+ Code + Markdown

```
# This Python 3 environment comes with many helpful analytics libraries installe
# It is defined by the kaggle/python docker image: https://github.com/kaggle/doc
# For example, here's several helpful packages to load in

import numpy as np # linear algebra
import pandas as pd # data processing, CSV file I/O (e.g. pd.read_csv)

# Input data files are available in the "../input/" directory.
# For example, running this (by clicking run or pressing Shift+Enter) will list

import os
for dirname, _, filenames in os.walk('/kaggle/input'):
    for filename in filenames:
        print(os.path.join(dirname, filename))

# Any results you write to the current directory are saved as output.
```

```
/kaggle/input/house-prices-advanced-regression-techniques/data_description.txt
/kaggle/input/house-prices-advanced-regression-techniques/test.csv
/kaggle/input/house-prices-advanced-regression-techniques/train.csv
/kaggle/input/house-prices-advanced-regression-techniques/sample_submission.csv
```

+ Code + Markdown

In[2]:

```
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
import numpy as np
%matplotlib inline
sns.set_style('whitegrid')
train_data = pd.read_csv("/kaggle/input/house-prices-advanced-regression-techni
test_data = pd.read_csv("/kaggle/input/house-prices-advanced-regression-techniq
```

In[3]:
```
train_data.head()
```

Out[3]:

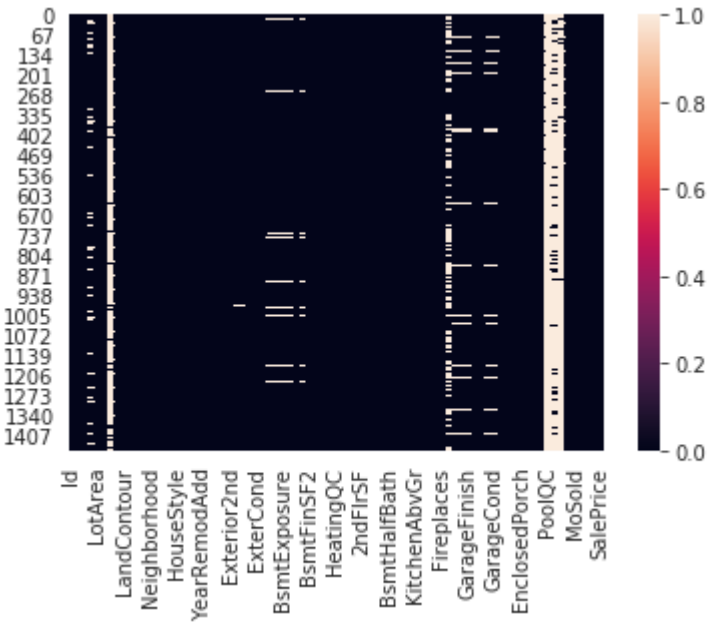
	Id	MSSubClass	MSZoning	LotFrontage	LotArea	Street	Alley	LotShape	LandContour	Ut
0	1	60	RL	65.0	8450	Pave	NaN	Reg	Lvl	A
1	2	20	RL	80.0	9600	Pave	NaN	Reg	Lvl	A
2	3	60	RL	68.0	11250	Pave	NaN	IR1	Lvl	A
3	4	70	RL	60.0	9550	Pave	NaN	IR1	Lvl	A
4	5	60	RL	84.0	14260	Pave	NaN	IR1	Lvl	A

