|  |
| --- |
| МИНОБРНАУКИ РОССИИ |
| Федеральное государственное бюджетное образовательное учреждение высшего образования **«МИРЭА − Российский технологический университет»**  **РТУ МИРЭА** |

**Институт информационных технологий (ИИТ)**

**Кафедра прикладной математики (ПМ)**

**ОТЧЕТ ПО ПРАКТИЧЕСКОЙ РАБОТЕ**

по дисциплине «Технологии и инструментарий анализа больших данных»

**Практическое занятие № 4**

|  |  |  |  |
| --- | --- | --- | --- |
| Студент группы ИНБО-01-17 | *ИМБО-02-22, Ким Кирилл Сергеевич* | (подпись) | |
| Преподаватель | *Тетерин Николай Николаевич, преподаватель* | (подпись) | |
| Отчет представлен | «\_\_\_»\_\_\_\_\_\_\_\_202\_\_г. | |  | |

Москва 2025 г.

СОДЕРЖАНИЕ

[ПРАКТИЧЕСКАЯ РАБОТА 4 3](#_Toc212551842)

[Введение 3](#_Toc212551843)

[Шаги выполнения 3](#_Toc212551844)

[Результат работы: 14](#_Toc212551845)

[Вывод: 14](#_Toc212551846)

[Список использованных источников и литературы: 14](#_Toc212551847)

ПРАКТИЧЕСКАЯ РАБОТА 4

Введение

**Цель**

Освоить методы статистического анализа данных, включая корреляционный анализ, построение линейной регрессии с использованием градиентного спуска и метода наименьших квадратов, а также применение дисперсионного анализа (ANOVA) для исследования влияния факторов на количественные показатели.

Шаги выполнения

1. Определить два вектора, представляющие собой число автомобилей, припаркованных в течении 5 рабочих дней у бизнес-центра на уличной стоянке и в подземном гараже

Листинг 1 – Задание 1

|  |
| --- |
| import numpy as np  import pandas as pd  import matplotlib.pyplot as plt  import seaborn as sns  from scipy import stats  df = pd.DataFrame({      'День': ['Понедельник', 'Вторник', 'Среда', 'Четверг', 'Пятница'],      'Улица': [80, 98, 75, 91, 78],      'Гараж': [100, 82, 105, 89, 102]  })  df |

1. Найти и интерпретировать корреляцию между переменными «Улица» и «Гараж» (подсчитать корреляцию по Пирсону).

Листинг 2 – Задание 1.1

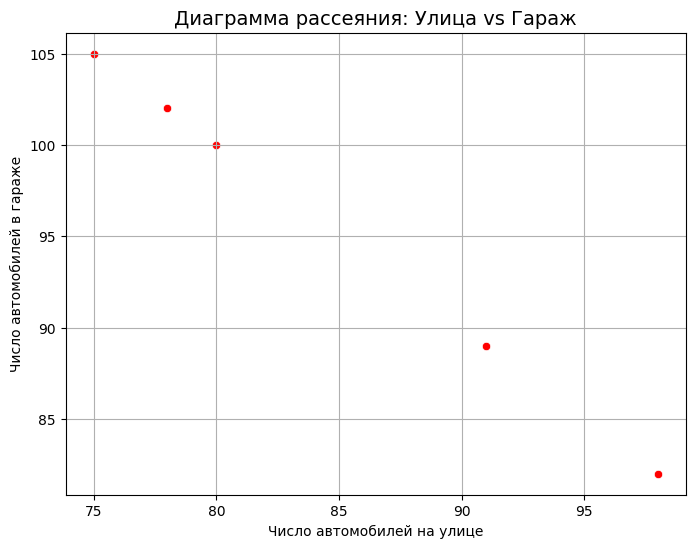
|  |
| --- |
| # Расчет корреляции Пирсона  var1 = df['Улица'].to\_numpy()  var2 = df['Гараж'].to\_numpy()  correlation = np.corrcoef(var1, var2)[0, 1]  print(f"Коэффициент корреляции Пирсона: {correlation:.4f}") |

Коэффициент корреляции практически равен -1, что указывает на полную обратную линейную зависимость.

1. Построить диаграмму рассеяния для вышеупомянутых переменных.

