|  |
| --- |
| 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 машинного обучения»

**Практическое занятие № 7**

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

Москва 2024 г.

СОДЕРЖАНИЕ

[ПРАКТИЧЕСКАЯ РАБОТА 7 3](#_Toc184928618)

[Задание 3](#_Toc184928619)

[Шаги выполнения 4](#_Toc184928620)

[Этап 1. Подготовка данных: 4](#_Toc184928621)

[Этап 2. Построение моделей на основе деревьев решений и оценка значимости признаков: 5](#_Toc184928622)

[Этап 3. Визуализация значимости признаков: 7](#_Toc184928623)

[Этап 4. Сравнение качества моделей: 11](#_Toc184928624)

[Этап 5. Снижение размерности и визуализация данных: 14](#_Toc184928625)

[Результат работы: 17](#_Toc184928626)

[Вывод: 17](#_Toc184928627)

[Список использованных источников и литературы: 18](#_Toc184928628)

ПРАКТИЧЕСКАЯ РАБОТА 7

Задание

1. Выбор набора данных.

* Выберите один из предложенных датасетов или другой подходящий набор данных с большим количеством признаков.

1. Построение моделей и оценка значимости признаков:

* Построение линейной модели:
* Использовать критерий значимости признаков (p-value или веса).
* Применить жадный алгоритм отбора признаков.
* Модель с Lasso-регуляризацией:
* Оценить влияние параметра alpha на зануление признаков.
* Построение моделей на основе деревьев решений:
* Случайный лес.
* Градиентный бустинг.
* Вывести метрики качества моделей.
* Оценка значимости признаков:
* Использовать встроенные методы моделей (например, feature\_importances\_).
* Применить SHAP-значения для интерпретации. Визуализация иерархической кластеризации.

1. Визуализация значимости признаков:

* Визуализировать отсортированную значимость признаков для каждого алгоритма.

1. Сравнение качества моделей:

* Оценить качество моделей:
* На всех признаках.
* На основе наиболее значимых признаков.

1. Снижение размерности и визуализация данных:

* Применить метод PCA, обучить модель на преобразованных

признаках.

* Визуализировать данные с помощью t-SNE или UMAP.

Шаги выполнения

Этап 1. Подготовка данных:

1. Найти и загрузить набор данных для задачи классификации. Из рекомендации я взял «Классификация курильщиков»

Листинг 1 – Загрузка данных

|  |
| --- |
| import pandas as pd  f = ('/content/smoking-drinking-dataset/smoking\_driking\_dataset\_Ver01.csv')  data = pd.read\_csv(f)  data |

1. Провести необходимые шаги по предобработке данных:

* Провести очистку данных на повторяющие данные и пропущенные данные.
* Заменяем значения в столбцах "sex" и "DRK\_YN".

Листинг 2 – Очистка данных

|  |
| --- |
| data = data.drop\_duplicates()  data = data.dropna()  data['sex'] = data['sex'].replace({'Male': 1, 'Female': 0})  data['DRK\_YN'] = data['DRK\_YN'].replace({'Y': 1, 'N': 0}) |

1. Провести нормализацию данных:

Листинг 3 – Нормализация данных

|  |
| --- |
| from sklearn.preprocessing import MinMaxScaler  smaller\_dataset = data.sample(n=20000, random\_state=42)  columns\_to\_scale = smaller\_dataset.columns.difference(["DRK\_YN", "SMK\_stat\_type\_cd"])  scaler = MinMaxScaler()  scaled\_data = scaler.fit\_transform(smaller\_dataset[columns\_to\_scale])  scaled\_df = pd.DataFrame(scaled\_data, columns=columns\_to\_scale)  scaled\_dataset = smaller\_dataset.copy()  scaled\_dataset = scaled\_dataset.reset\_index(drop=True)  scaled\_dataset[columns\_to\_scale] = scaled\_df |

Этап 2. Построение моделей на основе деревьев решений и оценка значимости признаков:

1. Случайный лес.

