*# This Python 3 environment comes with many helpful analytics libraries installed*

*# It is defined by the kaggle/python Docker image: https://github.com/kaggle/docker-python*

*# For example, here's several helpful packages to load*

import numpy as np *# linear algebra*

import pandas as pd *# data processing, CSV file I/O (e.g. pd.read\_csv)*

import matplotlib.pyplot as plt

plt.style.use("seaborn-paper")

import seaborn as sns

from collections import Counter

from sklearn.linear\_model import LinearRegression

from sklearn.preprocessing import PolynomialFeatures

from sklearn.ensemble import RandomForestRegressor

from sklearn.metrics import r2\_score

import warnings

warnings.filterwarnings("ignore")

*# Input data files are available in the read-only "../input/" directory*

*# For example, running this (by clicking run or pressing Shift+Enter) will list all files under the input directory*

import os

for dirname, \_, filenames **in** os.walk('/kaggle/input'):

for filename **in** filenames:

print(os.path.join(dirname, filename))

*# You can write up to 20GB to the current directory (/kaggle/working/) that gets preserved as output when you create a version using "Save & Run All"*

*# You can also write temporary files to /kaggle/temp/, but they won't be saved outside of the current session*

/kaggle/input/house-prices-advanced-regression-techniques/sample\_submission.csv

/kaggle/input/house-prices-advanced-regression-techniques/data\_description.txt

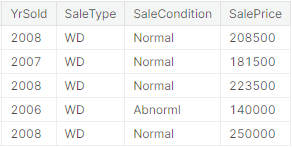
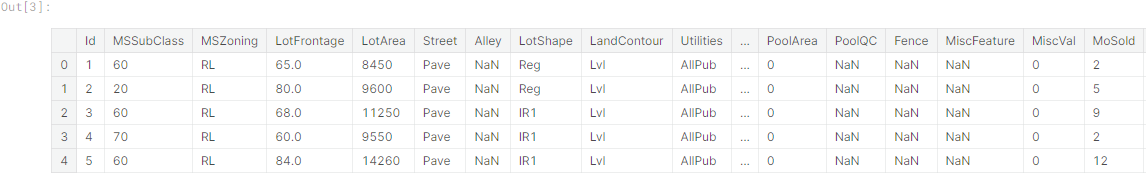
/kaggle/input/house-prices-advanced-regression-techniques/train.csv

/kaggle/input/house-prices-advanced-regression-techniques/test.csv

train\_data = pd.read\_csv("/kaggle/input/house-prices-advanced-regression-techniques/train.csv")

test\_data = pd.read\_csv("/kaggle/input/house-prices-advanced-regression-techniques/test.csv")

train\_data.head()



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

train\_data.isnull().sum()

Id 0

MSSubClass 0

MSZoning 0

LotFrontage 259

LotArea 0

...

MoSold 0

YrSold 0

SaleType 0

SaleCondition 0

SalePrice 0

Length: 81, dtype: int64

numerical = train\_data.drop(["Id"], axis = 1).select\_dtypes("number").columns

categorical = train\_data.select\_dtypes("object").columns

print (f"Numerical Columns: **{**train\_data[numerical].columns**}**", f"**\n**Categorical Columns: **{**train\_data[categorical].columns**}**")

Numerical Columns: Index(['MSSubClass', 'LotFrontage', 'LotArea', 'OverallQual', 'OverallCond',

'YearBuilt', 'YearRemodAdd', 'MasVnrArea', 'BsmtFinSF1', 'BsmtFinSF2',

'BsmtUnfSF', 'TotalBsmtSF', '1stFlrSF', '2ndFlrSF', 'LowQualFinSF',

'GrLivArea', 'BsmtFullBath', 'BsmtHalfBath', 'FullBath', 'HalfBath',

'BedroomAbvGr', 'KitchenAbvGr', 'TotRmsAbvGrd', 'Fireplaces',

'GarageYrBlt', 'GarageCars', 'GarageArea', 'WoodDeckSF', 'OpenPorchSF',

'EnclosedPorch', '3SsnPorch', 'ScreenPorch', 'PoolArea', 'MiscVal',

'MoSold', 'YrSold', 'SalePrice'],

dtype='object')

Categorical Columns: Index(['MSZoning', 'Street', 'Alley', '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',

