ml-task-01

May 17, 2024

1 ML Task-01

Implement a linear regression model to predict the prices of houses based on their square footage and the number of bedrooms and bathrooms.

Dataset : - https://www.kaggle.com/c/house-prices-advanced-regression-techniques/data

[302]:
```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
```

[303]:
```
df_train = pd.read_csv('train.csv')
df_test = pd.read_csv('test.csv')
```

[304]:
```
df_train.head()
```

[304]:

	Id	MSSubClass	MSZoning	LotFrontage	LotArea	Street	Alley	LotShape	…	PoolArea	PoolQC	Fence	MiscFeature	MiscVal	MoSold	YrSold	SaleType	SaleCondition	SalePrice
0	1	60	RL	65.0	8450	Pave	NaN	Reg	…	0	NaN	NaN	NaN	0	2	2008	WD	Normal	208500
1	2	20	RL	80.0	9600	Pave	NaN	Reg	…	0	NaN	NaN	NaN	0	5	2007	WD	Normal	181500
2	3	60	RL	68.0	11250	Pave	NaN	IR1	…	0	NaN	NaN	NaN	0	9	2008	WD	Normal	223500
3	4	70	RL	60.0	9550	Pave	NaN	IR1	…	0	NaN	NaN	NaN	0	2	2006	WD	Abnorml	140000
4	5	60	RL	84.0	14260	Pave	NaN	IR1	…	0	NaN	NaN	NaN	0	12	2008	WD	Normal	250000

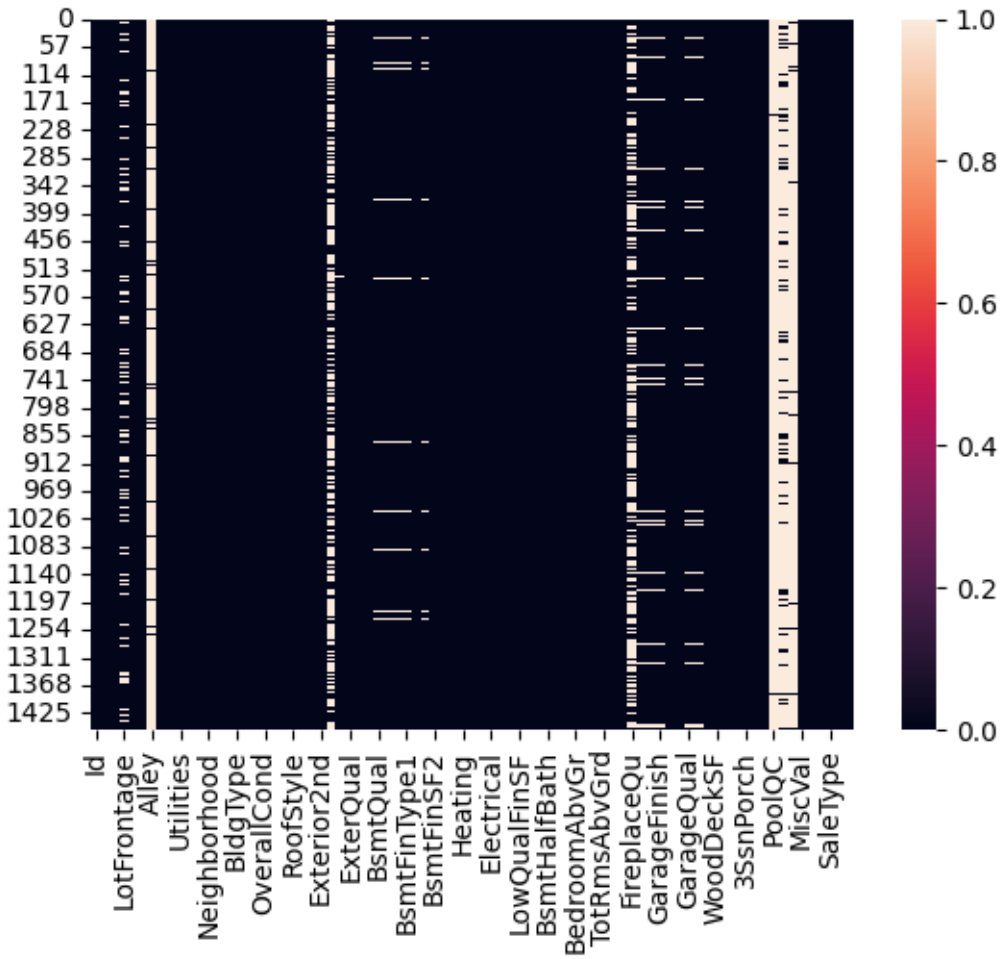
```
[5 rows x 81 columns]
```

[305]:
```
df_test.head()
```

[305]:
```
     Id  MSSubClass MSZoning  LotFrontage  LotArea Street Alley LotShape  \
0  1461          20       RH         80.0    11622   Pave   NaN      Reg
1  1462          20       RL         81.0    14267   Pave   NaN      IR1
2  1463          60       RL         74.0    13830   Pave   NaN      IR1
3  1464          60       RL         78.0     9978   Pave   NaN      IR1
4  1465         120       RL         43.0     5005   Pave   NaN      IR1

  LandContour Utilities  … ScreenPorch PoolArea PoolQC  Fence MiscFeature  \
0         Lvl    AllPub  …         120        0    NaN  MnPrv         NaN
1         Lvl    AllPub  …           0        0    NaN    NaN        Gar2
2         Lvl    AllPub  …           0        0    NaN  MnPrv         NaN
3         Lvl    AllPub  …           0        0    NaN    NaN         NaN
4         HLS    AllPub  …         144        0    NaN    NaN         NaN

  MiscVal MoSold  YrSold  SaleType  SaleCondition
0       0      6    2010        WD         Normal
1   12500      6    2010        WD         Normal
2       0      3    2010        WD         Normal
3       0      6    2010        WD         Normal
4       0      1    2010        WD         Normal

[5 rows x 80 columns]
```