Листинг 4 – Код реализации случайного леса

|  |
| --- |
| from sklearn.ensemble import RandomForestClassifier  from sklearn.model\_selection import train\_test\_split  # Загрузка данных  X = scaled\_dataset.iloc[:,:-1].values  y = scaled\_dataset.iloc[:,-1:].values  # Разделение на обучающие и тестовые данные  X\_train, X\_test, y\_train, y\_test = train\_test\_split(X, y, test\_size=0.2, random\_state=42)  clf = RandomForestClassifier(n\_estimators=25)  clf.fit(X\_train, y\_train)  y\_pred = clf.predict(X\_test) |

1. Градиентный бустинг

Листинг 5 – Код реализации градиентного бустинга

|  |
| --- |
| from sklearn.ensemble import GradientBoostingClassifier  from sklearn.model\_selection import train\_test\_split  from sklearn.metrics import accuracy\_score  X = scaled\_dataset.iloc[:,:-1].values  y = scaled\_dataset.iloc[:,-1:].values  X\_train, X\_test, y\_train, y\_test = train\_test\_split(X, y, test\_size=0.2, random\_state=42)  gbc = GradientBoostingClassifier(n\_estimators=300,                                   learning\_rate=0.05,                                   random\_state=100,                                   max\_features=5)  gbc.fit(X\_train, y\_train)  y\_pred = gbc.predict(X\_test) |

1. Метрики качества моделей

Листинг 6 – Код реализации метриков

|  |
| --- |
| from sklearn.metrics import accuracy\_score, confusion\_matrix, precision\_score, recall\_score, f1\_score, roc\_curve, roc\_auc\_score  # Метрики случайного леса  a = accuracy\_score(y\_test, y\_pred)  precision1 = precision\_score(y\_test, y\_pred, average='macro')  recall1 = recall\_score(y\_test, y\_pred, average='macro')  f11 = f1\_score(y\_test, y\_pred, average='macro')  roc\_auc1 = roc\_auc\_score(y\_test, y\_pred)  print("Accuracy:", a)  print("Точность (Precision):", precision1)  print("Полнота (Recall):", recall1)  print("F1-score:", f11)  print("ROC-AUC:", roc\_auc1)  # Матрица ошибок  cm1 = confusion\_matrix(y\_test, y\_pred)  print(f"Матрица ошибок:\n {cm1}") |

Листинг 7 – Код реализации метриков

|  |
| --- |
| from sklearn.metrics import accuracy\_score, confusion\_matrix, precision\_score, recall\_score, f1\_score, roc\_curve, roc\_auc\_score  # Метрики случайного леса  a = accuracy\_score(y\_test, y\_pred)  precision1 = precision\_score(y\_test, y\_pred, average='macro')  recall1 = recall\_score(y\_test, y\_pred, average='macro')  f11 = f1\_score(y\_test, y\_pred, average='macro')  roc\_auc1 = roc\_auc\_score(y\_test, y\_pred)  print("Accuracy:", a)  print("Точность (Precision):", precision1)  print("Полнота (Recall):", recall1)  print("F1-score:", f11)  print("ROC-AUC:", roc\_auc1)  # Матрица ошибок  cm1 = confusion\_matrix(y\_test, y\_pred)  print(f"Матрица ошибок:\n {cm1}") |

1. SHAP-значения

Листинг 8 – Код реализации SHAP-значения

|  |
| --- |
| import shap  # Создание объяснителя для модели Random Forest  explainer\_rf = shap.Explainer(clf)  shap\_values\_rf = explainer\_rf(X\_test)  # Создание объяснителя для модели Gradient Boosting  explainer\_gb = shap.Explainer(gbc)  shap\_values\_gb = explainer\_gb(X\_test) |

