In [1]:
```
import pandas as pd
import numpy as np
import seaborn as sns
import matplotlib.pyplot as plt
import warnings
warnings.filterwarnings("ignore")
```

In [2]:
```
# Define
# The problem of the project is to identify sales price for newly listed houses in t
# were provided with their corresponding sales price in the past. Here, I am tring t
# that can predict the new house sales price.
```

In [3]:
```
# Loading data
test_data = pd.read_csv("/Users/bishnupaudel/Downloads/house-prices-advanced-regress
train_data = pd.read_csv("/Users/bishnupaudel/Downloads/house-prices-advanced-regres
sample_submission = pd.read_csv("/Users/bishnupaudel/Downloads/house-prices-advanced
```

In [4]:
```
## Discover
```

In [5]:
```
test_data.head()
```

Out[5]:

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

5 rows × 80 columns

In [6]:
```
train_data.head()
```

Out[6]:

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

5 rows × 81 columns

In [7]:
```
train_data.describe()
```

Out[7]:

	Id	MSSubClass	LotFrontage	LotArea	OverallQual	OverallCond	YearBuil
count	1460.000000	1460.000000	1201.000000	1460.000000	1460.000000	1460.000000	1460.00000
mean	730.500000	56.897260	70.049958	10516.828082	6.099315	5.575342	1971.26780

	Id	MSSubClass	LotFrontage	LotArea	OverallQual	OverallCond	YearBuil
std	421.610009	42.300571	24.284752	9981.264932	1.382997	1.112799	30.20290
min	1.000000	20.000000	21.000000	1300.000000	1.000000	1.000000	1872.00000
25%	365.750000	20.000000	59.000000	7553.500000	5.000000	5.000000	1954.00000
50%	730.500000	50.000000	69.000000	9478.500000	6.000000	5.000000	1973.00000
75%	1095.250000	70.000000	80.000000	11601.500000	7.000000	6.000000	2000.00000
max	1460.000000	190.000000	313.000000	215245.000000	10.000000	9.000000	2010.00000

8 rows × 38 columns

In [8]:
```
train_data.shape
```

Out[8]:
```
(1460, 81)
```

In [9]:
```
# Checking for null values
train_data.isnull().sum()
```

Out[9]:
```
Id                0
MSSubClass        0
MSZoning          0
LotFrontage     259
LotArea           0
                 ...
MoSold            0
YrSold            0
SaleType          0
SaleCondition     0
SalePrice         0
Length: 81, dtype: int64
```

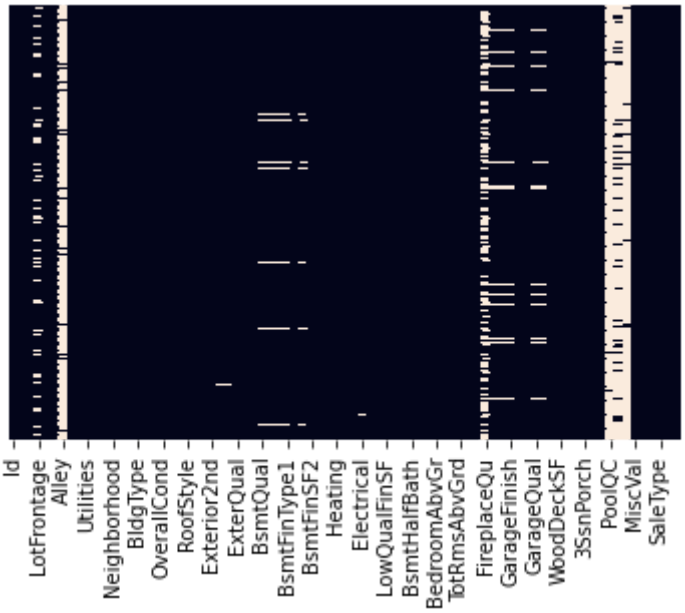
In [10]:
```
train_data.info()
```

```
<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
```

In [11]:
```
# Plotting heatmap for features that have null values
sns.heatmap(train_data.isnull(), yticklabels=False, cbar=False)
```

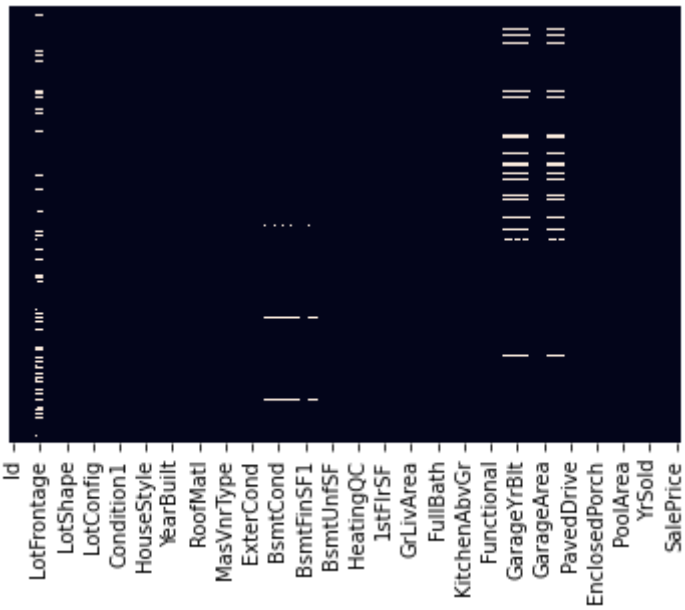
Out[11]:
```
<AxesSubplot:>
```



In [12]:
```
# Here features that have null values more than 40% are eliminated as they will affe
train = train_data.drop(['Alley', 'FireplaceQu', 'MiscFeature', 'PoolQC', 'Fence'],
test = test_data.drop(['Alley', 'FireplaceQu', 'MiscFeature', 'PoolQC', 'Fence'], ax
```

