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»
Вариант 1

Выполнил:
Бакулин Вадим Романович
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.

Ссылка на репозиторий: https://github.com/zepteloid/AI_ML_LR_6

Порядок выполнения работы:

1. Задание 1:

```python
import seaborn as sns
import pandas as pd
import missingno as msno
import matplotlib.pyplot as plt

# Загрузка датасета
df = sns.load_dataset("titanic")

# Определение количества пропущенных значений
print("Количество пропущенных значений до обработки:")
print(df.isna().sum())

# Визуализация пропусков
msno.matrix(df)
plt.show()

# Заполнение пропусков
df['age'] = df['age'].fillna(df['age'].mean())  # Среднее значение
df['embarked'] = df['embarked'].fillna(df['embarked'].mode()[0])  # Наиболее частое значение
df = df.drop(columns=['deck'])  # Удаление столбца

# Проверка после обработки
print("\nКоличество пропущенных значений после обработки:")
print(df.isna().sum())

# Общая информация
print("\nИнформация о таблице после обработки:")
print(df.info())
```

Рисунок 1. Листинг программы задание 1

2. Задание 2:

```python
# Загрузка датасета
df = sns.load_dataset("penguins")

# Построение boxplot для указанных признаков
numeric_cols = ['bill_length_mm', 'bill_depth_mm', 'flipper_length_mm', 'body_mass_g']
for col in numeric_cols:
    plt.figure(figsize=(6, 4))
    sns.boxplot(x=df[col])
    plt.title(f"Boxplot для {col}")
    plt.show()

# Удаление выбросов с использованием IQR
def remove_outliers(df, column):
    Q1 = df[column].quantile(0.25)
    Q3 = df[column].quantile(0.75)
    IQR = Q3 - Q1
    lower = Q1 - 1.5 * IQR
    upper = Q3 + 1.5 * IQR
    return df[(df[column] >= lower) & (df[column] <= upper)]

original_size = df.shape[0]
for col in numeric_cols:
    df = remove_outliers(df, col)
new_size = df.shape[0]

# Сравнение размеров датасета
print(f"Размер датасета до удаления выбросов: {original_size}")
print(f"Размер датасета после удаления выбросов: {new_size}")

# Boxplot после удаления выбросов
plt.figure(figsize=(6, 4))
sns.boxplot(x=df['bill_length_mm'])
plt.title("Boxplot для bill_length_mm после удаления выбросов")
plt.show()
```

Рисунок 2. Листинг программы задание 2

3. Задание 3:

```python
# Загрузка данных
data = fetch_california_housing(as_frame=True)
df = data.frame

# Стандартизация
scaler_standard = StandardScaler()
df_standardized = df.copy()
df_standardized[df.columns] = scaler_standard.fit_transform(df)

# Нормализация
scaler_minmax = MinMaxScaler()
df_normalized = df.copy()
df_normalized[df.columns] = scaler_minmax.fit_transform(df)

# Гистограммы до и после масштабирования
plt.figure(figsize=(12, 5))
plt.subplot(1, 3, 1)
plt.hist(df['MedInc'], bins=20, color='blue', alpha=0.7)
plt.title("До масштабирования")

plt.subplot(1, 3, 2)
plt.hist(df_standardized['MedInc'], bins=20, color='green', alpha=0.7)
plt.title("После StandardScaler")

plt.subplot(1, 3, 3)
plt.hist(df_normalized['MedInc'], bins=20, color='red', alpha=0.7)
plt.title("После MinMaxScaler")

plt.tight_layout()
plt.show()
```

