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 №4
дисциплины
«Искусственный интеллект и машинное обучение»
Вариант № 4

Выполнил:
Левашев Тимур Рашидович
2 курс, группа ИВТ-б-о-23-2,
09.03.01 «Информатика и вычислительная техника»,
направленность (профиль) «Программное обеспечение средств вычислительной техники и автоматизированных систем», очная форма обучения

_____
(подпись)

Проверил:
Доцент департамента цифровых, робототехнических систем и электроники института перспективной инженерии Воронкин В.И

_____
(подпись)

Отчет защищен с оценкой _____ Дата защиты_____

Ставрополь, 2025 г.

Тема работы: “Введение в pandas: изучение структуры Series и базовых операций”.

Цель работы: познакомить с основами работы с библиотекой pandas, в частности, со структурой данных Series.

Ссылка на git репозиторий: https://github.com/mazy99/ml_prakt_4

Порядок выполнения работы:

1. Создание Series из списка.

```
import pandas as pd
import numpy as np
ndarr = np.array([5,15,25,35,45])
indexes = ['a','b','c','d','e']
s3 = pd.Series(ndarr,indexes,dtype=int)
print(s3)
print(f'Тип данных Series: {s3.dtype}')
```

```
a     5
b    15
c    25
d    35
e    45
dtype: int32
Тип данных Series: int32
```

Рисунок 1 – Код и результат работы кода

2. Получение элемента Series.

```
num_arr = np.array([12, 24, 36,48, 60])
index_arr = ['A', 'B', 'C', 'D', 'E']
series = pd.Series(num_arr,index_arr,dtype=int)
print(f"Элеменет с меткой 'C': {series.loc['C']}")
print(f"Третий элемент: {series.iloc[2]}")
```

```
Элеменет с меткой 'C': 36
Третий элемент: 36
```

Рисунок 2 – Код и результат работы кода

3. Фильтрация данных с помощью логической индексации.

```
num_arr = np.array([4, 9, 16, 25, 36, 49, 64])
series = pd.Series(num_arr)
print(f"Элементы ряда больше 20:\n{series[series > 20]}")
```

```
Элементы ряда больше 20:
3    25
4    36
5    49
6    64
dtype: int32
```

Рисунок 3 – Код и результат работы кода

4. Просмотр первых и последних элементов.

```
series = pd.Series(np.random.randint(1,100,50))
print(f"Первые 7 элементов:\n{series.head(7)}")
print(f"Послдение 5 элементов:\n{series.tail(5)}")
```

```
Первые 7 элементов:
0    80
1    69
2    32
3    44
4    71
5    83
6    19
dtype: int32
Послдение 5 элементов:
45    85
46    40
47    75
48    65
49    72
dtype: int32
```

Рисунок 4 – Код и результат работы кода

5. Просмотр первых и последних элементов.

```
series = pd.Series(['cat', 'dog', 'rabbit', 'parrot', 'fish'])
print(f'Тип данных ряда series: {series.dtype}')
series=series.astype('category')
print(f'Привел к новому типу данных: {series.dtype}')
```

```
Тип данных ряда series: object
Привел к новому типу данных: category
```

Рисунок 5 – Код и результат работы кода

6. Проверка пропущенных значений.

```
series= pd.Series([1.2, np.nan, 3.4, np.nan, 5.6, 6.8])
print(f'Проверка на NaN:\n{series.isna()}')
nan_index = series.index[series.isna()]
print(f'Индексы NaN элементов:\n{nan_index}')
```

```
Проверка на NaN:
0    False
1     True
2    False
3     True
4    False
5    False
dtype: bool
Индексы NaN элементов:
Index([1, 3], dtype='int64')
```

Рисунок 6 – Код и результат работы кода

7. Заполнение пропущенных значений.

```
series= pd.Series([1.2, np.nan, 3.4, np.nan, 5.6, 6.8])
series=series.fillna(series.mean())
print(f'Новый ряд:\n{series}')
```

```
Новый ряд:
0    1.20
1    4.25
2    3.40
3    4.25
4    5.60
5    6.80
dtype: float64
```

