

哈 尔 滨 工 业 大 学 （ 深 圳 ）

**Harbin Institute of Technology
(Shenzhen)**

实验名称: _____

姓名: _____

学号: _____

日期: _____

1.实验内容

- 1. 利用各方各面的描述变量，采用特征工程、回归预测等关键技术预测爱荷华州艾姆斯每套住宅的最终价格。
- 2. 比赛网站: [House Prices – Advanced Regression Techniques | Kaggle](#)
- 3. 评价指标: 使用均方根误差 (RMSE) 为唯一评价指标，通过 `log()` 运算将价位高低不同的预测误差对最终结果的影响统一起来。

2.实验方法

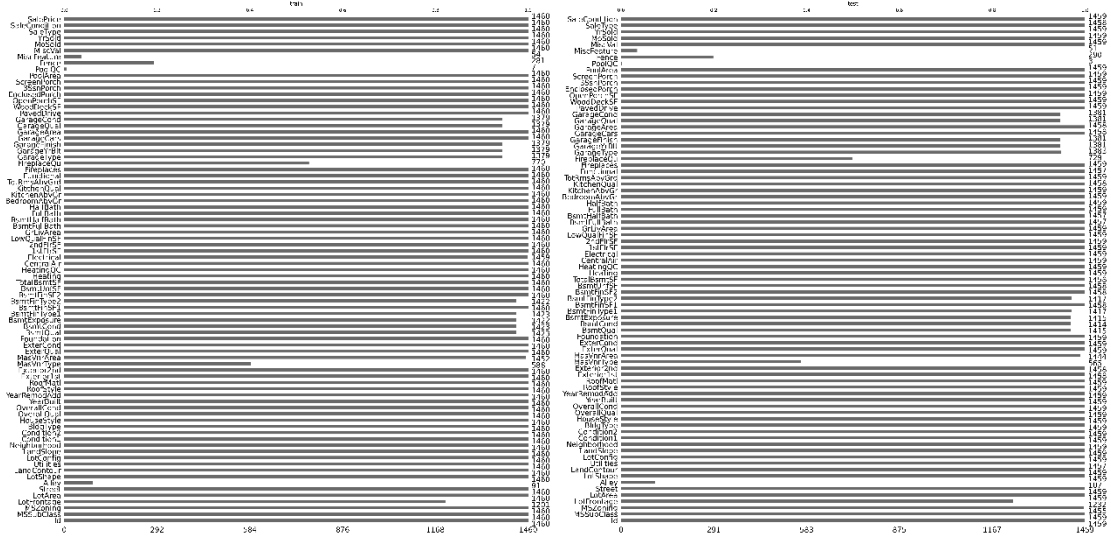
1. 观察初始数据

(1) 基本信息

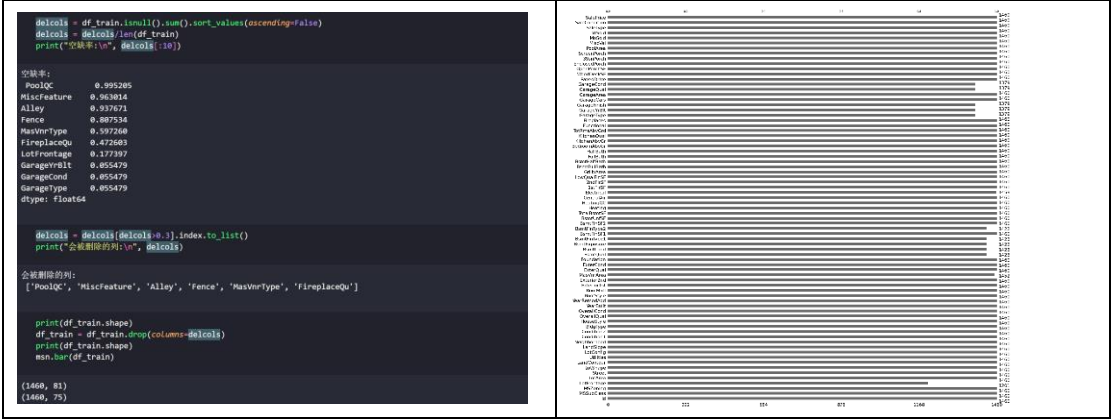
对于训练集，包括 `Id` 以及 `SalePrice` 在内，共有 81 个字段。训练集大小为 1461。

