Questions Related to Data Visualisation

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt

data= pd.read_csv("FuelConsumption.csv")

data.head()
```

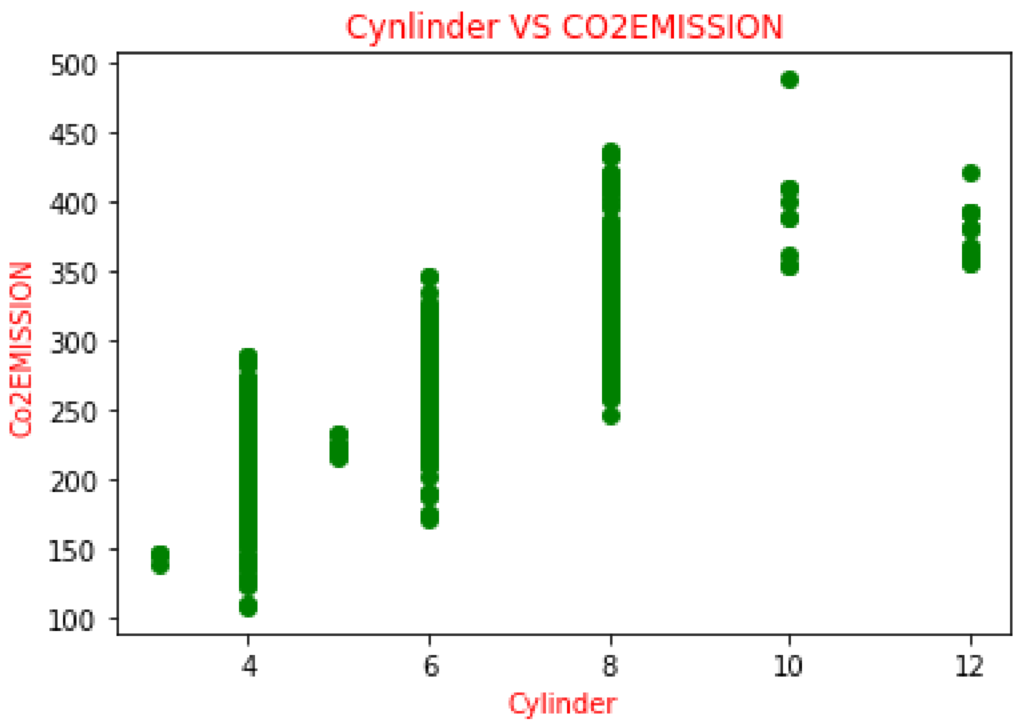
```
   MODELYEAR   MAKE       MODEL VEHICLECLASS  ENGINESIZE  CYLINDERS  \
0       2014  ACURA         ILX      COMPACT         2.0          4
1       2014  ACURA         ILX      COMPACT         2.4          4
2       2014  ACURA  ILX HYBRID      COMPACT         1.5          4
3       2014  ACURA     MDX 4WD  SUV - SMALL         3.5          6
4       2014  ACURA     RDX AWD  SUV - SMALL         3.5          6

  TRANSMISSION FUELTYPE  FUELCONSUMPTION_CITY  FUELCONSUMPTION_HWY  \
0          AS5        Z                   9.9                  6.7
1           M6        Z                  11.2                  7.7
2          AV7        Z                   6.0                  5.8
3          AS6        Z                  12.7                  9.1
4          AS6        Z                  12.1                  8.7

   FUELCONSUMPTION_COMB  FUELCONSUMPTION_COMB_MPG  CO2EMISSIONS
0                   8.5                        33           196
1                   9.6                        29           221
2                   5.9                        48           136
3                  11.1                        25           255
4                  10.6                        27           244
```

Q1 : Create a scatter plot between cylinder vs Co2Emission (green color)

```
data1 = data[['CYLINDERS']].values
data1

array([[4],
       [4],
       [4],
       ...,
       [6],
       [6],
       [6]], dtype=int64)

data2= data['CO2EMISSIONS'].values
data2

array([196, 221, 136, ..., 271, 260, 294], dtype=int64)
```

```
plt.scatter(data1, data2, c='g')
plt.xlabel("Cylinder", c='r')
plt.ylabel("Co2EMISSION", c='r')
plt.title("Cynlinder VS CO2EMISSION", c='r')
```

