

Линейная регрессия и sklearn

В этой домашней работе:

- Обучим линейную регрессию для предсказания цены дома;
- Научимся работать с разными типами признаков;
- Поймем, в чем отличие между разными регуляризаторами;
- Научимся пользоваться основными инструментами в `sklearn` : моделями, трансформерами и pipeline;
- Обсудим преобразования признаков и целевой переменной, которые могут помочь в обучении линейных моделей.

Скачайте тренировочную и тестовую выборку из соревнования на kaggle: [House Prices: Advanced Regression Techniques](#).

Разместите данные рядом с тетрадкой или поправьте пути при их чтении.

```
In [255... # Чтобы не перегружать ячейки с кодом, вынесем сюда все импорты

import warnings

import matplotlib.pyplot as plt
import numpy as np
import pandas as pd
import seaborn as sns
from sklearn.compose import ColumnTransformer
from sklearn.linear_model import Lasso, LinearRegression, Ridge
from sklearn.model_selection import GridSearchCV, cross_val_score, train_test_split
from sklearn.pipeline import Pipeline
from sklearn.preprocessing import KBinsDiscretizer, OneHotEncoder, PolynomialFeatures, StandardScaler

sns.set_theme(style="darkgrid")
warnings.simplefilter("ignore")
%matplotlib inline
```

Часть 0. Введение в линейные модели

Напомним, что линейная регрессия — это модель следующего вида:

$$a(x) = \langle w, x \rangle + w_0$$

где $w \in \mathbb{R}^d$, $w_0 \in \mathbb{R}$. Обучить линейную регрессию — значит найти w и w_0 .

В машинном обучении часто говорят об *обобщающей способности модели*, то есть о способности модели работать на новых, тестовых данных хорошо. Если модель будет идеально предсказывать выборку, на которой она обучалась, но при этом просто ее запомнит, не "вытащив" из данных никакой закономерности, от нее будет мало толку. Такую модель называют *переобученной*: она слишком подстроилась под обучающие примеры, не выявив никакой полезной закономерности, которая позволила бы ей совершать хорошие предсказания на данных, которые она ранее не видела.

Рассмотрим следующий пример, на котором будет хорошо видно, что значит переобучение модели. Для этого нам понадобится сгенерировать синтетические данные. Рассмотрим зависимость

$$y(x) = \cos(1.5\pi x)$$

y — целевая переменная, а x — объект (просто число от 0 до 1). В жизни мы наблюдаем какое-то конечное количество пар объект-таргет, поэтому смоделируем это, взяв 30 случайных точек x_i в отрезке $[0; 1]$. Более того, в реальной жизни целевая переменная может быть зашумленной (измерения в жизни не всегда точны), смоделируем это, зашумив значение функции нормальным шумом: $\tilde{y}_i = y(x_i) + \mathcal{N}(0, 0.01)$.

Попытаемся обучить три разных линейных модели: признаки для первой — $\{x\}$, для второй — $\{x, x^2, x^3, x^4\}$, для третьей — $\{x, \dots, x^{20}\}$.

```
In [256... np.random.seed(36)
x = np.linspace(0, 1, 100)
y = np.cos(1.5 * np.pi * x)

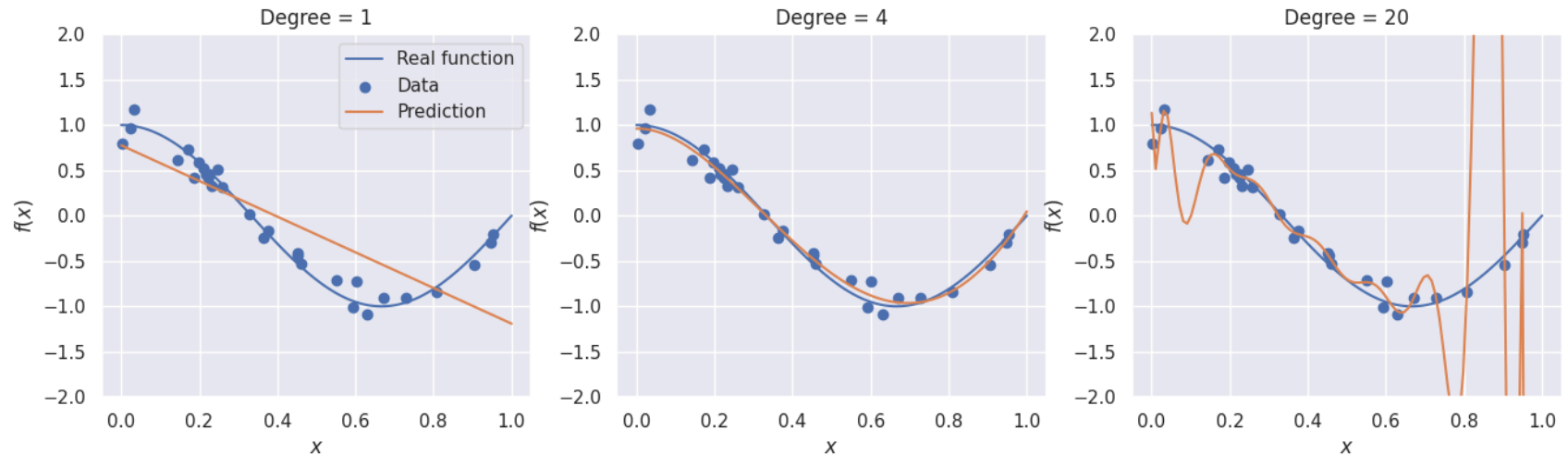
x_objects = np.random.uniform(0, 1, size=30)
y_objects = np.cos(1.5 * np.pi * x_objects) + np.random.normal(scale=0.1, size=x_objects.shape)

fig, axs = plt.subplots(figsize=(16, 4), ncols=3)
for i, degree in enumerate([1, 4, 20]):
    X_objects = PolynomialFeatures(degree, include_bias=False).fit_transform(x_objects[:, None])
    X = PolynomialFeatures(degree, include_bias=False).fit_transform(x[:, None])
    regr = LinearRegression().fit(X_objects, y_objects)
    y_pred = regr.predict(X)
    axs[i].plot(x, y, label="Real function")
```

```

axs[i].scatter(x_objects, y_objects, label="Data")
axs[i].plot(x, y_pred, label="Prediction")
if i == 0:
    axs[i].legend()
axs[i].set_title("Degree = %d" % degree)
axs[i].set_xlabel("$x$")
axs[i].set_ylabel("$f(x)$")
axs[i].set_ylim(-2, 2)

```



Вопрос 1: Почему первая модель получилась плохой, а третья переобучилась?

Ответ: Первая модель не может с помощью линейной функции отображать нелинейную зависимость. Третья смогла слишком хорошо подстроиться и пересечь все точки, но не отображает зависимости.

Чтобы избежать переобучения, модель регуляризуют. Обычно переобучения в линейных моделях связаны с большими весами, а поэтому модель часто штрафуют за большие значения весов, добавляя к функционалу качества, например, квадрат ℓ^2 -нормы вектора w :

$$Q_{reg}(X, y, a) = Q(X, y, a) + \lambda \|w\|_2^2$$

Это слагаемое называют ℓ_2 -регуляризатором, а коэффициент λ — коэффициентом регуляризации.

Вопрос 2: Почему большие веса в линейной модели — плохо?

Ответ: Большой вес фичи значит что минимальное ее изменение повлечет за собой большое изменение целевой функции

Вопрос 3: Почему регуляризовать w_0 — плохая идея?

Ответ: w_0 отвечает за изначальный сдвиг. Он влияет на все данные одинаково (не зависит ни от какой фичи) и не способствует переобучению. Его регуляризация сделает ответ неверным. Например для предсказания цены квартиры w_0 может быть большим (квартира не может быть бесплатной, а при нулевых значениях всех фич мы получим w_0). При его регуляризации получим нереалистично дешевые квартиры

Вопрос 4: На что влияет коэффициент λ ? Что будет происходить с моделью, если λ начать уменьшать? Что будет, если λ сделать слишком большим?

Ответ: λ отвечает за то, на сколько "плохо" иметь большие веса. $\lambda = 0$ Полностью уберет штраф. Маленькое λ будет разрешать большие веса, а большое λ будет их запрещать.

Часть 1. Загружаем данные

```
In [257... train_data = pd.read_csv("train.csv")
train_data.head()
```

```
Out[257... 
```

	Id	MSSubClass	MSZoning	LotFrontage	LotArea	Street	Alley	LotShape	LandContour	Utilities	...	PoolArea	PoolQC
0	1	60	RL	65.0	8450	Pave	NaN	Reg	Lvl	AllPub	...	0	NaN
1	2	20	RL	80.0	9600	Pave	NaN	Reg	Lvl	AllPub	...	0	NaN
2	3	60	RL	68.0	11250	Pave	NaN	IR1	Lvl	AllPub	...	0	NaN
3	4	70	RL	60.0	9550	Pave	NaN	IR1	Lvl	AllPub	...	0	NaN
4	5	60	RL	84.0	14260	Pave	NaN	IR1	Lvl	AllPub	...	0	NaN

5 rows × 81 columns



```
In [258... train_data.shape
```

Out[258... (1460, 81)

```
In [259... train_data.columns
```

```
Out[259... Index(['Id', 'MSSubClass', 'MSZoning', 'LotFrontage', 'LotArea', 'Street',  
        'Alley', 'LotShape', 'LandContour', 'Utilities', 'LotConfig',  
        'LandSlope', 'Neighborhood', 'Condition1', 'Condition2', 'BldgType',  
        'HouseStyle', 'OverallQual', 'OverallCond', 'YearBuilt', 'YearRemodAdd',  
        'RoofStyle', 'RoofMatl', 'Exterior1st', 'Exterior2nd', 'MasVnrType',  
        'MasVnrArea', 'ExterQual', 'ExterCond', 'Foundation', 'BsmtQual',  
        'BsmtCond', 'BsmtExposure', 'BsmtFinType1', 'BsmtFinSF1',  
        'BsmtFinType2', 'BsmtFinSF2', 'BsmtUnfSF', 'TotalBsmtSF', 'Heating',  
        'HeatingQC', 'CentralAir', 'Electrical', '1stFlrSF', '2ndFlrSF',  
        'LowQualFinSF', 'GrLivArea', 'BsmtFullBath', 'BsmtHalfBath', 'FullBath',  
        'HalfBath', 'BedroomAbvGr', 'KitchenAbvGr', 'KitchenQual',  
        'TotRmsAbvGrd', 'Functional', 'Fireplaces', 'FireplaceQu', 'GarageType',  
        'GarageYrBlt', 'GarageFinish', 'GarageCars', 'GarageArea', 'GarageQual',  
        'GarageCond', 'PavedDrive', 'WoodDeckSF', 'OpenPorchSF',  
        'EnclosedPorch', '3SsnPorch', 'ScreenPorch', 'PoolArea', 'PoolQC',  
        'Fence', 'MiscFeature', 'MiscVal', 'MoSold', 'YrSold', 'SaleType',  
        'SaleCondition', 'SalePrice'],  
        dtype='object')
```

Первое, что стоит заметить — у нас в данных есть уникальное для каждого объекта поле `id`. Обычно такие поля только мешают и способствуют переобучению. Удалим это поле из данных.

Выделим валидационную выборку, а также отделим значения целевой переменной от данных.

Вопрос 1: Почему поля типа `id` могут вызвать переобучение модели (не обязательно линейной)?

Ответ: `id` это выдуманное значение, никак не влияющее на цену дома. Оно может только запутать модель и создать не верную зависимость.

Вопрос 2: Почему стоит дополнительно отделять валидационную выборку?

Ответ: Подобрать лучшие гиперпараметры. А на тестовой выборке уже замерять результат лучших гиперпараметров.

Вопрос 3: Обратите внимание на фиксацию `random_state` при сплите данных. Почему это важно?

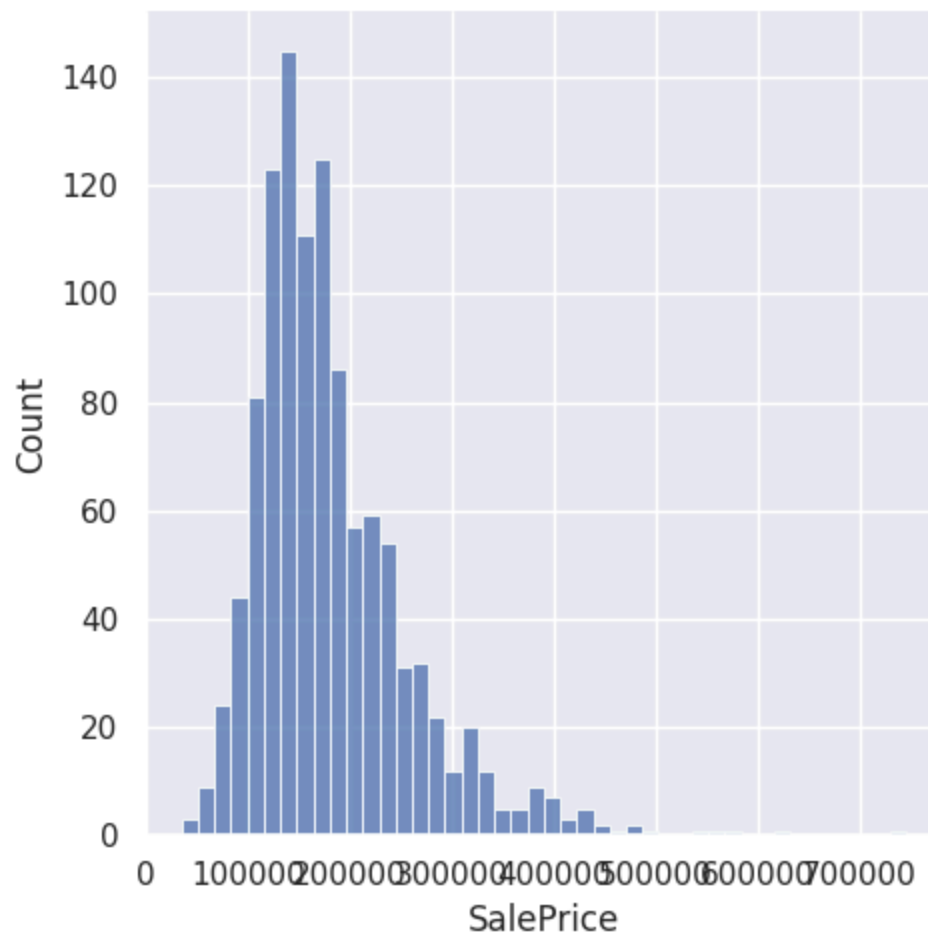
Ответ: Так мы имеем возможность воспроизвести "случайное" перемешивание данных.

```
In [260...] train_data_no_id = train_data.drop(columns="Id")
X = train_data_no_id.drop(columns="SalePrice")
y = train_data_no_id["SalePrice"]
X_train, X_val, y_train, y_val = train_test_split(X, y, random_state=42)
```

Посмотрим сначала на значения целевой переменной.

```
In [261...] sns.displot(y_train)
```

```
Out[261...] <seaborn.axisgrid.FacetGrid at 0x74726b4b3d10>
```



Судя по гистограмме, у нас есть примеры с нетипично большой стоимостью, что может помешать нам, если наша функция потерь слишком чувствительна к выбросам. В дальнейшем мы рассмотрим способы, как минимизировать ущерб от этого.