Рисунок 7 – Код и результат работы кода

8. Арифметические операции с series.

```
s1 = pd.Series([10, 20, 30, 40], index=['a', 'b', 'c', 'd'])
s2 = pd.Series([5, 15, 25, 35], index=['b', 'c', 'd', 'e'])
result = s1+s2
print(f'Ряд с NaN элементами:\n{result}')
#В РЕЗУЛЬТАТЕ СЛОЖЕНИЯ ПОЯВЯТСЯ  NaN ЭЛЕМЕНТЫ, Т.К ИНЕДКСЫ МАССИВОВ НЕ
result=result.fillna(0)
print(f'Ряд с NaN элементами, замененными 0:\n{result}')
```

```
Ряд с NaN элементами:
a     NaN
b    25.0
c    45.0
d    65.0
e     NaN
dtype: float64
Ряд с NaN элементами, замененными 0:
a     0.0
b    25.0
c    45.0
d    65.0
e     0.0
dtype: float64
```

Рисунок 8 – Код и результат работы кода

9. Применение функции к series.

```
series = pd.Series([2, 4, 6, 8, 10])
new_series = series.apply(lambda x: np.sqrt(x))
print(f'Применение функции к series:\n{new_series}')
```

```
Применение функции к series:
0    1.414214
1    2.000000
2    2.449490
3    2.828427
4    3.162278
dtype: float64
```

Рисунок 9 – Код и результат работы кода

10. Основные статистические методы.

```
series = pd.Series(np.random.uniform(50,150,size=20))
sum_ser = series.sum()
mean_ser = series.mean()
min_ser = series.min()
max_ser = series.max()
std_ser = series.std()
print(f'Сумма ряда: {sum_ser}')
print(f'Среднее ряда: {mean_ser}')
print(f'Максимальное значение: {max_ser}')
print(f'Минимальноее значение: {min_ser}')
print(f'Стандартное отклонение: {std_ser}')
```

```
Сумма ряда: 1914.9962640189926
Среднее ряда: 95.74981320094963
Максимальное значение: 141.7167860115191
Минимальноее значение: 51.57295673824039
Стандартное отклонение: 26.411980266261153
```

Рисунок 10 – Код и результат работы кода

11. Работа с временными рядами.

```
series = pd.Series(np.random.uniform(10,100,size=10),index=pd.date_ran
print(f'Данные за 5-8 марта:\n{series.loc['2024-03-05':'2024-03-08']}'
```

```
Данные за 5-8 марта:
2024-03-05    88.167756
2024-03-06    25.974880
2024-03-07    95.141806
2024-03-08    82.905370
Freq: D, dtype: float64
```

Рисунок 11 – Код и результат работы кода

12. Работа с временными рядами.

```
series = pd.Series([10, 20, 30, 40, 50, 60], index=['A', 'B', 'A', 'C', 'D', 'B'])
print("Уникальны ли индексы?", series.index.is_unique)
if not series.index.is_unique:
    series = series.groupby(level=0).sum()
print(f"Series после суммирования:\n{series}")
```

```
Уникальны ли индексы? False
Series после суммирования:
A    40
B    80
C    40
D    50
dtype: int64
```

Рисунок 12 – Код и результат работы кода

13. Преобразование строковых дат в DatetimeIndex.

```
series = pd.Series([100, 200, 300],index=['2024-03-10', '2024-03-11','2024-03-12'])
print(series)
series.index = pd.to_datetime(series.index)
print(f'Тип данных индекса: {series.index.dtype}')
```

```
2024-03-10    100
2024-03-11    200
2024-03-12    300
dtype: int64
Тип данных индекса: datetime64[ns]
```

Рисунок 13– Код и результат работы кода

14. Чтение данных из CSV-файла.

```
data = {
    'Дата': ['2024-03-01', '2024-03-02', '2024-03-03', '2024-03-04', '2024-03-05'],
    'Цена': [100, 110, 105, 120, 115]
}
df = pd.DataFrame(data)
df.to_csv('data.csv', index=False)
df = pd.read_csv('data.csv', parse_dates=['Дата'])
series = pd.Series(df['Цена'].values, index=df['Дата'])
print(series)
print("CSV файл 'data.csv' был создан.")
```

```
Дата
2024-03-01    100
2024-03-02    110
2024-03-03    105
2024-03-04    120
2024-03-05    115
dtype: int64
CSV файл 'data.csv' был создан.
```

