prodigy-ml-01

April 6, 2025

[1]:
```
# Import Required Libraries
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.linear_model import LinearRegression
from sklearn.model_selection import train_test_split
from sklearn.metrics import mean_squared_error, r2_score
```

[2]:
```
# Load the Dataset
df = pd.read_csv(r"C:\Users\TUSHAR CHOUDHARY\Downloads\train.csv")
```

[8]:
```
df
```

[8]:

	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	
…	…	…	…	…	…	…	…	…	
1455	1456	60	RL	62.0	7917	Pave	NaN	Reg	
1456	1457	20	RL	85.0	13175	Pave	NaN	Reg	
1457	1458	70	RL	66.0	9042	Pave	NaN	Reg	
1458	1459	20	RL	68.0	9717	Pave	NaN	Reg	
1459	1460	20	RL	75.0	9937	Pave	NaN	Reg	

	LandContour	Utilities	…	PoolArea	PoolQC	Fence	MiscFeature	MiscVal	\
0	Lvl	AllPub	…	0	NaN	NaN	NaN	0	
1	Lvl	AllPub	…	0	NaN	NaN	NaN	0	
2	Lvl	AllPub	…	0	NaN	NaN	NaN	0	
3	Lvl	AllPub	…	0	NaN	NaN	NaN	0	
4	Lvl	AllPub	…	0	NaN	NaN	NaN	0	
…	…	…	…	…	…	…	…	…	
1455	Lvl	AllPub	…	0	NaN	NaN	NaN	0	
1456	Lvl	AllPub	…	0	NaN	MnPrv	NaN	0	
1457	Lvl	AllPub	…	0	NaN	GdPrv	Shed	2500	

```
1458         Lvl    AllPub  …        0    NaN    NaN         NaN       0
1459         Lvl    AllPub  …        0    NaN    NaN         NaN       0

     MoSold YrSold  SaleType  SaleCondition  SalePrice
0         2   2008        WD         Normal     208500
1         5   2007        WD         Normal     181500
2         9   2008        WD         Normal     223500
3         2   2006        WD        Abnorml     140000
4        12   2008        WD         Normal     250000
…     …    …    …       …             …        …
1455      8   2007        WD         Normal     175000
1456      2   2010        WD         Normal     210000
1457      5   2010        WD         Normal     266500
1458      4   2010        WD         Normal     142125
1459      6   2008        WD         Normal     147500

[1460 rows x 81 columns]
```

[7]:
```
df.info
```

[7]:
```
<bound method DataFrame.info of         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
…     …          …       …           …       …    …   …      …
1455  1456          60       RL         62.0     7917   Pave   NaN      Reg
1456  1457          20       RL         85.0    13175   Pave   NaN      Reg
1457  1458          70       RL         66.0     9042   Pave   NaN      Reg
1458  1459          20       RL         68.0     9717   Pave   NaN      Reg
1459  1460          20       RL         75.0     9937   Pave   NaN      Reg

     LandContour Utilities  … PoolArea PoolQC  Fence MiscFeature MiscVal  \
0            Lvl    AllPub  …        0    NaN    NaN         NaN       0
1            Lvl    AllPub  …        0    NaN    NaN         NaN       0
2            Lvl    AllPub  …        0    NaN    NaN         NaN       0
3            Lvl    AllPub  …        0    NaN    NaN         NaN       0
4            Lvl    AllPub  …        0    NaN    NaN         NaN       0
…          …       …  …      …     …    …          …      …
1455         Lvl    AllPub  …        0    NaN    NaN         NaN       0
1456         Lvl    AllPub  …        0    NaN  MnPrv         NaN       0
1457         Lvl    AllPub  …        0    NaN  GdPrv        Shed    2500
1458         Lvl    AllPub  …        0    NaN    NaN         NaN       0
1459         Lvl    AllPub  …        0    NaN    NaN         NaN       0
```

```
     MoSold YrSold  SaleType  SaleCondition  SalePrice
0         2   2008        WD         Normal     208500
1         5   2007        WD         Normal     181500
2         9   2008        WD         Normal     223500
3         2   2006        WD        Abnorml     140000
4        12   2008        WD         Normal     250000
…      …     …       …             …          …
1455      8   2007        WD         Normal     175000
1456      2   2010        WD         Normal     210000
1457      5   2010        WD         Normal     266500
1458      4   2010        WD         Normal     142125
1459      6   2008        WD         Normal     147500

[1460 rows x 81 columns]>
```

