x y z 326 3.95 3.98 2.43

326 3.89 3.84 2.31

327 4 05 4 07 2 31

334 4.20 4.23 2.63

335 4.34 4.35 2.75

```
5/20/24, 8:30 PM
   #Importing important libraries
   import numpy as np
   import pandas as pd
   import seaborn as sns
   import matplotlib.pyplot as plt
   from \ sklearn.preprocessing \ import \ Label Encoder, Standard Scaler
   from scipy import stats
   from sklearn.linear_model import LinearRegression,Lasso
   from sklearn.metrics import mean_squared_error,mean_absolute_error
   from sklearn.ensemble import RandomForestRegressor
   import warnings
   warnings.filterwarnings("ignore")
   #Impoting Data from .csv and saving in variable TD
   TD = pd.read_csv('/content/diamonds.csv')
   TD.head()# to show 1st 5 rows of data
           Unnamed: 0 carat
                                cut color clarity depth table price
         0
                    1
                        0.23
                                Ideal
                                          Ε
                                                 SI2
                                                      61.5
                                                             55.0
         1
                    2 0.21 Premium
                                          Ε
                                                 SI1
                                                      59.8
                                                             61.0
         2
                    3 0.23
                                          Е
                                                VS1 56.9
                                                           65.0
                                Good
         3
                    4 0.29 Premium
                                        VS2
                                                      62.4
                                                             58.0
                                                 SI2 63.3 58.0
                    5 0.31
                                Good
                                          - 1
```

Next steps: Generate code with TD View recommended plots

TD.shape # to show the rows and colmns of the data

(53940, 11)

TD.isnull().sum() # to show the sum of all null fields

Unnamed: 0 0 carat cut color clarity 0 depth table 0 price 0 0 0 0 dtype: int64

TD.describe() # to show summary statistics of the data

	Unnamed: 0	carat	depth	table	price	х	У	z	
count	53940.000000	53940.000000	53940.000000	53940.000000	53940.000000	53940.000000	53940.000000	53940.000000	
mean	26970.500000	0.797940	61.749405	57.457184	3932.799722	5.731157	5.734526	3.538734	
std	15571.281097	0.474011	1.432621	2.234491	3989.439738	1.121761	1.142135	0.705699	
min	1.000000	0.200000	43.000000	43.000000	326.000000	0.000000	0.000000	0.000000	
25%	13485.750000	0.400000	61.000000	56.000000	950.000000	4.710000	4.720000	2.910000	
50%	26970.500000	0.700000	61.800000	57.000000	2401.000000	5.700000	5.710000	3.530000	
75%	40455.250000	1.040000	62.500000	59.000000	5324.250000	6.540000	6.540000	4.040000	
max	53940.000000	5.010000	79.000000	95.000000	18823.000000	10.740000	58.900000	31.800000	

TD = TD.dropna() # this removes all data which contains null values, since there are no null values the data remains the same TD.drop(TD.columns[0], axis=1, inplace=True) # removing the 1st columnn since the its just numbering

TD.isnull().sum() # repeating previous step to recheck

carat 0 color 0 clarity

```
depth 0 table 0 price 0 x 0 y 0 z 0 dtype: int64
```

TD.dtypes # to show the datatypes of the data

float64 carat cut color object object clarity object float64 depth table float64 price int64 float64 float64 float64 dtype: object