Рисунок 3. Листинг программы задание 3

4. Задание 4:

```python
import pandas as pd
from sklearn.preprocessing import LabelEncoder, OneHotEncoder

# Загрузка данных
file_path = "C:\\Users\\vadim\\jupyter-workspace\\AI_ML_LR_6\\data\\adult.data"
columns = ['age', 'workclass', 'fnlwgt', 'education', 'education-num', 'marital-status',
           'occupation', 'relationship', 'race', 'sex', 'capital-gain', 'capital-loss',
           'hours-per-week', 'native-country', 'income']

df = pd.read_csv(file_path, header=None, names=columns, na_values=' ?', skipinitialspace=True)

# Выбор нужных признаков
categorical_features = ['education', 'marital-status', 'occupation']
target_feature = 'income'

# Просмотр информации о данных перед обработкой
print("Информация о данных перед обработкой:")
print(df[categorical_features + [target_feature]].info())
print("\nПервые 5 строк данных:")
print(df[categorical_features + [target_feature]].head())

# 1. Label Encoding для признака education (предполагаем порядок)
# Создаем порядок уровней образования (от низшего к высшему)
education_order = [
    'Preschool', '1st-4th', '5th-6th', '7th-8th', '9th', '10th', '11th', '12th',
    'HS-grad', 'Some-college', 'Assoc-voc', 'Assoc-acdm', 'Bachelors', 'Masters',
    'Prof-school', 'Doctorate'
]

# Создаем словарь для соответствия
education_mapping = {v: i for i, v in enumerate(education_order)}
```

```python
# Применяем Label Encoding
df['education_encoded'] = df['education'].map(education_mapping)

# Проверяем результат
print("\nРезультат Label Encoding для education:")
print(df[['education', 'education_encoded']].head(10))

# 2. One-Hot Encoding для marital-status и occupation
# Сначала проверим наличие пропущенных значений
print("\nКоличество пропущенных значений:")
print(df[['marital-status', 'occupation']].isna().sum())

# Заполним пропуски в occupation модой
df['occupation'].fillna(df['occupation'].mode()[0], inplace=True)

# Применяем One-Hot Encoding с исключением одного столбца (избегаем дамми-ловушку)
df_encoded = pd.get_dummies(df, columns=['marital-status', 'occupation'], drop_first=True)

# Проверяем результат
print("\nСтолбцы после One-Hot Encoding:")
print(df_encoded.filter(regex='marital-status|occupation').columns)

# Проверяем итоговую размерность таблицы
print("\nРазмерность таблицы до кодирования:", df.shape)
print("Размерность таблицы после кодирования:", df_encoded.shape)

# Проверяем, что нет дамми-ловушки (один столбец удален для каждой категории)
print("\nПроверка на дамми-ловушку:")
print("Уникальные значения marital-status:", df['marital-status'].nunique())
print("Количество столбцов после кодирования:",
      len(df_encoded.filter(regex='marital-status').columns))
print("Уникальные значения occupation:", df['occupation'].nunique())
print("Количество столбцов после кодирования:",
      len(df_encoded.filter(regex='occupation').columns))

# Сохраняем обработанные данные
output_path = "C:\\Users\\vadim\\jupyter-workspace\\AI_ML_LR_6\\data\\adult_processed.csv"
df_encoded.to_csv(output_path, index=False)
print(f"\nОбработанные данные сохранены в: {output_path}")
```