[3]:
```
df.head()
```

[3]:
```
   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]
```

[4]:
```
df.isnull().sum()
```

[4]:
```
Id                0
MSSubClass        0
MSZoning          0
LotFrontage     259
LotArea           0
```

```
                ...
MoSold              0
YrSold              0
SaleType            0
SaleCondition       0
SalePrice           0
Length: 81, dtype: int64
```

[6]:
```
df.shape
```

[6]:
```
(1460, 81)
```

[10]:
```
# Convert categorical columns to numerical
df = pd.get_dummies(df, drop_first=True)
```

[11]:
```
# Define features (X) and target variable (y)
X = df.iloc[:, :-1]  # Select all columns except the last one as features
y = df.iloc[:, -1]   # Select the last column as the target variable
```

[17]:
```
#missing values detection in array
np.isnan(X).sum()
```

[17]:
```
Id                          0
MSSubClass                  0
LotFrontage               259
LotArea                     0
OverallQual                 0
                           ...
SaleType_WD                 0
SaleCondition_AdjLand       0
SaleCondition_Alloca        0
SaleCondition_Family        0
SaleCondition_Normal        0
Length: 245, dtype: int64
```

[18]:
```
#missing values detection in array
np.isnan(y).sum()
```

[18]:
```
0
```

[20]:
```
# fixing the missing values
from sklearn.impute import SimpleImputer
imputer = SimpleImputer(missing_values=np.nan, strategy='mean')
X = imputer.fit_transform(X)  # to transform all the changes in x.
```

[22]:
```
from sklearn.model_selection import train_test_split
```

```
X_train, X_test, y_train,y_test = train_test_split(X,y,test_size=.30,␣
 ↪random_state = 0)  # .30 means 30% data will be taken as testing and␣
 ↪remaning 70% will be taken for training the model.
```

[23]:
```
from sklearn.linear_model import LinearRegression
regr = LinearRegression()
regr.fit(X_train, y_train)
```

