House Prices: Advanced Regression Techniques

Kaggle kompetition

dima ziaziulia

Задача: по исходным данным предсказать стоимость дома. Регрессия.

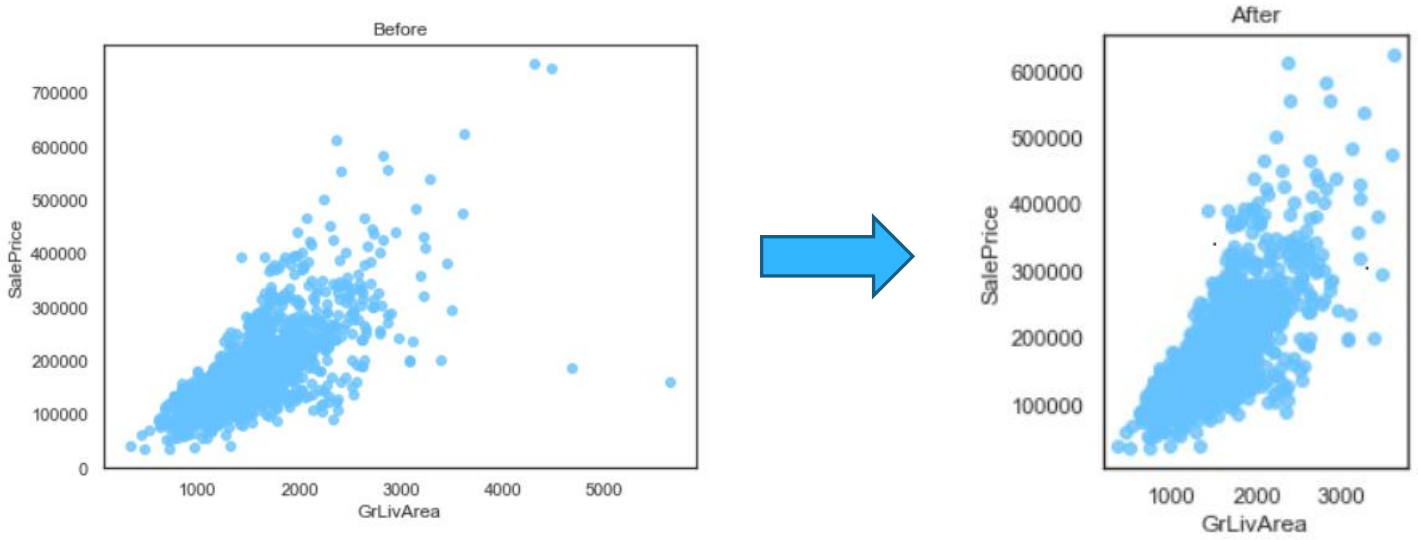
Действия:

1. EDA - загрузка и изучение данных, нахождение выбросов, замена пропусков, создание новых признаков.

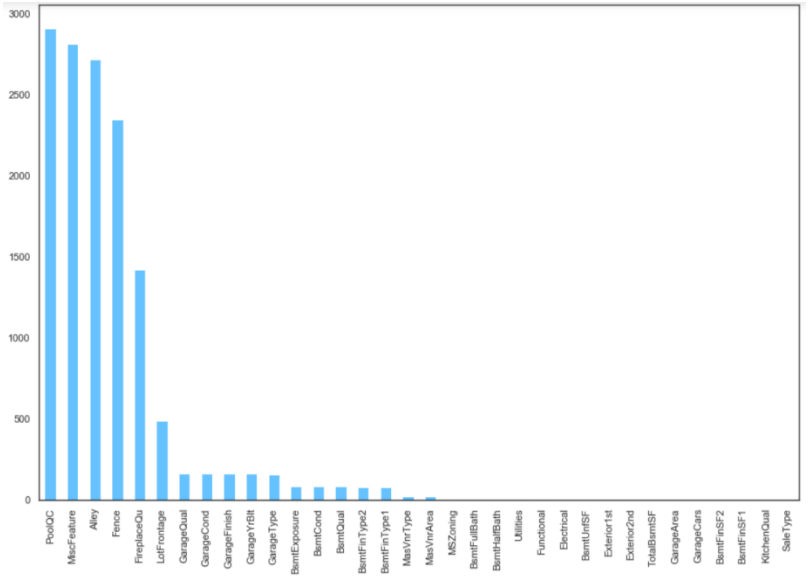
2. MODELING - выбор алгоритма, настройка и поиск оптимальных параметров. Применение нейросети, используя библиотеку PyTorch.