[306]:
```
sns.heatmap(df_train.isnull())
```

[306]:
```
<Axes: >
```



[307]:
```
df_train.columns
```

[307]:
```
Index(['Id', 'MSSubClass', 'MSZoning', 'LotFrontage', 'LotArea', 'Street',
       'Alley', 'LotShape', 'LandContour', 'Utilities', 'LotConfig',
       'LandSlope', 'Neighborhood', 'Condition1', 'Condition2', 'BldgType',
       'HouseStyle', 'OverallQual', 'OverallCond', 'YearBuilt', 'YearRemodAdd',
       'RoofStyle', 'RoofMatl', 'Exterior1st', 'Exterior2nd', 'MasVnrType',
       'MasVnrArea', 'ExterQual', 'ExterCond', 'Foundation', 'BsmtQual',
       'BsmtCond', 'BsmtExposure', 'BsmtFinType1', 'BsmtFinSF1',
       'BsmtFinType2', 'BsmtFinSF2', 'BsmtUnfSF', 'TotalBsmtSF', 'Heating',
       'HeatingQC', 'CentralAir', 'Electrical', '1stFlrSF', '2ndFlrSF',
       'LowQualFinSF', 'GrLivArea', 'BsmtFullBath', 'BsmtHalfBath', 'FullBath',
       'HalfBath', 'BedroomAbvGr', 'KitchenAbvGr', 'KitchenQual',
       'TotRmsAbvGrd', 'Functional', 'Fireplaces', 'FireplaceQu', 'GarageType',
       'GarageYrBlt', 'GarageFinish', 'GarageCars', 'GarageArea', 'GarageQual',
       'GarageCond', 'PavedDrive', 'WoodDeckSF', 'OpenPorchSF',
       'EnclosedPorch', '3SsnPorch', 'ScreenPorch', 'PoolArea', 'PoolQC',
```

```
       'Fence', 'MiscFeature', 'MiscVal', 'MoSold', 'YrSold', 'SaleType',
       'SaleCondition', 'SalePrice'],
      dtype='object')
```

[308]:
```
null_columns=[]
for i in df_train.columns.tolist():
    if df_train[i].isnull().sum() >= 500:
        null_columns.append(i)
        print(i,df_train[i].isnull().sum())
```

```
Alley 1369
MasVnrType 872
FireplaceQu 690
PoolQC 1453
Fence 1179
MiscFeature 1406
```

[309]:
```
df_train = df_train.drop(columns=null_columns)
df_test = df_test.drop(columns=null_columns)
```

[310]:
```
df_train_cleaned = df_train.ffill()
df_test_cleaned = df_test.ffill()
```

[311]:
```
df_train_cleaned.head()
```

[311]:

	Id	MSSubClass	MSZoning	LotFrontage	LotArea	Street	LotShape	LandContour	\
0	1	60	RL	65.0	8450	Pave	Reg	Lvl	
1	2	20	RL	80.0	9600	Pave	Reg	Lvl	
2	3	60	RL	68.0	11250	Pave	IR1	Lvl	
3	4	70	RL	60.0	9550	Pave	IR1	Lvl	
4	5	60	RL	84.0	14260	Pave	IR1	Lvl	

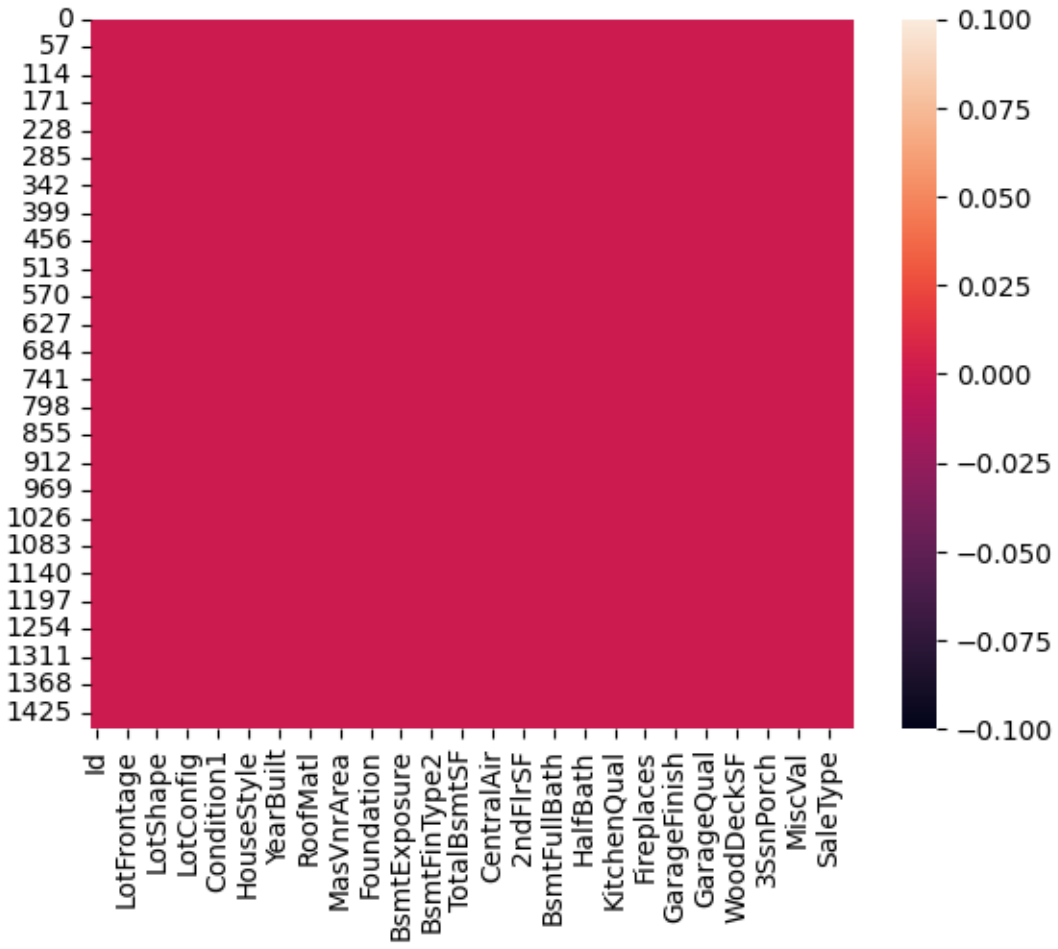
	Utilities	LotConfig	…	EnclosedPorch	3SsnPorch	ScreenPorch	PoolArea	\
0	AllPub	Inside	…	0	0	0	0	
1	AllPub	FR2	…	0	0	0	0	
2	AllPub	Inside	…	0	0	0	0	
3	AllPub	Corner	…	272	0	0	0	
4	AllPub	FR2	…	0	0	0	0	