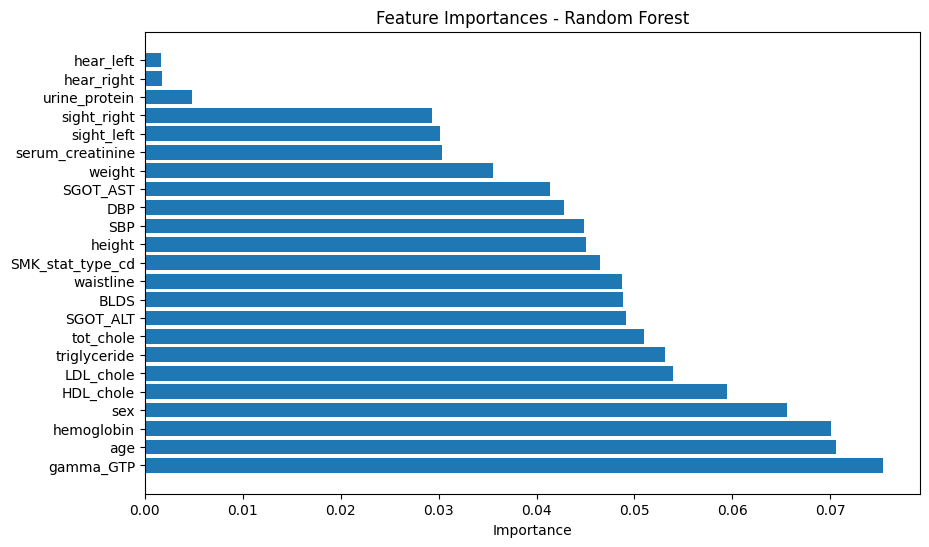
Этап 3. Визуализация значимости признаков:

1. Гистограммы и столбчатые диаграммы

Выявляем наиболее и наименее важные признаки.

Листинг 9 – Код визуализации

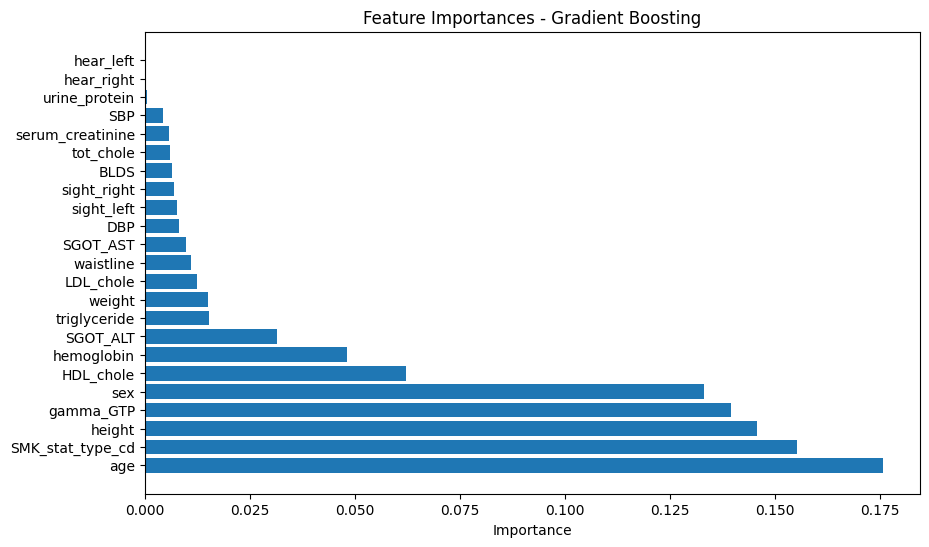
|  |
| --- |
| # Визуализация значимости для случайного леса  import matplotlib.pyplot as plt  import numpy as np  X1 = scaled\_dataset.drop('DRK\_YN', axis=1)  im = clf.feature\_importances\_  fea = X1.columns  ind = np.argsort(im)[::-1]  sor\_im = im[ind]  sor\_fea = fea[ind]  plt.figure(figsize=(10, 6))  plt.barh(sor\_fea, sor\_im)  plt.title('Feature Importances - Random Forest')  plt.xlabel('Importance')  plt.show() |



**Рисунок 1 – График важных характеристик случайного леса**

Листинг 10 – Код визуализации

|  |
| --- |
| # Визуализация значимости для случайного леса  import matplotlib.pyplot as plt  import numpy as np  X1 = scaled\_dataset.drop('DRK\_YN', axis=1)  im = clf.feature\_importances\_  fea = X1.columns  ind = np.argsort(im)[::-1]  sor\_im = im[ind]  sor\_fea = fea[ind]  plt.figure(figsize=(10, 6))  plt.barh(sor\_fea, sor\_im)  plt.title('Feature Importances - Random Forest')  plt.xlabel('Importance')  plt.show() |