3. STACKING and BLANDING - использование одновременно нескольких алгоритмов для решения одной задачи.

Нашёл и удалил выбросы:



Нашёл и решил проблему пропусков данных:



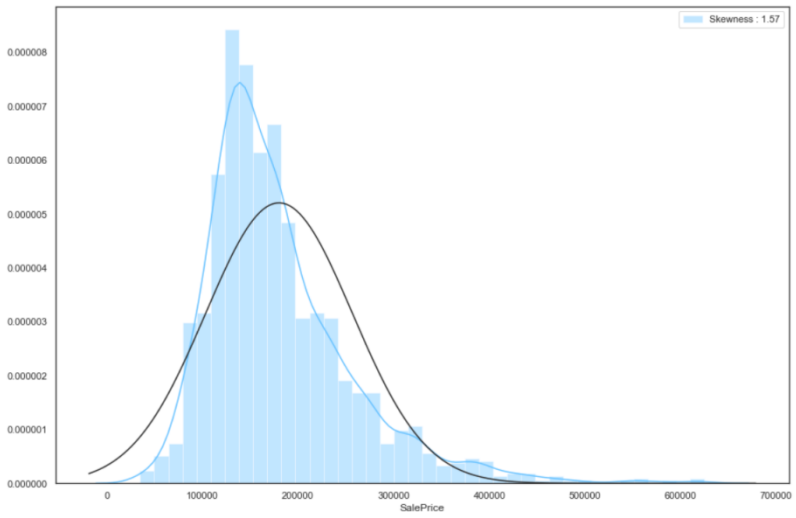
```
# From inspection, we can remove Utilities
all_data = all_data.drop(['Utilities'], axis=1)

all_data_na = all_data.isnull().sum()
print("Features with missing values: ", len(all_data_na.drop(all_data_na[all_data_na == 0].index)))
```

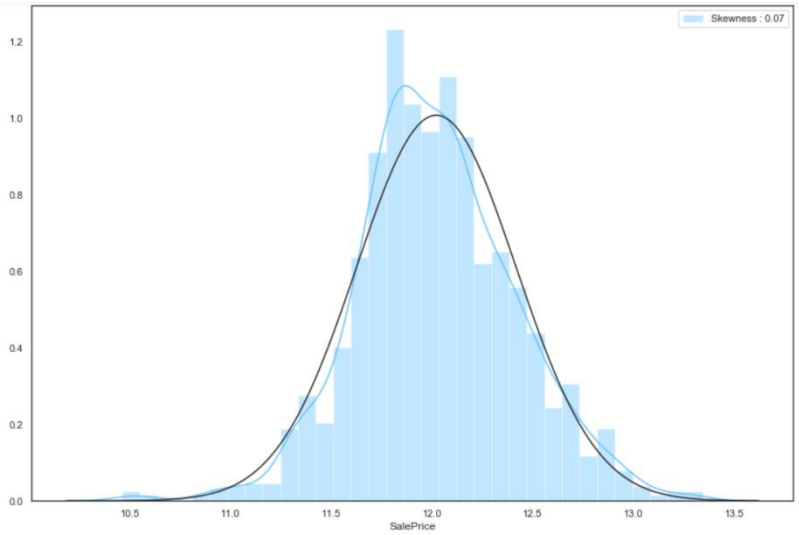
```
Features with missing values:  0
```

Получил нормальное распределение SalePrice:

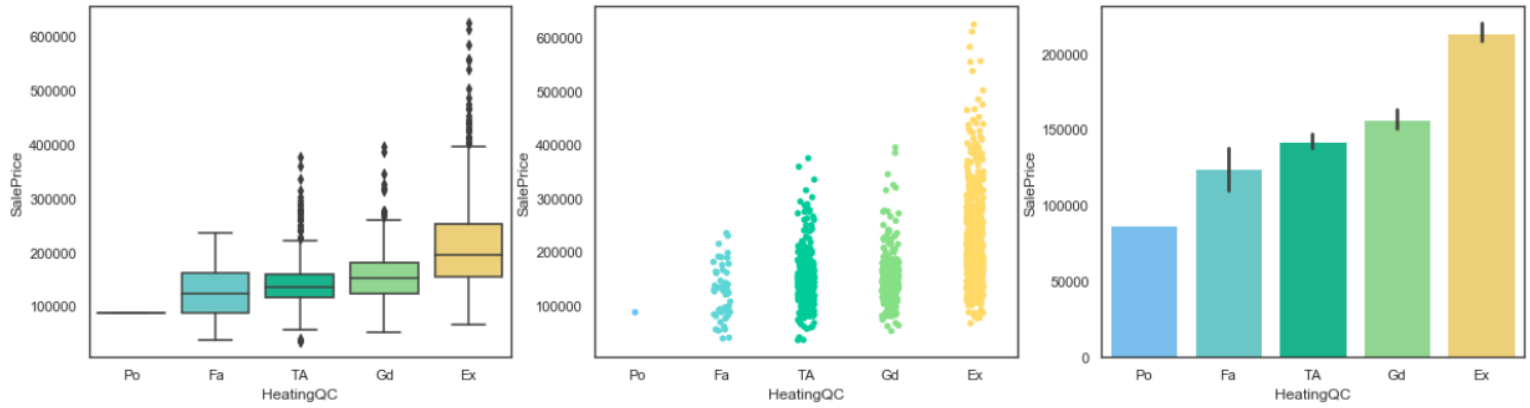
Исходное распределение данных:



Результат обработки :



Изучил каждый признак по графикам распределений:



Здесь мы видим положительную корреляцию с SalePrice по мере повышения качества отопления. С буквой "Ex" SalePrice - самая высокая средняя цена. Мы также видим большое количество домов с таким качеством отопления, а это значит, что большинство домов имеют очень хорошее отопление! Это категорическая особенность, но так как она показывает порядок, я заменю значения вручную цифрами.