In [13]:
```
sns.heatmap(train.isnull(), yticklabels = False, cbar=False)
```

Out[13]: <AxesSubplot:>



In [14]:
```
# Freatures that have less than 5% null values are filled with most "frequent value"
train["GarageCond"] = train["GarageCond"].fillna(train["GarageCond"].mode()[0])
train["GarageQual"] = train["GarageQual"].fillna(train["GarageQual"].mode()[0])
train["GarageFinish"] = train["GarageFinish"].fillna(train["GarageFinish"].mode()[0]
train["GarageYrBlt"] = train["GarageYrBlt"].fillna(train["GarageYrBlt"].mode()[0])
train["GarageType"] = train["GarageType"].fillna(train["GarageType"].mode()[0])
train["BsmtFinType2"] = train["BsmtFinType2"].fillna(train["BsmtFinType2"].mode()[0]
train["BsmtFinType1"] = train["BsmtFinType1"].fillna(train["BsmtFinType1"].mode()[0]
train["BsmtExposure"] = train["BsmtExposure"].fillna(train["BsmtExposure"].mode()[0]
train["BsmtCond"] = train["BsmtCond"].fillna(train["BsmtCond"].mode()[0])
train["BsmtQual"] = train["BsmtQual"].fillna(train["BsmtQual"].mode()[0])
train["MasVnrArea"] = train["MasVnrArea"].fillna(train["MasVnrArea"].mode()[0])
train["MasVnrType"] = train["MasVnrType"].fillna(train["MasVnrType"].mode()[0])
train["LotFrontage"] = train["LotFrontage"].fillna(train["LotFrontage"].mode()[0])
```

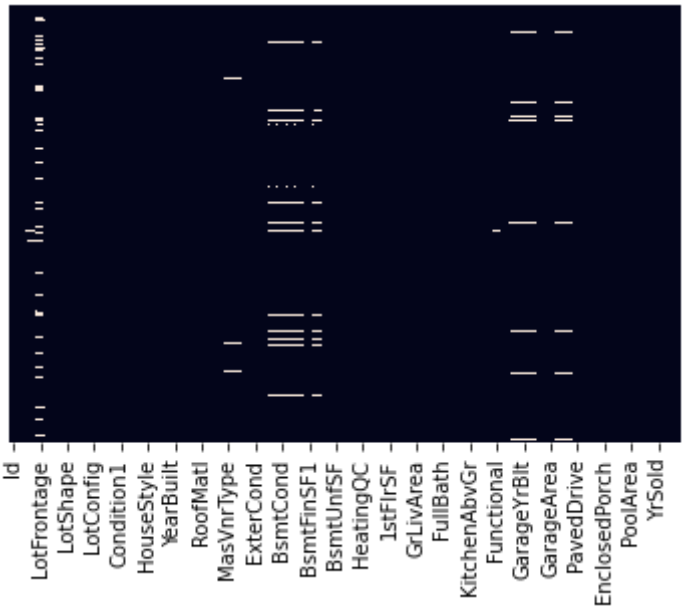
In [15]:
```
train.isnull().sum()
```

Out[15]:
```
Id               0
MSSubClass       0
MSZoning         0
LotFrontage      0
LotArea          0
                ..
MoSold           0
YrSold           0
SaleType         0
SaleCondition    0
SalePrice        0
Length: 76, dtype: int64
```

In [16]:
```
#train.info()
```

In [17]:
```
sns.heatmap(test.isnull(), yticklabels=False, cbar=False)
```

Out[17]:
```
<AxesSubplot:>
```



In [18]:
```
# test data set are also checked for missing and in appropriate values
test.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1459 entries, 0 to 1458
Data columns (total 75 columns):
 #   Column         Non-Null Count  Dtype
---  ------         --------------  -----
 0   Id             1459 non-null   int64
 1   MSSubClass     1459 non-null   int64
 2   MSZoning       1455 non-null   object
 3   LotFrontage    1232 non-null   float64
 4   LotArea        1459 non-null   int64
 5   Street         1459 non-null   object
 6   LotShape       1459 non-null   object
 7   LandContour    1459 non-null   object
 8   Utilities      1457 non-null   object
 9   LotConfig      1459 non-null   object
 10  LandSlope      1459 non-null   object
 11  Neighborhood   1459 non-null   object
 12  Condition1     1459 non-null   object
 13  Condition2     1459 non-null   object
 14  BldgType       1459 non-null   object
 15  HouseStyle     1459 non-null   object
 16  OverallQual    1459 non-null   int64
 17  OverallCond    1459 non-null   int64
```

```
 18  YearBuilt      1459 non-null   int64
 19  YearRemodAdd   1459 non-null   int64
 20  RoofStyle      1459 non-null   object
 21  RoofMatl       1459 non-null   object
 22  Exterior1st    1458 non-null   object
 23  Exterior2nd    1458 non-null   object
 24  MasVnrType     1443 non-null   object
 25  MasVnrArea     1444 non-null   float64
 26  ExterQual      1459 non-null   object
 27  ExterCond      1459 non-null   object
 28  Foundation     1459 non-null   object
 29  BsmtQual       1415 non-null   object
 30  BsmtCond       1414 non-null   object
 31  BsmtExposure   1415 non-null   object
 32  BsmtFinType1   1417 non-null   object
 33  BsmtFinSF1     1458 non-null   float64
 34  BsmtFinType2   1417 non-null   object
 35  BsmtFinSF2     1458 non-null   float64
 36  BsmtUnfSF      1458 non-null   float64
 37  TotalBsmtSF    1458 non-null   float64
 38  Heating        1459 non-null   object
 39  HeatingQC      1459 non-null   object
 40  CentralAir     1459 non-null   object
 41  Electrical     1459 non-null   object
 42  1stFlrSF       1459 non-null   int64
 43  2ndFlrSF       1459 non-null   int64
 44  LowQualFinSF   1459 non-null   int64
 45  GrLivArea      1459 non-null   int64
 46  BsmtFullBath   1457 non-null   float64
 47  BsmtHalfBath   1457 non-null   float64
 48  FullBath       1459 non-null   int64
 49  HalfBath       1459 non-null   int64
 50  BedroomAbvGr   1459 non-null   int64
 51  KitchenAbvGr   1459 non-null   int64
 52  KitchenQual    1458 non-null   object
 53  TotRmsAbvGrd   1459 non-null   int64
 54  Functional     1457 non-null   object
 55  Fireplaces     1459 non-null   int64
 56  GarageType     1383 non-null   object
 57  GarageYrBlt    1381 non-null   float64
 58  GarageFinish   1381 non-null   object
 59  GarageCars     1458 non-null   float64
 60  GarageArea     1458 non-null   float64
 61  GarageQual     1381 non-null   object
 62  GarageCond     1381 non-null   object
 63  PavedDrive     1459 non-null   object
 64  WoodDeckSF     1459 non-null   int64
 65  OpenPorchSF    1459 non-null   int64
 66  EnclosedPorch  1459 non-null   int64
 67  3SsnPorch      1459 non-null   int64
 68  ScreenPorch    1459 non-null   int64
 69  PoolArea       1459 non-null   int64
 70  MiscVal        1459 non-null   int64
 71  MoSold         1459 non-null   int64
 72  YrSold         1459 non-null   int64
 73  SaleType       1458 non-null   object
 74  SaleCondition  1459 non-null   object
dtypes: float64(11), int64(26), object(38)
memory usage: 855.0+ KB
```