5 rows × 81 columns

In[4]:
```
#train_data.info()
sns.heatmap(train_data.isnull())
```

Out[4]:

```
<matplotlib.axes._subplots.AxesSubplot at 0x7fb22c82bf60>
```

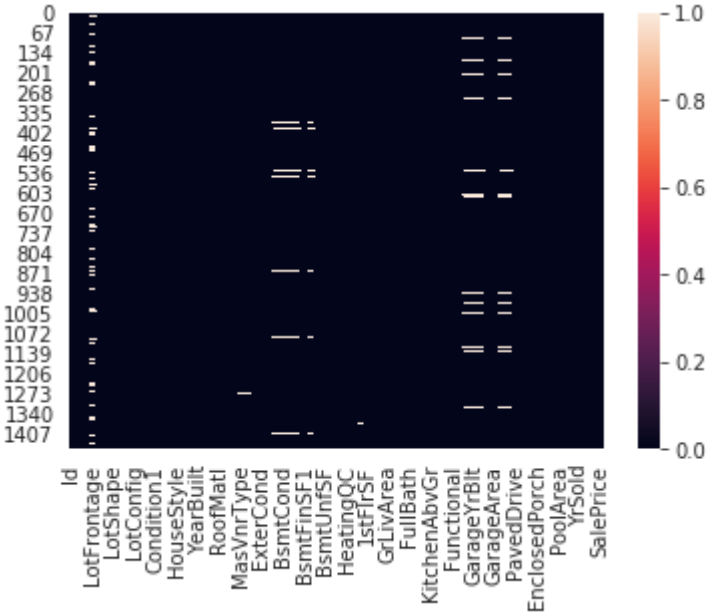


In[5]:
```
train_data= train_data.drop(['Alley','PoolQC','FireplaceQu','Fence','MiscFeature
```

In[6]:
```
sns.heatmap(train_data.iloc[:,:].isnull())
```

Out[6]:

```
<matplotlib.axes._subplots.AxesSubplot at 0x7fb22a747198>
```



In[7]:

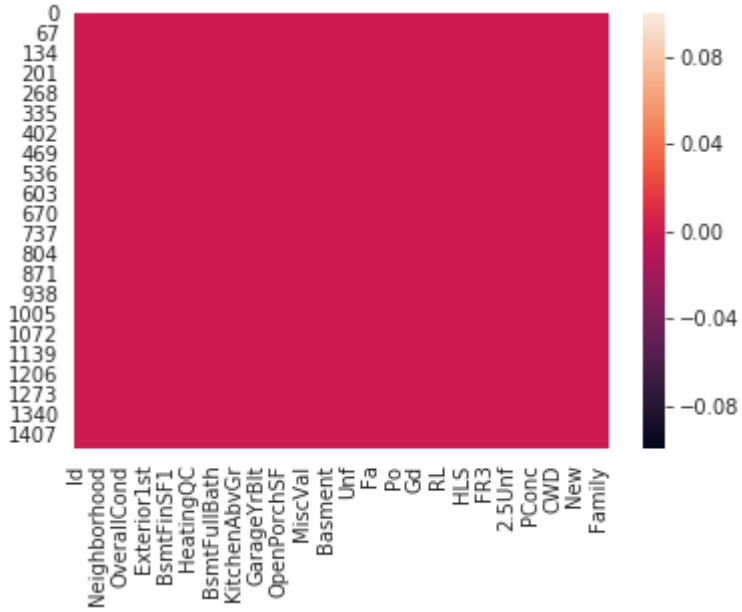
```
train_data['LotFrontage']= train_data[['LotFrontage']].fillna(value=train_data[

garageType=pd.get_dummies(train_data['GarageType'], drop_first=True)
train_data.drop(['GarageType'], axis=1,inplace=True)
train_data=pd.concat([train_data,garageType], axis=1)

train_data['GarageYrBlt']= train_data[['GarageYrBlt']].fillna(value=train_data[

GarageFinish=pd.get_dummies(train_data['GarageFinish'], drop_first=True)
train_data.drop(['GarageFinish'], axis=1,inplace=True)
train_data=pd.concat([train_data,GarageFinish], axis=1)

Electrical=pd.get_dummies(train_data['Electrical'], drop_first=True)
train_data.drop(['Electrical'], axis=1,inplace=True)
train_data=pd.concat([train_data,Electrical], axis=1)

train_data['BsmtFinSF2']= train_data[['BsmtFinSF2']].fillna(value=train_data['Bs

BsmtQual=pd.get_dummies(train_data['BsmtQual'], drop_first=True)
train_data.drop(['BsmtQual'], axis=1,inplace=True)
train_data=pd.concat([train_data,BsmtQual], axis=1)

train_data.drop(['GarageCond'], axis=1,inplace=True)

GarageQual=pd.get_dummies(train_data['GarageQual'], drop_first=True)
train_data.drop(['GarageQual'], axis=1,inplace=True)
train_data=pd.concat([train_data,GarageQual], axis=1)

train_data['MasVnrArea']= train_data[['MasVnrArea']].fillna(value=train_data['Ma

MasVnrType=pd.get_dummies(train_data['MasVnrType'], drop_first=True)
train_data.drop(['MasVnrType'], axis=1,inplace=True)
train_data=pd.concat([train_data,MasVnrType], axis=1)

train_data.drop(['BsmtCond','BsmtFinType1','BsmtFinType2'], axis=1,inplace=True)
test_data['BsmtFullBath']= test_data[['BsmtFullBath']].fillna(value=test_data['B
test_data['BsmtHalfBath']= test_data[['BsmtHalfBath']].fillna(value=test_data['B

BsmtExposure=pd.get_dummies(train_data['BsmtExposure'], drop_first=True)
train_data.drop(['BsmtExposure'], axis=1,inplace=True)
train_data=pd.concat([train_data,BsmtExposure], axis=1)

MSZoning=pd.get_dummies(train_data['MSZoning'], drop_first=True)
train_data.drop(['MSZoning'], axis=1,inplace=True)
train_data=pd.concat([train_data,MSZoning], axis=1)

train_data.drop(['Street'], axis=1,inplace=True)

LotShape=pd.get_dummies(train_data['LotShape'], drop_first=True)
train_data.drop(['LotShape'], axis=1,inplace=True)
train_data=pd.concat([train_data,LotShape], axis=1)

LandContour=pd.get_dummies(train_data['LandContour'], drop_first=True)
train_data.drop(['LandContour'], axis=1,inplace=True)
```

```
train_data=pd.concat([train_data,LandContour], axis=1)

train_data.drop(['Utilities'], axis=1,inplace=True)

LotConfig=pd.get_dummies(train_data['LotConfig'], drop_first=True)
train_data.drop(['LotConfig'], axis=1,inplace=True)
train_data=pd.concat([train_data,LotConfig], axis=1)

HouseStyle=pd.get_dummies(train_data['HouseStyle'], drop_first=True)
train_data.drop(['HouseStyle'], axis=1,inplace=True)
train_data=pd.concat([train_data,HouseStyle], axis=1)

Foundation=pd.get_dummies(train_data['Foundation'], drop_first=True)
train_data.drop(['Foundation'], axis=1,inplace=True)
train_data=pd.concat([train_data,Foundation], axis=1)

CentralAir=pd.get_dummies(train_data['CentralAir'], drop_first=True)
train_data.drop(['CentralAir'], axis=1,inplace=True)
train_data=pd.concat([train_data,CentralAir], axis=1)

SaleType=pd.get_dummies(train_data['SaleType'], drop_first=True)
train_data.drop(['SaleType'], axis=1,inplace=True)
train_data=pd.concat([train_data,SaleType], axis=1)

SaleCondition=pd.get_dummies(train_data['SaleCondition'], drop_first=True)
train_data.drop(['SaleCondition'], axis=1,inplace=True)
train_data=pd.concat([train_data,SaleCondition], axis=1)
```

In[8]:
```
sns.heatmap(train_data.iloc[:,:].isnull())
```

Out[8]:

```
<matplotlib.axes._subplots.AxesSubplot at 0x7fb22a651908>
```



In[9]:

```
from sklearn.model_selection import train_test_split
X=train_data.drop(['SalePrice','Id','LandSlope',
                   'Neighborhood','Condition1','Condition2','BldgType','RoofSty
                   'Exterior1st','Exterior2nd','ExterQual','ExterCond','Heating
                   'KitchenQual','Functional','PavedDrive','Mix','IR2','IR3','Re
y=train_data['SalePrice']
X_train, X_test, y_train, y_test = train_test_split(X,y, test_size=0.30,random_s

X
# print(y)
```

