УДК 004.989

ОПИСАТЕЛЬНЫЙ И РАЗВЕДОЧНЫЙ АНАЛИЗ ДАННЫХ С ИСПОЛЬЗОВАНИЕМ ВОЗМОЖНОСТЕЙ ЯЗЫКА PYTHON

Канавальчик А.Д. [1]*, студент гр.253502*

Белорусский государственный университет информатики и радиоэлектроники[1]*, г. Минск, Республика Беларусь*

Жвакина А.В. – канд. техн. наук

Аннотация. В научной работе исследованы особенности описательного и разведочного анализа данных с использованием языка Python. В качестве практического примера проведен анализ готового набора данных расходов на медицинское обслуживание, построены модель и график линейной регрессии, отображающие прогнозирование данных.

Ключевые слова. Анализ данных, описательный анализ, разведочный анализ, python.

Введение.

Постоянно растущий объем информации вызывает необходимость анализа данных, используемых в науке, бизнесе, медицине и других областях, с целью их оптимального использования. Для решения данной задачи существует целый ряд методов, среди которых следует выделить описательный и разведочный анализ данных.

Описательный анализ позволяет получить общее представление о данных, выявить основные характеристики и закономерности. Разведочный анализ, в свою очередь, помогает выявить скрытые зависимости и паттерны в данных, что может послужить основой для дальнейших исследований и принятия решений.

Основная часть.

Широкие возможности для исследования данных предоставляет язык программирования Python. Одним из основных инструментов для описательного анализа данных является библиотека Pandas, которая позволяет загружать, обрабатывать и анализировать данные в форме таблицы [1]. С помощью Pandas можно проводить различные статистические расчеты, визуализировать данные и исследовать их распределение. Не менее полезной является библиотека NumPy, которая чаще всего используется для работы с числовыми массивами.

Для разведочного анализа данных часто используют библиотеку Matplotlib или более современный инструмент Seaborn. С их помощью можно создавать графики различных типов: гистограммы, диаграммы рассеяния, ящики с усами и т.д. Это помогает наглядно представить данные и выявить закономерности. Кроме того, популярны библиотеки SciPy для выполнения статистических тестов и scikit-learn для машинного обучения.

Важным этапом в анализе данных является обработка пропущенных значений, выбросов и дубликатов. Для этого можно использовать методы Pandas, такие как dropna(), fillna() и drop_duplicates().

Кроме того, в Python есть возможность создания интерактивных дашбордов с помощью библиотеки Plotly или Dash, что позволяет пользователю взаимодействовать с данными и исследовать их более глубоко.

Для более глубокого понимания особенностей описательного и разведочного анализа исследуем уже готовый набор данных, позаимствованный из открытого источника [2]. У нас имеется таблица, которая отображает расходы на медицинское обслуживание людей, которые имеют медицинскую страховку, в зависимости от их возраста, пола, индекса массы тела, наличия детей, наличия привычки курения и региона проживания.

Для начала подключим все нужные нам библиотеки: Pandas, NumPy, MatPlotLib.pyplot, Seaborn, Warnings (рис. 1). Воспользовавшись функцией .describe(), выведем сводку описательных статистик для используемого набора данных. Как видим, сводка включает только количественные показатели (рис. 1).

```python
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
import warnings
warnings.filterwarnings('ignore')

df = pd.read_csv('../SCIENTIFIC WORK/insurance.csv')
print(df.describe())
```

```
                age          bmi     children       charges
count  1338.000000  1338.000000  1338.000000   1338.000000
mean     39.207025    30.663397     1.094918  13270.422265
std      14.049960     6.098187     1.205493  12110.011237
min      18.000000    15.960000     0.000000   1121.873900
25%      27.000000    26.296250     0.000000   4740.287150
50%      39.000000    30.400000     1.000000   9382.033000
75%      51.000000    34.693750     2.000000  16639.912515
max      64.000000    53.130000     5.000000  63770.428010
```

Рисунок 1 – Результат выполнения метода .describe()

Иногда параметры описательной статистики нужно рассчитать по группам. Например, вычислить средние значения возраста (age), индекса массы тела (bmi) и медицинских расходов (charges) отдельно для курящих и для некурящих застрахованных. Можно применить метод mean() к сгруппированным с помощью метода .groupby() данным (рис. 2).

```python
print(df.groupby('smoker').mean())
```

```
              age       sex        bmi  children    region       charges
smoker
0       39.385338  0.485902  30.651795  1.090226  1.516917   8434.268298
1       38.514599  0.580292  30.708449  1.113139  1.510949  32050.231832
```

Рисунок 2 – Результат выполнения метода .groupby('smoker').mean()

Как видим, средний возраст для курящих застрахованных и некурящих практически одинаков, а вот расходы отличаются значительно.

