```
import pandas as pd
import numpy as np
import seaborn as sns
import matplotlib.pyplot as plt
import matplotlib as mpl
from mpl_toolkits.mplot3d import Axes3D
from sklearn.linear_model import LinearRegression
from sklearn.tree import DecisionTreeRegressor
from sklearn.ensemble import RandomForestRegressor
from sklearn.preprocessing import StandardScaler,QuantileTransformer
from sklearn.model_selection import train_test_split
from sklearn.metrics import mean_squared_error, r2_score

%matplotlib inline
```

Working on Train dataframe

```
from google.colab import files
uploaded = files.upload()
```


```
import os
os.listdir('/content/')
```

```
['.config',
 'drive',
 'house prices prediction (1).zip',
 'house prices prediction.zip',
 'house prices prediction - 1.zip',
 'sample_data']
```

```
traindf = pd.read_csv('/content/house prices prediction - 1.zip')
traindf.head()
```

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

5 rows × 81 columns

```
traindf.columns
```

```
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')
```

```
numeric_df = traindf.select_dtypes(include='number')
correlation_matrix = numeric_df.corr()
correlation_matrix['SalePrice'].sort_values(ascending = False)
```

```
SalePrice        1.000000
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
```

```
req_tr = ["GarageArea","OverallQual","TotalBsmtSF","1stFlrSF","2ndFlrSF","LowQualFinSF","GrLivArea","BsmtFullBath","BsmtHalfBath","FullI
selected_tr = traindf[req_tr]
```

```
selected_tr.loc[:, 'TotalBath'] = (selected_tr['BsmtFullBath'].fillna(0) +
                                   selected_tr['BsmtHalfBath'].fillna(0) +
                                   selected_tr['FullBath'].fillna(0) +
                                   selected_tr['HalfBath'].fillna(0))

selected_tr.loc[:, 'TotalSF'] = (selected_tr['TotalBsmtSF'].fillna(0) +
                                 selected_tr['1stFlrSF'].fillna(0) +
                                 selected_tr['2ndFlrSF'].fillna(0) +
                                 selected_tr['LowQualFinSF'].fillna(0) +
                                 selected_tr['GrLivArea'].fillna(0))
```

```
<ipython-input-32-530500eb8649>:1: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-view-versus
  selected_tr.loc[:, 'TotalBath'] = (selected_tr['BsmtFullBath'].fillna(0) +
<ipython-input-32-530500eb8649>:6: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-view-versus
  selected_tr.loc[:, 'TotalSF'] = (selected_tr['TotalBsmtSF'].fillna(0) +
```

```
selected_tr.head()
```

	GarageArea	OverallQual	TotalBsmtSF	1stFlrSF	2ndFlrSF	LowQualFinSF	GrLivArea
0	548	7	856	856	854	0	1710
1	460	6	1262	1262	0	0	1262
2	608	7	920	920	866	0	1786
3	642	7	756	961	756	0	1717
4	836	8	1145	1145	1053	0	2198

Next steps: Generate code with `selected_tr` View recommended plots

Keeping only the necessary columns

```
train_df = selected_tr[['TotRmsAbvGrd','TotalBath','GarageArea','TotalSF','OverallQual','SalePrice']]

train_df.head()
```

	TotRmsAbvGrd	TotalBath	GarageArea	TotalSF	OverallQual	SalePrice
0	8	4	548	4276	7	208500
1	6	3	460	3786	6	181500
2	6	4	608	4492	7	223500
3	7	2	642	4190	7	140000
4	9	4	836	5541	8	250000