	MiscVal	MoSold	YrSold	SaleType	SaleCondition	SalePrice
0	0	2	2008	WD	Normal	208500
1	0	5	2007	WD	Normal	181500
2	0	9	2008	WD	Normal	223500
3	0	2	2006	WD	Abnorml	140000
4	0	12	2008	WD	Normal	250000

[5 rows x 75 columns]

[312]:
```
sns.heatmap(df_train_cleaned.isnull())
```

[312]: <Axes: >



[313]:
```
df_train_cleaned.describe()
```

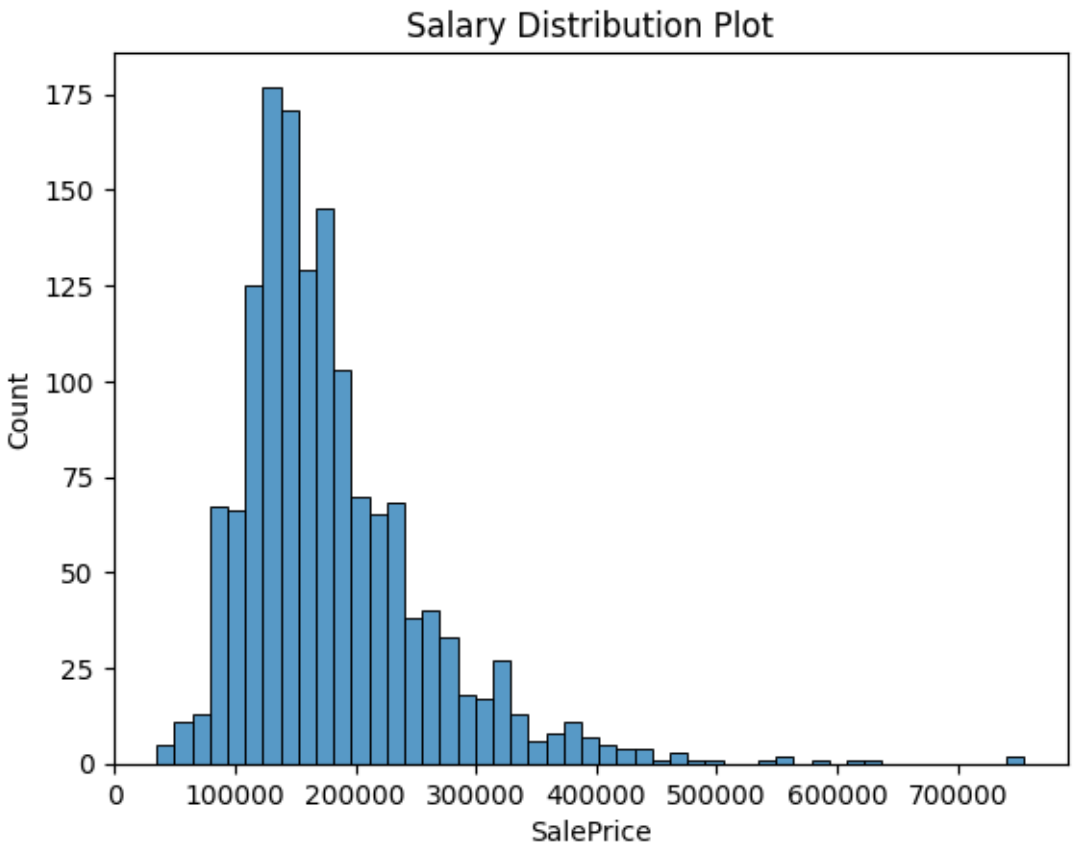
[313]:

	Id	MSSubClass	LotFrontage	LotArea	OverallQual	\
count	1460.000000	1460.000000	1460.000000	1460.000000	1460.000000	
mean	730.500000	56.897260	70.104795	10516.828082	6.099315	
std	421.610009	42.300571	23.846996	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	70.000000	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  1460.000000  1460.000000  …
mean      5.575342  1971.267808   1984.865753   103.492466   443.639726  …
std       1.112799    30.202904     20.645407   180.795612   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   165.250000   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]
```

[314]:
```
plt.title('Salary Distribution Plot')
sns.histplot(df_train_cleaned['SalePrice'])
plt.show()
```



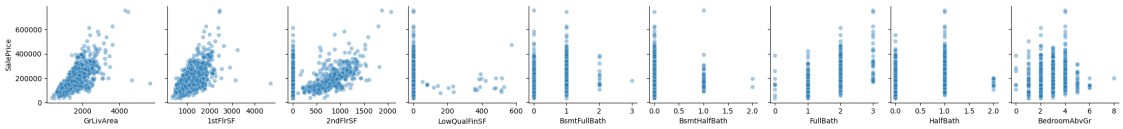
2 Split data

[315]:
```
from sklearn.model_selection import train_test_split
```

[316]:
```
features = ['GrLivArea', '1stFlrSF', '2ndFlrSF', 'LowQualFinSF',␣
 ↪'BsmtFullBath', 'BsmtHalfBath', 'FullBath', 'HalfBath', 'BedroomAbvGr']