# FDA

TD.describe() # to show summary statistics of the data

carat	depth	table	price	Х	у	z
nt 53940.000000	53940.000000	53940.000000	53940.000000	53940.000000	53940.000000	53940.000000
an 0.797940	61.749405	57.457184	3932.799722	5.731157	5.734526	3.538734
d 0.474011	1.432621	2.234491	3989.439738	1.121761	1.142135	0.705699
n 0.200000	43.000000	43.000000	326.000000	0.000000	0.000000	0.000000
% 0.400000	61.000000	56.000000	950.000000	4.710000	4.720000	2.910000
% 0.700000	61.800000	57.000000	2401.000000	5.700000	5.710000	3.530000
% 1.040000	62.500000	59.000000	5324.250000	6.540000	6.540000	4.040000
5.010000	79.000000	95.000000	18823.000000	10.740000	58.900000	31.800000
	nnt 53940.000000 an 0.797940 d 0.474011 in 0.200000 % 0.400000 % 0.700000 % 1.040000	nnt 53940.000000 53940.000000 an 0.797940 61.749405 d 0.474011 1.432621 in 0.200000 43.000000 % 0.400000 61.000000 % 0.700000 61.800000 % 1.040000 62.500000	nnt 53940.000000 53940.000000 53940.000000 an	int         53940.00000         53940.00000         53940.00000         53940.00000           an         0.797940         61.749405         57.457184         3932.799722           d         0.474011         1.432621         2.234491         3989.439738           in         0.200000         43.000000         43.000000         326.000000           %         0.40000         61.800000         57.000000         2401.000000           %         1.040000         62.500000         59.000000         5324.250000	int         53940.00000         53940.00000         53940.00000         53940.00000         53940.00000           an         0.797940         61.749405         57.457184         3932.799722         5.731157           d         0.474011         1.432621         2.234491         3989.439738         1.121761           in         0.200000         43.000000         43.000000         326.000000         0.000000           %         0.400000         61.800000         57.000000         2401.000000         5.700000           %         1.040000         62.500000         59.000000         5324.250000         6.540000	int         53940.00000         0.000000         0.000000         0.000000         0.000000         0.000000         0.000000         0.000000         0.000000         0.000000         0.000000         4.710000         4.720000         4.720000         5.700000         5.710000         5.710000         6.540