Также можно вывести сводку по описательным статистикам для сгруппированных данных (рис. 3).

```python
print(df.groupby('smoker').describe())
```

```
          age                                  ...      charges
        count       mean        std  ...          50%          75%          max
smoker                                 ...
0      1064.0  39.385338  14.083410  ...   7345.40530  11362.887050  36910.60803
1       274.0  38.514599  13.923186  ...  34456.34845  41019.207275  63770.42801

[2 rows x 48 columns]
```

Рисунок 3 – Результат выполнения метода .groupby('smoker').describe()

При помощи метода .isnull() мы можем получить объект, в котором на соответствующих элементам исходного DataFrame стоят значения False/True, и каждое значение True соответствует пропуску в исходном наборе данных. Число пропусков в каждом столбце можно вычислить с использованием метода .isnull().sum() (рис. 4).

```python
print(df.isnull().sum())
```

```
age         0
sex         0
bmi         0
children    0
smoker      0
region      0
charges     0
dtype: int64
```

Рисунок 4 – Результат выполнения метода .isnull().sum()

Узнать тип данных каждого столбца можно при помощи метода .dtypes(), как видно из рисунка 5.

```
print(df.dtypes)
```

```
age           int64
sex          object
bmi         float64
children      int64
smoker       object
region       object
charges     float64
dtype: object
```

Рисунок 5 – Результат выполнения метода .dtypes()

Разведочный анализ данных основан на построении визуализаций. Начнем с возможностей визуализации данных библиотеки Matplotlib [3]. Matplotlib изначально предназначался для работы с массивами NumPy. Применение функций Matplotlib для визуализации DataFrame, наиболее распространенного представления для набора данных, во многих случаях требует дополнительных преобразований данных. Кроме библиотек для работы с данными pandas и NumPy, нам понадобится и модуль библиотеки Matplotlib: pyplot. Модуль matplotlib.pyplot представляет собой менеджер графического интерфейса фигур, подобный MATLAB.

Построим столбчатую диаграмму медицинских расходов в зависимости от региона проживания пациента. Для этого сгруппируем данные из столбца “charges” по регионам, после чего просуммируем значения затрат в каждом регионе и сортируем результаты по возрастанию. Далее создадим новую фигуру с использованием метода .subplots() для построения графика. После чего при помощи метода .barplot() библоитеки Seaborn создаем столбчатую диаграмму на основании полученных выше результатов, вдобавок задаем цветовую палитру “Blues”. Отобразим полученную диаграмму, воспользовавшись методом .show() (рис. 6).

```
charges = df['charges'].groupby(df.region).sum().sort_values(ascending_=_True)
f, ax = plt.subplots( nrows: 1,  ncols: 1, figsize=(8, 6))
ax = sns.barplot(charges.head(), palette='Blues')
plt.show()
```

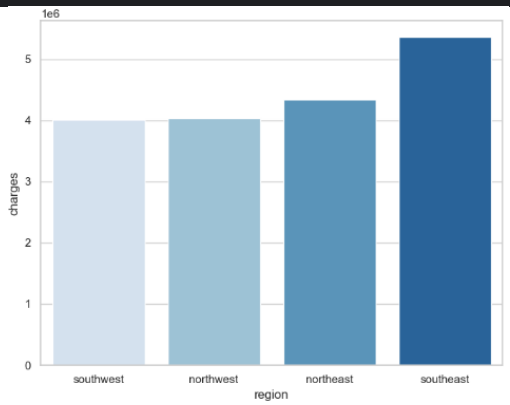

Рисунок 6 – Столбчатая диаграмма расходов в зависимости от региона проживания

Таким образом, самые высокие расходы на медицинское обслуживание наблюдаются на юго-востоке, а самые низкие — на юго-западе. Учитывая определенные факторы (пол, курение, наличие детей) посмотрим, как расходы меняются по регионам. Создадим три диаграммы, каждая из которых выражает зависимость медицинских расходов от пола (рис. 8), курения (рис. 9) и наличия детей (рис. 10) соответственно по регионам. Воспользуемся ранее упомянутыми методами: .subplots() – создание фигуры для построения диаграммы, .barplot() – создание столбчатой диаграммы с заданными параметрами и .show() – отображение созданной диаграммы (рис. 7).

```
f, ax = plt.subplots( nrows: 1,  ncols: 1, figsize=(12, 8))
ax = sns.barplot(x='region', y='charges', hue='sex', data=df, palette='cool')
plt.show()

f, ax = plt.subplots( nrows: 1, ncols: 1, figsize=(12,8))
ax = sns.barplot(x='region', y='charges', hue='smoker', data=df, palette='Reds_r')
plt.show()

f, ax = plt.subplots( nrows: 1,  ncols: 1, figsize=(12, 8))
ax = sns.barplot(x='region', y='charges', hue='children', data=df, palette='Set1')
plt.show()
```