Листинг 3 – Задание 1.2

|  |
| --- |
| plt.figure(figsize=(8, 6))  sns.scatterplot(x=var1, y=var2, color='r')  plt.grid(True)  plt.title('Диаграмма рассеяния: Улица vs Гараж', fontsize=14)  plt.xlabel('Число автомобилей на улице')  plt.ylabel('Число автомобилей в гараже')  plt.grid(True)  plt.show() |



**Рисунок 1 – Статистика по категориальным признакам**

1. Найти и выгрузить данные. Вывести, провести предобработку и описать признаки. Набор данных о ценах мобильных телефонов. Поля: разрешение, бренд, размер, вес, качество изображения, объём оперативной памяти, ёмкость аккумулятора и мощность процессора. В этом наборе данных мы хотим оценить стоимость мобильных телефонов с учётом вышеперечисленных характеристик.

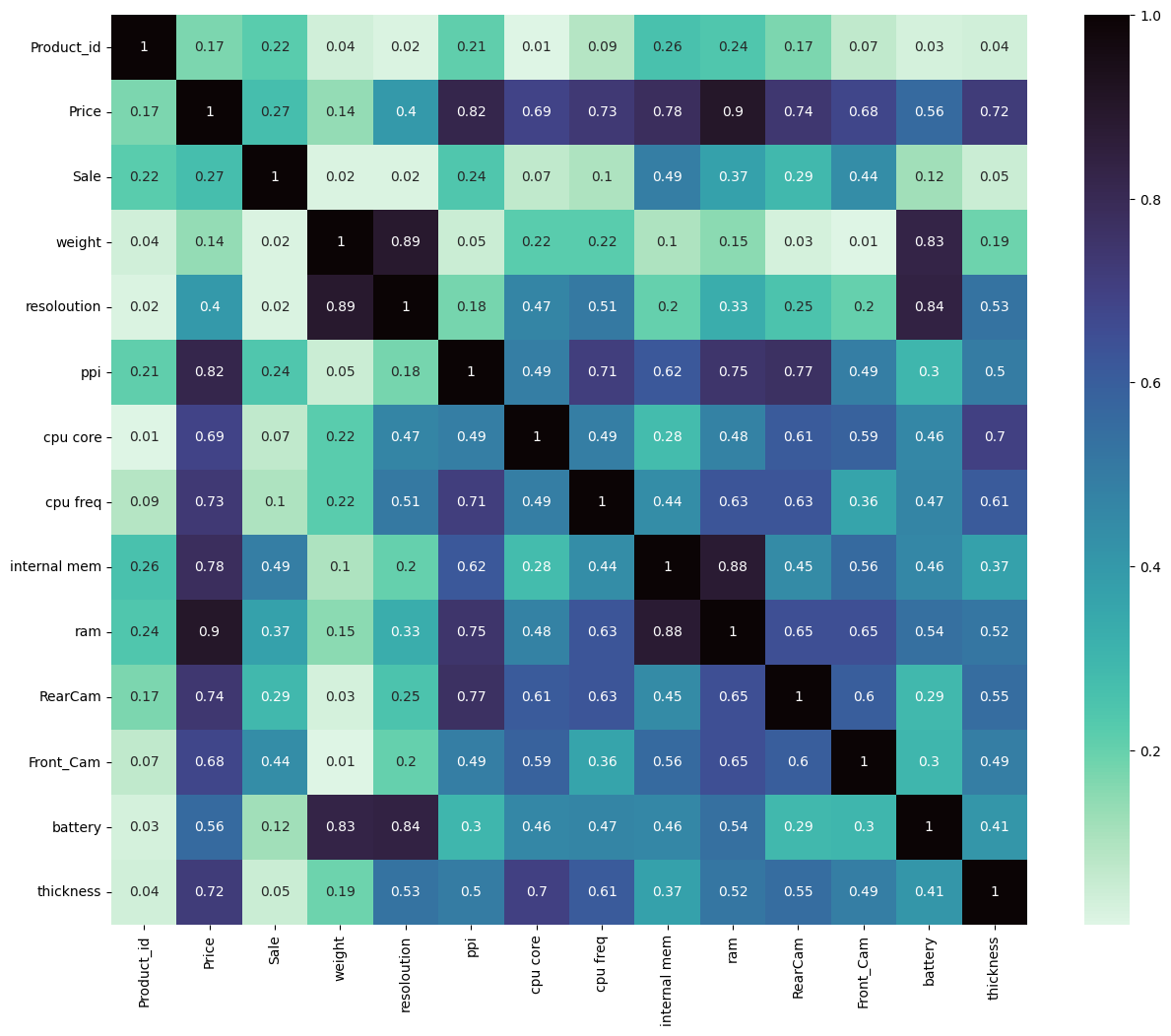
Листинг 4 – Задание 2

|  |
| --- |
| !pip install opendatasets  import pandas as pd  import numpy as np  from sklearn.linear\_model import LinearRegression  from sklearn.preprocessing import StandardScaler  from sklearn.metrics import mean\_squared\_error  import matplotlib.pyplot as plt  # kirillkim71  # 