Так как для решения нашей задачи мы бы хотели обучить линейную регрессию, было бы хорошо найти признаки, "наиболее линейно" связанные с целевой переменной, иначе говоря, посмотреть на коэффициент корреляции Пирсона между признаками и целевой переменной. Заметим, что не все признаки являются числовыми, пока что мы не будем рассматривать такие признаки.

Вопрос: Что означает, что коэффициент корреляции Пирсона между двумя случайными величинами равен 1? -1? 0?

Ответ: $\rho = 1$: Существует точная и прямая линейная зависимость. $\rho = 0$: Не существует линейной зависимости. $\rho = -1$: Существует точная и обратная линейная зависимость

```
In [262... numeric_data = X_train.select_dtypes([np.number])
numeric_features = numeric_data.columns
numeric_data.head()
```

```
Out[262...      MSSubClass  LotFrontage  LotArea  OverallQual  OverallCond  YearBuilt  YearRemodAdd  MasVnrArea  BsmtFinSF1  Bsm
```

1023	120	43.0	3182	7	5	2005	2006	14.0	16
810	20	78.0	10140	6	6	1974	1999	99.0	663
1384	50	60.0	9060	6	5	1939	1950	0.0	204
626	20	NaN	12342	5	5	1960	1978	0.0	0
813	20	75.0	9750	6	6	1958	1958	243.0	608

5 rows × 36 columns

Заметим, что в данных присутствуют пропуски, заполним их средним значением по признаку.

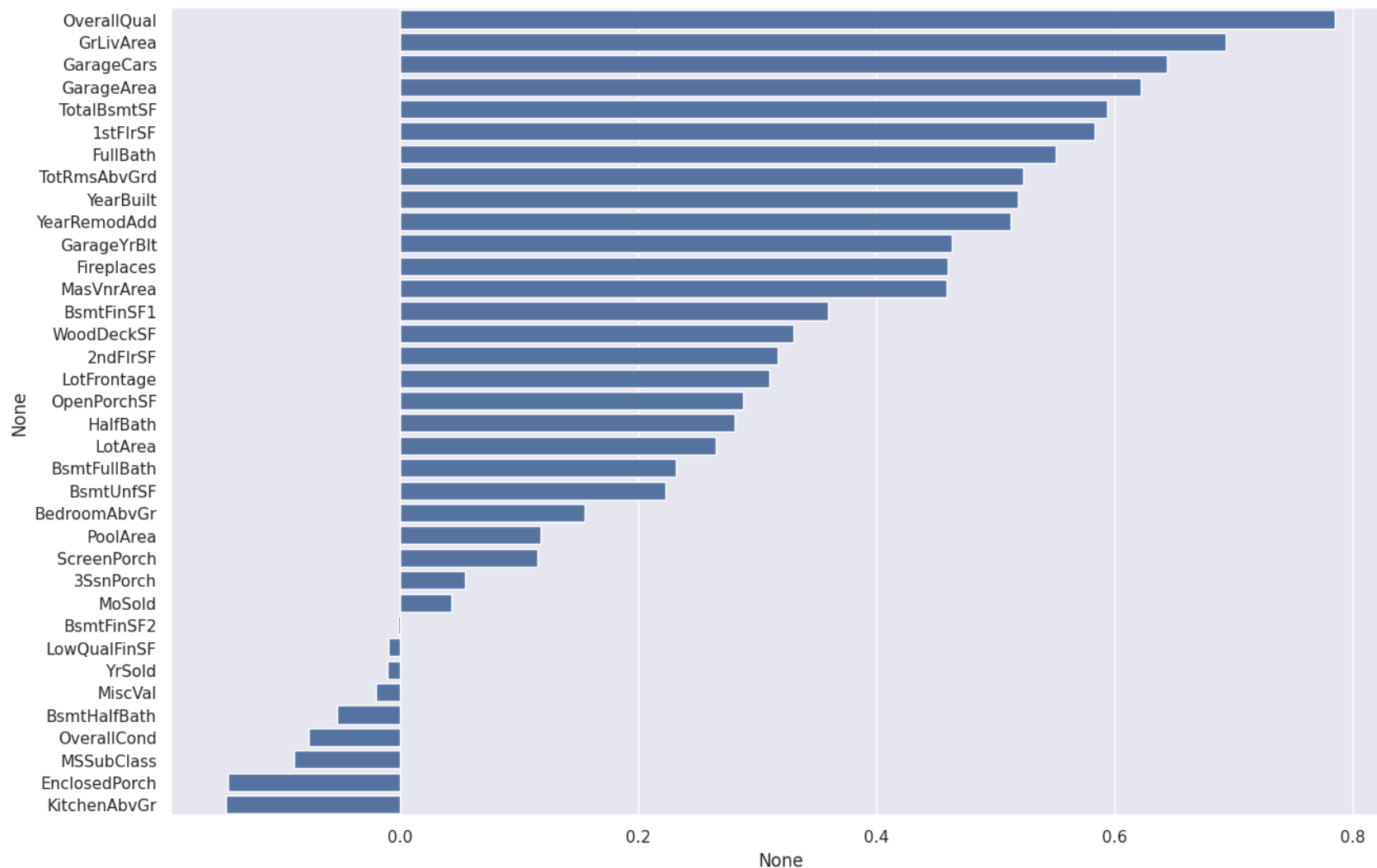
Вопрос: Как правильно заполнять пропуски для валидационной и тестовой выборки?

Ответ: Для валидационной и тестовой выборки пропуски нужно заполнять средними значениями, вычисленными на основе тренировочной выборки.

```
In [263... X_train_mean = X_train[numeric_features].mean()
X_train[numeric_features] = X_train[numeric_features].fillna(X_train_mean)
X_val[numeric_features] = X_val[numeric_features].fillna(X_train_mean)
```

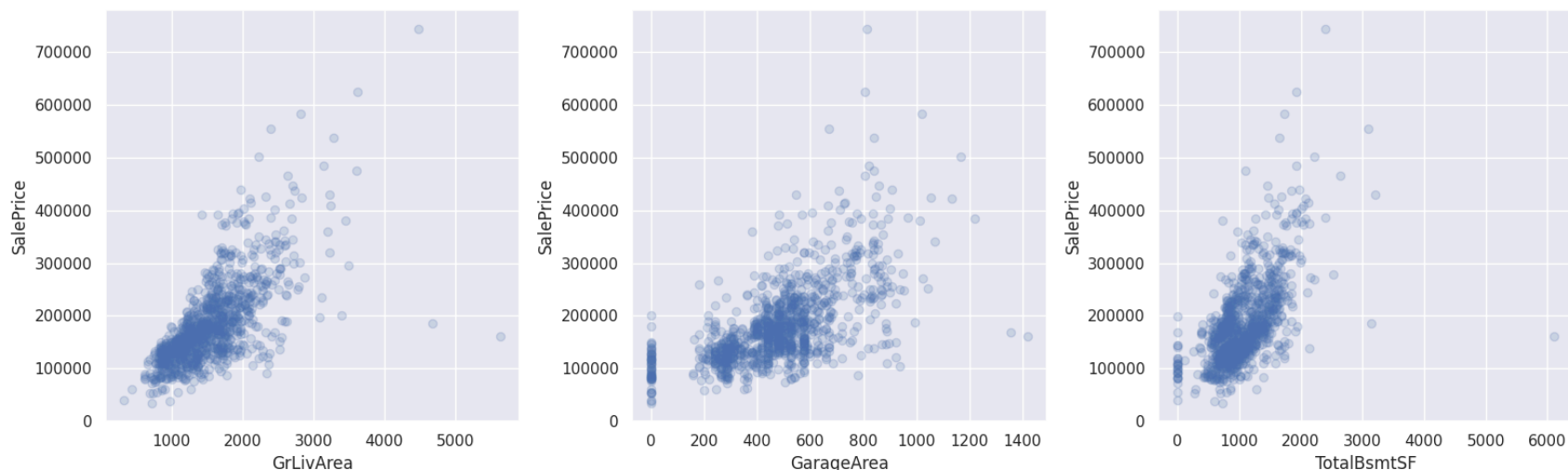
```
In [264... correlations = X_train[numeric_features].corrwith(y_train).sort_values(ascending=False)

plot = sns.barplot(y=correlations.index, x=correlations)
plot.figure.set_size_inches(15, 10)
```



Посмотрим на признаки из начала списка. Для этого нарисуем график зависимости целевой переменной от каждого из признаков. На этом графике каждая точка соответствует паре признак-таргет (такие графики называются scatter-plot).

```
In [265... fig, axs = plt.subplots(figsize=(16, 5), ncols=3)
for i, feature in enumerate(["GrLivArea", "GarageArea", "TotalBsmtSF"]):
    axs[i].scatter(X_train[feature], y_train, alpha=0.2)
    axs[i].set_xlabel(feature)
    axs[i].set_ylabel("SalePrice")
plt.tight_layout()
```



Видим, что между этими признаками и целевой переменной действительно наблюдается линейная зависимость.

Часть 2. Первая модель

Немного об обучении моделей. В арсенале ML-специалиста кроме `pandas` и `matplotlib` должны быть библиотеки, позволяющие обучать модели. Для простых моделей (линейные модели, решающее дерево, ...) отлично подходит `sklearn`: в нем очень понятный и простой интерфейс. Несмотря на то, что в `sklearn` есть реализация бустинга и простых нейронных сетей, ими все же не пользуются и предпочитают специализированные библиотеки: `XGBoost`, `LightGBM` и пр. для градиентного бустинга над деревьями, `PyTorch`, и пр. для нейронных сетей. Так как мы будем обучать линейную регрессию, нам подойдет реализация из `sklearn`.

Попробуем обучить линейную регрессию на числовых признаках из нашего датасета. В `sklearn` есть несколько классов, реализующих линейную регрессию:

- `LinearRegression` — "классическая" линейная регрессия с оптимизацией MSE. Веса находятся как точное решение: $w^* = (X^T X)^{-1} X^T y$
- `Ridge` — линейная регрессия с оптимизацией MSE и ℓ_2 -регуляризацией
- `Lasso` — линейная регрессия с оптимизацией MSE и ℓ_1 -регуляризацией

У моделей из `sklearn` есть методы `fit` и `predict`. Первый принимает на вход обучающую выборку и вектор целевых переменных и обучает модель, второй, будучи вызванным после обучения модели, возвращает предсказание на выборке. Попробуем обучить нашу первую модель на числовых признаках, которые у нас сейчас есть:

```
In [266... model = Ridge()
model.fit(X_train[numeric_features], y_train)
y_pred = model.predict(X_val[numeric_features])
y_train_pred = model.predict(X_train[numeric_features])
```

Стандартный способ оценить качество регрессии — **MSE**

```
In [267... def mean_squared_error(y_true: np.ndarray, y_pred: np.ndarray, squared: bool = True) -> float:
    """Calculate Mean Squared Error


    Args:
        y_true: array with ground truth values, [n_samples,]
        y_pred: array with predicted values, [n_samples,]
        squared: whether to return squared MSE or not

    Returns:
        number, calculated error
    """
    mse = np.mean((y_true - y_pred) ** 2)
    if squared:
        return mse
    return mse ** 0.5
```

```
In [268... print("Val RMSE = %.4f" % mean_squared_error(y_val, y_pred, squared=False))
print("Train RMSE = %.4f" % mean_squared_error(y_train, y_train_pred, squared=False))
```

Val RMSE = 35057.1670
Train RMSE = 34459.1400

Мы обучили первую модель и даже посчитали ее качество на отложенной выборке! Давайте теперь посмотрим на то, как можно оценить качество модели с помощью кросс-валидации. Принцип кросс-валидации изображен на рисунке

 No description has been provided for this image

При кросс-валидации мы делим обучающую выборку на n частей (fold). Затем мы обучаем n моделей: каждая модель обучается при отсутствии соответствующего фолда, то есть i -ая модель обучается на всей обучающей выборке, кроме объектов, которые попали в i -ый фолд (out-of-fold). Затем мы измеряем качество i -ой модели на i -ом фолде. Так как он не участвовал в обучении этой модели, мы получим "честный результат". После этого, для получения финального значения метрики качества, мы можем усреднить полученные нами n значений.

```
In [269... cv_scores = cross_val_score(model, X_train[numeric_features], y_train, cv=10, scoring="neg_root_mean_squared_error")
print("Cross validation scores:\n\t", "\n\t".join("%.4f" % x for x in cv_scores))
print("Mean CV MSE = %.4f" % np.mean(-cv_scores))
```

Cross validation scores:

```
-50555.6407
-30683.0234
-44527.1351
-69910.1412
-44617.8750
-28718.6690
-26655.1929
-24070.2862
-27081.7721
-28525.4677
```

Mean CV MSE = 37534.5203

Обратите внимание на то, что результаты `cv_scores` получились отрицательными. Это соглашение в `sklearn` (скоринговую функцию нужно максимизировать). Поэтому все стандартные скореры называются `neg_*`, например, `neg_root_mean_squared_error`.

Обратите внимание, что по отложенной выборке и при кросс-валидации мы считаем RMSE (Root Mean Squared Error), хотя в функционале ошибки при обучении модели используется MSE.

$$\text{RMSE}(X, y, a) = \sqrt{\frac{1}{\ell} \sum_{i=1}^{\ell} (y_i - a(x_i))^2}$$

Вопрос: Почему оптимизация RMSE эквивалентна оптимизации MSE?

Ответ: т.к. MSE всегда положительна, то всегда можно найти RMSE. Так же \sqrt{x} монотонно возрастающая функция. Поэтому если $\sqrt{x_1} < \sqrt{x_2} \Rightarrow x_1 < x_2$

Для того, чтобы иметь некоторую точку отсчета, удобно посчитать оптимальное значение функции потерь при константном предсказании.

Вопрос: Чему равна оптимальная константа для RMSE?

