Московский Авиационный Институт

(Национальный Исследовательский Университет)

# **Отчет по лабораторной работе 2**

## **по курсу “Искусственный интеллект”**

**На тему: “Алгоритмы классификации”**

Студент: Ваньков Д. А.

Группа: М80-307Б-17

Преподаватель: Самир Халид

Москва, 2020

Постановка задачи:

Необходимо реализовать алгоритмы машинного обучения. Применить данные алгоритмы на наборы данных, подготовленных в первой лабораторной работе. Провести анализ полученных моделей, вычислить метрики классификатора. Произвести тюнинг параметров в случае необходимости. Сравнить полученные результаты с моделями реализованными в scikit-learn. Аналогично построить метрики классификации. Показать, что полученные модели не переобучились. Также необходимо сделать выводы о применимости данных моделей к вашей задаче.

* Логистическая регрессия
* KNN
* SVM
* Дерево Решений

Реализация.

Логистическая Регрессия

Описание модели

Для мультиклассовой классификации, для большого класса распределений апостериорные вероятности задаются преобразованием softmax линейных функций переменных признаков, так что:

Там я использовал максимальное правдоподобие для отдельного определения условных плотностей классов и априорных значений класса, а затем нашел соответствующие апостериорные вероятности с помощью теоремы Байеса, тем самым неявно определяя параметры {wk}. Использование максимальной вероятности для определения параметров этой модели напрямую. Для этого потребуются производные от yk по всем активациям a\_j. Это дало:

,

где - элементы единичной матрицы. Затем нужно записать функцию правдоподобия. Это легче всего сделать, используя схему кодирования 1 - из - K, в которой целевой вектор для вектора признаков , принадлежащего классу , представляет собой двоичный вектор со всеми элементами, равными нулю, кроме элемента k, который равен единице. Функция правдоподобия тогда выглядит:

где , а T - матрица целевых переменных размером N × K с элементами . Применяя логарифм, получается

То есть функция кросс-энтропийной ошибки для задачи мультиклассовой классификации. Градиент функции ошибки относительно одного из векторов параметров :

,

Для комбинации функции активации логистической сигмоиды и функции кросс-энтропийной ошибки, а также для функции активации softmax с мультиклассовой функцией кросс-энтропийной ошибки получается данная форма. С помощью метода Ньютона-Рафсона, получаю алгоритм IRLS для задачи мультикласса. Это требует оценки матрицы Гессе, которая содержит блоки размера M × M, в которых блок j, k задается как

.

**Реализованная модель.**

**Без регуляризации:**  
Ошибки на кросс валидации:

Accuracy = 0.4045112781954887

Precision = 0.4045112781954887

Roc auc = 0.6272036634337573

Ошибки на выборках:

Train Test

Accuracy = 0.5 | 0.4992412746585736

Precision = 0.5 | 0.4992412746585736

Roc auc = 0.6720754157429131 | 0.672596230890465

**L2 регуляризация:**

Оцененый параметр λ = 1

Ошибки на кросс валидации:

Accuracy = 0.3568922305764411

Precision = 0.3568922305764411

Roc auc = 0.579102713717152

Ошибки на выборках

Train Test

Accuracy = 0.47380239520958084 | 0.4628224582701062

Precision = 0.47380239520958084 | 0.4628224582701062

Roc auc = 0.6019687936145997 | 0.6233187374443899

**Logistic Regression with Sklearn.**

Ошибка на выборках: Accuracy = 0.622154779969651

Ошибки на кросс валидации: Accuracy = 0.6889226100151745

Ошибки на выборках

Train Test

Accuracy = 0.7155688622754491 | 0.6889226100151745

Precision = 0.7155688622754491 | 0.6889226100151745

Roc auc = 0.8097067621777602 | 0.7940238612896533

**Выводы по модели:**

1. Из того, что метрики классификации на трейновой и тестовой выборках не отличаются следует, что модель не переобучилась.