target = 'SalePrice'
```

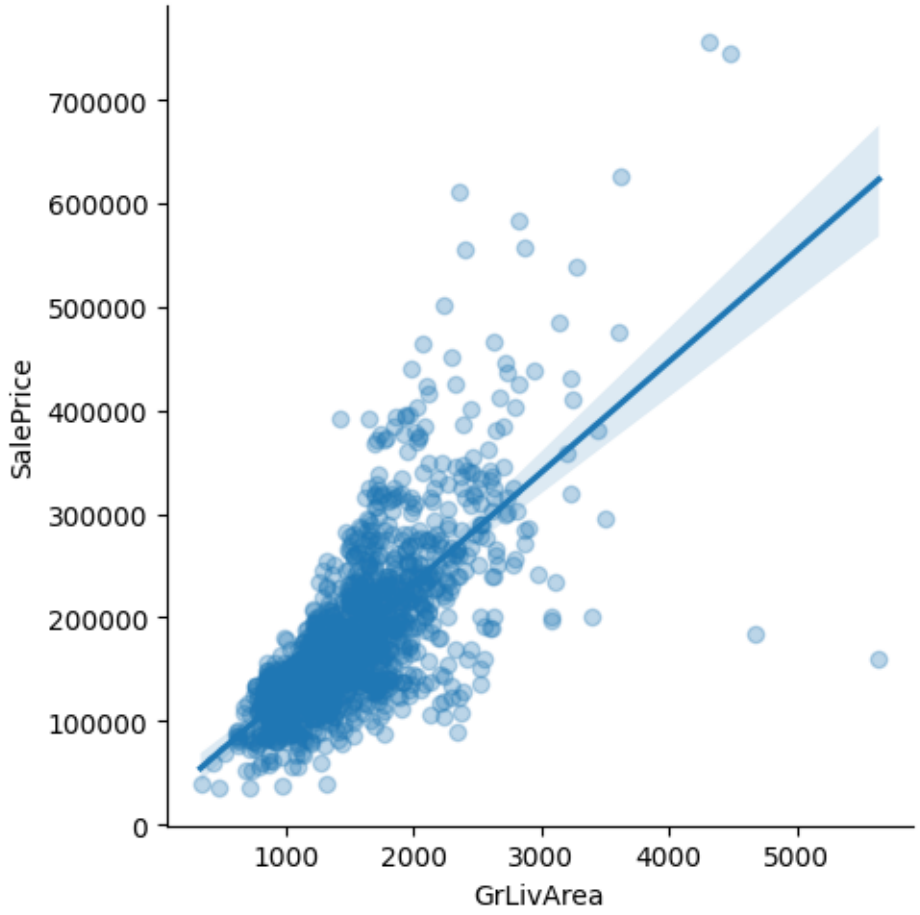
[317]:
```
X = df_train_cleaned[features]
y = df_train_cleaned[target]
```

[318]:
```
df = pd.concat([X, y], axis=1)

# Plotting grid of scatter plots for each feature against the target variable
sns.pairplot(df, x_vars=features, y_vars=[target], kind='scatter',
             plot_kws={'alpha':0.4}, diag_kws={'alpha':0.55, 'bins':40})