	carat	cut	color	clarity	depth	table	price	Х	У	Z
11182	1.07	Ideal	F	SI2	61.6	56.0	4954	0.0	6.62	0.0
11963	1.00	Very Good	Н	VS2	63.3	53.0	5139	0.0	0.00	0.0
15951	1.14	Fair	G	VS1	57.5	67.0	6381	0.0	0.00	0.0
24520	1.56	Ideal	G	VS2	62.2	54.0	12800	0.0	0.00	0.0
26243	1.20	Premium	D	VVS1	62.1	59.0	15686	0.0	0.00	0.0
27429	2.25	Premium	Н	SI2	62.8	59.0	18034	0.0	0.00	0.0
49556	0.71	Good	F	SI2	64.1	60.0	2130	0.0	0.00	0.0
49557	0.71	Good	F	SI2	64.1	60.0	2130	0.0	0.00	0.0
	carat	cut	color	clarity	depth	table	price	X	У	Z
11963	1.00	Very Good	Н	VS2	63.3	53.0	5139	0.0	0.0	0.0
15951	1.14	Fair	G	VS1	57.5	67.0	6381	0.0	0.0	0.0
24520	1.56	Ideal	G	VS2	62.2	54.0	12800	0.0	0.0	0.0
26243	1.20	Premium	D	VVS1	62.1	59.0	15686	0.0	0.0	0.0
27429	2.25	Premium	Н	SI2	62.8	59.0	18034	0.0	0.0	0.0
49556	0.71	Good	F	SI2	64.1	60.0	2130	0.0	0.0	0.0
49557	0.71	Good	F	SI2	64.1	60.0	2130	0.0	0.0	0.0
	carat	cut	color	clarity	depth	table	price	X	)	/ Z
2207	1.00	Premium	G	SI2	59.1	59.0	3142	6.55	6.48	0.0
2314	1.01	Premium	Н	I1	58.1	59.0	3167	6.66	6.60	0.0
4791	1.10	Premium	G	SI2	63.0	59.0	3696	6.50	6.47	7 0.0
5471	1.01	Premium	F	SI2	59.2	58.0	3837	6.50	6.47	7 0.0
10167	1.50	Good	G	I1	64.0	61.0	4731	7.15	7.04	1 0.0
11182	1.07	Ideal	F	SI2	61.6	56.0	4954	0.00	6.62	0.0
11963	1.00	Very Good	Н	VS2	63.3	53.0	5139	0.00	0.00	0.0
13601	1.15	Ideal	G	VS2	59.2	56.0	5564	6.88	6.83	0.0
15951	1.14	Fair	G	VS1	57.5	67.0	6381	0.00	0.00	0.0
24394	2.18	Premium	Н	SI2	59.4	61.0	12631	8.49	8.45	0.0
24520	1.56	Ideal	G	VS2	62.2	54.0	12800	0.00	0.00	0.0
26123	2.25	Premium	I	SI1	61.3	58.0	15397	8.52	8.42	0.0
26243	1.20	Premium	D	VVS1	62.1	59.0	15686	0.00	0.00	0.0
27112	2.20	Premium	Н	SI1	61.2	59.0	17265	8.42	8.37	7 0.0
27429	2.25	Premium	Н	SI2	62.8	59.0	18034	0.00	0.00	0.0
27503	2.02	Premium	Н	VS2	62.7	53.0	18207	8.02	7.95	0.0
27739	2.80	Good	G	SI2	63.8	58.0	18788	8.90	8.8	0.0
49556	0.71	Good	F	SI2	64.1	60.0	2130	0.00	0.00	0.0
49557	0.71	Good	F	SI2	64.1	60.0	2130	0.00	0.00	0.0
51506	1.12	Premium	G	I1	60.4	59.0	2383	6.71	6.67	7 0.0
	11963 15951 24520 26243 27429 49556 49557 11963 15951 24520 26243 27429 49556 49557 2207 2314 4791 10167 11182 11963 13601 15951 24394 24520 26123 26243 27112 27429 27429 27739 49556 49557	11182         1.07           11963         1.00           15951         1.14           24520         1.56           26243         1.20           27429         2.25           49556         0.71           49557         0.71           24520         1.56           26243         1.00           15951         1.14           24520         1.56           26243         1.20           27429         2.25           49556         0.71           49557         0.71           2207         1.00           2314         1.01           4791         1.01           5471         1.01           10167         1.50           11182         1.07           11963         1.00           13601         1.15           15951         1.14           24394         2.18           24520         1.56           26123         2.25           26243         1.20           27112         2.20           27503         2.02           277503         2.02	11182         1.07         Ideal           11963         1.00         Very Good           15951         1.14         Fair           24520         1.56         Ideal           26243         1.20         Premium           27429         2.25         Premium           49556         0.71         Good           49557         0.71         Good           49557         1.04         Fore           11963         1.00         Very Good           15951         1.14         Fair           24520         1.56         Ideal           26243         1.20         Premium           27429         2.25         Premium           27429         2.25         Premium           49556         0.71         Good           49557         0.71         Good           49556         0.71         Good           49557         0.71         Good           49557         1.00         Premium           207         1.00         Premium           2471         1.01         Premium           5471         1.00         Premium           5471	11182         1.07         Ideal         F           11963         1.00         Very Good         H           15951         1.14         Fair         G           24520         1.56         Ideal         G           26243         1.20         Premium         D           27429         2.25         Premium         H           49556         0.71         Good         F           49557         0.71         Good         F           carat         cut         color           11963         1.00         Very Good         H           15951         1.14         Fair         G           24520         1.56         Ideal         G           49556         0.71         Good         F           49556         0.71         Good         F           49556         0.71         Good         F           24749         2	11182         1.07         Ideal         