39d1b74c7680ac8e32302e0efa5ae562  import opendatasets as od  od.download('https://www.kaggle.com/datasets/mohannapd/mobile-price-prediction')  df2 = pd.read\_csv('/content/mobile-price-prediction/Cellphone.csv')  df2  df2.info()  df2.isnull().sum()  df2.duplicated().sum()  scaler = StandardScaler()  col = df2.columns  result = scaler.fit\_transform(df2)  df2 = pd.DataFrame(result, columns=col)  df2 |

1. Построить корреляционную матрицу по одной целевой переменной. Определить наиболее коррелирующую переменную, продолжить с ней работу в следующем пункте представлено на Рисунке 2.

Листинг 5 – Задание 2.1

|  |
| --- |
| corr\_matrix = df2.corr().Price.abs().to\_frame()  corr\_matrix  fig, ax = plt.subplots(figsize=(15, 12))  sns.heatmap(round(df2.corr(), 2).abs(), annot=True, cmap='mako\_r') |



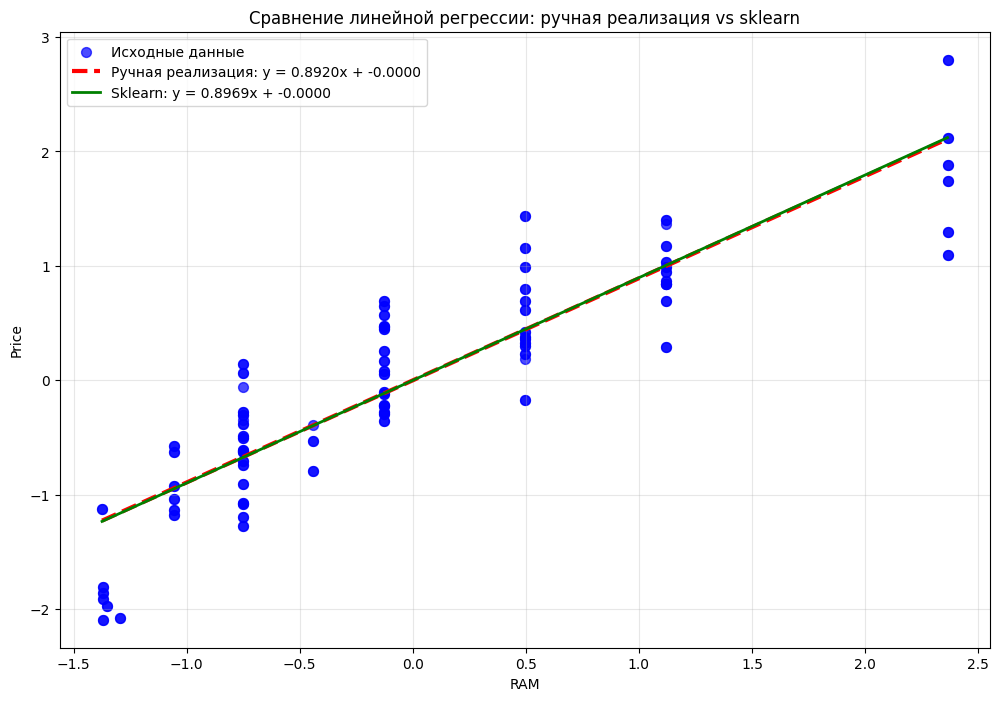
**Рисунок 2 – Heatmap**

Наиболее коррелируемая переменная – Price.

1. Реализовать регрессию вручную, отобразив наклон, сдвиг и MSE. Так же реализуем линейную регрессию с помощью библиотеки sklearn представлено на Рисунке 3.

