Exercício Prático 01 _ Projeto de Aprendizado de Máquina de Ponta a Ponta

October 29, 2025

Usaremos o dataset **Ames Housing** para construir um modelo de regressão.

Objetivo: Prever o preço final de venda (`SalePrice`) de cada casa, com base em suas características.

0.1 1. Setup Inicial

Primeiro, vamos importar todas as bibliotecas que precisaremos ao longo do exercício.

[71]:
```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns

from sklearn.datasets import fetch_openml
from sklearn.model_selection import train_test_split, cross_val_score,␣
↪GridSearchCV
from sklearn.preprocessing import StandardScaler, OneHotEncoder
from sklearn.impute import SimpleImputer
from sklearn.pipeline import Pipeline
from sklearn.compose import ColumnTransformer

from sklearn.linear_model import LinearRegression
from sklearn.tree import DecisionTreeRegressor
from sklearn.ensemble import RandomForestRegressor
from sklearn.metrics import mean_squared_error

sns.set(style="whitegrid")
```

0.2 2. Obtenção dos Dados

Vamos carregar o dataset Ames Housing diretamente do OpenML.

[72]:
```
housing = fetch_openml(name="house_prices", as_frame=True)
df = housing.frame
print("Dados carregados com sucesso!")
```

```
Dados carregados com sucesso!
```

0.3 3. Exploração dos Dados

0.3.1 Exercício 1: Explorar o conteúdo do DataFrame

Use os métodos do Pandas para visualizar as primeiras linhas, obter um resumo das colunas (tipos de dados e valores nulos) e gerar estatísticas descritivas.