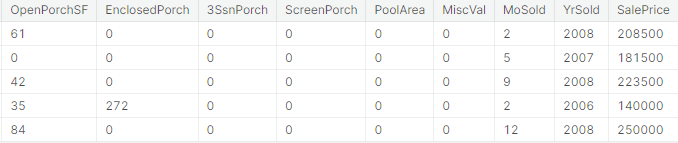
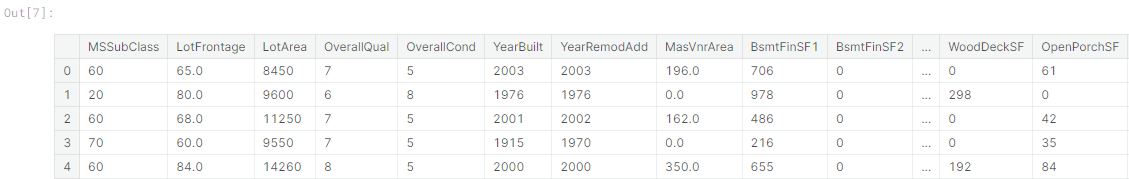
'Functional', 'FireplaceQu', 'GarageType', 'GarageFinish', 'GarageQual',

'GarageCond', 'PavedDrive', 'PoolQC', 'Fence', 'MiscFeature',

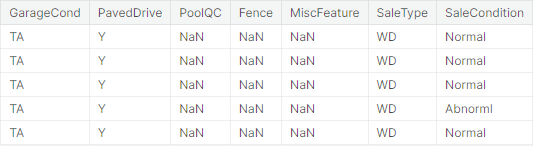
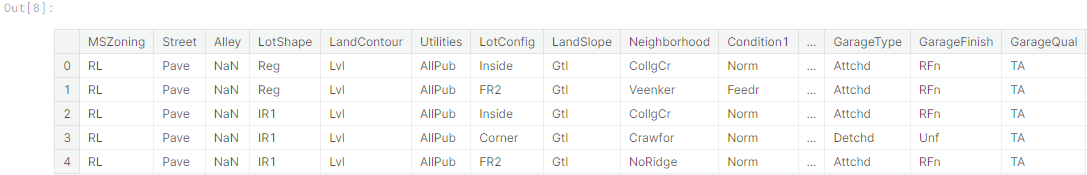
'SaleType', 'SaleCondition'],

dtype='object')

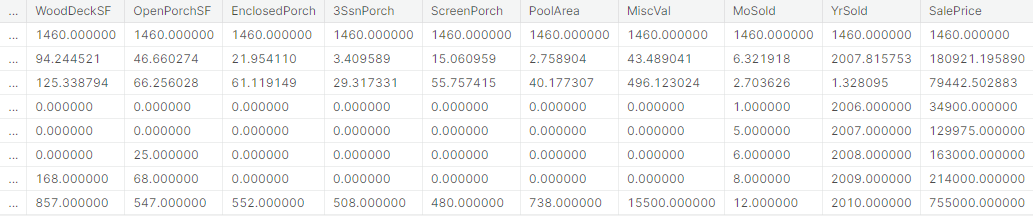
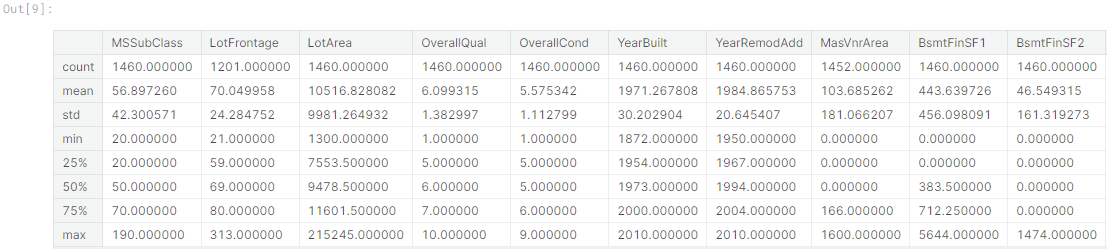
train\_data[numerical].head()



train\_data[categorical].head()



train\_data[numerical].describe()



train\_data[categorical].nunique()