plt.show()
```



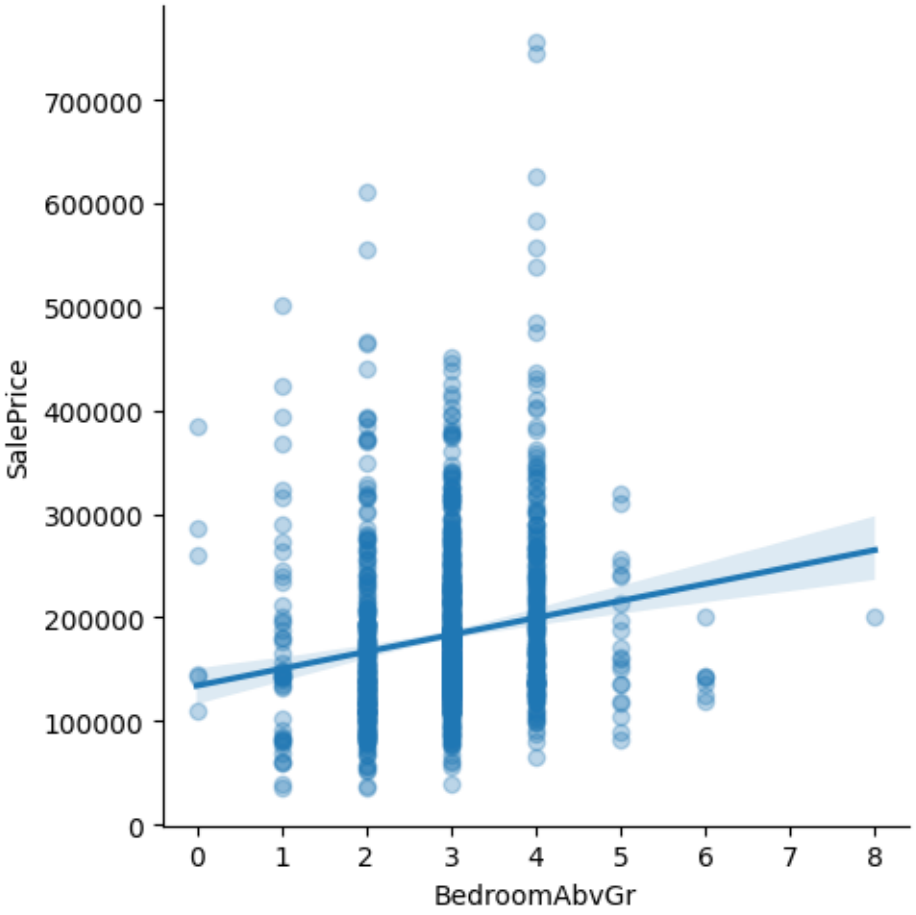
[319]:
```
sns.lmplot(x='GrLivArea',
           y='SalePrice',
           data=df_train_cleaned,
           scatter_kws={'alpha':0.3})
```

[319]: <seaborn.axisgrid.FacetGrid at 0x34f7c99a0>



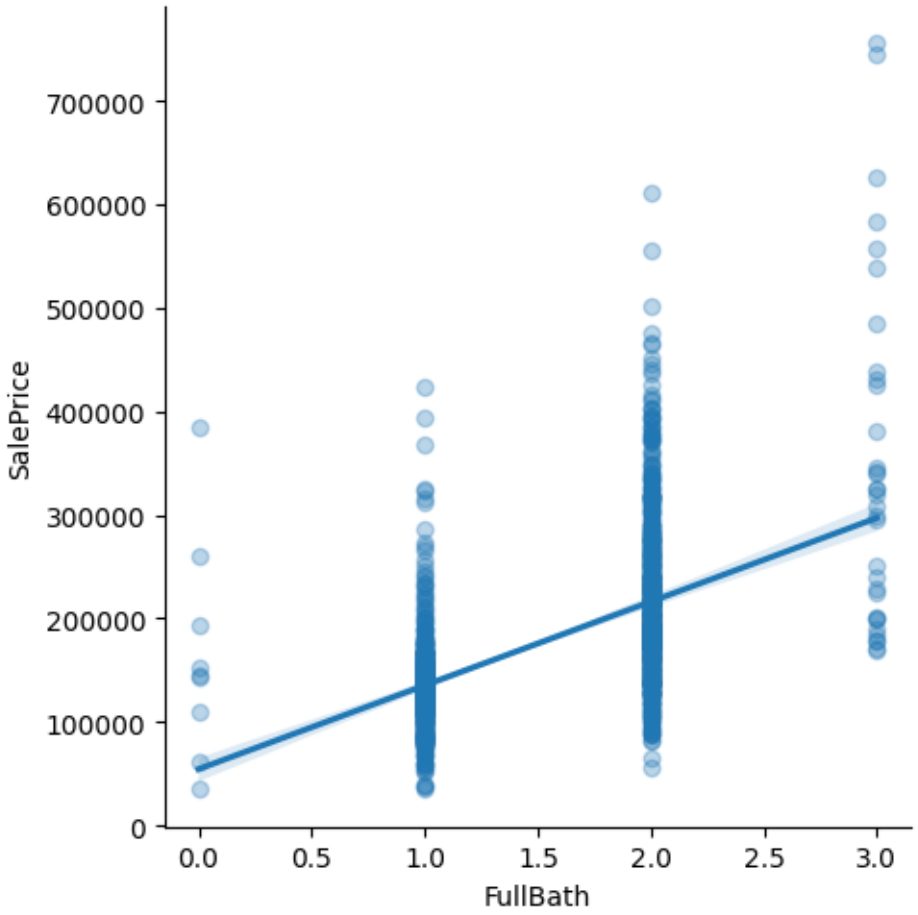
[320]:
```
sns.lmplot(x='BedroomAbvGr',
           y='SalePrice',
           data=df_train_cleaned,
           scatter_kws={'alpha':0.3})
```

[320]: <seaborn.axisgrid.FacetGrid at 0x34fac90d0>



[321]:
```
sns.lmplot(x='FullBath',
           y='SalePrice',
           data=df_train_cleaned,
           scatter_kws={'alpha':0.3})
```

[321]: <seaborn.axisgrid.FacetGrid at 0x2ed606360>



[322]:
```
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size = 0.2,␣
↪random_state = 42)
```

3 Train model (Linear Regression)

[323]:
```
model = LinearRegression()
```

[324]:
```
model.fit(X_train, y_train)
```

[324]:
```
LinearRegression()
```

[325]:
```
coefficients = model.coef_
```

[326]:
```
model.score(X, y)
```

[326]:
```
0.6541686106425739
```

[327]:
```
for feature, coef in zip(features, coefficients):
    print(f'{feature}: {coef}')
```