2. Модель логистической мультиклассовой регрессии показалась довольно сложной и заняла достаточно времени. Также, из полученных оценок можно сделать вывод, что данная модель не подходит для решения задачи мультиклассовой классификации.

KNN

Описание модели

**Алгоритм**

Для классификации каждого из объектов тестовой выборки необходимо последовательно выполнить следующие операции:

* Вычислить расстояние до каждого из объектов обучающей выборки
* Отобрать k объектов обучающей выборки, расстояние до которых минимально
* Класс классифицируемого объекта — это класс, наиболее часто встречающийся среди k ближайших соседей

Для простоты я выбрал двумерное пространство, в котором случайным образом на участке от 0 до 5 по каждой из осей выбирается местоположение мат.ожидания двумерного гауссиана со среднеквадратичным отклонением 0.5. Значение 0.5 выбрано, чтобы объекты оказались достаточно хорошо разделимыми (следует из правила трех сигм). Для определения расстояния между объектами можно использовать не только евклидово расстояние: также применяются манхэттенское расстояние, косинусная мера, критерий корелляции Пирсона и др.

Основная функция алгоритма. На вход приходит матрица расстояний между объектами обучающей и тестовой выборки, метки обучающей выборки, число ближайших «соседей». На выходе предсказанные метки для новых объектов и вероятности каждой метки.

Результат выполнения

**Реализованная модель**

k — ближайших соседей = 5

CV scores: [0.8951310861423221, 0.9026217228464419, 0.9213483146067416, 0.9213483146067416, 0.947565543071161]

Train data accuracy: 0.9176029962546817

Ошибки на кросс валидации:

Accuracy = 0.9241274658573596

Precision = 0.9241274658573596

Roc auc = 0.9495108095293339

Ошибки на выборках:

Train Test

Accuracy = 0.9520958083832335 | 0.9241274658573596

Precision = 0.9520958083832335 | 0.9241274658573596

Roc auc = 0.9680700989877046 | 0.9495108095293339

**KNN with Sklearn.**

k — ближайших соседей = 5

Ошибка на выборках: Accuracy = 0.9241274658573596

Ошибки на выборках:

Train Test

Accuracy = 0.9520958083832335 | 0.9241274658573596

Precision = 0.9520958083832335 | 0.9241274658573596

Roc auc = 0.9680700989877046 | 0.9495108095293339

**Вывод по модели:**

1. Из того, что метрики классификации на трейновой и тестовой выборках не отличаются следует, что модель не переобучилась.

2. Модель KNN считается не сложной, но даже при этом на трейновых и тестовых данных выдает хороший результат: accuracy свыше 0.9.

SVM

Описание модели

Мультиклассовые SVM классифицируют входной вектор x ∈ в один из k классов, используя следующее простое правило:

**,**

Каждый вектор ∈ можно рассматривать как прототип, представляющий m-й класс, а - как оценку m-го класса по x. Следовательно, уравнение выбирает класс с наивысшим баллом. Для n обучающих примеров ∈ и связанных с ними меток ∈ [k] мультиклассовая SVM-формулировка оценивает оценки , решив следующую задачу оптимизации:

,

где C > 0 - параметр регуляризации и = 0, если u <0, и u в противном случае. Это уравнение означает, что для каждого учебного случая мы не несем потерь, если оценка правильного класса больше, чем оценка «ближайшего» класса хотя бы на 1. Оценками решив следующую задачу оптимизации:

,

с учетом , если ,

.

Градиент f играет важную роль и определяется:

.

В таком случае, оптимальное решение будет выглядеть:

.

Чем больше , тем больше . Остановка происходит, если .

