Санкт-Петербургский политехнический университет Петра Великого

Институт машиностроения, материалов и транспорта

[**Высшая школа автоматизации и робототехники**](https://www.spbstu.ru/structure/vysshaya_shkola_avtomatizatsii_i_robototekhniki/)

**ЛАБОРАТОРНАЯ РАБОТА №3**

**«Методы регрессии»**

по дисциплине

«Математические методы интеллектуальных технологий»

Выполнил

студент гр.3331506/60401 <*подпись*> Комаров А.Е.

Руководитель

<*подпись*> Орлова С.Р.

«\_\_\_» \_\_\_\_\_\_\_\_\_\_ 2020 г.

Санкт-Петербург

2020

***Цель работы:*** освоение методов решения задачи регрессии

***Задачи работы:***

* освоить основные алгоритмы регрессии;
* изучить различные способы измерения качества регрессионной модели;
* научиться применять полученные знания об алгоритмах и навыки по формированию признаков для получения максимально качественных моделей.

# Ход работы и результаты

# Часть 1. Анализ и подготовка данных

1. Описание базы

В таблице 1 представлено описание базы *House Prices - Advanced Regression Techniques*.

Таблица 1

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Признак | Описание признака | Тип | Крайние значения | |
| Id | Номер записи | int64 | 1 | 1460 |
| SalePrice | Цена продажи объекта в долларах | int64 | 34900 | 755000 |
| MSSubClass | Тип жилья | int64 | 20 | 190 |
| MSZoning | Класс продажи | object |  |  |
| LotFrontage | Длина в футах улицы, соединенной с объетом | float64 | 21 | 313 |
| LotArea | Размер в квадратных футах | int64 | 1300 | 215245 |
| Street | Тип подъездной работы | object |  |  |
| Alley | Тип подъездной аллеи к объекту | object |  |  |
| LotShape | Общая форма объета | object |  |  |
| LandContour | Плокскостность объета | object |  |  |
| Utilities | Имеющиеся коммунальные услуги (газ, вода, электричество) | object |  |  |
| LotConfig | Конфигурация объета | object |  |  |
| LandSlope | Склон объета | object |  |  |
| Neighborhood | Район в пределах города Эймс | object |  |  |
| Condition1 | Близость к главной дорге или железной дороге | object |  |  |
| Condition2 | Близость к главной дорге или железной дороге (при наличии второй) | object |  |  |
| BldgType | Тип жилья | object |  |  |
| HouseStyle | Стиль жилья | object |  |  |
| OverallQual | Общее качество материалов и отделки | int64 | 1 | 10 |
| OverallCond | Общая оценка состояния | int64 | 1 | 9 |
| YearBuilt | Дата постройки | int64 | 1872 | 2010 |
| YearRemodAdd | Дата реконструкции | int64 | 1950 | 2010 |
| RoofStyle | Тип крыши | object |  |  |
| RoofMatl | Материал крыши | object |  |  |
| Exterior1st | Внешнее покрытие дома | object |  |  |
| Exterior2nd | Наружное покрытие дома (если используется более одного материала) | object |  |  |
| MasVnrType | Тип облицовки кладки | object |  |  |
| MasVnrArea | Площадь облицовки кладки в квадратных футах | float64 | 0 | 1600 |
| ExterQual | Качество наружных материалов | object |  |  |
| ExterCond | Текущее состояние материала снаружи | object |  |  |
| Foundation | Тип фундамента | object |  |  |
| BsmtQual | Высота подвала | object |  |  |
| BsmtCond | Общее состояние подвала | object |  |  |
| BsmtExposure | Подвальные стены на уровне сада или выхода | object |  |  |
| BsmtFinType1 | Качество готовой площади подвала | object |  |  |