```
GrLivArea: 45.45358200866404
1stFlrSF: 70.93123244880826
2ndFlrSF: 19.004245633922277
LowQualFinSF: -44.481896073804585
BsmtFullBath: 22125.53078348177
BsmtHalfBath: 6538.890647874652
FullBath: 34869.22887902238
HalfBath: 22717.7813535538
BedroomAbvGr: -18379.60698218067
```

4 Predict results

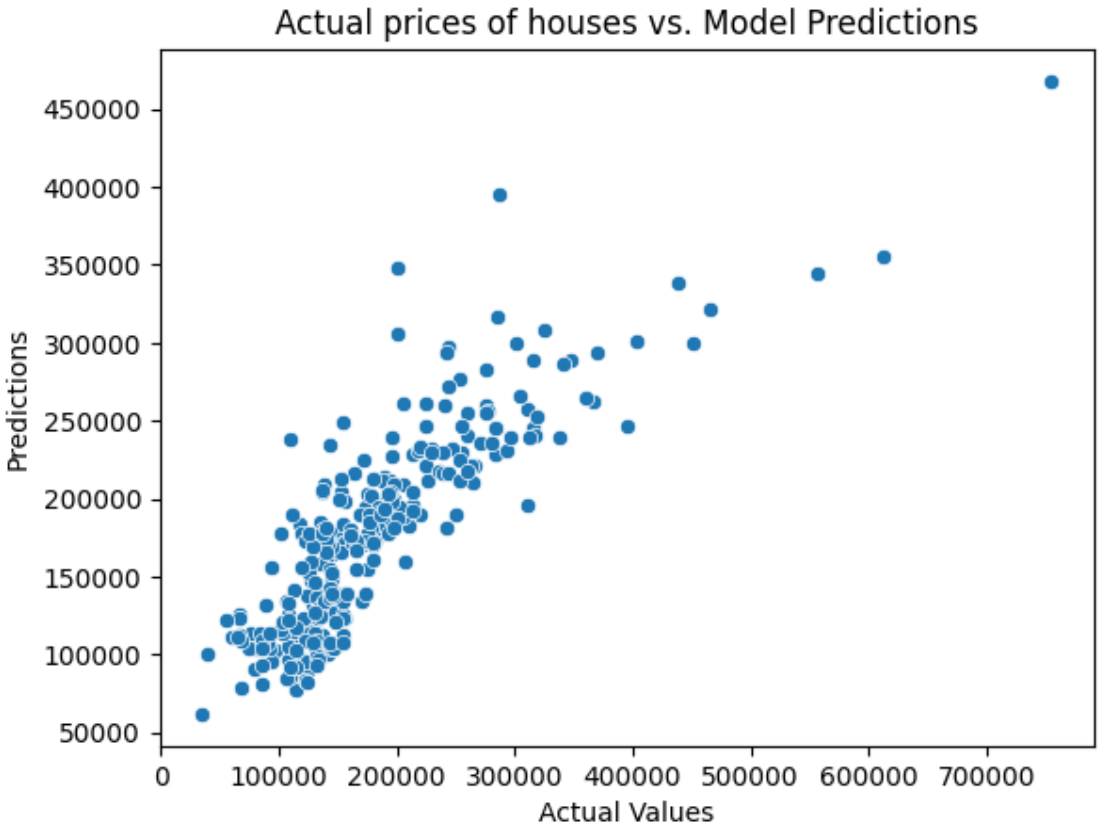
[328]:
```
predictions = model.predict(X_test)
```

[329]:
```
predictions
```

[329]:
```
array([112866.76155431, 308640.3343831 , 120052.08531031, 176290.63790428,
       245946.65513164, 114236.0669219 , 196277.14028942, 177864.07007276,
       114236.0669219 , 139816.75682811, 164181.1874575 , 105874.74768516,
       104456.52017301, 195548.80813709, 197776.41151491, 139849.91980465,
       194102.03176818, 132388.87564281, 118189.92827899, 211156.78931096,
       248865.46919518, 182918.45688664, 195203.29995175, 131470.21316371,
       205570.318217  , 174823.11746156, 172853.76931789, 100964.97573929,
       191090.64462772, 192629.99407356, 111333.7489331 , 245860.69417442,
       305226.89148007, 111206.83941155, 229429.20518078, 136367.56907302,
       208588.96205608, 189493.14184336, 257621.22795059, 103059.90239952,
        77348.7045997 , 261202.22429556,  92594.46280045, 277273.69042059,
       119142.61653332, 183760.69353856, 106551.44683325, 115301.91765622,
       299466.43556541, 148364.58622292,  97017.08574983, 239778.25996539,
       120786.70154331, 394686.44800829, 139607.95647363, 240834.91699761,
       228392.62288327, 154486.30967186, 134156.25759833,  96658.73760436,
        79308.2065481 , 171887.6800169 , 239993.04600578, 210553.25550001,
       245723.07920483, 271802.4713454 , 105224.22513941, 266121.22368692,
        95387.69834743, 189829.79237155, 177169.20115491, 122970.89937385,
        87589.91577878, 156317.93196283, 355254.73844155, 172786.82229611,
       288985.82425152, 286403.71485141, 137996.95647543,  91188.65132486,
       111826.88791641, 125754.94121873, 115396.69273201, 103525.44165735,
       129882.56425863, 134776.84190756, 221459.6776597 , 214358.94359177,
       139607.95647363, 171269.34274727, 203977.95257071, 166161.02281285,
        95378.50464533, 299974.30308751, 204286.8629235 , 170060.53377091,
       188831.10793692, 192358.13741796, 209622.17698435, 254902.9073403 ,
       216286.64614878, 221606.32569836, 297747.09126732,  99801.12759471,
       211914.86248817, 198888.95697103, 123565.2394827 , 260145.60249545,
       103788.74602662, 185320.47624053, 111895.95459619, 137938.32498064,
```