(2) 空缺值统计

统计 `train.csv` 和 `test.csv` 中各属性的 `null` 值，结果如下。可以很明显看出，无论是在训练集和测试集当中（左边为训练集，右边为测试集），部分字段的空缺值占比都很多。比如 `'PoolQC'`，`'MiscFeature'`，`'Alley'`，`'Fence'`，`'MasVnrType'`，`'FireplaceQu'` 6 个字段，空缺值占比几乎都在一半左右或者一半以上。



采取以 0.3 为空缺值删除阈值，删除空缺率大于 0.3 的字段。下左图为具体代码实现，右图为删除空缺值占比过大的字段之后的训练集。



(3) 划分离散型属性和连续型属性

对于离散型属性与连续型属性，有着不同的特征工程的实现方法。通过对属性的类型进行简单判断，来确定属性是离散型还是连续型。如下，列属性只有 int64, float64, object 三种 (object 一般对应一些字符串描述类型的字段，可以看作是离散型变量)。进行简单的划分，前两者数据类型看作是连续型属性，后者看作是离散型属性。划分结果：

➤ 连续型属性：38 个

当然连续型属性中的 Id 和 SalePrice 不参与训练，后续从训练集中删除。

```
['Id', 'MSSubClass', 'LotArea', 'OverallQual', 'OverallCond', 'YearBuilt', 'YearRemodAdd', 'BsmtFinSF1', 'BsmtFinSF2', 'BsmtUnfSF', 'TotalBsmtSF', '1stFlrSF', '2ndFlrSF', 'LowQualFinSF', 'GrLivArea', 'BsmtFullBath', 'BsmtHalfBath', 'FullBath', 'HalfBath', 'BedroomAbvGr', 'KitchenAbvGr', 'TotRmsAbvGrd', 'Fireplaces', 'GarageCars', 'GarageArea', 'WoodDeckSF', 'OpenPorchSF', 'EnclosedPorch', '3SsnPorch', 'ScreenPorch', 'PoolArea', 'MiscVal', 'MoSold', 'YrSold', 'SalePrice', 'LotFrontage', 'MasVnrArea', 'GarageYrBlt']
```

➤ 离散型属性：37 个

```
['MSZoning', 'Street', 'LotShape', 'LandContour', 'Utilities', 'LotConfig', 'LandSlope', 'Neighborhood', 'Condition1', 'Condition2', 'BldgType', 'HouseStyle', 'RoofStyle', 'RoofMatl', 'Exterior1st', 'Exterior2nd', 'ExterQual', 'ExterCond', 'Foundation', 'BsmtQual', 'BsmtCond', 'BsmtExposure', 'BsmtFinType1', 'BsmtFinType2', 'Heating', 'HeatingQC', 'CentralAir', 'Electrical', 'KitchenQual', 'Functional', 'GarageType', 'GarageFinish', 'GarageQual', 'GarageCond', 'PavedDrive', 'SaleType', 'SaleCondition']
```