MSZoning 5

Street 2

Alley 2

LotShape 4

LandContour 4

Utilities 2

LotConfig 5

LandSlope 3

Neighborhood 25

Condition1 9

Condition2 8

BldgType 5

HouseStyle 8

RoofStyle 6

RoofMatl 8

Exterior1st 15

Exterior2nd 16

MasVnrType 4

ExterQual 4

ExterCond 5

Foundation 6

BsmtQual 4

BsmtCond 4

BsmtExposure 4

BsmtFinType1 6

BsmtFinType2 6

Heating 6

HeatingQC 5

CentralAir 2

Electrical 5

KitchenQual 4

Functional 7

FireplaceQu 5

GarageType 6

GarageFinish 3

GarageQual 5

GarageCond 5

PavedDrive 3

PoolQC 3

Fence 4

MiscFeature 4

SaleType 9

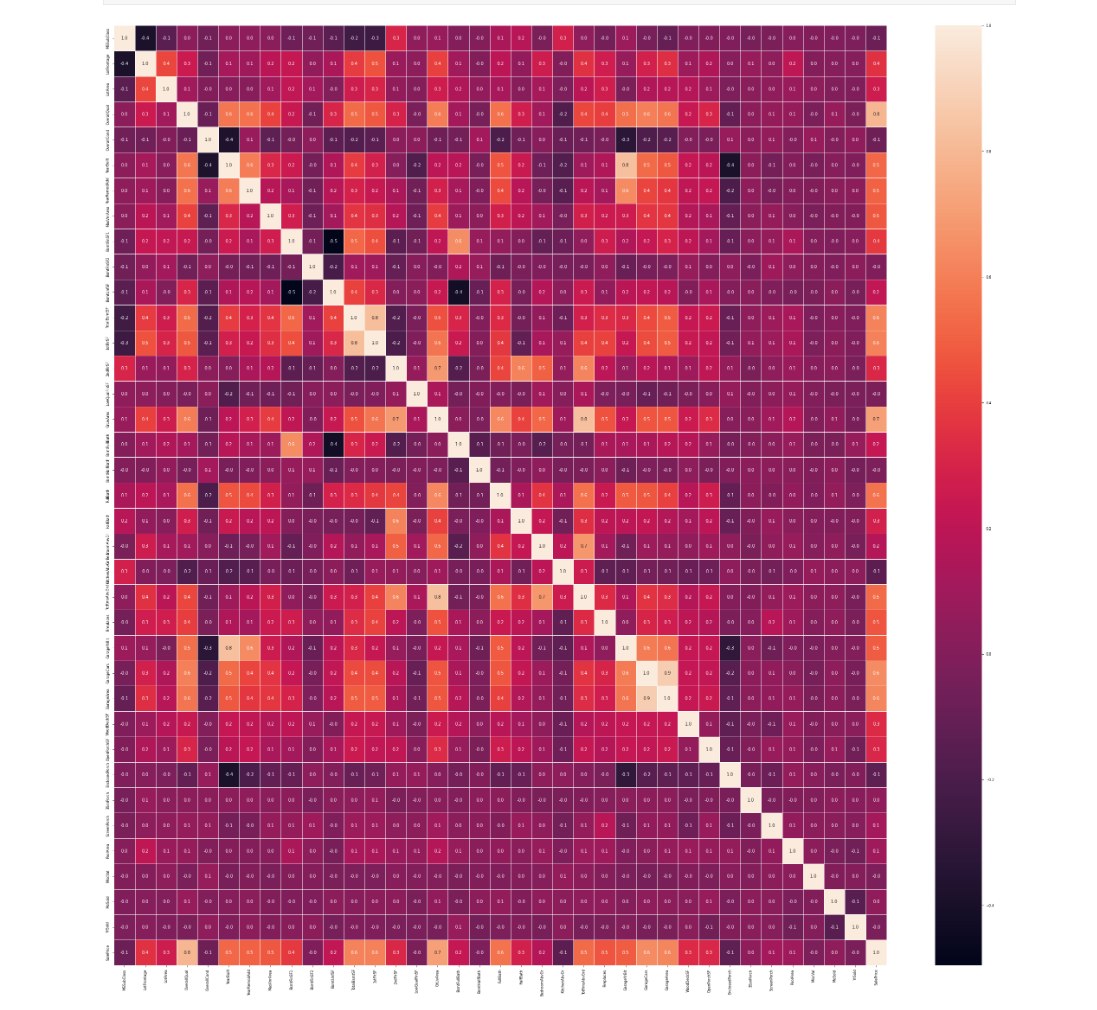
SaleCondition 6

dtype: int64

f,ax = plt.subplots(figsize = (40,40))

sns.heatmap(train\_data[numerical].corr(), annot = True, linewidths = .5, fmt ='.1f', ax = ax)

plt.show()



train\_data[numerical].plot(figsize = (40,40),subplots = True)

plt.show()



train\_data[numerical].nunique()

MSSubClass 15

LotFrontage 110

LotArea 1073

OverallQual 10

OverallCond 9

YearBuilt 112

YearRemodAdd 61

MasVnrArea 327

BsmtFinSF1 637

BsmtFinSF2 144

BsmtUnfSF 780

TotalBsmtSF 721

1stFlrSF 753

2ndFlrSF 417

LowQualFinSF 24

GrLivArea 861

BsmtFullBath 4

BsmtHalfBath 3

FullBath 4

HalfBath 3

BedroomAbvGr 8

KitchenAbvGr 4

TotRmsAbvGrd 12

Fireplaces 4

GarageYrBlt 97

GarageCars 5

GarageArea 441

WoodDeckSF 274

OpenPorchSF 202

EnclosedPorch 120

3SsnPorch 20

ScreenPorch 76

PoolArea 8

MiscVal 21

MoSold 12

YrSold 5

SalePrice 663

dtype: int64

def detect\_outliers(data,features):

outlier\_indices = []

for i **in** features:

Q1 = np.percentile(data[i],25)

Q3 = np.percentile(data[i],75)

IQR = Q3 - Q1

outlier\_step = IQR \* 1.5

outlier\_list\_col = data[(data[i] < Q1 - outlier\_step) | (data[i] > Q3 + outlier\_step)].index

outlier\_indices.extend(outlier\_list\_col)

outlier\_indices = Counter(outlier\_indices)

multiple\_outliers = list(i for i, v **in** outlier\_indices.items() if v > 2)

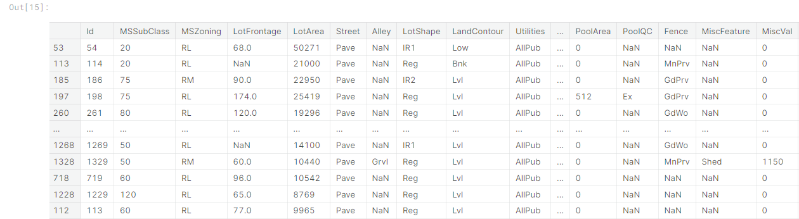
return multiple\_outliers

train\_data.loc[detect\_outliers(train\_data,["LotFrontage","LotArea","YearBuilt","MasVnrArea","BsmtFinSF1",

"BsmtFinSF2","BsmtUnfSF","TotalBsmtSF","1stFlrSF","2ndFlrSF",

"GrLivArea","GarageYrBlt","GarageArea","WoodDeckSF","OpenPorchSF",

"EnclosedPorch","ScreenPorch","SalePrice"])]



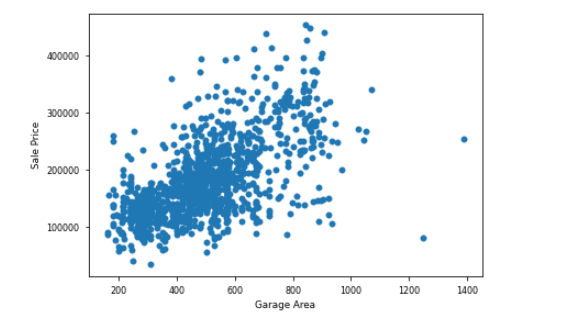
ml\_data = train\_data[train\_data.GarageArea > 100]

plt.scatter(ml\_data.GarageArea,ml\_data.SalePrice)

plt.xlabel("Garage Area")

plt.ylabel("Sale Price")

plt.show()



ml\_data.SalePrice.describe()

count 1319.000000

mean 178469.774071

std 65918.466844

min 35311.000000

25% 132500.000000

50% 164900.000000

75% 212450.000000

max 451950.000000

Name: SalePrice, dtype: float64

linear\_reg = LinearRegression()

x = ml\_data.GarageArea.values.reshape(-1,1)

y = ml\_data.SalePrice.values.reshape(-1,1)

linear\_reg.fit(x,y)

LinearRegression()

a = linear\_reg.predict([[0]])

a

array([[65877.18804627]])

a\_ = linear\_reg.intercept\_

a\_

array([65877.18804627])

b1 = linear\_reg.coef\_

b1

array([[229.95619652]])

linear\_reg.get\_params()

{'copy\_X': True, 'fit\_intercept': True, 'n\_jobs': None, 'normalize': False}

array = np.arange(150,1500,1).reshape(-1,1)

plt.scatter(x,y)

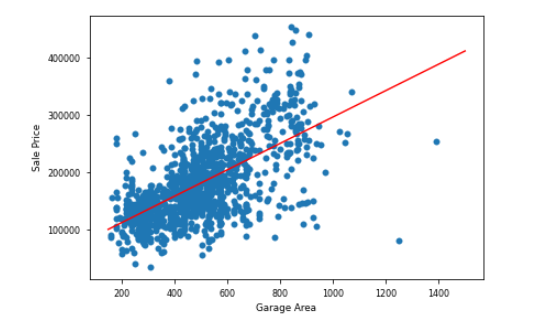
y\_head = linear\_reg.predict(array)

plt.plot(array, y\_head, color = "red")

plt.xlabel("Garage Area")

plt.ylabel("Sale Price")

plt.show()

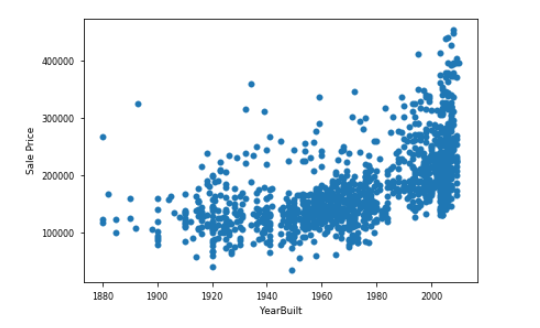


plt.scatter(ml\_data.YearBuilt,ml\_data.SalePrice)

plt.xlabel("YearBuilt")

plt.ylabel("Sale Price")

plt.show()



filtered\_ml\_data = ml\_data[(ml\_data.YearBuilt > 1920) & (ml\_data.YearBuilt < 1960)]

filtered\_ml\_data = ml\_data[ml\_data.SalePrice > 200000]

filtered\_ml\_data = ml\_data[ml\_data.LotArea > 40000]

filtered\_ml\_data.LotArea.describe()

count 5.000000

mean 90248.800000

std 46034.358888

min 53107.000000

25% 53227.000000

50% 70761.000000

75% 115149.000000

max 159000.000000

Name: LotArea, dtype: float64

x1 = filtered\_ml\_data.YearBuilt.values.reshape(-1,1)

y1 = filtered\_ml\_data.SalePrice.values.reshape(-1,1)

linear\_regression2 = LinearRegression()

linear\_regression2.fit(x1,y1)

x1 = np.sort(x1, axis = 0)

poly = PolynomialFeatures(degree = 2)

x1\_poly = poly.fit\_transform(x1)

poly.fit(x1\_poly, y1)

linear\_regression3 = LinearRegression()

linear\_regression3.fit(x1\_poly,y1)

y1\_head = linear\_regression3.predict(poly.fit\_transform(x1))

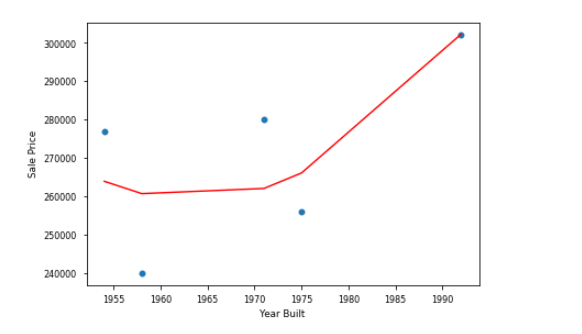
plt.scatter(x1,y1)

plt.plot(x1,y1\_head, color = "red", label = "poly")

plt.xlabel("Year Built")

plt.ylabel("Sale Price")

plt.show()



lot\_area = filtered\_ml\_data.LotArea.values.reshape(-1,1)

sale\_price = filtered\_ml\_data.SalePrice.values.reshape(-1,1)

rf = RandomForestRegressor(n\_estimators = 100, random\_state = 42)

rf.fit(lot\_area,sale\_price)

RandomForestRegressor(random\_state=42)

lot\_area1 = np.arange(min(lot\_area),max(lot\_area)).reshape(-1,1)

y\_head\_lot\_area = rf.predict(lot\_area1)

r2\_score(lot\_area1,y\_head\_lot\_area)

-32.79416430072849

plt.scatter(lot\_area,sale\_price)

plt.plot(lot\_area1,y\_head\_lot\_area, color = "red")

plt.xlabel("Lot Area")

plt.ylabel("Sale Price")

plt.show()

