Aprendizado de Máquina

Alexandre Furtado Violante, RA: 632880

House Prices - Advanced Regression Techniques

- **Objetivo:** estimar o preço de casas (SalePrice) em Ames, Iowa, usando dados históricos de vendas e atributos detalhados das casas.

- **Kaggle fornece 2 planilhas:**
  - **train.csv**, com 1460 casas e 79 colunas de características e uma SalePrice.
  - **test.csv**, com 1459 casas e as mesmas 79 colunas, sem a coluna SalePrice

Desafios

- O conjunto de dados possui variáveis numéricas e categóricas, muitas com valores ausentes.

- O modelo deve lidar com dados heterogêneos, diferentes escalas, outliers e relações complexas entre atributos.

Modelo

- **RandomForest Regressor:** método robusto para regressão em dados tabulares, lidando bem com muitas variáveis e relações não lineares.

- Funcionamento:
  - Algoritmo de ensemble que combina centenas de árvores de decisão.
  - Cada árvore recebe uma amostra aleatória dos dados e um subconjunto aleatório de features.
  - A predição final é a média das previsões de todas as árvores.

Bibliotecas

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.ensemble import RandomForestRegressor
```

- ***Pandas:*** manipulação de dados
- ***Numpy:*** operações numéricas
- ***Matplotlib e Seaborn:*** visualização gráfica
- ***RandomForestRegressor:*** modelo utilizado

Carregamento dos dados

```
train = pd.read_csv("train.csv")
test  = pd.read_csv("test.csv")
print(f"Train: {train.shape} | Test: {test.shape}")
```

```
Train: (1460, 81) | Test: (1459, 80)
```

- Os arquivos train.csv e test.csv são carregados.

- A inspeção com shape e info() mostra quantidade de dados, tipos de variáveis e presença de valores ausentes.

```
train.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1460 entries, 0 to 1459
Data columns (total 81 columns):
 #   Column         Non-Null Count  Dtype  
---  ------         --------------  -----  
 0   Id             1460 non-null   int64  
 1   MSSubClass     1460 non-null   int64  
 2   MSZoning       1460 non-null   object 
 3   LotFrontage    1201 non-null   float64
 4   LotArea        1460 non-null   int64  
 5   Street         1460 non-null   object 
 6   Alley          91 non-null     object 
 7   LotShape       1460 non-null   object 
 8   LandContour    1460 non-null   object 
 9   Utilities      1460 non-null   object 
 10  LotConfig      1460 non-null   object 
 11  LandSlope      1460 non-null   object 
 12  Neighborhood   1460 non-null   object 
 13  Condition1     1460 non-null   object 
 14  Condition2     1460 non-null   object 
 15  BldgType       1460 non-null   object
```

Valores ausentes

```
for df in [train, test]:
    df.fillna(df.select_dtypes(include=np.number).median(), inplace=True)
    df.fillna('Missing', inplace=True)

print("Missing values tratados → nulos:", train.isnull().sum().sum())
```

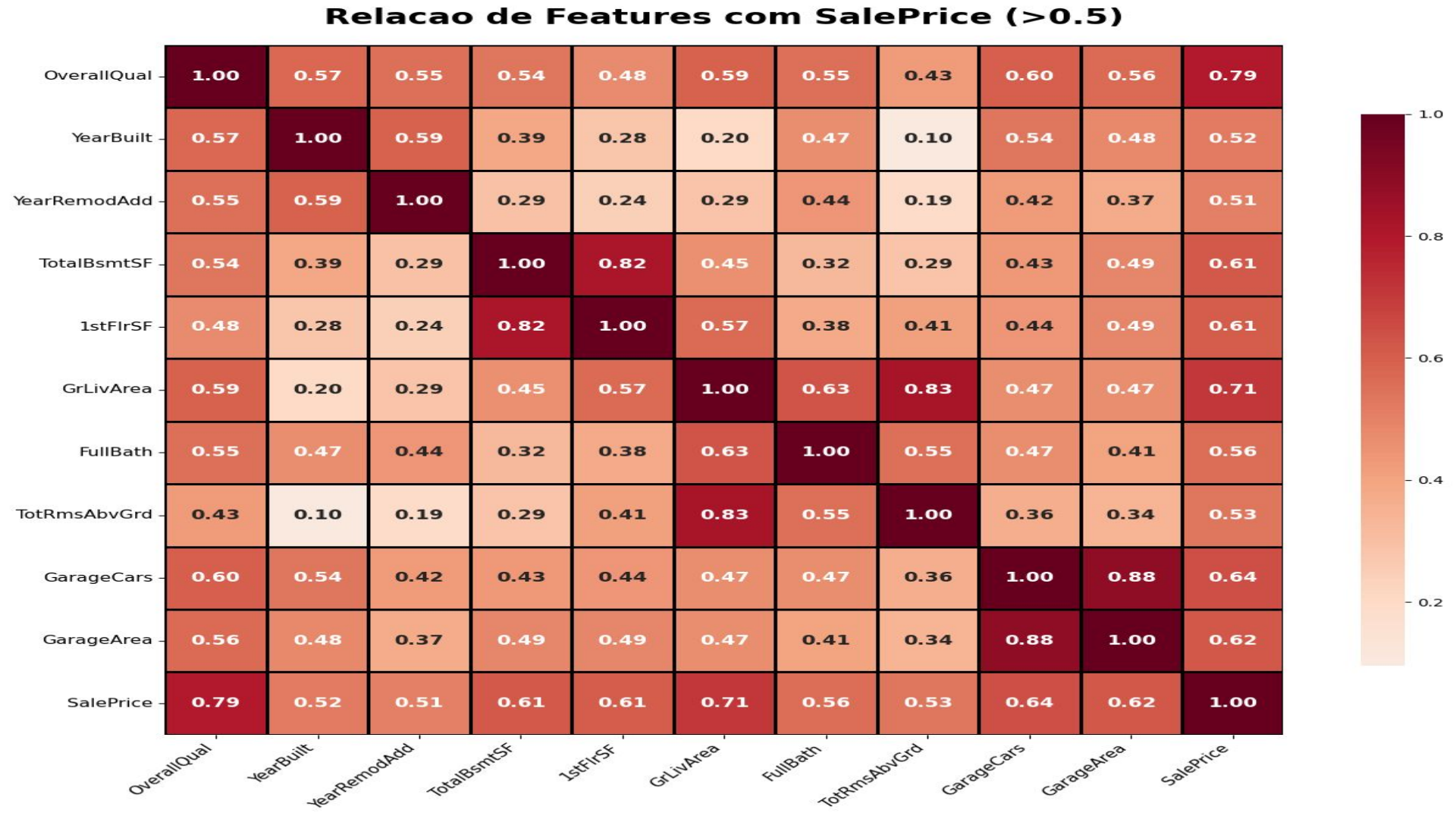
```
Missing values tratados → nulos: 0
```

- Numéricos → preenchidos com a mediana (menos sensível a outliers)
- Categóricos → preenchidos com “Missing”

- Essa etapa evita falhas no modelo e permite que ele interprete ausência como informação.

Análise de Correlação

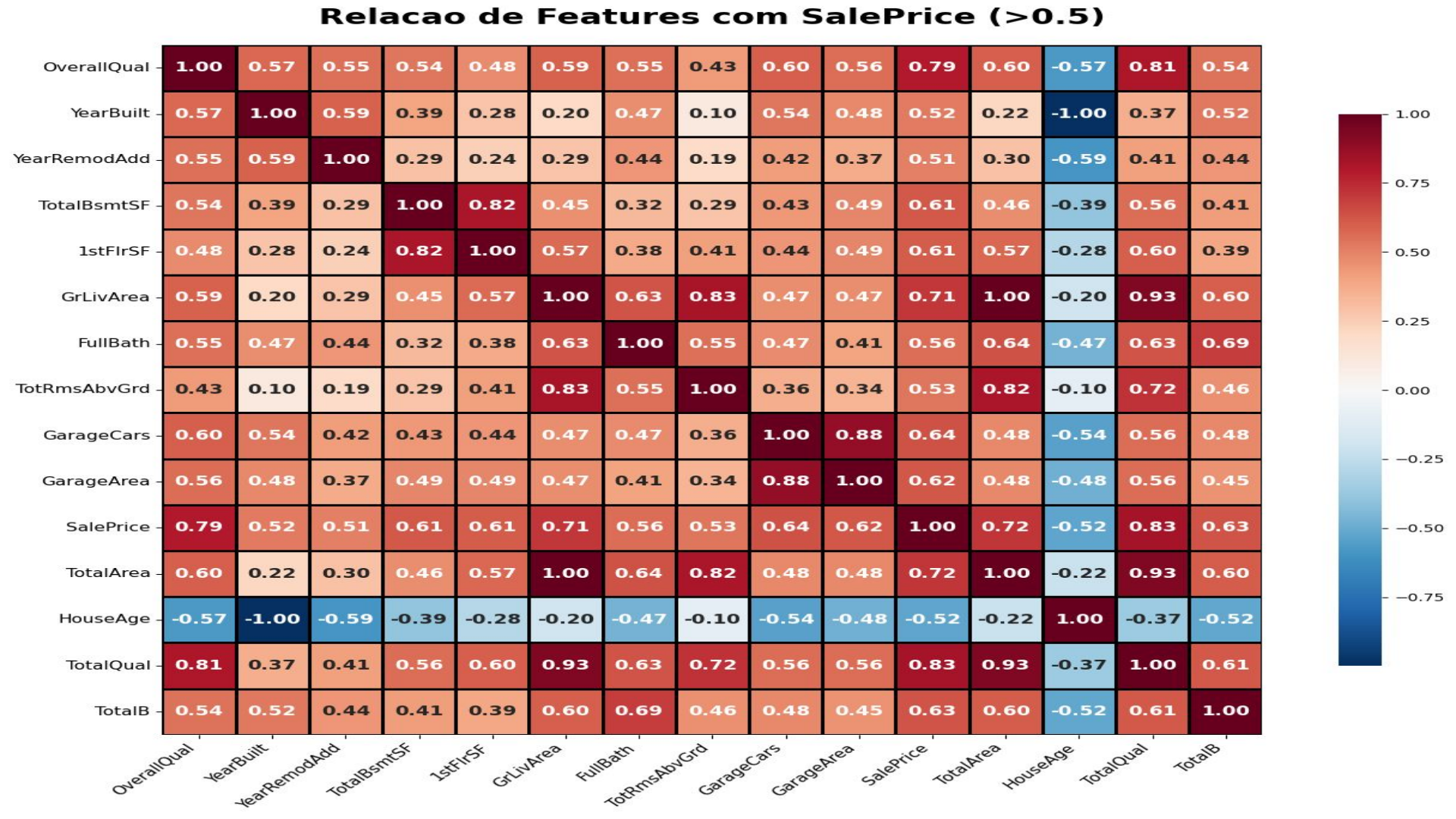
```
numeric_cols = train.select_dtypes(include='number')
corr = numeric_cols.corr()
top_corr = corr.index[abs(corr["SalePrice"]) > 0.5].tolist()

plt.figure(figsize=(14, 11))
sns.heatmap(
    train[top_corr].corr(),
    annot=True,
    fmt=".2f",
    cmap="RdBu_r",
    center=0,
    linewidths=1,
    linecolor='black',
    cbar_kws={"shrink": .8},
    square=True,
    annot_kws={"size": 12, "weight": "bold"}
)
plt.title('Relacao de Features com SalePrice (>0.5)', fontsize=20, fontweight='bold', pad=20)
plt.xticks(rotation=45, ha='right', fontsize=11)
plt.yticks(fontsize=11)
plt.tight_layout()
plt.show()
```


Relacao de Features com SalePrice (>0.5)
OverallQual 1.00 0.57 0.55 0.54 0.48 0.59 0.55 0.43 0.60 0.56 0.79
YearBuilt 0.57 1.00 0.59 0.39 0.28 0.20 0.47 0.10 0.54 0.48 0.52
YearRemodAdd 0.55 0.59 1.00 0.29 0.24 0.29 0.44 0.19 0.42 0.37 0.51
TotalBsmtSF 0.54 0.39 0.29 1.00 0.82 0.45 0.32 0.29 0.43 0.49 0.61
1stFlrSF 0.48 0.28 0.24 0.82 1.00 0.57 0.38 0.41 0.44 0.49 0.61
GrLivArea 0.59 0.20 0.29 0.45 0.57 1.00 0.63 0.83 0.47 0.47 0.71
FullBath 0.55 0.47 0.44 0.32 0.38 0.63 1.00 0.55 0.47 0.41 0.56
TotRmsAbvGrd 0.43 0.10 0.19 0.29 0.41 0.83 0.55 1.00 0.36 0.34 0.53
GarageCars 0.60 0.54 0.42 0.43 0.44 0.47 0.47 0.36 1.00 0.88 0.64
GarageArea 0.56 0.48 0.37 0.49 0.49 0.47 0.41 0.34 0.88 1.00 0.62
SalePrice 0.79 0.52 0.51 0.61 0.61 0.71 0.56 0.53 0.64 0.62 1.00
OverallQual
YearBuilt
YearRemodAdd
TotalBsmtSF
1stFlrSF
GrLivArea
FullBath
TotRmsAbvGrd
GarageCars
GarageArea
SalePrice
1.0
0.8
0.6
0.4
0.2

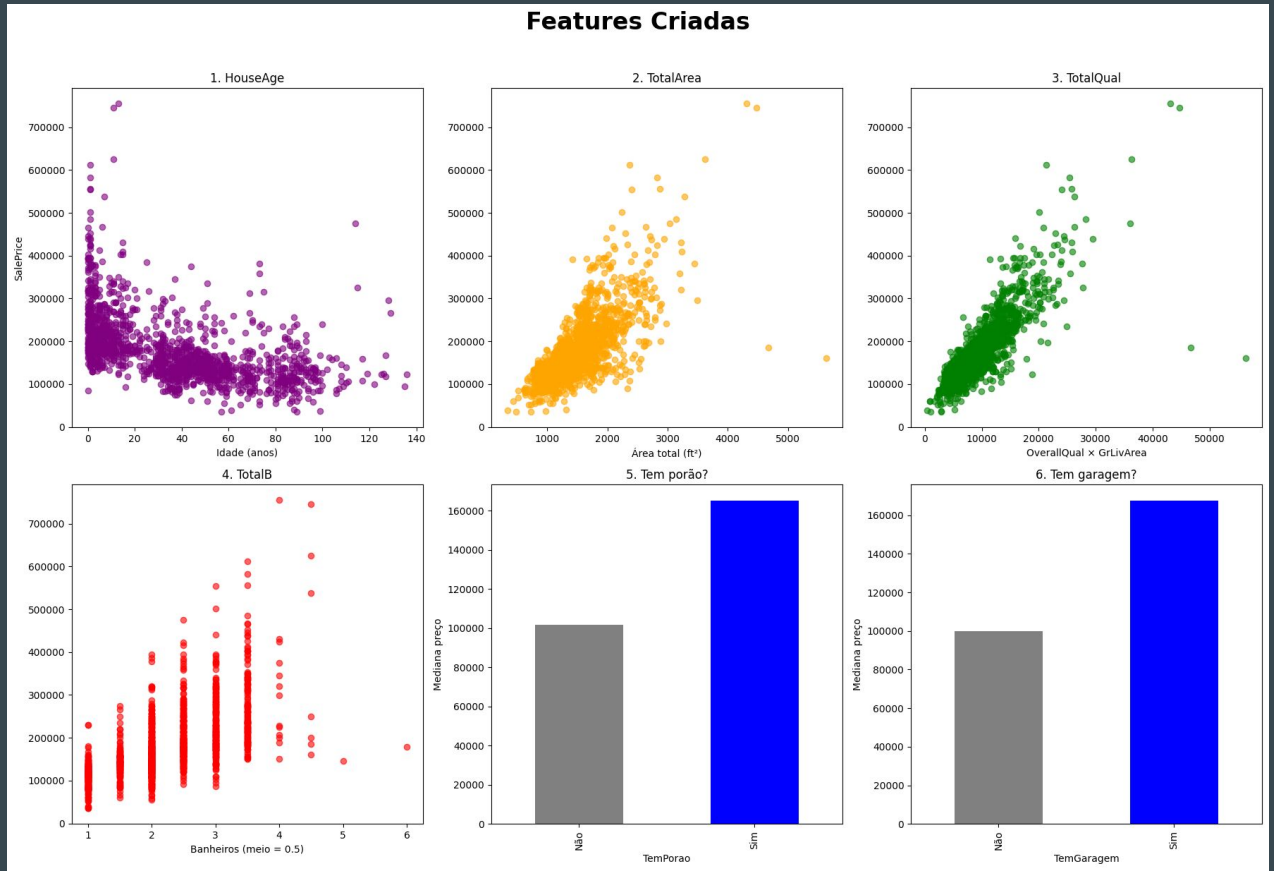
Criação das Features

```
numeric_cols = train.select_dtypes(include='number')
for df in [train, test]:
    df['TotalArea'] = df['1stFlrSF'] + df['2ndFlrSF']
    df['HouseAge']  = df['YrSold'] - df['YearBuilt']
    df['TotalQual'] = df['OverallQual'] * df['GrLivArea']
    df['HasGarage'] = (df['GarageArea'] > 0).astype(int)
    df['TotalB']    = df['FullBath'] + 0.5*df['HalfBath'] + df['BsmtFullBath'] +  + 0.5*df['BsmtHalfBath']
```

- Área total = 1° andar + 2° andar
- Idade da casa = Ano da venda - Ano da construção
- Qualidade total = Qualidade * Área
- Tem garagem?
- Quantidade de banheiros


Relacao de Features com SalePrice (>0.5)
OverallQual 1.00 0.57 0.55 0.54 0.48 0.59 0.55 0.43 0.60 0.56 0.79 0.60 -0.57 0.81 0.54
YearBuilt 0.57 1.00 0.59 0.39 0.28 0.20 0.47 0.10 0.54 0.48 0.52 0.22 -1.00 0.37 0.52
YearRemodAdd 0.55 0.59 1.00 0.29 0.24 0.29 0.44 0.19 0.42 0.37 0.51 0.30 -0.59 0.41 0.44
TotalBsmtSF 0.54 0.39 0.29 1.00 0.82 0.45 0.32 0.29 0.43 0.49 0.61 0.46 -0.39 0.56 0.41
1stFlrSF 0.48 0.28 0.24 0.82 1.00 0.57 0.38 0.41 0.44 0.49 0.61 0.57 -0.28 0.60 0.39
GrLivArea 0.59 0.20 0.29 0.45 0.57 1.00 0.63 0.83 0.47 0.47 0.71 1.00 -0.20 0.93 0.60
FullBath 0.55 0.47 0.44 0.32 0.38 0.63 1.00 0.55 0.47 0.41 0.56 0.64 -0.47 0.63 0.69
TotRmsAbvGrd 0.43 0.10 0.19 0.29 0.41 0.83 0.55 1.00 0.36 0.34 0.53 0.82 -0.10 0.72 0.46
GarageCars 0.60 0.54 0.42 0.43 0.44 0.47 0.47 0.36 1.00 0.88 0.64 0.48 -0.54 0.56 0.48
GarageArea 0.56 0.48 0.37 0.49 0.49 0.47 0.41 0.34 0.88 1.00 0.62 0.48 -0.48 0.56 0.45
SalePrice 0.79 0.52 0.51 0.61 0.61 0.71 0.56 0.53 0.64 0.62 1.00 0.72 -0.52 0.83 0.63
TotalArea 0.60 0.22 0.30 0.46 0.57 1.00 0.64 0.82 0.48 0.48 0.72 1.00 -0.22 0.93 0.60
HouseAge -0.57 -1.00 -0.59 -0.39 -0.28 -0.20 -0.47 -0.10 -0.54 -0.48 -0.52 -0.22 1.00 -0.37 -0.52
TotalQual 0.81 0.37 0.41 0.56 0.60 0.93 0.63 0.72 0.56 0.56 0.83 0.93 -0.37 1.00 0.61
TotalB 0.54 0.52 0.44 0.41 0.39 0.60 0.69 0.46 0.48 0.45 0.63 0.60 -0.52 0.61 1.00
OverallQual
YearBuilt
YearRemodAdd
TotalBsmtSF
1stFlrSF
GrLivArea
FullBath
TotRmsAbvGrd
GarageCars
GarageArea
SalePrice
TotalArea
HouseAge
TotalQual
TotalB
1.00
0.75
0.50
0.25
0.00
−0.25
−0.50
−0.75

Visualização



Remoção das colunas

```
cols_to_drop = [
    '1stFlrSF',
    '2ndFlrSF',
    'YearBuilt',
    'OverallQual',
    'GrLivArea',
    'GarageArea',
    'FullBath',
    'HalfBath',
    'BsmtFullBath',
    'BsmtHalfBath'
]


train = train.drop(columns=cols_to_drop, errors='ignore')
test  = test.drop(columns=cols_to_drop, errors='ignore')
```

- Remoção das colunas utilizadas na criação das novas features.

```
y_train = train['SalePrice']


X_train = train.drop(['SalePrice'], axis=1)
X_test  = test.copy()
```

- Extrai a coluna alvo SalePrice do DataFrame train e armazena em y_train
- Cria X_train removendo a coluna alvo de train
- Faz uma cópia do DataFrame test para usar como X_test

One-hot encoding

```
X_train = pd.get_dummies(X_train, drop_first=True)
X_test  = pd.get_dummies(X_test, drop_first=True)
```

- O modelo não entende texto, só números.
- Todas as colunas categóricas viram colunas binárias (dummies).
- drop_first=True evita multicolinearidade (remove uma das categorias).

Alinhamento

```
X_train, X_test = X_train.align(X_test, join='left', axis=1, fill_value=0)

print(f"Shapes finais → Train: {X_train.shape} | Test: {X_test.shape}")
```

- align() faz os dois datasets terem exatamente as mesmas colunas.

```
Shapes finais → Train: (1460, 258) | Test: (1459, 258)
```

Treino do Modelo Random Forest

```
rf = RandomForestRegressor(n_estimators=1000, random_state=50, n_jobs=-1)
rf.fit(X_train, y_train)
print("Modelo treinado!")
```

- Criação de um Random Forest Regressor:
  - n_estimators=1000 → 1000 árvores (mais estabilidade).
  - random_state=50 → reprodução consistente dos resultados.
  - n_jobs=-1 → usa todos os núcleos da máquina (mais rápido).

- fit() treina o modelo com as features X e os preços y.

Previsão e criação do submission.csv

```
pred = rf.predict(X_test)


submission = pd.DataFrame({'Id': test['Id'], 'SalePrice': pred})
submission.to_csv('submission.csv', index=False)
```

- predict() gera o preço previsto para cada casa do dataset de teste.

- O arquivo final é salvo como submission.csv.

Resultado

```
print(submission.head(15))
```

```
      Id   SalePrice
0   1461  122897.004
1   1462  165727.655
2   1463  178779.357
3   1464  183641.636
4   1465  190005.514
5   1466  179902.925
6   1467  166941.752
7   1468  178063.812
8   1469  184326.727
9   1470  124258.614
10  1471  192269.158
11  1472   92721.317
12  1473   93871.900
13  1474  153041.190
14  1475  110081.410
```

Obrigado!