Результат выполнения

**Реализованная модель.**

C = 2

Ошибки на кросс валидации:

Accuracy = 0.5548872180451128

Precision = 0.5548872180451128

Roc auc = 0.6726917825855202

Ошибки на выборках:

Train Test

Accuracy = 0.4588323353293413 | 0.44764795144157815

Precision = 0.4588323353293413 | 0.44764795144157815

Roc auc = 0.6191507423349809 | 0.6112348013912755

**SVM with Sklearn.**

C = 21

Ошибки на выборках:

Train Test

Accuracy = 0.9468562874251497 | 0.9559939301972686

Precision = 0.9468562874251497 | 0.9559939301972686

Roc auc = 0.9645010781616375 | 0.9707807420761585

**Вывод по модели:**

1. Из того, что метрики классификации на трейновой и тестовой выборках не отличаются следует, что модель не переобучилась.

2. Реализованная модель показала aссuracy 0.45, а sklearn 0.94. Из-за того что поиск парамаетров происходил очень долго параметр был выбран не верно, поэтому accuracy на реализованной модели очень низкий.

Решающее дерево

Описание модели

**Дерево решений представляет собой двоичное дерево:**

* Узел — это одна входная переменная (x) и точка ее разделения.
* Конечные узлы (терминальные) дерева содержат выходную переменную (y), которая используется для прогнозирования.

**Рекурсивный алгоритм**

1. На каждом шаге выбирается признак, при разделении по которому прирост информации оказывается наибольшим. Создание разделения включает в себя три части:
2. Вычисление показателя Джини:
3. Разделение набора данных:
   1. Проверка на то, находится ли значение атрибута ниже или выше значения разделения.
   2. Присвоение его левой или правой группе соответственно.
4. Оценка всех разделений:

* Проверяются значения каждого атрибута как разделение кандидатов, оценивается стоимость разделения и находится наилучшее возможное разделение, которое мы могли бы сделать.
* Как только будет найдено лучшее разделение, оно используется в качестве узла в дереве решений.

1. Выборка делится на левую и правую часть.

* Выбирается разделение с лучшей стоимостью.
* Все входные переменные и все возможные точки разделения оцениваются и выбираются наилучшие на основе функции стоимости.

1. Процедура повторяется рекурсивно к каждой из частей, пока энтропия не окажется равной нулю или очень малой величине.

Дополнительные условия останова:

* Достижение нужной глубины дерева.
* Достижение минимальной записи узла.

**Представление дерева**

* Используется словарь для представления узла в дереве решений, поскольку мы можем хранить данные по имени.
* При выборе наилучшего разделения и использовании его в качестве нового узла для дерева, хранится индекс выбранного атрибута, его значения, по которому нужно разделяться, и две группы данных, разбитые по выбранной точке разделения.
* Каждая группа данных представляет собой собственный небольшой набор данных, состоящий только из тех строк, которые были назначены левой или правой группой в процессе разделения.

**Предсказание**

После создания дерева можно перемещаться с новой строкой данных, следующей за каждой ветвью с разбиениями, пока не будет сделан окончательный прогноз.

Результат выполнения.

**Реализованная модель.**

Максимальная глубина = 9

Минимальная запись узла = 9

Ошибки на кросс валидации:

Accuracy = 0.8441102756892229

Precision = 0.8441102756892229

Roc auc = 0.8965187796097869

Ошибки на выборках:

Train Test

Accuracy = 0.9461077844311377 | 0.8194233687405159

Precision = 0.9461077844311377 | 0.8194233687405159

Roc auc = 0.9641322657183546 | 0.8795584678359948

**Decision tree with Sklearn.**

Максимальная глубина = 6

Минимальная запись узла = 9

Ошибки на выборках:

Train Test

Accuracy = 0.9558383233532934 | 0.8270106221547799

Precision = 0.9558383233532934 | 0.8270106221547799

Roc auc = 0.9705206250064274 | 0.8851529675356118

**Вывод по модели:**

1. Из того, что метрики классификации на трейновой и тестовой выборках не отличаются следует, что модель не переобучилась.

2. Как видно из полученных результатов дерево решений очень хорошо подходит для решения данной задачи классификации.