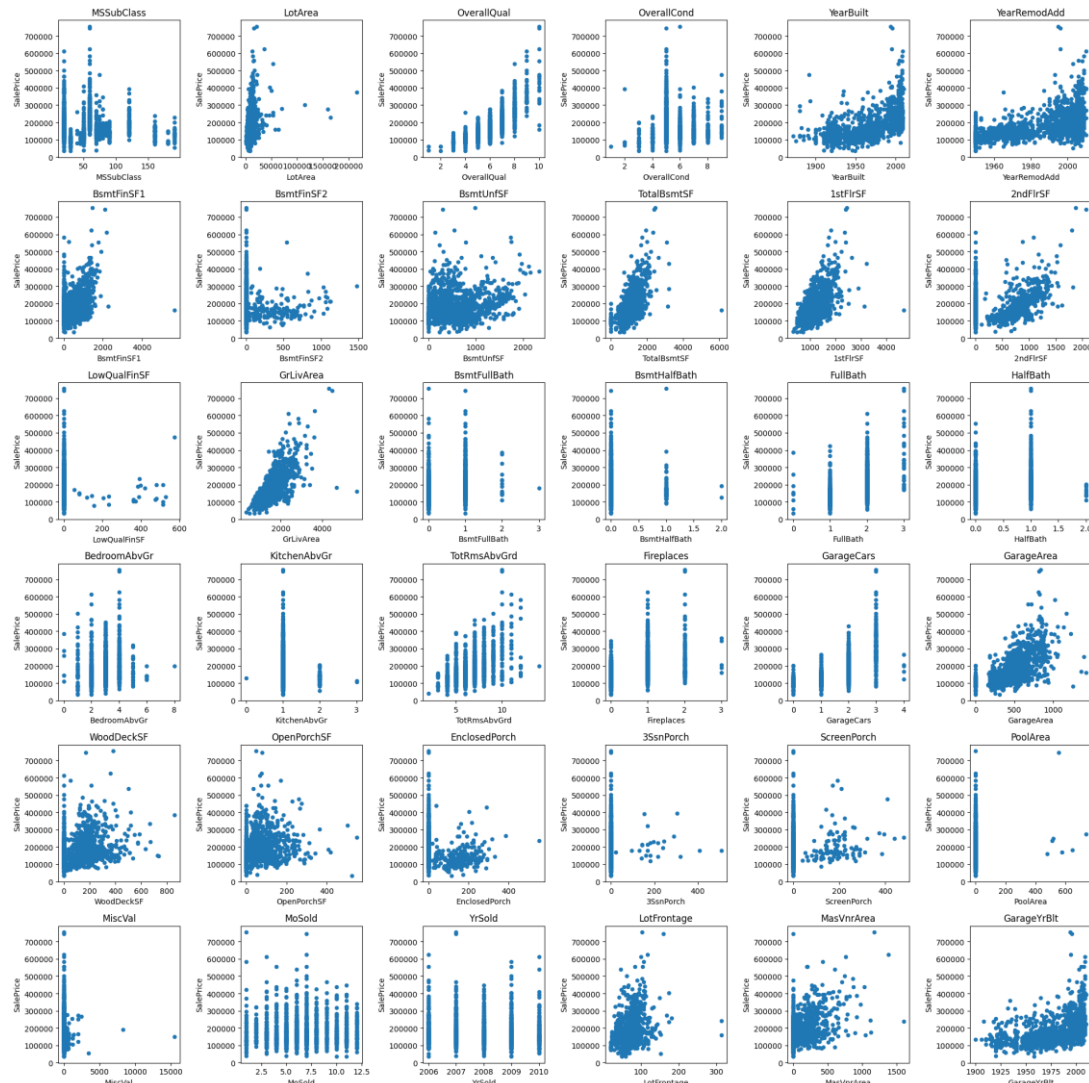
```
13 # 列的数据类型划分连续型属性和离散型属性
14
15 p = df.dtypes.to_dict()
16 dtype_colDict[str, list[str]] = dict()
17 for ele in set(p.values()):
18     temp = []
19     ele = str(ele)
20     for key in p.keys():
21         if p[key] == ele:
22             temp.append(key)
23     dtype_col[ele] = temp
24     print(ele, len(temp), temp)
25
26 object 37 ['MSZoning', 'Street', 'LotShape', 'LandContour', 'Utilities', 'LotConfig', 'LandSlope', 'Neighborhood', 'Condition1', 'Condition2', 'BldgType', 'HouseStyle', 'RoofStyle', 'RoofMatl', 'Exterior1st', 'Exterior2nd', 'ExterQual', 'ExterCond', 'Foundation', 'BsmtQual', 'BsmtCond', 'BsmtExposure', 'BsmtFinType1', 'BsmtFinType2', 'Heating', 'HeatingQC', 'CentralAir', 'Electrical', 'KitchenQual', 'Functional', 'GarageType', 'GarageFinish', 'GarageQual', 'GarageCond', 'PavedDrive', 'SaleType', 'SaleCondition']
27
28 int64 35 ['Id', 'MSSubClass', 'LotArea', 'OverallQual', 'OverallCond', 'YearBuilt', 'YearRemodAdd', 'BsmtFinSF1', 'BsmtFinSF2', 'BsmtUnfSF', 'TotalBsmtSF', '1stFlrSF', '2ndFlrSF', 'LowQualFinSF', 'TotRmsAbvGrd', 'Fireplaces', 'GarageCars', 'GarageArea', 'WoodDeckSF', 'OpenPorchSF', 'EnclosedPorch', '3SsnPorch', 'ScreenPorch', 'PoolArea', 'MiscVal', 'MoSold', 'YrSold', 'SalePrice', 'LotFrontage', 'MasVnrArea', 'GarageYrBlt']
29
30 float64 3 ['LotFrontage', 'MasVnrArea', 'GarageYrBlt']
31
32
33
34
35
36
37
38
39
40
41
42
43
44
45
46
47
48
49
50
51
52
53
54
55
56
57
58
59
60
61
62
63
64
65
66
67
68
69
70
71
72
73
74
75
76
77
78
79
80
81
82
83
84
85
86
87
88
89
90
91
92
93
94
95
96
97
98
99
100
101
102
103
104
105
106
107
108
109
110
111
112
113
114
115
116
117
118
119
120
121
122
123
124
125
126
127
128
129
130
131
132
133
134
135