Next steps: Generate code with `train_df` View recommended plots

Splitting the dataset and Creating Pipeline

```
from sklearn.model_selection import train_test_split
train_set,test_set =train_test_split(train_df,test_size = 0.2,random_state = 42)
print(f"Rows in train set: {len(train_set)}\nRows in test set:{len(test_set)}\n")
```

```
Rows in train set: 1168
Rows in test set:292
```

```
housing = train_set.drop("SalePrice",axis=1)
housing_labels = train_set["SalePrice"].copy()
```

```
from sklearn.impute import SimpleImputer
from sklearn.pipeline import Pipeline
from sklearn.preprocessing import StandardScaler
my_pipeline = Pipeline([
    ('imputer',SimpleImputer(strategy="median")),
    ('std_scaler',StandardScaler())
])
```

```
X_train = my_pipeline.fit_transform(housing)

X_train
```

```
array([[-0.96456591, -0.48377079, -0.86383727, -0.13352109, -0.82044456],
       [ 0.27075534,  0.61127627, -0.45626397, -0.13428593, -0.08893368],
       [-1.58222654, -1.57881784, -2.25716927, -1.32207838, -0.82044456],
       ...,
       [-0.96456591, -0.48377079,  0.45366713, -1.16605156, -0.82044456],
       [ 0.27075534, -0.48377079, -1.23349678, -0.26966215,  0.64257719],
       [ 0.27075534, -0.48377079,  0.87071888,  0.28025593,  0.64257719]])
```

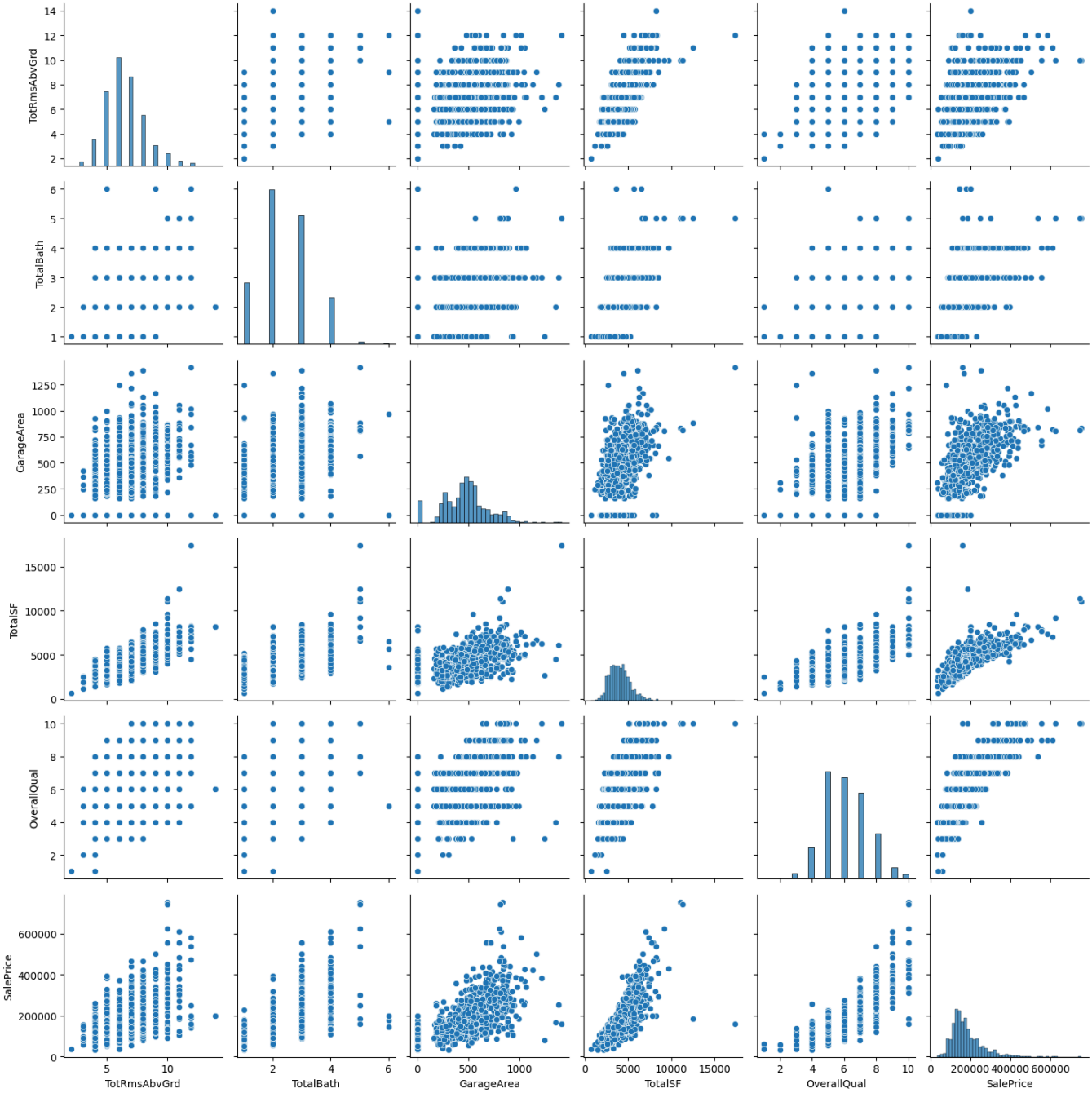
```
Y_train = housing_labels

Y_train.head()
```

```
254     145000
1066    178000
638      85000
799     175000
380     127000
Name: SalePrice, dtype: int64
```