Ответ: $\text{mean}(y)$.

```
In [270... best_constant = np.mean(y_train)
```

```
In [271... print(
    "Test RMSE with best constant = %.4f"
    % mean_squared_error(y_val, best_constant * np.ones(y_val.shape), squared=False)
)
print(
    "Train RMSE with best constant = %.4f"
    % mean_squared_error(y_train, best_constant * np.ones(y_train.shape), squared=False)
)
```

```
Test RMSE with best constant = 83757.5205
Train RMSE with best constant = 77919.4785
```

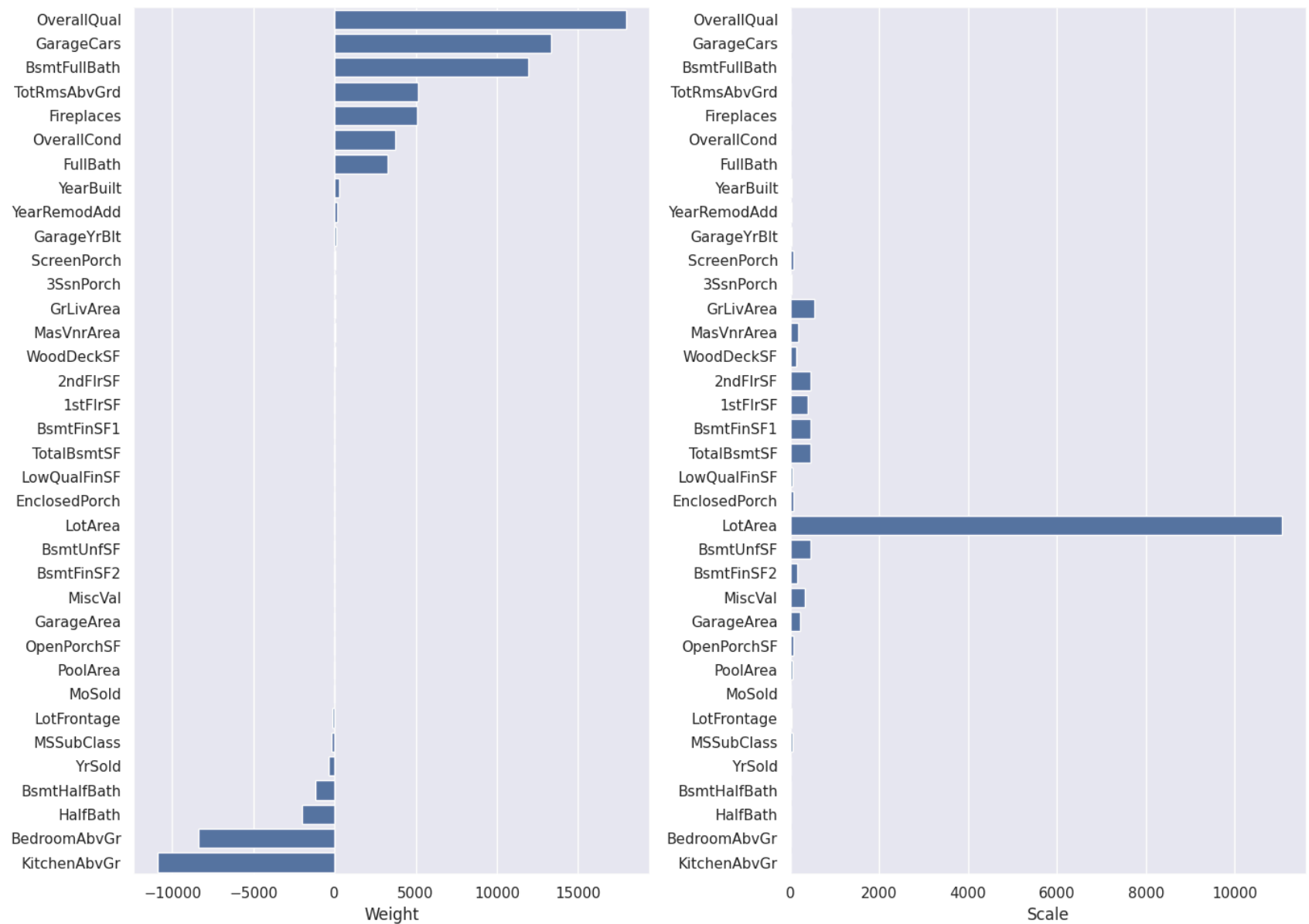
Давайте посмотрим на то, какие же признаки оказались самыми "сильными". Для этого визуализируем веса, соответствующие признакам. Чем больше вес — тем более сильным является признак.

Вопрос: Почему это не совсем правда?

Ответ: Из-за разных значений в фичах. Например если значения первой фичи измеряется в миллионах, а второй в десятых, то их веса не будут отображать силу признаков

```
In [272... def show_weights(features, weights, scales):  
    fig, axs = plt.subplots(figsize=(14, 10), ncols=2)  
    sorted_weights = sorted(zip(weights, features, scales), reverse=True)  
    weights = [x[0] for x in sorted_weights]  
    features = [x[1] for x in sorted_weights]  
    scales = [x[2] for x in sorted_weights]  
    sns.barplot(y=features, x=weights, ax=axs[0])  
    axs[0].set_xlabel("Weight")  
    sns.barplot(y=features, x=scales, ax=axs[1])  
    axs[1].set_xlabel("Scale")  
    plt.tight_layout()
```

```
In [273... show_weights(numeric_features, model.coef_, X_train[numeric_features].std())
```



Будем масштабировать наши признаки перед обучением модели. Это, среди, прочего, сделает нашу регуляризацию более честной: теперь все признаки будут регуляризоваться в равной степени.

Для этого воспользуемся трансформером `StandardScaler`. Трансформеры в `sklearn` имеют методы `fit` и `transform` (а еще `fit_transform`). Метод `fit` принимает на вход обучающую выборку и считает по ней необходимые значения (например статистики, как `StandardScaler`: среднее и стандартное отклонение каждого из признаков). `transform` применяет преобразование к переданной выборке.

```
In [274... X_train_scaled = StandardScaler().fit_transform(X_train[numeric_features])
X_val_scaled = StandardScaler().fit_transform(X_val[numeric_features])

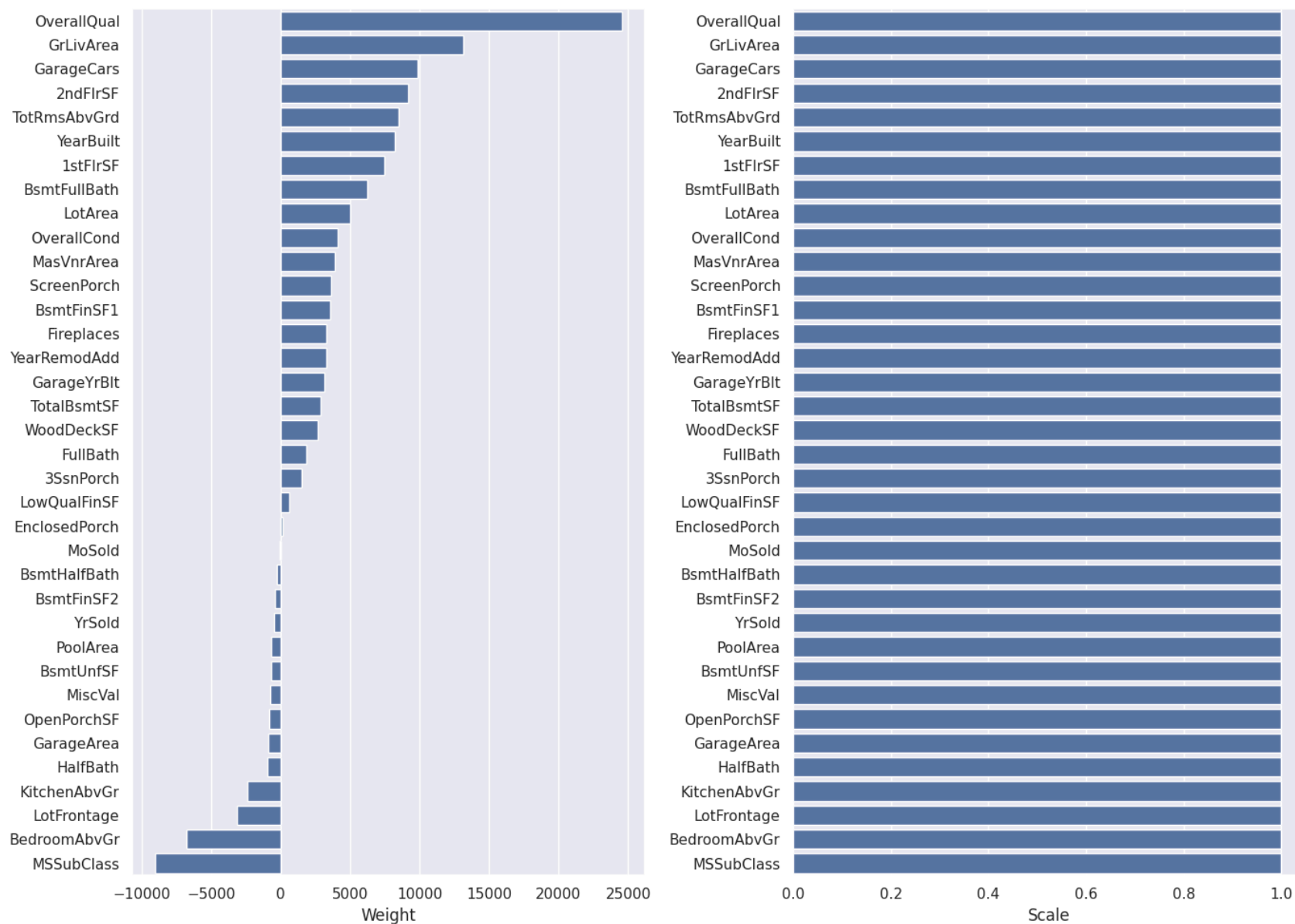
# For test
X_test_scaled = StandardScaler().fit_transform(X_test[numeric_features])
```

```
In [275... model = Ridge()
model.fit(X_train_scaled, y_train)
y_pred = model.predict(X_val_scaled)
y_train_pred = model.predict(X_train_scaled)

print("Test RMSE = %.4f" % mean_squared_error(y_val, y_pred, squared=False))
print("Train RMSE = %.4f" % mean_squared_error(y_train, y_train_pred, squared=False))
```

```
Test RMSE = 35052.3629
Train RMSE = 34459.0859
```

```
In [276... scales = pd.Series(data=X_train_scaled.std(axis=0), index=numeric_features)
show_weights(numeric_features, model.coef_, scales)
```



Наряду с параметрами (веса w , w_0), которые модель оптимизирует на этапе обучения, у модели есть и гиперпараметры. У нашей модели это **alpha** — коэффициент регуляризации. Подбирают его обычно по сетке, измеряя качество на валидационной (не тестовой) выборке или с помощью кросс-валидации. Посмотрим, как это можно сделать.

Для начала зададим возможные значения гиперпараметра, воспользуемся `np.logspace`, чтобы узнать оптимальный порядок величины. Ограничим допустимые значения 10^{-2} и 10^3 , возьмем 20 точек.

```
In [277... alphas = np.logspace(-2, 3, num=20)
```

```
assert alphas[0] == 1e-2
assert alphas[-1] == 1e3
assert len(alphas) == 20
```

```
alphas
```

```
Out[277... array([1.00000000e-02, 1.83298071e-02, 3.35981829e-02, 6.15848211e-02,
        1.12883789e-01, 2.06913808e-01, 3.79269019e-01, 6.95192796e-01,
        1.27427499e+00, 2.33572147e+00, 4.28133240e+00, 7.84759970e+00,
        1.43844989e+01, 2.63665090e+01, 4.83293024e+01, 8.85866790e+01,
        1.62377674e+02, 2.97635144e+02, 5.45559478e+02, 1.00000000e+03])
```

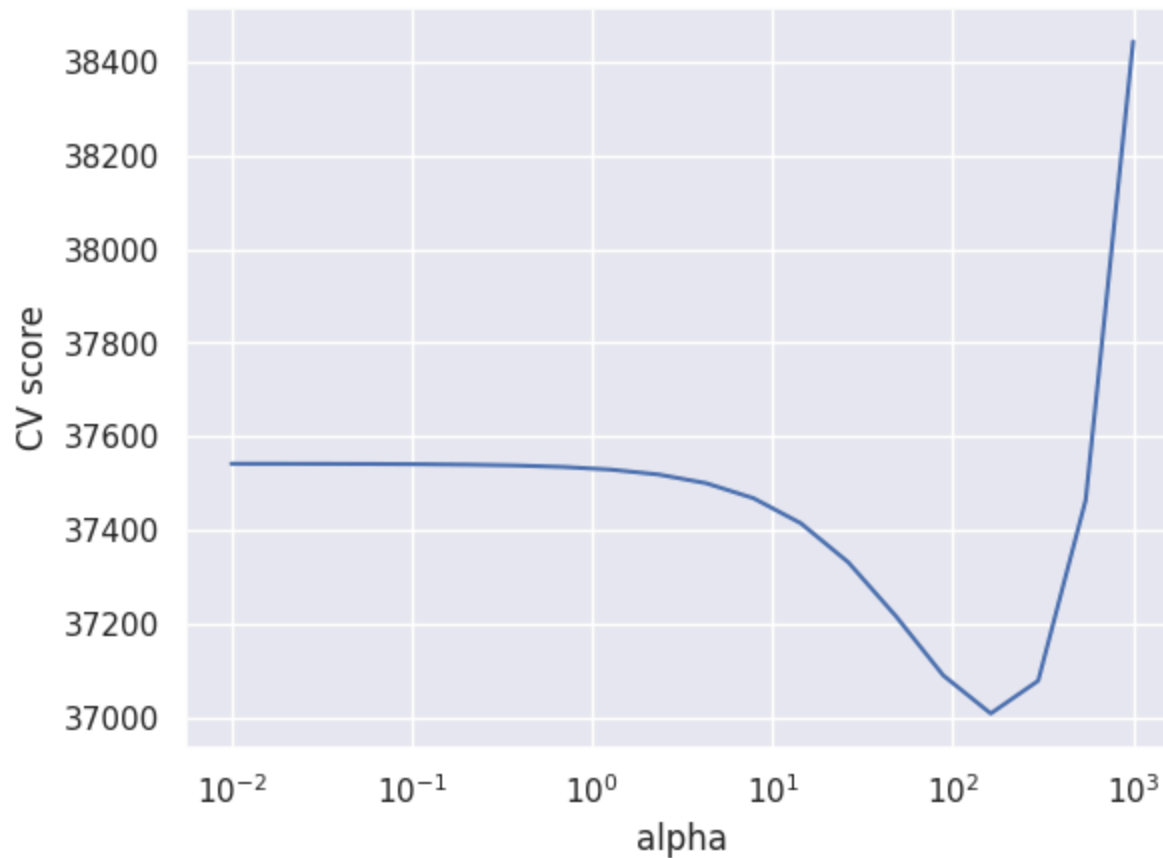
```
In [278... searcher = GridSearchCV(Ridge(), [{"alpha": alphas}], scoring="neg_root_mean_squared_error", cv=10)
searcher.fit(X_train_scaled, y_train)
```

```
best_alpha = searcher.best_params_["alpha"]
print("Best alpha = %.4f" % best_alpha)

plt.plot(alphas, -searcher.cv_results_["mean_test_score"])
plt.xscale("log")
plt.xlabel("alpha")
plt.ylabel("CV score")
```

```
Best alpha = 162.3777
```

```
Out[278... Text(0, 0.5, 'CV score')
```



Вопрос: Почему мы не подбираем коэффициент регуляризации по обучающей выборке? По тестовой выборке?

Ответ: Тестовая выборка нужна для подсчета итогового результата модели. Если на ней изучать гиперпараметры, то мы можем подстроиться под тестовую выборку и переобучиться. Подбор на обучающей выборке переобучит модель т.к. может подстроиться под все точки для минимальной ошибки. Такие гиперпараметры не будут корректно работать.