In [19]:
```
test.isnull().sum().tail(40)
```

Out[19]:
```
BsmtFinSF2        1
BsmtUnfSF         1
TotalBsmtSF       1
Heating           0
HeatingQC         0
CentralAir        0
Electrical        0
```

```
1stFlrSF          0
2ndFlrSF          0
LowQualFinSF      0
GrLivArea         0
BsmtFullBath      2
BsmtHalfBath      2
FullBath          0
HalfBath          0
BedroomAbvGr      0
KitchenAbvGr      0
KitchenQual       1
TotRmsAbvGrd      0
Functional        2
Fireplaces        0
GarageType       76
GarageYrBlt      78
GarageFinish     78
GarageCars        1
GarageArea        1
GarageQual       78
GarageCond       78
PavedDrive        0
WoodDeckSF        0
OpenPorchSF       0
EnclosedPorch     0
3SsnPorch         0
ScreenPorch       0
PoolArea          0
MiscVal           0
MoSold            0
YrSold            0
SaleType          1
SaleCondition     0
dtype: int64
```

In [20]:

```
# Filling null values with mode in all features as done for train data set

test["LotFrontage"] = test["LotFrontage"].fillna(train["LotFrontage"].mode()[0])
test["MSZoning"] = test["MSZoning"].fillna(train["MSZoning"].mode()[0])
test["Utilities"] = test["Utilities"].fillna(train["Utilities"].mode()[0])
test["Exterior1st"] = test["Exterior1st"].fillna(train["Exterior1st"].mode()[0])
test["Exterior2nd"] = test["Exterior2nd"].fillna(train["Exterior2nd"].mode()[0])

test["MasVnrType"] = test["MasVnrType"].fillna(train["MasVnrType"].mode()[0])
test["MasVnrArea"] = test["MasVnrArea"].fillna(train["MasVnrArea"].mode()[0])
test["BsmtQual"] = test["BsmtQual"].fillna(train["BsmtQual"].mode()[0])

test["BsmtCond"] = test["BsmtCond"].fillna(train["BsmtCond"].mode()[0])
test["BsmtExposure"] = test["BsmtExposure"].fillna(train["BsmtExposure"].mode()[0])
test["BsmtFinType1"] = test["BsmtFinType1"].fillna(train["BsmtFinType1"].mode()[0])
test["BsmtFinSF1"] = test["BsmtFinSF1"].fillna(train["BsmtFinSF1"].mode()[0])
test["BsmtFinType2"] = test["BsmtFinType2"].fillna(train["BsmtFinType2"].mode()[0])
test["BsmtFinSF2"] = test["BsmtFinSF2"].fillna(train["BsmtFinSF2"].mode()[0])
test["BsmtUnfSF"] = test["BsmtUnfSF"].fillna(train["BsmtUnfSF"].mode()[0])
test["TotalBsmtSF"] = test["TotalBsmtSF"].fillna(train["TotalBsmtSF"].mode()[0])

test["BsmtFullBath"] = test["BsmtFullBath"].fillna(train["BsmtFullBath"].mode()[0])
test["BsmtHalfBath"] = test["BsmtHalfBath"].fillna(train["BsmtHalfBath"].mode()[0])

test["KitchenQual"] = test["KitchenQual"].fillna(train["KitchenQual"].mode()[0])
test["Functional"] = test["Functional"].fillna(train["Functional"].mode()[0])
test["Fireplaces"] = test["Fireplaces"].fillna(train["Fireplaces"].mode()[0])
test["GarageType"] = test["GarageType"].fillna(train["GarageType"].mode()[0])
test["GarageYrBlt"] = test["GarageYrBlt"].fillna(train["GarageYrBlt"].mode()[0])
test["GarageFinish"] = test["GarageCars"].fillna(train["GarageCars"].mode()[0])
test["GarageCars"] = test["BsmtHalfBath"].fillna(train["BsmtHalfBath"].mode()[0])
test["GarageArea"] = test["GarageArea"].fillna(train["GarageArea"].mode()[0])
```

```
test["GarageQual"] = test["GarageQual"].fillna(train["GarageQual"].mode()[0])
test["GarageCond"] = test["GarageCond"].fillna(train["GarageCond"].mode()[0])
test["SaleType"] = test["SaleType"].fillna(train["SaleType"].mode()[0])
```