Out[9]:

	MSSubClass	LotFrontage	LotArea	OverallQual	OverallCond	YearBuilt	YearRemodAdd
0	60	65.0	8450	7	5	2003	2003
1	20	80.0	9600	6	8	1976	1976
2	60	68.0	11250	7	5	2001	2002
3	70	60.0	9550	7	5	1915	1970
4	60	84.0	14260	8	5	2000	2000
...	...	...	...	...	...	...	...
1455	60	62.0	7917	6	5	1999	2000
1456	20	85.0	13175	6	6	1978	1988
1457	70	66.0	9042	7	9	1941	2006
1458	20	68.0	9717	5	6	1950	1996
1459	20	75.0	9937	5	6	1965	1965

1460 rows × 89 columns

In[10]:

```
from sklearn.ensemble import RandomForestClassifier
from sklearn.metrics import mean_absolute_error

rf = RandomForestClassifier(n_estimators=800)
rf.fit(X_train,y_train)
predict_with_rf=rf.predict(X_test)

rf_val_mae = mean_absolute_error(predict_with_rf,y_test)
print(rf_val_mae)
X_test
```

```
22998.851598173515
```

Out[10]:

	MSSubClass	LotFrontage	LotArea	OverallQual	OverallCond	YearBuilt	YearRemodAdd
345	50	65.000000	6435	6	5	1939	1950
47	20	84.000000	11096	8	5	2006	2006
180	160	70.049958	2117	6	5	2000	2000
1346	20	70.049958	20781	7	7	1968	2003
1183	30	60.000000	10800	5	6	1920	1950
...	...	...	...	...	...	...	...
334	60	59.000000	9042	6	5	1998	1998
1342	60	70.049958	9375	8	5	2002	2002
1313	60	108.000000	14774	9	5	1999	1999
549	60	75.000000	9375	7	5	2003	2004
190	70	70.000000	10570	8	8	1932	1994

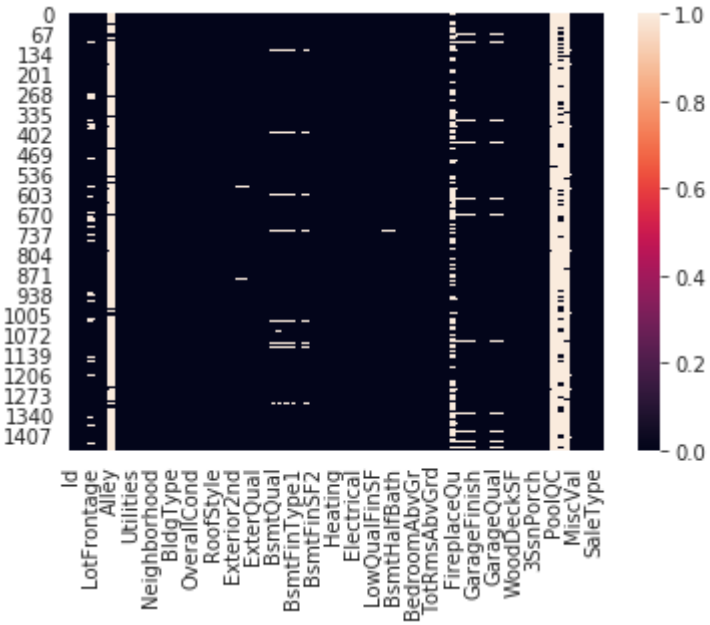
438 rows × 89 columns

In[11]:

```
test_data = pd.read_csv("/kaggle/input/house-prices-advanced-regression-techniqu
sns.heatmap(test_data.iloc[:,:].isnull())
```

Out[11]:

```
<matplotlib.axes._subplots.AxesSubplot at 0x7fb22c8f5ac8>
```



In[12]:
```
test_data
```

Out[12]:

	Id	MSSubClass	MSZoning	LotFrontage	LotArea	Street	Alley	LotShape	LandContou
0	1461	20	RH	80.0	11622	Pave	NaN	Reg	L
1	1462	20	RL	81.0	14267	Pave	NaN	IR1	L
2	1463	60	RL	74.0	13830	Pave	NaN	IR1	L
3	1464	60	RL	78.0	9978	Pave	NaN	IR1	L
4	1465	120	RL	43.0	5005	Pave	NaN	IR1	HL
...	...	...	...	...	...	...	...	...	
1454	2915	160	RM	21.0	1936	Pave	NaN	Reg	L
1455	2916	160	RM	21.0	1894	Pave	NaN	Reg	L
1456	2917	20	RL	160.0	20000	Pave	NaN	Reg	L
1457	2918	85	RL	62.0	10441	Pave	NaN	Reg	L
1458	2919	60	RL	74.0	9627	Pave	NaN	Reg	L

1459 rows × 80 columns

In[13]:
```
test_data= test_data.drop(['Alley','PoolQC','FireplaceQu','Fence','MiscFeature'
```

In[14]:

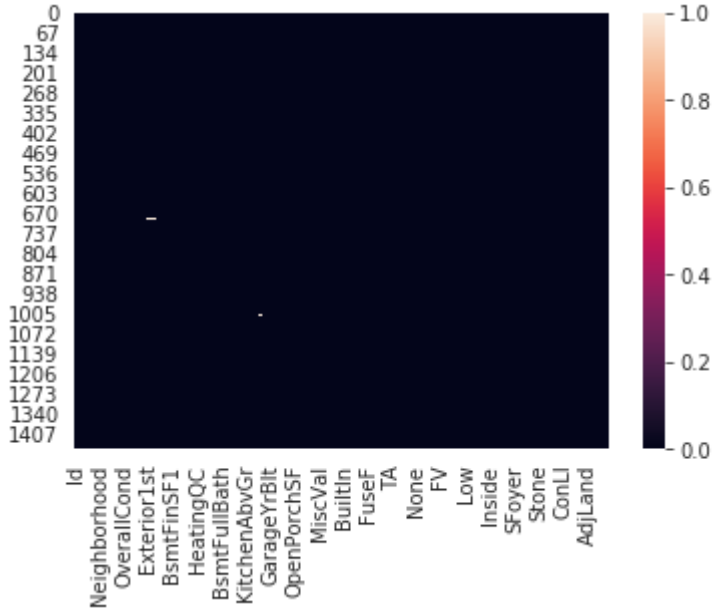
```
test_data['LotFrontage']= test_data[['LotFrontage']].fillna(value=test_data['Lot

garageType=pd.get_dummies(test_data['GarageType'], drop_first=True)
test_data.drop(['GarageType'], axis=1,inplace=True)
test_data=pd.concat([test_data,garageType], axis=1)

test_data['GarageYrBlt']= test_data[['GarageYrBlt']].fillna(value=test_data['Gar

GarageFinish=pd.get_dummies(test_data['GarageFinish'], drop_first=True)
test_data.drop(['GarageFinish'], axis=1,inplace=True)
test_data=pd.concat([test_data,GarageFinish], axis=1)

Electrical=pd.get_dummies(test_data['Electrical'], drop_first=True)
test_data.drop(['Electrical'], axis=1,inplace=True)
test_data=pd.concat([test_data,Electrical], axis=1)

test_data['BsmtFinSF2']= test_data[['BsmtFinSF2']].fillna(value=test_data['BsmtF

BsmtQual=pd.get_dummies(test_data['BsmtQual'], drop_first=True)
test_data.drop(['BsmtQual'], axis=1,inplace=True)
test_data=pd.concat([test_data,BsmtQual], axis=1)

test_data.drop(['GarageCond'], axis=1,inplace=True)

GarageQual=pd.get_dummies(test_data['GarageQual'], drop_first=True)
test_data.drop(['GarageQual'], axis=1,inplace=True)
test_data=pd.concat([test_data,GarageQual], axis=1)

test_data['MasVnrArea']= test_data[['MasVnrArea']].fillna(value=test_data['MasVn
test_data['BsmtFullBath']= test_data[['BsmtFullBath']].fillna(value=test_data['B
test_data['BsmtHalfBath']= test_data[['BsmtHalfBath']].fillna(value=test_data['B