Попробуем обучить модель с подобранным коэффициентом регуляризации. Заодно воспользуемся очень удобным классом `Pipeline`: обучение модели часто представляется как последовательность некоторых действий с обучающей и тестовой выборками (например, сначала нужно отмасштабировать выборку (причем для обучающей выборки нужно применить метод `fit`, а для тестовой — `transform`), а затем обучить/применить модель (для обучающей `fit`, а для тестовой — `predict`). `Pipeline` позволяет хранить эту последовательность шагов и корректно обрабатывает разные типы выборок: и обучающую, и тестовую.

```
In [279... simple_pipeline = Pipeline([("scaling", StandardScaler()), ("regression", Ridge(best_alpha))])

model = simple_pipeline.fit(X_train[numeric_features], y_train)
y_pred = model.predict(X_val[numeric_features])
print("Test RMSE = %.4f" % mean_squared_error(y_val, y_pred, squared=False))
```

Test RMSE = 35293.4397

Часть 3. Работаем с категориальными признаками

Сейчас мы явно вытягиваем из данных не всю информацию, что у нас есть, просто потому, что мы не используем часть признаков. Эти признаки в датасете закодированы строками, каждый из них обозначает некоторую категорию. Такие признаки называются категориальными. Давайте выделим такие признаки и сразу заполним пропуски в них специальным значением (то, что у признака пропущено значение, само по себе может быть хорошим признаком).

```
In [280... categorical = list(X_train.dtypes[X_train.dtypes == "object"].index)
X_train[categorical].head()
```

```
Out[280...      MSZoning  Street  Alley  LotShape  LandContour  Utilities  LotConfig  LandSlope  Neighborhood  Condition1  ...  Garage
```

1023	RL	Pave	NaN	Reg	Lvl	AllPub	Inside	Gtl	Blmngtn	Norm	...	A
810	RL	Pave	NaN	Reg	Lvl	AllPub	Inside	Gtl	NWAmes	Norm	...	A
1384	RL	Pave	NaN	Reg	Lvl	AllPub	Inside	Gtl	Edwards	Norm	...	D
626	RL	Pave	NaN	IR1	Lvl	AllPub	Inside	Gtl	NAmes	Norm	...	A
813	RL	Pave	NaN	Reg	Lvl	AllPub	Inside	Gtl	NAmes	Norm	...	A

5 rows × 43 columns



В категориальных данных также есть пропуски, заполним их отдельной новой категорией **NotGiven**

```
In [281... X_train[categorical] = X_train[categorical].fillna("NotGiven")
X_val[categorical] = X_val[categorical].fillna("NotGiven")
```

```
In [282... X_train[categorical].sample(5)
```

```
Out[282...      MSZoning  Street  Alley  LotShape  LandContour  Utilities  LotConfig  LandSlope  Neighborhood  Condition1  ...  Gar
```

	MSZoning	Street	Alley	LotShape	LandContour	Utilities	LotConfig	LandSlope	Neighborhood	Condition1	...	Gar
214	RL	Pave	NotGiven	IR1	Lvl	AllPub	FR2	Gtl	CollgCr	Norm	...	
559	RL	Pave	NotGiven	Reg	Lvl	AllPub	Inside	Gtl	Blmngtn	Norm	...	
230	RL	Pave	NotGiven	Reg	Lvl	AllPub	Inside	Gtl	NAmes	Norm	...	
845	RL	Pave	NotGiven	IR1	Lvl	AllPub	CulDSac	Gtl	Sawyer	RRAe	...	
702	RL	Pave	NotGiven	IR1	Lvl	AllPub	Inside	Gtl	StoneBr	Norm	...	

5 rows × 43 columns



Сейчас нам нужно как-то закодировать эти категориальные признаки числами, ведь линейная модель не может работать с такими абстракциями. Два стандартных трансформера из `sklearn` для работы с категориальными признаками — `OrdinalEncoder` (просто перенумеровывает значения признака натуральными числами) и `OneHotEncoder`.

`OneHotEncoder` ставит в соответствие каждому признаку целый вектор, состоящий из нулей и одной единицы (которая стоит на месте, соответствующем принимаемому значению, таким образом кодируя его).

Вопрос: Проинтерпретируйте, что означают веса модели перед OneHot-кодированными признаками. Почему пользоваться `OrdinalEncoder` в случае линейной модели — скорее плохой вариант? Какие недостатки есть у OneHot-кодирования?

Ответ: Веса модели перед OneHot-кодированными признаками обозначают то, на сколько увеличится целевая переменная, если этот признак верен. `OrdinalEncoder` просто кодирует признаки и эти значения не имеют смысла. Т.к. это значение будет умножаться на вес, то одно название будет давать больший прирост чем другое, хотя это ничем не обусловлено.

```
In [283... column_transformer = ColumnTransformer(
    [ ("ohe", OneHotEncoder(handle_unknown="ignore"), categorical), ("scaling", StandardScaler(), numeric_fea
    )

pipeline = Pipeline(steps=[ ("ohe_and_scaling", column_transformer), ("regression", Ridge()) ])
```

```
model = pipeline.fit(X_train, y_train)
y_pred = model.predict(X_val)
print("Test RMSE = %.4f" % mean_squared_error(y_val, y_pred, squared=False))
```

Test RMSE = 28230.0853

Вопрос: Как вы думаете, почему мы не производим скейлинг OneHot-кодированных признаков?

Ответ: Эти признаки имеют только значения 1 или 0. При их скейлинге они могут выйти за это ограничение и потеряют свой смысл.

Посмотрим на размеры матрицы после OneHot-кодирования:

```
In [284... print("Size before OneHot:", X_train.shape)
print("Size after OneHot:", column_transformer.transform(X_train).shape)
```

Size before OneHot: (1095, 79)

Size after OneHot: (1095, 299)

Как видим, количество признаков увеличилось более, чем в 3 раза. Это может повысить риски переобучиться: соотношение количества объектов к количеству признаков сильно сократилось.

Попытаемся обучить линейную регрессию с ℓ_1 -регуляризатором.

Вопрос: Каким полезным свойством обладает такой регуляризатор?

Ответ: Уменьшает риски переобучения т.к. штрафует большие веса.

```
In [285... column_transformer = ColumnTransformer(
    [("one", OneHotEncoder(handle_unknown="ignore"), categorical), ("scaling", StandardScaler(), numeric_features)]
)

lasso_pipeline = Pipeline(steps=[("one_and_scaling", column_transformer), ("regression", Lasso())])

model = lasso_pipeline.fit(X_train, y_train)
y_pred = model.predict(X_val)
print("RMSE = %.4f" % mean_squared_error(y_val, y_pred, squared=False))
```

RMSE = 27609.9863

```
In [286... ridge_zeros = np.sum(pipeline.steps[-1][-1].coef_ == 0)
lasso_zeros = np.sum(lasso_pipeline.steps[-1][-1].coef_ == 0)
print("Zero weights in Ridge:", ridge_zeros)
print("Zero weights in Lasso:", lasso_zeros)
```