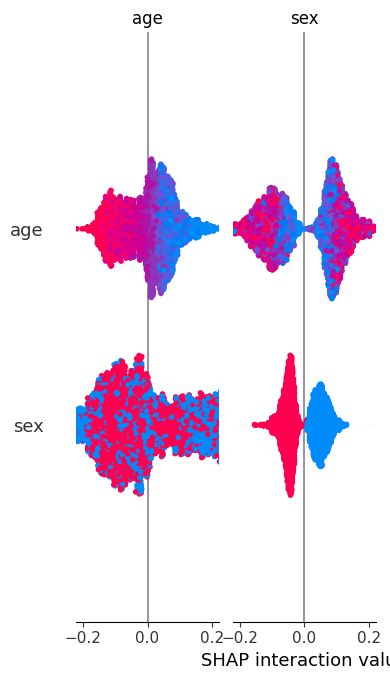


**Рисунок 2 – График важных характеристик градиентного бустинга**

1. Графики SHAP

Листинг 11 – Код визуализации SHAP Summary Plot

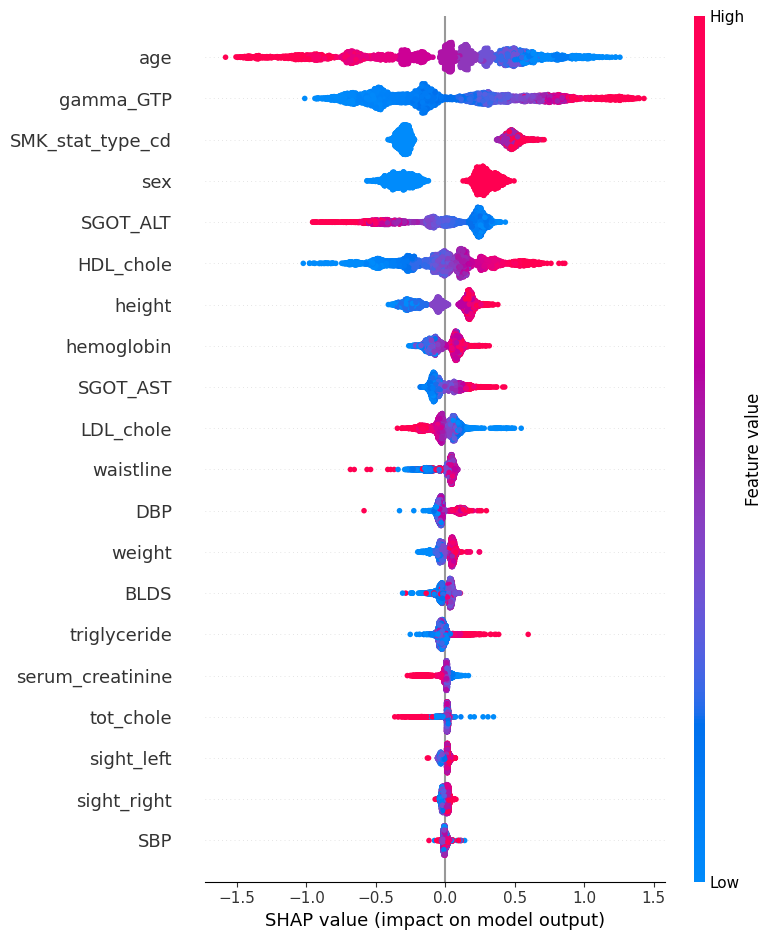
|  |
| --- |
| shap.summary\_plot(shap\_values\_rf, X\_test, scaled\_dataset.columns) |



**Рисунок 3 – Визуализация SHAP значений для случайного леса**

Листинг 12 – Код визуализации SHAP Summary Plot

|  |
| --- |
| shap.summary\_plot(shap\_values\_gb, X\_test, scaled\_dataset.columns) |



**Рисунок 4 – Визуализация SHAP значений для градиентного бустинга**

Этап 4. Сравнение качества моделей:

