Министерство науки и высшего образования Российской Федерации

Федеральное государственное автономное образовательное учреждение высшего образования

Университет ИТМО

Факультет программной инженерии и компьютерной техники

Отчет по модулю №2

по дисциплине

“Системы искусственного интеллекта”

Выполнил: студент группы
P33131

Бусыгин Дмитрий Алексеевич

Преподаватель:

Королёва Юлия Александровна

Санкт-Петербург
2023

Лабораторная 1. Линейная регрессия

Введение

- Датасет: жилье в Калифорнии
- Получите и визуализируйте (графически) статистику по датасету (включая количество, среднее значение, стандартное отклонение, минимум, максимум и различные квантили).
- Проведите предварительную обработку данных, включая обработку отсутствующих значений, кодирование категориальных признаков и нормировка.
- Разделите данные на обучающий и тестовый наборы данных.
- Реализуйте линейную регрессию с использованием метода наименьших квадратов без использования сторонних библиотек, кроме NumPy и Pandas (для использования коэффициентов использовать библиотеки тоже нельзя). Использовать минимизацию суммы квадратов разностей между фактическими и предсказанными значениями для нахождения оптимальных коэффициентов.
- Постройте **три модели** с различными наборами признаков.
- Для каждой модели проведите оценку производительности, используя метрику коэффициент детерминации, чтобы измерить, насколько хорошо модель соответствует данным.
- Сравните результаты трех моделей и сделайте выводы о том, какие признаки работают лучше всего для каждой модели.
- Бонусное задание
    - Ввести синтетический признак при построении модели

Описание метода

Назначение линейной регрессии: моделирование зависимости целевой переменной от различных признаков, и, как следствие, прогноз возможных результатов при других значениях признаков.

Псевдокод метода (не очень псевдо, но так же понятнее?):

```
# Функция для выполнения линейной регрессии
def linear_regression(X, y, num_epochs=1500, learning_rate=0.5):
   num_samples, num_features = X.shape
   weights = np.zeros(num_features)
   bias = 0

   for epoch in range(num_epochs):
       # Вычисляем предсказания
```

```
        y_pred = np.dot(X, weights) + bias

        # Вычисляем градиенты
        dw = (1 / num_samples) * np.dot(X.T, (y_pred - y))
        db = (1 / num_samples) * np.sum(y_pred - y)

        # Обновляем веса и смещение
        weights -= learning_rate * dw
        bias -= learning_rate * db

    return weights, bias
```

Результат выполнения: `R^2 Score for Model 1: 0.6255785809767773`

Примеры использования метода:
Метод линейной регрессии полезен и применим в случаях, когда между набором признаков и результатом прослеживается линейная зависимость.
Пример: прогноз доходов компании по активности инвесторов в нее
Почему линейная регрессия: метод подходит своей простотой и скоростью, а также в кейсе не требуется высокая точность, достаточно узнать порядок результата.
Пример: Анализ населения в зависимости от местности (по сути то, что делали в лабе)
Почему линейная регрессия: метод подходит своей простотой и скоростью, а также в кейсе не требуется высокая точность, достаточно узнать порядок результата.

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

Введение

Выбор датасета:

Четный номер в группе - Датасет о вине

- Проведите предварительную обработку данных, включая обработку отсутствующих значений, кодирование категориальных признаков и масштабирование.
- Получите и визуализируйте (графически) статистику по датасету (включая количество, среднее значение, стандартное отклонение, минимум, максимум и различные квантили), постройте 3d-визуализацию признаков.
- 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.
- Для каждой модели проведите оценку на тестовом наборе данных при разных значениях k. Выберите несколько различных значений k, например, k=3, k=5, k=10, и т. д. Постройте матрицу ошибок.

Описание метода:
Назначение: классификация элементов по набору признаков, основываясь на элементах, у которых набор признаков наиболее схож. В геометрическом смысле - объекту присваивается категория его ближайших соседей.