In [21]:
```
test.isnull().info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1459 entries, 0 to 1458
Data columns (total 75 columns):
 #   Column         Non-Null Count  Dtype
---  ------         --------------  -----
 0   Id             1459 non-null   bool
 1   MSSubClass     1459 non-null   bool
 2   MSZoning       1459 non-null   bool
 3   LotFrontage    1459 non-null   bool
 4   LotArea        1459 non-null   bool
 5   Street         1459 non-null   bool
 6   LotShape       1459 non-null   bool
 7   LandContour    1459 non-null   bool
 8   Utilities      1459 non-null   bool
 9   LotConfig      1459 non-null   bool
 10  LandSlope      1459 non-null   bool
 11  Neighborhood   1459 non-null   bool
 12  Condition1     1459 non-null   bool
 13  Condition2     1459 non-null   bool
 14  BldgType       1459 non-null   bool
 15  HouseStyle     1459 non-null   bool
 16  OverallQual    1459 non-null   bool
 17  OverallCond    1459 non-null   bool
 18  YearBuilt      1459 non-null   bool
 19  YearRemodAdd   1459 non-null   bool
 20  RoofStyle      1459 non-null   bool
 21  RoofMatl       1459 non-null   bool
 22  Exterior1st    1459 non-null   bool
 23  Exterior2nd    1459 non-null   bool
 24  MasVnrType     1459 non-null   bool
 25  MasVnrArea     1459 non-null   bool
 26  ExterQual      1459 non-null   bool
 27  ExterCond      1459 non-null   bool
 28  Foundation     1459 non-null   bool
 29  BsmtQual       1459 non-null   bool
 30  BsmtCond       1459 non-null   bool
 31  BsmtExposure   1459 non-null   bool
 32  BsmtFinType1   1459 non-null   bool
 33  BsmtFinSF1     1459 non-null   bool
 34  BsmtFinType2   1459 non-null   bool
 35  BsmtFinSF2     1459 non-null   bool
 36  BsmtUnfSF      1459 non-null   bool
 37  TotalBsmtSF    1459 non-null   bool
 38  Heating        1459 non-null   bool
 39  HeatingQC      1459 non-null   bool
 40  CentralAir     1459 non-null   bool
 41  Electrical     1459 non-null   bool
 42  1stFlrSF       1459 non-null   bool
 43  2ndFlrSF       1459 non-null   bool
 44  LowQualFinSF   1459 non-null   bool
 45  GrLivArea      1459 non-null   bool
 46  BsmtFullBath   1459 non-null   bool
 47  BsmtHalfBath   1459 non-null   bool
 48  FullBath       1459 non-null   bool
 49  HalfBath       1459 non-null   bool
 50  BedroomAbvGr   1459 non-null   bool
 51  KitchenAbvGr   1459 non-null   bool
 52  KitchenQual    1459 non-null   bool
 53  TotRmsAbvGrd   1459 non-null   bool
 54  Functional     1459 non-null   bool
 55  Fireplaces     1459 non-null   bool
 56  GarageType     1459 non-null   bool
```

```
 57  GarageYrBlt    1459 non-null   bool
 58  GarageFinish   1459 non-null   bool
 59  GarageCars     1459 non-null   bool
 60  GarageArea     1459 non-null   bool
 61  GarageQual     1459 non-null   bool
 62  GarageCond     1459 non-null   bool
 63  PavedDrive     1459 non-null   bool
 64  WoodDeckSF     1459 non-null   bool
 65  OpenPorchSF    1459 non-null   bool
 66  EnclosedPorch  1459 non-null   bool
 67  3SsnPorch      1459 non-null   bool
 68  ScreenPorch    1459 non-null   bool
 69  PoolArea       1459 non-null   bool
 70  MiscVal        1459 non-null   bool
 71  MoSold         1459 non-null   bool
 72  YrSold         1459 non-null   bool
 73  SaleType       1459 non-null   bool
 74  SaleCondition  1459 non-null   bool
dtypes: bool(75)
memory usage: 107.0 KB
```

In [22]:
```
# merging of train and test datasets
merge_df = pd.concat([train, test], axis=0)
```