Correlations

```
plt.figure(figsize=(5,5))
import warnings
warnings.filterwarnings("ignore", category=UserWarning)
%matplotlib inline
sns.pairplot(train_df)
plt.tight_layout()
plt.show()
```

<Figure size 500x500 with 0 Axes>



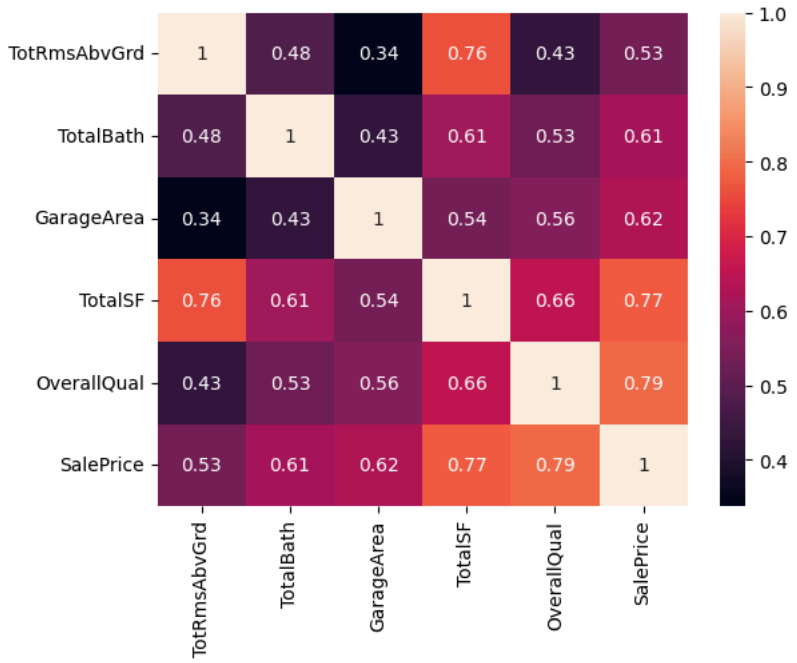

```
corr_matrix = train_df.corr()
corr_matrix['SalePrice'].sort_values(ascending = False)
```

```
SalePrice       1.000000
OverallQual     0.790982
TotalSF         0.773909
```

```
GarageArea      0.623431
TotalBath       0.613005
TotRmsAbvGrd    0.533723
Name: SalePrice, dtype: float64
```

```
sns.heatmap(train_df.corr(),annot = True)
```

```
<Axes: >
```