Листинг 6 – Задание 2.2

|  |
| --- |
| def mse(X, w1, w0, y):      y\_pred = w1 \* X + w0      return np.mean((y - y\_pred) \*\* 2)  def grad(X, w1, w0, y):      y\_pred = w1 \* X + w0      error = y - y\_pred      n = len(y)        grad\_w0 = -2 \* np.mean(error)      grad\_w1 = -2 \* np.mean(error \* X)        return np.array([grad\_w0, grad\_w1])  X\_data = df2[['ram']].values.flatten()  y\_data = df2['Price'].values  eps = 0.0001  lr = 0.01  w1, w0 = 0, 0  mse\_history = []  w1\_history = []  w0\_history = []  # Градиентный спуск  n = 10000  for i in range(n):      current\_w1, current\_w0 = w1, w0        # Вычисляем градиент      gradient = grad(X\_data, current\_w1, current\_w0, y\_data)        # Обновляем веса      w0 = current\_w0 - lr \* gradient[0]      w1 = current\_w1 - lr \* gradient[1]        # Вычисляем MSE      current\_mse = mse(X\_data, current\_w1, current\_w0, y\_data)      mse\_history.append(current\_mse)      w1\_history.append(current\_w1)      w0\_history.append(current\_w0)        # Вывод прогресса каждые 1000 итераций      if i % 1000 == 0:          print(f"Итерация {i}: MSE = {current\_mse:.6f}, w1 = {current\_w1:.6f}, w0 = {current\_w0:.6f}")        # Критерий остановки      if (abs(w1 - current\_w1) < eps and abs(w0 - current\_w0) < eps):          print(f"Сходимость достигнута на итерации {i}")          break  final\_w1, final\_w0 = w1, w0  final\_mse = mse(X\_data, final\_w1, final\_w0, y\_data)  print(f"w1 = {final\_w1:.6f}, w0 = {final\_w0:.6f}")  print(f"MSE: {final\_mse:.6f}")  print(f"y = {final\_w1:.4f} \* x + {final\_w0:.4f}")  # Сравнение с sklearn  # Модель sklearn на одном признаке (ram)  X\_ram = df2[['ram']]  y = df2['Price']  model\_simple = LinearRegression()  model\_simple.fit(X\_ram, y)  print(f"Ручная модель:  y = {final\_w1:.4f}x + {final\_w0:.4f}")  print(f"Sklearn модель: y = {model\_simple.coef\_[0]:.4f}x + {model\_simple.intercept\_:.4f}")  # Предсказания  y\_pred\_manual = final\_w1 \* X\_data + final\_w0  y\_pred\_sklearn\_simple = model\_simple.predict(X\_ram)  # MSE для сравнения  mse\_manual = mse(X\_data, final\_w1, final\_w0, y\_data)  mse\_sklearn\_simple = mean\_squared\_error(y, y\_pred\_sklearn\_simple)  print(f"MSE ручной модели: {mse\_manual:.6f}")  print(f"MSE sklearn (ram): {mse\_sklearn\_simple:.6f}")  # Модель sklearn на всех признаках  X\_all = df2.drop('Price', axis=1)  y\_all = df2['Price']  X\_train, X\_test, y\_train, y\_test = train\_test\_split(X\_all, y\_all, test\_size=0.3, random\_state=42)  model\_all = LinearRegression()  model\_all.fit(X\_train, y\_train)  y\_pred\_all = model\_all.predict(X\_test)  mse\_all = mean\_squared\_error(y\_test, y\_pred\_all)  print(f"MSE sklearn (все признаки): {mse\_all:.6f}")  # Визуализация линейной регрессии: сравнение ручной реализации и sklearn  plt.figure(figsize=(12, 8))  # Данные  plt.scatter(X\_data, y\_data, alpha=0.7, color='blue', label='Исходные данные', s=50)  # Сортируем данные для гладких линий  sorted\_indices = np.argsort(X\_data)  X\_sorted = X\_data[sorted\_indices]  # Линия ручной модели (градиентный спуск)  y\_manual\_sorted = final\_w1 \* X\_sorted + final\_w0  plt.plot(X\_sorted, y\_manual\_sorted, '--', linewidth=3, color='red',           label=f'Ручная реализация: y = {final\_w1:.4f}x + {final\_w0:.4f}')  # Линия sklearn  y\_sklearn\_sorted = model\_simple.coef\_[0] \* X\_sorted + model\_simple.intercept\_  plt.plot(X\_sorted, y\_sklearn\_sorted, linewidth=2, color='green',           label=f'Sklearn: y = {model\_simple.coef\_[0]:.4f}x + {model\_simple.intercept\_:.4f}')  plt.xlabel('RAM')  plt.ylabel('Price)')  plt.title('Сравнение линейной регрессии: ручная реализация vs sklearn')  plt.legend()  plt.grid(True, alpha=0.3)  plt.show()  print(f"Ручная реализация: y = {final\_w1:.6f}x + {final\_w0:.6f}")  print(f"Sklearn:           y = {model\_simple.coef\_[0]:.6f}x + {model\_simple.intercept\_:.6f}")  print(f"MSE ручной:  {mse\_manual:.6f}")  print(f"MSE sklearn: {mse\_sklearn\_simple:.6f}") |