Рисунок 7 – Создание столбчатых диаграмм

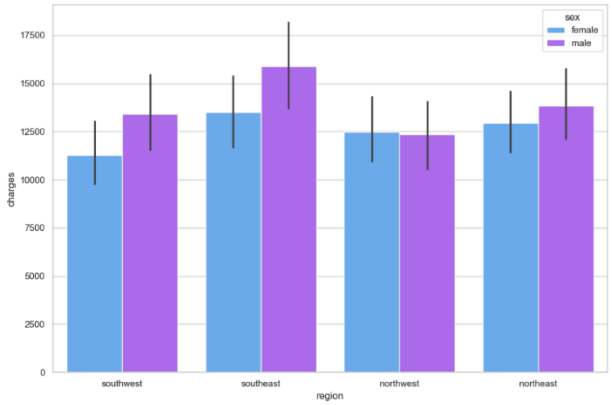

Рисунок 8 – Столбчатая диаграмма расходов в зависимости от пола и региона

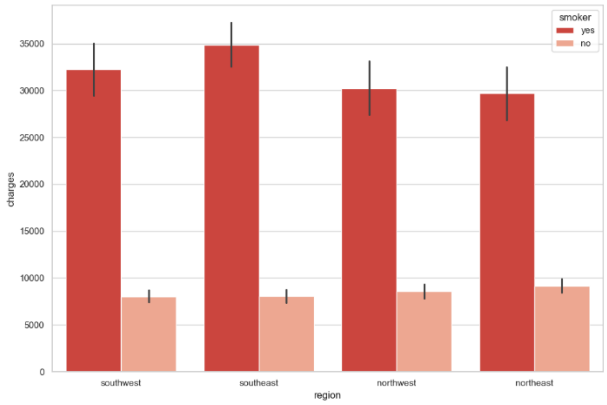

Рисунок 9 – Столбчатая диаграмма расходов в зависимости от курения и региона

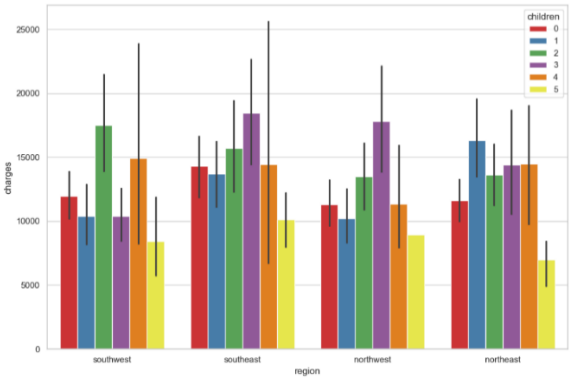

Рисунок 10 – Столбчатая диаграмма расходов в зависимости от наличия детей и региона

Как мы видим из этих графиков, самые высокие расходы у курящих людей по-прежнему наблюдаются на юго-востоке, а самые низкие - на северо-востоке. Жители юго-запада, как правило, курят больше, чем жители северо-востока, но у жителей северо-востока более высокие расходы в разбивке по полу, чем на юго-западе и северо-западе в целом. А у людей с детьми, как правило, также более высокие расходы на медицинское обслуживание.

Теперь давайте проанализируем расходы на медицинское обслуживание в разбивке по возрасту (рис. 12), ИМТ (индексу массы тела) (рис. 13) и количеству детей (рис. 14) в зависимости от фактора курения. Создадим 3 точечных графика при помощи метода .lmplot() библиотеки Seaborn. В качестве аргументов функции указываем по оси абсцисс (x) – значения категориального показателя, по оси ординат (y) – распределения количественной переменной, data = df – аргумент, который