Working with Test Dataframe

```
from google.colab import files
uploaded = files.upload()
```

```
Choose Files house price...iction - 2.zip
• house prices prediction - 2.zip(application/x-zip-compressed) - 87192 bytes, last modified: 12/7/2024 - 100% done
Saving house prices prediction - 2.zip to house prices prediction - 2.zip
```

```
import os
os.listdir('/content/')
```

```
['.config',
 'drive',
 'house prices prediction (1).zip',
 'house prices prediction.zip',
 'house prices prediction - 2.zip',
 'house prices prediction - 1.zip',
 'sample_data']
```

```
testdf = pd.read_csv('/content/house prices prediction - 2.zip')
```

```
testdf.head()
```

	Id	MSSubClass	MSZoning	LotFrontage	LotArea	Street	Alley	LotShape	LandContour	Utilities	...	ScreenPorch	PoolArea	Poo
0	1461	20	RH	80.0	11622	Pave	NaN	Reg	Lvl	AllPub	...	120	0	I
1	1462	20	RL	81.0	14267	Pave	NaN	IR1	Lvl	AllPub	...	0	0	I
2	1463	60	RL	74.0	13830	Pave	NaN	IR1	Lvl	AllPub	...	0	0	I
3	1464	60	RL	78.0	9978	Pave	NaN	IR1	Lvl	AllPub	...	0	0	I
4	1465	120	RL	43.0	5005	Pave	NaN	IR1	HLS	AllPub	...	144	0	I

5 rows × 80 columns

```
req_tst = ["GarageArea","OverallQual","TotalBsmtSF","1stFlrSF","2ndFlrSF","LowQualFinSF","GrLivArea","BsmtFullBath","BsmtHalfBath","Ful
```

```
selected_tst = testdf[req_tst]
```

```
selected_tst.loc[:, 'TotalBath'] = (selected_tst['BsmtFullBath'].fillna(0) +
                                    selected_tst['BsmtHalfBath'].fillna(0) +
                                    selected_tst['FullBath'].fillna(0) +
                                    selected_tst['HalfBath'].fillna(0))

selected_tst.loc[:, 'TotalSF'] = (selected_tst['TotalBsmtSF'].fillna(0) +
                                  selected_tst['1stFlrSF'].fillna(0) +
                                  selected_tst['2ndFlrSF'].fillna(0) +
                                  selected_tst['LowQualFinSF'].fillna(0) +
                                  selected_tst['GrLivArea'].fillna(0))
```

```
<ipython-input-51-88b69dcf1189>:1: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-view-versus
  selected_tst.loc[:, 'TotalBath'] = (selected_tst['BsmtFullBath'].fillna(0) +
<ipython-input-51-88b69dcf1189>:6: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-view-versus
  selected_tst.loc[:, 'TotalSF'] = (selected_tst['TotalBsmtSF'].fillna(0) +
```

```
selected_tst.head()
```

	GarageArea	OverallQual	TotalBsmtSF	1stFlrSF	2ndFlrSF	LowQualFinSF	GrLivArea
0	730.0	5	882.0	896	0	0	896
1	312.0	6	1329.0	1329	0	0	1329
2	482.0	5	928.0	928	701	0	1629
3	470.0	6	926.0	926	678	0	1604
4	506.0	8	1280.0	1280	0	0	1280

Next steps: Generate code with selected_tst | View recommended plots

```
test_df_unproc = selected_tst[['TotRmsAbvGrd','TotalBath','GarageArea','TotalSF','OverallQual']]

test_df_unproc.head()
```

	TotRmsAbvGrd	TotalBath	GarageArea	TotalSF	OverallQual
0	5	1.0	730.0	2674.0	5
1	6	2.0	312.0	3987.0	6
2	6	3.0	482.0	4186.0	5
3	7	3.0	470.0	4134.0	6
4	5	2.0	506.0	3840.0	8