F         SI2           11963         1.00         Very Good         H         VS2           15951         1.14         Fair         G         VS1           24520         1.56         Ideal         G         VS2           26243         1.20         Premium         D         VVS1           27429         2.25         Premium         H         SI2           49556         0.71         Good         F         SI2           49557         0.71         Good         F         SI2           49557         0.71         Good         F         SI2           1963         1.00         Very Good         H         VS2           19551         1.14         Fair         G         VS1           24520         1.56         Ideal         G         VS2           26243         1.20         Premium         D         VVS1           27429         2.25         Premium         H         SI2           49556         0.71         Good         F         SI2           49556         0.71         Good         F         SI2	11182         1.07         Ideal         F         SI2         61.6           11963         1.00         Very Good         H         VS2         63.3           15951         1.14         Fair         G         VS1         57.5           24520         1.56         Ideal         G         VS2         62.2           26243         1.20         Premium         D         VVS1         62.2           27429         2.25         Premium         H         SI2         64.1           49556         0.71         Good         F         SI2         64.1           49557         0.71         Good         F         SI2         64.1           1963         1.00         Very Good         H         VS2         63.3           15951         1.14         Fair         G         VS1         57.5           24520         1.56         Ideal         G         VS2         62.2           25423         1.20         Premium         D         VVS1         62.1           27429         2.25         Premium         H         SI2         64.1           49556         0.71         Good         F	11182         1.07         Ideal         F         SI2         61.6         56.0           11963         1.00         Very Good         H         VS2         63.3         53.0           15951         1.14         Fair         G         VS1         57.5         67.0           24520         1.56         Ideal         G         VS2         62.2         54.0           26243         1.20         Premium         D         VVS1         62.1         59.0           49556         0.71         Good         F         SI2         64.1         60.0           49557         0.71         Good         F         SI2         64.1         60.0           49557         0.71         Good         F         SI2         64.1         60.0           49557         0.71         Good         H         VS2         63.3         53.0           11963         1.00         Very Good         H         VS2         63.3         53.0           11963         1.01         Permium         D         VVS1         62.2         54.0           24520         1.56         Ideal         G         VS2         62.2         54.	11182         1.07         Ideal         F         SI2         61.6         56.0         4954           11963         1.00         Very Good         H         VS2         63.3         53.0         5139           15951         1.14         Fair         G         VS1         57.5         67.0         6381           24520         1.56         Ideal         G         VS2         62.2         54.0         12800           26243         1.20         Premium         H         SI2         62.1         59.0         15686           27429         2.25         Premium         H         SI2         62.1         59.0         18034           49556         0.71         Good         F         SI2         64.1         60.0         2130           49557         0.71         Good         F         SI2         64.1         60.0         2130           49557         0.71         Good         F         SI2         64.1         60.0         2130           1963         1.00         Veremium         D         VVS1         62.2         54.0         1280           19551         1.14         Fail         G         <	11182	1182

```
TD = TD[TD['x'] != 0] # removing data where x = 0
TD = TD[TD['y'] != 0] # removing data where x = 0
TD = TD[TD['z'] != 0] # removing data where x = 0
print(TD[TD['x'] == 0]) # rechecking for x values where x = 0
print(TD[TD['y'] == 0]) # rechecking for y values where y = 0
print(TD[TD['z'] == 0]) # rechecking for z values where z = 0

Empty DataFrame
    Columns: [carat, cut, color, clarity, depth, table, price, x, y, z]
    Index: []
    Empty DataFrame
    Columns: [carat, cut, color, clarity, depth, table, price, x, y, z]
    Index: []
    Empty DataFrame
    Columns: [carat, cut, color, clarity, depth, table, price, x, y, z]
    Index: []
    Index: []
```