| BsmtFinSF1 | Готовые квадратные футы типа 1 | int64 | 0 | 5644 |
| BsmtFinType2 | Качество второй законченной области подвала (если есть) | object |  |  |
| BsmtFinSF2 | Готовые квадратные футы типа 2 | int64 | 0 | 1474 |
| BsmtUnfSF | Незавершенные квадратные футы подвала | int64 | 0 | 2336 |
| TotalBsmtSF | Общая площадь подвала в квадратных футах | int64 | 0 | 6110 |
| Heating | Тип отопления | object |  |  |
| HeatingQC | Качество и состояние отопления | object |  |  |
| CentralAir | Наличие центральное кондиционирование | object |  |  |
| Electrical | Описание электрической системы | object |  |  |
| 1stFlrSF | Квадратные футы на первом этаже | int64 | 334 | 4692 |
| 2ndFlrSF | Квадратные футы второго этажа | int64 | 0 | 2065 |
| LowQualFinSF | Квадратные футы с отделкой низкого качества (все этажи) | int64 | 0 | 572 |
| GrLivArea | Квадратные футы жилой площади над уровнем земли | int64 | 334 | 5642 |
| BsmtFullBath | Полные ванные комнаты в подвале | int64 | 0 | 3 |
| BsmtHalfBath | Полуподвальные ванные комнаты | int64 | 0 | 2 |
| FullBath | Полные ванные комнаты над цокольным этажом | int64 | 0 | 3 |
| HalfBath | Полуванны над цокольным этажом | int64 | 0 | 2 |
| Bedroom | Количество спален над цокольным этажом | int64 | 0 | 8 |
| Kitchen | Количество кухонь | int64 | 0 | 3 |
| KitchenQual | Качество кухни | object |  |  |
| TotRmsAbvGrd | Общее количество комнат над цокольным этажом (без ванных комнат) | int64 | 2 | 14 |
| Functional | Рейтинг функциональности дома | object |  |  |
| Fireplaces | Количество каминов | int64 | 0 | 3 |
| FireplaceQu | Качество камина | object |  |  |
| GarageType | Расположение гаража | object |  |  |
| GarageYrBlt | Год постройки гаража | float64 | 1900 | 2010 |
| GarageFinish | Внутренняя отделка гаража | object |  |  |
| GarageCars | Размер гаража по вместимости машин | int64 | 0 | 4 |
| GarageArea | Размер гаража в квадратных футах | int64 | 0 | 1418 |
| GarageQual | Качество гаража | object |  |  |
| GarageCond | Состояние гаража | object |  |  |
| PavedDrive | Подъездная дорога с твердым покрытием | object |  |  |
| WoodDeckSF | Площадь деревянного настила в квадратных футах | int64 | 0 | 857 |
| OpenPorchSF | Площадь открытой веранды в квадратных футах | int64 | 0 | 547 |
| EnclosedPorch | Площадь веранды в квадратных футах | int64 | 0 | 552 |
| 3SsnPorch | Площадь веранды на три сезона в квадратных футах | int64 | 0 | 508 |
| ScreenPorch | Площадь крыльца в квадратных футах | int64 | 0 | 480 |
| PoolArea | Площадь бассейна в квадратных футах | int64 | 0 | 738 |
| PoolQC | Качество бассейна | object |  |  |
| Fence | Качество забора | object |  |  |
| MiscFeature | Прочие функции, не включенные в другие категории (лифт, сарай, теннисный корт и др.) | object |  |  |
| MiscVal | Ценность дополнительных функций | int64 | 0 | 15500 |
| MoSold | Месяц продажи | int64 | 1 | 12 |
| YrSold | Год продажи | int64 | 2006 | 2010 |
| SaleType | Тип продажи | object |  |  |
| SaleCondition | Состояние продажи | object |  |  |

1. Устранение “технических” ошибок

Первым шагом были выявлены признаки, не содержащие в себе достаточное количество информации для последующего анализа (малое количество ненулевых строк), и удалены из дальнейшего рассмотрения.

