|  |
| --- |
| 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 № 5**

|  |  |  |  |
| --- | --- | --- | --- |
| 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.

**Цель занятия:**

Освоить методы построения и оценки моделей классификации/регрессии с использованием деревьев решений и ансамблей моделей (стекинг, бэггинг, бустинг) и оценить их производительность.

**Постановка задачи:**

**Этап 1. Подготовка данных:**

1. Найти и загрузить набор данных для задачи классификации.

Листинг 1 – Загрузка данных

|  |
| --- |
| import pandas as pd  f = ('/content/heart-disease-dataset/heart.csv')  data = pd.read\_csv(f) |

1. Провести необходимые шаги по предобработке данных:

* Проверить на наличие пропущенных значений и обработать их.

Листинг 2 – Проверка на наличие пропущенных значений

|  |
| --- |
| # Удаление строк с любыми пропущенными значениями  clean\_data = data.dropna()  # Выведем информацию о данных до и после очистки  print("Размер исходных данных:", data.shape)  print("Размер данных после удаления строк с пропусками:", clean\_data.shape)  # воспользуемся функциями isnull() и sum()  clean\_data.isnull().sum() |

**Этап 2. Построить дерево решений:**

1. Реализуйте дерево решений для задачи классификации или

регрессии с использованием библиотеки scikit-learn.

Листинг 3 – Разделение данных на обучающую и тестовую выборки

|  |
| --- |
| from sklearn import tree  from sklearn.model\_selection import train\_test\_split  # Загрузка данных  X = clean\_data.iloc[:,:-1].values  y = clean\_data.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) |

1. Проведите обучение модели на ваших данных.

Листинг 4 – Обучение модели

|  |
| --- |
| # Создание дерева решений для классификации  clf = tree.DecisionTreeClassifier(max\_depth=3)  # Обучение модели  clf.fit(X\_train, y\_train)  # Прогнозирование  y\_pred = clf.predict(X\_test)  # Оценка точности  from sklearn.metrics import accuracy\_score  print("Accuracy:", accuracy\_score(y\_test, y\_pred))  # Визуализация дерева  import matplotlib.pyplot as plt  plt.figure(figsize=(12,8))  tree.plot\_tree(clf, filled=True)  plt.show() |

**Этап 3. Подобрать гиперпараметры дерева решений:**

1. С помощью GridSearchCV выполните подбор оптимальных гиперпараметров для дерева решений (например, глубина дерева, минимальное количество образцов для разделения узла и т.д.)

Листинг 5 – Подбор гиперпараметра

|  |
| --- |
| from sklearn.model\_selection import GridSearchCV, train\_test\_split  from sklearn.tree import DecisionTreeClassifier  # Загрузка данных  X = clean\_data.iloc[:,:-1].values  y = clean\_data.iloc[:,-1:].values  # Разделение данных на тренировочные и тестовые  X\_train, X\_test, y\_train, y\_test = train\_test\_split(X, y, test\_size=0.2, random\_state=42)  # Определение модели  dt = DecisionTreeClassifier()  # Определение сетки гиперпараметров  param\_grid = {  'max\_depth': [4, 6, 8, 10],  'min\_samples\_split': [3, 5, 10],  'min\_samples\_leaf': [1, 2, 7],  'criterion': ['gini', 'entropy']  }  # Настройка GridSearchCV  grid\_search = GridSearchCV(estimator=dt, param\_grid=param\_grid, cv=5, scoring='accuracy')  # Обучение и подбор гиперпараметров  grid\_search.fit(X\_train, y\_train)  # Лучшие гиперпараметры  best\_params = grid\_search.best\_params\_  print("Лучшие параметры:", best\_params)  # Оценка модели с лучшими параметрами  best\_model = grid\_search.best\_estimator\_  print(best\_model.score(X\_test, y\_test))  best\_dt\_classifier = DecisionTreeClassifier(\*\*best\_params)  best\_dt\_classifier.fit(X\_train, y\_train) |

1. Оцените качество полученной модели с подобранными параметрами с использованием метрик, таких как точность, F1-score, RMSE или R^2 (в зависимости от задачи).

Листинг 6 – Метрика F1-score

|  |
| --- |
| from sklearn.metrics import accuracy\_score, f1\_score  y\_pred = best\_dt\_classifier.predict(X\_test)  accuracy = accuracy\_score(y\_test, y\_pred)  f1 = f1\_score(y\_test, y\_pred)  print("Точность:", accuracy)  print("F1-скор:", f1) |

**Этап 4. Реализовать ансамбли моделей:**

1. Реализовать бэггинг (bagging).

Листинг 7 – Bagging

|  |
| --- |
| from sklearn.model\_selection import train\_test\_split  from sklearn.metrics import accuracy\_score  from sklearn.tree import DecisionTreeClassifier  from sklearn.ensemble import BaggingClassifier    # Загрузка данных  X = clean\_data.iloc[:,:-1].values  y = clean\_data.iloc[:,-1:].values  # Разделение данных на тренировочные и тестовые  X\_train, X\_test, y\_train, y\_test = train\_test\_split(X, y, test\_size=0.2, random\_state=42)  # Создаем классификатор  bagging\_classifier = BaggingClassifier(n\_estimators=10) # Количество базовых моделей (итераций)  # Обучаем классификатора  bagging\_classifier.fit(X\_train, y\_train) |

1. Реализовать бустинг (XGBoost).