Рисунок 14 – Код и результат работы кода

15. Построение графиков на основе series.

```
import matplotlib.pyplot as plt
series = pd.Series(np.random.uniform(50,150,size=30),index=pd.date_range(start='2024-03-01', periods=30, freq='D'))
series.plot(kind='bar')
plt.title('Пример графика для Series')
plt.xlabel('Индекс')
plt.ylabel('Значения')
plt.grid(True)
plt.show()
print(series)
```

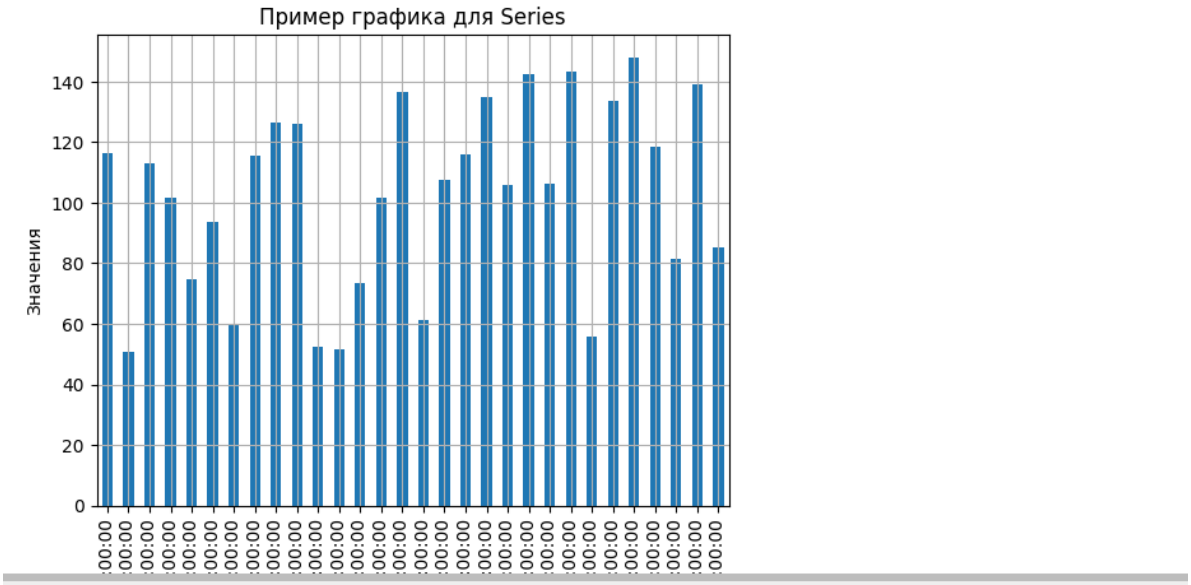

Рисунок 15 – Код и результат работы кода

16. Выполнение индивидуального задания.

```
import pandas as pd

data = {
    'Дата': ['2024-06-01', '2024-06-02', '2024-06-03', '2024-06-04', '2024-06-05'],
    'Курс USD/RUB': [80.5, 81.2, 79.8, 82.0, 80.9]
}
df = pd.DataFrame(data)
df.to_csv('exchange_rate.csv', index=False)
print("CSV файл 'exchange_rate.csv' был создан.")
df = pd.read_csv('exchange_rate.csv', parse_dates=['Дата'])
df.set_index('Дата', inplace=True)
df['Разница'] = df['Курс USD/RUB'].diff()
plt.figure(figsize=(10, 6))
plt.plot(df.index, df['Курс USD/RUB'], marker='o', label='Курс USD/RUB')
plt.vlines(df.index[df['Разница'].notna()], ymin=0, ymax=df['Курс USD/RUB'].max(), color='r', linestyle='--', label='Изменения курса')
plt.title('График курса USD/RUB с вертикальными линиями изменений')
plt.xlabel('Дата')
plt.ylabel('Курс USD/RUB')
plt.legend()
plt.grid(True)
plt.xticks(rotation=45)
plt.tight_layout()
```

```
CSV файл 'exchange_rate.csv' был создан.
```

Рисунок 16 – Код и результат работы кода

Ответы на контрольные вопросы:

1.Что такое pandas.Series и чем она отличается от списка в Python?

pandas.Series — это одномерная структура данных в библиотеке pandas, аналогичная списку, но с возможностью задания индексов и работы с данными более эффективно. В отличие от списка, Series поддерживает векторизированные операции и удобные методы для анализа данных.