136
137
138
139
140
141
142
143
144
145
146
147
148
149
150
151
152
153
154
155
156
157
158
159
160
161
162
163
164
165
166
167
168
169
170
171
172
173
174
175
176
177
178
179
180
181
182
183
184
185
186
187
188
189
190
191
192
193
194
195
196
197
198
199
200
201
202
203
204
205
206
207
208
209
210
211
212
213
214
215
216
217
218
219
220
221
222
223
224
225
226
227
228
229
230
231
232
233
234
235
236
237
238
239
240
241
242
243
244
245
246
247
248
249
250
251
252
253
254
255
256
257
258
259
260
261
262
263
264
265
266
267
268
269
270
271
272
273
274
275
276
277
278
279
280
281
282
283
284
285
286
287
288
289
290
291
292
293
294
295
296
297
298
299
300
301
302
303
304
305
306
307
308
309
310
311
312
313
314
315
316
317
318
319
320
321
322
323
324
325
326
327
328
329
330
331
332
333
334
335
336
337
338
339
340
341
342
343
344
345
346
347
348
349
350
351
352
353
354
355
356
357
358
359
360
361
362
363
364
365
366
367
368
369
370
371
372
373
374
375
376
377
378
379
380
381
382
383
384
385
386
387
388
389
390
391
392
393
394
395
396
397
398
399
400
401
402
403
404
405
406
407
408
409
410
411
412
413
414
415
416
417
418
419
420
421
422
423
424
425
426
427
428
429
430
431
432
433
434
435
436
437
438
439
440
441
442
443
444
445
446
447
448
449
450
451
452
453
454
455
456
457
458
459
460
461
462
463
464
465
466
467
468
469
470
471
472
473
474
475
476
477
478
479
480
481
482
483
484
485
486
487
488
489
490
491
492
493
494
495
496
497
498
499
500
501
502
503
504
505
506
507
508
509
510
511
512
513
514
515
516
517
518
519
520
521
522
523
524
525
526
527
528
529
530
531
532
533
534
535
536
537
538
539
540
541
542
543
544
545
546
547
548
549
550
551
552
553
554
555
556
557
558
559
560
561
562
563
564
565
566
567
568