Feature Engineering:

```
print("all_data shape: {}".format(all_data.shape))
```

```
all_data shape: (2915, 79)
```

```
print(all_data.shape)
```

```
(2915, 330)
```

Нашёл лучший алгоритм и оптимальные параметры:

```
[ ] model=Lasso(random_state=1)

[ ] ww = np.linspace(-0.1,0.1,20)

[ ] LASS_param_grid = {'alpha': [0.0005050505050505048], 'max_iter': [100,200,300,350,400,500], 'tol': ww}

[ ] GS = GridSearchCV(model,LASS_param_grid, scoring = 'neg_mean_absolute_error', cv = 10 )

[ ] GS.fit(xgb_train,price)
```

```
GridSearchCV(cv=10, error_score=nan,
             estimator=Lasso(alpha=1.0, copy_X=True, fit_intercept=True,
                             max_iter=1000, normalize=False, positive=False,
                             precompute=False, random_state=1,
                             selection='cyclic', tol=0.0001, warm_start=False),
             iid='deprecated', n_jobs=None,
             param_grid={'alpha': [0.0005050505050505048],
                         'max_iter': [100, 200, 300, 350, 400, 500],
                         'tol': array([-0.1       , -0.08947368, -0.07894737, -0.06842105, -0.05789474,
       -0.04736842, -0.03684211, -0.02631579, -0.01578947, -0.00526316,
        0.00526316,  0.01578947,  0.02631579,  0.03684211,  0.04736842,
        0.05789474,  0.06842105,  0.07894737,  0.08947368,  0.1       ])},
             pre_dispatch='2*n_jobs', refit=True, return_train_score=False,
             scoring='neg_mean_absolute_error', verbose=0)
```

```
[ ] print(GS.best_params_)
    print(GS.best_score_)
```

```
{'alpha': 0.0005050505050505048, 'max_iter': 500, 'tol': -0.1}
-0.08483518465018755
```

С помощью 8 алгоритмов создал таблицу предсказаний:

```
[ ] stacked_validation_train.head()
```

	KernelRidge	ElasticNet	Lasso	Gradient Boosting	Bayesian Ridge	Lasso Lars IC	Random Forest	XGBoost
0	12.073728	12.052635	12.061486	12.115832	12.075085	12.044550	12.142664	12.105165
1	11.909246	11.918707	11.917863	12.041782	11.911173	11.956942	12.080748	12.023828
2	11.785589	11.784994	11.785688	11.874111	11.786584	11.768158	11.866465	11.766501
3	11.814794	11.807620	11.798751	11.838076	11.815880	11.793635	11.751453	11.756207
4	11.357140	11.353308	11.356007	11.175413	11.357378	11.363881	11.361253	11.344832

```
[ ] stacked_test_train.head()
```

	KernelRidge	ElasticNet	Lasso	Gradient Boosting	Bayesian Ridge	Lasso Lars IC	Random Forest	XGBoost
0	11.636958	11.640085	11.642436	11.720710	11.640184	11.657661	11.734794	11.681759
1	11.990914	11.987235	11.987553	11.959877	11.990921	11.956845	11.962638	11.990759
2	12.097899	12.087723	12.093394	12.130536	12.098948	12.039650	12.096746	12.120154
3	12.189059	12.182578	12.188087	12.145993	12.189636	12.123748	12.123311	12.101163
4	12.088862	12.096829	12.095181	12.135041	12.087660	12.126271	12.159151	12.121401

Лучший результат заблендил по формуле:

```
[44] ensemble3 = (meta_model_pred*(1/10) + final_predictions['XGBoost']*(1.5/10) + final_predictions['Gradient Boosting']*(2/10)
                + final_predictions['Bayesian Ridge']*(1/10) + final_predictions['Lasso']*(1/10) + final_predictions['KernelRidge']*(1/10)
                + final_predictions['Lasso Lars IC']*(1/10) + final_predictions['Random Forest']*(1.5/10))

     submission = pd.DataFrame()
     submission['Id'] =test2['Id']
     submission['SalePrice'] = ensemble3
     submission.to_csv('final_submission3.csv',index=False)
     print("Файл сохранён!")
```

```
Файл сохранён!
```