1. Модели на всех признаках

Листинг 13 – Код для метрики оценки качества модели

|  |
| --- |
| feature\_importances\_rf = clf.feature\_importances\_  feature\_importances\_sorted\_rf = sorted(zip(feature\_importances\_rf, X1.columns), reverse=True)  feature\_importances\_gb = gbc.feature\_importances\_  feature\_importances\_sorted\_gb = sorted(zip(feature\_importances\_gb, X1.columns), reverse=True)  models = {      'Random Forest': {'model': clf, 'score': a},      'Gradient Boosting': {'model': gbc, 'score': acc}  }  for name, info in models.items():      print(f"\n{name}:")      print(f"Accuracy: {info['score']}")      print(f"AUC: {a if name == 'Random Forest' else acc}")      print(f"F1-score: {f11 if name == 'Random Forest' else f12}") |

1. Модели на отобранных признаках:

Листинг 14 – Код для метрики оценки качества на отобранных моделей

|  |
| --- |
| import numpy as np  important\_features = [col for \_, col in feature\_importances\_sorted\_rf[:5]] + \                      [col for \_, col in feature\_importances\_sorted\_gb[:5]]  important\_feature\_indices = np.where(important\_features)[0]  X\_train\_important = X\_train[important\_feature\_indices]  X\_test\_important = X\_test[important\_feature\_indices]  y\_train20=y\_train[:10]  X\_train\_original = X\_train.copy()  y\_test\_original = y\_test.copy()  y\_test\_original20=y\_test\_original[:10]  rf\_model\_important = RandomForestClassifier(random\_state=42)  gb\_model\_important = GradientBoostingClassifier(random\_state=42)  rf\_model\_important.fit(X\_train\_important, y\_train20)  gb\_model\_important.fit(X\_train\_important, y\_train20)  y\_pred\_rf\_important = rf\_model\_important.predict(X\_test\_important)  y\_pred\_gb\_important = gb\_model\_important.predict(X\_test\_important)  accuracy\_rf\_important = accuracy\_score(y\_test\_original20, y\_pred\_rf\_important)  precision\_rf\_important = precision\_score(y\_test\_original20, y\_pred\_rf\_important, average='macro')  recall\_rf\_important = recall\_score(y\_test\_original20, y\_pred\_rf\_important, average='macro')  auc\_rf\_important = roc\_auc\_score(y\_test\_original20, rf\_model\_important.predict\_proba(X\_test\_important)[:, 1])  f1\_rf\_important = f1\_score(y\_test\_original20, y\_pred\_rf\_important)  accuracy\_gb\_important = accuracy\_score(y\_test\_original20, y\_pred\_gb\_important)  precision\_gb\_important = precision\_score(y\_test\_original20, y\_pred\_rf\_important, average='macro')  recall\_gb\_important = recall\_score(y\_test\_original20, y\_pred\_rf\_important, average='macro')  auc\_gb\_important = roc\_auc\_score(y\_test\_original20, gb\_model\_important.predict\_proba(X\_test\_important)[:, 1])  f1\_gb\_important = f1\_score(y\_test\_original20, y\_pred\_gb\_important)  print("Random Forest Accuracy:", accuracy\_rf\_important)  print("Точность (Precision):", precision\_rf\_important)  print("Полнота (Recall):", recall\_rf\_important)  print("Random Forest AUC:", auc\_rf\_important)  print("Random Forest F1-score:", f1\_rf\_important)  print()  print("Gradient Boosting Accuracy:", accuracy\_gb\_important)  print("Точность (Precision):", precision\_gb\_important)  print("Полнота (Recall):", recall\_gb\_important)  print("Gradient Boosting AUC:", auc\_gb\_important)  print("Gradient Boosting F1-score:", f1\_gb\_important) |

Этап 5. Снижение размерности и визуализация данных:

1. Использование PCA

Листинг 15 – Код для использования PCA

|  |
| --- |
| from sklearn.decomposition import PCA  pca = PCA(n\_components=2)  X\_pca = pca.fit\_transform(X\_test)  X\_pca |

1. Использование t-SNE

Листинг 16 – Код для использования t-SNE

|  |
| --- |
| from sklearn.manifold import TSNE  tsne = TSNE(n\_components=2, perplexity=30, random\_state=42)  X\_tsne = tsne.fit\_transform(X\_test)  X\_tsne |