Псевдокод метода

```
def euclidean_distance(x1, x2):
    return np.sqrt(np.sum((x1 - x2) ** 2))

def k_nearest_neighbors(X, y, query_point, k):
    distances = [euclidean_distance(query_point, x) for x in X]
    k_indices = np.argsort(distances)[:k]
    k_nearest_labels = [y[i] for i in k_indices]
    most_common = np.bincount(k_nearest_labels).argmax()
    return most_common
```

Результат выполнения:

```
Confusion Matrix for Model 2 with k=3:
 [[14  0  0]
 [ 1 11  2]
 [ 0  0  8]]
```

```
Confusion Matrix for Model 2 with k=5:
 [[14  0  0]
 [ 1 12  1]
 [ 0  0  8]]
Confusion Matrix for Model 2 with k=10:
 [[13  0  1]
 [ 1 12  1]
 [ 0  0  8]]
```

Примеры использования метода:
Пример: определение категории продукта питания по его химическим признакам (сыр, вино, пиво и др.)
Почему k-NN: категории в этой сфере имеют определенные правила попадания в ту или иную категорию (например просто сыр с плесенью не может назвать себя Рокфором), эти правила можно будет добавить к классификатору, чтобы улучшить выдачу

Пример: классификация текстов
Почему k-NN: Текстовые данные могут быть представлены в виде векторов признаков, и k-NN хорошо справляется с задачами классификации в многомерных пространствах.

Лабораторная работа 3. Деревья решений

Задание

1. Для студентов с четным порядковым номером в группе – датасет с классификацией грибов
2. Отобрать **случайным** образом sqrt(n) признаков
3. Реализовать без использования сторонних библиотек построение дерева решений (дерево не бинарное, numpy и pandas использовать можно, использовать список списков для реализации дерева - нельзя) для решения задачи бинарной классификации
4. Провести оценку реализованного алгоритма с использованием Accuracy, precision и recall
5. Построить кривые AUC-ROC и AUC-PR (в пунктах 4 и 5 использовать библиотеки нельзя)

Описание метода:
Назначение: Деревья решений представляют собой алгоритм для классификации и решения задачи регрессии. Суть в построении древовидной структуры, каждый узел которой - небинарный предикат, от решения которого зависит дальнейший путь по дереву.
Делает объекты одной и той же группы максимально однородными

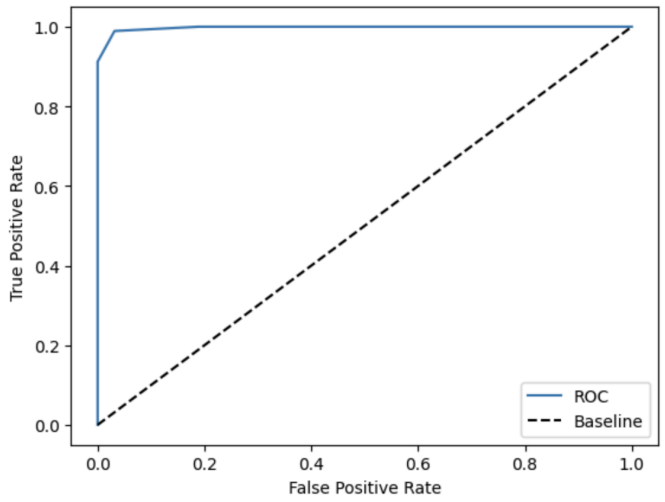
Псевдокод метода

```
def _build_node(self, x, y, parent_info: float) -> Node:
        if len(y.unique()) == 1:
            return LeafNode(y.unique()[0], 1)

        best_gain = 0
        best_gain_info = 0
        best_gain_col = ''

        for col in self.cols:
            cat_names = x[col].unique()
            info = 0

            for cat in cat_names:
                cat_weight = x[col].value_counts()[cat] / len(x)
                entropy = DecisionTree._entropy(y[x[col] == cat])
                info += cat_weight * entropy

            if parent_info - info > best_gain:
                best_gain = parent_info - info
                best_gain_info = info
                best_gain_col = col

        if best_gain_col == '':
            mode = y.mode()[0]
            return LeafNode(mode, y.value_counts()[mode] / len(y))

        values_to_node = {}

        cat_names = x[best_gain_col].unique()
        for cat in cat_names:
            values_to_node[cat] = self._build_node(x[x[best_gain_col] ==
cat], y[x[best_gain_col] == cat], best_gain_info)

        return Node(best_gain_col, values_to_node)
```

