Линейная регрессия

Практические задания для самостоятельного выполнения

Внимание: некоторые задания базируются на результатах выполнения предыдущих заданий. Поэтому рекомендуется сохранять результаты выполнения всех заданий.

Задание 1.

1. Используя модуль ***datasets*** библиотеки ***Scikit-learn***, сгенерировать модельный набор данных для задачи линейной регрессии с одной целевой переменной и двумя признаками, из которых информативным является один. Параметр, определяющий степень рассеянности данных, установить равным 5.0, количество объектов положить равным 200. Обеспечить воспроизводимость результатов, задав значение соответствующему параметру.
2. Вывести на одном графике сгенерированный набор данных в координатах информативный признак – целевая переменная (использовать ***pylab.scatter***) и прямую со сгенерированными коэффициентами.
3. Поэкспериментировать с величиной шума: задать значения соответствующего параметра равными 7.0, 10.0 и 15.0, вывести графики (вместе с прямой регрессии) в одном ряду, с заголовками, сообщающими об используемом значении параметра шума.
4. Выполнить разовое разбиение всех полученных наборов данных (с различными значениями шума) на обучающую и тестовую выборки в соотношении 70/30.
5. Для каждого набора данных создать модель линейной регрессии и обучить ее на обучающей выборке, используя метод градиентного спуска.
6. Получить предсказания обученных моделей для объектов тестовых выборок. Вывести массивы ответов на тестовых выборках и массивы предсказанных моделью значений. Представить результаты в графической форме и проанализировать их.
7. Вывести коэффициенты уравнений, сформированных при генерации наборов данных, и коэффициенты регрессии, полученные после обучения моделей.
8. Записать уравнение исходной зависимости (использованное в процессе генерации набора данных) и полученные уравнения регрессии. Внимание: не следует забывать про свободный член в уравнении регрессии!

Задание 2.

Часть 1.

1. Имеются данные о росте и весе 25 тыс. подростков в дюймах и фунтах соответственно (источник). Данные содержатся в файле HeightsWeights.csv (выдается преподавателем), либо можно скачать данные из источника. Импортировать эти данные в DataFrame и вывести несколько первых записей (для контроля корректности импорта и получения представления о наборе данных).

2. Выполнить первичный анализ данных: построить гистограммы распределения признаков (с заголовками) и проанализировать характер их распределения, наличие/отсутствие выбросов.
3. Добавить в DataFrame новый признак – индекс массы тела (BMI). Формулу для его вычисления можно, получить, например, здесь (не забыть перевести фунты в килограммы, а дюймы в метры!). Указание: удобно использовать метод ***apply*** в сочетании с лямбда-функцией Python. Вывести несколько первых записей и убедиться, что значения нового признака вычислены корректно.
4. Визуализировать попарные зависимости признаков ($m \times m$ графиков, где m – число признаков: по диагонали – гистограммы распределения признаков, вне диагонали – scatter-графики зависимостей двух признаков). Указание: использовать метод ***scatter_matrix*** из модуля ***plotting*** библиотеки ***Pandas***. Проанализировать полученные графики: выяснить, есть ли пары признаков, для которых можно предполагать линейную зависимость.
5. Написать функцию, которая по двум параметрам $\boldsymbol{w_0}$ и $\boldsymbol{w_1}$ вычисляет квадратичную ошибку приближения зависимости роста $\boldsymbol{y}$ от веса $\boldsymbol{x}$ линейной функцией $\boldsymbol{y = w_0 + w_1 \cdot x}$.
6. Реализовать метод градиентного спуска для минимизации ошибки, обеспечив сохранение на каждой итерации текущего значения ошибки.
7. Применить градиентный спуск к имеющемуся набору данных; вывести уравнение полученной линейной зависимости с оптимальными значениями коэффициентов.
8. Вывести на одном графике исходный набор данных в координатах вес – рост (использовать ***pylab.scatter***) и прямую с оптимальными значениями коэффициентов.
9. Вывести график, показывающий изменение значения ошибки алгоритма в зависимости от числа итераций.

Часть 2.

10. Реализовать метод стохастического градиентного спуска для минимизации ошибки, обеспечив сохранение на каждой итерации текущего значения ошибки. Условие останова использовать то же, что и в п. 6.
11. Применить стохастический градиентный спуск к имеющемуся набору данных; вывести уравнение полученной линейной зависимости с оптимальными значениями коэффициентов.
12. Вывести на одном графике исходный набор данных в координатах вес – рост (использовать ***pylab.scatter***) и прямую с оптимальными значениями коэффициентов, полученных с помощью стохастического градиентного спуска.
13. Вывести график, показывающий изменение значения ошибки алгоритма в зависимости от числа итераций. Сопоставить этот график с графиком, полученным в п. 9.
14. Сравнить результаты работы алгоритмов Full GD и SGD по достигнутому значению функции ошибки, числу итераций и затраченному времени (для вывода затраченного времени использовать в коде команду %%time – перед вызовом метода).
15. Сделать выводы.

Часть 3.