In [23]:
```
merge_df.shape, train.shape, test.shape
```

Out[23]:
```
((2919, 76), (1460, 76), (1459, 75))
```

In [24]:
```
merge_df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 2919 entries, 0 to 1458
Data columns (total 76 columns):
 #   Column         Non-Null Count  Dtype
---  ------         --------------  -----
 0   Id             2919 non-null   int64
 1   MSSubClass     2919 non-null   int64
 2   MSZoning       2919 non-null   object
 3   LotFrontage    2919 non-null   float64
 4   LotArea        2919 non-null   int64
 5   Street         2919 non-null   object
 6   LotShape       2919 non-null   object
 7   LandContour    2919 non-null   object
 8   Utilities      2919 non-null   object
 9   LotConfig      2919 non-null   object
 10  LandSlope      2919 non-null   object
 11  Neighborhood   2919 non-null   object
 12  Condition1     2919 non-null   object
 13  Condition2     2919 non-null   object
 14  BldgType       2919 non-null   object
 15  HouseStyle     2919 non-null   object
 16  OverallQual    2919 non-null   int64
 17  OverallCond    2919 non-null   int64
 18  YearBuilt      2919 non-null   int64
 19  YearRemodAdd   2919 non-null   int64
 20  RoofStyle      2919 non-null   object
 21  RoofMatl       2919 non-null   object
 22  Exterior1st    2919 non-null   object
 23  Exterior2nd    2919 non-null   object
 24  MasVnrType     2919 non-null   object
 25  MasVnrArea     2919 non-null   float64
 26  ExterQual      2919 non-null   object
 27  ExterCond      2919 non-null   object
 28  Foundation     2919 non-null   object
 29  BsmtQual       2919 non-null   object
 30  BsmtCond       2919 non-null   object
 31  BsmtExposure   2919 non-null   object
 32  BsmtFinType1   2919 non-null   object
```

```
 33  BsmtFinSF1     2919 non-null   float64
 34  BsmtFinType2   2919 non-null   object
 35  BsmtFinSF2     2919 non-null   float64
 36  BsmtUnfSF      2919 non-null   float64
 37  TotalBsmtSF    2919 non-null   float64
 38  Heating        2919 non-null   object
 39  HeatingQC      2919 non-null   object
 40  CentralAir     2919 non-null   object
 41  Electrical     2918 non-null   object
 42  1stFlrSF       2919 non-null   int64
 43  2ndFlrSF       2919 non-null   int64
 44  LowQualFinSF   2919 non-null   int64
 45  GrLivArea      2919 non-null   int64
 46  BsmtFullBath   2919 non-null   float64
 47  BsmtHalfBath   2919 non-null   float64
 48  FullBath       2919 non-null   int64
 49  HalfBath       2919 non-null   int64
 50  BedroomAbvGr   2919 non-null   int64
 51  KitchenAbvGr   2919 non-null   int64
 52  KitchenQual    2919 non-null   object
 53  TotRmsAbvGrd   2919 non-null   int64
 54  Functional     2919 non-null   object
 55  Fireplaces     2919 non-null   int64
 56  GarageType     2919 non-null   object
 57  GarageYrBlt    2919 non-null   float64
 58  GarageFinish   2919 non-null   object
 59  GarageCars     2919 non-null   float64
 60  GarageArea     2919 non-null   float64
 61  GarageQual     2919 non-null   object
 62  GarageCond     2919 non-null   object
 63  PavedDrive     2919 non-null   object
 64  WoodDeckSF     2919 non-null   int64
 65  OpenPorchSF    2919 non-null   int64
 66  EnclosedPorch  2919 non-null   int64
 67  3SsnPorch      2919 non-null   int64
 68  ScreenPorch    2919 non-null   int64
 69  PoolArea       2919 non-null   int64
 70  MiscVal        2919 non-null   int64
 71  MoSold         2919 non-null   int64
 72  YrSold         2919 non-null   int64
 73  SaleType       2919 non-null   object
 74  SaleCondition  2919 non-null   object
 75  SalePrice      1460 non-null   float64
dtypes: float64(12), int64(26), object(38)
memory usage: 1.7+ MB
```

merge_df.info()

In [25]:
```
merge_df.head()
```

Out[25]:

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

5 rows × 76 columns

In [26]:
```
category = merge_df.select_dtypes('object').columns
```

In [27]:
```
category
```

Out[27]:
```
Index(['MSZoning', 'Street', 'LotShape', 'LandContour', 'Utilities',
       'LotConfig', 'LandSlope', 'Neighborhood', 'Condition1', 'Condition2',
       'BldgType', 'HouseStyle', 'RoofStyle', 'RoofMatl', 'Exterior1st',
       'Exterior2nd', 'MasVnrType', 'ExterQual', 'ExterCond', 'Foundation',
       'BsmtQual', 'BsmtCond', 'BsmtExposure', 'BsmtFinType1', 'BsmtFinType2',
       'Heating', 'HeatingQC', 'CentralAir', 'Electrical', 'KitchenQual',
       'Functional', 'GarageType', 'GarageFinish', 'GarageQual', 'GarageCond',
       'PavedDrive', 'SaleType', 'SaleCondition'],
      dtype='object')
```

In [28]:
```
# Applying get dummies to all the category columns altogether

df = pd.get_dummies(merge_df, columns=category, drop_first=True)
```

In [29]:
```
df.shape
```

Out[29]:
```
(2919, 240)
```

In [30]:
```
final_df = df.loc[:,~df.columns.duplicated()]
```

In [31]:
```
final_df.shape
```

Out[31]:
```
(2919, 240)
```

In [32]:
```
# spliting data into train and test data sets after applying get_dummies

train_final = final_df.iloc[:1460,:]
test_final = final_df.iloc[1460:,:]
```

In [33]:
```
train_final.shape, test_final.shape
```

Out[33]:
```
((1460, 240), (1459, 240))
```

In [34]:
```
train_final.columns
```

Out[34]:
```
Index(['Id', 'MSSubClass', 'LotFrontage', 'LotArea', 'OverallQual',
       'OverallCond', 'YearBuilt', 'YearRemodAdd', 'MasVnrArea', 'BsmtFinSF1',
       ...
       'SaleType_ConLI', 'SaleType_ConLw', 'SaleType_New', 'SaleType_Oth',
       'SaleType_WD', 'SaleCondition_AdjLand', 'SaleCondition_Alloca',
       'SaleCondition_Family', 'SaleCondition_Normal',
       'SaleCondition_Partial'],
      dtype='object', length=240)
```

Develop

In [35]:
```
# Development of models

X = train_final.drop(['Id','SalePrice'], axis=1)
y = train_final['SalePrice']
```

In [36]:
```
from sklearn.model_selection import train_test_split

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_stat
```

In [37]:
```
## As the target variable is numeric so regression model is required. Therefore, for
# regression model is developed

from sklearn.linear_model import LinearRegression
```

```
model_lr = LinearRegression()
model_lr.fit(X_train,y_train)
y_train_pred = model_lr.predict(X_train)
y_test_pred = model_lr.predict(X_test)
```