[73]:
```
print("As 5 primeiras linhas (head):")
print(df.head()) # SEU CÓDIGO AQUI

print("\nResumo dos dados (info):")
df.info() # SEU CÓDIGO AQUI

print("\nEstatísticas Descritivas (describe):")
print(df.describe()) # SEU CÓDIGO AQUI

# Categorias de cada atributo categórico
print("\nCategorias de cada atributo categórico:")
for col in df.select_dtypes(include=['object', 'category']):
    print(f"{col}: {df[col].unique()}")
```

```
As 5 primeiras linhas (head):
   Id  MSSubClass MSZoning  LotFrontage  LotArea Street Alley LotShape  \
0   1          60       RL         65.0     8450   Pave   NaN      Reg
1   2          20       RL         80.0     9600   Pave   NaN      Reg
2   3          60       RL         68.0    11250   Pave   NaN      IR1
3   4          70       RL         60.0     9550   Pave   NaN      IR1
4   5          60       RL         84.0    14260   Pave   NaN      IR1

  LandContour Utilities  … PoolArea PoolQC Fence MiscFeature MiscVal MoSold  \
0         Lvl    AllPub  …        0    NaN   NaN         NaN       0      2
1         Lvl    AllPub  …        0    NaN   NaN         NaN       0      5
2         Lvl    AllPub  …        0    NaN   NaN         NaN       0      9
3         Lvl    AllPub  …        0    NaN   NaN         NaN       0      2
4         Lvl    AllPub  …        0    NaN   NaN         NaN       0     12

  YrSold  SaleType  SaleCondition  SalePrice
0   2008        WD         Normal     208500
1   2007        WD         Normal     181500
2   2008        WD         Normal     223500
3   2006        WD        Abnorml     140000
4   2008        WD         Normal     250000

[5 rows x 81 columns]

Resumo dos dados (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
 16  HouseStyle     1460 non-null   object
 17  OverallQual    1460 non-null   int64
 18  OverallCond    1460 non-null   int64
 19  YearBuilt      1460 non-null   int64
 20  YearRemodAdd   1460 non-null   int64
 21  RoofStyle      1460 non-null   object
 22  RoofMatl       1460 non-null   object
 23  Exterior1st    1460 non-null   object
 24  Exterior2nd    1460 non-null   object
 25  MasVnrType     1452 non-null   object
 26  MasVnrArea     1452 non-null   float64
 27  ExterQual      1460 non-null   object
 28  ExterCond      1460 non-null   object
 29  Foundation     1460 non-null   object
 30  BsmtQual       1423 non-null   object
 31  BsmtCond       1423 non-null   object
 32  BsmtExposure   1422 non-null   object
 33  BsmtFinType1   1423 non-null   object
 34  BsmtFinSF1     1460 non-null   int64
 35  BsmtFinType2   1422 non-null   object
 36  BsmtFinSF2     1460 non-null   int64
 37  BsmtUnfSF      1460 non-null   int64
 38  TotalBsmtSF    1460 non-null   int64
 39  Heating        1460 non-null   object
 40  HeatingQC      1460 non-null   object
 41  CentralAir     1460 non-null   object
 42  Electrical     1459 non-null   object
 43  1stFlrSF       1460 non-null   int64
 44  2ndFlrSF       1460 non-null   int64
 45  LowQualFinSF   1460 non-null   int64
```

```
 46  GrLivArea      1460 non-null   int64
 47  BsmtFullBath   1460 non-null   int64
 48  BsmtHalfBath   1460 non-null   int64
 49  FullBath       1460 non-null   int64
 50  HalfBath       1460 non-null   int64
 51  BedroomAbvGr   1460 non-null   int64
 52  KitchenAbvGr   1460 non-null   int64