указывает на DataFrame, содержащий данные для построения графика, hue = 'smoker' – это аргумент, который указывает на столбец 'smoker' в DataFrame df, который будет использоваться для цветовой группировки данных (в данном случае, данные будут разделены по категориям курящих и некурящих), palette = 'Set1' – это аргумент, который задает цветовую палитру для графика (в данном случае используется палитра 'Set1') (рис. 11).

```
ax = sns.lmplot(x = 'age', y = 'charges', data=df, hue='smoker', palette='Set1')
ax = sns.lmplot(x = 'bmi', y = 'charges', data=df, hue='smoker', palette='Set2')
ax = sns.lmplot(x = 'children', y = 'charges', data=df, hue='smoker', palette='Set3')
plt.show()
```

Рисунок 11 – Создание точечных графиков

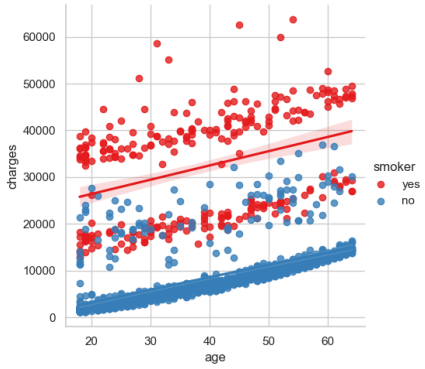

Рисунок 12 – Точечный график расходов в разбивке по возрасту в зависимости от курения

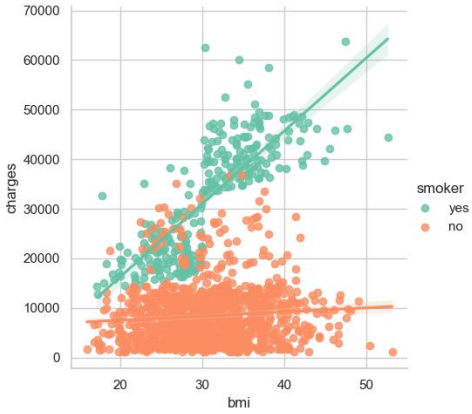

Рисунок 13 – Точечный график расходов в разбивке по ИМТ в зависимости от курения

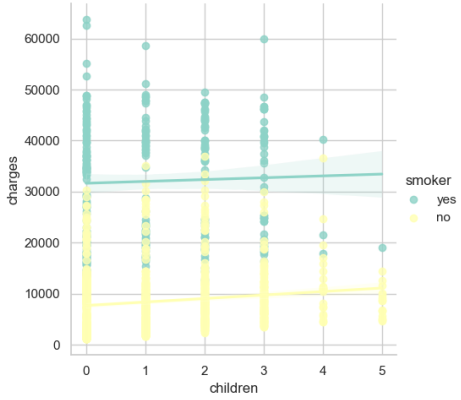

Рисунок 14 – Точечный график расходов в разбивке по количеству детей в зависимости от курения

Проанализировав полученные точечные графики, можно сделать вывод, что курение оказывает огромное влияние на медицинские расходы, хоть затраты и растут с возрастом, ИМТ и количеством детей. Кроме того, люди, у которых есть дети, обычно меньше курят, о чем свидетельствует

скрипичный график (рис. 15). Для создания скрипичного графика расходов в разбивке по количеству детей в зависимости от курения воспользуемся методом .violinplot() библиотеки Seaborn (рис. 16). В качестве аргументов данной функции будут выступать: x = 'children' – указывает на столбец 'children' в DataFrame df, который будет использоваться для оси X на графике, y='charges' – указывает на столбец 'charges' в DataFrame df, который будет использоваться для оси Y на графике, data=df – указывает на DataFrame, содержащий данные для построения графика, orient='v' – определяет ориентацию скрипичного графика как вертикальную, hue='smoker' – указывает на столбец 'smoker' в DataFrame df, который будет использоваться для цветовой группировки данных (в данном случае данные будут разделены по категориям курящих и некурящих), palette='inferno' – задает цветовую палитру для графика (в данном случае используется палитра 'inferno').

