Министерство образования и науки РФ
федеральное государственное автономное образовательное учреждение
высшего образования
«Омский государственный технический университет»

Факультет информационных технологий и компьютерных систем
Кафедра «Автоматизированные системы обработки информации и управления»

КУРСОВОЙ ПРОЕКТ (РАБОТА)

по дисциплине Динамические языки программирования
на тему Разработка классификатора банковских данных

Пояснительная записка

Шифр проекта (работы)_____

Студента (ки) Дергачева Александра Дмитриевича
Курс 4 Группа ПИН-201
Направление (специальность) 09.03.04 Программная инженерия
Руководитель Старший преподаватель
Кабанов А. А.
Выполнил (а)_____
К защите_____

Проект (работа) защищен (а) с оценкой_____

Омск 2024

Задание

на выполнение курсового проекта

Задача 1. Найти набор данных (датасет) для классификации, удовлетворяющий следующим условиям: более 10 000 строк, более 20 столбцов, разные типы в столбцах, обязательно наличие целевого признака (таргета).

Задача 2. Провести классификацию найденного датасета, методом к-ближайших соседей. В формате Markdown писать пояснения. Объяснить почему были выбраны именно такие гиперпараметры, была ли перекрестная проверка, и т.д.

Задача 3. Провести классификацию найденного датасета, методом машины опорных векторов. В формате Markdown писать пояснения. Объяснить почему были выбраны именно такие гиперпараметры, была ли перекрестная проверка, и т.д.

Задача 4. Провести классификацию найденного датасета, методами линеной и логистической регрессий. В формате Markdown написать пояснения. Объяснить почему были выбраны именно такие гиперпараметры, была ли перекрестная проверка, и т.д.

Задача 5. Провести классификацию найденного датасета, методами наивного Байеса. В формате Markdown написать пояснения. Объяснить почему были выбраны именно такие гиперпараметры, была ли перекрестная проверка, и т.д.

Задача 6. Провести классификацию найденного датасета, методами решающего дерева и случайного леса. В формате Markdown написать пояснения. Объяснить почему были выбраны именно такие гиперпараметры, была ли перекрестная проверка, и т.д.

Задача 7. Провести классификацию найденного датасета, методами CatBoost. В формате Markdown написать пояснения. Объяснить почему были выбраны именно такие гиперпараметры, была ли перекрестная проверка, и т.д.

РЕФЕРАТ

Пояснительная записка 47 с., 20 рис., 10 источников.

ИССЛЕДОВАНИЕ МОДЕЛЕЙ МАШИННОГО ОБУЧЕНИЯ, K-БЛИЖАЙШИХ СОСЕДЕЙ, МЕТОД ОПОРНЫХ ВЕКТОРОВ (SVM), ЛОГИСТИЧЕСКАЯ РЕГРЕССИЯ, НАИВНЫЙ БАЙЕСОВСКИЙ КЛАССИФИКАТОР, МОДЕЛЬ РЕШАЮЩЕГО ДЕРЕВА, СЛУЧАЙНЫЙ ЛЕС, CATBOOST, ОЦЕНКА ПРОИЗВОДИТЕЛЬНОСТИ, СРАВНИТЕЛЬНЫЙ АНАЛИЗ, ГИПЕРПАРАМЕТРЫ МОДЕЛЕЙ, МЕТРИКИ КАЧЕСТВА, ИНТЕРПРЕТИРУЕМОСТЬ МОДЕЛЕЙ

Объектом исследования данной работы являются различные модели машинного обучения, такие как k-ближайших соседей, метод опорных векторов, логистическая регрессия, наивный байесовский классификатор, модель решающего дерева, случайный лес и CatBoost.

Целью работы является использование моделей и проведение сравнительного анализа производительности различных моделей машинного обучения на основе метрик классификации. В процессе исследования будут проведены эксперименты с определением оптимальных гиперпараметров каждой модели, обучены модели на выборке данных, и оценены их показатели эффективности с использованием различных метрик.

Содержание

Введение 7

Метод K-ближайших соседей 8

Теория 8

Задачи 9

Вывод 13

Модель машины опорных векторов 14

Теория 14

Задачи 15

Вывод 17

Линейная регрессия 18

Теория 18

Задачи 20

Вывод 21

Логистическая регрессия 22

Теория 22

Задачи 24

Вывод 25

Модель наивного Байеса 26

Теория 26

Задачи 28

Вывод 30

Модель решающего дерева 31

Теория 31

Задачи 33
Вывод 35
Модель случайного леса 36
Теория 36
Задачи 38
Вывод 40
Модель CatBoost 41
Теория 41
Задачи 43
Вывод 45
Заключение 46
Список используемых источников 47

Введение

Машинное обучение стало незаменимым инструментом для анализа данных и принятия решений в различных областях, от медицины до финансов. В данном отчете мы рассмотрим различные методы машинного обучения, такие как k-ближайших соседей, метод опорных векторов, логистической регрессии, наивного байесовского классификатора, модель решающего дерева, случайный лес и CatBoost.

Каждый из этих методов имеет свои уникальные особенности, преимущества и ограничения, которые мы изучим в рамках данного исследования. Мы также рассмотрим их применение в различных сценариях и задачах, чтобы понять, какой метод лучше всего подходит для конкретной задачи.

Целью этого отчета является обзор основных методов машинного обучения, их применение и сравнение производительности на различных наборах данных. Это позволит нам лучше понять их применимость и эффективность, а также определить наиболее подходящий метод для конкретной задачи.

Метод K-ближайших соседей

Теория

Классификатор k-ближайших соседей (k-nn) — это метод машинного обучения, используемый для принятия решений на основе данных ближайших объектов в пространстве признаков. Он может выполнять как задачи классификации, определяя класс объекта, так и задачи регрессии, предсказывая его значение.

Основные аспекты метода k-nn:

Число соседей (k): это один из важных параметров модели. Определяет количество ближайших соседей, учитываемых при принятии решения. Выбор значения k влияет на точность модели и её обобщающую способность.

Метрики расстояния: для определения ближайших соседей необходимо использовать метрику расстояния. К примеру, Евклидово расстояние, Манхэттенское расстояние или Расстояние Чебышева - каждая из них определяет расстояние между объектами в пространстве признаков. Выбор подходящей метрики зависит от конкретной задачи и природы данных.

Процесс классификации объекта: после определения k ближайших соседей для нового объекта происходит классификация путем голосования большинства соседей. Объект относится к классу, который представлен наибольшим числом ближайших соседей.

k находит применение во многих областях, включая рекомендательные системы, классификацию текстов или изображений, анализ медицинских данных для определения паттернов или диагнозов, прогнозирование финансовых показателей и других задачах, требующих анализа данных. Важно выбирать параметры метода, исходя из специфики данных и требований задачи.

Задачи

Задача. Провести классификацию найденного датасета, методом k-ближайших соседей. В формате Markdown писать пояснения. Объяснить почему были выбраны именно такие гиперпараметры, была ли перекрестная проверка, и т.д.

Ход работы:

Был найден датасет с 23 столбцами и 10127 строками. Информация о датасете представлен на рисунке 1.

```
Количество строк: 10127
Количество столбцов: 24

Типы данных по столбцам:
Attrition_Flag               int64
Customer_Age                 int64
Dependent_count              int64
Education_Level              int64
Income_Category              int64
Card_Category                int64
Months_on_book               int64
Total_Relationship_Count     int64
Months_Inactive_12_mon       int64
Contacts_Count_12_mon        int64
Credit_Limit               float64
Total_Revolving_Bal          int64
Avg_Open_To_Buy            float64
Total_Amt_Chng_Q4_Q1       float64
Total_Trans_Amt              int64
Total_Trans_Ct               int64
Total_Ct_Chng_Q4_Q1        float64
Avg_Utilization_Ratio      float64
c_F                          int64
c_M                          int64
c_Divorced                   int64
c_Married                    int64
c_Single                     int64
c_Unknown                    int64
dtype: object
```

Рисунок 1 – Информация о датасете

На рисунках 2-6 представлен код программы.

```
from sklearn.model_selection import train_test_split
from sklearn.neighbors import KNeighborsClassifier
from sklearn.metrics import confusion_matrix, classification_report
from sklearn.model_selection import GridSearchCV
from sklearn.preprocessing import StandardScaler
```

Рисунок 2 – Подключение необходимых модулей

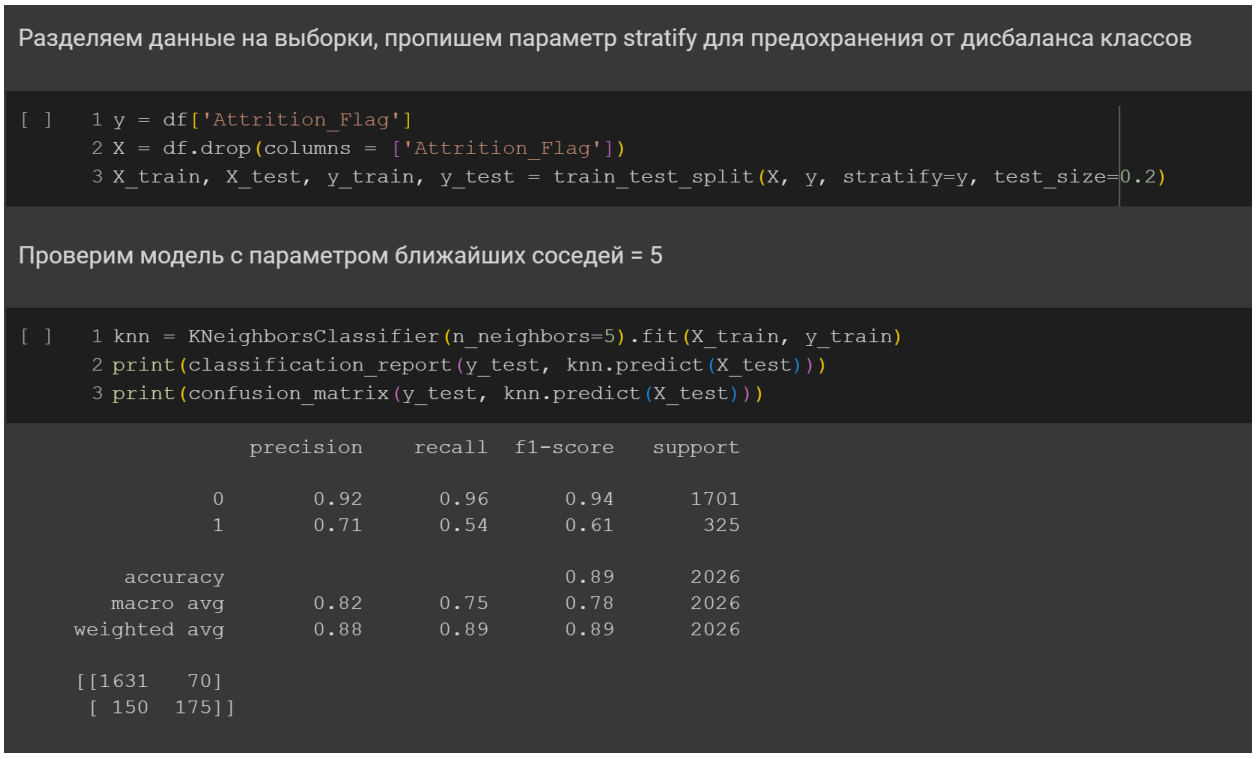

Рисунок 3 – Разделение данных и первичная проверка модели

```
[ ]  1 parameters = {'n_neighbors': [1, 2, 3, 4, 5, 6, 7, 8, 9]}
     2
     3 knn_optimal = GridSearchCV(KNeighborsClassifier(), parameters).fit(X_train, y_train)
     4 knn_optimal.best_params_
```

```
{'n_neighbors': 7}
```

```
[ ]  1 print(classification_report(y_test, knn_optimal.predict(X_test)))
     2 print(confusion_matrix(y_test, knn_optimal.predict(X_test)))
```

```
                  precision    recall  f1-score   support

               0       0.92      0.96      0.94      1701
               1       0.74      0.55      0.63       325

        accuracy                           0.90      2026
       macro avg       0.83      0.76      0.79      2026
    weighted avg       0.89      0.90      0.89      2026

    [[1638   63]
     [ 146  179]]
```

Рисунок 4 – Подбор параметров и повторное обучение модели

```
[ ]  1 scaler = StandardScaler()
     2 X_train = scaler.fit_transform(X_train)
     3 X_test = scaler.transform(X_test)
     4 parameters = {'n_neighbors': [3, 5, 7, 9]}
     5
     6 knn_optimal = GridSearchCV(KNeighborsClassifier(), parameters).fit(X_train, y_train)
     7 knn_optimal.best_params_
```

```
{'n_neighbors': 5}
```

```
[ ]  1 print(classification_report(y_test, knn_optimal.predict(X_test)))
     2 print(confusion_matrix(y_test, knn_optimal.predict(X_test)))
```

```
                  precision    recall  f1-score   support

               0       0.91      0.98      0.94      1701
               1       0.83      0.49      0.62       325

        accuracy                           0.90      2026
       macro avg       0.87      0.74      0.78      2026
    weighted avg       0.90      0.90      0.89      2026

    [[1669   32]
     [ 166  159]]
```

Рисунок 5 – Скалирование данных, подбор параметров и повторное обучение модели

```
              precision    recall  f1-score   support

           0       0.91      0.98      0.94      1701
           1       0.83      0.49      0.62       325

    accuracy                           0.90      2026
   macro avg       0.87      0.74      0.78      2026
weighted avg       0.90      0.90      0.89      2026

[[1669   32]
 [ 166  159]]
```

Рисунок 6 – Результаты

По результатам обучения модели k-ближайших соседей на датасете отеля, где изучается параметр Attrition_Flag, можно сделать следующие выводы. Модель достигла высокой точности в 90%, что свидетельствует о ее хорошей способности классифицировать данные. Однако, стоит отметить, что метрика recall для класса 1 довольно низкая.

Вывод

Таким образом, модель k-ближайших соседей показывает хорошие результаты в обработке классов с несбалансированным распределением. Для улучшения модели можно попробовать балансировку классов или использование других методов классификации, таких как случайный лес или градиентный бустинг, чтобы улучшить ее способность предсказывать оба класса более точно.

Модель машины опорных векторов

Теория

Модель опорных векторов (SVM) — это алгоритм машинного обучения, который используется как для задач классификации, так и для регрессии. Основная идея SVM заключается в поиске оптимальной разделяющей гиперплоскости, которая максимально отделяет два класса данных.

Теория модели опорных векторов в основном включает в себя следующие концепции:

Разделяющая гиперплоскость: SVM строит разделяющую гиперплоскость в n-мерном пространстве, где n - количество признаков. Для линейно разделимых данных гиперплоскость строится таким образом, чтобы максимизировать зазор между классами.

Опорные вектора: Опорные вектора — это наблюдения, которые лежат на границе зазора и определяют положение разделяющей гиперплоскости. Эти вектора играют важную роль в определении гиперплоскости.

Ядерные функции: SVM может использовать ядерные функции для перехода в пространство более высокой размерности, что позволяет разделить данные, которые не являются линейно разделимыми в исходном пространстве.

Оптимизация зазора: Целью SVM является максимизация ширины зазора между классами, что обеспечивает более точную классификацию новых данных.

Регуляризация: SVM имеет встроенную регуляризацию, которая позволяет управлять сложностью модели и предотвращать переобучение.

Эти концепции являются основой для понимания работы модели опорных векторов и её применения в задачах машинного обучения, включая классификацию, регрессию и определение функций ранжирования.

Задачи

Задача. Провести классификацию найденного датасета, методом машины опорных векторов. В формате Markdown писать пояснения. Объяснить почему были выбраны именно такие гиперпараметры, была ли перекрестная проверка, и т.д.

Ход работы:

```
1 from sklearn.model_selection import train_test_split, GridSearchCV
2 from sklearn.preprocessing import StandardScaler
3 from sklearn.svm import SVC
4 from sklearn.metrics import accuracy_score, classification_report, confusion_matrix
```

Рисунок 7 – Подключение необходимых модулей

На рисунках 8-9 представлен код программы.

```
[5]  1 y = df['Attrition_Flag']
     2 X = df.drop(columns = ['Attrition_Flag'])
     3 X_train, X_test, y_train, y_test = train_test_split(X, y, stratify=y, test_size=0.2)
```

Попробуем обучить модель

```
1 svc = SVC (C = 5, gamma = 0.5)
2 svc.fit(X_train, y_train)
3
4 print(classification_report(y_test, svc.predict(X_test)))
5 print(confusion_matrix(y_test, svc.predict(X_test)))
```

```
/usr/local/lib/python3.10/dist-packages/sklearn/metrics/_classification.py:1344: Undefined
  _warn_prf(average, modifier, msg_start, len(result))
/usr/local/lib/python3.10/dist-packages/sklearn/metrics/_classification.py:1344: Undefined
  _warn_prf(average, modifier, msg_start, len(result))
/usr/local/lib/python3.10/dist-packages/sklearn/metrics/_classification.py:1344: Undefined
  _warn_prf(average, modifier, msg_start, len(result))
              precision    recall  f1-score   support

           0       0.84      1.00      0.91      1701
           1       0.00      0.00      0.00       325

    accuracy                           0.84      2026
   macro avg       0.42      0.50      0.46      2026
weighted avg       0.70      0.84      0.77      2026

[[1701    0]
 [ 325    0]]
```

Рисунок 8 – Первичное обучение модели

На данном этапе модель ничего не предсказывает, необходимо подобрать параметры.

```
scaler = StandardScaler()
X_train_scaled = scaler.fit_transform(X_train)
X_test_scaled = scaler.transform(X_test)

# Поиск оптимальных параметров с помощью кросс-валидации
param_grid = {'C': [0.1, 1, 10], 'gamma': [0.1, 1, 10]}
grid_search = GridSearchCV(SVC(), param_grid, cv=10)
grid_search.fit(X_train_scaled, y_train)

# Обучение модели с наилучшими параметрами
best_C = grid_search.best_params_['C']
best_gamma = grid_search.best_params_['gamma']
best_svc = SVC(C=best_C, gamma=best_gamma)
best_svc.fit(X_train_scaled, y_train)

# Предсказание на тестовом наборе
y_pred = best_svc.predict(X_test_scaled)

print(classification_report(y_test, y_pred))
print(confusion_matrix(y_test, y_pred))
```

```
Наилучшие параметры: {'C': 10, 'gamma': 0.1}
Точность модели: 0.9279368213228035
              precision    recall  f1-score   support

           0       0.94      0.97      0.96      1701
           1       0.83      0.69      0.75       325

    accuracy                           0.93      2026
   macro avg       0.89      0.83      0.86      2026
weighted avg       0.92      0.93      0.93      2026

[[1656   45]
 [ 101  224]]
```

Рисунок 9 – Скалирование данных, подбор параметров и результат обучения

Вывод

Исследование модели опорных векторов для предсказания признака Attrition_Flag в датасете дало неплохие результаты. Наилучшие параметры привели к точности модели 93%. Это означает, что модель способна с высокой точностью предсказывать вернет ли клиент кредит. Такие высокие показатели точности говорят о том, что модель опорных векторов хорошо подходит для данной задачи классификации и может быть использована для прогнозирования с высокой достоверностью. Минусом является долгое время обучения модели.

Линейная регрессия

Теория

Линейная регрессия — это один из наиболее простых и широко используемых методов машинного обучения, который применяется для предсказания значений переменной на основе линейной зависимости между набором признаков и целевой переменной. Основная идея линейной регрессии заключается в том, чтобы найти линейную функцию, которая наилучшим образом описывает отношения между независимыми и зависимой переменными.

Основные концепции линейной регрессии:

Линейная зависимость:

Линейная регрессия предполагает, что зависимость между предсказываемой переменной (целевой) и независимыми признаками может быть аппроксимирована линейной функцией. Формула линейной регрессии имеет следующий вид:

$y = w0 + w1*x1 + w2*x2 + ... + wn*xn + \varepsilon$, где:

y - целевая переменная, которую мы пытаемся предсказать. x1, x2, ..., xn - независимые признаки (факторы).

w0, w1, ..., wn - коэффициенты модели (веса), которые определяют веса каждого признака.

ε - ошибка модели (шум, который не удается объяснить моделью).

Метод наименьших квадратов (Ordinary Least Squares, OLS): Основной метод оценки параметров модели линейной регрессии -

метод наименьших квадратов (OLS). Цель состоит в том, чтобы найти такие значения коэффициентов w0, w1, ..., wn, которые минимизируют сумму квадратов отклонений (расхождений) между фактическими и предсказанными значениями целевой переменной.

Оценка качества модели:

Для оценки качества предсказаний модели линейной регрессии используют различные метрики, включая:

Средняя квадратическая ошибка (Mean Squared Error, MSE): Средняя ошибка между фактическими и предсказанными значениями, возведенная в квадрат.

Коэффициент детерминации (Coefficient of Determination, R^2): показывает, какую часть дисперсии целевой переменной объясняет модель.

Множественная линейная регрессия:

Множественная линейная регрессия включает более одной независимой переменной. В этом случае модель будет иметь вид:

$y = w0 + w1*x1 + w2*x2 + ... + wnxn + \varepsilon,$

где x1, x2, ..., xn - это несколько независимых переменных.

Задачи

Задача. Провести классификацию найденного датасета, методом линейной регрессии. В формате Markdown написать пояснения. Объяснить почему были выбраны именно такие гиперпараметры, была ли перекрестная проверка, и т.д.

Ход работы:

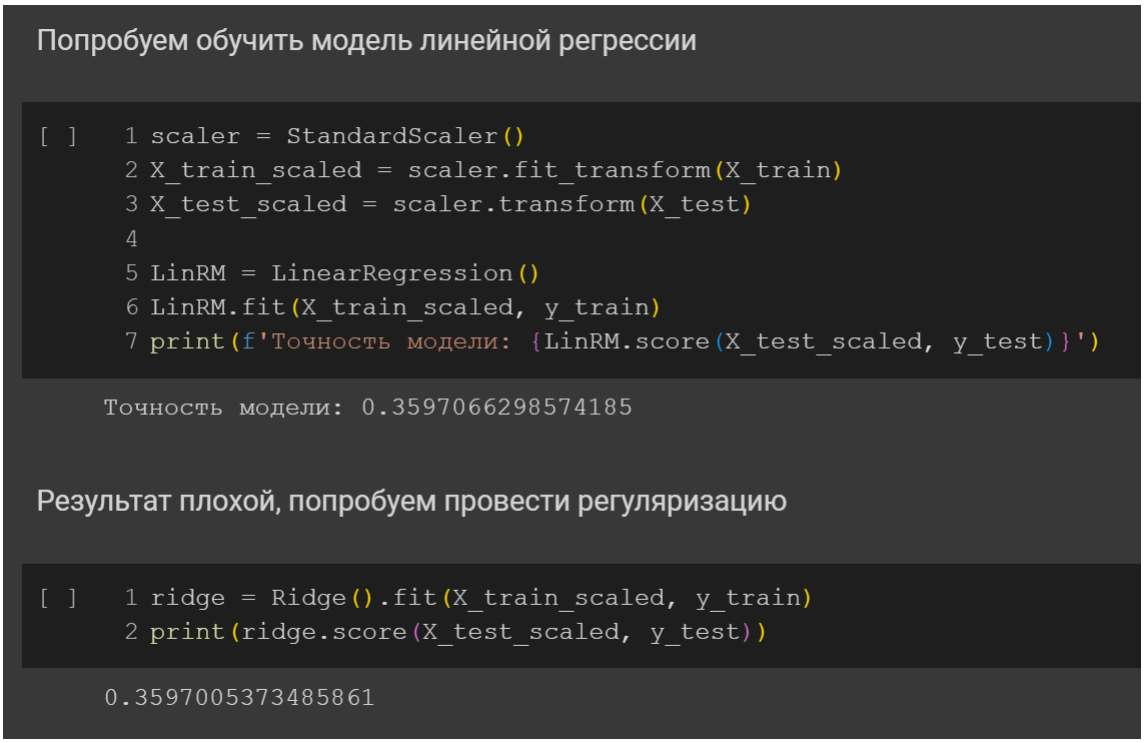

Рисунок 10 – Скалирование данных и обучение модели

Вывод

Обучение модели линейной регрессии для предсказания параметра Attrition_Flag в датасете дало следующие результаты: коэффициент детерминации - 0.36.

Данная модель линейной регрессии показывает плохую предсказательную способность, так как коэффициент детерминации ниже 0.5 (что означает плохое качество аппроксимации данных).

Эти результаты говорят о том, что модель линейной регрессии не может быть использована для прогнозирования вероятности возврата кредитов.

Логистическая регрессия

Теория

Логистическая регрессия — это модель бинарной классификации, используемая для оценки вероятности принадлежности наблюдения к определенному классу. Несмотря на название, логистическая регрессия используется для задач классификации, а не регрессии.

Основные концепции:

Логистическая функция (сигмоид): Основой логистической регрессии является логистическая (или сигмоидальная) функция. Она преобразует линейную комбинацию входных признаков в вероятность принадлежности к классу 1.

Формула логистической функции:

P(Y=1|X) = 1 / (1 + exp(-(w0 + w1x1 + w2x2 + ... + wnxn))) где:

P(Y=1|X) - вероятность того, что целевая переменная Y равна 1 при заданных значениях признаков X.

x1, x2, ..., xn - признаки (факторы).

w0, w1, ..., wn - коэффициенты модели (веса), которые определяют влияние каждого признака.

exp - экспоненциальная функция.

Функция потерь (Loss function): В логистической регрессии используется логарифмическая функция потерь (log loss), также известная как бинарная кросс-энтропия (binary cross-entropy). Она измеряет разницу между предсказанными и фактическими значениями и используется в процессе обучения модели для нахождения оптимальных весов.

Оценка параметров: для оценки параметров модели, таких как веса w, используются методы оптимизации, например, метод градиентного спуска, который минимизирует функцию потерь.

Гиперпараметры:

Гиперпараметры — это настройки модели, которые не могут быть изучены самой моделью и должны быть определены до начала процесса

обучения. Для модели логистической регрессии заданы следующие гиперпараметры:

C: Параметр регуляризации (обратная сила регуляризации). Он контролирует влияние регуляризации на модель. Меньшие значения C указывают на более сильную регуляризацию. Увеличение значения C уменьшает силу регуляризации, что может привести к более точной подгонке модели под обучающие данные. Слишком большие значения C могут привести к переобучению.

penalty: Тип регуляризации, который определяет тип штрафа, применяемого к коэффициентам модели. Варианты: L1-регуляризация (Lasso) добавляет абсолютное значение весов признаков в функцию потерь. Она может привести к разреженности модели, уменьшая веса при некоторых признаках до нуля, что позволяет выполнить отбор признаков. L2-регуляризация (Ridge) добавляет квадраты весов признаков в функцию потерь. Она штрафует большие значения весов, но не обнуляет их.

solver: это алгоритм, используемый для оптимизации функции потерь при поиске оптимальных весов модели. Варианты:

'liblinear': Оптимизация основана на библиотеке LIBLINEAR. Этот solver подходит для небольших датасетов и поддерживает только 'l1' и 'l2' для регуляризации.

'saga': Метод Stochastic Average Gradient. Это улучшенная версия solver'a 'sag', обычно эффективна для больших датасетов.

'lbfgs': Limited-memory Broyden-Fletcher-Goldfarb-Shanno. Этот solver подходит для небольших и средних датасетов. Он использует приближенный метод второго порядка и может обрабатывать различные типы регуляризации.

Выбор solver'a зависит от размера датасета, типа регуляризации и предпочтений при работе с моделью. Например, 'liblinear' может быть предпочтительным для маленьких датасетов с L1- или L2-регуляризацией, в то время как 'lbfgs' может быть хорошим выбором для средних по размеру датасетов. 'saga' часто рекомендуется для больших наборов данных.

Задачи

Задача. Провести классификацию найденного датасета, методом логистической регрессий. В формате Markdown написать пояснения. Объяснить почему были выбраны именно такие гиперпараметры, была ли перекрестная проверка, и т.д.

Ход работы:

Попробуем сразу обучить модель:

```
LogRM = LogisticRegression(max_iter=1000)
LogRM.fit(X_train, y_train)
print(classification_report(y_test, LogRM.predict(X_test)))
print(confusion_matrix(y_test, LogRM.predict(X_test)))
```

```
              precision    recall  f1-score   support

           0       0.91      0.97      0.94      1701
           1       0.77      0.53      0.62       325

    accuracy                           0.90      2026
   macro avg       0.84      0.75      0.78      2026
weighted avg       0.89      0.90      0.89      2026

[[1649   52]
 [ 154  171]]
```

Рисунок 11 – Модель логистической регрессии

```
scaler = StandardScaler()
X_train_scaled = scaler.fit_transform(X_train)
X_test_scaled = scaler.transform(X_test)
LogRM = LogisticRegression(penalty='l2', C=1.0, solver='lbfgs', max_iter=1000)
LogRM.fit(X_train_scaled, y_train)
print(classification_report(y_test, LogRM.predict(X_test_scaled)))
print(confusion_matrix(y_test, LogRM.predict(X_test_scaled)))
```

```
              precision    recall  f1-score   support

           0       0.92      0.97      0.95      1701
           1       0.78      0.58      0.67       325

    accuracy                           0.91      2026
   macro avg       0.85      0.78      0.81      2026
weighted avg       0.90      0.91      0.90      2026

[[1649   52]
 [ 136  189]]
```

Рисунок 12 – Масштабирование данных, подбор параметров и обучение модели

Вывод

Модель логистической регрессии, обученная на датасете банка для предсказания параметра Attrition_Flag, демонстрирует высокую точность. Как в случае использования начальных параметров, так и при применении подобранных параметров, точность модели составляет 90%. Это указывает на то, что модель хорошо справляется с классификацией и предсказанием возвратов кредитов.

Анализ classification report также подтверждает высокие показатели модели.

Таким образом, модель логистической регрессии демонстрирует отличные показатели в отличие от линейной регрессии в контексте данного датасета.

Из плюсов: быстрая обучаемость модели.

Из минусов: может быть неэффективной при низкой линейной разделимости классов: В случае, если классы плохо разделимы линейно, модель может демонстрировать низкую точность.

Модель наивного Байеса

Теория

Теорема Байеса позволяет вычислить условные вероятности и предсказать вероятность принадлежности объекта к определенному классу на основе известных признаков.

Основные концепции:

Теорема Байеса: Теорема Байеса позволяет вычислить условные вероятности и предсказать вероятность принадлежности объекта к определенному классу на основе известных признаков.

Формула теоремы Байеса:

P(C|X) = (P(X|C) * P(C)) / P(X)

где:

P(C|X) - вероятность того, что объект принадлежит классу C при условии, что известны его признаки X.

P(X|C) - вероятность признаков X при условии принадлежности объекта классу C.

P(C) - априорная вероятность класса C. P(X) - вероятность признаков X.

Наивное предположение о независимости признаков:

Модель наивного Байеса предполагает, что все признаки являются независимыми между собой при условии принадлежности к определенному классу. Это "наивное" предположение позволяет снизить вычислительную сложность модели.

Типы наивных Байесовских моделей:

Мультиномиальный наивный Байес (Multinomial Naive Bayes): подходит для признаков с дискретными или счетными значениями, такими как частота слов в тексте.

Бернуллиев наивный Байес (Bernoulli Naive Bayes): применяется к бинарным признакам, где каждый признак представляет собой булево значение.

Гауссов наивный Байес (Gaussian Naive Bayes): предполагает, что непрерывные признаки распределены по нормальному (гауссовому) закону. Применение модели:

Модель наивного Байеса широко используется в задачах классификации текстов, фильтрации спама, анализе тональности текста и других областях, где требуется быстрая и эффективная классификация на основе небольшого объема данных.

Выбор конкретной вариации модели зависит от типа данных и характеристик признаков в задаче классификации.

Задачи

Задача. Провести классификацию найденного датасета, методами наивного Байеса. В формате Markdown написать пояснения. Объяснить почему были выбраны именно такие гиперпараметры, была ли перекрестная проверка, и т.д.

Ход работы:

На рисунках 13-14 представлен код программы.

Подготовим данные для обучения

```
[ ]  1 y = df['Attrition_Flag']
     2 X = df.drop(columns = ['Attrition_Flag'])
     3 X_train, X_test, y_train, y_test = train_test_split(X, y, stratify=y, test_size=0.2)
```

Попробуем обучить модель

```
[ ]  1 nb = GaussianNB()
     2 nb.fit(X_train, y_train)
     3 predictions = nb.predict(X_test)
     4 print(classification_report(y_test, predictions))
```

```
                  precision    recall  f1-score   support

               0       0.93      0.95      0.94      1701
               1       0.69      0.62      0.65       325

        accuracy                           0.89      2026
       macro avg       0.81      0.78      0.79      2026
    weighted avg       0.89      0.89      0.89      2026
```

Рисунок 13 – Подготовка данных и обучение модели

```
[ ]  1 scaler = StandardScaler()
     2 X_train_scaled = scaler.fit_transform(X_train)
     3 X_test_scaled = scaler.transform(X_test)
     4
     5 nb.fit(X_train_scaled, y_train)
     6 predictions = nb.predict(X_test_scaled)
     7 print(classification_report(y_test, predictions))
```

```
                  precision    recall  f1-score   support

               0       0.93      0.93      0.93      1701
               1       0.64      0.64      0.64       325

        accuracy                           0.88      2026
       macro avg       0.78      0.79      0.78      2026
    weighted avg       0.88      0.88      0.88      2026
```

Рисунок 14 – Скалирование данных и повторное обучение модели

Вывод

Исходя из представленного отчета по классификации и точности модели наивного Байеса, можно сделать следующие выводы:

Модель, обученная методом наивного Байеса для предсказания параметра Attrition_Flag на датасете банка, показывает хорошие результаты. Точность модели составляет 89%, что свидетельствует о ее способности правильно классифицировать данные.

Анализ classification report также подтверждает неплохие показатели модели. Метрики precision, recall и f1-score для обоих классов (0 и 1) также демонстрируют высокие значения, особенно для класса 0, где precision и recall достигают 93%.

В целом, модель наивного Байеса демонстрирует высокую точность и хорошие метрики precision, recall и accuracy, что подтверждает ее способность предсказывать параметр Attrition_Flag на основе предоставленных данных эффективно и точно.

Модель решающего дерева

Теория

Модель решающего дерева — это древообразная структура, которая представляет собой последовательность правил принятия решений, каждое из которых разбивает данные на более чистые подгруппы. Эта модель применяется как в задачах классификации, так и в задачах регрессии.

Основные концепции:

Разделение по признакам: Решающее дерево основывается на разделении набора данных на более мелкие подгруппы, выбирая оптимальное разделение по признакам. Цель состоит в минимизации неопределенности или увеличении чистоты (преимущественно увеличение одного класса или уменьшение дисперсии в случае регрессии) в каждой подгруппе.

Узлы и листья:

В решающем дереве каждый узел представляет собой условие на признаке, а каждый лист - прогноз или класс. Алгоритм построения дерева стремится минимизировать ошибку прогнозирования, разделяя данные на чистые подгруппы до определенной глубины или до выполнения критерия останова.

Принцип работы:

Дерево строится путем выбора признака, который лучше всего разделяет данные на основе определенного критерия (например, индекс Джини или энтропия для классификации, среднеквадратичная ошибка для регрессии). Процесс построения дерева продолжается рекурсивно для каждого полученного узла, пока не будет выполнен критерий останова.

Гиперпараметры:

max_depth: Максимальная глубина дерева. Этот параметр контролирует количество уровней в дереве.

min_samples_split: Минимальное количество выборок, необходимое для разделения узла. Если количество выборок в узле меньше этого значения, разделение прекращается.

min_samples_leaf: Минимальное количество выборок, необходимое для формирования листа. Этот параметр определяет критерий останова построения дерева.

Гиперпараметры позволяют настроить процесс построения дерева для достижения лучшей производительности и предотвращения переобучения или недообучения модели. Варьирование этих параметров позволяет найти оптимальное дерево для конкретного набора данных.

Задачи

Задача. Провести классификацию найденного датасета, методом решающего дерева. В формате Markdown написать пояснения. Объяснить почему были выбраны именно такие гиперпараметры, была ли перекрестная проверка, и т.д.

Ход работы:

Разделим данные и попробуем обучить модель:

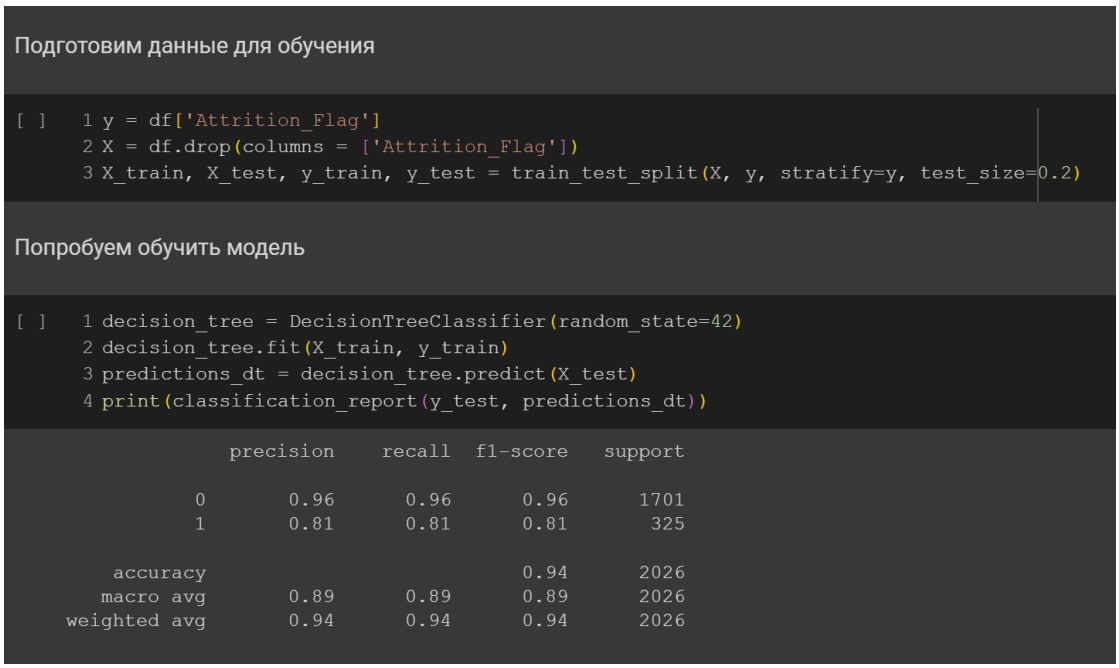

Рисунок 15 – Первичное обучение модели

Попробуем достичь более хороших результатов с помощью скалирования и подбора параметров.

```python
params_dt = {
    'max_depth': [3, 5, 7, 10],
    'min_samples_split': [2, 5, 10],
    'min_samples_leaf': [1, 2, 4]
}

scaler = StandardScaler()
X_train_scaled = scaler.fit_transform(X_train)
X_test_scaled = scaler.transform(X_test)

grid_search_dt = GridSearchCV(decision_tree, params_dt, cv=5)
grid_search_dt.fit(X_train_scaled, y_train)

best_decision_tree = grid_search_dt.best_estimator_
predictions_dt_tuned = best_decision_tree.predict(X_test_scaled)
accuracy_dt_tuned = accuracy_score(y_test, predictions_dt_tuned)
print(classification_report(y_test, predictions_dt_tuned))
```

```
              precision    recall  f1-score   support

           0       0.96      0.97      0.96      1701
           1       0.83      0.78      0.80       325

    accuracy                           0.94      2026
   macro avg       0.89      0.88      0.88      2026
weighted avg       0.94      0.94      0.94      2026
```

Рисунок 16 – Скалирование и подбор параметров с результатами

Вывод

Модель решающего дерева, обученная для предсказания параметра Attrition_Flag на датасете банка, показывает приемлемые результаты.

Анализ classification report также позволяет проанализировать эффективность модели. Метрики precision, recall и f1-score для обоих классов (0 и 1) говорят о том, что модель достаточно хорошо предсказывает оба класса, но с некоторыми ограничениями. Для класса 0 значение precision составляет 96%, а recall - 97%, а для класса 1 - 83% и 78% соответственно.

Таким образом, модель решающего дерева достигла точности на уровне 94% и демонстрирует хорошие метрики для класса 0, но менее убедительные результаты для класса 1.

Модель случайного леса

Теория

Случайный лес — это ансамбль (ensemble) деревьев решений, который использует метод "усреднения прогнозов" для улучшения точности и уменьшения переобучения. Он состоит из большого количества деревьев, где каждое дерево строится на основе подвыборки данных и подмножества признаков.

Основные концепции:

Бутстрэп выборка: Случайный лес использует бутстрэп выборку (подвыборки с возвращением) из обучающего набора данных для построения различных деревьев в ансамбле.

Случайный выбор признаков: Каждое дерево строится на основе случайного подмножества признаков. Это позволяет модели быть более разнообразной и уменьшает корреляцию между деревьями, что способствует улучшению обобщающей способности модели.

Процесс усреднения: Предсказания каждого дерева усредняются для получения окончательного прогноза модели. В задачах классификации используется голосование большинства, а в задачах регрессии - усреднение.

Гиперпараметры модели случайного леса:

n_estimators: Количество деревьев в случайном лесу.

max_depth: Максимальная глубина дерева в случайном лесу. Этот параметр контролирует количество уровней в каждом дереве.

min_samples_split: Минимальное количество выборок, необходимое для разделения узла в дереве.

min_samples_leaf: Минимальное количество выборок, необходимое для формирования листа в дереве.

Гиперпараметры позволяют настраивать процесс построения деревьев в случайном лесу, чтобы достичь оптимальной производительности модели.

Использование различных комбинаций значений гиперпараметров позволяет найти наилучшую модель с учетом специфики данных и задачи.

Задачи

Задача. Провести классификацию найденного датасета, методом решающего дерева. В формате Markdown написать пояснения. Объяснить почему были выбраны именно такие гиперпараметры, была ли перекрестная проверка, и т.д.

Ход работы:

В модели используем следующие гиперпараметры:

```
params_rf = {
    'n_estimators': [50, 100],
    'max_depth': [None, 5],
    'min_samples_split': [2, 5],
    'min_samples_leaf': [1]
}
```

Рисунок 17 – Гиперпараметры

На рисунке 18 представлен код программы и результаты.

```python
params_rf = {
    'n_estimators': [50, 100],
    'max_depth': [None, 5],
    'min_samples_split': [2, 5],
    'min_samples_leaf': [1]
}

random_forest = RandomForestClassifier(random_state=42)
grid_search_rf = GridSearchCV(random_forest, params_rf, cv=5)
grid_search_rf.fit(X_train_scaled, y_train)

best_random_forest = grid_search_rf.best_estimator_
predictions_rf_tuned = best_random_forest.predict(X_test_scaled)
print(classification_report(y_test, predictions_rf_tuned))
```

```
              precision    recall  f1-score   support

           0       0.96      0.99      0.97      1701
           1       0.92      0.78      0.85       325

    accuracy                           0.95      2026
   macro avg       0.94      0.89      0.91      2026
weighted avg       0.95      0.95      0.95      2026
```

Рисунок 18 – Обучение модели с параметрами и результаты

Вывод

Модель случайного леса, обученная для предсказания параметра Attrition_Flag на датасете банка, показывает очень хорошие результаты. Точность модели 95%.

Анализ classification report также позволяет проанализировать эффективность модели. Метрики precision, recall и f1-score для обоих классов (0 и 1) говорят о том, что модель успешно предсказывает оба класса.

Таким образом, модель случайного леса достигла высокой точности на уровне 95%. Метрики precision, recall и f1-score подтверждают, что модель успешно предсказывает оба класса.

Модель CatBoost

Теория

CatBoost (Categorical Boosting) — это метод градиентного бустинга, который автоматически обрабатывает категориальные признаки без необходимости их предварительного кодирования или обработки. Он является мощным алгоритмом машинного обучения для задач классификации, регрессии и ранжирования.

Основные концепции:

Обработка категориальных признаков: CatBoost проводит автоматическую обработку категориальных признаков, что позволяет использовать их напрямую в моделировании без необходимости предварительного кодирования.

Структура градиентного бустинга: это итеративный алгоритм, который комбинирует множество слабых моделей (обычно деревьев решений), улучшая каждую новую модель путем корректировки предыдущих ошибок.

Регуляризация: CatBoost имеет встроенную регуляризацию для предотвращения переобучения модели. Параметры регуляризации, такие как l2_leaf_reg, помогают контролировать сложность модели.

Гиперпараметры:

depth: Максимальная глубина деревьев.

learning_rate: Скорость обучения, контролирующая величину шага при обновлении весов модели.

l2_leaf_reg: Коэффициент регуляризации L2 для весов листьев деревьев.

iterations: Количество итераций (деревьев), выполняемых в процессе обучения.

loss_function: Функция потерь для оптимизации.

Гибкость и удобство в работе с категориальными признаками, автоматическая обработка данных и регуляризация делают CatBoost популярным инструментом для работы с данными, где присутствуют категориальные признаки. Выбор оптимальных гиперпараметров играет важную роль в повышении производительности модели.

Задачи

Задача. Провести классификацию найденного датасета, методом решающего дерева. В формате Markdown написать пояснения. Объяснить почему были выбраны именно такие гиперпараметры, была ли перекрестная проверка, и т.д.

Ход работы:

В модели используем следующие гиперпараметры:

```
param_grid = {
    'iterations': [100, 200],
    'learning_rate': [0.01, 0.1],
    'depth': [3, 6],
    'loss_function': ['MultiClass']
}
```

Рисунок 19 – Гиперпараметры

На рисунке 20 представлен код программы и результаты.

```python
scaler = StandardScaler()
X_train_scaled = scaler.fit_transform(X_train)
X_test_scaled = scaler.transform(X_test)

param_grid = {
    'iterations': [100, 200],
    'learning_rate': [0.01, 0.1],
    'depth': [3, 6],
    'loss_function': ['MultiClass']
}

grid_search = GridSearchCV(
    estimator=CatBoostClassifier(verbose=False),
    param_grid=param_grid,
    cv=5,
    n_jobs=-1
)

grid_search.fit(X_train_scaled, y_train)
best_params = grid_search.best_params_
best_model = grid_search.best_estimator_
y_pred = best_model.predict(X_test_scaled)
print(classification_report(y_test, y_pred, zero_division=1))

```

```
              precision    recall  f1-score   support

           0       0.98      0.99      0.98      1701
           1       0.94      0.90      0.92       325

    accuracy                           0.97      2026
   macro avg       0.96      0.94      0.95      2026
weighted avg       0.97      0.97      0.97      2026
```

Рисунок 20 – Подбор параметров и обучение модели

Вывод

Точность модели CatBoost для предсказания параметра Attrition_Flag на датасете отеля составляет 97%, что является высоким показателем. Метрики precision, recall и f1-score для обоих классов (0 и 1) также составляют 90-99%, что указывает на то, что модель успешно предсказывает оба класса. Эти результаты наводят на мысль, что модель работает очень хорошо, и ее прогнозы могут быть достаточно надежными.

В текущем контексте исходя из предоставленных данных, модель CatBoost показывает идеальную точность предсказаний и может использоваться в приложениях, где требуется высокая надежность предсказаний.

Заключение

Результаты анализа различных моделей машинного обучения выявили сильные и слабые стороны каждого подхода. Модель k-ближайших соседей показывает хорошие результаты в обработке классов с несбалансированным распределением, но имеет ограничения в предсказани. Для улучшения модели можно попробовать балансировку классов или использование других методов классификации

Метод опорных векторов (SVM) обладает хорошей способностью предсказывать классы, но в то же время чувствителен к выбору гиперпараметров, что требует дополнительной тщательной настройки для достижения оптимальной производительности.

Логистическая регрессия демонстрирует высокую точность, но ограничена в моделировании нелинейных связей в данных.

Наивный Байесовский классификатор имеет неплохую точность, однако для достижения лучших результатов требует предварительной обработки данных.

Модель решающего дерева обладает хорошей интерпретируемостью, но может страдать от переобучения на некоторых типах данных.

Случайный лес показывает высокую точность, но подвержен переобучению и требует тщательной настройки гиперпараметров.

CatBoost в моём наборе данных показал самую высокую точность.

Список используемых источников

1. Кадурин, С., Грудинин, С., & Николаев, М. (2018). Deep Learning. Бином.
2. Лукьяненко, Д. (2019). Python и машинное обучение. БХВ-Петербург.
3. Трофимов, А. (2020). Машинное обучение для текстов. ДМК Пресс.
4. Воронцов, К. (2019). Математические методы обучения по прецедентам. Издательский дом "Физико-математическая литература".
5. Кантарович, А., & Гергель, В. (2017). Методы машинного обучения в задачах анализа данных. БХВ-Петербург.
6. Зенкевич, С. (2019). Машинное обучение на Python. Питер.
7. Попов, М. (2018). Глубокое обучение. Символ-Плюс.
8. Бахтеев, О., & Бурков, А. (2020). Основы машинного обучения. Учебное пособие. БХВ-Петербург.
9. Воронцов, К. В. (2014). Математические методы обучения по прецедентам. МФТИ.
10. Юдин, Д. (2018). Практический курс машинного обучения. Питер.