Open ML Course: Линейные модели

Open ML Course: Линейные модели 2023

Домашнее задание

Для решения нужно использовать python и sklearn, желательно последних версий.

Вопрос 1. Создайте данные используя make_regression(n_samples=100, n_features=20, noise=10, random_state=42)
Что можно сказать о полученных данных?

- [x] **Среднее признаков по модулю меньше единицы и стандартное отклонение около единицы (отличается не более, чем на 20%)**
- [ ] Есть корреляции больше .5 между разными признаками
- [ ] Для вычисления значений признаков требуется регуляризованная регрессия
- [ ] Для лучшего предсказания признаки нужно отнормировать

Вопрос 2. Используя sklearn.linear_model постройте LinearRegression, Ridge(random_state=1), Lasso на дефолтных параметрах и ненормированных признаках. Для оценки регрессии используем R2 из

функции score модели.

Ответьте на вопросы 1-3.

Теперь отнормируйте признаки. Постройте опять 3 регрессии.

Ответьте на вопросы 4-6.

Вопрос 7 относится ко всем построенным в вопросе регрессиям.

- [ ] По score (R2) Ridge лучше всего
- [x] **По score (R2) LinearRegression лучше всего**
- [ ] По score (R2) Lasso лучше всего. Теперь отнормируйте признаки. Постройте опять 3 регрессии
- [ ] По score (R2) Ridge лучше всего
- [x] **По score (R2) LinearRegression лучше всего**
- [ ] По score (R2) Lasso лучше всего
- [ ] Из всех построенных регрессий в вопросе, Ridge лучше всего

Вопрос 3. Из вопроса 2 используйте LinearRegression, Ridge, Lasso, построенные на признаках без нормирования. Для ответов на вопросы сравните полученные коэффициенты.

Для LinearRegression посчитайте значимость признаков, используя statsmodels. Достаньте истинные значения признаков из make_regression.

Для вопросов 5 и 6 формула расчета:
d = abs(round(reg.coef_[i],3) - round(coefs[i],3))/round(coefs[i],3)
if d == np.inf: d=1
if d == np.nan: d=0
coefs - истинные коэффициенты.

- [ ] Значение коэффициента у константы лучше всего оценила LinearRegression (без проверки значимости)
- [ ] Lasso правильно занулила все нулевые признаки
- [ ] Модуль statsmodels для LinearRegression неправильно определил значимость некоторых коэффициентов
- [ ] У Ridge и Lasso коэффициенты всегда меньше, чем у LinearRegression
- [ ] LinearRegression точнее всего определила коэффициенты, если посчитать сумму долей абсолютных отклонений от истинного значения

- [x] **LinearRegression точнее всего определила коэффициенты, если посчитать сумму долей абсолютных отклонений от истинного значения и занулить те коэффициенты, которые незначимо отличаются от нуля по p-value**

Вопрос 4. Давайте теперь разберем, как правильно нормировать X. Изменим немного данные:
X['x5'] = X['x5'] + 5
X['x10'] = X['x10'] + 10
X['x15'] = X['x15'] + 15
(Порядковый номер столбца X с 0)

Разобьем выборку на train - где мы будем обучать регрессию и test - где мы будем проверять качество регрессии:
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.33, random_state=42)

В качестве реализации регрессии возьмем Ridge(random_state=1). Метрика качества R2.

- [ ] Нормирование X_train и X_test с помощью StandardScalar, который обучен на X_train, дает лучше результат, по сравнению с отсутствием нормировки, причем значительно (во 2-м знаке)
- [x] **Нормирование X_train и X_test с помощью StandardScalar, который обучен на X_train для X_train, и на X_test для X_test, дает хуже результат, чем StandardScalar, который обучен только на X_train, причем значительно (во 2-м знаке)**
- [ ] Нормирование X_train и X_test с помощью StandardScalar, который обучен на всем X, дает самые лучшие результаты, сравнивая все варианты обучения StandardScalar

Вопрос 5. Давайте теперь изменим только X_test (моделируя ситуацию, когда тестовые данные "поплыли"):
X_test['x5'] = X_test['x5'] - 2
X_test['x10'] = X_test['x10'] - 5
X_test['x15'] = X_test['x15'] - 7

В качестве реализации регрессии продолжаем с Ridge(random_state=1). Метрика качества R2.

- [x] **Нормирование X_train и X_test с помощью StandardScalar, который обучен на X_train, дает лучше результат, по сравнению с отсутствием нормировки, причем значительно (во 2-м знаке)**
- [x] **Нормирование X_train и X_test с помощью StandardScalar, который обучен на X_train для X_train, и на X_test для X_test, дает лучше результат, чем StandardScalar, который обучен только на X_train, причем значительно (во 2-м знаке)**
- [ ] Нормирование X_train и X_test с помощью StandardScalar, который обучен на всем X, дает самые лучшие результаты, сравнивая все варианты обучения StandardScalar

- [ ] R2 не может быть меньше 0

Вопрос 6.
Давайте теперь изменим X_test и X_train следующим образом:

X_train['x5'] = X_train['x5'] * 2.5
X_train['x10'] = X_train['x10'] * 5.5
X_train['x15'] = X_train['x15'] * 7.7

X_test['x5'] = X_test['x5'] * 2 + 0.5
X_test['x10'] = X_test['x10'] * 5 + 1
X_test['x15'] = X_test['x15'] * 7 + 1.5

В качестве реализации регрессии продолжаем с Ridge(random_state=1). Метрика качества R2.