MasVnrType=pd.get_dummies(test_data['MasVnrType'], drop_first=True)
test_data.drop(['MasVnrType'], axis=1,inplace=True)
test_data=pd.concat([test_data,MasVnrType], axis=1)

test_data.drop(['BsmtCond','BsmtFinType1','BsmtFinType2'], axis=1,inplace=True)

BsmtExposure=pd.get_dummies(test_data['BsmtExposure'], drop_first=True)
test_data.drop(['BsmtExposure'], axis=1,inplace=True)
test_data=pd.concat([test_data,BsmtExposure], axis=1)

MSZoning=pd.get_dummies(test_data['MSZoning'], drop_first=True)
test_data.drop(['MSZoning'], axis=1,inplace=True)
test_data=pd.concat([test_data,MSZoning], axis=1)

test_data.drop(['Street'], axis=1,inplace=True)

LotShape=pd.get_dummies(test_data['LotShape'], drop_first=True)
test_data.drop(['LotShape'], axis=1,inplace=True)
train_data=pd.concat([test_data,LotShape], axis=1)

LandContour=pd.get_dummies(train_data['LandContour'], drop_first=True)
test_data.drop(['LandContour'], axis=1,inplace=True)
```

```
test_data=pd.concat([test_data,LandContour], axis=1)

test_data.drop(['Utilities'], axis=1,inplace=True)

LotConfig=pd.get_dummies(test_data['LotConfig'], drop_first=True)
test_data.drop(['LotConfig'], axis=1,inplace=True)
test_data=pd.concat([test_data,LotConfig], axis=1)

HouseStyle=pd.get_dummies(test_data['HouseStyle'], drop_first=True)
test_data.drop(['HouseStyle'], axis=1,inplace=True)
test_data=pd.concat([test_data,HouseStyle], axis=1)

Foundation=pd.get_dummies(test_data['Foundation'], drop_first=True)
test_data.drop(['Foundation'], axis=1,inplace=True)
test_data=pd.concat([test_data,Foundation], axis=1)

CentralAir=pd.get_dummies(test_data['CentralAir'], drop_first=True)
test_data.drop(['CentralAir'], axis=1,inplace=True)
train_data=pd.concat([test_data,CentralAir], axis=1)

SaleType=pd.get_dummies(test_data['SaleType'], drop_first=True)
test_data.drop(['SaleType'], axis=1,inplace=True)
test_data=pd.concat([test_data,SaleType], axis=1)

SaleCondition=pd.get_dummies(test_data['SaleCondition'], drop_first=True)
test_data.drop(['SaleCondition'], axis=1,inplace=True)
test_data=pd.concat([test_data,SaleCondition], axis=1)
```