Была получена таблица с количеством не-*NA/null* записей в каждом признаке, после чего обнаружено, что 6 признаков могут быть удалены из дальнейшего рассмотрения. Ниже представлены фрагмент кода и таблица 2 – фрагмент таблицы *count.xlsx*, отсортированной по количеству не-*NA/null* записей.

data\_frame.count().to\_excel('count.xlsx')

data\_frame = data\_frame.drop(['PoolQC', 'Alley', 'FireplaceQu', 'Fence', 'MiscFeature', 'LotFrontage'], axis=1)

Таблица 2

|  |  |
| --- | --- |
| Признак | count() |
| PoolQC | 7 |
| MiscFeature | 54 |
| Alley | 91 |
| Fence | 281 |
| FireplaceQu | 770 |
| LotFrontage | 1201 |
| GarageType | 1379 |
| GarageYrBlt | 1379 |
| GarageFinish | 1379 |
| … | |

1. Устранение “смысловых” ошибок

Для устранения выбросов в числовых признаках использовался следующий фрагмент кода.

data\_frame = data\_frame[(data\_frame[["GrLivArea"]] < 4000).all(axis=1)]  
data\_frame = data\_frame[(data\_frame[["GarageCars"]] < 4).all(axis=1)]  
data\_frame = data\_frame[(data\_frame[["OverallQual"]] > 2).all(axis=1)]  
data\_frame = data\_frame[(data\_frame[["1stFlrSF"]] < 2000).all(axis=1)]  
data\_frame = data\_frame[(data\_frame[["GrLivArea"]] < 2500).all(axis=1)]  
data\_frame = data\_frame[(data\_frame[["TotalBsmtSF"]] < 2000).all(axis=1)]  
data\_frame = data\_frame[(data\_frame[["TotalBsmtSF"]] > 450).all(axis=1)]  
data\_frame = data\_frame[(data\_frame[["GarageArea"]] < 1000).all(axis=1)]

После устранения выбросов было произведено удаление всех строк, содержащих *NA/null* записи, и обновление номеров строк, что отражает представленный ниже фрагмент кода.

data\_frame = data\_frame.dropna()  
data\_frame.reset\_index(drop=True)

Далее был произведен перевод категориальных признаков в числовые.

for feature in data\_frame.columns:  
 if data\_frame[feature].dtype == object:  
 data\_frame[feature] = pandas.Categorical(data\_frame[feature])  
 sex\_map\_train = dict(zip(data\_frame[feature].cat.codes, data\_frame[feature]))  
 data\_frame[feature] = data\_frame[feature].cat.codes

Из-за большого количества признаков в базе на рисунках 1-3 представлены гистограммы только тех признаков, которые были выбраны в дальнейшем для анализа.

|  |  |
| --- | --- |
|  |  |
|  |  |
|  |  |
| Рисунок 1 | |

|  |  |
| --- | --- |
|  |  |
|  |  |
|  |  |
| Рисунок 2 | |

|  |  |
| --- | --- |
|  |  |
|  | |
| Рисунок 3 | |

1. Определение целевой переменной и итогового набора данных

В качестве целевой переменной выбрана цена продажи объекта в долларах.

Для определения итогового набора данных были построены матрицы корреляции признаков между собой. Вследствие большого количества признаков и следующей из этого сложности анализа корреляций признаков с целевой переменной на основании графиков, были найдены корреляции всех признаков с ценой продажи объекта с помощью фрагмента кода, представленного ниже.

corr = data\_frame[['SalePrice'] + data\_frame.columns.to\_list()].corr().iloc[0]  
corr.sort\_values().to\_excel('correlation.xlsx')

На основании полученных данных были выбраны для дальнейшего анализа признаки с большими отрицательными и положительными значениями корреляции. В таблице 3 представлены значения корреляции выбранных признаков (выделены жирным шрифтом).

