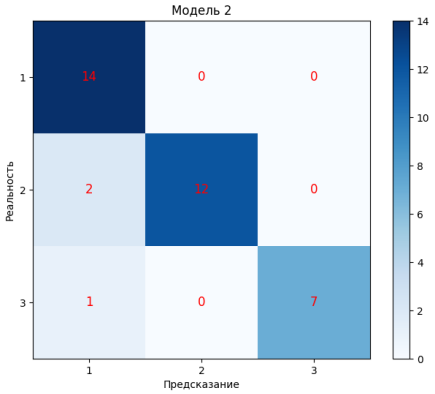
K = 7


Модель 1
Реальность
Предсказание
12 0 2
0 10 4
3 4 1

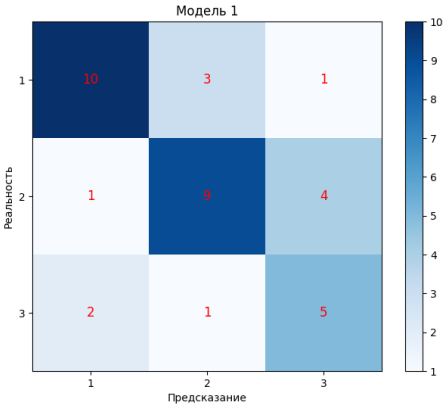

Модель 2
Реальность
Предсказание
14 0 0
1 13 0
1 0 7

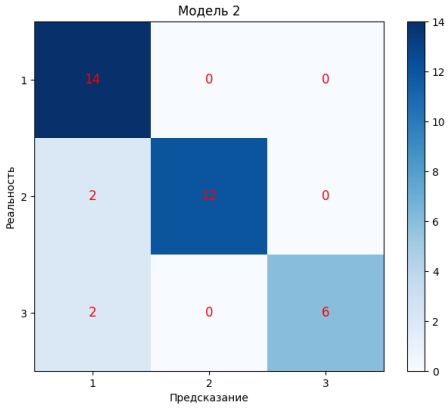
K=10


Модель 1
Реальность
Предсказание
14 0 0
0 9 5
3 4 1

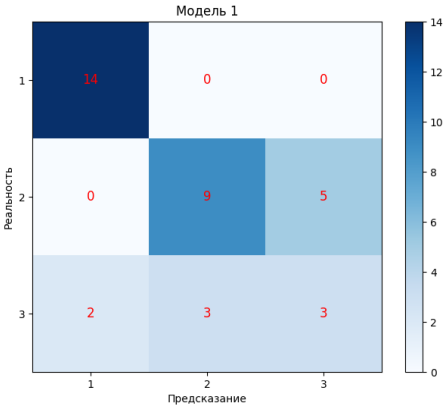

Модель 2
Реальность
Предсказание
14 0 0
2 12 0
1 0 7

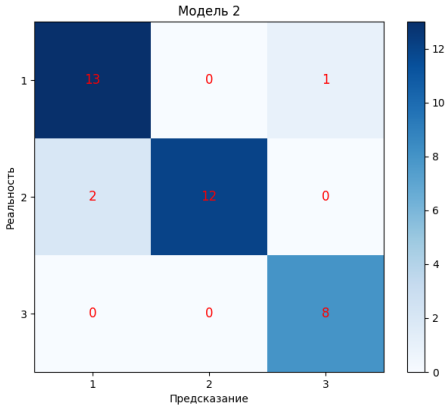
K=13


Модель 1
Реальность
Предсказание
10 3 1
1 9 4
2 1 5

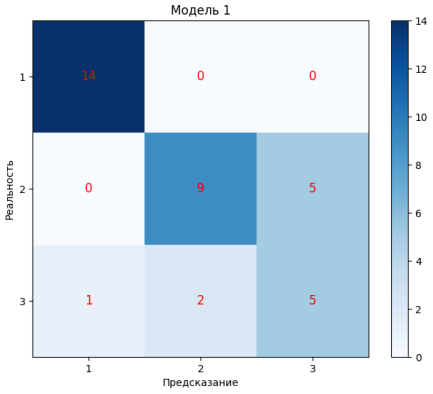

Модель 2
Реальность
Предсказание
14 0 0
2 12 0
2 0 6

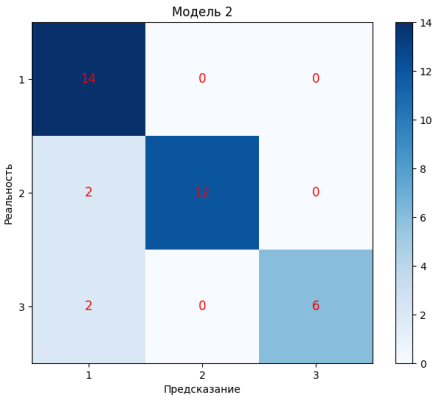
K=17


Модель 1
Реальность
Предсказание
14 0 0
0 9 5
2 3 3

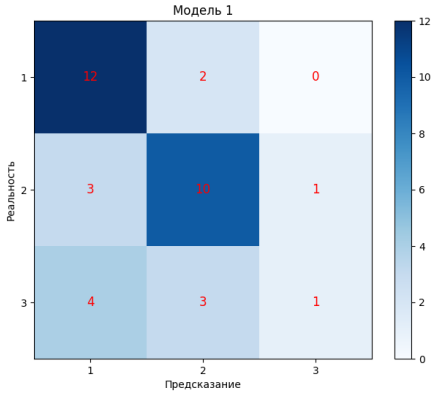

Модель 2
Реальность
Предсказание
13 0 1
2 12 0
0 0 8

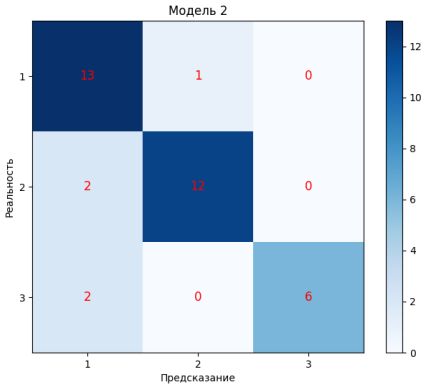
K = 21


Модель 1
Реальность
Предсказание
14 0 0
0 9 5
1 2 5


Модель 2
Реальность
Предсказание
14 0 0
2 12 0
2 0 6

K =30





Вывод:

При увеличении количества соседей увеличивается и погрешность. Я считаю, что для этих данных оптимальным значением K будет от 3 до 13, в других случаях ошибка выше.