Next steps: Generate code with test_df_unproc | View recommended plots

```
test_df = test_df_unproc.fillna(test_df_unproc.mean())


x_test = my_pipeline.transform(test_df[['TotRmsAbvGrd','TotalBath','GarageArea','TotalSF','OverallQual']].values)

x_test
```

```
array([[-0.96456591, -1.57881784,  1.2024646 , -1.10333489, -0.82044456],
       [-0.34690528, -0.48377079, -0.77853123, -0.09910341, -0.08893368],
       [-0.34690528,  0.61127627,  0.02713693,  0.05309923, -0.82044456],
       ...,
       [ 0.27075534, -0.48377079,  0.47262403, -0.34002719, -0.82044456],
       [-0.34690528, -0.48377079, -2.25716927, -0.96719384, -0.82044456],
       [ 1.50607659,  0.61127627,  0.82332664,  0.67261751,  0.64257719]])
```

Model Selection

```
model = LinearRegression()
#model = DecisionTreeRegressor()
#model = RandomForestRegressor()
model.fit(X_train,Y_train)
```

```
▾ LinearRegression
LinearRegression()
```

```
y_train_pred = model.predict(X_train)
```

```
y_train_pred[:5]
```

```
array([137667.57956441, 175034.19941421,  79632.65207176, 149652.98819495,
       146674.06395314])
```

```
some_data = housing.iloc[:5]
some_labels = housing_labels.iloc[:5]
```

```
proc_data = my_pipeline.transform(some_data)
```

```
model.predict(proc_data)
```

```
array([137667.57956441, 175034.19941421,  79632.65207176, 149652.98819495,
       146674.06395314])
```

```
list(some_labels)
```

```
[145000, 178000, 85000, 175000, 127000]
```

```
train_mse = mean_squared_error(Y_train,y_train_pred)
```

```
train_rmse = np.sqrt(train_mse)
```

```
print(f"Training MSE: {train_mse:.2f}, Training RMSE: {train_rmse:.2f}")
```

```
Training MSE: 1460715662.52, Training RMSE: 38219.31
```