In [38]:
```
# Meausuing R-sqaure value to the model using R-square

from sklearn.metrics import r2_score
print("The R-squared value of train data prediction is:", r2_score(y_train_pred, y_t
```

```
The R-squared value of train data prediction is: 0.937014932285923
```

In [39]:
```
# Thus train model has R-square value of 0.93 which is pretty good
```

In [40]:
```
print("The R-squared value of test data prediction is:", r2_score(y_test_pred, y_tes
```

```
The R-squared value of test data prediction is: 0.7796815713258304
```

In [41]:
```
# The R-square value of test data set is 0.77 which is already a good model.
```

In [42]:
```
# Now, linear regression model is the base model here. I can compare and contrast wi
# can select best model. I am going to develop Random Forest Regression and Decison
```

In [43]:
```
X_test.shape, test_final.shape
```

Out[43]:
```
((292, 238), (1459, 240))
```

In [44]:
```
# Sample data provided in Kaggle

sample_submission.shape
```

Out[44]:
```
(1459, 2)
```

In [45]:
```
# Now, prepared test data using linear regresssion model just developed above to pre

test_final1 = test_final.drop(['SalePrice','Id'], axis=1)
```

In [46]:
```
test_final1.head()
```

Out[46]:

	MSSubClass	LotFrontage	LotArea	OverallQual	OverallCond	YearBuilt	YearRemodAdd	MasVnr
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	

5 rows × 238 columns

In [47]:
```
test_final1.shape, y.shape
```

Out[47]:
```
((1459, 238), (1460,))
```

In [48]:
```
X_train.shape
```

Out[48]:
```
(1168, 238)
```

In [49]:
```
y_final_test_pred = model_lr.predict(test_final1)
```

In [50]:
```
prediction = pd.DataFrame(y_final_test_pred)
```

In [51]:
```
sample_submission.drop('SalePrice', axis=1)
```

Out[51]:

	Id
0	1461
1	1462
2	1463
3	1464
4	1465
...	...
1454	2915
1455	2916
1456	2917
1457	2918
1458	2919

1459 rows × 1 columns

In [52]:
```
sample_submission["lr_prediction"] = prediction
```

In [53]:
```
sample_submission.head()
```

Out[53]:

	Id	SalePrice	lr_prediction
0	1461	169277.052498	99296.283132
1	1462	187758.393989	162949.819484
2	1463	183583.683570	181199.181187
3	1464	179317.477511	183731.713507
4	1465	150730.079977	177469.509602

Developing Random Forest Regressor Model

In [54]:
```
from sklearn.ensemble import RandomForestRegressor
model_rf = RandomForestRegressor()
model_rf.fit(X_train, y_train)
y_rf_train_pred = model_rf.predict(X_train)
y_rf_test_pred = model_rf.predict(X_test)
```

In [55]:
```
print("The R-square value of train dataset using RF model is:r", r2_score(y_rf_train
print("The R-square value of test dataset using RF model is:r", r2_score(y_rf_test_p
```

```
The R-square value of train dataset using RF model is:r 0.9726095006503043
The R-square value of test dataset using RF model is:r 0.8430272624351509
```

In [ ]:

In [56]:
```
# Here, the R-square value for both train and test data sets are pretty good and bet
```

In [56]:
```
# Prediction of test data set with the developed Random forest model
y_rf_test_final = model_rf.predict(test_final1)
```

In [57]:
```
sample_submission["rf_prediction"] = pd.DataFrame(y_rf_test_final)
```

In [58]:
```
sample_submission.head()
```

Out[58]:

	Id	SalePrice	lr_prediction	rf_prediction
0	1461	169277.052498	99296.283132	129336.00
1	1462	187758.393989	162949.819484	154716.72
2	1463	183583.683570	181199.181187	182125.90
3	1464	179317.477511	183731.713507	177340.00
4	1465	150730.079977	177469.509602	196863.92

Optimization of RandomForest using Grid Search

In [221…
```
from sklearn.model_selection import GridSearchCV
# Create the parameter grid based on the results of random search
param_grid = {
    'bootstrap': [True, False],
    'max_depth': [50, 60,110],
    'max_features': [20,30,40],
    'min_samples_leaf': [1,2],
    'min_samples_split': [1,2,3],
    'n_estimators': [100, 200, 300]
}
# Create a based model
rf = RandomForestRegressor()
# Instantiate the grid search model
grid_search = GridSearchCV(estimator = rf, param_grid = param_grid,
                           cv = 3, n_jobs = -1, verbose = 2)
```

In [222…
```
# Fit the grid search to the data
grid_search.fit(X_train, y_train)
best_grid = grid_search.best_estimator_
#grid_accuracy = evaluate(best_grid, test_features, test_labels)

#print('Improvement of {:0.2f}%.'.format( 100 * (grid_accuracy - base_accuracy) / ba
#Improvement of 0.50%.
```

```
Fitting 3 folds for each of 324 candidates, totalling 972 fits
[Parallel(n_jobs=-1)]: Using backend LokyBackend with 4 concurrent workers.
[Parallel(n_jobs=-1)]: Done  33 tasks      | elapsed:   11.6s
[Parallel(n_jobs=-1)]: Done 154 tasks      | elapsed:   44.9s
[Parallel(n_jobs=-1)]: Done 357 tasks      | elapsed:  1.6min
[Parallel(n_jobs=-1)]: Done 640 tasks      | elapsed:  2.9min
[Parallel(n_jobs=-1)]: Done 972 out of 972 | elapsed:  4.6min finished
```