- [ ] Нормирование X_train и X_test с помощью StandardScalar, который обучен на X_train, дает лучше результат, по сравнению с отсутствием нормировки, причем значительно (во 2-м знаке)
- [x] **Нормирование X_train и X_test с помощью StandardScalar, который обучен на X_train для X_train, и на X_test для X_test, дает хуже результат, чем StandardScalar, который обучен только на X_train, причем значительно (в 2-м знаке)**
- [x] **Нормирование X_train и X_test с помощью StandardScalar, который обучен на всем X, дает самые лучшие результаты, сравнивая все варианты обучения StandardScalar**

Вопрос 7. Работаем с датасетом, определенном в вопросе 4 (где 5й,10й,15й признаки увеличены). Разобьём на трейн и тест как в вопросе 4. Используем нормированные признаки и Ridge(random_state=1):

Теперь давайте сделаем пропущенные значения для важного признака:
X_train.loc[X_train['x1'].isin(X_train['x1'].sample(smpl, random_state=1+exp)), 'x1'] = np.nan,
где попробуем 5 вариантов случайности exp = range(5). smpl - кол-во пропущенных значений.

Ответьте на вопросы 1-2.

Потом сделаем пропущенные значения только для неважного признака:
X_train.loc[X_train['x10'].isin(X_train['x10'].sample(smpl, random_state=1+exp)), 'x10'] = np.nan,
где попробуем 5 вариантов случайности exp = range(5).

Ответьте на вопросы 3-4.

- [ ] При 0-60% пропущенных значений, заполнение средним по x1 работает лучше для важного признака, чем отбрасывание наблюдений с пропущенными значениями
- [x] **Для 70-90% пропущенных значений, заполнение средним по x1 работает лучше для важного признака, чем отбрасывание наблюдений с пропущенными значениями**

- [x] **При 0-60% пропущенных значений, заполнение средним по x10 работает в среднем лучше для неважного признака, чем отбрасывание наблюдений с пропущенными значениями**
- [x] **Для 70-90% пропущенных значений, заполнение средним по x10 работает лучше для неважного признака, чем отбрасывание наблюдений с пропущенными значениями**

Вопрос 8.
Вопросы 4-6 были про правильное нормирование. Вопрос 7 был про заполнение пропусков. Подведем итоги экспериментов. Какие выводы можно сделать?

- [x] **Лучше всегда нормировать при применении регуляризации**
- [x] **Если мы предполагаем, что распределение X_test (данные, на которых мы будем применять модель) может измениться, то лучше отдельно обучать и применять StandardScalar для train и отдельно обучать и применять для test**
- [ ] Заполнение пропусков средним работает всегда
- [x] **Если признак важный, а доля пропущенных значений невелика, то лучше наблюдения с пропущенными значениями просто отбросить**

Вопрос 9. Создадим данные, используя
X, y, coefs = make_regression(n_samples=1000, n_features=200, noise=10, random_state=42, coef=True)

Давайте построим Ridge(random_state=1), Lasso, LinearRegression на ненормированных данных (т.к. сырые данные близки к нормированным) и на 100 случайно выбранных наблюдениях. Качество оценим по mean_squared_error от истинных значений и предсказаний на наблюдениях, которые не участвовали в обучении (900).

- [ ] Самая лучшая регрессия – LinearRegression
- [ ] Самая лучшая регрессия – Ridge
- [x] **Самая лучшая регрессия – Lasso**
- [ ] Самые близкие к истинным значения коэффициентов у Linear Regression, если оценить по методу вопроса 3
- [x] **Самые близкие к истинным значения коэффициентов у Lasso, если оценить по методу вопроса 3**
- [x] **При количестве наблюдений меньше, чем количество признаков, Lasso может помочь**
- [ ] Регрессию можно использовать при любом соотношении количества признаков и наблюдений

Вопрос 10.
Мультиколлинеарность. Вернемся к первоначальным данным:
X, y, coefs = make_regression(n_samples=100, n_features=20, noise=10, random_state=42, coef=True)

Добавим к X столбец x['x20'] = x['x1'] (столбец, равный первому признаку, если нумерация столбцов с 0).
Давайте построим Ridge(random_state=1), Lasso, LinearRegression на нормированных данных.

Ответьте на вопросы 1-3.

Пропуск важной переменной. Вернемся к первоначальным данным:
X, y, coefs = make_regression(n_samples=100, n_features=20, noise=10, random_state=42, coef=True)
Уберем из X важный x['x1'] (столбец, равный первому признаку, если нумерация признаков с 0).
Давайте построим Ridge(random_state=1), Lasso, LinearRegression на нормированных данных.

Ответьте на вопросы 4-6.

- [ ] Коэффициенты, кроме тех, которые при x['x20'] и x['x1'], изменились существенно (в среднем более, чем на 10%)
- [ ] Lasso занулила коэффициенты для x['x20'] и x['x1']
- [x] **Ridge и LinearRegression уменьшила коэффициенты для x['x20'] и x['x1'] примерно в 2 раза, по сравнению с регрессией без x['x20']**
- [x] **Коэффициенты изменились существенно (в среднем более, чем на 10%)**
- [ ] Lasso занулила все истинно нулевые коэффициенты
- [x] **R2 по сравнению с полной регрессией (с x['x1']) упало более, чем на 25%**

Этот блок пройден, но вы можете его перепройти.
Внимание! После нажатия на кнопку текущий результат будет утерян.

Сбросить результат и попробовать еще раз

🙃 4 🥲 4 +

Конкурс "Турникеты-2.0"
Соревнование
Следующий блок →

Open ML Course: Линейные модели 2023
Главная страница трека
На главную ↑