Zero weights in Ridge: 0

Zero weights in Lasso: 37

Подберем для нашей модели оптимальный коэффициент регуляризации. Обратите внимание, как перебираются параметры у `Pipeline`.

```
In [287... alphas = np.logspace(-2, 4, 20)
searcher = GridSearchCV(
    lasso_pipeline, [{"regression__alpha": alphas}], scoring="neg_root_mean_squared_error", cv=10, n_jobs=
)
searcher.fit(X_train, y_train)

best_alpha = searcher.best_params_["regression__alpha"]
print("Best alpha = %.4f" % best_alpha)

plt.plot(alphas, -searcher.cv_results_["mean_test_score"])
plt.xscale("log")
plt.xlabel("alpha")
plt.ylabel("CV score")
```

```
/home/wrdx/Desktop/spbu_ml_sem4/.venv/lib/python3.12/site-packages/sklearn/linear_model/_coordinate_descen
t.py:656: ConvergenceWarning: Objective did not converge. You might want to increase the number of iteratio
ns. Duality gap: 101832280886.43358, tolerance: 568954314.8675103
    model = cd_fast.sparse_enet_coordinate_descent(
/home/wrdx/Desktop/spbu_ml_sem4/.venv/lib/python3.12/site-packages/sklearn/linear_model/_coordinate_descen
t.py:656: ConvergenceWarning: Objective did not converge. You might want to increase the number of iteratio
ns. Duality gap: 190692315172.5784, tolerance: 593335415.6996667
    model = cd_fast.sparse_enet_coordinate_descent(
/home/wrdx/Desktop/spbu_ml_sem4/.venv/lib/python3.12/site-packages/sklearn/linear_model/_coordinate_descen
t.py:656: ConvergenceWarning: Objective did not converge. You might want to increase the number of iteratio
ns. Duality gap: 180859161938.03278, tolerance: 610957735.0128442
    model = cd_fast.sparse_enet_coordinate_descent(
/home/wrdx/Desktop/spbu_ml_sem4/.venv/lib/python3.12/site-packages/sklearn/linear_model/_coordinate_descen
t.py:656: ConvergenceWarning: Objective did not converge. You might want to increase the number of iteratio
ns. Duality gap: 165765887086.21555, tolerance: 568471343.1704118
    model = cd_fast.sparse_enet_coordinate_descent(
/home/wrdx/Desktop/spbu_ml_sem4/.venv/lib/python3.12/site-packages/sklearn/linear_model/_coordinate_descen
t.py:656: ConvergenceWarning: Objective did not converge. You might want to increase the number of iteratio
ns. Duality gap: 159579578619.53107, tolerance: 611603075.72175
    model = cd_fast.sparse_enet_coordinate_descent(
/home/wrdx/Desktop/spbu_ml_sem4/.venv/lib/python3.12/site-packages/sklearn/linear_model/_coordinate_descen
t.py:656: ConvergenceWarning: Objective did not converge. You might want to increase the number of iteratio
ns. Duality gap: 153204748668.07202, tolerance: 599961812.0571665
    model = cd_fast.sparse_enet_coordinate_descent(
/home/wrdx/Desktop/spbu_ml_sem4/.venv/lib/python3.12/site-packages/sklearn/linear_model/_coordinate_descen
t.py:656: ConvergenceWarning: Objective did not converge. You might want to increase the number of iteratio
ns. Duality gap: 187958485944.6777, tolerance: 607603680.0773777
    model = cd_fast.sparse_enet_coordinate_descent(
/home/wrdx/Desktop/spbu_ml_sem4/.venv/lib/python3.12/site-packages/sklearn/linear_model/_coordinate_descen
t.py:656: ConvergenceWarning: Objective did not converge. You might want to increase the number of iteratio
ns. Duality gap: 189086887105.1485, tolerance: 603974920.6996291
    model = cd_fast.sparse_enet_coordinate_descent(
/home/wrdx/Desktop/spbu_ml_sem4/.venv/lib/python3.12/site-packages/sklearn/linear_model/_coordinate_descen
t.py:656: ConvergenceWarning: Objective did not converge. You might want to increase the number of iteratio
ns. Duality gap: 192151650183.0622, tolerance: 598262324.2191347
    model = cd_fast.sparse_enet_coordinate_descent(
/home/wrdx/Desktop/spbu_ml_sem4/.venv/lib/python3.12/site-packages/sklearn/linear_model/_coordinate_descen
t.py:656: ConvergenceWarning: Objective did not converge. You might want to increase the number of iteratio
ns. Duality gap: 190421059732.08145, tolerance: 593335415.6996667
    model = cd_fast.sparse_enet_coordinate_descent(
/home/wrdx/Desktop/spbu_ml_sem4/.venv/lib/python3.12/site-packages/sklearn/linear_model/_coordinate_descen
t.py:656: ConvergenceWarning: Objective did not converge. You might want to increase the number of iteratio
```

```
ns. Duality gap: 126757597405.56113, tolerance: 619865924.1635104
  model = cd_fast.sparse_enet_coordinate_descent(
/home/wrdx/Desktop/spbu_ml_sem4/.venv/lib/python3.12/site-packages/sklearn/linear_model/_coordinate_descen
t.py:656: ConvergenceWarning: Objective did not converge. You might want to increase the number of iteratio
ns. Duality gap: 76957844413.73854, tolerance: 568954314.8675103
  model = cd_fast.sparse_enet_coordinate_descent(
/home/wrdx/Desktop/spbu_ml_sem4/.venv/lib/python3.12/site-packages/sklearn/linear_model/_coordinate_descen
t.py:656: ConvergenceWarning: Objective did not converge. You might want to increase the number of iteratio
ns. Duality gap: 159349582158.69733, tolerance: 611603075.72175
  model = cd_fast.sparse_enet_coordinate_descent(
/home/wrdx/Desktop/spbu_ml_sem4/.venv/lib/python3.12/site-packages/sklearn/linear_model/_coordinate_descen
t.py:656: ConvergenceWarning: Objective did not converge. You might want to increase the number of iteratio
ns. Duality gap: 180450777130.2276, tolerance: 610957735.0128442
  model = cd_fast.sparse_enet_coordinate_descent(
/home/wrdx/Desktop/spbu_ml_sem4/.venv/lib/python3.12/site-packages/sklearn/linear_model/_coordinate_descen
t.py:656: ConvergenceWarning: Objective did not converge. You might want to increase the number of iteratio
ns. Duality gap: 165560327423.74527, tolerance: 568471343.1704118
  model = cd_fast.sparse_enet_coordinate_descent(
/home/wrdx/Desktop/spbu_ml_sem4/.venv/lib/python3.12/site-packages/sklearn/linear_model/_coordinate_descen
t.py:656: ConvergenceWarning: Objective did not converge. You might want to increase the number of iteratio
ns. Duality gap: 152993094099.3401, tolerance: 599961812.0571665
  model = cd_fast.sparse_enet_coordinate_descent(
/home/wrdx/Desktop/spbu_ml_sem4/.venv/lib/python3.12/site-packages/sklearn/linear_model/_coordinate_descen
t.py:656: ConvergenceWarning: Objective did not converge. You might want to increase the number of iteratio
ns. Duality gap: 187640380579.085, tolerance: 607603680.0773777
  model = cd_fast.sparse_enet_coordinate_descent(
/home/wrdx/Desktop/spbu_ml_sem4/.venv/lib/python3.12/site-packages/sklearn/linear_model/_coordinate_descen
t.py:656: ConvergenceWarning: Objective did not converge. You might want to increase the number of iteratio
ns. Duality gap: 188805653291.7378, tolerance: 603974920.6996291
  model = cd_fast.sparse_enet_coordinate_descent(
/home/wrdx/Desktop/spbu_ml_sem4/.venv/lib/python3.12/site-packages/sklearn/linear_model/_coordinate_descen
t.py:656: ConvergenceWarning: Objective did not converge. You might want to increase the number of iteratio
ns. Duality gap: 191850856400.98032, tolerance: 598262324.2191347
  model = cd_fast.sparse_enet_coordinate_descent(
/home/wrdx/Desktop/spbu_ml_sem4/.venv/lib/python3.12/site-packages/sklearn/linear_model/_coordinate_descen
t.py:656: ConvergenceWarning: Objective did not converge. You might want to increase the number of iteratio
ns. Duality gap: 126796237640.15909, tolerance: 619865924.1635104
  model = cd_fast.sparse_enet_coordinate_descent(
/home/wrdx/Desktop/spbu_ml_sem4/.venv/lib/python3.12/site-packages/sklearn/linear_model/_coordinate_descen
t.py:656: ConvergenceWarning: Objective did not converge. You might want to increase the number of iteratio
ns. Duality gap: 189852937603.2021, tolerance: 593335415.6996667
  model = cd_fast.sparse_enet_coordinate_descent(
```



```
/home/wrdx/Desktop/spbu_ml_sem4/.venv/lib/python3.12/site-packages/sklearn/linear_model/_coordinate_descen
t.py:656: ConvergenceWarning: Objective did not converge. You might want to increase the number of iteratio
ns. Duality gap: 179592957990.5571, tolerance: 610957735.0128442
    model = cd_fast.sparse_enet_coordinate_descent(
/home/wrdx/Desktop/spbu_ml_sem4/.venv/lib/python3.12/site-packages/sklearn/linear_model/_coordinate_descen
t.py:656: ConvergenceWarning: Objective did not converge. You might want to increase the number of iteratio
ns. Duality gap: 81480810814.86258, tolerance: 568954314.8675103
    model = cd_fast.sparse_enet_coordinate_descent(
/home/wrdx/Desktop/spbu_ml_sem4/.venv/lib/python3.12/site-packages/sklearn/linear_model/_coordinate_descen
t.py:656: ConvergenceWarning: Objective did not converge. You might want to increase the number of iteratio
ns. Duality gap: 158866781323.49777, tolerance: 611603075.72175
    model = cd_fast.sparse_enet_coordinate_descent(
/home/wrdx/Desktop/spbu_ml_sem4/.venv/lib/python3.12/site-packages/sklearn/linear_model/_coordinate_descen
t.py:656: ConvergenceWarning: Objective did not converge. You might want to increase the number of iteratio
ns. Duality gap: 165132717784.28305, tolerance: 568471343.1704118
    model = cd_fast.sparse_enet_coordinate_descent(
/home/wrdx/Desktop/spbu_ml_sem4/.venv/lib/python3.12/site-packages/sklearn/linear_model/_coordinate_descen
t.py:656: ConvergenceWarning: Objective did not converge. You might want to increase the number of iteratio
ns. Duality gap: 152549516885.6468, tolerance: 599961812.0571665
    model = cd_fast.sparse_enet_coordinate_descent(
/home/wrdx/Desktop/spbu_ml_sem4/.venv/lib/python3.12/site-packages/sklearn/linear_model/_coordinate_descen
t.py:656: ConvergenceWarning: Objective did not converge. You might want to increase the number of iteratio
ns. Duality gap: 186974295635.7799, tolerance: 607603680.0773777
    model = cd_fast.sparse_enet_coordinate_descent(
/home/wrdx/Desktop/spbu_ml_sem4/.venv/lib/python3.12/site-packages/sklearn/linear_model/_coordinate_descen
t.py:656: ConvergenceWarning: Objective did not converge. You might want to increase the number of iteratio
ns. Duality gap: 188216980866.31778, tolerance: 603974920.6996291
    model = cd_fast.sparse_enet_coordinate_descent(
/home/wrdx/Desktop/spbu_ml_sem4/.venv/lib/python3.12/site-packages/sklearn/linear_model/_coordinate_descen
t.py:656: ConvergenceWarning: Objective did not converge. You might want to increase the number of iteratio
ns. Duality gap: 191220674005.914, tolerance: 598262324.2191347
    model = cd_fast.sparse_enet_coordinate_descent(
/home/wrdx/Desktop/spbu_ml_sem4/.venv/lib/python3.12/site-packages/sklearn/linear_model/_coordinate_descen
t.py:656: ConvergenceWarning: Objective did not converge. You might want to increase the number of iteratio
ns. Duality gap: 126874632693.89508, tolerance: 619865924.1635104
    model = cd_fast.sparse_enet_coordinate_descent(
/home/wrdx/Desktop/spbu_ml_sem4/.venv/lib/python3.12/site-packages/sklearn/linear_model/_coordinate_descen
t.py:656: ConvergenceWarning: Objective did not converge. You might want to increase the number of iteratio
ns. Duality gap: 81816450365.79642, tolerance: 568954314.8675103
    model = cd_fast.sparse_enet_coordinate_descent(
/home/wrdx/Desktop/spbu_ml_sem4/.venv/lib/python3.12/site-packages/sklearn/linear_model/_coordinate_descen
t.py:656: ConvergenceWarning: Objective did not converge. You might want to increase the number of iteratio
```

```
ns. Duality gap: 188649813186.25623, tolerance: 593335415.6996667
  model = cd_fast.sparse_enet_coordinate_descent(
/home/wrdx/Desktop/spbu_ml_sem4/.venv/lib/python3.12/site-packages/sklearn/linear_model/_coordinate_descen
t.py:656: ConvergenceWarning: Objective did not converge. You might want to increase the number of iteratio
ns. Duality gap: 151607689196.286, tolerance: 599961812.0571665
  model = cd_fast.sparse_enet_coordinate_descent(
/home/wrdx/Desktop/spbu_ml_sem4/.venv/lib/python3.12/site-packages/sklearn/linear_model/_coordinate_descen
t.py:656: ConvergenceWarning: Objective did not converge. You might want to increase the number of iteratio
ns. Duality gap: 164237619303.83646, tolerance: 568471343.1704118
  model = cd_fast.sparse_enet_coordinate_descent(
/home/wrdx/Desktop/spbu_ml_sem4/.venv/lib/python3.12/site-packages/sklearn/linear_model/_coordinate_descen
t.py:656: ConvergenceWarning: Objective did not converge. You might want to increase the number of iteratio
ns. Duality gap: 177759855212.79654, tolerance: 610957735.0128442
  model = cd_fast.sparse_enet_coordinate_descent(
/home/wrdx/Desktop/spbu_ml_sem4/.venv/lib/python3.12/site-packages/sklearn/linear_model/_coordinate_descen
t.py:656: ConvergenceWarning: Objective did not converge. You might want to increase the number of iteratio
ns. Duality gap: 157848657336.24347, tolerance: 611603075.72175
  model = cd_fast.sparse_enet_coordinate_descent(
/home/wrdx/Desktop/spbu_ml_sem4/.venv/lib/python3.12/site-packages/sklearn/linear_model/_coordinate_descen
t.py:656: ConvergenceWarning: Objective did not converge. You might want to increase the number of iteratio
ns. Duality gap: 185561937433.4928, tolerance: 607603680.0773777
  model = cd_fast.sparse_enet_coordinate_descent(
/home/wrdx/Desktop/spbu_ml_sem4/.venv/lib/python3.12/site-packages/sklearn/linear_model/_coordinate_descen
t.py:656: ConvergenceWarning: Objective did not converge. You might want to increase the number of iteratio
ns. Duality gap: 186968350947.97913, tolerance: 603974920.6996291
  model = cd_fast.sparse_enet_coordinate_descent(
/home/wrdx/Desktop/spbu_ml_sem4/.venv/lib/python3.12/site-packages/sklearn/linear_model/_coordinate_descen
t.py:656: ConvergenceWarning: Objective did not converge. You might want to increase the number of iteratio
ns. Duality gap: 189885649156.28702, tolerance: 598262324.2191347
  model = cd_fast.sparse_enet_coordinate_descent(
/home/wrdx/Desktop/spbu_ml_sem4/.venv/lib/python3.12/site-packages/sklearn/linear_model/_coordinate_descen
t.py:656: ConvergenceWarning: Objective did not converge. You might want to increase the number of iteratio
ns. Duality gap: 127498677383.4271, tolerance: 619865924.1635104
  model = cd_fast.sparse_enet_coordinate_descent(
/home/wrdx/Desktop/spbu_ml_sem4/.venv/lib/python3.12/site-packages/sklearn/linear_model/_coordinate_descen
t.py:656: ConvergenceWarning: Objective did not converge. You might want to increase the number of iteratio
ns. Duality gap: 186042500998.61377, tolerance: 593335415.6996667
  model = cd_fast.sparse_enet_coordinate_descent(
/home/wrdx/Desktop/spbu_ml_sem4/.venv/lib/python3.12/site-packages/sklearn/linear_model/_coordinate_descen
t.py:656: ConvergenceWarning: Objective did not converge. You might want to increase the number of iteratio
ns. Duality gap: 115270742794.79807, tolerance: 568954314.8675103
  model = cd_fast.sparse_enet_coordinate_descent(
```

```
/home/wrdx/Desktop/spbu_ml_sem4/.venv/lib/python3.12/site-packages/sklearn/linear_model/_coordinate_descen
t.py:656: ConvergenceWarning: Objective did not converge. You might want to increase the number of iteratio
ns. Duality gap: 155648208132.84006, tolerance: 611603075.72175
    model = cd_fast.sparse_enet_coordinate_descent(
/home/wrdx/Desktop/spbu_ml_sem4/.venv/lib/python3.12/site-packages/sklearn/linear_model/_coordinate_descen
t.py:656: ConvergenceWarning: Objective did not converge. You might want to increase the number of iteratio
ns. Duality gap: 162341650687.47226, tolerance: 568471343.1704118
    model = cd_fast.sparse_enet_coordinate_descent(
/home/wrdx/Desktop/spbu_ml_sem4/.venv/lib/python3.12/site-packages/sklearn/linear_model/_coordinate_descen
t.py:656: ConvergenceWarning: Objective did not converge. You might want to increase the number of iteratio
ns. Duality gap: 182500743395.73215, tolerance: 607603680.0773777
    model = cd_fast.sparse_enet_coordinate_descent(
/home/wrdx/Desktop/spbu_ml_sem4/.venv/lib/python3.12/site-packages/sklearn/linear_model/_coordinate_descen
t.py:656: ConvergenceWarning: Objective did not converge. You might want to increase the number of iteratio
ns. Duality gap: 184266395498.9654, tolerance: 603974920.6996291
    model = cd_fast.sparse_enet_coordinate_descent(
/home/wrdx/Desktop/spbu_ml_sem4/.venv/lib/python3.12/site-packages/sklearn/linear_model/_coordinate_descen
t.py:656: ConvergenceWarning: Objective did not converge. You might want to increase the number of iteratio
ns. Duality gap: 173712655482.74377, tolerance: 610957735.0128442
    model = cd_fast.sparse_enet_coordinate_descent(
/home/wrdx/Desktop/spbu_ml_sem4/.venv/lib/python3.12/site-packages/sklearn/linear_model/_coordinate_descen
t.py:656: ConvergenceWarning: Objective did not converge. You might want to increase the number of iteratio
ns. Duality gap: 149568158039.33777, tolerance: 599961812.0571665
    model = cd_fast.sparse_enet_coordinate_descent(
/home/wrdx/Desktop/spbu_ml_sem4/.venv/lib/python3.12/site-packages/sklearn/linear_model/_coordinate_descen
t.py:656: ConvergenceWarning: Objective did not converge. You might want to increase the number of iteratio
ns. Duality gap: 127308205217.95435, tolerance: 619865924.1635104
    model = cd_fast.sparse_enet_coordinate_descent(
/home/wrdx/Desktop/spbu_ml_sem4/.venv/lib/python3.12/site-packages/sklearn/linear_model/_coordinate_descen
t.py:656: ConvergenceWarning: Objective did not converge. You might want to increase the number of iteratio
ns. Duality gap: 42287210773.59897, tolerance: 568954314.8675103
    model = cd_fast.sparse_enet_coordinate_descent(
/home/wrdx/Desktop/spbu_ml_sem4/.venv/lib/python3.12/site-packages/sklearn/linear_model/_coordinate_descen
t.py:656: ConvergenceWarning: Objective did not converge. You might want to increase the number of iteratio
ns. Duality gap: 180158160473.6864, tolerance: 593335415.6996667
    model = cd_fast.sparse_enet_coordinate_descent(
/home/wrdx/Desktop/spbu_ml_sem4/.venv/lib/python3.12/site-packages/sklearn/linear_model/_coordinate_descen
t.py:656: ConvergenceWarning: Objective did not converge. You might want to increase the number of iteratio
ns. Duality gap: 150612788458.82687, tolerance: 611603075.72175
    model = cd_fast.sparse_enet_coordinate_descent(
/home/wrdx/Desktop/spbu_ml_sem4/.venv/lib/python3.12/site-packages/sklearn/linear_model/_coordinate_descen
t.py:656: ConvergenceWarning: Objective did not converge. You might want to increase the number of iteratio
```

```
ns. Duality gap: 186989142506.43994, tolerance: 598262324.2191347
  model = cd_fast.sparse_enet_coordinate_descent(
/home/wrdx/Desktop/spbu_ml_sem4/.venv/lib/python3.12/site-packages/sklearn/linear_model/_coordinate_descen
t.py:656: ConvergenceWarning: Objective did not converge. You might want to increase the number of iteratio
ns. Duality gap: 144904355275.6842, tolerance: 599961812.0571665
  model = cd_fast.sparse_enet_coordinate_descent(
/home/wrdx/Desktop/spbu_ml_sem4/.venv/lib/python3.12/site-packages/sklearn/linear_model/_coordinate_descen
t.py:656: ConvergenceWarning: Objective did not converge. You might want to increase the number of iteratio
ns. Duality gap: 164183155578.2536, tolerance: 610957735.0128442
  model = cd_fast.sparse_enet_coordinate_descent(
/home/wrdx/Desktop/spbu_ml_sem4/.venv/lib/python3.12/site-packages/sklearn/linear_model/_coordinate_descen
t.py:656: ConvergenceWarning: Objective did not converge. You might want to increase the number of iteratio
ns. Duality gap: 158282287586.4377, tolerance: 568471343.1704118
  model = cd_fast.sparse_enet_coordinate_descent(
/home/wrdx/Desktop/spbu_ml_sem4/.venv/lib/python3.12/site-packages/sklearn/linear_model/_coordinate_descen
t.py:656: ConvergenceWarning: Objective did not converge. You might want to increase the number of iteratio
ns. Duality gap: 175492339594.7438, tolerance: 607603680.0773777
  model = cd_fast.sparse_enet_coordinate_descent(
/home/wrdx/Desktop/spbu_ml_sem4/.venv/lib/python3.12/site-packages/sklearn/linear_model/_coordinate_descen
t.py:656: ConvergenceWarning: Objective did not converge. You might want to increase the number of iteratio
ns. Duality gap: 180416764888.89407, tolerance: 598262324.2191347
  model = cd_fast.sparse_enet_coordinate_descent(
/home/wrdx/Desktop/spbu_ml_sem4/.venv/lib/python3.12/site-packages/sklearn/linear_model/_coordinate_descen
t.py:656: ConvergenceWarning: Objective did not converge. You might want to increase the number of iteratio
ns. Duality gap: 178157710417.93326, tolerance: 603974920.6996291
  model = cd_fast.sparse_enet_coordinate_descent(
/home/wrdx/Desktop/spbu_ml_sem4/.venv/lib/python3.12/site-packages/sklearn/linear_model/_coordinate_descen
t.py:656: ConvergenceWarning: Objective did not converge. You might want to increase the number of iteratio
ns. Duality gap: 127908217976.92102, tolerance: 619865924.1635104
  model = cd_fast.sparse_enet_coordinate_descent(
/home/wrdx/Desktop/spbu_ml_sem4/.venv/lib/python3.12/site-packages/sklearn/linear_model/_coordinate_descen
t.py:656: ConvergenceWarning: Objective did not converge. You might want to increase the number of iteratio
ns. Duality gap: 39062948904.11499, tolerance: 568954314.8675103
  model = cd_fast.sparse_enet_coordinate_descent(
/home/wrdx/Desktop/spbu_ml_sem4/.venv/lib/python3.12/site-packages/sklearn/linear_model/_coordinate_descen
t.py:656: ConvergenceWarning: Objective did not converge. You might want to increase the number of iteratio
ns. Duality gap: 138014422576.77246, tolerance: 611603075.72175
  model = cd_fast.sparse_enet_coordinate_descent(
/home/wrdx/Desktop/spbu_ml_sem4/.venv/lib/python3.12/site-packages/sklearn/linear_model/_coordinate_descen
t.py:656: ConvergenceWarning: Objective did not converge. You might want to increase the number of iteratio
ns. Duality gap: 165565545498.87903, tolerance: 593335415.6996667
  model = cd_fast.sparse_enet_coordinate_descent(
```

```
/home/wrdx/Desktop/spbu_ml_sem4/.venv/lib/python3.12/site-packages/sklearn/linear_model/_coordinate_descen
t.py:656: ConvergenceWarning: Objective did not converge. You might want to increase the number of iteratio
ns. Duality gap: 138005809391.14093, tolerance: 610957735.0128442
    model = cd_fast.sparse_enet_coordinate_descent(
/home/wrdx/Desktop/spbu_ml_sem4/.venv/lib/python3.12/site-packages/sklearn/linear_model/_coordinate_descen
t.py:656: ConvergenceWarning: Objective did not converge. You might want to increase the number of iteratio
ns. Duality gap: 133243605184.79762, tolerance: 599961812.0571665
    model = cd_fast.sparse_enet_coordinate_descent(
/home/wrdx/Desktop/spbu_ml_sem4/.venv/lib/python3.12/site-packages/sklearn/linear_model/_coordinate_descen
t.py:656: ConvergenceWarning: Objective did not converge. You might want to increase the number of iteratio
ns. Duality gap: 149312398247.4555, tolerance: 568471343.1704118
    model = cd_fast.sparse_enet_coordinate_descent(
/home/wrdx/Desktop/spbu_ml_sem4/.venv/lib/python3.12/site-packages/sklearn/linear_model/_coordinate_descen
t.py:656: ConvergenceWarning: Objective did not converge. You might want to increase the number of iteratio
ns. Duality gap: 162782640393.42612, tolerance: 603974920.6996291
    model = cd_fast.sparse_enet_coordinate_descent(
/home/wrdx/Desktop/spbu_ml_sem4/.venv/lib/python3.12/site-packages/sklearn/linear_model/_coordinate_descen
t.py:656: ConvergenceWarning: Objective did not converge. You might want to increase the number of iteratio
ns. Duality gap: 163950786710.36682, tolerance: 598262324.2191347
    model = cd_fast.sparse_enet_coordinate_descent(
/home/wrdx/Desktop/spbu_ml_sem4/.venv/lib/python3.12/site-packages/sklearn/linear_model/_coordinate_descen
t.py:656: ConvergenceWarning: Objective did not converge. You might want to increase the number of iteratio
ns. Duality gap: 28194442654.032623, tolerance: 619865924.1635104
    model = cd_fast.sparse_enet_coordinate_descent(
/home/wrdx/Desktop/spbu_ml_sem4/.venv/lib/python3.12/site-packages/sklearn/linear_model/_coordinate_descen
t.py:656: ConvergenceWarning: Objective did not converge. You might want to increase the number of iteratio
ns. Duality gap: 157654477870.1571, tolerance: 607603680.0773777
    model = cd_fast.sparse_enet_coordinate_descent(
/home/wrdx/Desktop/spbu_ml_sem4/.venv/lib/python3.12/site-packages/sklearn/linear_model/_coordinate_descen
t.py:656: ConvergenceWarning: Objective did not converge. You might want to increase the number of iteratio
ns. Duality gap: 121369899699.83585, tolerance: 593335415.6996667
    model = cd_fast.sparse_enet_coordinate_descent(
/home/wrdx/Desktop/spbu_ml_sem4/.venv/lib/python3.12/site-packages/sklearn/linear_model/_coordinate_descen
t.py:656: ConvergenceWarning: Objective did not converge. You might want to increase the number of iteratio
ns. Duality gap: 95747683001.10901, tolerance: 611603075.72175
    model = cd_fast.sparse_enet_coordinate_descent(
/home/wrdx/Desktop/spbu_ml_sem4/.venv/lib/python3.12/site-packages/sklearn/linear_model/_coordinate_descen
t.py:656: ConvergenceWarning: Objective did not converge. You might want to increase the number of iteratio
ns. Duality gap: 28618925739.16919, tolerance: 568954314.8675103
    model = cd_fast.sparse_enet_coordinate_descent(
/home/wrdx/Desktop/spbu_ml_sem4/.venv/lib/python3.12/site-packages/sklearn/linear_model/_coordinate_descen
t.py:656: ConvergenceWarning: Objective did not converge. You might want to increase the number of iteratio
```



```
ns. Duality gap: 97084079263.48663, tolerance: 599961812.0571665
  model = cd_fast.sparse_enet_coordinate_descent(
/home/wrdx/Desktop/spbu_ml_sem4/.venv/lib/python3.12/site-packages/sklearn/linear_model/_coordinate_descen
t.py:656: ConvergenceWarning: Objective did not converge. You might want to increase the number of iteratio
ns. Duality gap: 128177856960.58018, tolerance: 568471343.1704118
  model = cd_fast.sparse_enet_coordinate_descent(
/home/wrdx/Desktop/spbu_ml_sem4/.venv/lib/python3.12/site-packages/sklearn/linear_model/_coordinate_descen
t.py:656: ConvergenceWarning: Objective did not converge. You might want to increase the number of iteratio
ns. Duality gap: 113790329754.42035, tolerance: 603974920.6996291
  model = cd_fast.sparse_enet_coordinate_descent(
/home/wrdx/Desktop/spbu_ml_sem4/.venv/lib/python3.12/site-packages/sklearn/linear_model/_coordinate_descen
t.py:656: ConvergenceWarning: Objective did not converge. You might want to increase the number of iteratio
ns. Duality gap: 93048772344.42937, tolerance: 607603680.0773777
  model = cd_fast.sparse_enet_coordinate_descent(
/home/wrdx/Desktop/spbu_ml_sem4/.venv/lib/python3.12/site-packages/sklearn/linear_model/_coordinate_descen
t.py:656: ConvergenceWarning: Objective did not converge. You might want to increase the number of iteratio
ns. Duality gap: 108863429870.55188, tolerance: 598262324.2191347
  model = cd_fast.sparse_enet_coordinate_descent(
/home/wrdx/Desktop/spbu_ml_sem4/.venv/lib/python3.12/site-packages/sklearn/linear_model/_coordinate_descen
t.py:656: ConvergenceWarning: Objective did not converge. You might want to increase the number of iteratio
ns. Duality gap: 71419548056.59756, tolerance: 610957735.0128442
  model = cd_fast.sparse_enet_coordinate_descent(
/home/wrdx/Desktop/spbu_ml_sem4/.venv/lib/python3.12/site-packages/sklearn/linear_model/_coordinate_descen
t.py:656: ConvergenceWarning: Objective did not converge. You might want to increase the number of iteratio
ns. Duality gap: 29226560707.819305, tolerance: 619865924.1635104
  model = cd_fast.sparse_enet_coordinate_descent(
/home/wrdx/Desktop/spbu_ml_sem4/.venv/lib/python3.12/site-packages/sklearn/linear_model/_coordinate_descen
t.py:656: ConvergenceWarning: Objective did not converge. You might want to increase the number of iteratio
ns. Duality gap: 14269689297.186493, tolerance: 611603075.72175
  model = cd_fast.sparse_enet_coordinate_descent(
/home/wrdx/Desktop/spbu_ml_sem4/.venv/lib/python3.12/site-packages/sklearn/linear_model/_coordinate_descen
t.py:656: ConvergenceWarning: Objective did not converge. You might want to increase the number of iteratio
ns. Duality gap: 18960075515.418945, tolerance: 593335415.6996667
  model = cd_fast.sparse_enet_coordinate_descent(
/home/wrdx/Desktop/spbu_ml_sem4/.venv/lib/python3.12/site-packages/sklearn/linear_model/_coordinate_descen
t.py:656: ConvergenceWarning: Objective did not converge. You might want to increase the number of iteratio
ns. Duality gap: 12842790965.72284, tolerance: 610957735.0128442
  model = cd_fast.sparse_enet_coordinate_descent(
/home/wrdx/Desktop/spbu_ml_sem4/.venv/lib/python3.12/site-packages/sklearn/linear_model/_coordinate_descen
t.py:656: ConvergenceWarning: Objective did not converge. You might want to increase the number of iteratio
ns. Duality gap: 11436693030.353271, tolerance: 607603680.0773777
  model = cd_fast.sparse_enet_coordinate_descent(
```