1. Использование UMAP

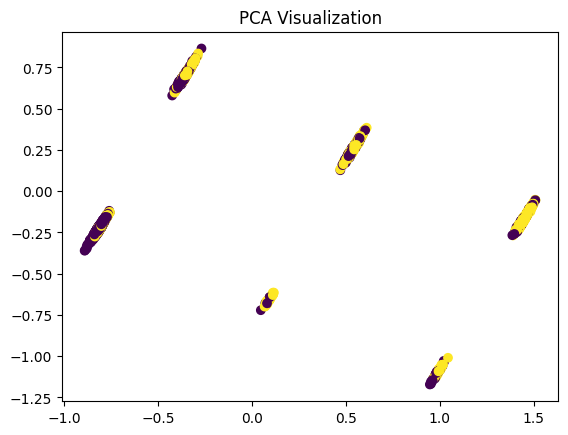
Листинг 17 – Код для использования UMAP

|  |
| --- |
| import umap  umap\_model = umap.UMAP(n\_components=2, random\_state=42)  X\_umap = umap\_model.fit\_transform(X\_test)  X\_umap |

1. Визуализация метода PCA

Листинг 18 – Код визуализации метода PCA

|  |
| --- |
| import matplotlib.pyplot as plt  plt.scatter(X\_pca[:, 0], X\_pca[:, 1], c=y\_test)  plt.title("PCA Visualization")  plt.show() |

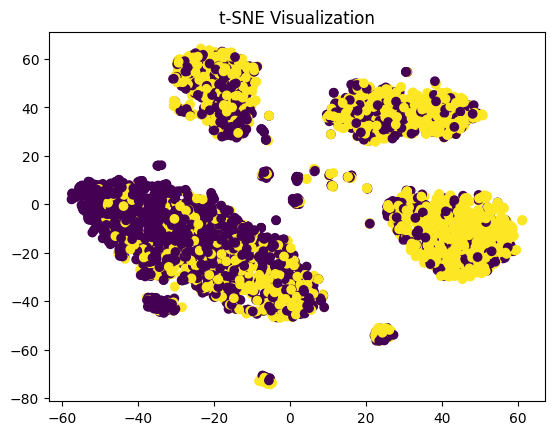


**Рисунок 5 – Визуализация метода PCA**

1. Визуализация метода t-SNE

Листинг 19 – Код визуализации метода t-SNE

|  |
| --- |
| plt.scatter(X\_tsne[:, 0], X\_tsne[:, 1], c=y\_test)  plt.title("t-SNE Visualization")  plt.show() |

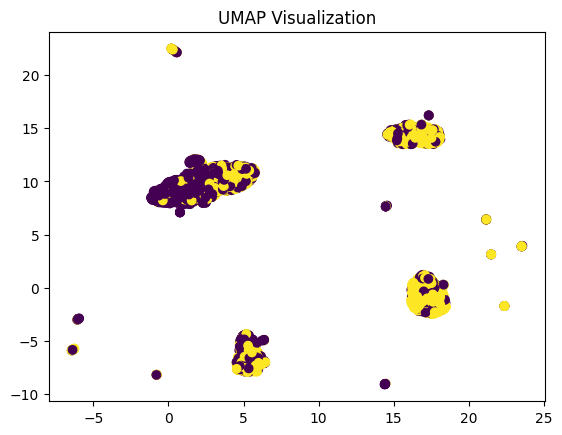


**Рисунок 6 – Визуализация метода t-SNE**

1. Визуализация метода UMAP

Листинг 20 – Код визуализации метода UMAP

|  |
| --- |
| plt.scatter(X\_umap[:, 0], X\_umap[:, 1], c=y\_test)  plt.title('UMAP Visualization')  plt.show() |



**Рисунок 7 – Визуализация метода UMAP**

Результат работы:

Данную работу можете увидеть в блокноте Google Colab: <https://colab.research.google.com/drive/1Ag-L_T0LqEEuEazGNANPArYB4kzZM44P?usp=sharing>

Вывод:

Я освоил метод отбора и оценки значимости признаков, построение и анализ моделей с использованием различных подходов, а также визуализацию данных. Это включает как линейные модели, так и деревья решений, метод главных компонент (PCA), а также современные методы объяснимости моделей, такие как SHAP. Практика направлена на развитие навыков работы с большими наборами данных, интерпретации результатов и визуализации значимости признаков.

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