Рисунок 4. Листинг программы задание 4

5. Задание 5:

```python
# Загрузка данных
file_path = "C:\\Users\\vadim\\jupyter-workspace\\AI_ML_LR_6\\data\\heart.csv"
df = pd.read_csv(file_path)

# Первые 5 строк датасета
print("Первые 5 строк датасета:")
display(df.head())

# Общая информация о данных
print("\nИнформация о датасете:")
display(df.info())

# Описательная статистика
print("\nОписательная статистика:")
display(df.describe().T)

# Проверка на пропущенные значения
print("Количество пропущенных значений в каждом столбце:")
display(df.isna().sum())

# Визуализация пропусков
msno.matrix(df)
plt.title('Матрица пропущенных значений')
plt.show()

# Выбор числовых признаков для анализа выбросов
numeric_cols = ['Age', 'RestingBP', 'Cholesterol', 'MaxHR', 'Oldpeak']

# Функция для удаления выбросов по методу IQR
def remove_outliers_iqr(data, column):
    Q1 = data[column].quantile(0.25)
    Q3 = data[column].quantile(0.75)
    IQR = Q3 - Q1
    lower = Q1 - 1.5 * IQR
    upper = Q3 + 1.5 * IQR
    return data[(data[column] >= lower) & (data[column] <= upper)]

# Построение boxplot до удаления выбросов
plt.figure(figsize=(15, 8))
for i, col in enumerate(numeric_cols, 1):
    plt.subplot(2, 3, i)
    sns.boxplot(y=df[col])
    plt.title(f'Boxplot для {col} (до обработки)')
plt.tight_layout()
plt.show()
```

```python
# Удаление выбросов
original_size = len(df)
for col in numeric_cols:
    df = remove_outliers_iqr(df, col)
new_size = len(df)

# Построение boxplot после удаления выбросов
plt.figure(figsize=(15, 8))
for i, col in enumerate(numeric_cols, 1):
    plt.subplot(2, 3, i)
    sns.boxplot(y=df[col])
    plt.title(f'Boxplot для {col} (после обработки)')
plt.tight_layout()
plt.show()

print(f"Размер датасета до удаления выбросов: {original_size}")
print(f"Размер датасета после удаления выбросов: {new_size}")
print(f"Удалено записей: {original_size - new_size} ({((original_size - new_size)/original_size)*100:.2f}%)")

# Создаем копию датасета для масштабирования
df_scaled = df.copy()

# Стандартизация (Z-преобразование)
scaler = StandardScaler()
df_scaled[numeric_cols] = scaler.fit_transform(df_scaled[numeric_cols])

# Визуализация распределения до и после масштабирования
plt.figure(figsize=(15, 6))

# До масштабирования
plt.subplot(1, 2, 1)
sns.histplot(df['Age'], kde=True)
plt.title('Распределение Age до масштабирования')

# После масштабирования
plt.subplot(1, 2, 2)
sns.histplot(df_scaled['Age'], kde=True)
plt.title('Распределение Age после StandardScaler')

plt.tight_layout()
plt.show()

# Проверка среднего и стандартного отклонения после масштабирования
print("\nСредние значения после StandardScaler:")
display(df_scaled[numeric_cols].mean())

print("\nСтандартные отклонения после StandardScaler:")
display(df_scaled[numeric_cols].std())
```

```python
# Определим порядковые и номинальные признаки
ordinal_features = ['ST_Slope']  # Упорядоченный признак: Down, Flat, Up
nominal_features = ['Sex', 'ChestPainType', 'RestingECG', 'ExerciseAngina']  # Номинальные признаки

# Label Encoding для порядкового признака
ordinal_mapping = {
    'Down': 0,
    'Flat': 1,
    'Up': 2
}
df_scaled['ST_Slope'] = df_scaled['ST_Slope'].map(ordinal_mapping)

# One-Hot Encoding для номинальных признаков
# Используем drop='first' для избежания дамми-ловушки
encoder = OneHotEncoder(drop='first', sparse_output=False)
encoded_nominal = encoder.fit_transform(df_scaled[nominal_features])
encoded_df = pd.DataFrame(encoded_nominal, columns=encoder.get_feature_names_out(nominal_features))

# Объединяем закодированные признаки с основным датасетом
df_final = pd.concat([df_scaled.drop(nominal_features, axis=1), encoded_df], axis=1)

# Проверка размерности до и после кодирования
print(f"Размерность до кодирования: {df_scaled.shape}")
print(f"Размерность после кодирования: {df_final.shape}")

# Просмотр итогового датасета
print("\nПервые 5 строк итогового датасета:")
display(df_final.head())

# Проверка итогового датасета
print("Информация об итоговом датасете:")
display(df_final.info())

# Сохранение обработанного датасета
output_path = "C:\\Users\\vadim\\jupyter-workspace\\AI_ML_LR_6\\data\\heart_processed.csv"
df_final.to_csv(output_path, index=False)
print(f"\nОбработанный датасет сохранен по пути: {output_path}")
```

