

Universidade de São Paulo
Instituto de Física de São Carlos

Competições de Ciência de Dados

Éverton Luís Mendes da Silva (10728171)

Conteúdo

1	Questão 1	2
2	Questão 2	4
3	Questão 3	5
4	Questão 4	10

1 Questão 1

Nesse projeto, a minha proposta não é criar apenas três variáveis explicativas, mas sim um conjunto de novas variáveis que comparam a disponibilidade e preços estabelecidos pela própria área (explicado na seção abaixo).

Primeiramente, para analisar a qualidade das variáveis propostas, vamos pensar quais são os fatores mais importantes que influenciam o valor de uma casa. A compra de uma casa é um momento importante na vida das pessoas, isto é, elas gastam um bom tempo analisando qual é a melhor residência para os seus próprios propósitos. Tendo isso em vista, um fator crítico é a comparação das características como número de quartos, número de banheiros, área disponível, dentre outros, com casas da mesma vizinhança ou de bairros vizinhos.

Por isso, foi criado uma função para transformar as colunas numéricas em novas colunas, levando em consideração outras residências do mesmo bairro. Ou seja, são variáveis que demonstram o daquela residência com as outras da mesma vizinhança.

```
1 def NeighborComparative(df):
2
3     numeric_features = df.drop(
4         columns='Id')._get_numeric_data().columns.tolist()
5
6     for feature in numeric_features:
7
8         neighbors_dict = {}
9         for neighbor in df['Neighborhood'].value_counts().
index:
10
11             neighbors = df[df['Neighborhood'] == neighbor]
12             neighbors_dict[neighbor] = sum(neighbors[feature
13 ])
14
15         for index in range(df.shape[0]):
16             df.loc[index:index+1, str(feature)+'Comp'] = df[
index:index+1][feature].values[0]/(
17                 neighbors_dict[df[index:index+1]['
Neighborhood'].values[0]])
18
19     return df
```

Listing 1: Compare neighbors

```
1 from Neighbor import NeighborComparative
2
3
4 def feature_eng(df, y_activation=False):
```

```

5     '''feature engineering of House Prices - Advanced
Regression Techniques kaggle competition
6     Args:
7         df, kaggle dataframe
8         y_activation, create y if y_activation is True else
return y=None
9     Returns:
10        df kaggle dataframe
11        X(DataFrame) ,features of analysis
12        y(DataFrame), target of analysis
13    '''
14
15    df['AgeSold'] = df['YrSold'] - df['YearBuilt']
16    df['AgeRemodSold'] = df['YrSold'] - df['YearRemodAdd']
17    df['GarageAgeBlt'] = df['YrSold'] - df['GarageYrBlt']
18
19    if y_activation == True:
20        target_col = 'SalePrice'
21        future_cols = [
22            'MoSold',
23            'YrSold',
24            'SaleType',
25            'SaleCondition',
26            target_col
27        ]
28    else:
29        future_cols = [
30            'MoSold',
31            'YrSold',
32            'SaleType',
33            'SaleCondition',
34        ]
35
36    drop_fe_cols = [
37        'YearBuilt',
38        'YearRemodAdd',
39        'GarageYrBlt'
40    ]
41
42    X = df.drop(future_cols, axis=1)
43    X.drop(drop_fe_cols, axis=1, inplace=True)
44
45    if y_activation == True:
46        y = df['SalePrice']
47    else:
48        y = None
49
50    X = NeighborComparative(X)
51

```

```
52     return df, X, y
```

Listing 2: feature engineering

2 Questão 2

O pipeline desenvolvido em aula foi aprimorado através do uso de classe, isto é, foi criada uma classe com o nome 'RegSwitcher' que pode aceitar qualquer tipo de modelo, considerando que ele possui os métodos encontrados no código abaixo.

```
1 from sklearn.neighbors import KNeighborsRegressor
2 from sklearn.base import BaseEstimator
3
4 class RegSwitcher(BaseEstimator):
5
6     def __init__(
7         self,
8         estimator = KNeighborsRegressor(),
9     ):
10         """
11         A Custom BaseEstimator that can switch between
12         classifiers.
13         :param estimator: sklearn object - The classifier
14         """
15
16         self.estimator = estimator
17
18     def fit(self, X, y=None, **kwargs):
19         self.estimator.fit(X, y)
20         return self
21
22
23     def predict(self, X, y=None):
24         return self.estimator.predict(X)
25
26
27     def predict_proba(self, X):
28         return self.estimator.predict_proba(X)
29
30
31     def score(self, X, y):
32         return self.estimator.score(X, y)
```