569
570
571
572
573
574
575
576
577
578
579
580
581
582
583
584
585
586
587
588
589
590
591
592
593
594
595
596
597
598
599
600
601
602
603
604
605
606
607
608
609
610
611
612
613
614
615
616
617
618
619
620
621
622
623
624
625
626
627
628
629
630
631
632
633
634
635
636
637
638
639
640
641
642
643
644
645
646
647
648
649
650
651
652
653
654
655
656
657
658
659
660
661
662
663
664
665
666
667
668
669
670
671
672
673
674
675
676
677
678
679
680
681
682
683
684
685
686
687
688
689
690
691
692
693
694
695
696
697
698
699
700
701
702
703
704
705
706
707
708
709
710
711
712
713
714
715
716
717
718
719
720
721
722
723
724
725
726
727
728
729
730
731
732
733
734
735
736
737
738
739
740
741
742
743
744
745
746
747
748
749
750
751
752
753
754
755
756
757
758
759
760
761
762
763
764
765
766
767
768
769
770
771
772
773
774
775
776
777
778
779
780
781
782
783
784
785
786
787
788
789
790
791
792
793
794
795
796
797
798
799
800
801
802
803
804
805
806
807
808
809
810
811
812
813
814
815
816
817
818
819
820
821
822
823
824
825
826
827
828
829
830
831
832
833
834
835
836
837
838
839
840
841
842
843
844
845
846
847
848
849
850
851
852
853
854
855
856
857
858
859
860
861
862
863
864
865
866
867
868
869
870
871
872
873
874
875
876
877
878
879
880
881
882
883
884
885
886
887
888
889
890
891
892
893
894
895
896
897
898
899
900
901
902
903
904
905
906
907
908
909
910
911
912
913
914
915
916
917
918
919
920
921
922
923
924
925
926
927
928
929
930
931
932
933
934
935
936
937
938
939
940
941
942
943
944
945
946
947
948
949
950
951
952
953
954
955
956
957
958
959
960
961
962
963
964
965
966
967
968
969
970
971
972
973
974
975
976
977
978
979
980
981
982
983
984
985
986
987
988
989
990
991
992
993
994
995
996
997
998
999
1000
```