Рисунок 5. Листинг программы задание 5

6. Индивидуальное задание:

```python
# 1. Обзор структуры данных
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.preprocessing import StandardScaler, LabelEncoder
from sklearn.impute import SimpleImputer

# Загрузка данных
file_path = "C:\\Users\\vadim\\jupyter-workspace\\AI_ML_LR_6\\data\\life_expectancy.csv"
data = pd.read_csv(file_path)

# Проверим реальные названия столбцов
print("Столбцы в датасете:", data.columns.tolist())

# Попробуем найти столбец, который может быть целевой переменной
# Обычно он может называться по-разному, например:
possible_target_names = ['Life expectancy', 'Life_Expectancy', 'LifeExpectancy', 'Life expectancy (years)', 'target']
target_col = None

for name in possible_target_names:
    if name in data.columns:
        target_col = name
        break

if target_col is None:
    # Если ни одно из стандартных названий не подошло, используем последний столбец как целевую переменную
    target_col = data.columns[-1]
    print(f"\nЦелевая переменная не найдена, используем последний столбец: {target_col}")

print("\nПервые 5 строк датасета:")
print(data.head())
print("\nИнформация о датасете:")
print(data.info())
print("\nОписательная статистика:")
print(data.describe(include='all'))

selected_features = [target_col] + ['Adult Mortality', 'GDP', 'Schooling']
selected_features = [col for col in selected_features if col in data.columns]

# Визуализация выбросов с помощью boxplot
plt.figure(figsize=(15, 5))
for i, col in enumerate(selected_features, 1):
    plt.subplot(1, len(selected_features), i)
    sns.boxplot(y=data[col])
    plt.title(col)
plt.tight_layout()
plt.show()
```

```python
# Удаление выбросов с помощью метода IQR
def remove_outliers(df, columns):
    for col in columns:
        Q1 = df[col].quantile(0.25)
        Q3 = df[col].quantile(0.75)
        IQR = Q3 - Q1
        lower_bound = Q1 - 1.5 * IQR
        upper_bound = Q3 + 1.5 * IQR
        df = df[(df[col] >= lower_bound) & (df[col] <= upper_bound)]
    return df

print(f"Размер датасета до удаления выбросов: {data.shape}")
data_clean = remove_outliers(data, selected_features)
print(f"Размер датасета после удаления выбросов: {data_clean.shape}")
print(f"Удалено {len(data) - len(data_clean)} строк ({((len(data) - len(data_clean))/len(data))*100:.2f}% данных)")

# Масштабирование числовых признаков (кроме целевой переменной)
numeric_features = data_clean.select_dtypes(include=['int64', 'float64']).columns
numeric_features = numeric_features.drop('Life expectancy', errors='ignore')  # исключаем целевую переменную

scaler = StandardScaler()
data_scaled = data_clean.copy()
data_scaled[numeric_features] = scaler.fit_transform(data_scaled[numeric_features])

print("\nДанные после масштабирования:")
print(data_scaled[numeric_features].head())

# Анализ категориальных признаков
categorical_features = data_scaled.select_dtypes(include=['object']).columns
print("\nКатегориальные признаки:", categorical_features)

# Label Encoding для порядковых признаков (если есть)
# Пример для признака 'Income composition of resources' (если он порядковый)
if 'Income composition of resources' in categorical_features:
    le = LabelEncoder()
    data_scaled['Income composition of resources'] = le.fit_transform(data_scaled['Income composition of resources'])

# One-Hot Encoding для номинальных признаков
nominal_features = ['Country', 'Status']  # пример номинальных признаков
data_encoded = pd.get_dummies(data_scaled, columns=nominal_features, drop_first=True)  # drop_first для избежания дамми-ловушки

print("\nДанные после кодирования:")
print(data_encoded.head())

# Проверка на дамми-ловушку
print("\nПроверка на дамми-ловушку:")
print(f"Количество столбцов до One-Hot Encoding: {len(data_scaled.columns)}")
print(f"Количество столбцов после One-Hot Encoding: {len(data_encoded.columns)}")

# Проверка финального датасета
print("\nФинальная информация о датасете:")
print(data_encoded.info())

# Сохранение обработанных данных
output_path = "C:\\Users\\vadim\\jupyter-workspace\\AI_ML_LR_6\\data\\life_expectancy_processed.csv"
data_encoded.to_csv(output_path, index=False)
print(f"\nОбработанные данные сохранены по пути: {output_path}")

# Вывод первых 5 строк финального датасета
print("\nПервые 5 строк финального датасета:")
print(data_encoded.head())
```

Рисунок 6. Листинг программы индивидуального задания

Ответы на контрольные вопросы:

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

Пропущенные значения могут искажать статистические показатели, приводить к ошибкам в моделях машинного обучения и снижать мощность выборки.

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

В pandas можно использовать методы `df.isna().sum()` для подсчета пропусков по столбцам и `df.isna().any()` для проверки их наличия.

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

Метод `.dropna()` удаляет строки или столбцы с пропущенными значениями. Основные параметры:

- `axis=0` (строки) или `axis=1` (столбцы)

- `how='any'` (удалить, если есть хотя бы один пропуск) или `'all'` (если все значения пропущены)

- `subset` для указания столбцов

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

- Среднее подходит для нормального распределения, но чувствительно к выбросам

- Медиана устойчива к выбросам, хороша для асимметричных данных

- Мода используется для категориальных данных

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

`fillna(method='ffill')` заполняет пропуски предыдущим известным значением. Применяется в временных рядах и данных с естественным порядком.

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

`interpolate()` вычисляет промежуточные значения между известными точками (линейная, полиномиальная интерполяция), тогда как `fillna()` просто заменяет пропуски фиксированными значениями.

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

Выбросы - аномальные значения, значительно отличающиеся от основной массы данных. Они искажают статистические показатели и работу моделей.