## TD.head()

	carat	cut	color	clarity	depth	table	price	Х	у	z	
0	0.23	Ideal	Е	SI2	61.5	55.0	326	3.95	3.98	2.43	11.
1	0.21	Premium	Е	SI1	59.8	61.0	326	3.89	3.84	2.31	
2	0.23	Good	Е	VS1	56.9	65.0	327	4.05	4.07	2.31	
3	0.29	Premium	I	VS2	62.4	58.0	334	4.20	4.23	2.63	
4	0.31	Good	J	SI2	63.3	58.0	335	4.34	4.35	2.75	
	1 2 3	<ul><li>0 0.23</li><li>1 0.21</li><li>2 0.23</li><li>3 0.29</li></ul>	<ul><li>0 0.23 Ideal</li><li>1 0.21 Premium</li><li>2 0.23 Good</li><li>3 0.29 Premium</li></ul>	0       0.23       Ideal       E         1       0.21       Premium       E         2       0.23       Good       E         3       0.29       Premium       I	0         0.23         Ideal         E         SI2           1         0.21         Premium         E         SI1           2         0.23         Good         E         VS1           3         0.29         Premium         I         VS2	0         0.23         Ideal         E         SI2         61.5           1         0.21         Premium         E         SI1         59.8           2         0.23         Good         E         VS1         56.9           3         0.29         Premium         I         VS2         62.4	0         0.23         Ideal         E         SI2         61.5         55.0           1         0.21         Premium         E         SI1         59.8         61.0           2         0.23         Good         E         VS1         56.9         65.0           3         0.29         Premium         I         VS2         62.4         58.0	0         0.23         Ideal         E         SI2         61.5         55.0         326           1         0.21         Premium         E         SI1         59.8         61.0         326           2         0.23         Good         E         VS1         56.9         65.0         327           3         0.29         Premium         I         VS2         62.4         58.0         334	0         0.23         Ideal         E         SI2         61.5         55.0         326         3.95           1         0.21         Premium         E         SI1         59.8         61.0         326         3.89           2         0.23         Good         E         VS1         56.9         65.0         327         4.05           3         0.29         Premium         I         VS2         62.4         58.0         334         4.20	0         0.23         Ideal         E         SI2         61.5         55.0         326         3.95         3.98           1         0.21         Premium         E         SI1         59.8         61.0         326         3.89         3.84           2         0.23         Good         E         VS1         56.9         65.0         327         4.05         4.07           3         0.29         Premium         I         VS2         62.4         58.0         334         4.20         4.23	1       0.21       Premium       E       SI1       59.8       61.0       326       3.89       3.84       2.31         2       0.23       Good       E       VS1       56.9       65.0       327       4.05       4.07       2.31

Next steps: Generate code with TD View recommended plots

 $TD. describe (include='object') \ \# \ to \ show \ summary \ of \ objects/categories \ which \ without \ specifying \ gives \ summary \ without \ objects$ 

```
        cut
        color
        clarity

        count
        53920
        53920
        53920

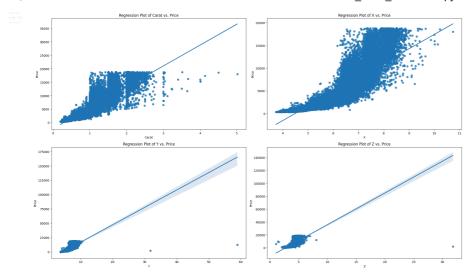
        unique
        5
        7
        8

        top
        Ideal
        G
        SI1

        freq
        21548
        11284
        13063
```

```
#Regression plot against all the categories
features = ['carat', 'x', 'y', 'z']
plt.figure(figsize=(20, 12))
for i, feature in enumerate(features):
    plt.subplot(2, 2, i + 1)
    sns.regplot(x=feature, y='price', data=TD)
    plt.title(f'Regression Plot of {feature.capitalize()} vs. Price')
    plt.xlabel(feature.capitalize())
    plt.ylabel('Price')

plt.tight_layout()
plt.show()
```