[23]:
```
LinearRegression()
```

[24]:
```
pred = regr.predict(X_test)
```

[25]:
```
pred
```

[25]:
```
array([ 2.59927798e-02,  3.41933634e-02, -2.38978627e-02, -2.79982764e-03,
       -1.91471912e-02, -3.10991434e-02,  4.29617587e-02, -5.22610530e-03,
        1.76043156e-02,  1.71005281e-03,  9.88766740e-01,  3.17715774e-02,
       -1.02635681e-02, -2.01370883e-02, -2.57144772e-02, -7.62931202e-04,
       -3.15026306e-03,  8.39894892e-03,  1.46055561e-02,  2.78866590e-02,
       -1.37241902e-02,  2.30349816e-01, -1.20947832e-01,  2.52337205e-03,
        5.39597108e-04, -6.01419723e-04, -1.75106294e-02, -2.60431631e-03,
       -1.13679351e-02, -7.23268471e-03,  3.14955387e-03, -1.04523590e-02,
       -2.05671515e-03,  2.70806847e-02,  1.00217263e+00, -2.26928745e-02,
        6.31319446e-03, -3.17236398e-03,  1.55498075e-03, -6.16877718e-04,
       -1.47395557e-02, -5.81618938e-02,  9.86994820e-01,  2.38805345e-03,
        4.72254198e-03, -3.85571056e-02,  4.76183980e-04, -4.44296744e-02,
       -1.12753447e-02,  4.50616728e-03,  2.11069208e-03, -7.89264509e-03,
       -2.47082611e-02,  2.34407775e-02, -2.58896968e-03, -1.36321120e-02,
        1.73846578e-02, -1.34900308e-02,  2.43481505e-02,  2.34633271e-02,
        3.84636915e-02, -1.77578466e-02, -2.90312093e-02, -2.15723805e-02,
        1.28696934e-02, -2.52420940e-02, -3.25569220e-03,  9.77907530e-01,
       -1.09114790e-02, -5.24029181e-03,  5.68977861e-03, -1.92920532e-02,
        1.29777691e-01,  9.91198142e-01,  1.16017803e-02,  9.89374634e-01,
       -2.02581239e-02, -5.52201060e-02,  9.74863316e-01, -7.56903625e-02,
        5.58297871e-03,  1.75632738e-02, -2.63735796e-02, -2.60373697e-02,
        1.95825228e-02,  1.09848165e-02, -3.68605579e-02,  9.95030991e-01,
       -2.87280692e-03, -8.10388647e-03, -2.70068171e-03,  3.92679098e-03,
       -3.44859646e-02, -3.31052748e-03,  1.00000302e+00, -1.42087526e-02,
        5.85871640e-03, -9.58065248e-03, -4.52201688e-03, -6.00788886e-04,
       -3.51103466e-04,  2.48120288e-01, -3.88190614e-02, -2.04705552e-02,
        7.72753088e-03, -4.78952646e-04,  2.49509553e-02,  3.47975339e-02,
        1.00131568e+00, -6.93913541e-03, -9.10776929e-03,  1.50942630e-02,
       -1.17639666e-02, -2.95252660e-02,  5.38678823e-02,  8.27468419e-03,
        1.35540063e-02,  2.74359103e-02, -2.14502015e-02,  1.69445391e-03,
        1.46826051e-02,  4.19281803e-02,  1.63466909e-02,  1.67629916e-02,
       -3.08217570e-03,  9.85726267e-01,  9.86447871e-01,  1.00009806e+00,
        1.00486846e+00, -2.73834207e-02,  4.55339101e-02,  2.51872483e-03,
```