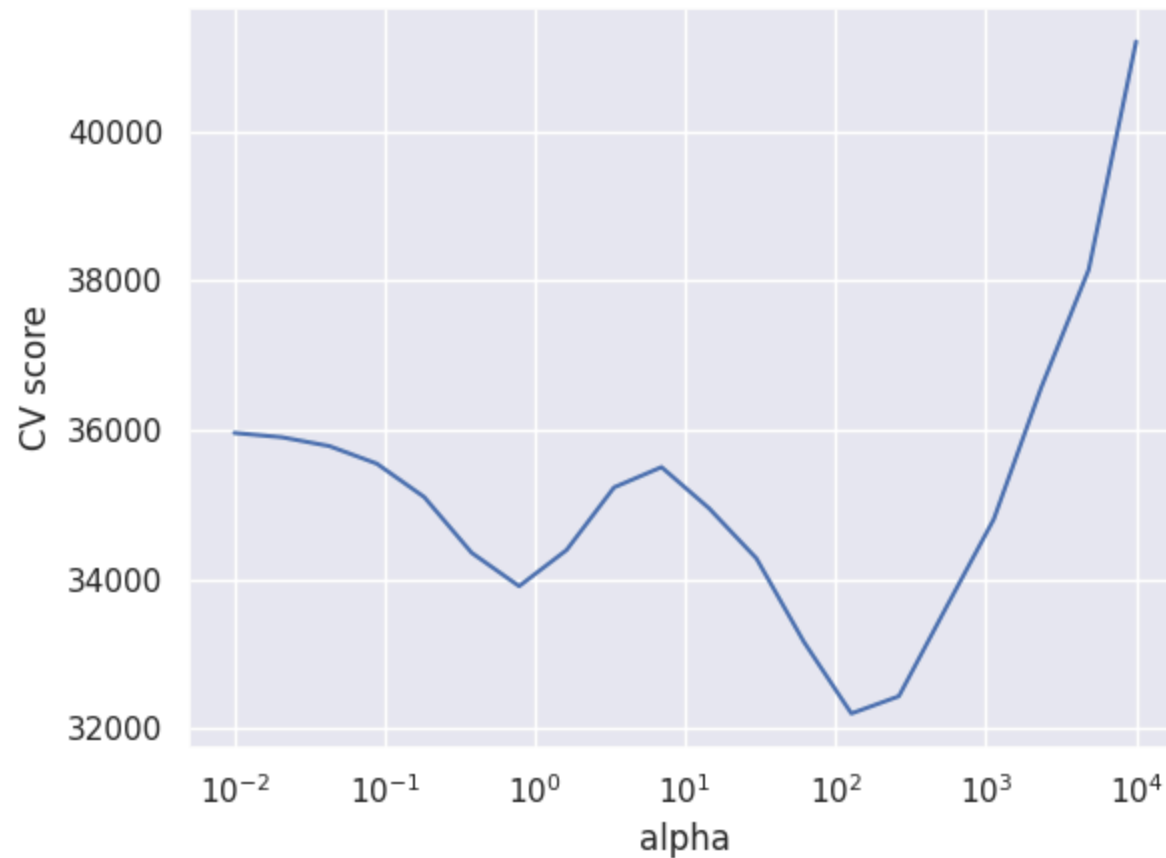
```
/home/wrdx/Desktop/spbu_ml_sem4/.venv/lib/python3.12/site-packages/sklearn/linear_model/_coordinate_descen
t.py:656: ConvergenceWarning: Objective did not converge. You might want to increase the number of iteratio
ns. Duality gap: 52398408264.83627, tolerance: 568471343.1704118
    model = cd_fast.sparse_enet_coordinate_descent(
/home/wrdx/Desktop/spbu_ml_sem4/.venv/lib/python3.12/site-packages/sklearn/linear_model/_coordinate_descen
t.py:656: ConvergenceWarning: Objective did not converge. You might want to increase the number of iteratio
ns. Duality gap: 1707158097.7772217, tolerance: 593335415.6996667
    model = cd_fast.sparse_enet_coordinate_descent(
/home/wrdx/Desktop/spbu_ml_sem4/.venv/lib/python3.12/site-packages/sklearn/linear_model/_coordinate_descen
t.py:656: ConvergenceWarning: Objective did not converge. You might want to increase the number of iteratio
ns. Duality gap: 15272881918.279907, tolerance: 603974920.6996291
    model = cd_fast.sparse_enet_coordinate_descent(
/home/wrdx/Desktop/spbu_ml_sem4/.venv/lib/python3.12/site-packages/sklearn/linear_model/_coordinate_descen
t.py:656: ConvergenceWarning: Objective did not converge. You might want to increase the number of iteratio
ns. Duality gap: 13862069940.492432, tolerance: 599961812.0571665
    model = cd_fast.sparse_enet_coordinate_descent(
/home/wrdx/Desktop/spbu_ml_sem4/.venv/lib/python3.12/site-packages/sklearn/linear_model/_coordinate_descen
t.py:656: ConvergenceWarning: Objective did not converge. You might want to increase the number of iteratio
ns. Duality gap: 1898869868.666504, tolerance: 611603075.72175
    model = cd_fast.sparse_enet_coordinate_descent(
/home/wrdx/Desktop/spbu_ml_sem4/.venv/lib/python3.12/site-packages/sklearn/linear_model/_coordinate_descen
t.py:656: ConvergenceWarning: Objective did not converge. You might want to increase the number of iteratio
ns. Duality gap: 11519105726.272827, tolerance: 598262324.2191347
    model = cd_fast.sparse_enet_coordinate_descent(
/home/wrdx/Desktop/spbu_ml_sem4/.venv/lib/python3.12/site-packages/sklearn/linear_model/_coordinate_descen
t.py:656: ConvergenceWarning: Objective did not converge. You might want to increase the number of iteratio
ns. Duality gap: 2204109196.8410645, tolerance: 568471343.1704118
    model = cd_fast.sparse_enet_coordinate_descent(
/home/wrdx/Desktop/spbu_ml_sem4/.venv/lib/python3.12/site-packages/sklearn/linear_model/_coordinate_descen
t.py:656: ConvergenceWarning: Objective did not converge. You might want to increase the number of iteratio
ns. Duality gap: 1431865611.0014648, tolerance: 610957735.0128442
    model = cd_fast.sparse_enet_coordinate_descent(
/home/wrdx/Desktop/spbu_ml_sem4/.venv/lib/python3.12/site-packages/sklearn/linear_model/_coordinate_descen
t.py:656: ConvergenceWarning: Objective did not converge. You might want to increase the number of iteratio
ns. Duality gap: 1053345853.1907349, tolerance: 599961812.0571665
    model = cd_fast.sparse_enet_coordinate_descent(
/home/wrdx/Desktop/spbu_ml_sem4/.venv/lib/python3.12/site-packages/sklearn/linear_model/_coordinate_descen
t.py:656: ConvergenceWarning: Objective did not converge. You might want to increase the number of iteratio
ns. Duality gap: 1759697609.45813, tolerance: 607603680.0773777
    model = cd_fast.sparse_enet_coordinate_descent(
/home/wrdx/Desktop/spbu_ml_sem4/.venv/lib/python3.12/site-packages/sklearn/linear_model/_coordinate_descen
t.py:656: ConvergenceWarning: Objective did not converge. You might want to increase the number of iteratio
```

```
ns. Duality gap: 2002981243.642517, tolerance: 603974920.6996291
  model = cd_fast.sparse_enet_coordinate_descent(
/home/wrdx/Desktop/spbu_ml_sem4/.venv/lib/python3.12/site-packages/sklearn/linear_model/_coordinate_descen
t.py:656: ConvergenceWarning: Objective did not converge. You might want to increase the number of iteratio
ns. Duality gap: 2121429597.8883667, tolerance: 598262324.2191347
  model = cd_fast.sparse_enet_coordinate_descent(
/home/wrdx/Desktop/spbu_ml_sem4/.venv/lib/python3.12/site-packages/sklearn/linear_model/_coordinate_descen
t.py:656: ConvergenceWarning: Objective did not converge. You might want to increase the number of iteratio
ns. Duality gap: 3010179850.9106445, tolerance: 593335415.6996667
  model = cd_fast.sparse_enet_coordinate_descent(
/home/wrdx/Desktop/spbu_ml_sem4/.venv/lib/python3.12/site-packages/sklearn/linear_model/_coordinate_descen
t.py:656: ConvergenceWarning: Objective did not converge. You might want to increase the number of iteratio
ns. Duality gap: 2014818693.4418335, tolerance: 610957735.0128442
  model = cd_fast.sparse_enet_coordinate_descent(
/home/wrdx/Desktop/spbu_ml_sem4/.venv/lib/python3.12/site-packages/sklearn/linear_model/_coordinate_descen
t.py:656: ConvergenceWarning: Objective did not converge. You might want to increase the number of iteratio
ns. Duality gap: 2052690405.7070312, tolerance: 599961812.0571665
  model = cd_fast.sparse_enet_coordinate_descent(
/home/wrdx/Desktop/spbu_ml_sem4/.venv/lib/python3.12/site-packages/sklearn/linear_model/_coordinate_descen
t.py:656: ConvergenceWarning: Objective did not converge. You might want to increase the number of iteratio
ns. Duality gap: 2837620274.7385254, tolerance: 568471343.1704118
  model = cd_fast.sparse_enet_coordinate_descent(
/home/wrdx/Desktop/spbu_ml_sem4/.venv/lib/python3.12/site-packages/sklearn/linear_model/_coordinate_descen
t.py:656: ConvergenceWarning: Objective did not converge. You might want to increase the number of iteratio
ns. Duality gap: 3014160574.0756836, tolerance: 598262324.2191347
  model = cd_fast.sparse_enet_coordinate_descent(
/home/wrdx/Desktop/spbu_ml_sem4/.venv/lib/python3.12/site-packages/sklearn/linear_model/_coordinate_descen
t.py:656: ConvergenceWarning: Objective did not converge. You might want to increase the number of iteratio
ns. Duality gap: 4389639570.378784, tolerance: 593335415.6996667
  model = cd_fast.sparse_enet_coordinate_descent(
/home/wrdx/Desktop/spbu_ml_sem4/.venv/lib/python3.12/site-packages/sklearn/linear_model/_coordinate_descen
t.py:656: ConvergenceWarning: Objective did not converge. You might want to increase the number of iteratio
ns. Duality gap: 2875751771.6954956, tolerance: 603974920.6996291
  model = cd_fast.sparse_enet_coordinate_descent(
/home/wrdx/Desktop/spbu_ml_sem4/.venv/lib/python3.12/site-packages/sklearn/linear_model/_coordinate_descen
t.py:656: ConvergenceWarning: Objective did not converge. You might want to increase the number of iteratio
ns. Duality gap: 4141958945.4595947, tolerance: 568471343.1704118
  model = cd_fast.sparse_enet_coordinate_descent(
/home/wrdx/Desktop/spbu_ml_sem4/.venv/lib/python3.12/site-packages/sklearn/linear_model/_coordinate_descen
t.py:656: ConvergenceWarning: Objective did not converge. You might want to increase the number of iteratio
ns. Duality gap: 2496711376.85791, tolerance: 607603680.0773777
  model = cd_fast.sparse_enet_coordinate_descent(
```



```
/home/wrdx/Desktop/spbu_ml_sem4/.venv/lib/python3.12/site-packages/sklearn/linear_model/_coordinate_descen
t.py:656: ConvergenceWarning: Objective did not converge. You might want to increase the number of iteratio
ns. Duality gap: 3289442803.0151367, tolerance: 599961812.0571665
    model = cd_fast.sparse_enet_coordinate_descent(
/home/wrdx/Desktop/spbu_ml_sem4/.venv/lib/python3.12/site-packages/sklearn/linear_model/_coordinate_descen
t.py:656: ConvergenceWarning: Objective did not converge. You might want to increase the number of iteratio
ns. Duality gap: 3603339320.1621704, tolerance: 607603680.0773777
    model = cd_fast.sparse_enet_coordinate_descent(
/home/wrdx/Desktop/spbu_ml_sem4/.venv/lib/python3.12/site-packages/sklearn/linear_model/_coordinate_descen
t.py:656: ConvergenceWarning: Objective did not converge. You might want to increase the number of iteratio
ns. Duality gap: 4187405452.336975, tolerance: 603974920.6996291
    model = cd_fast.sparse_enet_coordinate_descent(
/home/wrdx/Desktop/spbu_ml_sem4/.venv/lib/python3.12/site-packages/sklearn/linear_model/_coordinate_descen
t.py:656: ConvergenceWarning: Objective did not converge. You might want to increase the number of iteratio
ns. Duality gap: 4377351878.776978, tolerance: 598262324.2191347
    model = cd_fast.sparse_enet_coordinate_descent(
Best alpha = 127.4275
```

```
Out[287... Text(0, 0.5, 'CV score')
```



```
In [288... column_transformer = ColumnTransformer(
    [("ohe", OneHotEncoder(handle_unknown="ignore"), categorical), ("scaling", StandardScaler(), numeric_features)]
)

pipeline = Pipeline(steps=[("ohe_and_scaling", column_transformer), ("regression", Lasso(best_alpha))])

model = pipeline.fit(X_train, y_train)
y_pred = model.predict(X_val)
print("Test RMSE = %.4f" % mean_squared_error(y_val, y_pred, squared=False))
```