 53  KitchenQual    1460 non-null   object
 54  TotRmsAbvGrd   1460 non-null   int64
 55  Functional     1460 non-null   object
 56  Fireplaces     1460 non-null   int64
 57  FireplaceQu    770 non-null    object
 58  GarageType     1379 non-null   object
 59  GarageYrBlt    1379 non-null   float64
 60  GarageFinish   1379 non-null   object
 61  GarageCars     1460 non-null   int64
 62  GarageArea     1460 non-null   int64
 63  GarageQual     1379 non-null   object
 64  GarageCond     1379 non-null   object
 65  PavedDrive     1460 non-null   object
 66  WoodDeckSF     1460 non-null   int64
 67  OpenPorchSF    1460 non-null   int64
 68  EnclosedPorch  1460 non-null   int64
 69  3SsnPorch      1460 non-null   int64
 70  ScreenPorch    1460 non-null   int64
 71  PoolArea       1460 non-null   int64
 72  PoolQC         7 non-null      object
 73  Fence          281 non-null    object
 74  MiscFeature    54 non-null     object
 75  MiscVal        1460 non-null   int64
 76  MoSold         1460 non-null   int64
 77  YrSold         1460 non-null   int64
 78  SaleType       1460 non-null   object
 79  SaleCondition  1460 non-null   object
 80  SalePrice      1460 non-null   int64
dtypes: float64(3), int64(35), object(43)
memory usage: 924.0+ KB

Estatísticas Descritivas (describe):
                Id   MSSubClass  LotFrontage        LotArea  OverallQual  \
count  1460.000000  1460.000000  1201.000000    1460.000000  1460.000000
mean    730.500000    56.897260    70.049958   10516.828082     6.099315
std     421.610009    42.300571    24.284752    9981.264932     1.382997
min       1.000000    20.000000    21.000000    1300.000000     1.000000
25%     365.750000    20.000000    59.000000    7553.500000     5.000000
50%     730.500000    50.000000    69.000000    9478.500000     6.000000
75%    1095.250000    70.000000    80.000000   11601.500000     7.000000
max    1460.000000   190.000000   313.000000  215245.000000    10.000000
```

```
       OverallCond    YearBuilt  YearRemodAdd   MasVnrArea   BsmtFinSF1  …  \
count  1460.000000  1460.000000   1460.000000  1452.000000  1460.000000  …
mean      5.575342  1971.267808   1984.865753   103.685262   443.639726  …
std       1.112799    30.202904     20.645407   181.066207   456.098091  …
min       1.000000  1872.000000   1950.000000     0.000000     0.000000  …
25%       5.000000  1954.000000   1967.000000     0.000000     0.000000  …
50%       5.000000  1973.000000   1994.000000     0.000000   383.500000  …
75%       6.000000  2000.000000   2004.000000   166.000000   712.250000  …
max       9.000000  2010.000000   2010.000000  1600.000000  5644.000000  …

        WoodDeckSF  OpenPorchSF  EnclosedPorch    3SsnPorch  ScreenPorch  \
count  1460.000000  1460.000000    1460.000000  1460.000000  1460.000000
mean     94.244521    46.660274      21.954110     3.409589    15.060959
std     125.338794    66.256028      61.119149    29.317331    55.757415
min       0.000000     0.000000       0.000000     0.000000     0.000000
25%       0.000000     0.000000       0.000000     0.000000     0.000000
50%       0.000000    25.000000       0.000000     0.000000     0.000000
75%     168.000000    68.000000       0.000000     0.000000     0.000000
max     857.000000   547.000000     552.000000   508.000000   480.000000

          PoolArea       MiscVal       MoSold       YrSold      SalePrice
count  1460.000000   1460.000000  1460.000000  1460.000000    1460.000000
mean      2.758904     43.489041     6.321918  2007.815753  180921.195890
std      40.177307    496.123024     2.703626     1.328095   79442.502883
min       0.000000      0.000000     1.000000  2006.000000   34900.000000
25%       0.000000      0.000000     5.000000  2007.000000  129975.000000
50%       0.000000      0.000000     6.000000  2008.000000  163000.000000
75%       0.000000      0.000000     8.000000  2009.000000  214000.000000
max     738.000000  15500.000000    12.000000  2010.000000  755000.000000

[8 rows x 38 columns]

Categorias de cada atributo categórico:
MSZoning: ['RL' 'RM' "'C (all)'" 'FV' 'RH']
Street: ['Pave' 'Grvl']
Alley: [nan 'Grvl' 'Pave']
LotShape: ['Reg' 'IR1' 'IR2' 'IR3']
LandContour: ['Lvl' 'Bnk' 'Low' 'HLS']
Utilities: ['AllPub' 'NoSeWa']
LotConfig: ['Inside' 'FR2' 'Corner' 'CulDSac' 'FR3']
LandSlope: ['Gtl' 'Mod' 'Sev']
Neighborhood: ['CollgCr' 'Veenker' 'Crawfor' 'NoRidge' 'Mitchel' 'Somerst'
'NWAmes'
 'OldTown' 'BrkSide' 'Sawyer' 'NridgHt' 'NAmes' 'SawyerW' 'IDOTRR'
 'MeadowV' 'Edwards' 'Timber' 'Gilbert' 'StoneBr' 'ClearCr' 'NPkVill'
 'Blmngtn' 'BrDale' 'SWISU' 'Blueste']
Condition1: ['Norm' 'Feedr' 'PosN' 'Artery' 'RRAe' 'RRNn' 'RRAn' 'PosA' 'RRNe']
```

```
Condition2: ['Norm' 'Artery' 'RRNn' 'Feedr' 'PosN' 'PosA' 'RRAn' 'RRAe']
BldgType: ['1Fam' '2fmCon' 'Duplex' 'TwnhsE' 'Twnhs']
HouseStyle: ['2Story' '1Story' '1.5Fin' '1.5Unf' 'SFoyer' 'SLvl' '2.5Unf'
'2.5Fin']
RoofStyle: ['Gable' 'Hip' 'Gambrel' 'Mansard' 'Flat' 'Shed']
RoofMatl: ['CompShg' 'WdShngl' 'Metal' 'WdShake' 'Membran' 'Tar&Grv' 'Roll'
 'ClyTile']
Exterior1st: ['VinylSd' 'MetalSd' "'Wd Sdng'" 'HdBoard' 'BrkFace' 'WdShing'
'CemntBd'
 'Plywood' 'AsbShng' 'Stucco' 'BrkComm' 'AsphShn' 'Stone' 'ImStucc'
 'CBlock']
Exterior2nd: ['VinylSd' 'MetalSd' "'Wd Shng'" 'HdBoard' 'Plywood' "'Wd Sdng'"
'CmentBd'
 'BrkFace' 'Stucco' 'AsbShng' "'Brk Cmn'" 'ImStucc' 'AsphShn' 'Stone'
 'Other' 'CBlock']
MasVnrType: ['BrkFace' 'None' 'Stone' 'BrkCmn' nan]
ExterQual: ['Gd' 'TA' 'Ex' 'Fa']
ExterCond: ['TA' 'Gd' 'Fa' 'Po' 'Ex']
Foundation: ['PConc' 'CBlock' 'BrkTil' 'Wood' 'Slab' 'Stone']
BsmtQual: ['Gd' 'TA' 'Ex' nan 'Fa']
BsmtCond: ['TA' 'Gd' nan 'Fa' 'Po']
BsmtExposure: ['No' 'Gd' 'Mn' 'Av' nan]
BsmtFinType1: ['GLQ' 'ALQ' 'Unf' 'Rec' 'BLQ' nan 'LwQ']
BsmtFinType2: ['Unf' 'BLQ' nan 'ALQ' 'Rec' 'LwQ' 'GLQ']
Heating: ['GasA' 'GasW' 'Grav' 'Wall' 'OthW' 'Floor']
HeatingQC: ['Ex' 'Gd' 'TA' 'Fa' 'Po']
CentralAir: ['Y' 'N']
Electrical: ['SBrkr' 'FuseF' 'FuseA' 'FuseP' 'Mix' nan]
KitchenQual: ['Gd' 'TA' 'Ex' 'Fa']
Functional: ['Typ' 'Min1' 'Maj1' 'Min2' 'Mod' 'Maj2' 'Sev']
FireplaceQu: [nan 'TA' 'Gd' 'Fa' 'Ex' 'Po']
GarageType: ['Attchd' 'Detchd' 'BuiltIn' 'CarPort' nan 'Basment' '2Types']
GarageFinish: ['RFn' 'Unf' 'Fin' nan]
GarageQual: ['TA' 'Fa' 'Gd' nan 'Ex' 'Po']
GarageCond: ['TA' 'Fa' nan 'Gd' 'Po' 'Ex']
PavedDrive: ['Y' 'N' 'P']
PoolQC: [nan 'Ex' 'Fa' 'Gd']
Fence: [nan 'MnPrv' 'GdWo' 'GdPrv' 'MnWw']
MiscFeature: [nan 'Shed' 'Gar2' 'Othr' 'TenC']
SaleType: ['WD' 'New' 'COD' 'ConLD' 'ConLI' 'CWD' 'ConLw' 'Con' 'Oth']
SaleCondition: ['Normal' 'Abnorml' 'Partial' 'AdjLand' 'Alloca' 'Family']
```

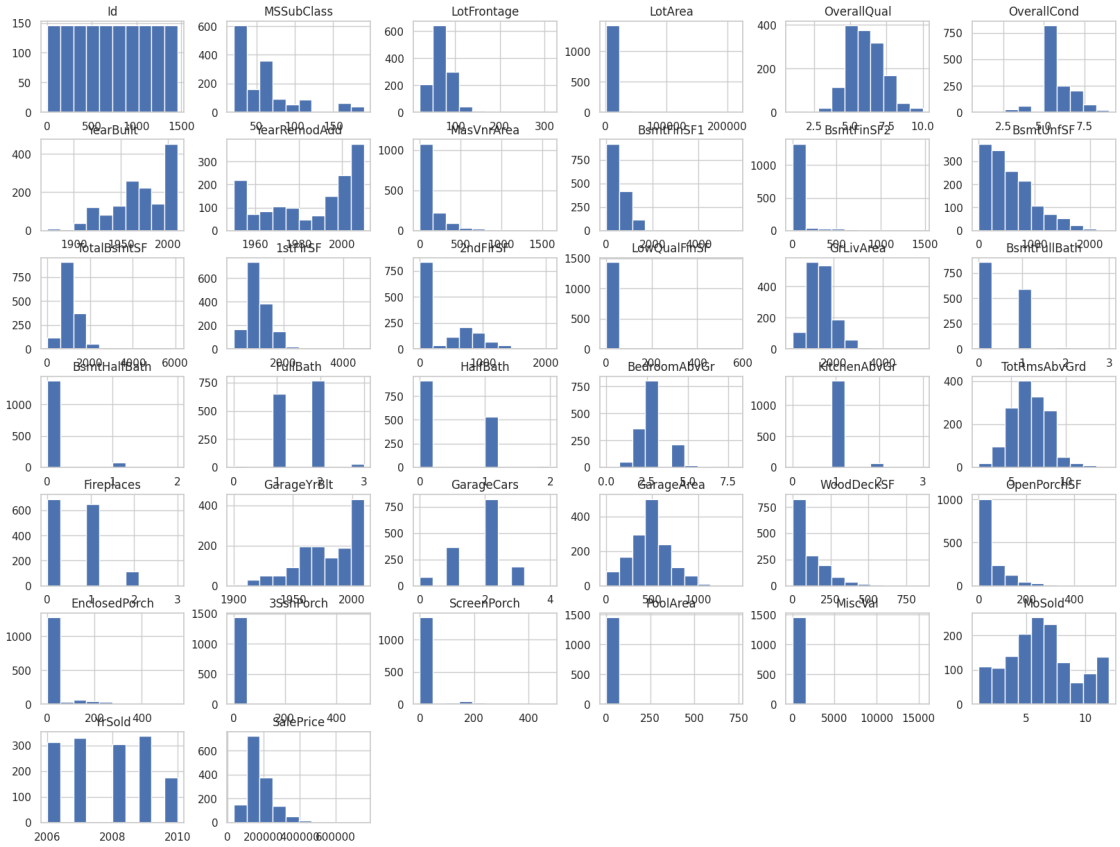
0.3.2 Exercício 2: Gerar os histogramas

Visualize a distribuição dos atributos numéricos. Use o parâmetro `figsize=(20,15)` do método `hist` para melhorar a visualização. É possível identificar fortes assimetrias e valores truncados?

[74]:
```
print("\nHistogramas dos atributos numéricos:")
df.hist(figsize=(20, 15))
```

```
Histogramas dos atributos numéricos:
```

[74]:
```
array([[<Axes: title={'center': 'Id'}>,
        <Axes: title={'center': 'MSSubClass'}>,
        <Axes: title={'center': 'LotFrontage'}>,
        <Axes: title={'center': 'LotArea'}>,
        <Axes: title={'center': 'OverallQual'}>,
        <Axes: title={'center': 'OverallCond'}>],
       [<Axes: title={'center': 'YearBuilt'}>,
        <Axes: title={'center': 'YearRemodAdd'}>,
        <Axes: title={'center': 'MasVnrArea'}>,
        <Axes: title={'center': 'BsmtFinSF1'}>,
        <Axes: title={'center': 'BsmtFinSF2'}>,
        <Axes: title={'center': 'BsmtUnfSF'}>],
       [<Axes: title={'center': 'TotalBsmtSF'}>,
        <Axes: title={'center': '1stFlrSF'}>,
        <Axes: title={'center': '2ndFlrSF'}>,
        <Axes: title={'center': 'LowQualFinSF'}>,
        <Axes: title={'center': 'GrLivArea'}>,
        <Axes: title={'center': 'BsmtFullBath'}>],
       [<Axes: title={'center': 'BsmtHalfBath'}>,
        <Axes: title={'center': 'FullBath'}>,
        <Axes: title={'center': 'HalfBath'}>,
        <Axes: title={'center': 'BedroomAbvGr'}>,
        <Axes: title={'center': 'KitchenAbvGr'}>,
        <Axes: title={'center': 'TotRmsAbvGrd'}>],
       [<Axes: title={'center': 'Fireplaces'}>,
        <Axes: title={'center': 'GarageYrBlt'}>,
        <Axes: title={'center': 'GarageCars'}>,
        <Axes: title={'center': 'GarageArea'}>,
        <Axes: title={'center': 'WoodDeckSF'}>,
        <Axes: title={'center': 'OpenPorchSF'}>],
       [<Axes: title={'center': 'EnclosedPorch'}>,
        <Axes: title={'center': '3SsnPorch'}>,
        <Axes: title={'center': 'ScreenPorch'}>,
        <Axes: title={'center': 'PoolArea'}>,
        <Axes: title={'center': 'MiscVal'}>,
        <Axes: title={'center': 'MoSold'}>],
       [<Axes: title={'center': 'YrSold'}>,
        <Axes: title={'center': 'SalePrice'}>, <Axes: >, <Axes: >,
        <Axes: >, <Axes: >]], dtype=object)
```



0.3.3 Exercício 3: Listar os atributos mais correlacionados com o target

Calcule a correlação de Pearson de todos os atributos em relação ao atributo alvo (`SalePrice`) e exiba os atributos em ordem decrescente de correlação. Lembre de usar o parâmetro `numeric_only=True` no método `corr`, para evitar o erro provocado ao tentar calcular correlação envolvendo atributos categóricos.

[75]:
```
# SEU CÓDIGO AQUI
target_name = 'SalePrice' # Define o nome do atributo alvo

print(f"\nCorrelação de Pearson com o target ({target_name}):")

# 1. Calcular a matriz de correlação (apenas para atributos numéricos)
# Usa o parâmetro numeric_only=True conforme solicitado
correlation_matrix = df.corr(numeric_only=True)

# 2. Selecionar a coluna de correlação com o atributo alvo (SalePrice)
target_correlation = correlation_matrix[target_name]

# 3. Filtrar o próprio alvo (que tem correlação 1.0) e obter a ordem decrescente
# O .sort_values(ascending=False) coloca o maior valor de correlação no topo
```

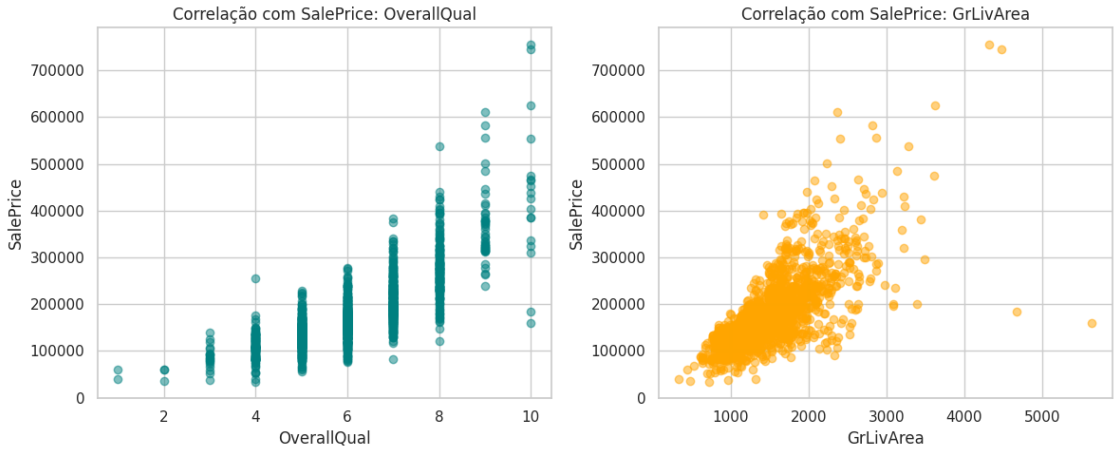
```
sorted_correlations = target_correlation.drop(target_name).
  ↪sort_values(ascending=False)

# 4. Exibir o resultado
print(sorted_correlations)# SEU CÓDIGO AQUI
```

```
Correlação de Pearson com o target (SalePrice):
OverallQual      0.790982
GrLivArea        0.708624
GarageCars       0.640409
GarageArea       0.623431
TotalBsmtSF      0.613581
1stFlrSF         0.605852
FullBath         0.560664
TotRmsAbvGrd     0.533723
YearBuilt        0.522897
YearRemodAdd     0.507101
GarageYrBlt      0.486362
MasVnrArea       0.477493
Fireplaces       0.466929
BsmtFinSF1       0.386420
LotFrontage      0.351799
WoodDeckSF       0.324413
2ndFlrSF         0.319334
OpenPorchSF      0.315856
HalfBath         0.284108
LotArea          0.263843
BsmtFullBath     0.227122
BsmtUnfSF        0.214479
BedroomAbvGr     0.168213
ScreenPorch      0.111447
PoolArea         0.092404
MoSold           0.046432
3SsnPorch        0.044584
BsmtFinSF2      -0.011378
BsmtHalfBath    -0.016844
MiscVal         -0.021190
Id              -0.021917
LowQualFinSF    -0.025606
YrSold          -0.028923
OverallCond     -0.077856
MSSubClass      -0.084284
EnclosedPorch   -0.128578
KitchenAbvGr    -0.135907
Name: SalePrice, dtype: float64
```

Faça o scatter plot do atributo de maior correlação (em módulo) e do atributo `SalePrice`. Faça

também um scatter plot para o segundo atributo de maior correlação (em módulo) e o atributo `SalePrice`. É possível observar uma relação aproximadamente linear?

[76]:
```
import matplotlib.pyplot as plt

# 5. Selecionar os dois atributos com maior correlação (em módulo)
top1_feature = sorted_correlations.index[0]
top2_feature = sorted_correlations.index[1]

# 6. Gerar os scatter plots
plt.figure(figsize=(12, 5))

# Scatter plot do atributo mais correlacionado
plt.subplot(1, 2, 1)
plt.scatter(df[top1_feature], df[target_name], alpha=0.5, color='teal')
plt.xlabel(top1_feature)
plt.ylabel(target_name)
plt.title(f"Correlação com {target_name}: {top1_feature}")

# Scatter plot do segundo atributo mais correlacionado
plt.subplot(1, 2, 2)
plt.scatter(df[top2_feature], df[target_name], alpha=0.5, color='orange')
plt.xlabel(top2_feature)
plt.ylabel(target_name)
plt.title(f"Correlação com {target_name}: {top2_feature}")

plt.tight_layout()
plt.show()
```



0.4 4. Preparação dos Dados

0.4.1 Exercício 4: Separar os conjuntos de treino e teste

Para melhorar a eficiência dos experimentos, vamos usar apenas os atributos `OverallQual`, `GrLivArea`, `Neighborhood` e `GarageCars` (código fornecido abaixo).

Para garantir que a distribuição de preços seja semelhante nos conjuntos de treino e teste, vamos criar uma categoria de preços para usar na amostragem estratificada (código fornecido abaixo).

Em seguida, faça a divisão das instâncias em `train_set` e `test_set` usando a função `train_test_split`, deixando 20% das instâncias no conjunto de teste, e usando `random_state=42`.

Vamos chamar as features do conjunto de treino de `housing_features`, e seus rótulos de `housing_labels` (código fornecido abaixo).

[77]:
```
# Seleciona apenas os atributos relevantes
df = df[["OverallQual", "GrLivArea", "Neighborhood", "GarageCars",␣
 ↪"SalePrice"]].copy()

# Cria a coluna de categoria de preço para estratificação
df["price_cat"] = pd.cut(df["SalePrice"],
                         bins=[0., 100000, 150000, 200000, 300000, np.inf],
                         labels=[1, 2, 3, 4, 5])


train_set, test_set = train_test_split(df, test_size=0.2, random_state=42,␣
 ↪stratify=df["price_cat"])

# Remove a coluna auxiliar
for set_ in (train_set, test_set):
    set_.drop("price_cat", axis=1, inplace=True)

# Define features e rótulos
housing_features = train_set.drop("SalePrice", axis=1)
housing_labels = train_set["SalePrice"].copy()
```

0.4.2 Exercício 5: Criar um Pipeline de Pré-processamento Completo

Crie um `ColumnTransformer` (chamado `preprocessor`) que: - Preencha os valores faltantes dos atributos numéricos com a mediana e os padronize. - Preencha os valores faltantes dos atributos categóricos com o valor mais frequente (`SimpleImputer(strategy='most_frequent')`) e aplique One-Hot encoding.

Em seguida, aplicamos esta transformação nas features (`housing_features`), e armazenamos o resultado em `housing_prepared` (código fornecido abaixo).

[78]:
```
# Identifica os tipos de atributos
num_attribs = ["OverallQual", "GrLivArea", "GarageCars"]
cat_attribs = ["Neighborhood"]
```

```
# Pipeline para atributos numéricos
num_pipeline = Pipeline([
    ("imputer", SimpleImputer(strategy="median")),
    ("scaler", StandardScaler())
])

# Pipeline para atributos categóricos
cat_pipeline = Pipeline([
    ("imputer", SimpleImputer(strategy="most_frequent")),
    ("encoder", OneHotEncoder(handle_unknown="ignore"))
])

# ColumnTransformer combinando os dois pipelines
preprocessor = ColumnTransformer([
    ("num", num_pipeline, num_attribs),
    ("cat", cat_pipeline, cat_attribs)
])

# Aplica a transformação
housing_prepared = preprocessor.fit_transform(housing_features)
```

0.5 5. Seleção e Treinamento de Modelos

0.5.1 Exercício 6: Treinar e avaliar modelos de base

Treine os modelos `LinearRegression`, `DecisionTreeRegressor` e `RandomForestRegressor` com parâmetros default e `random_state=42` e imprima o RMSE de cada um no conjunto de treino.