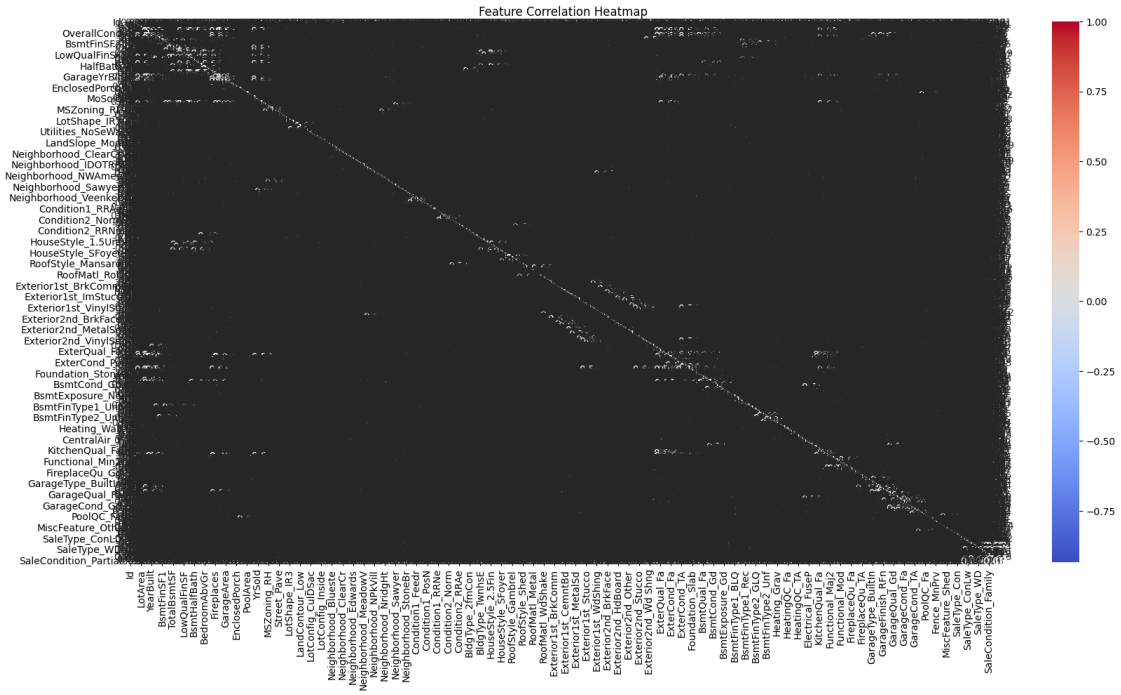
```
 5.05958805e-02,  2.35125316e-03,  2.34944769e-03,  7.45944429e-03,
 1.07142925e-03,  1.71347113e-02,  1.45220430e-02, -5.53366544e-03,
-8.91663267e-03, -5.96225580e-03, -6.52610926e-03,  1.95291510e-02,
 3.65867433e-02, -8.13134030e-03, -1.51086252e-02, -1.44882757e-04,
 9.61026972e-03,  4.41275636e-02, -1.43264844e-02,  9.98142333e-01,
 1.21567501e-02,  1.63496205e-02,  2.58289142e-02, -1.27935208e-02,
 6.15506466e-02, -2.01956432e-02,  1.02033809e+00, -2.29491971e-02,
 5.32380241e-02,  7.21742091e-03,  9.85783579e-01, -7.68109119e-02,
 4.39004415e-02, -1.91363386e-03, -4.76094052e-02,  1.21781533e-02,
 5.74314162e-02,  3.86842337e-02, -3.76175558e-02,  8.80301781e-03,
-8.45692382e-03,  1.40021503e-02, -1.93760211e-02,  5.99881704e-02,
 4.94291222e-05,  2.32077151e-02, -1.93093390e-02, -1.68714377e-02,
 5.01092046e-02, -9.43472729e-03,  4.53010607e-02,  4.35149505e-03,
-2.11101961e-02,  3.96843413e-02, -1.17378595e-03,  9.94462450e-01,
 9.62870330e-01,  2.33703171e-03, -6.08302039e-03, -1.19006045e-02,
 3.23184368e-04, -1.90671309e-02, -6.69927929e-02, -2.78766665e-03,
-7.92275002e-03, -4.80904892e-02, -1.37059956e-02, -1.11686244e-03,
 1.72301228e-05,  1.03473470e-03, -1.79066993e-02, -1.16511362e-02,
-2.69434753e-02, -1.32312732e-02,  6.01312931e-04,  1.57083414e-02,
-2.57818952e-02,  1.54070210e-02, -1.47927028e-03,  1.60161649e-02,
-6.47044691e-03, -1.15418971e-02, -3.55319197e-03,  1.16153477e-01,
-2.91789691e-02, -6.73880869e-02, -3.26168229e-04, -1.00851266e-02,
-1.64562845e-02,  1.90870614e-02,  6.55835959e-03,  5.61606431e-03,
 1.16129077e-03, -1.88771210e-03,  4.57565613e-03, -5.64182297e-02,
 2.55130300e-02, -7.35813536e-03,  1.28424232e-02,  2.31238207e-02,
-2.17325029e-03,  5.86667401e-03, -1.22812009e-02, -1.14097300e-02,
 1.71977295e-02,  5.38067406e-03, -1.42686539e-01,  1.65917790e-02,
 5.52425608e-03,  2.15711966e-02, -1.08720842e-02,  4.03173190e-03,
 5.48903109e-02,  1.24966220e-02,  2.69810669e-02, -2.80490756e-03,
-1.88657995e-02,  6.50532168e-03, -1.14012531e-02,  9.98137130e-01,
 2.90573049e-02,  9.66607797e-01,  9.89725588e-01, -3.90788243e-03,
 4.60378622e-03, -4.89567251e-03, -6.77453584e-02,  1.09692326e+00,
 2.47844645e-02, -2.78904412e-02,  2.59554810e-02, -6.38806942e-02,
 9.85123852e-01,  9.89374912e-01, -4.50528514e-03,  3.06329659e-02,
-1.75585898e-02,  4.51682434e-03,  1.92443521e-03,  2.13714130e-02,
-3.84028534e-03, -7.61065814e-03,  4.57688871e-02,  1.32715071e-02,
 1.73548618e-02, -1.48240906e-02,  7.40930675e-03, -4.28700801e-02,
 2.53035555e-02,  5.49643202e-02,  2.07683941e-02, -1.73302432e-02,