Результат выполнения:

```
Используются признаки: population habitat bruises
stalk-surface-above-ring ring-type
ring-type == p:
  population == s:
    habitat == u:
      -> p (1)
    habitat == m:
      -> e (1)
    habitat == g:
      -> e (0.6875)
    habitat == p:
      -> e (1)
    habitat == d:
      -> p (1)
  population == n:
    -> e (1)
  population == v:
    habitat == g:
      -> p (1)
    habitat == u:
      bruises == t:
        -> p (1)
      bruises == f:
        -> e (1)
    habitat == d:
      bruises == t:
        -> e (1)
      bruises == f:
        -> p (1)
    habitat == m:
      -> p (1)
    habitat == l:
      -> e (1)
    habitat == p:
      -> e (1)
  population == y:
    -> e (1)
  population == c:
    bruises == t:
      -> p (1)
    bruises == f:
      -> e (1)
ring-type == e:
  habitat == g:
    -> e (1)
  habitat == w:
    -> e (1)
  habitat == p:
    -> p (1)
```

```
  habitat == l:
    -> p (0.9223744292237442)
  habitat == d:
    -> p (1)
ring-type == l:
  -> p (1)
ring-type == f:
  -> e (1)
ring-type == n:
  -> p (1)
```



Примеры использования метода:
Пример: процесс найма новых работников
Почему дерево решений: множество правил найма и требований к кандидату легко решаются этим способом

Пример: медицинская диагностика
Почему дерево решений: диагнозы формируются по показаниям и симптомам, так что во-избежание человеческих ошибок, зачастую такие умозаключения уже производятся машинами

Лабораторная работа 4. Логистическая регрессия

Введение

1. Выбор датасета:
    - Датасет о диабете: Diabetes Dataset

2. Загрузите выбранный датасет и выполните предварительную обработку данных.
3. Получите и визуализируйте (графически) статистику по датасету (включая количество, среднее значение, стандартное отклонение, минимум, максимум и различные квантили).
4. Разделите данные на обучающий и тестовый наборы в соотношении, которое вы считаете подходящим.
5. 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.
6. Исследование гиперпараметров:
    - Проведите исследование влияния гиперпараметров на производительность модели. Варьируйте следующие гиперпараметры:
        - Коэффициент обучения (learning rate).
        - Количество итераций обучения.
        - Метод оптимизации (например, градиентный спуск или оптимизация Ньютона).
7. Оценка модели:
    - Для каждой комбинации гиперпараметров оцените производительность модели на тестовом наборе данных, используя метрики, такие как accuracy, precision, recall и F1-Score.

Описание метода:
Назначение: логистическая регрессия используется для бинарной классификации: т.е для определения к какому из двух классов принадлежит элемент. Может также быть применен для многоклассовой классификации или регрессии.

Псевдокод метода

```
def fit(self, X, Y):
        self.weights = np.zeros(X.shape[1])
        self.bias = 0
        for _ in range(self.n_iter):
            weights = np.dot(self.weights, X.T) + self.bias
            pred = self._get_sigmoid(weights)
            match self.method:
                case 1: error_weights, error_bias =
self._calculate_gradients(X, Y, pred)
                case 2: error_weights, error_bias =
self._calculate_newton(X, Y, pred)
            self.weights = self.weights - self.learning_rate *
error_weights
            self.bias = self.bias - self.learning_rate * error_bias


        pred_to_class = [1 if p > 0.5 else 0 for p in pred]
        self.train_accuracy = accuracy_score(Y, pred_to_class)
        self.loss = self._calculate_loss(Y, pred)

    def predict(self, x):
        x_dot_weights = np.dot(x, self.weights.T) + self.bias
        probabilities = self._get_sigmoid(x_dot_weights)
        return [1 if p > 0.5 else 0 for p in probabilities]
```