Таблица 3

|  |  |  |  |
| --- | --- | --- | --- |
| Признак | Корреляция | Признак | Корреляция |
| **ExterQual** | -0,63888 | GarageYrBlt | 0,532007 |
| **BsmtQual** | -0,60824 | **FullBath** | 0,54562 |
| **KitchenQual** | -0,58622 | YearRemodAdd | 0,553244 |
| **GarageFinish** | -0,53602 | **1stFlrSF** | 0,571015 |
| **HeatingQC** | -0,43744 | **YearBuilt** | 0,588213 |
| **GarageType** | -0,4121 | **TotalBsmtSF** | 0,598793 |
| BsmtExposure | -0,28911 | **GarageArea** | 0,599214 |
| … | | **GarageCars** | 0,637872 |
| Fireplaces | 0,41244 | **GrLivArea** | 0,66375 |
| **TotRmsAbvGrd** | 0,438394 | **OverallQual** | 0,785075 |
| Foundation | 0,509838 | SalePrice | 1 |

1. Создание генератора разбиений с параметрами

kf = KFold(n\_splits=5, shuffle=True, random\_state=12)

# Часть 2. Регрессия

1. Используемые метрики

В качестве метрик используются средняя абсолютная ошибка и среднеквадратичная ошибка.

Средняя абсолютная ошибка – метрика риска, соответствующая ожидаемому значению потери абсолютной ошибки. Формула расчета средней абсолютной ошибки представлена на рисунке 4.



Рисунок 4

Среднеквадратичная ошибка – метрика риска, соответствующая ожидаемому значению квадратичной ошибки или убытка. Формула расчета среднеквадратичной ошибки представлена на рисунке 5.

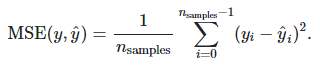


Рисунок 5

Выбор представленных метрик обусловлен тем, что среднеквадратичная ошибка чувствительна к далеким промахам в предсказаниях (выбросам) из-за квадрата ошибки, в то время как средняя абсолютная ошибка учитывает модуль ошибки. Таким образом, оптимизируя по этим метрикам, можно добиться баланса между устранением сильных промахов в предсказаниях (что может быть признаком переобучения) и построением качественной модели.

1. Решение задачи регрессии

Под задачей классификации понимаем определение стоимости жилья на основе объявлений о продаже дома.

В таблице 4 представлены результаты работы классификации следующими методами:

* *Elasctic-Net*;
* Полиномиальная регрессия (*Polynomial Features*);
* *Kernel Ridge*.

Метод *Kernel Ridge (KRR)* регрессии сочетает в себе гребневую регрессию с ядерным трюком. Таким образом, данный метод использует линейную функцию регрессии в пространстве, индуцированную соответствующим ядром и данными. Регрессионная модель идентична модели, используемой в векторной регрессии, но используются другие функции потерь. *KRR* использует функцию квадратичной потери ошибок совместно с регуляризацией *l2*. Модель *KRR* может быть инициализирована в закрытой форме и быстрее, чем *SVR* для данных среднего размера.

В приложении 1 представлен вывод *GridSearchCV*, на основании которого были выбраны оптимальные параметры для всех методов регрессии. В качестве исходных данных использовались нормализованные, стандартизованные и оригинальные данные, в качестве метрик, по которым производилась оптимизация, использовались все три выбранные метрики: метрика ,средняя абсолютная ошибка и среднеквадратичная ошибка.