Cross - Validation

```
from sklearn.model_selection import cross_val_score
scores = cross_val_score(model,X_train,Y_train,scoring="neg_mean_squared_error",cv = 200)
rmse_scores = np.sqrt(-scores)
rmse_scores
```

```
array([ 16599.40639047,  18325.22018653,  29307.41492256,  30900.62181786,
        42713.28158503,  13517.92728875,  28466.43684333,  26159.1789478 ,
        16694.35747291,  44185.04242239,  14432.4514348 ,  29621.4888739 ,
        15288.08646946,  20343.83312791,  26928.48898473,  30286.90600734,
        27591.27659228,  49704.26333128,  48966.56719948,  24506.05029869,
        51513.14668133,  16136.46039114,  18384.50681523,  28433.24926957,
        55483.42476232,  13885.14122222,  40018.30732342,  22655.21809999,
       122894.24572375,  36980.15773466,  16527.48051189,  26495.91857289,
        40892.94110057,  26980.63750536,  67733.44140093,  13173.00917782,
        17673.54848644,  28785.9566716 ,  34551.83798472,  37734.44853911,
        23765.3506236 ,  19538.738731  ,  29343.3959128 ,  32180.98454967,
        35319.05373078,  38228.16511844,  26372.59349178,  28064.67081093,
        36009.27109903,  26029.75222025,  22669.10866239,  37583.20518266,
        27672.46756544,  26860.18618909,  25473.16521999,  21242.95056337,
        20053.45561874,  30889.23473073,  52117.08856944,  36544.31900032,
       133524.62025163,  42636.85773552,  33816.59478726,  44585.1371374 ,
        42323.63535859,  21634.77565968,  41502.66165963,  30636.54557842,
        26746.16642826,  15309.84320432,  54226.73045149,  33988.79845478,
        47823.05327476,  25585.65094945,  45620.12007825,  49937.324193  ,
        52677.22772022,  17932.7405338 ,  39969.26806934,  31599.91071606,
        17807.77621148,  35954.46526772, 222456.2247121 ,  29094.22010999,
        41529.46627467,  27720.37789678,  19879.42245214,  19675.41805076,
        27383.1292489 , 121547.27719409,  26427.98421929,  26079.76589143,
        24510.70593577,  26086.37349027,  21568.58923443,  20429.16535944,
        18474.26446438,  21666.68595951,  94822.5377186 ,  32141.09120185,
         8609.78735922,  41595.17941681,  33000.48070685,  58430.66413254,
        24607.56247791,  19281.14045192,  48953.28952053,  21346.29508891,
        33273.25058659,  21946.70716609,  23678.86240679,  20746.75417475,
        21033.03402566,  23193.73512545,  45194.85383363,  36114.70313285,
        63940.04213846,  24185.33375536,  49971.70176585,  23553.19770034,
        28356.19941834,  31601.84917239,  19743.02818178,  17925.67263047,
        34345.32569996,  38262.53910803,  23418.1606015 ,  64217.99081748,
        41955.49432203,  13247.58126398,  18706.70154528,  14929.35502952,
        22986.02338424,  26476.05190717,  29223.71986739,  36860.45324045,
        53228.19467845,  26809.73400444,  17945.13852327,  13424.19020028,
        38847.92761809,  18784.0050538 ,  23201.64372092,  26436.65261068,
        34499.10132104,  22410.38113845,  44522.02672058,  35721.51276097,
        23316.23467097,  19451.35764774,  16880.77324834,  26453.2948389 ,
        19300.88970697,  52027.66249345,  39417.73705746,  25123.5414658 ,
        27711.68996535,  23370.47417554,  23421.00277133,  19624.96789495,
```

```
            53322.45020398,  21718.09001705,  32640.1271753 ,  19244.51215297,
            46049.03252092,  16703.9195712 ,  19728.41459881,  66589.43304565,
            23503.5915548 ,  40376.68920578,  21942.97964157,  26392.09561029,
            15193.36709344,  19879.70112264,  18899.08624785,  15821.45950323,
            23143.73085732,  16814.75337692,  45032.51530946,   9162.69902713,
            34188.77514645,  18846.8667917 ,   9098.06239376,  15607.68522491,
            25658.10246257,  50905.34291163,  28106.85449056,  10840.69989811,
            19224.05599128,  29592.35328826,  23886.98660851,  41962.2412104 ,
            23032.83632699,  17362.50355856,  10380.08339328,  25285.63428966,
            99149.81938064,  16461.75020861,  33451.89182822,  23785.07005972])


def print_scores(scores):
    print("Scores:",scores)
    print("Mean:",scores.mean())
    print("Standard Deviation",scores.std())

print_scores(rmse_scores)
```

```
Scores: [ 16599.40639047  18325.22018653  29307.41492256  30900.62181786