**Рисунок 3 – Линейная регрессия**

1. Загрузить данные: 'insurance.csv'. Вывести и провести предобработку. Список уникальных регионов

Листинг 7 – Задание 3

|  |
| --- |
| df3 = pd.read\_csv("insurance.csv")  df3  df3.isnull().sum()  df3.duplicated().sum()  df3['region'].unique() |

1. Выполнить однофакторный ANOVA тест, чтобы проверить влияние региона на индекс массы тела (BMI), используя первый способ, через библиотеку Scipy.

Листинг 8 – Задание 3.1

|  |
| --- |
| !pip install scipy  from scipy.stats import f\_oneway  df3\_anova = df3[['region', 'bmi']]  groups = df3\_anova.groupby('region').groups  southwest = groups['southwest']  southeast = groups['southeast']  northwest = groups['northwest']  northeast = groups['northeast']  f\_stat, p\_value = f\_oneway(southeast, southwest, northeast, northwest)  print(f"One-way ANOVA: F-statistic = {f\_stat:.4f}, p-value = {p\_value:.4f}") |

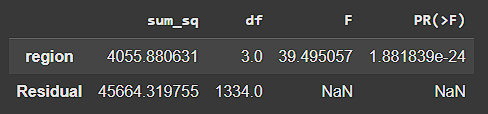
One-way ANOVA: F-statistic = 0.3467, p-value = 0.7915

Следовательно, регион не оказывает статически значительно влияние на bmi.

1. Выполнить однофакторный ANOVA тест, чтобы проверить влияние региона на индекс массы тела (BMI), используя второй способ, с помощью функции anova\_lm() из библиотеки statsmodels.

Листинг 9 – Задание 3.2

|  |
| --- |
| import statsmodels.api as sm  from statsmodels.formula.api import ols  anova\_model = ols('bmi ~ region', data=df3).fit()  anova\_result = sm.stats.anova\_lm(anova\_model, typ=2)  anova\_result |

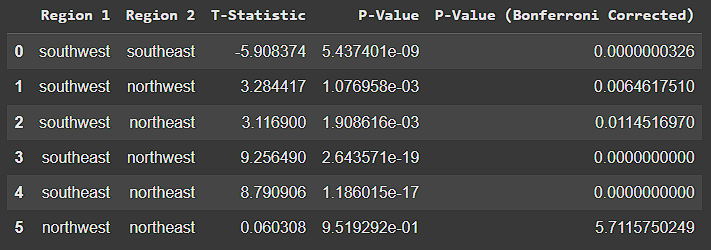


**Рисунок 4 – Однофакторный ANOVA через statsmodels**

1. С помощью t-критерия Стьюдента перебрать все пары. Определить поправку Бонферрони.

Листинг 10 – Задание 3.3

|  |
| --- |
| regions = ['southwest', 'southeast', 'northwest', 'northeast']  region\_pair = []  from itertools import combinations  from scipy.stats import ttest\_ind  for region1, region2 in combinations(regions, 2):      group1 = df3[df3['region'] == region1]['bmi']      group2 = df3[df3['region'] == region2]['bmi']      t\_stat, p\_value = ttest\_ind(group1, group2)      region\_pair.append(          {              'Region 1': region1,              'Region 2': region2,              'T-Statistic': t\_stat,              'P-Value': p\_value          }      )  results\_df = pd.DataFrame(region\_pair)  alpha = 0.05  results\_df['P-Value (Bonferroni Corrected)'] = results\_df['P-Value'] \* len(region\_pair)  results\_df['P-Value (Bonferroni Corrected)'] = results\_df['P-Value (Bonferroni Corrected)'].apply(lambda x: "{:.10f}".format(x))  results\_df |