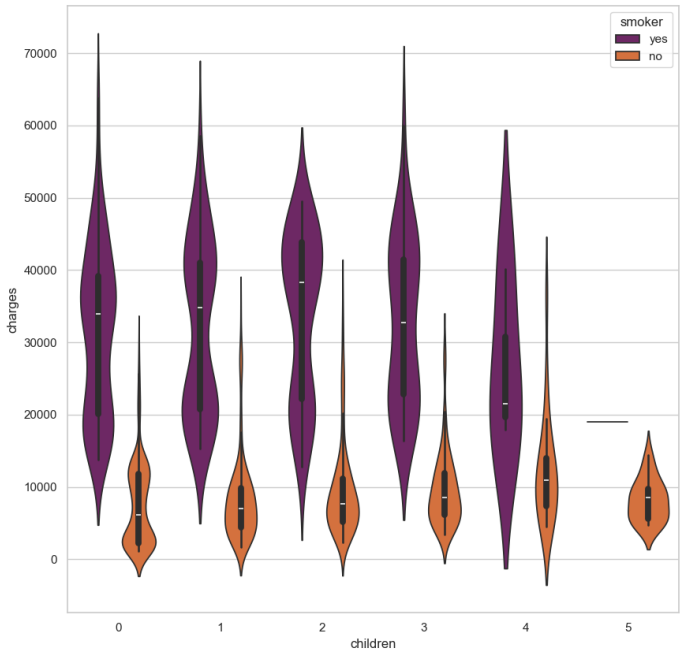

Рисунок 15 – Скрипичный график расходов в разбивке по количеству детей в зависимости от курения

```
f, ax = plt.subplots( nrows: 1,  ncols: 1, figsize=(10, 10))
ax = sns.violinplot(x = 'children', y = 'charges', data=df, orient='v', hue='smoker', palette='inferno')
plt.show()
```

Рисунок 16 – Создание скрипичного графика

Построим модель линейной регрессии для прогнозирования затрат на медицинское обслуживание. Воспользуемся библиотекой scikit-learn, которая представляет возможности машинного обучения в Python [4]. Импортируем функцию train_test_split из модуля model_selection данной библиотеки (рис. 17). Эта функция используется для разделения данных на обучающий и тестовый наборы. К тому же импортируем класс LinearRegression из модуля linear_model той же библиотеки (рис. 17). Этот класс представляет линейную регрессию, алгоритм машинного обучения для построения линейной модели. Импортируем модуль metrics из библиотеки scikit-learn, который содержит различные метрики оценки моделей машинного обучения (рис. 17).

```
from sklearn.model_selection import train_test_split as holdout
from sklearn.linear_model import LinearRegression
from sklearn import metrics
```

Рисунок 17 – Подключение библиотек и модулей

Приступим к созданию линейной модели. Создадим переменную x, содержащую все столбцы данных, кроме столбца 'charges' (рис. 18). Это признаки, которые будут использоваться для предсказания целевой переменной. После чего создадим переменную y, содержащую столбец 'charges', который является целевой переменной для предсказания (рис. 18). Далее вызовем функцию

.train_test_split (которую мы переименовали в holdout) для разделения данных на обучающий (x_train, y_train) и тестовый (x_test, y_test) наборы (рис. 18). Тестовый набор составляет 20% от всех данных, а random_state=0 используется для воспроизводимости результатов. Создадим экземпляр класса LinearRegression, который будет использоваться для построения линейной модели (рис. 18). При помощи функции .fit() обучается модель линейной регрессии на обучающих данных (x_train, y_train) (рис. 18). Выведем полученные результаты: значение смещения (intercept) модели линейной регрессии, значения коэффициентов наклона (coefficients) модели линейной регрессии, коэффициент детерминации (R^2) модели на тестовых данных (рис. 18). Коэффициент детерминации показывает, насколько хорошо модель соответствует данным. Чем ближе значение к 1, тем лучше модель объясняет изменчивость данных.

```
x = df.drop( labels: ['charges'], axis = 1)
y = df['charges']
x_train, x_test, y_train, y_test = holdout( *arrays: x, y, test_size=0.2, random_state=0)
Lin_reg = LinearRegression()
Lin_reg.fit(x_train, y_train)
print(Lin_reg.intercept_)
print(Lin_reg.coef_)
print(Lin_reg.score(x_test, y_test))
```