```
147912.40823866, 132651.19727964, 204206.05619133, 116107.41765532,
105271.21387421,  83516.44727277, 189557.06776091, 228614.51147958,
130180.78650258, 139607.95647363, 190313.543449  , 160164.2460368 ,
203359.00674231, 112974.22136707, 217541.59855651,  91223.84951624,
132653.58150817, 171697.3705435 , 181571.96247646, 293431.65943029,
199255.14525628, 113632.98509428,  61909.11587426, 238972.75996629,
261718.36937387, 113323.37581044, 231524.13184101, 467753.95172413,
300828.20388729, 108413.60386457, 201989.52289353, 179888.58266193,
127056.78391642, 108900.75286106, 190301.94843139, 183328.40261906,
 96330.92419924, 123090.50652277, 137742.57710785, 125907.63337944,
348221.07898574, 184095.3244702 , 107375.33423655, 205989.08099145,
238397.20199494, 160437.61592706,  92946.83957828, 137298.64758868,
160516.97898539, 169451.90676742, 230593.05332535, 141252.50894577,
121992.4674914 , 171251.96302178, 215937.47534598, 282486.14211011,
344409.12860946, 216086.54387339, 265275.72803368, 121439.5093817 ,
116333.85050634, 199217.53245323, 316911.72823465, 159035.76649179,
142033.19560221, 211927.32271002, 152486.29340786, 180939.91504086,
170663.79888145, 121914.24234163,  93060.00205828, 130990.97269308,
246600.39187805, 173959.14532072, 235597.60034703, 231980.47739674,
194492.15110488, 103486.75300604, 172960.88783059, 125289.4019609 ,
180243.3858729 , 134272.64241279, 227781.43037418, 261067.00485184,
208763.27879572, 100620.462514  , 224877.8203914 , 133893.6621316 ,
235821.17627384, 177215.58210553, 103555.14455709, 289218.58041761,
194630.145658  , 130955.2361132 , 229121.33007905, 134621.79685616,
165115.6994213 ,  85844.14356192, 246202.80970835, 154301.15699726,
169443.59891329, 172963.18797732, 181166.52893705, 221282.26816876,
187224.12104124, 137298.64758868, 145920.04886023, 177897.32092792,
108202.44397431, 211496.75005223, 230709.43813981, 112751.11657003,
187518.25593953, 123436.43863168, 109361.96242566, 111179.01135753,
139642.71824971, 120945.09868499, 108646.37349348, 172560.20839992,
123090.50652277, 125065.82603408, 233314.48679486, 108257.05325169,
197996.11157516, 138453.30203116, 255927.31512249, 142700.9086713 ,
 84205.56245741, 293369.51702834, 204684.31533739, 338361.31378289,
191126.32554018, 133559.2428721 , 155114.44920237, 189732.78974868,
127522.32317425,  92236.11465498, 177824.18784545, 176143.26501385,
152467.31342453,  80830.40283814, 139636.67037562, 159693.09158209,
103401.54531495, 156253.99258245, 185002.65632955, 259809.21376908,
246525.35686947, 167465.05706751, 108202.44397431, 255278.78132649,
239903.83848195, 192255.99936187, 213009.75264018, 139607.95647363,
117375.23457779, 171982.08062205, 321735.9118778 , 213009.75264018,
224457.01390693, 104256.63969074, 177732.737823  , 120772.00393572,
177425.72259649, 252800.94314807, 234473.99629303, 181716.97885204,
202696.66085039, 114236.0669219 , 181600.63140512,  92585.26909835,
238972.75996629, 190195.10667205, 229362.90849752, 132039.72119943,
217092.41484829, 193394.87353075, 122311.0076527 ,  82585.36875711])
```

[330]:
```
sns.scatterplot(x=y_test, y=predictions)
plt.xlabel('Actual Values')
plt.ylabel('Predictions')
plt.title('Actual prices of houses vs. Model Predictions')
plt.show()
```



5 Evaluation of the model

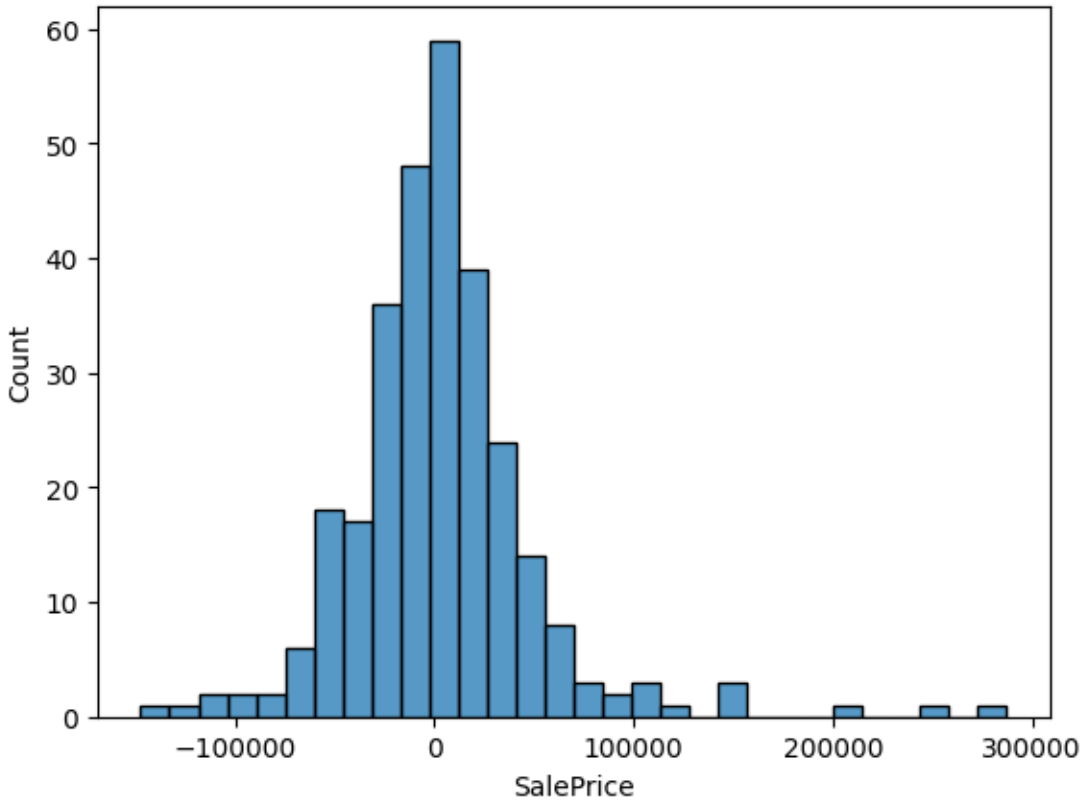
[331]:
```
from sklearn.metrics import mean_squared_error, mean_absolute_error
import math
```

[332]:
```
print('Mean Absolute Error:',mean_absolute_error(y_test, predictions))
print('Mean Squared Error:',mean_squared_error(y_test, predictions))
print('Root Mean Squared Error:',math.sqrt(mean_squared_error(y_test,␣
 ↪predictions)))
```

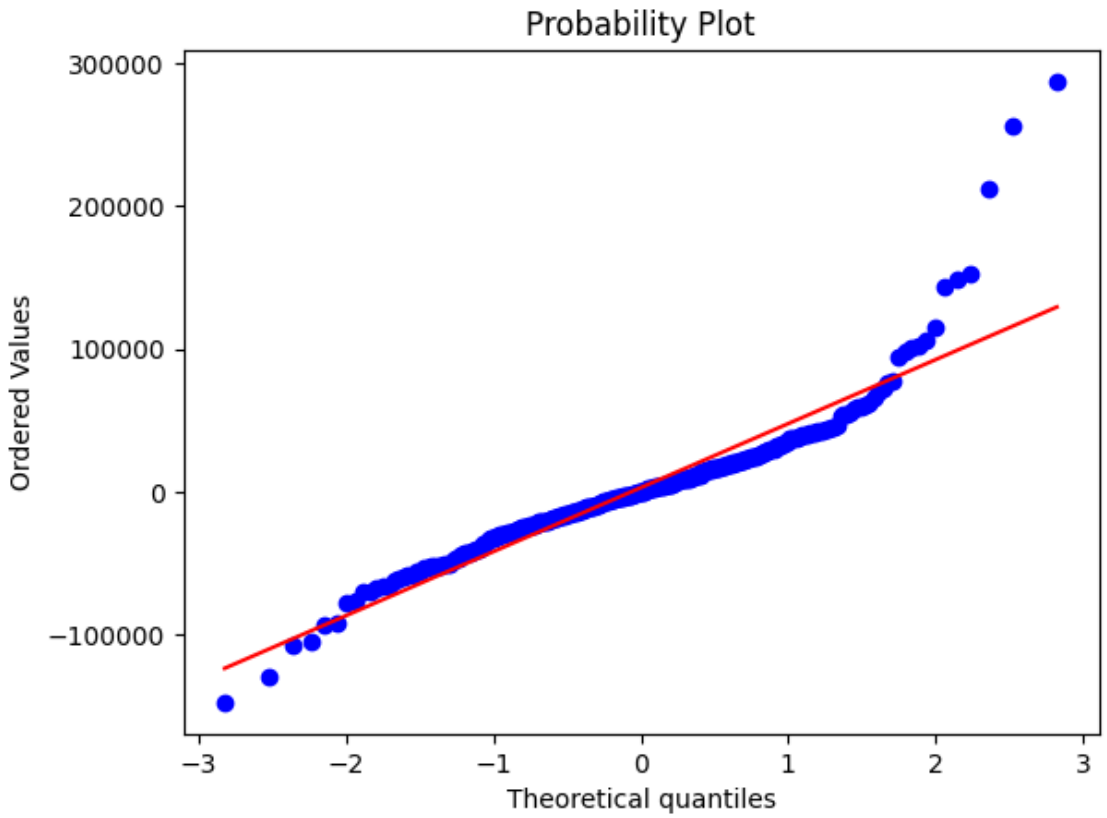
```
Mean Absolute Error: 31410.205502278408
Mean Squared Error: 2269069226.157571
Root Mean Squared Error: 47634.74809587609
```

[333]:
```
residuals = y_test-predictions
sns.histplot(residuals, bins=30)
```

[333]: <Axes: xlabel='SalePrice', ylabel='Count'>



[334]:
```
import pylab
import scipy.stats as stats