Test RMSE = 26800.4043

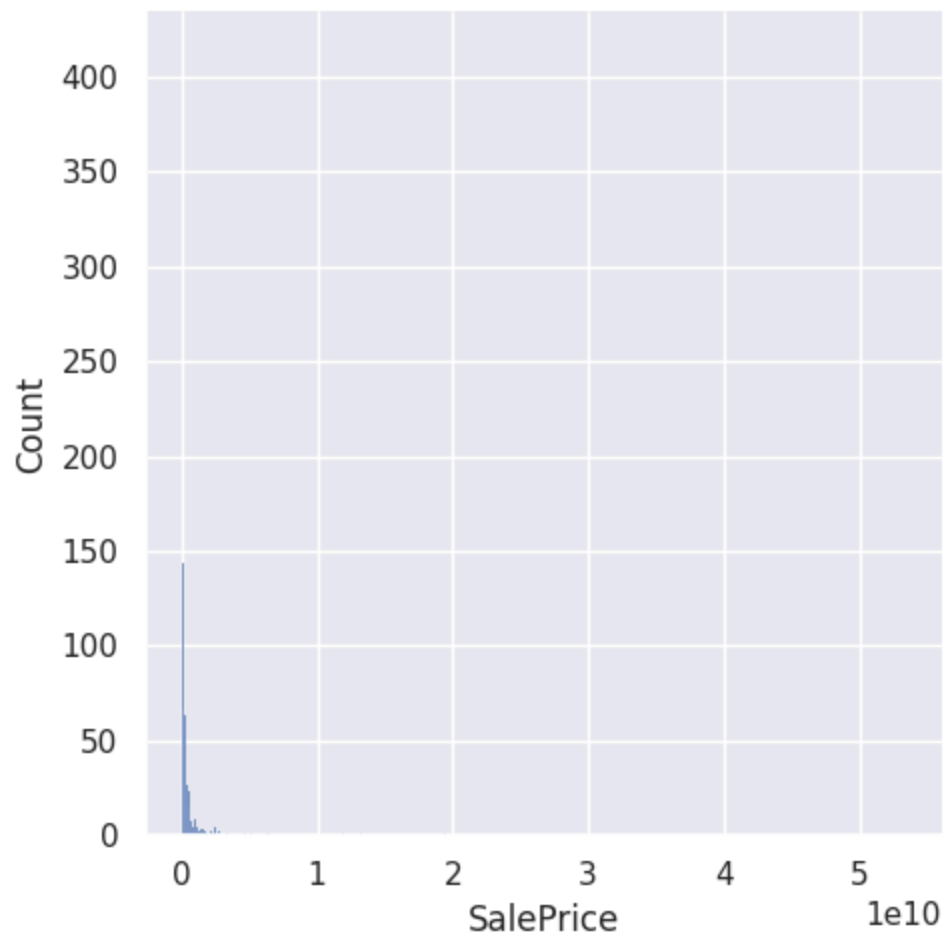
```
In [289... lasso_zeros = np.sum(pipeline.steps[-1][-1].coef_ == 0)
print("Zero weights in Lasso:", lasso_zeros)
```

Zero weights in Lasso: 195

Иногда очень полезно посмотреть на распределение остатков. Нарисуем гистограмму распределения квадратичной ошибки на обучающих объектах:

```
In [290... error = (y_train - model.predict(X_train)) ** 2  
sns.displot(error)
```

```
Out[290... <seaborn.axisgrid.FacetGrid at 0x74726b4d89e0>
```



Как видно из гистограммы, есть примеры с очень большими остатками. Попробуем их выбросить из обучающей выборки. Например, выбросим примеры, остаток у которых больше 0.95-квантили.

```
In [291...] mask = error < np.quantile(error, 0.95)
```

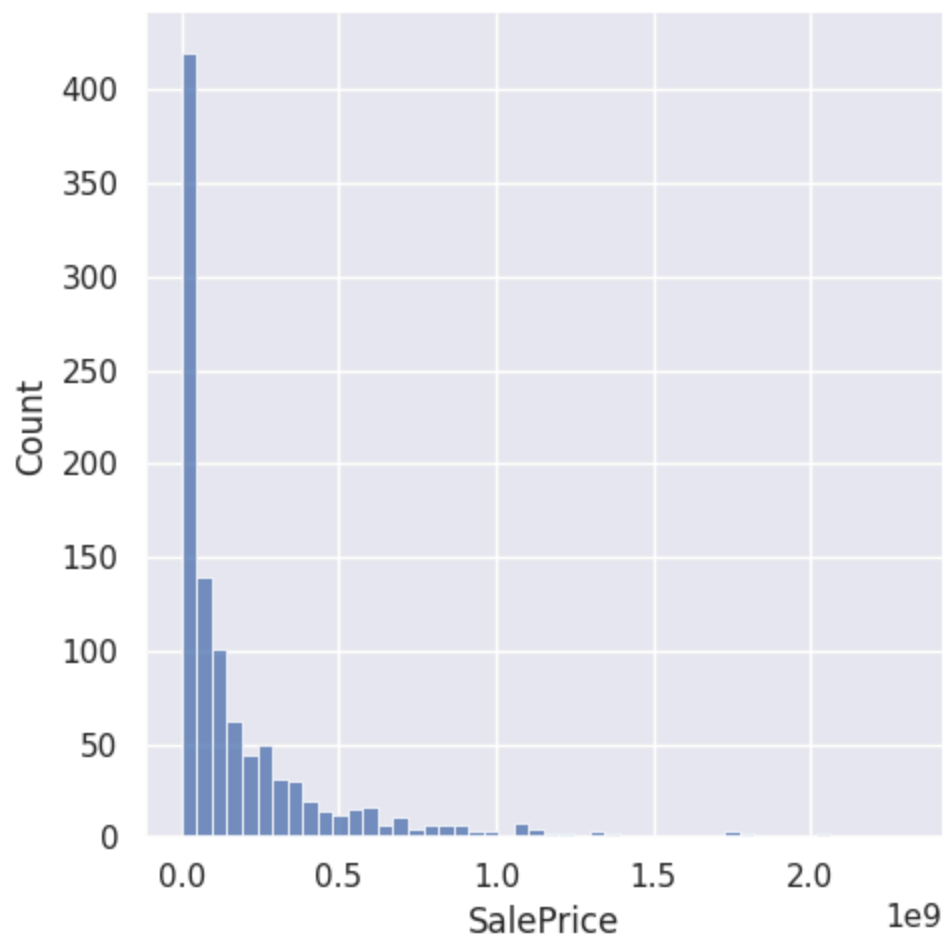
```
In [292...] column_transformer = ColumnTransformer(  
    [ ("ohe", OneHotEncoder(handle_unknown="ignore"), categorical), ("scaling", StandardScaler(), numeric_features) ],  
    remainder="passthrough")  
  
pipeline = Pipeline(steps=[ ("ohe_and_scaling", column_transformer), ("regression", Lasso(best_alpha)) ])  
  
model = pipeline.fit(X_train[mask], y_train[mask])  
y_pred = model.predict(X_val)  
print("Test RMSE = %.4f" % mean_squared_error(y_val, y_pred, squared=False))
```

Test RMSE = 26597.4640

```
In [293...] X_train = X_train[mask]  
            y_train = y_train[mask]
```

```
In [294...] error = (y_train[mask] - model.predict(X_train[mask])) ** 2  
sns.displot(error)
```

```
Out[294...] <seaborn.axisgrid.FacetGrid at 0x74726c18d340>
```



Видим, что качество модели заметно улучшилось! Также бывает очень полезно посмотреть на примеры с большими остатками и попытаться понять, почему же модель на них так сильно ошибается: это может дать понимание, как модель можно улучшить.

Часть 4. Подготовка данных для линейных моделей

Есть важное понятие, связанное с применением линейных моделей, — *спрямляющее пространство*. Под ним понимается такое признаковое пространство для наших объектов, в котором линейная модель хорошо описывает данные, даёт хорошее качество прогнозов.

Не существует общих рекомендаций о том, как найти спрямляющее пространство для произвольной выборки. Есть лишь некоторые общие советы — например, если добавить в выборку полиномиальных признаков, то скорее всего модель станет работать лучше (если не переобучится). Есть и другие трюки.

У линейных моделей есть огромное преимущество: они имеют мало параметров, а поэтому их можно обучить даже на небольшой выборке. Если выборка большая, то параметры модели получится оценить более надёжно — но в то же время есть риск, что данные будут слишком разнообразными, чтобы линейная модель могла уловить все закономерности в них. Иногда можно улучшить ситуацию путём разбиения признакового пространства на несколько областей и построения своей модели в каждой из них.

Попробуем для примера в нашей задаче разделить выборку на две части по признаку OverallQual. Это один из самых сильных признаков, и, возможно, разбиение по нему даст нам две выборки с заведомо разными ценами на дома.