[79]:
```
# Treinamento dos modelos
lin_reg = LinearRegression()
tree_reg = DecisionTreeRegressor(random_state=42)
forest_reg = RandomForestRegressor(random_state=42)

lin_reg.fit(housing_prepared, housing_labels)
tree_reg.fit(housing_prepared, housing_labels)
forest_reg.fit(housing_prepared, housing_labels)

# Previsões no conjunto de treino
lin_preds = lin_reg.predict(housing_prepared)
tree_preds = tree_reg.predict(housing_prepared)
forest_preds = forest_reg.predict(housing_prepared)

# Cálculo do RMSE
lin_rmse = np.sqrt(mean_squared_error(housing_labels, lin_preds))
tree_rmse = np.sqrt(mean_squared_error(housing_labels, tree_preds))
forest_rmse = np.sqrt(mean_squared_error(housing_labels, forest_preds))
```

```
# Exibição dos resultados
print(f"Linear Regression RMSE: {lin_rmse:.2f}")
print(f"Decision Tree RMSE: {tree_rmse:.2f}")
print(f"Random Forest RMSE: {forest_rmse:.2f}")
```

```
Linear Regression RMSE: 35254.91
Decision Tree RMSE: 2465.67
Random Forest RMSE: 13196.16
```

0.5.2 Exercício 7: Validação cruzada para o RandomForestRegressor

Avalie o `RandomForestRegressor` (com parâmetros default, e `random_state=42`) usando validação cruzada com 10 folds para ter uma estimativa mais robusta de seu desempenho. Qual a média e desvio padrão do RMSE?

[80]:
```
# Modelo com parâmetros padrão
forest_reg = RandomForestRegressor(random_state=42)

# Validação cruzada com scoring negativo do MSE
scores = cross_val_score(forest_reg, housing_prepared, housing_labels,
                         scoring="neg_mean_squared_error", cv=10)

# Converte para RMSE
rmse_scores = np.sqrt(-scores)

# Exibe os resultados
print("RMSE por fold:", rmse_scores)
print(f"Média do RMSE: {rmse_scores.mean():.2f}")
print(f"Desvio padrão do RMSE: {rmse_scores.std():.2f}")
```

```
RMSE por fold: [35480.79669454 31793.8360906  34002.17498559 26642.53198639
 31021.77910471 36676.36066141 32218.38285276 40534.07772505
 46470.13928305 33203.69695214]
Média do RMSE: 34804.38
Desvio padrão do RMSE: 5222.66
```

0.6 6. Ajuste Fino e Avaliação Final

0.6.1 Exercício 8: Otimização com GridSearchCV

Use `GridSearchCV` para encontrar os melhores hiperparâmetros para o `RandomForestRegressor`. Teste os valores 20, 30 e 50 para o parâmetro `n_estimators`. Use 5 folds na validação cruzada.

Qual o melhor RMSE encontrado? Para qual valor do parâmetro `n_estimators`?

[81]:
```
# Define o modelo base
forest_reg = RandomForestRegressor(random_state=42)

# Define os hiperparâmetros a testar
```

```
param_grid = {
    "n_estimators": [20, 30, 50]
}

# Configura o GridSearchCV
grid_search = GridSearchCV(forest_reg, param_grid, cv=5,
                           scoring="neg_mean_squared_error",
                           return_train_score=True)

# Executa a busca
grid_search.fit(housing_prepared, housing_labels)

# Extrai o melhor modelo e RMSE
best_model = grid_search.best_estimator_
best_params = grid_search.best_params_
best_rmse = np.sqrt(-grid_search.best_score_)

# Exibe os resultados
print(f"Melhor RMSE: {best_rmse:.2f}")
print(f"Melhor valor de n_estimators: {best_params['n_estimators']}")
```

```
Melhor RMSE: 35381.42
Melhor valor de n_estimators: 30
```

0.6.2 Exercício 9: Avaliar o melhor modelo no conjunto de teste

Finalmente, avalie no conjunto de teste o desempenho do melhor modelo obtido na otimização de hiparparâmetros. Lembre de aplicar o pré-processamento nos atributos do conjunto de teste.

Qual RMSE obtido?

[82]:
```
# Aplica o pré-processamento nas features do conjunto de teste
X_test_prepared = preprocessor.transform(test_set.drop("SalePrice", axis=1))
y_test = test_set["SalePrice"].copy()

# Faz previsões com o melhor modelo encontrado no GridSearchCV
final_predictions = best_model.predict(X_test_prepared)

# Calcula o RMSE no conjunto de teste
final_rmse = np.sqrt(mean_squared_error(y_test, final_predictions))

# Exibe o resultado
print(f"RMSE no conjunto de teste: {final_rmse:.2f}")
```

```
RMSE no conjunto de teste: 33369.64
```