16. Используя инструментарий библиотеки ***Scikit-learn,*** выполнить разовое разбиение данных исходного набора (значения веса и роста) на обучающую и тестовую выборки в соотношении 70/30. Значения веса рассматривать в качестве признака, значения роста – в качестве целевой переменной.
17. На обучающей выборке обучить две модели линейной регрессии: используя метод градиентного спуска и метод стохастического градиентного спуска.
18. Получить предсказания обученных моделей для объектов тестовой выборки. Вывести массивы ответов на тестовой выборке и массив предсказанных моделью значений. Представить результаты в графической форме.
19. Записать полученные уравнения регрессии. Сравнить их с уравнениями, полученными при выполнении части 1 и части 2 данного задания.
20. Сделать выводы.

Задание 3.

1. Сгенерировать модельный набор данных для задачи линейной регрессии с одной целевой переменной и четырьмя признаками, из которых информативными являются два. Параметр, определяющий степень рассеянности данных, установить равным 5.0. Обеспечить воспроизводимость результатов, задав значение соответствующему параметру.
2. Выполнить разбиение полученного набора данных на обучающую и тестовую выборки в соотношении 70/30.
3. Построить модели линейной регрессии, использующие $\boldsymbol{L_2}$- и $\boldsymbol{L_1}$-регуляризаторы. Обучить эти модели на обучающей выборке, используя следующие значения коэффициента регуляризации: 0.001, 0.01, 0.2, 1, 2, 10.
4. Для каждой обученной модели получить предсказания на объектах тестовой выборки.
5. Вывести коэффициенты уравнений регрессии, полученные после обучения каждой модели. Записать соответствующие уравнения регрессии.
6. Выполнить анализ полученных уравнений: проследить, какое влияние на получаемые результаты оказывает тип регуляризатора, значение коэффициента регуляризации.
7. Создать отчет по результатам выполнения задания: постановка задачи, описание каждой модели (используемый регуляризатор, используемое значение коэффициента регуляризации, полученные результаты, общие выводы по результатам анализа п. 6).

Задание 4.

1. Оценить качество моделей регрессии, полученных при выполнении части 3 задания 2. Для оценки использовать метрики ***MSE***, ***MAE*** и $\mathbf{R^2}$. В целях более корректного сопоставления оценок ***MSE*** и ***MAE*** найти значение квадратного корня из ***MSE***.
2. Дать интерпретацию полученным результатам.

Задание 5 (дополнительно).

Исходная постановка задачи представлена на **kaggle**: https://www.kaggle.com/c/bike-sharing-demand. Исходные данные: https://www.kaggle.com/c/bike-sharing-demand/data. Для анализа следует использовать данные из файла *train.csv* (на указанном ресурсе). Требуется получить прогноз спроса (по часам) на велосипеды, выдаваемые в прокат. Необходимо учесть, что в наборе признаков присутствуют вещественные, категориальные, и бинарные данные, которые должны обрабатываться раздельно.

1. Импортировать данные в DataFrame и вывести несколько первых записей (для контроля корректности импорта и получения представления о наборе данных).
2. Выполнить проверку на присутствие в наборе пропущенных значений (при наличии пропусков их следует обработать).
3. Проанализировать, какая информация находится в столбцах *casual*, *registered* и *count* (последний – прогнозируемый признак). Объяснить, почему столбцы *casual* и *registered* не должны использоваться в качестве признаков в задаче построения прогноза общего спроса на велосипеды. Удалить эти столбцы из набора.
4. Вывести информацию обо всех полях полученного датафрейма. Обратить внимание на тип данных поля *datetime*. Преобразовать это поле к типу *datetime* (для возможности дальнейшего применения различных преобразований даты).
5. Создать два новых поля, содержащих месяц и час (выделив их из соответствующей даты).
   Пример: data['month'] = data.datetime.apply(lambda x: x.month)
   Удалить столбец с датами (оставив для анализа два новых столбца).
6. Разделить исходный набор данных на обучающую и тестовую выборки таким образом, чтобы более ранние по времени данные использовались для обучения, а более поздние – для проверки качества модели. Для проверки использовать последние 1000 записей, остальные - для обучения. Выполнить проверку: вывести временные диапазоны для обучающей и тестовой выборки.
7. Выделить значения целевого признака в отдельный объект (и для обучающего, и для тестового набора). Удалить соответствующий столбец из обучающего и тестового наборов.
8. Организовать раздельную обработку бинарных, числовых и категориальных признаков:
   - бинарные признаки не нужно преобразовывать (только отделить);
   - числовые признаки – отделить и отмасштабировать (метод ***StandardScaler*** из модуля ***preprocessing***);
   - категориальные признаки – отделить и применить бинарное кодирование;
   - после обработки столбцы собрать вместе, причем после сборки порядок столбцов должен сохраняться (для этого можно использовать трансформер ***FeatureUnion***);
   - для преобразованного набора данных создать модель линейной регрессии.

   Указание: можно использовать метод ***Pipeline***, который создает цепочку из

нескольких шагов (в данном случае – двух шагов: обработки данных и построения модели регрессии) как единый объект, который можно обучать на обучающих данных.

9. Обучить модель линейной регрессии на обучающих данных. Оценить качество полученной модели на тестовых данных с помощью метрики *MAE*.
10. Для получения наглядного представления об адекватности модели выполнить визуализацию: вывести облако точек в координатах «истинные значения целевой функции» – «предсказанные моделью значения». Проанализировать полученные результаты, сделать выводы.

    Указание: ясно, что при хорошей предсказательной способности модели облако точек должно располагаться вдоль биссектрисы первого координатного угла (прогнозируемые моделью результаты близки к истинным значениям целевой функции).

    Сделать выводы.