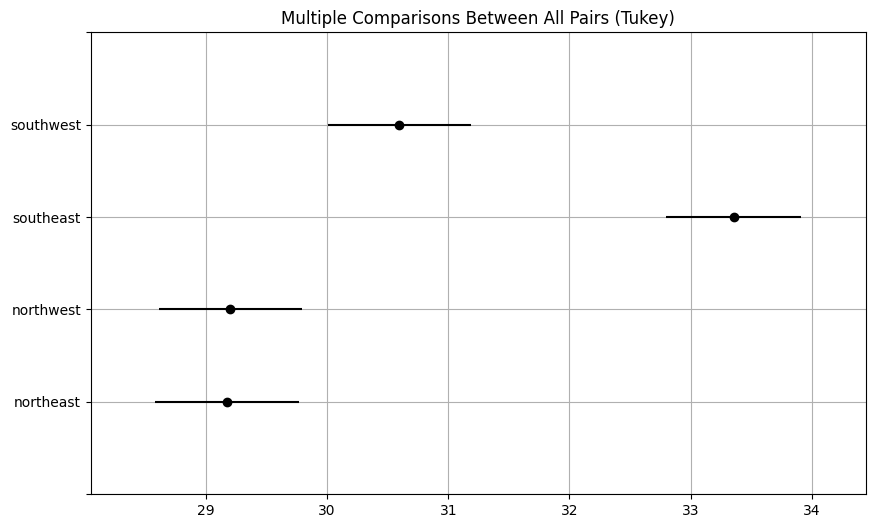


**Рисунок 5 – Результат t-критерия**

1. Выполнить пост-хок тесты Тьюки и построить график

Листинг 11 – Задание 3.4

|  |
| --- |
| from statsmodels.stats.multicomp import pairwise\_tukeyhsd  tukey = pairwise\_tukeyhsd(endog=df3['bmi'].values, groups=df3['region'], alpha=0.05)  tukey.plot\_simultaneous()  plt.vlines(x=49.57, ymin=-0.5, ymax=4.5, color="red")  plt.grid()  tukey.summary() |

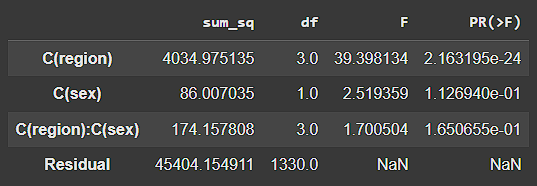


**Рисунок 6 – Пост-хок тест Тьюки**

1. Выполнить двухфакторный ANOVA тест, чтобы проверить влияние региона и пола на индекс массы тела (BMI), используя функцию anova\_lm() из библиотеки statsmodels.

Листинг 12 – Задание 3.5

|  |
| --- |
| model = ols('bmi ~ C(region) \* C(sex)', data=df3).fit()  anova\_table = sm.stats.anova\_lm(model, typ=2)  anova\_table |



**Рисунок 7 – Двухфакторный ANOVA**

Результат работы:

Данную работу можете увидеть в блокноте Jupyter Notebook.

<https://drive.google.com/file/d/15jXq4rj4JR_JSOUqIIYxIztoqIjoiYBa/view?usp=sharing>

Вывод:

В ходе практической работы были успешно освоены и применены на практике основные методы статистического анализа данных: корреляционный анализ, линейная регрессия и дисперсионный анализ.

Список использованных источников и литературы:

1. Ростовцев В.С. Искусственные нейронные сети,   
   Издательство "Лань", 2019. — 216 с. — URL: https://e.lanbook.com/book/122180
2. Араки М. Манга: Машинное обучение,   
   Издательство "ДМК Пресс", 2020. — 214 с. — URL: <https://e.lanbook.com/book/179473>
3. Платонов, А. В. Машинное обучение : учебное пособие для вузов / А. В. Платонов. — Москва : Издательство Юрайт, 2022. — 85 с. — (Высшее образование). — ISBN 978-5-534-15561-7. — Текст : электронный // Образовательная платформа Юрайт [сайт]. — URL: https://urait.ru/bcode/508804