In [223…
```
grid_search.best_params_
```

Out[223…
```
{'bootstrap': False,
 'max_depth': 60,
 'max_features': 40,
 'min_samples_leaf': 1,
```

```
 'min_samples_split': 2,
 'n_estimators': 300}
```

In [159…
```
best_grid
```

Out[159…
```
RandomForestRegressor(bootstrap=False, max_depth=90, max_features=3,
                      min_samples_leaf=3, min_samples_split=10,
                      n_estimators=200)
```

In [224…
```
model_rf = RandomForestRegressor(
    n_estimators=300,
    min_samples_split=2,
    min_samples_leaf=1,
    max_depth=60,
    max_features=40,
    bootstrap=False
)
model_rf.fit(X_train, y_train)
y_rf_train_pred_best = model_rf.predict(X_train)
y_rf_test_pred_best = model_rf.predict(X_test)
```

In [225…
```
print("The R-square value of train dataset using RF model is:r", r2_score(y_rf_train
print("The R-square value of test dataset using RF model is:r", r2_score(y_rf_test_p
```

```
The R-square value of train dataset using RF model is:r 0.999999999508116
The R-square value of test dataset using RF model is:r 0.8444924449587848
```

Measuring accuracy and RMSE of the developed models using a function calculate

In [231…
```
def calculate(model, X_test, y_test):
    predictions = model.predict(X_test)
    errors = abs(predictions - y_test)
    mape = 100 * np.mean(errors / y_test)
    accuracy = 100 - mape
    RMSE = np.sqrt(mean_squared_error(predictions, y_test))
    print('Model Performance')
    print('--------------------')
    print('Average Error:', np.mean(errors))
    print('Accuracy:', (accuracy))
    print('RMSE:', RMSE)

    return accuracy

base_model = RandomForestRegressor(random_state = 2)
base_model.fit(X_train, y_train)
base_model_accuracy = calculate(base_model, X_test, y_test)
print('-----------------------------------------------------')
optimized_model_accuracy = calculate(model_rf, X_test, y_test)
print('Improvement in model is:',optimized_model_accuracy- base_model_accuracy)
```

```
Model Performance
--------------------
Average Error: 17704.7516438356
Accuracy: 89.554730294521
RMSE: 28471.471455143983
-----------------------------------------------------
Model Performance
--------------------
Average Error: 16474.530456621018
Accuracy: 89.99501556972096
RMSE: 26736.909287015907
Improvement in model is: 0.44028527519996885
```

Development of Decision Tree

In [234…
```
# Decision Tree Model
from sklearn.tree import DecisionTreeRegressor
model_tree = DecisionTreeRegressor()
model_tree.fit(X_train, y_train)
y_tree_train_pred = model_tree.predict(X_train)
y_tree_test_pred = model_tree.predict(X_test)
```

In [235…
```
print("The R-square value of train dataset using RF model is:r", r2_score(y_tree_tra
print("The R-square value of test dataset using RF model is:r", r2_score(y_tree_test
decision_tree_accuracy = calculate(model_tree, X_test, y_test)
```

```
The R-square value of train dataset using RF model is:r 1.0
The R-square value of test dataset using RF model is:r 0.8099511864510447
Model Performance
--------------------
Average Error: 23698.794520547945
Accuracy: 86.26702640143371
RMSE: 33697.27516330522
```

Development of XGBoost model

In [282…
```
import xgboost as xgb

model_xgb = xgb.XGBRegressor(objective ='reg:linear', n_estimators = 100)
dtrain = xgb.DMatrix(X_train, y_train)
model_xgb.fit(X_train, y_train)
y_train_model_xgb_pred = model_xgb.predict(X_train)
y_test_model_xgb_pred = model_xgb.predict(X_test)
```

```
[15:54:20] WARNING: /opt/concourse/worker/volumes/live/7a2b9f41-3287-451b-6691-43e9a
6c0910f/volume/xgboost-split_1619728204606/work/src/objective/regression_obj.cu:170:
reg:linear is now deprecated in favor of reg:squarederror.
```

In [283…
```
calculate(model_xgb, X_test, y_test)
```

```
Model Performance
--------------------
Average Error: 17368.97236194349
Accuracy: 89.93378002409649
RMSE: 28701.369139196675
```

Out[283…
```
89.93378002409649
```

Optimization of XGBoost models using Grid Search

In [288…
```
params = {
    'objective' :['reg:linear'], 'colsample_bytree':[0.2, 0.3, 0.5], 'learning_rate'
                'max_depth':[2,3,5], 'alpha' : [2,5, 7], 'n_estimators' : [180, 100,
}
```

In [299…
```
from sklearn.model_selection import GridSearchCV

xgb_grid_searh = GridSearchCV(estimator = model_xgb, param_grid = params, cv = 3, n_
```

In [300…
```
xgb_grid_searh.fit(X_train, y_train)
```

```
Fitting 3 folds for each of 243 candidates, totalling 729 fits
[Parallel(n_jobs=-1)]: Using backend LokyBackend with 4 concurrent workers.
[Parallel(n_jobs=-1)]: Done  33 tasks      | elapsed:   18.8s
[Parallel(n_jobs=-1)]: Done 154 tasks      | elapsed:  1.3min
[Parallel(n_jobs=-1)]: Done 357 tasks      | elapsed:  3.3min
```

```
[Parallel(n_jobs=-1)]: Done 640 tasks      | elapsed:  6.2min
[Parallel(n_jobs=-1)]: Done 729 out of 729 | elapsed:  7.3min finished
[16:12:33] WARNING: /opt/concourse/worker/volumes/live/7a2b9f41-3287-451b-6691-43e9a
6c0910f/volume/xgboost-split_1619728204606/work/src/objective/regression_obj.cu:170:
reg:linear is now deprecated in favor of reg:squarederror.
```