Для начала вспомним, какое качество получается у обычной гребневой регрессии.

```
In [295... column_transformer = ColumnTransformer(
    [("ohe", OneHotEncoder(handle_unknown="ignore"), categorical), ("scaling", StandardScaler(), numeric_features)]
)

pipeline = Pipeline(steps=[("ohe_and_scaling", column_transformer), ("regression", Ridge())])

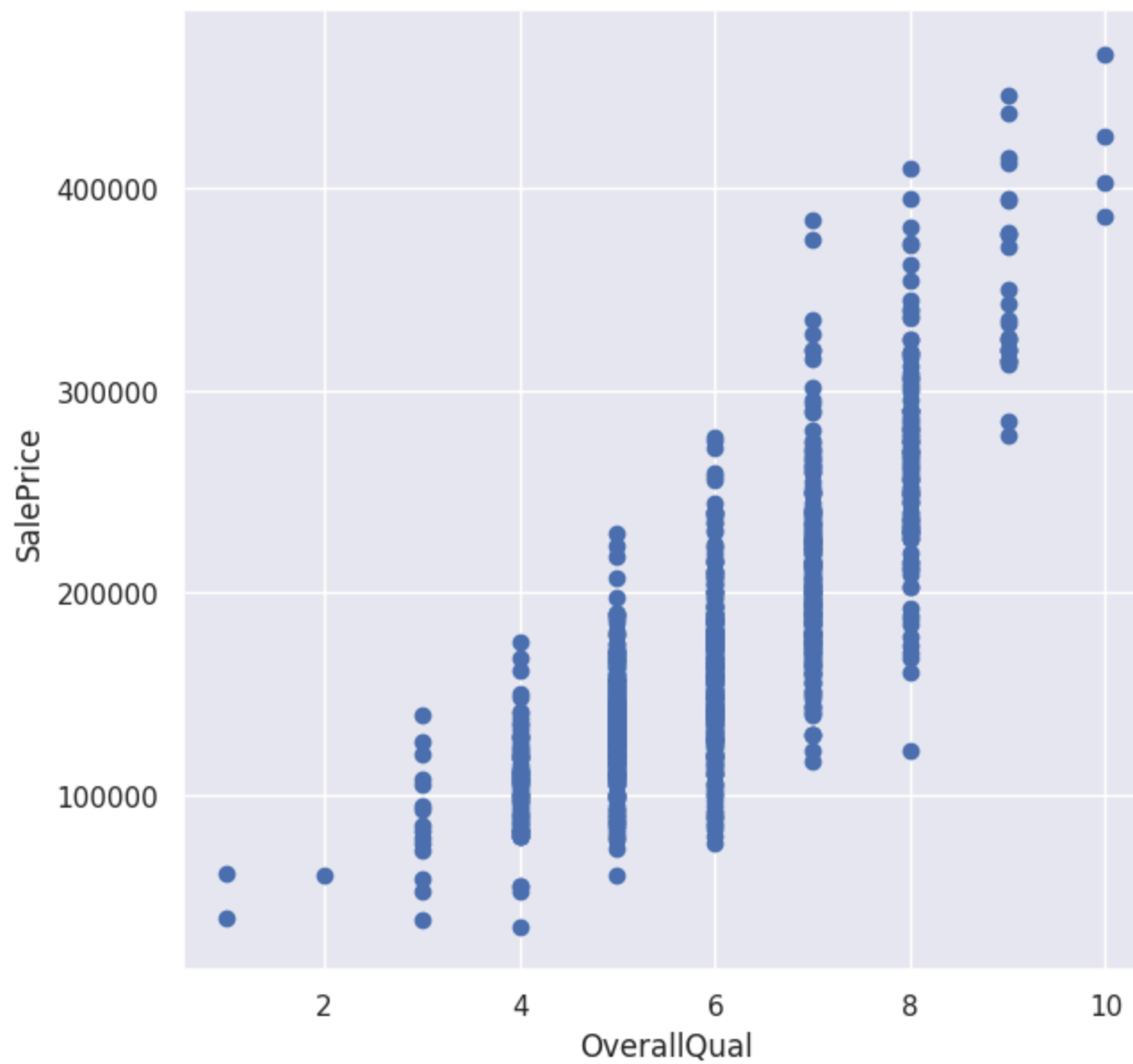
model = pipeline.fit(X_train, y_train)
y_pred = model.predict(X_val)
print("Test RMSE = %.4f" % mean_squared_error(y_val, y_pred, squared=False))
```

Test RMSE = 26167.2059

Посмотрим на связь OverallQual и целевой переменной.

```
In [296... plt.figure(figsize=(7, 7))
plt.scatter(X_train["OverallQual"], y_train)
plt.xlabel("OverallQual")
plt.ylabel("SalePrice")
```

```
Out[296... Text(0, 0.5, 'SalePrice')
```



```
In [297... threshold = 5
mask = X_train["OverallQual"] <= threshold
X_train_1 = X_train[mask]
y_train_1 = y_train[mask]
```

```
X_train_2 = X_train[~mask]
y_train_2 = y_train[~mask]
```

```
In [298... column_transformer1 = ColumnTransformer(
    [("ohe", OneHotEncoder(handle_unknown="ignore"), categorical), ("scaling", StandardScaler(), numeric_fe
)]

pipeline1 = Pipeline(steps=[("ohe_and_scaling", column_transformer1), ("regression", Ridge())])

column_transformer2 = ColumnTransformer(
    [("ohe", OneHotEncoder(handle_unknown="ignore"), categorical), ("scaling", StandardScaler(), numeric_fe
)]

pipeline2 = Pipeline(steps=[("ohe_and_scaling", column_transformer2), ("regression", Ridge())])

model1 = pipeline1.fit(X_train_1, y_train_1)
model2 = pipeline2.fit(X_train_2, y_train_2)

y_pred_1 = model1.predict(X_val)
y_pred_2 = model2.predict(X_val)
mask_test = X_val["OverallQual"] <= threshold
y_pred = y_pred_1.copy()
y_pred[~mask_test] = y_pred_2[~mask_test]

print("Test RMSE = %.4f" % mean_squared_error(y_val, y_pred, squared=False))
```

Test RMSE = 25759.4559

Получилось лучше! И это при практически случайном выборе разбиения. Если бы мы поработали над этим получше, то и качество, скорее всего, получилось бы выше.

Перейдём к следующему трюку — бинаризации признаков. Мы выбираем n порогов t_1, \dots, t_n для признака x_j и генерируем $n + 1$ новый признак: $[x_j \leq t_1], [t_1 < x_j \leq t_2], \dots, [t_{n-1} < x_j \leq t_n], [x_j > t_n]$. Такое преобразование может неплохо помочь в случае, если целевая переменная нелинейно зависит от одного из признаков. Рассмотрим синтетический пример.

```
In [299... x_plot = np.linspace(0, 1, 10000)

X = np.random.uniform(0, 1, size=30)
```



```

y = np.cos(1.5 * np.pi * X) + np.random.normal(scale=0.1, size=X.shape)

fig, axs = plt.subplots(figsize=(16, 4), ncols=2)

regr = LinearRegression()
regr.fit(X[:, np.newaxis], y)
y_pred_regr = regr.predict(x_plot[:, np.newaxis])
axs[0].scatter(X[:, np.newaxis], y, label="Data")
axs[0].plot(x_plot, y_pred_regr, label="Predictions")
axs[0].legend()
axs[0].set_title("Linear regression on original feature")
axs[0].set_xlabel("$X$")
axs[0].set_ylabel("$y$")
axs[0].set_ylim(-2, 2)

binner = KBinsDiscretizer(n_bins=5, strategy="quantile")
pipeline = Pipeline(steps=[("binning", binner), ("regression", LinearRegression())])
pipeline.fit(X[:, np.newaxis], y)
y_pred_binned = pipeline.predict(x_plot[:, np.newaxis])
axs[1].scatter(X[:, np.newaxis], y, label="Data")
axs[1].plot(x_plot, y_pred_binned, label="Predictions")
axs[1].set_title("Linear regression on binned feature")
axs[1].set_xlabel("$X$")
axs[1].set_ylabel("$y$")
axs[1].set_ylim(-2, 2)

```

Out[299... (-2.0, 2.0)



Видно, что качество модели существенно возросло. С другой стороны, увеличилось и количество параметров модели (из-за увеличения числа признаков), поэтому при бинаризации важно контролировать переобучение.

Иногда может помочь преобразование целевой переменной. Может оказаться, что по мере роста признаков целевая переменная меняется экспоненциально. Например, может оказаться, что при линейном уменьшении продолжительности видео число его просмотров растёт экспоненциально. Учесть это можно с помощью логарифмирования целевой переменной — ниже синтетический пример с такой ситуацией.

```
In [300... X = np.random.exponential(1, size=30)
y = np.exp(X) + np.random.normal(scale=0.1, size=X.shape)

x_plot = np.linspace(np.min(X), np.max(X), 10000)

fig, axs = plt.subplots(figsize=(16, 4), ncols=2)

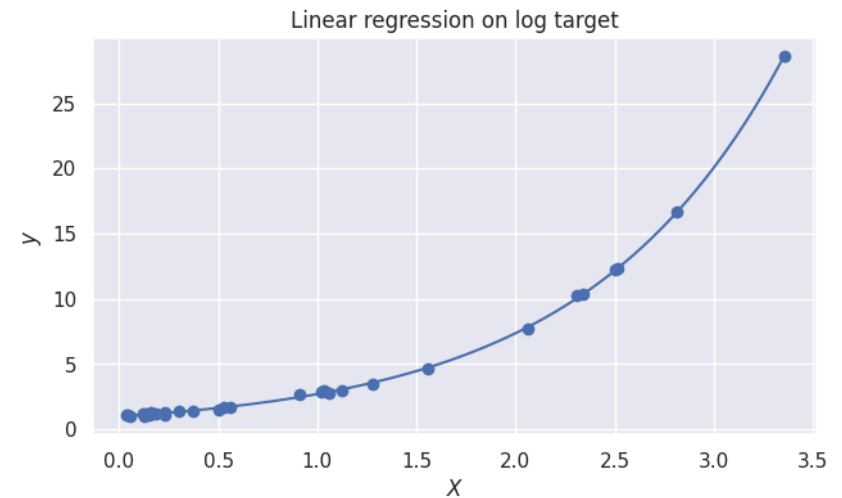
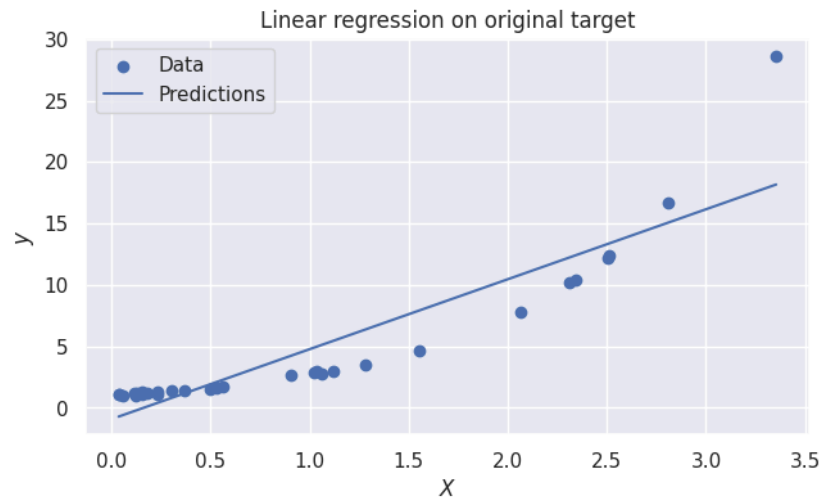
regr = LinearRegression()
regr.fit(X[:, np.newaxis], y)
y_pred_regr = regr.predict(x_plot[:, np.newaxis])
axs[0].scatter(X[:, np.newaxis], y, label="Data")
axs[0].plot(x_plot, y_pred_regr, label="Predictions")
axs[0].legend()
axs[0].set_title("Linear regression on original target")
axs[0].set_xlabel("$X$")
axs[0].set_ylabel("$y$")
```

```

y_log = np.log(y)
regr.fit(X[:, np.newaxis], y_log)
y_pred_log = np.exp(regr.predict(x_plot[:, np.newaxis]))
axs[1].scatter(X[:, np.newaxis], y, label="Data")
axs[1].plot(x_plot, y_pred_log, label="Predictions")
axs[1].set_title("Linear regression on log target")
axs[1].set_xlabel("$X$")
axs[1].set_ylabel("$y$")

```

Out[300... Text(0, 0.5, '\$y\$')



Но, конечно, вряд ли в реальных данных будет действительно экспоненциальная связь между целевой переменной и линейной комбинацией признаков. Тем не менее, логарифмирование всё равно может помочь.

Часть 5. Свободный полет

Оцените качество лучшей модели на тестовой выборке. Сравните с результатами на leaderboard, вы также можете отправить свое решение. Вы можете продолжить экспериментировать и постараться получить качество как можно выше.

```

In [301... test_data = pd.read_csv("test.csv")
test_data.head()

```

```
Out[301...
```

	Id	MSSubClass	MSZoning	LotFrontage	LotArea	Street	Alley	LotShape	LandContour	Utilities	...	ScreenPorch	Poo
0	1461	20	RH	80.0	11622	Pave	NaN	Reg	Lvl	AllPub	...	120	
1	1462	20	RL	81.0	14267	Pave	NaN	IR1	Lvl	AllPub	...	0	
2	1463	60	RL	74.0	13830	Pave	NaN	IR1	Lvl	AllPub	...	0	
3	1464	60	RL	78.0	9978	Pave	NaN	IR1	Lvl	AllPub	...	0	
4	1465	120	RL	43.0	5005	Pave	NaN	IR1	HLS	AllPub	...	144	

5 rows × 80 columns



```
In [302... ids = test_data["Id"]
X_test = test_data.drop(["Id"], axis=1)
```

```
In [303... numeric_data = X_test.select_dtypes([np.number])
numeric_features = numeric_data.columns
X_test[numeric_features] = X_test[numeric_features].fillna(X_train[numeric_features].mean())
```

```
In [304... categorical = list(X_train.dtypes[X_train.dtypes == "object"].index)
X_test[categorical] = X_test[categorical].fillna("NotGiven")
```

```
In [305... threshold = 5
mask = X_train["OverallQual"] <= threshold
X_train_1 = X_train[mask]
y_train_1 = y_train[mask]
X_train_2 = X_train[~mask]
y_train_2 = y_train[~mask]
```

```
In [306... column_transformer1 = ColumnTransformer(
    [("ohe", OneHotEncoder(handle_unknown="ignore"), categorical), ("scaling", StandardScaler(), numeric_f
)]

pipeline1 = Pipeline(steps=[("ohe_and_scaling", column_transformer1), ("regression", Ridge())])

column_transformer2 = ColumnTransformer(
    [("ohe", OneHotEncoder(handle_unknown="ignore"), categorical), ("scaling", StandardScaler(), numeric_f
)]
```

```
pipeline2 = Pipeline(steps=[("ohe_and_scaling", column_transformer2), ("regression", Ridge())])

model1 = pipeline1.fit(X_train_1, y_train_1)
model2 = pipeline2.fit(X_train_2, y_train_2)

y_pred_1 = model1.predict(X_test)
y_pred_2 = model2.predict(X_test)
mask_test = X_test["OverallQual"] <= threshold
y_pred = y_pred_1.copy()
y_pred[~mask_test] = y_pred_2[~mask_test]
```

```
In [307... output = pd.DataFrame({"Id": ids, "SalePrice": y_pred})

output.to_csv('predict.csv', index=False)
```

```
In [254... # Kaggle Score: 0.14896
```