Listing 3: RegSwitcher Class

3 Questão 3

A predição do preço das casas foi realizado através de diversos modelos e parâmetros, que podem ser encontrados abaixo.

```
1 from xgboost import XGBRegressor
2 from sklearn.tree import DecisionTreeRegressor
3 from sklearn.ensemble import RandomForestRegressor,
  GradientBoostingRegressor
4 from sklearn.preprocessing import MinMaxScaler,
  StandardScaler
5 import numpy as np
6 import pandas as pd
7 from sklearn.metrics import mean_squared_error,
  mean_absolute_percentage_error
8 from sklearn.model_selection import train_test_split,
  GridSearchCV
9 from sklearn.neighbors import KNeighborsRegressor
10 from sklearn.preprocessing import FunctionTransformer,
  StandardScaler, MinMaxScaler, OneHotEncoder
11 from sklearn.impute import SimpleImputer
12 from sklearn.pipeline import Pipeline
13 from sklearn.compose import ColumnTransformer,
  make_column_selector
14 from sklearn.base import BaseEstimator
15 from RegSwitcher import RegSwitcher
16 import joblib as jb
17 from feature_eng import feature_eng
18 from sklearn import svm
19 from sklearn.linear_model import SGDRegressor
20 from sklearn.naive_bayes import GaussianNB
21 #from skopt import skopt
22
23
24 house_df=pd.read_csv('train.csv')
25 #print(house_df)
26
27
28 house_df, X, y=feature_eng(house_df, y_activation=True)
29
30 X_train, X_test, y_train, y_test = train_test_split(
31     X, y,
32     train_size=0.65,
33     random_state=42
34 )
35
36 len(y_train), len(y_test)
```

```

39 pipeline_num = Pipeline(
40     steps=[
41         ('imputer', SimpleImputer()),
42         ('scaler', MinMaxScaler())
43     ]
44 )
45
46 ct = ColumnTransformer([
47     (
48         'num_transf',
49         pipeline_num,
50         make_column_selector(dtype_exclude=object)
51     ),
52     (
53         'categ_transf',
54         OneHotEncoder(sparse=False, handle_unknown='ignore'),
55         make_column_selector(dtype_include=object)
56     )
57 ])
58
59 pipeline = Pipeline(
60     steps=[
61         ('ct', ct),
62         ('reg', RegSwitcher())
63     ]
64 )
65
66 parameters = [
67     {
68
69         'ct__num_transf__imputer__strategy': ['mean', 'median
70         '],
71         'ct__num_transf__scaler'           : [MinMaxScaler(),
72         StandardScaler()],
73
74         'reg__estimator': [XGBRegressor()],
75         'reg__estimator__eta': [index/10.00 for index in range
76         (1, 3)],
77         'reg__estimator__gamma': [2**index for index in range
78         (1, 4)],
79         'reg__estimator__min_child_weight': [2**index for index in
80         range(1, 4)],
81         'reg__estimator__objective': ['reg:squarederror', 'reg:
82         squaredlogerror', 'reg:pseudohubererror'],
83         'reg__estimator__rate_drop': [index/10.00 for index
84         in range(1, 3)],
85         'reg__estimator__skip_drop': [index/10.00 for index
86         in range(1, 3)],

```

```

80     },
81     {
82
83         'ct__num_transf__imputer__strategy': ['mean', 'median
84     '],
85         'ct__num_transf__scaler'                : [MinMaxScaler(),
86         StandardScaler()],
87
88         'reg__estimator': [svm.SVR()],
89         'reg__estimator__kernel': ['linear', 'poly', 'rbf', '
90         sigmoid'],
91     'reg__estimator__epsilon': [1.0/(10.0**index) for index in
92     range(1, 4)],
93
94     },
95     {
96
97         'ct__num_transf__imputer__strategy': ['mean', 'median
98     '],
99         'ct__num_transf__scaler'                : [MinMaxScaler(),
100        StandardScaler()],
101
102         'reg__estimator': [SGDRegressor()],
103         'reg__estimator__loss': ['huber', 'epsilon_insensitive', '
104        squared_epsilon_insensitive'],
105         'reg__estimator__penalty': ['l2', 'l1', 'elasticnet'],
106         'reg__estimator__alpha': [1.0/(10.0**index) for index in
107        range(3, 6)],
108         'reg__estimator__eta0': [1.0/(10.0**index) for index in
109        range(1, 3)],
110
111     },
112     {
113
114         'ct__num_transf__imputer__strategy': ['mean', 'median
115     '],
116         'ct__num_transf__scaler'                : [MinMaxScaler(),
117        StandardScaler()],
118
119         'reg__estimator': [GaussianNB()],
120
121     }
122 ]