stats.probplot(residuals, dist="norm", plot=pylab)
pylab.show()
```



6 Use the trained model on our test data

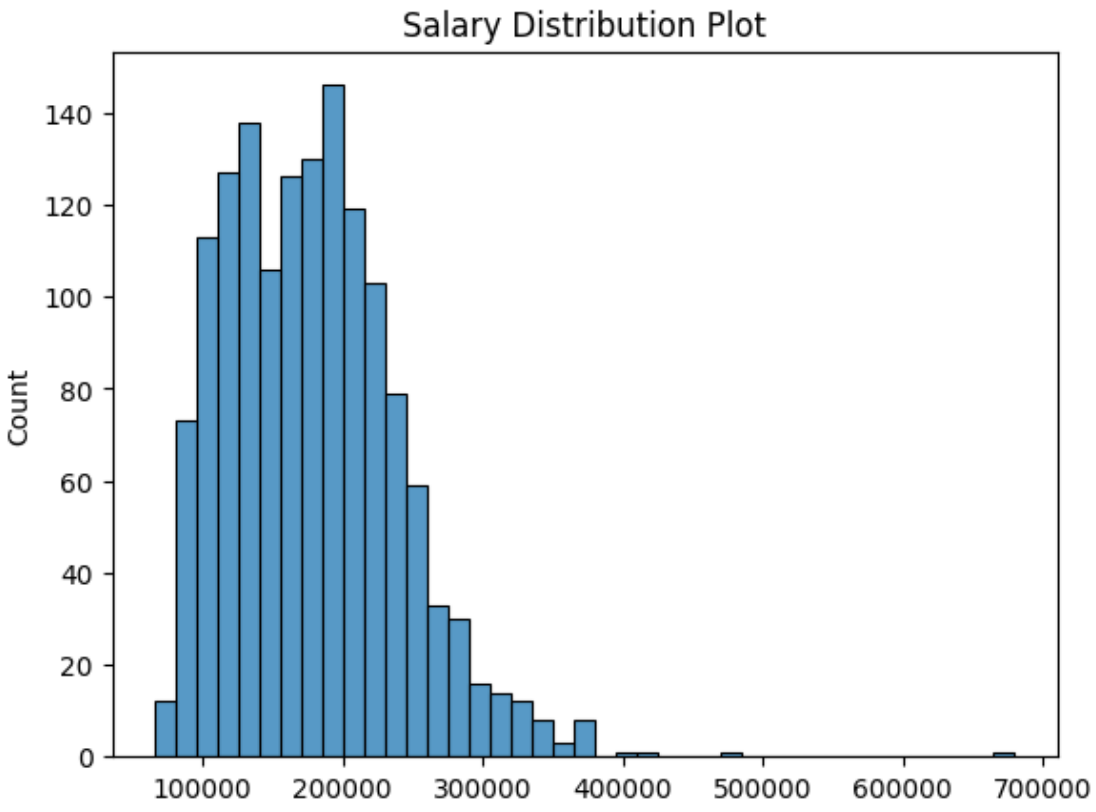
[335]:
```
Xt = df_test_cleaned[features]
```

[336]:
```
test_predictions = model.predict(Xt)
```

[337]:
```
test_predictions
```

[337]:
```
array([104689.28980193, 159422.08883339, 192805.94429241, …,
       128229.8257631 , 101461.04973747, 220250.83345123])
```

[338]:
```
plt.title('Salary Distribution Plot')
sns.histplot(test_predictions)
plt.show()
```



[339]:
```
submission_df = pd.DataFrame({
    'Id': df_test_cleaned['Id'],
    'SalePrice': test_predictions
})

# Save the predictions to a CSV file
submission_df.to_csv('submission.csv', index=False)
```

7 Conclusion

The current linear regression model provides a basic understanding of house price predictions, but the relatively high error metrics indicate room for improvement.

[ ]: