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** 3](#_Toc179509344)

[**Статистика по датасету** 4](#_Toc179509345)

[**Код на Python реализации модели** 7](#_Toc179509346)

[**Показатели точности модели** 10](#_Toc179509347)

[**Визуализация матриц ошибок и визуализация нескольких признаков 3D** 11](#_Toc179509348)

[**Вывод** 15](#_Toc179509349)

**Задание**

Вариант: Четный номер в группе - Датасет [о вине](https://www.kaggle.com/datasets/davorbudimir/winedataset)

* 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**

Изображение выглядит как зарисовка, рисунок, диаграмма, дизайн

Автоматически созданное описание

Изображение выглядит как текст, снимок экрана, Шрифт

Автоматически созданное описание

Изображение выглядит как текст, снимок экрана, Шрифт

Автоматически созданное описаниеИзображение выглядит как текст, снимок экрана, Шрифт

Автоматически созданное описание

Изображение выглядит как текст, снимок экрана, Шрифт

Автоматически созданное описание

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**

Изображение выглядит как текст, снимок экрана, Шрифт

Автоматически созданное описание

**Визуализация матриц ошибок и визуализация нескольких признаков 3D**

Изображение выглядит как текст, снимок экрана, диаграмма, Прямоугольник

Автоматически созданное описание

Изображение выглядит как текст, снимок экрана, Прямоугольник, диаграмма

Автоматически созданное описание

Изображение выглядит как текст, снимок экрана, Прямоугольник, диаграмма

Автоматически созданное описаниеИзображение выглядит как текст, снимок экрана, Прямоугольник, диаграмма

Автоматически созданное описаниеИзображение выглядит как текст, снимок экрана, Прямоугольник, диаграмма

Автоматически созданное описаниеИзображение выглядит как текст, снимок экрана, Прямоугольник, диаграмма

Автоматически созданное описаниеИзображение выглядит как зарисовка, диаграмма, дизайн

Автоматически созданное описание

**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).