2. Какие типы данных можно использовать для создания Series ?

Series поддерживает любые скалярные типы: int, float, str, bool, datetime, а также object (любой Python-объект).

3. Как задать индексы при создании Series ?

s = pd.Series([10, 20, 30], index=['a', 'b', 'c'])

4. Каким образом можно обратиться к элементу Series по его индексу?

print(s['b'])

5. В чём разница между .iloc[] и .loc[] при индексации Series ?

.iloc[] — индексирует по порядковому номеру (s.iloc[0] — первый элемент).

.loc[] — индексирует по заданному индексу (s.loc['b'] — элемент с индексом 'b').

6. Как использовать логическую индексацию в Series ?

s[s > 15]

7. Какие методы можно использовать для просмотра первых и последних элементов Series ?

s.head(n) — первые n элементов.

s.tail(n) — последние n элементов.

8. Как проверить тип данных элементов Series ?

s.dtype

9. Каким способом можно изменить тип данных Series ?

s = s.astype(float)

10. Как проверить наличие пропущенных значений в Series ?

s.isna()

11. Какие методы используются для заполнения пропущенных значений в Series ?

s.fillna(value) — замена NaN значением.

s.ffill() — замена предыдущим значением.

s.bfill() — замена следующим значением.

12. Чем отличается метод .fillna() от .dropna() ?

.fillna() заменяет пропущенные значения.

.dropna() удаляет строки с NaN.

13. Какие математические операции можно выполнять с Series ?

Арифметические (+, -, *, /), логические (>, <, ==), а также встроенные методы (sum(), mean(), std() и др.).

14. В чём преимущество векторизированных операций по сравнению с циклами Python?

Они выполняются быстрее за счёт использования оптимизированных C-библиотек.

15. Как применить пользовательскую функцию к каждому элементу Series ?

s.apply(lambda x: x * 2)

16. Какие агрегирующие функции доступны в Series ?

sum(), mean(), median(), std(), var(), min(), max(), count().

17. Как узнать минимальное, максимальное, среднее и стандартное отклонение Series ?

s.min(), s.max(), s.mean(), s.std()

18. Как сортировать Series по значениям и по индексам?

По значениям: s.sort_values()

По индексам: s.sort_index()

19. Как проверить, являются ли индексы Series уникальными?

s.index.is_unique

20. Как сбросить индексы Series и сделать их числовыми?

s.reset_index(drop=True)

21. Как можно задать новый индекс в Series ?

s.index = ['x', 'y', 'z']

22. Как работать с временными рядами в Series ?

Использовать DatetimeIndex и функции resample(), rolling().

23. Как преобразовать строковые даты в формат DatetimeIndex ?

s.index = pd.to_datetime(s.index)

24. Каким образом можно выбрать данные за определённый временной диапазон?

s['2023-01-01':'2023-02-01']

25. Как загрузить данные из CSV-файла в Series ?

s = pd.read_csv('data.csv', usecols=['column_name'], squeeze=True)

26. Как установить один из столбцов CSV-файла в качестве индекса Series ?

s = pd.read_csv('data.csv', index_col='column_name', squeeze=True)

27. Для чего используется метод .rolling().mean() в Series ?

Для вычисления скользящего среднего.

28. Как работает метод .pct_change() ? Какие задачи он решает?

Вычисляет процентное изменение между соседними значениями.

29. В каких ситуациях полезно использовать .rolling() и .pct_change() ?

.rolling() — при анализе временных рядов.

.pct_change() — для расчёта прироста (например, в финансах).

30. Почему NaN могут появляться в Series , и как с ними работать?

NaN появляются из-за отсутствующих данных, деления на ноль и преобразований.

Работа с ними: fillna(), dropna(), isna().