Out[300…

```
GridSearchCV(cv=3,
             estimator=XGBRegressor(base_score=0.5, booster='gbtree',
                                    colsample_bylevel=1, colsample_bynode=1,
                                    colsample_bytree=1, gamma=0, gpu_id=-1,
                                    importance_type='gain',
                                    interaction_constraints='',
                                    learning_rate=0.300000012, max_delta_step=0,
                                    max_depth=6, min_child_weight=1,
                                    missing=nan, monotone_constraints='()',
                                    n_estimators=100, n_jobs=4,
                                    num_parallel_tree=1, objective='reg:linear',
                                    random_state=0, reg_alpha=0, reg_lambda=1,
                                    scale_pos_weight=1, subsample=1,
                                    tree_method='exact', validate_parameters=1,
                                    verbosity=None),
             n_jobs=-1,
             param_grid={'alpha': [2, 5, 7],
                         'colsample_bytree': [0.2, 0.3, 0.5],
                         'learning_rate': [0.02, 0.05, 0.1],
                         'max_depth': [2, 3, 5],
                         'n_estimators': [180, 100, 500],
                         'objective': ['reg:linear']},
             verbose=2)
```

In [304…

```
xgb_grid_searh.best_estimator_
```

Out[304…

```
XGBRegressor(alpha=2, base_score=0.5, booster='gbtree', colsample_bylevel=1,
             colsample_bynode=1, colsample_bytree=0.2, gamma=0, gpu_id=-1,
             importance_type='gain', interaction_constraints='',
             learning_rate=0.1, max_delta_step=0, max_depth=5,
             min_child_weight=1, missing=nan, monotone_constraints='()',
             n_estimators=500, n_jobs=4, num_parallel_tree=1,
             objective='reg:linear', random_state=0, reg_alpha=0, reg_lambda=1,
             scale_pos_weight=1, subsample=1, tree_method='exact',
             validate_parameters=1, verbosity=None)
```

In [ ]:

```
y_tree_test_final = model_tree.predict(test_final1)
```

In [305…

```
model_xgb_best_param = xgb.XGBRegressor(alpha=2,
                        colsample_bytree=0.2,
                        learning_rate=0.1,
                        max_depth=5,
                        n_estimators=500,
                        objective='reg:linear')
```

In [308…

```
model_xgb_best_param.fit(X_train, y_train)
model_xgb_best_param_train_pred = model_xgb_best_param.predict(X_train)
model_xgb_best_param_test_pred = model_xgb_best_param.predict(X_test)
```

```
[16:22:36] WARNING: /opt/concourse/worker/volumes/live/7a2b9f41-3287-451b-6691-43e9a
6c0910f/volume/xgboost-split_1619728204606/work/src/objective/regression_obj.cu:170:
reg:linear is now deprecated in favor of reg:squarederror.
```

In [317…

```
model_xgb_best_param_accuracy = calculate(model_xgb_best_param, X_test, y_test)
model_xgb_accuracy = calculate(model_xgb,X_test,y_test)
print('The improvment in accuracy is:', model_xgb_best_param_accuracy-model_xgb_accu
```

```
Model Performance
-------------------
Average Error: 15490.368177440068
Accuracy: 90.88597570290618
RMSE: 25454.57221451317
```

```
Model Performance
-------------------
Average Error: 17368.97236194349
Accuracy: 89.93378002409649
RMSE: 28701.369139196675
The improvment in accuracy is: 0.9521956788096873
```

In [ ]:
```
# Based on all the model tested above optimized xgboost exhibits best RMSE value and
# is selected to predict the house price onwards.
```

Deployment of Model

In [319…
```
model_xgb_best_param_final_predict = model_xgb_best_param.predict(test_final1)
```

In [321…
```
xgb_final = pd.DataFrame(model_xgb_best_param_final_predict)
xgb_final.head()
```

Out[321…

	0
0	137631.015625
1	166751.812500
2	186546.875000
3	185962.390625
4	167579.062500

In [323…
```
sample_submission.head()
```

Out[323…

	Id	SalePrice	lr_prediction	rf_prediction
0	1461	169277.052498	99296.283132	129336.00
1	1462	187758.393989	162949.819484	154716.72
2	1463	183583.683570	181199.181187	182125.90
3	1464	179317.477511	183731.713507	177340.00
4	1465	150730.079977	177469.509602	196863.92

In [327…
```
sample_submission["xgb_prediction"]=xgb_final
sample_submission.head()
```

Out[327…

	Id	SalePrice	lr_prediction	rf_prediction	xgb_prediction
0	1461	169277.052498	99296.283132	129336.00	137631.015625
1	1462	187758.393989	162949.819484	154716.72	166751.812500
2	1463	183583.683570	181199.181187	182125.90	186546.875000
3	1464	179317.477511	183731.713507	177340.00	185962.390625
4	1465	150730.079977	177469.509602	196863.92	167579.062500

In [328…
```
final_submission = sample_submission.drop(['SalePrice', 'lr_prediction', 'rf_predict
```

In [330…
```
final_submission.head()
```

Out[330…

	Id	xgb_prediction
0	1461	137631.015625
1	1462	166751.812500
2	1463	186546.875000
3	1464	185962.390625
4	1465	167579.062500

In [331…
```
final_submission1 = final_submission.rename(columns={'xgb_prediction':'SalePrice'})
```

In [332…
```
final_submission1.to_csv('/Users/bishnupaudel/Desktop/First_Portfolio_Python/final_s
```

In [99]:
```
final = sample_submission.drop(['SalePrice', 'lr_prediction'], axis=1)
```

In [107…
```
final.to_csv('/Users/bishnupaudel/Desktop/First_Portfolio_Python/final.csv', index=F
```

In [ ]:
```

```