In[15]:
```
sns.heatmap(test_data.iloc[:,:].isnull())
```

Out[15]:

```
<matplotlib.axes._subplots.AxesSubplot at 0x7fb2181ee3c8>
```



In[16]:

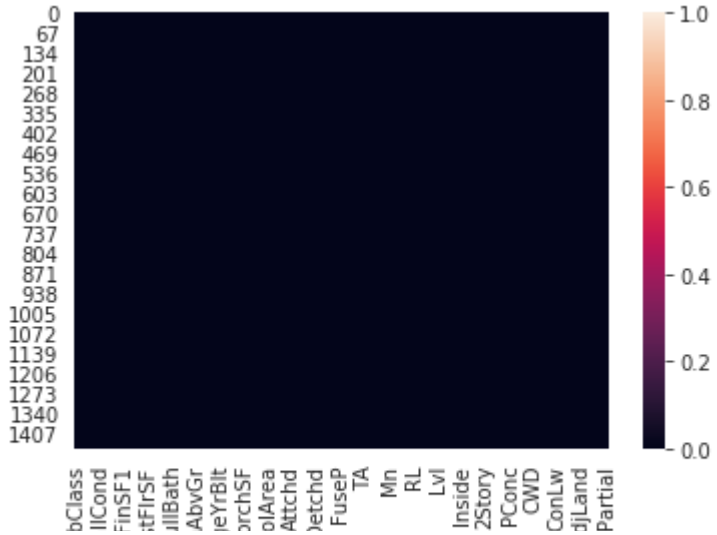
```
X_test_data=test_data.drop(['Id','LandSlope',
                    'Neighborhood','Condition1','Condition2','BldgType','RoofSty
                    'Exterior1st','Exterior2nd','ExterQual','ExterCond','Heating
                    'KitchenQual','Functional','PavedDrive','Fa','Gd'], axis=1)

sns.heatmap(X_test_data.iloc[:,:].isnull())
for column in X_test_data:
    X_test_data[column]= X_test_data[[column]].fillna(value=X_test_data[column]
X_test_data
```

```
/opt/conda/lib/python3.6/site-packages/ipykernel_launcher.py:8: SettingWithCopyWa
rning:
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: http://pandas.pydata.org/pandas-docs/stabl
e/user_guide/indexing.html#returning-a-view-versus-a-copy (http://pandas.pydata.o
rg/pandas-docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy)
```

Out[16]:

	MSSubClass	LotFrontage	LotArea	OverallQual	OverallCond	YearBuilt	YearRemodAdd
0	20	80.0	11622	5	6	1961	1961
1	20	81.0	14267	6	6	1958	1958
2	60	74.0	13830	5	5	1997	1998
3	60	78.0	9978	6	6	1998	1998
4	120	43.0	5005	8	5	1992	1992
...	...	...	...	...	...	...	...
1454	160	21.0	1936	4	7	1970	1970
1455	160	21.0	1894	4	5	1970	1970
1456	20	160.0	20000	5	7	1960	1996
1457	85	62.0	10441	5	5	1992	1992
1458	60	74.0	9627	7	5	1993	1994

1459 rows × 89 columns



In[17]:

```
X_test.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 438 entries, 345 to 190
Data columns (total 89 columns):
MSSubClass       438 non-null int64
LotFrontage      438 non-null float64
LotArea          438 non-null int64
OverallQual      438 non-null int64
OverallCond      438 non-null int64
YearBuilt        438 non-null int64
YearRemodAdd     438 non-null int64
MasVnrArea       438 non-null float64
BsmtFinSF1       438 non-null int64
BsmtFinSF2       438 non-null int64
BsmtUnfSF        438 non-null int64
TotalBsmtSF      438 non-null int64
1stFlrSF         438 non-null int64
2ndFlrSF         438 non-null int64
```

In[18]:
```
X_test_data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1459 entries, 0 to 1458
Data columns (total 89 columns):
MSSubClass       1459 non-null int64
LotFrontage      1459 non-null float64
LotArea          1459 non-null int64
OverallQual      1459 non-null int64
OverallCond      1459 non-null int64
YearBuilt        1459 non-null int64
YearRemodAdd     1459 non-null int64
MasVnrArea       1459 non-null float64
BsmtFinSF1       1459 non-null float64
BsmtFinSF2       1459 non-null float64
BsmtUnfSF        1459 non-null float64
TotalBsmtSF      1459 non-null float64
1stFlrSF         1459 non-null int64
2ndFlrSF         1459 non-null int64
```

In[19]:
```
y_pred=rf.predict(X_test_data)
# rf_val_mae = mean_absolute_error(predict_with_rf,y_test)
# print(rf_val_mae)
y_pred
```

Out[19]:

```
array([144000, 151500, 173000, ..., 168000,  93500, 290000])
```

In[20]:

```
submission= pd.DataFrame({
    'Id': test_data['Id'],
    'SalePrice': y_pred })
print(submission)
submission.to_csv("Submission_house_price.csv", index=False)
```

```
        Id  SalePrice
0     1461     144000
1     1462     151500
2     1463     173000
3     1464     181000
4     1465     153900
...    ...        ...
1454  2915      75000
1455  2916      75000
1456  2917     168000
1457  2918      93500
1458  2919     290000

[1459 rows x 2 columns]
```