Министерство науки и высшего образования Российской Федерации
Федеральное государственное автономное образовательное учреждение
высшего образования
«СЕВЕРО-КАВКАЗСКИЙ ФЕДЕРАЛЬНЫЙ УНИВЕРСИТЕТ»

Институт перспективной инженерии
Департамент цифровых, робототехнических систем и электроники

ОТЧЕТ
ПО ЛАБОРАТОРНОЙ РАБОТЕ №6
дисциплины
«Искусственный интеллект и машинное обучение»

Выполнил:
Митряшкина Дарина Сергеевна
2 курс, группа ИТС-б-о-23-1,
11.03.02 «Инфокоммуникационные технологии и системы связи»,
направленность (профиль) «Инфокоммуникационные системы и сети», очная форма обучения

_____
(подпись)

Проверил:
Доцент департамента цифровых, робототехнических систем и электроники Воронкин Р.А.

_____
(подпись)

Отчет защищен с оценкой _____ Дата защиты_____

Ставрополь, 2025 г.

Тема: Основные этапы исследовательского анализа данных

Цель: научиться применять методы обработки данных в pandas. DataFrame, необходимые для разведочного анализа данных (EDA), включая работу с пропусками, выбросами, масштабирование и кодирование категориальных признаков.

Порядок выполнения работы:

Лабораторная работа №6

```
[1]
    # Установка библиотеки missingno
    try:
        import missingno as msno
    except ImportError:
        !pip install missingno
        import missingno as msno

    import pandas as pd
    import numpy as np
    import seaborn as sns
    import matplotlib.pyplot as plt

    from sklearn.preprocessing import StandardScaler, MinMaxScaler, RobustScaler

    %matplotlib inline
```

1. Выполнение практической работы «Обнаружение и обработка пропущенных значений»

1. Обнаружение и обработка пропущенных значений

```
[2]
    # Создание DataFrame с пропусками
    data = {
        "Имя": ["Анна", "Иван", "Ольга", "Петр", "Мария", "Дмитрий", "Елена", "Сергей", "Алина", "Артем"],
        "Возраст": [25, np.nan, 22, 40, 35, np.nan, 28, 31, np.nan, 29],
        "Город": ["Москва", "СПб", np.nan, "Новосибирск", "СПб", "Екатеринбург", np.nan, "Казань", "Томск", np.nan],
        "Доход": [50000, 60000, np.nan, 70000, 65000, 55000, 48000, np.nan, 52000, 58000],
        "Образование": [np.nan, "Высшее", "Среднее", "Высшее", np.nan, "Высшее", "Среднее", "Высшее", np.nan, "Среднее"],
        "Стаж работы": [3, 8, 1, 15, 10, np.nan, 5, 7, np.nan, 2]
    }
    df = pd.DataFrame(data)
    df
```

	Имя	Возраст	Город	Доход	Образование	Стаж работы
0	Анна	25.0	Москва	50000.0	NaN	3.0
1	Иван	NaN	СПб	60000.0	Высшее	8.0
2	Ольга	22.0	NaN	NaN	Среднее	1.0
3	Петр	40.0	Новосибирск	70000.0	Высшее	15.0

```python
# Проверка пропусков
print("Количество пропусков в каждом столбце:")
print(df.isna().sum())

# Визуализация
import missingno as msno

msno.matrix(df)
plt.show()

msno.bar(df)
plt.show()

msno.heatmap(df)
plt.show()
```

```
Количество пропусков в каждом столбце:
Имя             0
Возраст         3
Город           3
Доход           2
Образование     3
Стаж работы     2
dtype: int64
```

Рисунок 1. Практическая работа «Обнаружение и обработка пропущенных значений»

2. Выполнение практической работы «Обнаружение и удаление выбросов»

2. Обнаружение и удаление выбросов

```
[ ]
    # Удалим строки с пропущенными значениями в доходе
    df_clean = df.dropna(subset=["Доход"])

    # Построение boxplot
    plt.figure(figsize=(6, 4))
    sns.boxplot(x=df_clean["Доход"])
    plt.title("Boxplot: Доход")
    plt.show()

    # IQR
    Q1 = df_clean["Доход"].quantile(0.25)
    Q3 = df_clean["Доход"].quantile(0.75)
    IQR = Q3 - Q1
    lower = Q1 - 1.5 * IQR
    upper = Q3 + 1.5 * IQR

    # Фильтрация
    df_iqr = df_clean[(df_clean["Доход"] >= lower) & (df_clean["Доход"] <= upper)]
    df_iqr
```

Рисунок 2. Практическая работа «Обнаружение и удаление выбросов»

3. Выполнение практической работы «Масштабирование числовых признаков»

3. Масштабирование числовых признаков

```
]
  df_scaled = df_iqr.dropna(subset=["Возраст", "Доход"]).copy()

  # StandardScaler
  scaler_std = StandardScaler()
  df_scaled[["Возраст_std", "Доход_std"]] = scaler_std.fit_transform(df_scaled[["Возраст", "Доход"]])

  # MinMaxScaler
  scaler_mm = MinMaxScaler()
  df_scaled[["Возраст_mm", "Доход_mm"]] = scaler_mm.fit_transform(df_scaled[["Возраст", "Доход"]])

  # RobustScaler
  scaler_rb = RobustScaler()
  df_scaled[["Возраст_rb", "Доход_rb"]] = scaler_rb.fit_transform(df_scaled[["Возраст", "Доход"]])

  df_scaled
```

Рисунок 3. Практическая работа «Масштабирование числовых признаков»

4. Выполнение практической работы «Кодирование категориальных признаков»

4. Кодирование категориальных признаков

```
]
  df_cat = df.copy()
  df_cat["Образование"] = df_cat["Образование"].fillna("Не указано")

  # Ordinal Encoding
  order = {"Не указано": 0, "Среднее": 1, "Высшее": 2}
  df_cat["Образование_код"] = df_cat["Образование"].map(order)

  # One-hot encoding
  df_encoded = pd.get_dummies(df_cat, columns=["Город"], drop_first=True)
  df_encoded.head()
```

Рисунок 4. Практическая работа «Кодирование категориальных признаков»

5. Выполнение практической работы «Комплексный EDA»

5. Комплексный EDA

```
# Пример DataFrame с числовыми признаками
data = {
    "Возраст": [25, 40, 35, 28, 31, 29],
    "Доход": [50000, 70000, 65000, 48000, 52000, 58000],
    "Стаж": [3, 15, 10, 5, 7, 2]
}
df = pd.DataFrame(data)

# Стандартизация
scaler = StandardScaler()
df_scaled = pd.DataFrame(scaler.fit_transform(df), columns=df.columns)

df_scaled.head()

sns.pairplot(df_scaled[["Возраст", "Доход"]])
plt.suptitle("Парные графики: Возраст и Доход", y=1.02)
plt.show()
```

Рисунок 5. Практическая работа «Комплексный EDA»

6. Выполнение индивидуального задания

6. Индивидуальное задание (вариант 6)

```
dates = pd.date_range("2024-01-01", periods=6, freq="D")
temps = [20.0, np.nan, 21.0, np.nan, np.nan, 24.0]

df_time = pd.DataFrame({"дата": dates, "температура": temps})
df_time.set_index("дата", inplace=True)
df_time["интерполяция"] = df_time["температура"].interpolate()
df_time
```

дата	температура	интерполяция
2024-01-01	20.0	20.0
2024-01-02	NaN	20.5
2024-01-03	21.0	21.0
2024-01-04	NaN	22.0
2024-01-05	NaN	23.0
2024-01-06	24.0	24.0

Рисунок 6. Индивидуальное задание

Ответы на контрольные вопросы:

1. Какие типы проблем могут возникнуть из-за пропущенных значений в данных?

Пропущенные значения вызывают искажение статистики (например, среднего или корреляции), сбой алгоритмов машинного обучения, снижение точности прогнозов и потерю информации, особенно если много NaN.

2. Как с помощью методов pandas определить наличие пропущенных значений?

В pandas метод .isna() показывает, где в DataFrame стоят NaN (True/False), .notna() — где данные есть, а .isna().sum() подсчитывает количество пропусков по столбцам, что позволяет быстро увидеть проблемные места.

3. Что делает метод .dropna() и какие параметры он принимает?

Метод .dropna() удаляет строки или столбцы с NaN; параметр axis выбирает — строки (0) или столбцы (1); how задаёт, удалять ли, если все значения NaN или хотя бы один; thresh задаёт минимальное число непустых значений для сохранения.

4. Чем различаются подходы заполнения пропусков средним, медианной и модой?

Среднее подходит для симметричных данных, медиана — когда есть выбросы (она устойчивее), а мода — для категориальных признаков, где важна самая частая категория.

5. Как работает метод fillna(method='ffill') и в каких случаях он применим?

fillna(method='ffill') заполняет пропуски предыдущим значением сверху вниз; применяют, например, во временных рядах, когда логично продолжить последнее известное значение.

6. Какую задачу решает метод interpolate() и чем он отличается от fillna()?

interpolate() восстанавливает пропуски на основе тренда между соседними точками (линейно, полиномиально и др.), в отличие от fillna(), который просто вставляет фиксированные или соседние значения без учёта закономерностей.

7. Что такое выбросы и почему они могут искажать результаты анализа?

Выбросы — это значения, сильно выбивающиеся из общего ряда данных; они искажают среднее, стандартное отклонение и могут сбить работу моделей, особенно чувствительных к масштабу.

8. В чём суть метода межквартильного размаха (IQR) и как он используется для обнаружения выбросов?

Метод IQR берёт разницу между третьим (Q3) и первым (Q1) квартилями; выбросами считают точки за пределами $Q1 - 1.5 \times IQR$ или

$Q3 + 1.5 \times IQR$ — это помогает выявить слишком большие или маленькие значения вне «нормального» диапазона.

9. Как вычислить границы IQR и применить их в фильтрации?

Сначала находят Q1 (25-й процентиль) и Q3 (75-й процентиль), считают $IQR = Q3 - Q1$; нижняя граница = $Q1 - 1.5 \times IQR$, верхняя = $Q3 + 1.5 \times IQR$; фильтруют данные, оставляя только те, что внутри этих границ.

10. Что делает метод .clip() и как его можно использовать для обработки выбросов?

Метод .clip() обрезает значения ниже или выше заданных порогов, заменяя их на эти пороги; с его помощью можно оставить выбросы в данных, но ограничить их влияние, обрезав экстремальные значения.

11. Зачем может потребоваться логарифмическое преобразование числовых признаков?

Логарифмическое преобразование уменьшает влияние больших значений, сжимает диапазон и делает распределение более нормальным, что помогает моделям и метрикам работать стабильнее.

12. Какие графические методы позволяют обнаружить выбросы (указать не менее двух)?

Выбросы хорошо видно на boxplot (ящик с усами), гистограммах, а также scatterplot, где видно, какие точки выбиваются из общего облака.

13. Почему важно быть осторожным при удалении выбросов из обучающих данных?

Удаляя выбросы, можно потерять редкие, но важные случаи (например, мошенничество или ошибки), а значит — ухудшить обобщающую способность модели.

14. Зачем необходимо масштабирование признаков перед обучением моделей?

Масштабирование приводит признаки к одному диапазону, чтобы ни один не доминировал; это важно для алгоритмов, чувствительных к масштабу (например, SVM, KNN, градиентные методы).

15. Чем отличается стандартизация от нормализации?

Стандартизация делает среднее 0 и стандартное отклонение 1; нормализация сжимает значения в диапазон [0, 1]; их выбирают в зависимости от задачи и модели.

16. Что делает StandardScaler и как рассчитываются преобразованные значения?

StandardScaler стандартизирует данные, делая среднее 0 и стандартное отклонение 1; формула: (x – mean) / std.

17. Как работает MinMaxScaler и когда его использование предпочтительно?

MinMaxScaler сжимает данные в диапазон [0, 1] по формуле (x – min) / (max – min); его лучше использовать, если нужны пропорции и нет сильных выбросов.

18. В чём преимущества RobustScaler при наличии выбросов?

RobustScaler использует медиану и IQR вместо среднего и std, поэтому лучше справляется с выбросами, не давая им искажать масштаб.

19. Как реализовать стандартизацию с помощью .mean() и .std() вручную в pandas?

В pandas: (df[col] – df[col].mean()) / df[col].std(), применяя по каждому столбцу — так получаем стандартизированные значения.

20. Какие типы моделей наиболее чувствительны к масштабу признаков?

Масштабу особенно чувствительны модели, основанные на расстояниях (KNN, SVM), градиентные методы (логистическая, линейная регрессия) и PCA.

21. Почему необходимо преобразовывать категориальные признаки перед обучением моделей?

Модели работают только с числами, а строковые признаки не имеют арифметического смысла, поэтому их нужно преобразовать, чтобы модель могла их использовать.

22. Что такое порядковый признак? Приведите пример.

Порядковый признак — это категории с логическим порядком, например: «низкий», «средний», «высокий».

23. Что такое номинальный признак? Приведите пример.

Номинальный признак — это категории без порядка, например: цвет («красный», «зелёный», «синий»).

24. Как работает метод .factorize() и в каких случаях он подходит?

.factorize() присваивает каждой категории уникальный номер в порядке появления; подходит, когда важна просто числовая замена без учёта порядка.

25. Чем принципиально отличается кодирование категориальных признаков с известными порядками?

Для порядковых признаков важно сохранить логический порядок (например, через map или OrdinalEncoder), а для номинальных используют one-hot, чтобы не создавать ложного числового сравнения.

26. Как работает LabelEncoder из scikit-learn?

LabelEncoder из scikit-learn присваивает каждой уникальной категории число, начиная с 0; работает для порядковых признаков, но нельзя напрямую применять на номинальных, чтобы не ввести ложный порядок.

27. Что такое one-hot encoding и когда оно применяется?

One-hot encoding создаёт отдельный бинарный столбец для каждой категории (1 — если категория, 0 — если нет); применяют для номинальных признаков без порядка.

28. Как избежать дамми-ловушки при one-hot кодировании?

Чтобы избежать дамми-ловушки, убирают один столбец из one-hot (например, через drop='first') — это предотвращает линейную зависимость между признаками.

29. Как работает OneHotEncoder из scikit-learn и чем он отличается от get_dummies()?

OneHotEncoder из scikit-learn создаёт one-hot кодировку и возвращает массив (часто разреженный), интегрируется с pipeline; get_dummies() из pandas возвращает DataFrame и проще в быстрой работе.

30. В чём суть метода target encoding и какие риски он с собой несёт?

Target encoding заменяет категорию на среднее целевого признака по этой категории; риски — утечка информации и переобучение, если применять без учёта разделения train/test.

Вывод: в ходе лабораторной работы были изучены основы исследовательского анализа данных с использованием библиотеки pandas. Рассмотрены этапы обработки данных в DataFrame, включая выявление и обработку пропущенных значений, удаление и коррекцию выбросов, масштабирование числовых признаков, а также методы кодирования категориальных переменных.

Ссылка на Google Colab:

https://colab.research.google.com/drive/1x2uUdAZaNypvEkvt_q0IIMHi0V5SmFSz?usp=sharing

Ссылка на Git Hub: https://github.com/darina-rtm/ai5lab.git