```
Text(0.5, 1.0, 'Cynlinder VS CO2EMISSION')
```

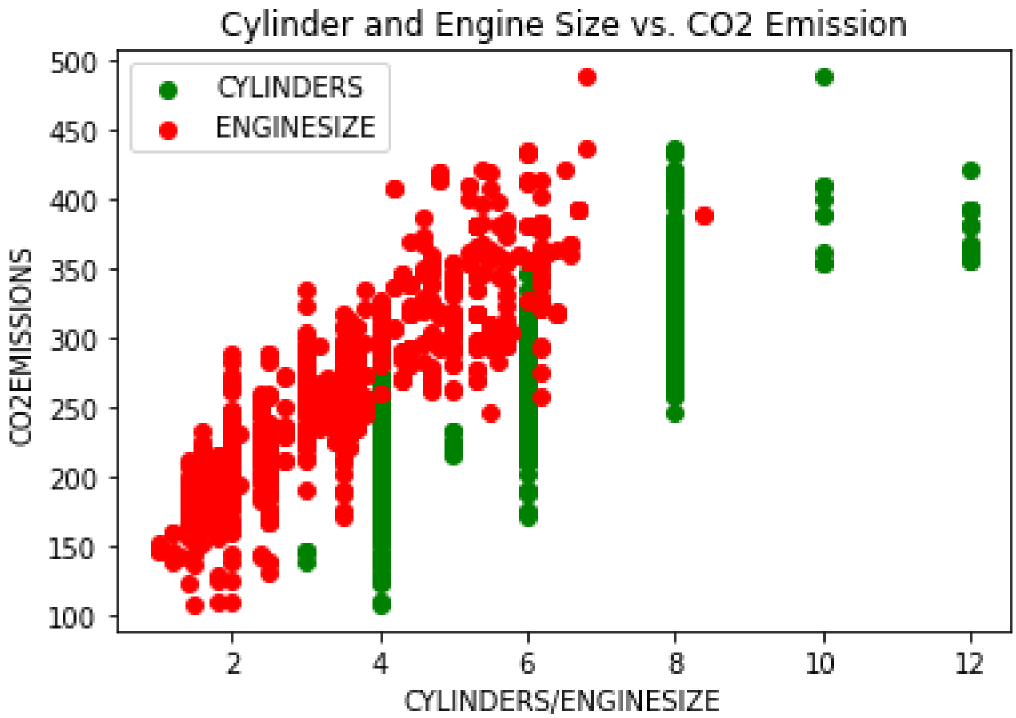


Q2 : using scatter plot compare data cylinder vs Co2Emission and Enginesize Vs Co2Emission using different colors

```
data3= data["ENGINESIZE"].values
data3
```

```
array([2. , 2.4, 1.5, ..., 3. , 3.2, 3.2])
```

```
plt.scatter(data1, data2, c='g', label='CYLINDERS')
plt.scatter(data3, data2, c='r', label= "ENGINESIZE")
plt.xlabel("CYLINDERS/ENGINESIZE", )
plt.ylabel("CO2EMISSIONS")
plt.title('Cylinder and Engine Size vs. CO2 Emission')
plt.legend()
plt.show()
```



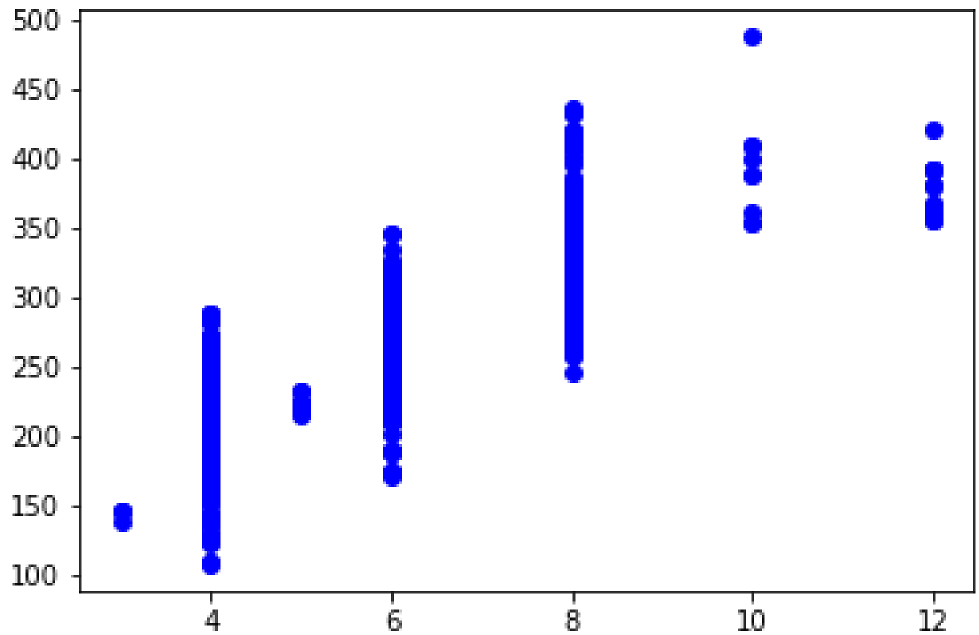
Q3 : using scatter plot compare data cylinder vs Co2Emission and Enginesize Vs Co2Emission and FuelConsumption_comb Co2Emission using different colors

```
data4= data["FUELCONSUMPTION_COMB"].values
data4
```

```
array([ 8.5,  9.6,  5.9, ..., 11.8, 11.3, 12.8])
```

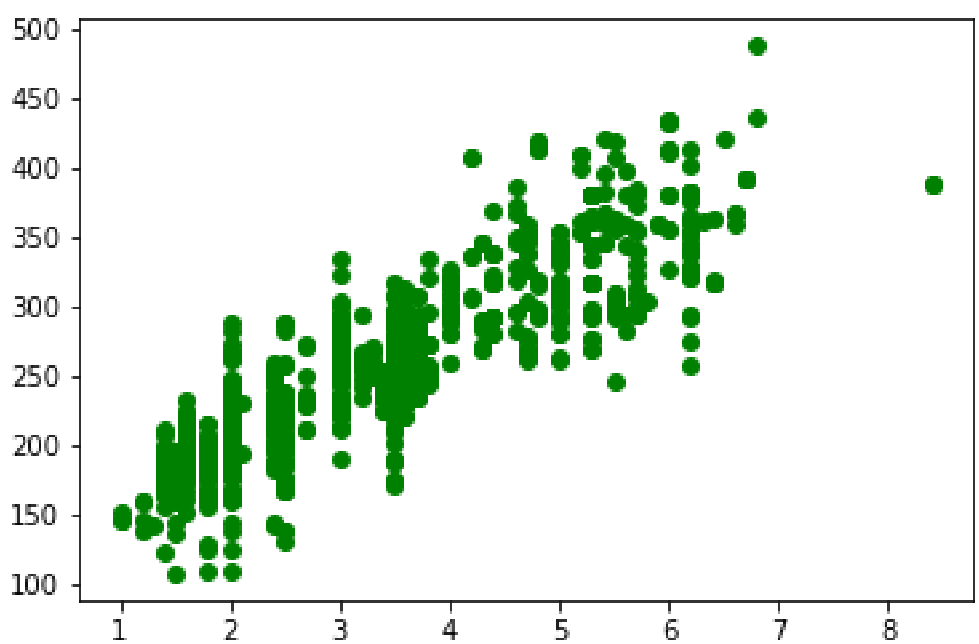
```
# Create a scatter plot for CYLINDERS vs CO2EMISSIONS
plt.scatter(data1, data2, c='blue', label='CYLINDERS vs CO2EMISSIONS')
```

```
<matplotlib.collections.PathCollection at 0x26893c2cac0>
```



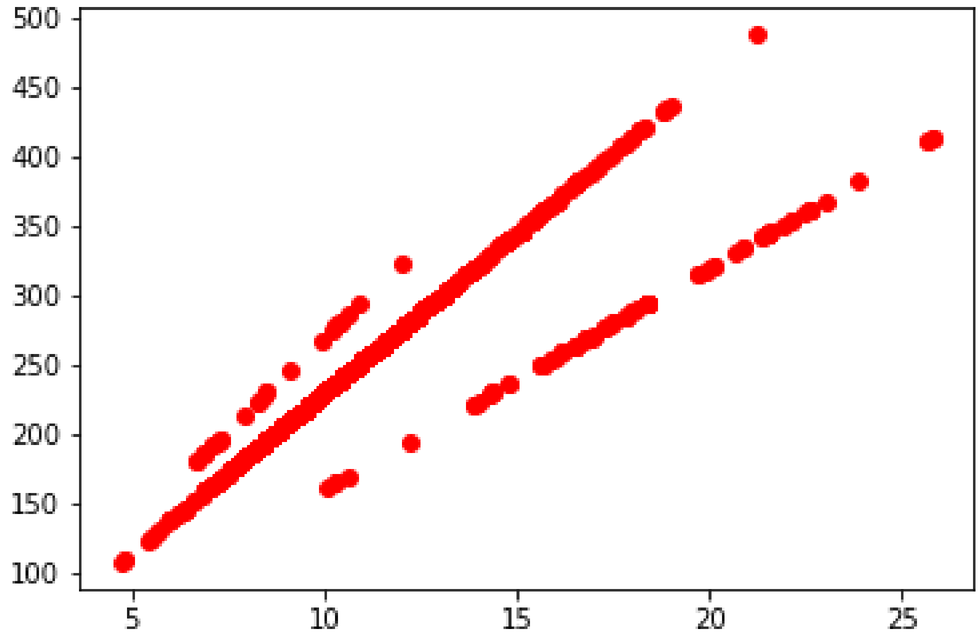

```
# Create a scatter plot for ENGINESIZE vs CO2EMISSIONS
plt.scatter(data3, data2, c='green', label='ENGINESIZE vs
CO2EMISSIONS')
```

```
<matplotlib.collections.PathCollection at 0x26893c9f7f0>
```

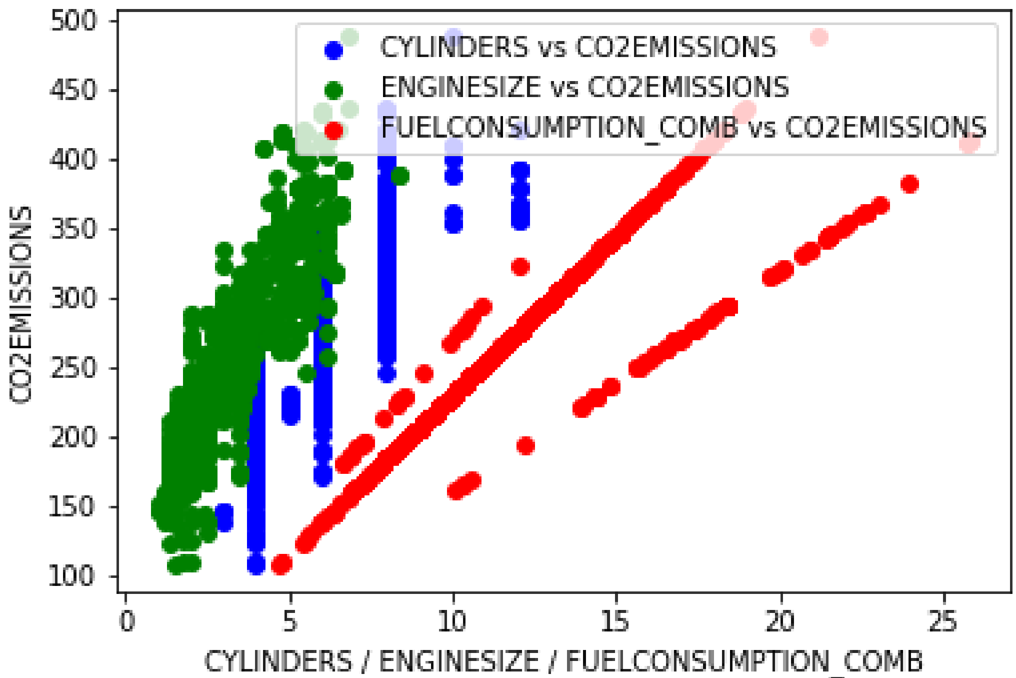



```
# Create a scatter plot for FUELCONSUMPTION_COMB vs CO2EMISSIONS
plt.scatter(data4, data2, c='red', label='FUELCONSUMPTION_COMB vs
CO2EMISSIONS')
```

```
<matplotlib.collections.PathCollection at 0x26893d0f970>
```




```python
import matplotlib.pyplot as plt

# Create a scatter plot for CYLINDERS vs CO2EMISSIONS
plt.scatter(data1, data2, c='blue', label='CYLINDERS vs CO2EMISSIONS')

# Create a scatter plot for ENGINESIZE vs CO2EMISSIONS
plt.scatter(data3, data2, c='green', label='ENGINESIZE vs
CO2EMISSIONS')

# Create a scatter plot for FUELCONSUMPTION_COMB vs CO2EMISSIONS
plt.scatter(data4, data2, c='red', label='FUELCONSUMPTION_COMB vs
CO2EMISSIONS')

# Add labels and legend
plt.xlabel('CYLINDERS / ENGINESIZE / FUELCONSUMPTION_COMB')
plt.ylabel('CO2EMISSIONS')
plt.legend()

# Show the plot
plt.show()
```



Questions Related to ML model Training

```
data.columns
```

```
Index(['MODELYEAR', 'MAKE', 'MODEL', 'VEHICLECLASS', 'ENGINESIZE',
'CYLINDERS',
       'TRANSMISSION', 'FUELTYPE', 'FUELCONSUMPTION_CITY',
       'FUELCONSUMPTION_HWY', 'FUELCONSUMPTION_COMB',
       'FUELCONSUMPTION_COMB_MPG', 'CO2EMISSIONS'],
      dtype='object')
```

```
data=data[['ENGINESIZE', "CYLINDERS", "FUELCONSUMPTION_COMB",
"CO2EMISSIONS"]]
```

```
data.head()
```

	ENGINESIZE	CYLINDERS	FUELCONSUMPTION_COMB	CO2EMISSIONS
0	2.0	4	8.5	196
1	2.4	4	9.6	221
2	1.5	4	5.9	136
3	3.5	6	11.1	255
4	3.5	6	10.6	244

Q4 : train your model with indepedent variable as cylinder and dependent variable as Co2Emission

```
X= data[['CYLINDERS']].values

X

array([[4],
       [4],
       [4],
       ...,
       [6],
       [6],
       [6]], dtype=int64)

y= data["CO2EMISSIONS"].values

y

array([196, 221, 136, ..., 271, 260, 294], dtype=int64)

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_state=50)

X_train.shape

(853, 1)

X_test.shape

(214, 1)

from sklearn.linear_model import LinearRegression

Lreg= LinearRegression()

Lreg.fit(X_train, y_train)

LinearRegression()

from sklearn.metrics import mean_squared_error, r2_score

y_pred= Lreg.predict(X_test)

mean_squared_error(y_pred, y_test)

1232.0265947622318

r2_score(y_pred, y_test)

0.5982384887400676
```

Q5 : Train another model with independent variable as FuelConsumption_comb and dependent variable as Co2Emission

```
X= data[["FUELCONSUMPTION_COMB"]].values
```

```
X
```

```
array([[ 8.5],
       [ 9.6],
       [ 5.9],
       ...,
       [11.8],
       [11.3],
       [12.8]])
```

```
y= data["CO2EMISSIONS"].values
```

```
y
```

```
array([196, 221, 136, ..., 271, 260, 294], dtype=int64)
```

```
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_state=50)
```

```
X_train.shape
```

```
(853, 1)
```

```
X_test.shape
```

```
(214, 1)
```

```
from sklearn.linear_model import LinearRegression
```

```
Lreg= LinearRegression()
```

```
Lreg.fit(X_train, y_train)
```

```
LinearRegression()
```

```
y_pred= Lreg.predict(X_test)
```

```
from sklearn.metrics import mean_squared_error, r2_score
```

```
mean_squared_error(y_pred, y_test)
```

```
740.7869638846828
```

```
r2_score(y_pred, y_test)
```

```
0.7630472129228487
```

```
```

Q6 : Train your model on different train test ratio and train the models and note down there accuracies

#training the model in a ration of 70- training and 30- testing

```
X= data[["FUELCONSUMPTION_COMB"]].values

y= data["CO2EMISSIONS"].values

X_train,X_test,y_train, y_test= train_test_split(X,y, test_size= 0.3,
random_state=50)

Lreg= LinearRegression()

Lreg.fit(X_train, y_train)

LinearRegression()

y_pred= Lreg.predict(X_test)

mean_squared_error(y_pred, y_test)

692.4736701217272

r2_score(y_pred, y_test)

0.7598960234281736
```

Q7 : we are providing you another dataset regarding housing prediction to need to apply Linear Regression on atleast 5 pairs of independent and dependent variable and store their accuracy and then make a plot of those accuracy

```
import pandas as pd
import numpy as np

Data= pd.read_csv('housing.csv')

Data.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')
```

```
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

#Selecting int datatypes only

Data = Data.select_dtypes(include='int')

Data.info()

<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1460 entries, 0 to 1459
Data columns (total 35 columns):
 #   Column         Non-Null Count  Dtype
---  ------         --------------  -----
 0   Id             1460 non-null   int64
 1   MSSubClass     1460 non-null   int64
 2   LotArea        1460 non-null   int64
 3   OverallQual    1460 non-null   int64
 4   OverallCond    1460 non-null   int64
 5   YearBuilt      1460 non-null   int64
 6   YearRemodAdd   1460 non-null   int64
 7   BsmtFinSF1     1460 non-null   int64
 8   BsmtFinSF2     1460 non-null   int64
 9   BsmtUnfSF      1460 non-null   int64
 10  TotalBsmtSF    1460 non-null   int64
 11  1stFlrSF       1460 non-null   int64
 12  2ndFlrSF       1460 non-null   int64
 13  LowQualFinSF   1460 non-null   int64
 14  GrLivArea      1460 non-null   int64
 15  BsmtFullBath   1460 non-null   int64
 16  BsmtHalfBath   1460 non-null   int64
 17  FullBath       1460 non-null   int64
 18  HalfBath       1460 non-null   int64
 19  BedroomAbvGr   1460 non-null   int64
 20  KitchenAbvGr   1460 non-null   int64
 21  TotRmsAbvGrd   1460 non-null   int64
 22  Fireplaces     1460 non-null   int64
 23  GarageCars     1460 non-null   int64
 24  GarageArea     1460 non-null   int64
```

```
 25  WoodDeckSF     1460 non-null   int64
 26  OpenPorchSF    1460 non-null   int64
 27  EnclosedPorch  1460 non-null   int64
 28  3SsnPorch      1460 non-null   int64
 29  ScreenPorch    1460 non-null   int64
 30  PoolArea       1460 non-null   int64
 31  MiscVal        1460 non-null   int64
 32  MoSold         1460 non-null   int64
 33  YrSold         1460 non-null   int64
 34  SalePrice      1460 non-null   int64
dtypes: int64(35)
memory usage: 399.3 KB
```

```
correlation_coefficient = Data['1stFlrSF'].corr(Data['SalePrice'])
print("Correlation coefficient:", correlation_coefficient)
```

```
Correlation coefficient: 0.6058521846919148
```

```
X = Data[['OverallQual', 'TotalBsmtSF', 'GrLivArea', 'GarageCars',
'GarageArea']].values #these columns are more relatively co- relative
to the Saleprice
```

```
X
```

```
array([[   7,  856, 1710,    2,  548],
       [   6, 1262, 1262,    2,  460],
       [   7,  920, 1786,    2,  608],
       ...,
       [   7, 1152, 2340,    1,  252],
       [   5, 1078, 1078,    1,  240],
       [   5, 1256, 1256,    1,  276]], dtype=int64)
```

```
Data.head()
```

```
   Id  MSSubClass  LotArea  OverallQual  OverallCond  YearBuilt
YearRemodAdd  \
0   1          60     8450            7            5       2003
2003
1   2          20     9600            6            8       1976
1976
2   3          60    11250            7            5       2001
2002
3   4          70     9550            7            5       1915
1970
4   5          60    14260            8            5       2000
2000

   BsmtFinSF1  BsmtFinSF2  BsmtUnfSF  ...  WoodDeckSF  OpenPorchSF  \
0         706           0        150  ...           0           61
1         978           0        284  ...         298            0
2         486           0        434  ...           0           42
```

```
3        216           0        540  ...           0           35
4        655           0        490  ...         192           84

   EnclosedPorch  3SsnPorch  ScreenPorch  PoolArea  MiscVal  MoSold
YrSold  \
0              0          0            0         0        0       2
2008
1              0          0            0         0        0       5
2007
2              0          0            0         0        0       9
2008
3            272          0            0         0        0       2
2006
4              0          0            0         0        0      12
2008

   SalePrice
0     208500
1     181500
2     223500
3     140000
4     250000

[5 rows x 35 columns]
```

```
y= Data['SalePrice'].values
```

```
y
```

```
array([208500, 181500, 223500, ..., 266500, 142125, 147500],
dtype=int64)
```

```
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_state= 50)
```

```
X_train.shape
```

```
(1168, 5)
```

```
X_test.shape
```

```
(292, 5)
```

```
from sklearn.linear_model import LinearRegression
```

```
Lreg= LinearRegression()
```

```
Lreg.fit(X_train, y_train)
```

```
LinearRegression()
```

```
y_pred= Lreg.predict(X_test)

from sklearn.metrics import mean_squared_error, r2_score

mean_squared_error(y_pred, y_test)

1083231779.1272151

r2_score(y_pred, y_test)

0.7696933064086375
```