МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ  
РОССИЙСКОЙ ФЕДЕРАЦИИ

ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ БЮДЖЕТНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ  
«ВЯТСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ»

Институт математики и информационных систем

Факультет автоматики и вычислительной техники

Кафедра систем автоматизации управления

**Композиции алгоритмов**

Отчет по лабораторной работе №3  
по дисциплине

«Большие данные»

Выполнил студент гр. ИТб-4302-02-00 \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ /Мартьянов М.С./

(Подпись)

Руководитель ст. преподаватель \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ /Родионов К.В./

(Подпись)

Работа защищена с оценкой «\_\_\_\_\_\_\_\_\_\_\_» «\_\_\_» \_\_\_\_\_\_\_\_\_\_ 2023 г.

Цель: научиться работать со случайным лесом, подбирать его параметры и решать с его помощью задачи регрессии; научиться работать с градиентным бустингом и подбирать его гиперпараметры, а также сравнивать разные способы построения композиций, научиться использовать метрику log-loss.

В ходе работы необходимо выполнить следующее задание:

* загрузить данные;
* обучить модели RandomForest и GradientBoosting;
* применить RF к данным и провести перебор деревьев;
* определить, при каком минимальном количестве деревьев случайный лес показывает качество на кросс-валидации выше 0,52.

Для начала загружаем данные из файла «abalone.csv» o физических измерениях ракушек для того, чтобы предсказать их возраст (число колец). Чтобы загрузить данные необходимо выполнить следующий код:

|  |
| --- |
| import pandas as pd  data = pd.read\_csv('abalone.csv') |

Преобразуем половой признак «Sex» в числовое значение: F должно перейти в -1, I – в 0, M – в 1. Чтобы преобразовать функцию используем следующий код:

|  |
| --- |
| data['Sex'] = data['Sex'].map({'M': 1,'F': -1,'I': 0}) |

Разделяем загруженные данные на признаки и целевое значение. Чтобы выполнить разделение используем следующий код:

|  |
| --- |
| X = data.iloc[:, :-1]  y = data.iloc[:, -1] |

Случайный лес – это ансамблевый алгоритм, который объединяет множество деревьев решений, обученных на разных подвыборках данных. Ключевые преимущества случайного леса:

* высокая точность за счет усреднения прогнозов отдельных деревьев;
* устойчивость к переобучению при увеличении количества деревьев;
* возможность оценить важность признаков на основе использования их в деревьях.

Создаем модель случайного леса и проводим перебор количества деревьев от 1 до 50. Для каждого числа деревьев вычисляем коэффициент детерминации "R2" с использованием кросс-валидации.

|  |
| --- |
| from sklearn.ensemble import RandomForestRegressor  from sklearn.model\_selection import cross\_val\_score  from sklearn.model\_selection import KFold  import matplotlib.pyplot as plt  r2\_scores = []  for n\_estimators in range(1, 51):  clf = RandomForestRegressor(n\_estimators=n\_estimators, random\_state=1)  cv = KFold(n\_splits=5, shuffle=True, random\_state=1)  r2 = cross\_val\_score(clf, X, y, cv=cv, scoring='r2')  mean\_r2 = np.mean(r2)  r2\_scores.append(mean\_r2)  print(f"n\_estimators = {n\_estimators}, R2 = {mean\_r2}") |

Определяем минимальное количество деревьев, где случайный лес показывает качество кросс-валидации выше 0,52:

|  |
| --- |
| min\_estimators = next(i for i, r2 in enumerate(r2\_scores) if r2 > 0.52)  print(f"Минимальное количество деревьев с R2 > 0.52: {min\_estimators}") |

Для определения изменения качества модели при изменении количества деревьев построили график (рисунок 1).

Изображение выглядит как текст, снимок экрана, линия, диаграмма

Автоматически созданное описание

Рисунок 1 – График изменений качества случайного леса

На рисунке 1 показано, что после точки перегиба между пятью и шестью деревьями модель растет медленнее.

На рисунке 2 предоставлен журнал выполнения программы.

Изображение выглядит как текст, Шрифт, снимок экрана, число

Автоматически созданное описание

Рисунок 2 – Результат выполнения программы

Из результатов выполнения программы видно, что минимальное количество деревьев с «R2» больше 0,52 – 21 дерево.

Переходим ко второй части лабораторной работы, для начала загружаем данные из файла «gbm-data.csv». В первой колонке записано, была или нет реакция. Остальные колонки содержат характеристики молекул. Чтобы загрузить данные необходимо выполнить следующий код:

|  |
| --- |
| import pandas as pd  import numpy as np  from sklearn.model\_selection import train\_test\_split  data = pd.read\_csv('gbm-data.csv')  X = data.iloc[:, 1:].values  y = data.iloc[:, 0].values  X\_train, X\_test, y\_train, y\_test =  train\_test\_split(X, y, test\_size=0.8, random\_state=241) |

Выборка разбивается на обучающую и тестовую в соотношении 80 на 20 соответственно.

Градиентный бустинг – это пошаговый алгоритм, который обучает слабые модели (чаще деревья решений) последовательно, учитывая ошибки предыдущей модели. Преимущества модели:

* высокая точность за счет композиции слабых моделей;
* хорошо находит сложные зависимости в данных;
* менее склонен к переобучению по сравнению с другими алгоритмами.

В качестве функции потерь используется «log-loss», представленная в формуле (1):

|  |  |  |
| --- | --- | --- |
|  | . | (1) |

Обучаем GradientBoostingClassifier с разными значениями параметра «learning\_rate» и фиксированными параметрами «n\_estimators=250», «verbose=True» и «random\_state=241». Для этого необходимо выполнить следующий код:

|  |
| --- |
| from sklearn.ensemble import GradientBoostingClassifier  from sklearn.metrics import log\_loss  test\_loss = []  learning\_rates = [1, 0.5, 0.3, 0.2, 0.1]  n\_estimators = 250  for lr in learning\_rates:  clf = GradientBoostingClassifier(n\_estimators=n\_estimators, learning\_rate=lr, verbose=True, random\_state=241)  clf.fit(X\_train, y\_train)  y\_pred = clf.staged\_decision\_function(X\_test)  test\_loss\_lr = [log\_loss(y\_test, 1 / (1 + np.exp(-y\_pred\_i))) for y\_pred\_i in y\_pred]  test\_loss.append(test\_loss\_lr) |

Необходимо построить график «log-loss» на тестовой выборке, для этого необходимо выполнить следующий код:

|  |
| --- |
| import matplotlib.pyplot as plt  min\_loss\_value = float('inf')  min\_loss\_iter = 0  for lr, loss in zip(learning\_rates, test\_loss):  plt.plot(range(1, n\_estimators + 1), loss, label=f'learning\_rate={lr}')  min\_lr\_loss = min(loss)  if min\_lr\_loss < min\_loss\_value:  min\_loss\_value = min\_lr\_loss  min\_loss\_iter = np.argmin(loss) + 1  plt.legend(loc='upper left')  plt.xlabel('Iteration')  plt.ylabel('Log-Loss')  plt.title('Gradient Boosting Log-Loss')  plt.show() |

На рисунке 3 предоставлен график log-loss на тестовой выборке.

Изображение выглядит как текст, линия, График, диаграмма

Автоматически созданное описание

Рисунок 3 – Результат выполнения программы

На графике показано, как значения функции потерь «loss» зависят от числа итераций «n\_estimators» для градиентного бустинга с разными значениями скорости обучения.

Для каждого значения «learning\_rate» на графике создается отдельная линия. В цикле строится точка графика для каждого значения «lr» от «learning\_rates» и соответствующих значений «loss» от «test\_loss».

Кроме того, «min\_loss\_iter» и «min\_loss\_value» используются для поиска минимального значения функции потерь.

На рисунке 4 представлен результат выполнения программы.

Изображение выглядит как текст, снимок экрана, меню, Шрифт

Автоматически созданное описание

Рисунок 4 – Результат выполнения программы

График качества на тестовой выборке можно охарактеризовать как overfitting(переобучение) по нескольким причинам:

* значение «log-loss» на тестовой выборке сначала уменьшается, а затем начинает расти после примерно 100 итераций. Это классический признак переобучения – снижение метрики качества на обучающей выборке и рост на тестовой;
* линии графиков для разных скоростей обучения «learning rate» демонстрируют схожую динамику – спад до определенной точки и последующий рост. Это значит, что переобучение происходит независимо от выбранного «lr»;
* минимальные значения «log-loss» достигаются на разных итерациях для разных «lr», но ни один из графиков не демонстрирует стабильного спада, который бы свидетельствовал об отсутствии переобучения.

Минимальное значение «log-loss» на тестовой выборке и номер итерации, на которой оно достигается, при «learning\_rate» = 0,2 равны 0,53 и 52 соответственно.

Значение «log-loss» на тесте равно 0,54.

На полученных ранее данных обучаем «RandomForestClassifier» с количеством деревьев, равным количеству итераций, на котором достигается наилучшее качество у градиентного бустинга, для этого необходимо выполнить следующий код:

|  |
| --- |
| answer4 = f"{min\_loss\_value:.2f} {min\_loss\_iter}"  rf\_clf = RandomForestClassifier(n\_estimators=min\_loss\_iter, random\_state=241)  rf\_clf.fit(X\_train, y\_train)  rf\_y\_pred = rf\_clf.predict\_proba(X\_test)  rf\_loss = log\_loss(y\_test, rf\_y\_pred) |

В ходе выполнения лабораторной работы были изучены алгоритмы машинного обучения – случайный лес и градиентный бустинг. Научились оптимизировать параметры композиций алгоритмов машинного обучения для решения задач с большими данными.