2. 数据预处理

(1) 连续型属性和离散型属性的数值分布

➤ 连续型属性：

由下图可以看出，虽然一些字段数据类型是 int 是 float，但实际上还是表示的不连续的数。比如第三行中倒数四个子图所对应的属性。



➤ 离散型属性:

一些离散型属性对应的字段数值很少，一些离散型属性对应的字段值很多。

```
for ele in type_col['discrete']:
    print(ele, df_train[ele].value_counts().to_dict())

[11]
MSZoning {'RL': 1151, 'RM': 218, 'FV': 65, 'RH': 16, 'C (all)': 10}
Street {'Pave': 1454, 'Grvl': 6}
LotShape {'Reg': 925, 'IR1': 484, 'IR2': 41, 'IR3': 10}
LandContour {'Lvl': 1311, 'Bnk': 63, 'HLS': 50, 'Low': 36}
Utilities {'AllPub': 1459, 'NoSeWa': 1}
LotConfig {'Inside': 1052, 'Corner': 263, 'CulDSac': 94, 'FR2': 47, 'FR3': 4}
LandSlope {'Gtl': 1382, 'Mod': 65, 'Sev': 13}
Neighborhood {'Names': 225, 'CollGr': 150, 'OldTown': 113, 'Edwards': 100, 'Somerst': 86, 'Gilbert': 79, 'Nridgnt': 77, 'Sawyer': 74, 'NWAmes': 73, 'SawyerW': 59, 'BrkSide': 58, 'Crawfo
Condition1 {'Norm': 1260, 'Feedr': 81, 'Artery': 48, 'RRAn': 26, 'PosN': 19, 'RAAe': 11, 'PosA': 8, 'RRAn': 5, 'RRNe': 2}
Condition2 {'Norm': 1415, 'Feedr': 6, 'Artery': 2, 'RRAn': 2, 'PosN': 2, 'PosA': 1, 'RAAe': 1}
BldgType {'1fam': 1220, 'Twnhse': 114, 'Duplex': 52, 'Twnhs': 43, '2fmCon': 31}
HouseStyle {'1Story': 726, '2Story': 445, '1.5Fin': 154, 'Svl': 65, 'SFoyer': 37, '1.5Unf': 14, '2.5Unf': 11, '2.5Fin': 8}
RoofStyle {'Gable': 1141, 'Hip': 286, 'Flat': 13, 'Gambrel': 11, 'Mansard': 7, 'Shed': 2}
RoofMatl {'CompShg': 1434, 'TarGrv': 11, 'WdShngl': 6, 'WdShake': 5, 'Metal': 1, 'Membran': 1, 'Roll': 1, 'ClyTile': 1}
Exterior1st {'VinylSd': 515, 'HdBoard': 222, 'MetalSd': 220, 'Wd Sdg': 206, 'Plywood': 108, 'CemntBd': 61, 'BrkFace': 50, 'WdShing': 26, 'Stucco': 25, 'AsbShng': 20, 'BrkComm': 2, 'Stor
Exterior2nd {'VinylSd': 504, 'MetalSd': 214, 'HdBoard': 207, 'Wd Sdg': 197, 'Plywood': 142, 'CemntBd': 60, 'Wd Shng': 38, 'Stucco': 26, 'BrkFace': 25, 'AsbShng': 20, 'ImStucc': 10, 'Br
ExterQual {'TA': 906, 'Gd': 488, 'Ex': 52, 'Fa': 14}
ExterCond {'TA': 1282, 'Gd': 146, 'Fa': 28, 'Ex': 3, 'Po': 1}
Foundation {'PCone': 647, 'CBlock': 634, 'BrkTill': 146, 'Slab': 24, 'Stone': 6, 'Wood': 3}
BsmQual {'TA': 649, 'Gd': 618, 'Ex': 121, 'Fa': 35}
BsmCond {'TA': 1311, 'Gd': 65, 'Fa': 45, 'Po': 2}
BsmExposure {'No': 953, 'Av': 221, 'Gd': 134, 'Mn': 114}
BsmFinType1 {'Unf': 430, 'GLQ': 418, 'ALQ': 220, 'BLQ': 148, 'Rec': 133, 'LwQ': 74}
BsmFinType2 {'Unf': 1256, 'Rec': 54, 'LwQ': 46, 'BLQ': 33, 'ALQ': 19, 'GLQ': 14}
Heating {'GasA': 1428, 'GasW': 18, 'Grav': 7, 'Wall': 4, 'OTHW': 2, 'Floor': 1}
HeatingQC {'Ex': 741, 'TA': 428, 'Gd': 241, 'Fa': 49, 'Po': 1}
CentralAir {'Y': 1365, 'N': 95}
Electrical {'SBrkr': 1334, 'FuseA': 94, 'FuseF': 27, 'FuseP': 3, 'Mix': 1}
KitchenQual {'TA': 735, 'Gd': 586, 'Ex': 100, 'Fa': 39}
```