1. Результаты регрессии

Таблица 4

|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **it** | **type** | **ElasticNet** | | | **PolynomialFeatures** | | | **KernelRidge** | | |
| **r2** | **MSE** | **MAE** | **r2** | **MSE** | **MAE** | **r2** | **MSE** | **MAE** |
| **0** | **original** | 0,82 | 607584063 | 17886 | 0,81 | 640320931 | 18036 | 0,82 | 597393833 | 16973 |
| **standardization** | 0,82 | 609197946 | 17857 | 0,81 | 633474975 | 17862 | 0,83 | 567343915 | 16564 |
| **normalization** | 0,31 | 2341097512 | 35990 | 0,80 | 666917681 | 18592 | 0,33 | 2257050581 | 35081 |
| **1** | **original** | 0,77 | 862162763 | 21115 | 0,77 | 869769199 | 20396 | 0,79 | 805222120 | 19469 |
| **standardization** | 0,77 | 868158598 | 21136 | 0,77 | 861170860 | 20278 | 0,79 | 796758489 | 19235 |
| **normalization** | 0,29 | 2663576226 | 39332 | 0,76 | 887473034 | 20453 | 0,31 | 2584889779 | 38539 |
| **2** | **original** | 0,83 | 678869639 | 19033 | 0,84 | 651707349 | 17988 | 0,86 | 589816181 | 16350 |
| **standardization** | 0,84 | 669478816 | 18892 | 0,84 | 651707349 | 17988 | 0,87 | 514069057 | 16109 |
| **normalization** | 0,28 | 2954305410 | 38433 | 0,85 | 632630168 | 17753 | 0,30 | 2868493924 | 37634 |
| **3** | **original** | 0,82 | 807745288 | 20899 | 0,82 | 777345038 | 19972 | 0,83 | 734663256 | 19674 |
| **standardization** | 0,82 | 808136651 | 20809 | 0,82 | 782057435 | 20063 | 0,84 | 698023252 | 19073 |
| **normalization** | 0,28 | 3183256488 | 43694 | 0,83 | 774486766 | 20220 | 0,30 | 3087915921 | 42895 |
| **4** | **original** | 0,84 | 490802595 | 16450 | 0,80 | 634266874 | 17923 | 0,83 | 538851108 | 16467 |
| **standardization** | 0,84 | 493360267 | 16426 | 0,80 | 619470180 | 17733 | 0,86 | 423293735 | 14982 |
| **normalization** | 0,32 | 2098021646 | 35285 | 0,79 | 643867972 | 18083 | 0,35 | 2016151418 | 34349 |

Как видно из таблицы 4, методы *Elasctic-Net* и *KRR* плохо работают с нормализованными данными, в то время как полиномиальная регрессия работает примерно одинаково со всеми типами данных. Результаты работы *Elasctic-Net* и полиномиальной регрессии на оригинальных и стандартизованных данных примерно одинаков, в то время как метод *KRR* работает на этих данных немного лучше остальных. Лучшем же методом регрессии для решения поставленной задачи определения стоимости жилья можно назвать метод *KRR* при работе со стандартизованными данными, что отражено во всех трех метриках (максимальное значение метрики , минимальные значения средней абсолютной ошибки и среднеквадратичной ошибки).

1. Время обучения и время предсказания

В таблице 5 представлены результаты измерения времени обучения и времени предсказания для всех трех рассматриваемых методов регрессии: *Elasctic-Net*, полиномиальная регрессия, *Kernel Ridge*.

Таблица 5

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **it** | **type** | **ElasticNet** | | **PolynomialFeatures** | | **KernelRidge** | |
| Время обучения | Время предска-зания | Время обучения | Время предска-зания | Время обучения | Время предска-зания |
| **0** | **original** | 0,0035 | 0,00017 | 0,031 | 0,00081 | 0,044 | 0,0019 |
| **standardization** | 0,0041 | 0,00013 | 0,028 | 0,00073 | 0,037 | 0,0019 |
| **normalization** | 0,0011 | 0,00010 | 0,030 | 0,00084 | 0,044 | 0,0020 |
| **1** | **original** | 0,0055 | 0,00011 | 0,028 | 0,00084 | 0,041 | 0,0019 |
| **standardization** | 0,0036 | 0,00013 | 0,028 | 0,00080 | 0,051 | 0,0020 |
| **normalization** | 0,0010 | 0,00013 | 0,028 | 0,00085 | 0,040 | 0,0019 |
| **2** | **original** | 0,0040 | 0,00012 | 0,031 | 0,00092 | 0,040 | 0,0018 |
| **standardization** | 0,0032 | 0,00015 | 0,035 | 0,00099 | 0,042 | 0,0021 |
| **normalization** | 0,0008 | 0,00011 | 0,028 | 0,00079 | 0,041 | 0,0018 |
| **3** | **original** | 0,0043 | 0,00014 | 0,028 | 0,00077 | 0,038 | 0,0019 |
| **standardization** | 0,0030 | 0,00012 | 0,030 | 0,00090 | 0,041 | 0,0019 |
| **normalization** | 0,0009 | 0,00011 | 0,030 | 0,00123 | 0,039 | 0,0019 |
| **4** | **original** | 0,0032 | 0,00010 | 0,028 | 0,00075 | 0,038 | 0,0022 |
| **standardization** | 0,0033 | 0,00011 | 0,030 | 0,00102 | 0,041 | 0,0020 |
| **normalization** | 0,0009 | 0,00014 | 0,030 | 0,00088 | 0,041 | 0,0023 |

По времени обучения на порядок меньшее значение предоставляет метод регрессии *Elasctic-Net*. Также стоит отметить, что наибольшее значение времени предсказания соответствует методу *KRR*, что вероятно объясняется необходимостью проведения ядерного трюка в том числе и при обучении.

# Заключение

В настоящей работе использовалась та же база, что и в лабораторной работе 2. Все подготовительные работы с базой оставлены прежними за исключением разбиения цен продажи домов из объявлений на категории.

Изначально предпринималась попытка отображения набора точек и графика полинома регрессионной модели. Стоит заметить, что для этого был применен метод главных компонент, сжимающий таблицу из пятнадцати отобранных признаков в один, и потеря данных имела весомое значение. Но тем не менее, по этому действию удалось установить, что набор данных стремиться к линейной зависимости.

В качестве метрик оценки были применены метрика , средняя квадратичная ошибка и средняя абсолютная ошибка. Стоит заметить, что оценка по средней квадратичной ошибке была неинформативна, так как в базе признаков попадались выбросы.

Как было сказано ранее, при попытки визуализировать базу данных было замечено стремление модели к линейной зависимости. При тестировании полиноминальный регрессии и подборе параметров полным перебором при помощи функции *GridSearchCV* было обнаружено, что наилучший результат достигался при степени полинома равной 1 или 2.

Среднее значение коэффициента детерминации равняется 0.8. Для нормализованных данных получились наихудшие результаты. Средняя абсолютная ошибка находится около 17000, что в измерении цен сотнями тысяч можно считать приемлемым.

Приложение 1

type = original methot = ElasticNet() metric: r2

{'alpha': 0.05, 'fit\_intercept': 'False'}

0.801992027720965

type = original methot = ElasticNet() metric: neg\_mean\_absolute\_error

{'alpha': 0.4, 'fit\_intercept': 'False'}

-19640.59265159963

type = original methot = ElasticNet() metric: neg\_mean\_squared\_error

{'alpha': 0.05, 'fit\_intercept': 'False'}

-740463113.2191767

type = original methot = Pipeline(steps=[('polynomialfeatures', PolynomialFeatures()),

('linearregression', LinearRegression())]) metric: r2

{'linearregression\_\_fit\_intercept': True, 'linearregression\_\_normalize': False, 'polynomialfeatures\_\_degree': 2}

0.8049086859071225

type = original methot = Pipeline(steps=[('polynomialfeatures', PolynomialFeatures()),

('linearregression', LinearRegression())]) metric: neg\_mean\_absolute\_error

{'linearregression\_\_fit\_intercept': True, 'linearregression\_\_normalize': False, 'polynomialfeatures\_\_degree': 2}

-19191.009752136746

type = original methot = Pipeline(steps=[('polynomialfeatures', PolynomialFeatures()),

('linearregression', LinearRegression())]) metric: neg\_mean\_squared\_error

{'linearregression\_\_fit\_intercept': True, 'linearregression\_\_normalize': False, 'polynomialfeatures\_\_degree': 2}