 9.84339485e-01, -2.47339509e-02, -9.56769637e-03,  1.46011442e-02,
-8.83750441e-03, -3.18550946e-02, -6.63005659e-02,  3.68093970e-02,
 9.68805335e-01, -3.63271297e-03,  2.39675597e-03, -1.12350640e-02,
-7.45549494e-03,  1.11757819e-02,  2.71485548e-02,  1.49015819e-02,
-1.04118905e-02, -2.34439781e-02, -4.36725275e-03, -1.34003910e-02,
-6.70864915e-03,  5.55881085e-03, -2.68519529e-02,  3.37021774e-02,
 9.93265904e-01, -1.03975196e-02, -2.34551701e-02,  9.89955540e-01,
 5.14640768e-02, -4.87992093e-03,  1.12971459e-02, -5.87202121e-03,
 3.16145405e-03,  1.69182407e-03,  7.07176519e-03,  2.89407357e-02,
```

```
 2.90722504e-01,  1.75137434e-02, -5.77570488e-03, -4.09258607e-03,
 2.18479403e-02,  2.88199454e-03, -1.47083172e-02,  6.83901955e-03,
 1.07767948e-02,  4.89172728e-03, -1.09231910e-02, -1.36309757e-02,
-8.68042703e-03,  9.89293264e-01, -4.56650376e-03, -1.00804230e-02,
 2.89010998e-02, -1.89799819e-02,  1.14721084e-02, -5.80192732e-03,
-5.77287328e-04,  2.56101557e-02,  9.90402683e-03,  1.83529021e-03,
-8.52728378e-03, -2.69221937e-02,  2.61076548e-02, -7.20767525e-02,
 2.46361408e-02,  2.52860378e-03, -5.25192443e-03, -1.27435734e-02,
 4.33062826e-03, -4.75146676e-02, -6.15015459e-03,  9.81737010e-01,
-1.73535933e-02,  3.16132530e-02, -2.49466325e-02,  6.60938097e-02,
 2.11399762e-02,  1.30355883e-02, -1.62593357e-03,  9.88898549e-01,
 1.61133299e-02,  2.46034454e-02, -1.26172473e-03, -1.85925387e-02,
 9.33925430e-03,  1.23963666e-02,  9.75233248e-03,  4.14654545e-02,
 8.68801703e-03, -4.75627048e-02,  9.98795083e-03,  2.11733394e-02,
 1.15205745e-02, -3.85167082e-02,  1.99094579e-03,  9.03090394e-03,
-3.66221272e-02, -1.18266110e-02,  3.18644495e-03,  2.42698969e-03,
 2.41084734e-02,  6.22714245e-03, -6.70204192e-02,  9.95957618e-01,
 9.89251934e-01,  1.02014801e+00,  2.22687367e-01,  1.77670915e-02,
-5.48249544e-03,  9.70549325e-03,  2.45571724e-03,  9.56889917e-03,
-1.51914242e-03,  9.99265062e-01, -5.06803619e-02,  9.43668336e-03,
 3.12899162e-03,  8.68664811e-03, -3.69587370e-03,  2.10010951e-03,
-2.03622998e-02,  9.70807436e-01, -5.98475422e-03,  9.83623006e-01,
-1.44167945e-02,  1.06040676e+00, -6.34800606e-02, -8.77264094e-03,
-2.79735180e-03, -1.38124741e-02,  8.55562980e-03,  1.52430382e-02,
 1.86813616e-02,  6.60177083e-03, -2.29464517e-02,  6.72772472e-03,
-6.28940165e-03,  9.63715592e-01, -6.88415106e-03,  2.63160130e-02,
-5.07050331e-03, -4.76687916e-03,  9.24661429e-04, -1.71680624e-02,
-6.33194680e-03,  8.83027600e-03,  2.21417471e-02,  9.97788443e-01,
-1.26971821e-03,  1.88484415e-01,  1.41667224e-02,  1.37422327e-02,
 8.27258092e-03,  1.00663222e+00])
```