(2) 空缺值填充

对于空缺值的填充根据连续型和离散型采取不同的填充策略。

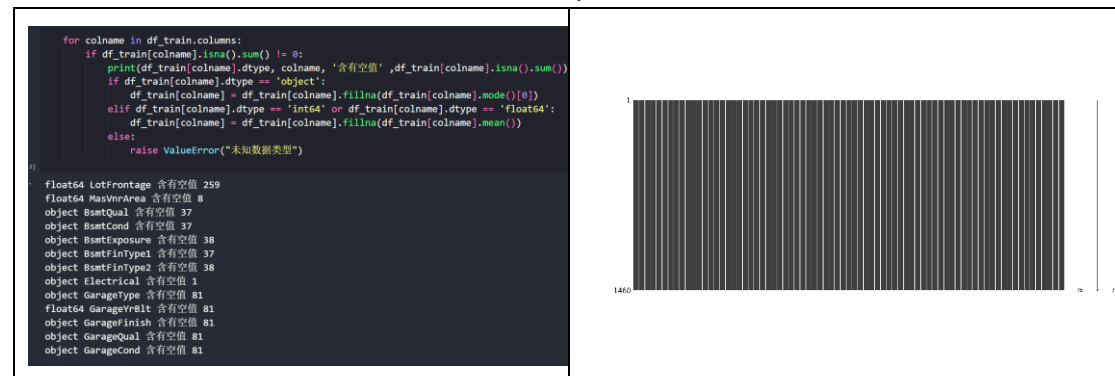
➤ 连续型:

采取列均值进行填充。

➤ 离散型：

采取该列中的出现次数最多的字段值进行填充（null 除外）

以下左图是填充实现代码以及含有空缺值的列，右图为填充之后的结果。



(3) 划分训练集中训练数据与实际房价结果

删除df_train中Id字段，划分属性集合与最终结果SalePrice

- 也对连续型属性和离散型属性的字典进行修改

```
X_train, y_train = df_train.drop(columns=['SalePrice', 'Id'], df_train['SalePrice'])
```

```
print(type_col['continue'])
type_col['continue'].remove('SalePrice')
type_col['continue'].remove('Id')
print(type_col['continue'])
```

['Id', 'MSSubClass', 'LotArea', 'OverallQual', 'OverallCond', 'YearBuilt', 'YearRemodAdd', 'BsmtFinSF1', 'BsmtFinSF2', 'BsmtUnfSF', 'TotalBsmtSF', 'SalePrice', 'Id']

(4) 对连续型属性进行数据标准化

对于连续型属性，采取 zscore 方法进行标准化，消除不同量纲大小对于数值的影响。

```
from sklearn.preprocessing import StandardScaler
```

```
scaler = StandardScaler()
X_train[type_col['continue']] = scaler.fit_transform(X_train[type_col['continue']])
display(X_train[type_col['continue']].head())
```

	MSSubClass	LotArea	OverallQual	OverallCond	YearBuilt	YearRemodAdd	BsmtFinSF1	BsmtFinSF2	BsmtUnfSF	TotalBsmtSF
0	0.073375	-0.207142	0.651479	-0.517200	1.050994	0.878668	0.575425	-0.288653	-0.944591	-0.459303
1	-0.872563	-0.091886	-0.071836	2.179628	0.156734	-0.429577	1.171992	-0.288653	-0.641228	0.466465
2	0.073375	0.073480	0.651479	-0.517200	0.984752	0.830215	0.092907	-0.288653	-0.301643	-0.313369
3	0.309859	-0.096897	0.651479	-0.517200	-1.863632	-0.720298	-0.499274	-0.288653	-0.061670	-0.687324
4	0.073375	0.375148	1.374795	-0.517200	0.951632	0.733308	0.463568	-0.288653	-0.174865	0.199680

5 rows x 36 columns

(5) 对于离散型属性的特征工程 onehot 编码

在此若采取字典映射的方式，即将一个属性中的不同值映射为不同的数字值，可能会对回归模型产生影响。因为该方式隐藏着同一属性下不同的值在“数值”方面有着固定的大小先后顺序。因此，对于离散型属性采取 onehot 编码方式。

```
from sklearn.preprocessing import OneHotEncoder

encoder = OneHotEncoder(sparse_output=False, handle_unknown='ignore')
encoded_discrete_arr = encoder.fit_transform(X_train[type_col['discrete']])
encoded_discrete_df = pd.DataFrame(encoded_discrete_arr, columns=encoder.get_feature_names_out(type_col['discrete']))
X_train = pd.concat([X_train.drop(columns=type_col['discrete']), encoded_discrete_df], axis=1)
```