Рисунок 18 – Создание модели линейной регрессии

Делаем предсказания на тестовом наборе признаков x_test, вызвав метод .predict() объекта Lin_reg (модели линейной регрессии) (рис. 19). Предсказанные значения сохраняются в переменной y_pred. Создадим новую фигуру с указанным размером (10, 6) в дюймах (рис. 19). Это задает размеры графика, на котором будут отображаться данные. Построим точечную диаграмму (scatter plot) с предсказанными значениями y_pred на оси ординат и реальными значениями y_test на оси абсцисс (рис. 19). Точки на графике будут синего цвета. На графике добавим пунктирную линию, представляющую идеальную ситуацию, когда предсказанные значения равны реальным значениям (рис. 19). Линия будет красного цвета. Подпишем ось абсцисс (X) графика как “Actual charges”, а ось ординат (Y) как “Predicted charges” (рис. 19). Установим заголовок графика как “Actual vs Predicted Charges (Linear Regression)” (рис. 19). Отобразим созданный нами график уже знакомой функцией .show() (рис. 20).

```
# Make predictions
y_pred = Lin_reg.predict(x_test)

# Plot actual vs predicted charges
plt.figure(figsize=(10, 6))
plt.scatter(y_test, y_pred, color='blue')
plt.plot( *args: [y_test.min(), y_test.max()], [y_test.min(), y_test.max()], '--', color='red')
plt.xlabel('Actual Charges')
plt.ylabel('Predicted Charges')
plt.title('Actual vs Predicted Charges (Linear Regression)')
plt.show()
```

Рисунок 19 – Создание графика линейной регрессии

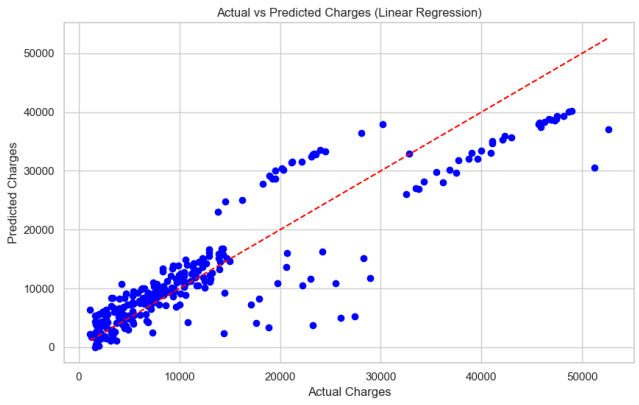

Рисунок 20 – График линейной регрессии

Таким образом, нам удалось построить график линейной регрессии, который демонстрирует актуальные и предсказуемые расходы на медицинское обслуживание. Применение машинного обучения в дальнейшем может помочь спрогнозировать наивероятнейшие исходы событий, тем самым минимизируя негативные исходы.

Как я уже отмечала ранее, курение является основным фактором, влияющим на расходы на медицинское обслуживание, затем следуют ИМТ и возраст.

Итак, с использованием возможностей библиотек Pandas, NumPy, Seaborn, MatPlotLib, Sklearn нам удалось провести описательный и разведочный анализ данных на примере конкретного датасета. Были построены графики расходов на медицинское обслуживание в зависимости от различных факторов. Кроме того, нам удалось построить модель линейной регрессии, на основании которой был построен график, отражающий актуальные и предсказуемые расходы на медицинское обслуживание.

Заключение.

Использование возможностей языка Python для описательного и разведочного анализа данных является эффективным и удобным инструментом для исследователей и аналитиков. Python предлагает богатый набор библиотек для работы с данными, визуализации и статистического анализа, что позволяет быстро и точно изучать и интерпретировать данные. К тому же, воспользовавшись методами библиотеки Sklearn, появляется возможность на основании анализируемых данных разрабатывать и обучать модели предсказания, что позволяет выявлять тенденции и закономерности в данных, а также принимать обоснованные решения на основе прогнозов.

Список использованных источников:

1. Введение в pandas: анализ данных на Python [Электронный ресурс] / – Режим доступа: https://khashtamov.com/ru/pandas-introduction. – Дата доступа: 10.04.2024

2. Medical Cost Personal Datasets [Электронный ресурс] / – Режим доступа: https://www.kaggle.com/datasets/mirichoi0218/insurance. – Дата доступа: 10.04.2024

3. Matplotlib: Visualization with Python [Электронный ресурс] / – Режим доступа: https://matplotlib.org. – Дата доступа: 10.04.2024

4. Scikit-learn: Machine Learning in Python [Электронный ресурс] / – Режим доступа: https://scikit-learn.org/stable. – Дата доступа: 10.04.2024

UDC 004.989

DESCRIPTIVE AND EXPLORATORY DATA ANALYSIS USING PYTHON

Kanavalchik A.D. [1]

Belarusian State University of Informatics and Radioelectronics[1]*, Minsk, Republic of Belarus*

Zhvakina A.V. – Candidate of Technical Sciences

Annotation. The scientific article explores the capabilities of the Python language for descriptive and exploratory data analysis. As a practical example, a descriptive and exploratory analysis of a completed data set was carried out. Using the machine learning capabilities of Python, a linear regression model and graph were built to show the prediction of the data.

Keywords. Data analysis, descriptive analysis, exploratory analysis, python.