```



```

118 gscv =GridSearchCV(pipeline, parameters, verbose=2)
119 # cv=5, n_jobs=5, return_train_score=False,
120
121 gscv.fit(X_train, y_train)
122
123 pred=gscv.best_estimator_.predict(X_test)
124 mse=mean_squared_error(y_test, pred)
125 mape=mean_absolute_percentage_error(y_test, pred)
126
127 print('Best model:\n', gscv.best_params_)
128 print('Best mse: {}\n Best mape: {}\n'.format(mse, mape))
129
130 #joblib.dump(grid.best_estimator_, 'best_tfidf.pkl', compress
    = 1) # this unfortunately includes the LogReg
131
132 jb.dump(gscv.best_params_, 'best_tfidf.pkl', compress = 1) #
133
134
135
136 '''
137 {
138     'ct__num_transf__imputer__strategy': ['mean', 'median
139     '],
140     'ct__num_transf__scaler'           : [MinMaxScaler(),
141     StandardScaler()],
142
143     'reg__estimator': [DecisionTreeRegressor()],
144     'reg__estimator__criterion' : ["mse", "friedman_mse",
145     "mae", "poisson"],
146     'reg__estimator__splitter' : ["best", "random"],
147     'reg__estimator__min_samples_split' : [2**index for
148     index in range(1, 3)],
149     'reg__estimator__max_features' : ["auto", "sqrt", "
150     log2"],
151
152     'reg__estimator__max_depth' : [2**index for index in
153     range(1, 3)],
154
155     },
156     {
157         'ct__num_transf__imputer__strategy': ['mean', 'median
158         '],
159         'ct__num_transf__scaler'           : [MinMaxScaler(),
160         StandardScaler()],
161
162         'reg__estimator': [RandomForestRegressor()],

```

```

157         'reg__estimator__n_estimators' : [2**index for index
158         in range(1, 3)],
159         'reg__estimator__criterion' : ["mse", "mae", "mape"],
160         'reg__estimator__min_samples_split' : [2**index for
161         index in range(1, 3)],
162         'reg__estimator__max_features' : ["auto", "sqrt", "
163         log2"],
164         'reg__estimator__max_depth' : [2**index for index in
165         range(1, 3)]
166     }
167 '''

```

Listing 4: Choose Best Model

Depois de rodar vários modelos, foi possível encontrar o melhor modelo abaixo e sua respectiva pontuação no kaggle.

```

Best model:
{'ct_num_transf_imputer_strategy': 'mean', 'ct_num_transf_scaler': StandardScaler(), 'reg_estimator': XGBRegressor(base_score=None, booster=None,
colsample_bylevel=None,
colsample_bynode=None, colsample_bytree=None, eta=0.1, gamma=2,
gpu_id=None, importance_type='gain', interaction_constraints=None,
learning_rate=None, max_delta_step=None, max_depth=None,
min_child_weight=4, missing=nan, monotone_constraints=None,
n_estimators=100, n_jobs=None, num_parallel_tree=None,
random_state=None, rate_drop=0.1, reg_alpha=None, reg_lambda=None,
scale_pos_weight=None, skip_drop=0.1, subsample=None,
tree_method=None, validate_parameters=None, verbosity=None), 'reg_estimator_eta': 0.1, 'reg_estimator_gamma': 2, 'reg_estimator_mi
n_child_weight': 4, 'reg_estimator_objective': 'reg:squarederror', 'reg_estimator_rate_drop': 0.1, 'reg_estimator_skip_drop': 0.1}

```

Figure 1: O melhor modelo encontrado


1619	everttonmendes73		0.13708	7	1s
------	------------------	--	---------	---	----

Figure 2: Pontuação no kaggle

4 Questão 4

O RMSE(root mean square error) é métrica que diz o qual longe as predições estão das medidas reais, isto é, nos mostra qual a concentração dos pontos com a linha de melhor fit. Contudo, essa métrica falha por não levar em consideração o percentual dos erros, ou seja, ela leva em consideração a magnitude dos erros. Para exemplificar, outliers tendem a ter mais relevância nessa métrica, levando os outros dados a uma insignificância.

Por outro lado, uma métrica boa para lidar com esses problemas é o MAPE(mean absolute percentage erro). Essa métrica consegue normalizar os erros e não deixar que pontos destoantes inviabilizem a análise.