Листинг 8 – XGBoost

|  |
| --- |
| import xgboost as xgb  from sklearn.model\_selection import train\_test\_split  # Загрузка данных  X = clean\_data.iloc[:,:-1].values  y = clean\_data.iloc[:,-1:].values  X\_train, X\_test, y\_train, y\_test = train\_test\_split(X, y, test\_size=0.2, random\_state=42)  # Инициализация и обучение XGBoost  xg\_clf = xgb.XGBClassifier()  xg\_clf.fit(X\_train, y\_train) |

1. Реализовать стекинг (stacking).

Листинг 9 – Stacking

|  |
| --- |
| from sklearn.ensemble import StackingClassifier  from sklearn.linear\_model import LogisticRegression  from sklearn.tree import DecisionTreeClassifier  from sklearn.ensemble import RandomForestClassifier  from sklearn.model\_selection import train\_test\_split  # Загрузка данных  X = clean\_data.iloc[:,:-1].values  y = clean\_data.iloc[:,-1:].values  X\_train, X\_test, y\_train, y\_test = train\_test\_split(X, y, test\_size=0.2, random\_state=42)  # Определение базовых моделей  base\_estimators = [  ('rf', RandomForestClassifier(n\_estimators=100, random\_state=42)),  ('dt', DecisionTreeClassifier(max\_depth=3, random\_state=42))  ]  # Метамодель  final\_estimator = LogisticRegression()  # Создание стекинг-классификатора  stacking\_clf = StackingClassifier(estimators=base\_estimators, final\_estimator=final\_estimator)  # Обучение и оценка модели  stacking\_clf.fit(X\_train, y\_train) |

**Этап 5. Оценить качество моделей:**

Оцените качество ансамблей моделей

Листинг 10 – Оценка качество ансамблей (bagging)

|  |
| --- |
| from sklearn.metrics import accuracy\_score, confusion\_matrix, precision\_score, recall\_score, f1\_score, roc\_auc\_score  # Предсказания модели  y\_pred = bagging\_classifier.predict(X\_test)  # Метрики  accuracy = accuracy\_score(y\_test, y\_pred)  precision = precision\_score(y\_test, y\_pred, average='macro')  recall = recall\_score(y\_test, y\_pred, average='macro')  f1 = f1\_score(y\_test, y\_pred, average='macro')  print(f"Точность: {accuracy}")  print(f"Точность (Precision): {precision}")  print(f"Полнота (Recall): {recall}")  print(f"F1-score: {f1}")  # Матрица ошибок  cm = confusion\_matrix(y\_test, y\_pred)  print(f"Матрица ошибок:\n {cm}") |

Листинг 11 – Оценка качество ансамблей (boosting)

|  |
| --- |
| from sklearn.metrics import accuracy\_score, confusion\_matrix, precision\_score, recall\_score, f1\_score, roc\_auc\_score  # Предсказания модели  y\_pred = xg\_clf.predict(X\_test)  # Метрики  accuracy = accuracy\_score(y\_test, y\_pred)  precision = precision\_score(y\_test, y\_pred, average='macro')  recall = recall\_score(y\_test, y\_pred, average='macro')  f1 = f1\_score(y\_test, y\_pred, average='macro')  print(f"Точность: {accuracy}")  print(f"Точность (Precision): {precision}")  print(f"Полнота (Recall): {recall}")  print(f"F1-score: {f1}")  # Матрица ошибок  cm = confusion\_matrix(y\_test, y\_pred)  print(f"Матрица ошибок:\n {cm}") |

Листинг 12 – Оценка качество ансамблей (stacking)

|  |
| --- |
| from sklearn.metrics import accuracy\_score, confusion\_matrix, precision\_score, recall\_score, f1\_score, roc\_auc\_score  # Предсказания модели  y\_pred = stacking\_clf.predict(X\_test)  # Метрики  accuracy = accuracy\_score(y\_test, y\_pred)  precision = precision\_score(y\_test, y\_pred, average='macro')  recall = recall\_score(y\_test, y\_pred, average='macro')  f1 = f1\_score(y\_test, y\_pred, average='macro')  print(f"Точность: {accuracy}")  print(f"Точность (Precision): {precision}")  print(f"Полнота (Recall): {recall}")  print(f"F1-score: {f1}")  # Матрица ошибок  cm = confusion\_matrix(y\_test, y\_pred)  print(f"Матрица ошибок:\n {cm}") |

**Результат работы:**

Данную работу можете увидеть в блокноте Google Colab:

<https://colab.research.google.com/drive/12TAd0lMN2f1AIpCVfLQgElHCyD_MOWvB?usp=sharing>

**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