[26]:
```python
#calculate R squared
from sklearn.metrics import mean_squared_error
r2_score = regr.score(X,y)
print(r2_score)
```

```
0.9758584544965914
```

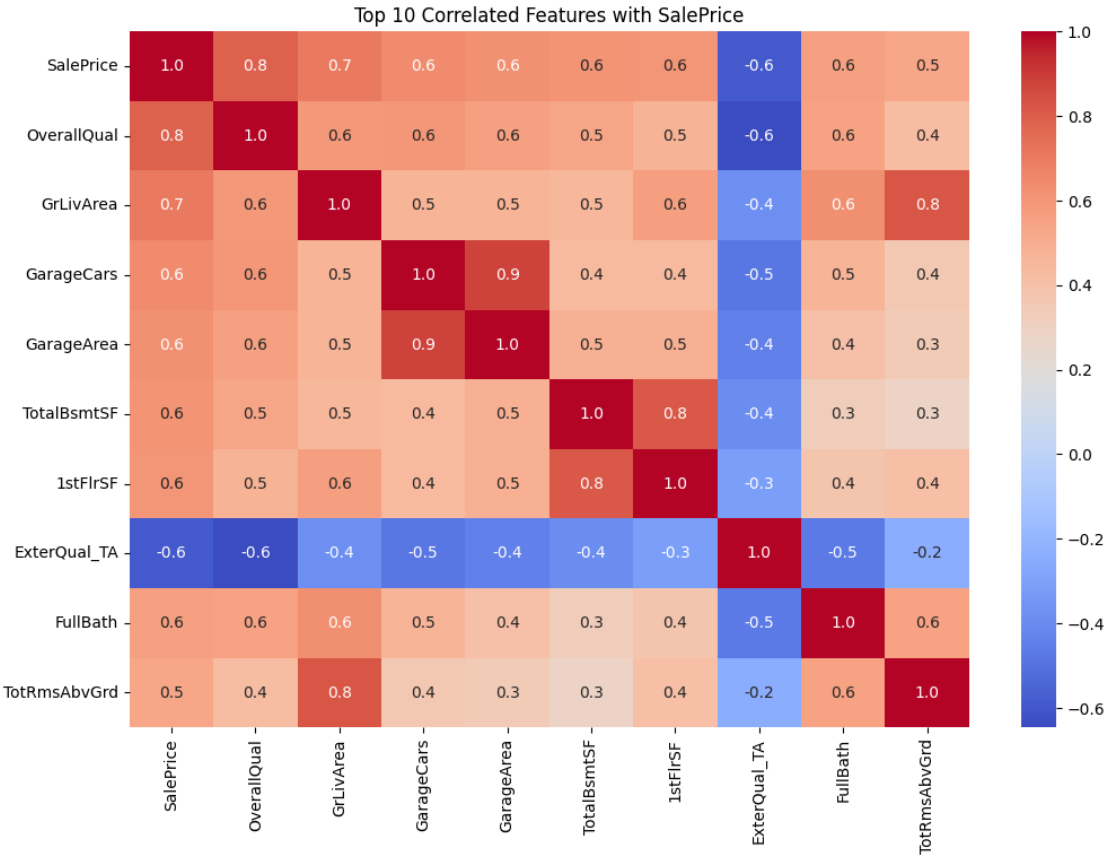
[36]:
```python
# Correlation Heatmap
plt.figure(figsize=(20, 10))
sns.heatmap(df.corr(), annot=True, cmap='coolwarm')
plt.title('Feature Correlation Heatmap')
plt.show()
```