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

Метод IQR (межквартильный размах):

- IQR = Q3 (75-й перцентиль) - Q1 (25-й перцентиль)
- Границы выбросов: Q1 - 1.5*IQR (нижняя), Q3 + 1.5*IQR (верхняя)

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

```
Q1 = df['column'].quantile(0.25)
Q3 = df['column'].quantile(0.75)
IQR = Q3 - Q1
df = df[(df['column'] >= Q1-1.5*IQR) & (df['column'] <= Q3+1.5*IQR)]
```

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

Метод `.clip()` ограничивает значения заданными границами, заменяя выбросы на пороговые значения.

11. Зачем может потребоваться логарифмическое преобразование числовых признаков?

Логарифмическое преобразование (`np.log1p()`) уменьшает асимметрию распределения и сжимает диапазон больших значений.

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

Для обнаружения выбросов используют:

- Boxplot (ящик с усами)
- Точечные диаграммы (scatter plot)

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

Удаление выбросов требует осторожности, так как они могут содержать важную информацию о редких, но значимых событиях.

14. Зачем необходимо масштабирование признаков перед обучением моделей?

Масштабирование признаков необходимо для:

- Алгоритмов, чувствительных к масштабу данных (KNN, SVM, нейросети)

- Ускорения сходимости градиентного спуска

15. Чем отличается стандартизация от нормализации?

Отличия:

- Стандартизация: (x - mean)/std, диапазон ≈[-3,3], сохраняет выбросы

- Нормализация: (x - min)/(max - min), диапазон [0,1], чувствительна к выбросам

16. Что делает StandardScaler и как рассчитываются преобразованные значения?

`StandardScaler` преобразует данные к среднему=0 и std=1. Формула: (x - μ)/σ

17. Как работает MinMaxScaler и когда его использование предпочтительно?

`MinMaxScaler` сжимает данные в диапазон [0,1]. Подходит, когда важны границы значений.

18. В чём преимущества RobustScaler при наличии выбросов?

`RobustScaler` использует медиану и IQR, устойчив к выбросам.

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

df['column'] = (df['column'] - df['column'].mean())/df['column'].std()

20. Какие типы моделей наиболее чувствительны к масштабу признаков?

Наиболее чувствительны к масштабу: KNN, SVM, линейные модели, нейросети.

21. Почему необходимо преобразовывать категориальные признаки перед обучением модели?

Категориальные признаки преобразуют в числовые, так как большинство алгоритмов работают только с числами.

22. Что такое порядковый признак? Приведите пример.

Порядковый признак - категории с естественным порядком (например, "низкий", "средний", "высокий").

23. Что такое номинальный признак? Приведите пример.

Номинальный признак - категории без порядка (например, цвета, названия городов).

24. Как работает метод .factorize() и для каких случаев он подходит?

Метод `.factorize()` присваивает категориям числовые коды (0,1,2...). Подходит для порядковых данных.

25. Как применить метод .map() для кодирования категориальных признаков с известным порядком?

mapping = {'низкий':0, 'средний':1, 'высокий':2}

df['column'] = df['column'].map(mapping)

26. Что делает класс OrdinalEncoder из scikit-learn?

`OrdinalEncoder` из sklearn аналогичен `.factorize()`, но работает с несколькими столбцами.

27. В чём суть one-hot кодирования и когда оно применяется?

One-hot кодирование создает отдельный бинарный столбец для каждой категории. Применяется для номинальных признаков.

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

Чтобы избежать дамми-ловушки, один из столбцов удаляют (`drop_first=True` в `pd.get_dummies()`).

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

`OneHotEncoder` из sklearn интегрируется в pipeline, а `pd.get_dummies()` проще в использовании.

30. В чём суть метода target encoding и какие риски он в себе несёт?

Target encoding заменяет категории средним значением целевой переменной. Риски: утечка данных и переобучение на редких категориях.

Вывод: В ходе лабораторной работы были изучены и применены основные этапы исследовательского анализа данных (EDA), включая обнаружение и обработку пропущенных значений, выявление и устранение выбросов, масштабирование числовых признаков и кодирование категориальных переменных. Практические задания позволили закрепить навыки работы с библиотеками pandas, scikit-learn и missingno для подготовки данных к дальнейшему анализу и моделированию. Результатом работы стало освоение универсальных методов EDA, которые могут быть применены к любым структурированным данным для повышения качества анализа и прогнозирования.