-731090409.0334231

type = original methot = KernelRidge() metric: r2

{'alpha': 4, 'degree': 2, 'kernel': 'polynomial'}

0.8108993010043338

type = original methot = KernelRidge() metric: neg\_mean\_absolute\_error

{'alpha': 2, 'degree': 2, 'kernel': 'polynomial'}

-18914.2320723009

type = original methot = KernelRidge() metric: neg\_mean\_squared\_error

{'alpha': 4, 'degree': 2, 'kernel': 'polynomial'}

-707685603.5701972

type = standardization methot = ElasticNet() metric: r2

{'alpha': 0.05, 'fit\_intercept': 'False'}

0.8017061261913249

type = standardization methot = ElasticNet() metric: neg\_mean\_absolute\_error

{'alpha': 0.25, 'fit\_intercept': 'False'}

-19737.393695039576

type = standardization methot = ElasticNet() metric: neg\_mean\_squared\_error

{'alpha': 0.05, 'fit\_intercept': 'False'}

-741848746.2148327

type = standardization methot = Pipeline(steps=[('polynomialfeatures', PolynomialFeatures()),

('linearregression', LinearRegression())]) metric: r2

{'linearregression\_\_fit\_intercept': True, 'linearregression\_\_normalize': False, 'polynomialfeatures\_\_degree': 2}

0.8049086859176194

type = standardization methot = Pipeline(steps=[('polynomialfeatures', PolynomialFeatures()),

('linearregression', LinearRegression())]) metric: neg\_mean\_absolute\_error

{'linearregression\_\_fit\_intercept': True, 'linearregression\_\_normalize': False, 'polynomialfeatures\_\_degree': 2}

-19191.009753006172

type = standardization methot = Pipeline(steps=[('polynomialfeatures', PolynomialFeatures()),

('linearregression', LinearRegression())]) metric: neg\_mean\_squared\_error

{'linearregression\_\_fit\_intercept': True, 'linearregression\_\_normalize': False, 'polynomialfeatures\_\_degree': 2}

-731090408.9902557

type = standardization methot = KernelRidge() metric: r2

{'alpha': 5, 'degree': 2, 'kernel': 'polynomial'}

0.8178679368414786

type = standardization methot = KernelRidge() metric: neg\_mean\_absolute\_error

{'alpha': 3, 'degree': 3, 'kernel': 'polynomial'}

-18671.54468519874

type = standardization methot = KernelRidge() metric: neg\_mean\_squared\_error

{'alpha': 5, 'degree': 2, 'kernel': 'polynomial'}

-682162007.7503682

type = normalization methot = ElasticNet() metric: r2

{'alpha': 0.05, 'fit\_intercept': 'False'}

0.0015042929755870738

type = normalization methot = ElasticNet() metric: neg\_mean\_absolute\_error

{'alpha': 0.05, 'fit\_intercept': 'False'}

-46842.534386660634

type = normalization methot = ElasticNet() metric: neg\_mean\_squared\_error

{'alpha': 0.05, 'fit\_intercept': 'False'}

-3753954600.3865104

type = normalization methot = Pipeline(steps=[('polynomialfeatures', PolynomialFeatures()),

('linearregression', LinearRegression())]) metric: r2

{'linearregression\_\_fit\_intercept': True, 'linearregression\_\_normalize': False, 'polynomialfeatures\_\_degree': 2}

0.7599197116332904

type = normalization methot = Pipeline(steps=[('polynomialfeatures', PolynomialFeatures()),

('linearregression', LinearRegression())]) metric: neg\_mean\_absolute\_error

{'linearregression\_\_fit\_intercept': True, 'linearregression\_\_normalize': False, 'polynomialfeatures\_\_degree': 2}

-21322.133825944173

type = normalization methot = Pipeline(steps=[('polynomialfeatures', PolynomialFeatures()),

('linearregression', LinearRegression())]) metric: neg\_mean\_squared\_error

{'linearregression\_\_fit\_intercept': True, 'linearregression\_\_normalize': False, 'polynomialfeatures\_\_degree': 2}

-902995249.6067324

type = normalization methot = KernelRidge() metric: r2

{'alpha': 0.001, 'kernel': 'laplacian'}

0.6909894209961208

type = normalization methot = KernelRidge() metric: neg\_mean\_absolute\_error

{'alpha': 0.01, 'kernel': 'laplacian'}

-24155.021384514243

type = normalization methot = KernelRidge() metric: neg\_mean\_squared\_error

{'alpha': 0.001, 'kernel': 'laplacian'}

-1157365605.4649327