3. 评估函数

采用均方根误差 (RMSE) 为唯一评价指标。同时用 $\log()$ 运算将价位高低不同的预测误差对最终结果的影响统一起来。

$$RMSE = \sqrt{\frac{\sum_{i=1}^N (\log \hat{y}_i - \log y_i)^2}{N}}$$

评估函数

- 即预测值和真实值之间的误差计算方法
- 参照网站上的要求

```
def cal_acc(real:list, pred:list)->float:
    if len(real) != len(pred):
        raise ValueError("输入的长度不一致")
    N = len(real)
    summary = sum([(math.log(p)-math.log(r))**2 for p, r in zip(pred, real)])
    rmse = (summary/N)**0.5
    return rmse
```

4. 训练模型

在回归算法的选择中，选取了随机森林算法进行回归预测。随机森林由多个决策树组成，在回归任务中通过对所有树的预测结果进行平均，通过组合多个决策树的结果来减少过拟合，提升预测性能。

- 该随机森林中一共有 100 个决策树；
- 每个决策树的构建不限制决策树的深度，即会一直进行节点分裂直至节点是纯为止；
- 衡量节点分裂的质量采取的标准是均方误差 squared_error。

模型在训练集上的精度为 0.05876。

```
from sklearn.ensemble import RandomForestRegressor
```

```
model = RandomForestRegressor(n_estimators=100, random_state=13)
```

```
model.fit(X_train, y_train)
```

```
RandomForestRegressor  ⓘ ⓘ
RandomForestRegressor(random_state=13)
```

```
y_train_pred = model.predict(X_train)
print("在训练集上的误差", cal_acc(y_train.to_list(), y_train_pred))
```

在训练集上的误差 0.05876237258950698

5. 利用模型对测试集进行预测

(1) 首先按照处理训练集的方法对测试集进行数据预处理

- 删除空缺值过多的列
- 对连续型属性和离散型属性的空缺值填补
- 对连续型属性的数据标准化
- 对离散型属性的 onehot 编码

(2) 调用 `model.predict()` 函数对测试集进行预测并将预测结果保存至文件

```
[35] y_test_pred = model.predict(X_test)
✓ 0.0s

[36] y_test_id = df_test['Id'].to_list()
if len(y_test_id) != len(y_test_pred):
    raise ValueError('')
✓ 0.0s

[37] res = pd.DataFrame(data={
    'Id': y_test_id,
    'SalePrice': y_test_pred
})
res.to_csv('res.csv', index=False)
res.head()
✓ 0.0s 在 Data Wrangler 中打开 "res"
...
  Id  SalePrice
0 1461 128350.00
1 1462 156391.32
2 1463 181188.60
3 1464 180211.35
4 1465 196018.89
```

3. 实验结果和分析

预测结果文件保存在 `res.csv` 文件当中。其中测试集预测房价与测试集实际房价的误差为 0.14593。测试集预测房价与实际房价之间误差较小，模型较优。下为 kaggle 评测截图。

Submissions

AllSuccessfulErrors

Recent ▾

Submission and Description	Public Score ⓘ
<div><div>✓</div><div>res.csv Complete · 2m ago</div></div>	0.14593

4.总结

在本次实验中，通过建立随机森林模型，对爱荷华州艾姆斯每套住宅的最终价格进行了预测。训练数据的质量决定了模型的上限，模型的选择和参数的调优是不断的接近这个上限的过程。与实验课中构建决策树实验提供的数据集不同，该实验提供的数据集，无论是训练集还是测试集都有许多空缺值。因此在特征工程方面，对数据集进行了规范处理。对于连续型属性，采取均值填充空缺值，进行 `zscore` 数据标准化；对于离散型属性，采取众数填充空缺值，`onehot` 编码的方式进行规范。

通过本次实验，自己的代码能力有了很大提高，了解了 `scikit-learn` 的使用，熟悉了数据预处理、参数选择、模型训练、模型评估等机器学习主要流程，对随机森林的应用有了更深刻的理解。