Модуль 2.

Лабораторная работа 1. Метод линейной регрессии

Введение

Реализуйте линейную регрессию с использованием метода наименьших квадратов без использования сторонних библиотек, кроме NumPy и Pandas

Описание метода

Принцип работы метода линейной регрессии основан на поиске линейной функции, которая наилучшим образом соответствует наблюдаемым данным. Эта линейная функция обычно представляется в виде уравнения прямой линии в двумерном случае или плоскости в многомерном случае. В общем случае уравнение линейной регрессии имеет вид:

$Y=\beta 0+\beta 1X1+\beta 2X2+\ldots+\beta nXn+\epsilon$

Псевдокод метода

```
n = количество наблюдений
p = количество предикторов (независимых переменных)

// Добавление столбца единиц для учета свободного члена
X = добавить_столбец_единиц(X)

// Вычисление оценок коэффициентов регрессии
beta = обратная_матрица(X.T * X) * X.T * y

вернуть beta
```

Результаты выполнения

Коэффициенты:

[[0.70815438]

[-0.89917085]

[-0.84000856]

[0.1195866]

[-0.67389719]

[1.59006959]

[-2.71104732]

[0.51541598]

[1.21160912]], представляющие собой переменные β0-βn

Примеры использования метода

Прогнозирование ответа в случаях, где связь между зависимой и независимыми признаками является линейной

Лабораторная работа 2. Метод k-ближайших соседей (k-NN)

Введение

Реализуйте метод k-ближайших соседей без использования сторонних библиотек, кроме NumPy и Pandas.

Постройте две модели k-NN с различными наборами признаков:

- Модель 1: Признаки случайно отбираются .
- Модель 2: Фиксированный набор признаков, который выбирается заранее.

Описание метода

Алгоритм классификации и регрессии, который использует информацию о близости объектов для принятия решений.
Применяется для решения задач как классификации, так и регрессии:

Классификация: прогнозирует категориальную метку (класс) для нового объекта на основе классов его k ближайших соседей.

Регрессия: прогнозирует числовое значение для нового объекта путем усреднения значений его k ближайших соседей.

Принцип работы:

1. Для нового объекта, для которого нужно сделать прогноз, вычисляется расстояние до каждого объекта в обучающем наборе с использованием евклидова расстояния.

2. Выбор k: задаётся количество ближайших соседей (k), которые будут использоваться для прогноза.
3. Определение прогноза: для задачи классификации, метка нового объекта определяется путем голосования: метка, которая встречается чаще среди k ближайших соседей, становится прогнозом для нового объекта. Для задачи регрессии, значение целевой переменной нового объекта вычисляется как среднее (или медиана) значений целевой переменной его k ближайших соседей.

Псевдокод метода

```
def euclidean_distance(p1, p2):
    # Вычисляем евклидово расстояние
    return np.sum((p1 - p2) ** 2)


def get_neighbors(x_train, y_train, x_test_chosen, k):
    # Находится к ближайших соседей
    return neighbors


def predict(neighbors):
    # Предсказываем класс на основе классов соседей
        return k


def test(x_train, y_train, x_test, k):
    # Выполняем предсказания для тестовой выброки
    return predicted
```

Результаты выполнения

Точность предсказания для разного количества соседей и матрица ошибок предсказаний

```
FOR K = 3
accuracy: 94.44%
Матрица ошибок
[[11.  0.  0.]
 [ 2. 12.  0.]
 [ 0.  0. 11.]]


FOR K = 5
accuracy: 97.22%
Матрица ошибок
[[11.  0.  0.]
 [ 1. 13.  0.]
 [ 0.  0. 11.]]


FOR K = 68
accuracy: 94.44%
Матрица ошибок
[[11.  0.  0.]
 [ 2. 12.  0.]
 [ 0.  0. 11.]]


FOR K = 100
accuracy: 72.22%
Матрица ошибок
[[ 9.  2.  0.]
 [ 1. 13.  0.]
 [ 0.  7.  4.]]
```

Примеры использования метода

В ситуациях, когда нужно классифицировать объекты на основе схожих признаков

Лабораторная работа 3. Деревья решений

Введение

Реализовать без использования сторонних библиотек построение дерева решений (numpy и pandas использовать можно, использовать списки для реализации дерева - нельзя)

Провести оценку реализованного алгоритма с использованием Accuracy, precision и recall

Построить AUC-ROC и AUC-PR (в пунктах 4 и 5 использовать библиотеки нельзя)

Описание метода

Выбор атрибута для разбиения: Для каждого узла дерева выбирается атрибут, который лучше всего разделяет данные. Обычно используются метрики, такие как прирост информации (information gain) или критерий Джини (Gini impurity), для оценки качества разбиения.

Разбиение данных: Данные разделяются на подгруппы в соответствии с выбранным атрибутом. Каждая подгруппа образует новый узел в дереве.