Результат выполнения:

```
Gradient descent method
ACCURACY:
           0.1      0.01     0.001
10    0.671875  0.723958  0.750000
100   0.671875  0.734375  0.723958
1000  0.645833  0.739583  0.734375

PRECISION:
           0.1      0.01     0.001
10    0.603175  0.761905  0.793651
100   0.619048  0.746032  0.761905
1000  0.492063  0.587302  0.746032

RECALL:
           0.1      0.01     0.001
10    0.500000  0.558140  0.588235
100   0.500000  0.573171  0.558140
```

```
1000  0.462687  0.606557  0.573171

F1_SCORE:
           0.1      0.01     0.001
10    0.546763  0.644295  0.675676
100   0.553191  0.648276  0.644295
1000  0.476923  0.596774  0.648276


Newton's optimization method
ACCURACY:
           0.1      0.01     0.001
10    0.755208  0.744792  0.744792
100   0.750000  0.744792  0.744792
1000  0.750000  0.744792  0.744792

PRECISION:
           0.1      0.01     0.001
10    0.650794  0.555556  0.571429
100   0.634921  0.555556  0.555556
1000  0.634921  0.555556  0.555556

RECALL:
           0.1   0.01    0.001
10    0.621212  0.625  0.62069
100   0.615385  0.625  0.62500
1000  0.615385  0.625  0.62500

F1_SCORE:
           0.1      0.01     0.001
10    0.635659  0.588235  0.595041
100   0.625000  0.588235  0.588235
1000  0.625000  0.588235  0.588235
```

Примеры использования метода:

Пример: прогноз покупки пользователя в онлайн магазине по его параметрам

Пример: классификация спама на почте

Почему логистическая регрессия: бинарная классификация, основанная на множестве признаков - работка для лог регрессии

Сравнительный анализ методов

Логистическая регрессия: Преимущества:

- Простота интерпретации: Результаты логистической регрессии легко интерпретировать. Веса при признаках указывают на их влияние на прогноз.

- Эффективность: Логистическая регрессия может хорошо работать в случае линейно разделимых данных.

- Низкое потребление ресурсов: Обучение логистической регрессии обычно требует меньше вычислительных ресурсов.

Ограничения:

- Линейные зависимости: Логистическая регрессия работает лучше в случае линейных зависимостей между признаками и целевой переменной.

- Неспособность моделировать сложные взаимосвязи: Этот метод может быть неэффективен, если в данных присутствуют сложные нелинейные взаимосвязи.

Деревья решений: Преимущества:

- Способность к работе с нелинейными данными: Деревья решений могут легко моделировать нелинейные зависимости и взаимосвязи в данных.

- Интерпретируемость: Деревья решений могут быть легко визуализированы и интерпретированы, что облегчает понимание принципов принятия решений.

- Низкие требования к предобработке данных: Деревья решений могут обрабатывать данные разного типа (категориальные и числовые) без дополнительной предобработки.

Ограничения:

- Склонность к переобучению: Деревья решений могут сильно настраиваться под обучающие данные, что может привести к переобучению.

- Неустойчивость к изменениям в данных: Небольшие изменения в данных могут привести к существенным изменениям в построенном дереве, что делает их менее стабильными.

Примеры лучшего использования каждого метода

K-ближайших соседей (KNN): Ситуации

- Мало признаков, локальные зависимости.

Почему KNN:
- Прост в реализации.
- Адаптивен к изменениям в данных.

Логистическая регрессия: Ситуации:
- Линейные зависимости
- бинарная/многоклассовая классификация.

Почему логистическая регрессия:
- Простота интерпретации.
- Эффективен при линейных зависимостях.

Линейная регрессия: Ситуации:

- Линейные зависимости, регрессия.

Почему линейная регрессия:
- Простота интерпретации.
- Эффективен при линейных зависимостях.

Дерево решений: Ситуации:

- Нелинейные зависимости, интерпретация не критична.

Почему деревья решений:

- Способны моделировать сложные взаимосвязи.
- Низкие требования к предобработке данных.

Заключение

В процессе выполнения лабораторных работ я освежил свое понимание работы метода k-NN и регрессий, познакомился с новым для себя видом классификации - деревом решений, а также прекрасно провел время, составляя этот отчет!