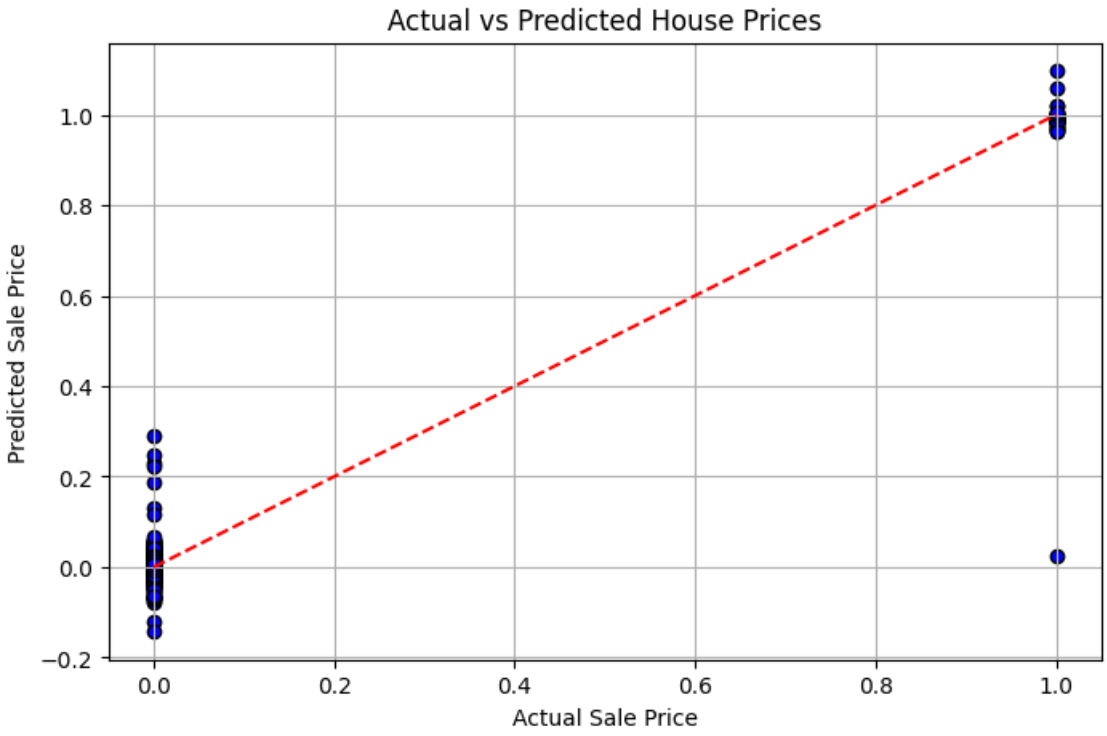
[51]:
```
# Compute the correlation matrix
corr_matrix = df.corr()

# Select the top 10 features most correlated with 'SalePrice'
top_10_features = corr_matrix['SalePrice'].abs().sort_values(ascending=False).
 ↪head(10).index # abs is used to include both positive and negative␣
 ↪correlations.

# Create a heatmap using only those top 10 features
plt.figure(figsize=(12, 8))
sns.heatmap(df[top_10_features].corr(), annot=True, cmap='coolwarm', fmt=".1f")
plt.title('Top 10 Correlated Features with SalePrice')
plt.show()
```