Рекурсивный процесс: Процесс разбиения повторяется рекурсивно для каждого узла, пока не будет выполнено некоторое критерии останова, например, максимальная глубина дерева или минимальное количество объектов в узле.

Листовые узлы: Когда процесс разбиения завершается, узлы, которые больше не могут быть разделены, становятся листовыми узлами, содержащими прогнозируемое значение (в задачах регрессии) или класс (в задачах классификации).

Прогнозирование: Для нового объекта дерево используется для определения пути от корня к одному из листовых узлов, который определяет прогноз для данного объекта.

Псевдокод метода

class Node:
#Класс, в котором будут хранится название класса, значение, критерий разделения и ссылки на следующие узлы

def entropy(Y_values):
# Метод для вычисления значения энтропии, на основе которого определяется следующих признак для разделения.

def gain(X, Y):
# Метод для вычисления прироста информации, по которому определяется значение разветвления

def fit(X: pd.DataFrame, Y: pd.DataFrame, c_depth, mx_depth):
#Метод для создания дерева

def predict(X_object_values, root):
# Предсказание класса на основе созданного дерева

Результаты выполнения

Accuracy: 0.6206896551724138

Precision: 0.6666666666666666

Recall: 0.16666666666666666

tprs=[1.0, 0.0], fprs=[1.0, 0.06666666666666667]

auc-roc = 0.4666666666666667

auc_pr = 0.2413793103448276

Примеры использования метода

Банковское дело. Оценка кредитоспособности клиентов банка при выдаче кредитов.
Промышленность. Контроль качества продукции (обнаружение дефектов в готовых товарах), испытания без нарушений (например, проверка качества сварки) и т.п.
Медицина. Диагностика заболеваний разной сложности.
Торговля. Классификация клиентов и товар.

Лабораторная работа 4. Логистическая регрессия

Введение

Реализуйте логистическую регрессию "с нуля" без использования сторонних библиотек, кроме NumPy и Pandas. Ваша реализация логистической регрессии должна включать в себя:

- Функцию для вычисления гипотезы (sigmoid function).
- Функцию для вычисления функции потерь (log loss).
- Метод обучения, который включает в себя градиентный спуск.
- Возможность варьировать гиперпараметры, такие как коэффициент обучения (learning rate) и количество итераций.

Описание метода

Статистический метод, используемый для моделирования вероятности принадлежности категориальному классу на основе одного или нескольких независимых переменных. Его назначение заключается в классификации объектов на основе значений их атрибутов.

Псевдокод метода

def sigmoid_function(val: float) -> float:
# Сводит значение в промежуток [0:1]

def log_loss_function(real, pred):
# Логарифмическая функция потерь

def newton_optimization(x_train, y_train, iterations):
# Уточнение весов через оптимизацию Ньютона

def gradient_descent(x_train, y_train, iterations=100, learning_rate=0.1):
# Уточнение весов через градиентный спуск

def predict(X_test, coeff):
# Предсказание на основе весов

Результаты выполнения

method	iterations	learning rate	accuracy	precision	recall	f1
Gradient	50	0.01	0.6233766233766234	0	0.0	0
Gradient	50	0.1	0.6233766233766234	0	0.0	0
Gradient	50	0.5	0.6233766233766234	0	0.0	0
Newton	50	-	0.7142857142857143	0.85	0.29310344827586204	0.4358974358974359
Gradient	500	0.01	0.6233766233766234	0	0.0	0
Gradient	500	0.1	0.6233766233766234	0	0.0	0
Gradient	500	0.5	0.7077922077922078	0.8421052631578947	0.27586206896551724	0.41558441558441556
Newton	500	-	0.7467532467532467	0.8064516129032258	0.43103448275862066	0.5617977528089887
Gradient	2000	0.01	0.6233766233766234	0	0.0	0
Gradient	2000	0.1	0.6948051948051948	0.8666666666666667	0.22413793103448276	0.35616438356164387
Gradient	2000	0.5	0.7402597402597403	0.8	0.41379310344827586	0.5454545454545454
Newton	2000	-	0.7597402597402597	0.8387096774193549	0.4482758620689655	0.5842696629213483

Best try:

method	iterations	learning rate	accuracy	precision	recall	f1
Newton	2000	-	0.7597402597402597	0.8387096774193549	0.4482758620689655	0.5842696629213483

Примеры использования метода

Применяется, когда мы хотим оценить связь между бинарной зависимой переменной и одной или несколькими независимыми переменными.

Сравнение методов

Сравнительный анализ методов

Методы отличаются по сложности реализации и затратности. Например, метод k-ближайших соседей и дерево решений. Так же методы разделяются на те, которые предсказывают значение переменной и те, которые определяют класс.

Заключение

В ходе выполнения данного модуля я ознакомился с рядом методов, применяющихся для автоматизации классификации данных и/или предсказаний их значений.