  42713.28158503  13517.92728875  28466.43684333  26159.1789478
  16694.35747291  44185.04242239  14432.4514348   29621.4888739
  15288.08646946  20343.83312791  26928.48898473  30286.90600734
  27591.27659228  49704.26333128  48966.56719948  24506.05029869
  51513.14668133  16136.46039114  18384.50681523  28433.24926957
  55483.42476232  13885.14122222  40018.30732342  22655.21809999
 122894.24572375  36980.15773466  16527.48051189  26495.91857289
  40892.94110057  26980.63750536  67733.44140093  13173.00917782
  17673.54848644  28785.9566716   34551.83798472  37734.44853911
  23765.3506236   19538.738731    29343.3959128   32180.98454967
  35319.05373078  38228.16511844  26372.59349178  28064.67081093
  36009.27109903  26029.75222025  22669.10866239  37583.20518266
  27672.46756544  26860.18618909  25473.16521999  21242.95056337
  20053.45561874  30889.23473073  52117.08856944  36544.31900032
 133524.62025163  42636.85773552  33816.59478726  44585.1371374
  42323.63535859  21634.77565968  41502.66165963  30636.54557842
  26746.16642826  15309.84320432  54226.73045149  33988.79845478
  47823.05327476  25585.65094945  45620.12007825  49937.324193
  52677.22772022  17932.7405338   39969.26806934  31599.91071606
  17807.77621148  35954.46526772 222456.2247121   29094.22010999
  41529.46627467  27720.37789678  19879.42245214  19675.41805076
  27383.1292489  121547.27719409  26427.98421929  26079.76589143
  24510.70593577  26086.37349027  21568.58923443  20429.16535944
  18474.26446438  21666.68595951  94822.5377186   32141.09120185
   8609.78735922  41595.17941681  33000.48070685  58430.66413254
  24607.56247791  19281.14045192  48953.28952053  21346.29508891
  33273.25058659  21946.70716609  23678.86240679  20746.75417475
  21033.03402566  23193.73512545  45194.85383363  36114.70313285
  63940.04213846  24185.33375536  49971.70176585  23553.19770034
  28356.19941834  31601.84917239  19743.02818178  17925.67263047
  34345.32569996  38262.53910803  23418.1606015   64217.99081748
  41955.49432203  13247.58126398  18706.70154528  14929.35502952
  22986.02338424  26476.05190717  29223.71986739  36860.45324045
  53228.19467845  26809.73400444  17945.13852327  13424.19020028
  38847.92761809  18784.0050538   23201.64372092  26436.65261068
  34499.10132104  22410.38113845  44522.02672058  35721.51276097
  23316.23467097  19451.35764774  16880.77324834  26453.2948389
  19300.88970697  52027.66249345  39417.73705746  25123.5414658
  27711.68996535  23370.47417554  23421.00277133  19624.96789495
  53322.45020398  21718.09001705  32640.1271753   19244.51215297
  46049.03252092  16703.9195712   19728.41459881  66589.43304565
  23503.5915548   40376.68920578  21942.97964157  26392.09561029
  15193.36709344  19879.70112264  18899.08624785  15821.45950323
  23143.73085732  16814.75337692  45032.51530946   9162.69902713
  34188.77514645  18846.8667917    9098.06239376  15607.68522491
  25658.10246257  50905.34291163  28106.85449056  10840.69989811
  19224.05599128  29592.35328826  23886.98660851  41962.2412104
  23032.83632699  17362.50355856  10380.08339328  25285.63428966
  99149.81938064  16461.75020861  33451.89182822  23785.07005972]
Mean: 32302.951288213964
Standard Deviation 22517.159521038964
```

```
y_pred=model.predict(x_test)

y_pred
```

```
array([126103.27519237, 162326.84508263, 163127.91620663, ...,
       145956.58005073,  97528.78037618, 233363.14624566])
```

```
from google.colab import files
uploaded = files.upload()
```

```
Choose Files house price…iction - 3.zip
```

- **house prices prediction - 3.zip**(application/x-zip-compressed) - 15808 bytes, last modified: 13/7/2024 - 100% done

```
Saving house prices prediction - 3.zip to house prices prediction - 3 (1).zip
```

```
import os
os.listdir('/content/')
```

```
['.config',
 'drive',
 'house prices prediction (1).zip',
 'house prices prediction.zip',
 'house prices prediction - 2.zip',
 'house prices prediction - 1.zip',
 'house prices prediction - 3.zip',
 'sample_data']
```

```
pred=pd.DataFrame(y_pred)
sub_df=pd.read_csv('/content/house prices prediction - 3.zip')
datasets=pd.concat([sub_df['Id'],pred],axis=1)
datasets.columns=['Id','SalePrice']
datasets.to_csv('sample_submission.csv',index=False)
```