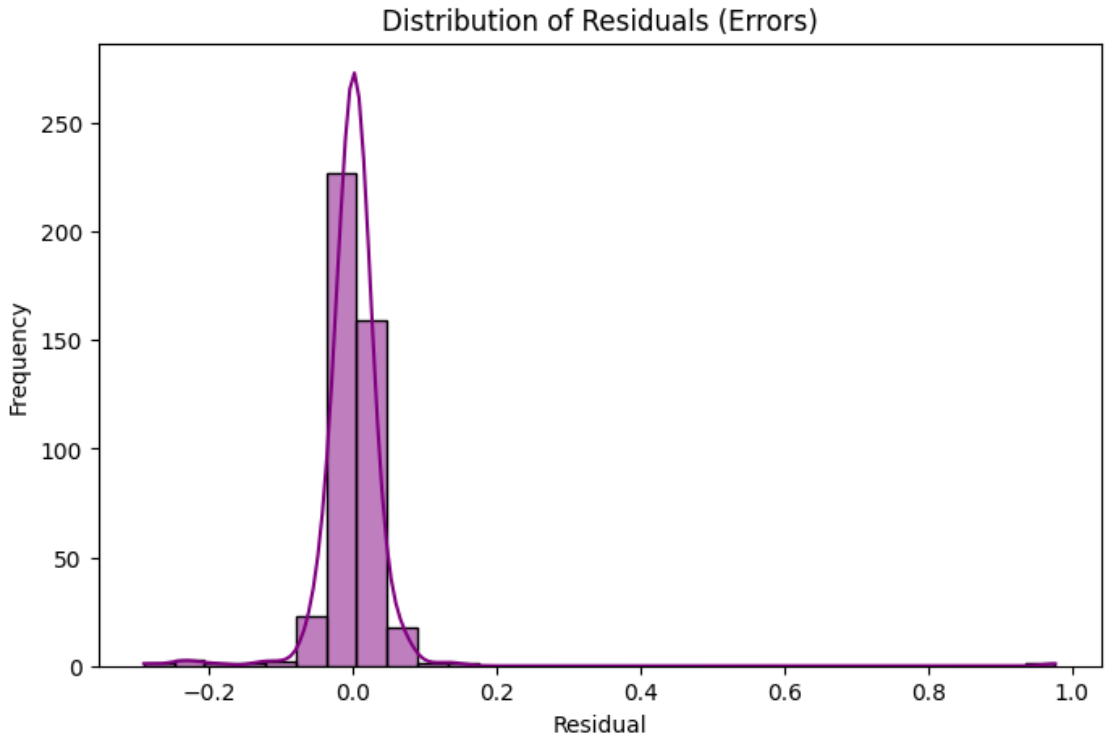


[42]:
```
# Actual vs Predicted
plt.figure(figsize=(8, 5))
plt.scatter(y_test, pred, color='blue', edgecolors='k')
plt.xlabel("Actual Sale Price")
plt.ylabel("Predicted Sale Price")
plt.title("Actual vs Predicted House Prices")
plt.plot([y_test.min(), y_test.max()], [y_test.min(), y_test.max()],␣
 ↪color='red', linestyle='--')
plt.grid(True)
plt.show()
```



[43]:
```
# Note:-  The red dashed line shows perfect prediction.
```

[46]:
```
# Residual Plot (Error Distribution)
residuals = y_test - pred
plt.figure(figsize=(8, 5))
sns.histplot(residuals, kde=True, bins=30, color='purple')
plt.title("Distribution of Residuals (Errors)")
plt.xlabel("Residual")
plt.ylabel("Frequency")
plt.show()
```



[47]:
```
# Note: -A centered, bell-shaped curve suggests good model behavior.
```

[ ]:
```

```