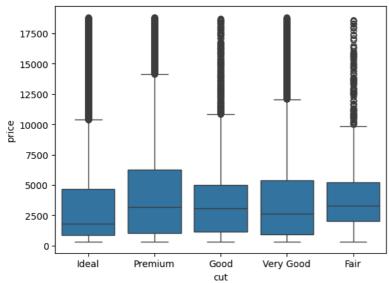


```
# Pearson Correlation Coefficient to show linear relationship
for feature in features:
    pcf, pvv = stats.pearsonr(TD[feature], TD['price'])
    print("The Pearson Correlation Coefficient is", pcf, " with a P-value of P =", pvv," for ",feature)

The Pearson Correlation Coefficient is 0.9215920634723974 with a P-value of P = 0.0 for carat
    The Pearson Correlation Coefficient is 0.887231372527643 with a P-value of P = 0.0 for x
    The Pearson Correlation Coefficient is 0.867864244674355 with a P-value of P = 0.0 for y
    The Pearson Correlation Coefficient is 0.8682064012988696 with a P-value of P = 0.0 for z

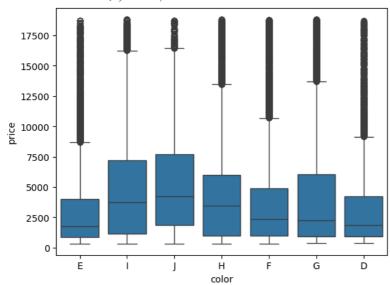
# box plot for the quality of cut
sns.boxplot(x='cut',y='price',data = TD)
```

<Axes: xlabel='cut', ylabel='price'>



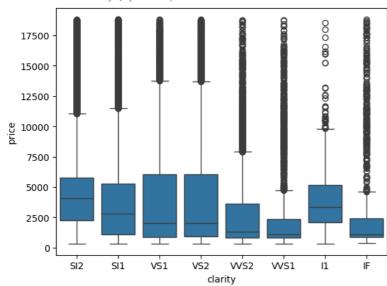
# box plot for color of the diamond
sns.boxplot(x='color',y='price',data = TD)





# box plot for the clarity of the diamond
sns.boxplot(x='clarity',y='price',data = TD)





```
5/20/24, 8:30 PM
                                                           Diamond Price Prediction.ipynb - Colab
   #nothing to remove since all data is required and affects the price of the diamond
    (53920, 10)
   # Data Transformation
   labelencoder = LabelEncoder()
   TD.cut = labelencoder.fit_transform(TD.cut)
   TD.color = labelencoder.fit_transform(TD.color)
   TD.clarity = labelencoder.fit_transform(TD.clarity)
   TD.head(10) # to check if label encoding is applied
           carat cut color clarity depth table price
                                                          Х
                                                             У
                                                                    Z
            0.23
                                  3
                                      61.5
                                             55.0
                                                    326 3.95 3.98 2.43
         1
             0.21
                    3
                                  2
                                      59.8
                                             61.0
                                                    326 3.89 3.84 2.31
         2
             0.23
                          1
                                      56.9
                                             65.0
                                                   327 4 05 4 07 2 31
                    1
                                  4
             0.29
                    3
                                      62.4
                                             58.0
                                                    334 4.20 4.23 2.63
                                  5
         4
             0.31
                    1
                          6
                                  3
                                      63.3
                                             58.0
                                                   335 4.34 4.35 2.75
         5
             0.24
                    4
                          6
                                  7
                                      62.8
                                             57.0
                                                   336 3.94 3.96 2.48
         6
             0.24
                    4
                                  6
                                      62.3
                                             57.0
                                                    336 3.95 3.98 2.47
                                  2
                                             55.0
             0.26
                          4
                                      61.9
                                                   337 407 411 253
                    4
         8
             0.22
                                  5
                                      65.1
                                             61.0
                                                    337 3.87 3.78 2.49
                    0
         q
             0.23
                    4
                                  4
                                      59 4
                                             61.0
                                                   338 400 405 239
    Next steps: Generate code with TD
                                      View recommended plots
   # normalization
   traindata = stats.zscore(TD)
   traindata
                                     color clarity
                                                                 table
                  carat
                             cut
                                                       depth
                                                                          price
               -1.198204 -0.538173 -0.936971 -0.484445 -0.174203 -1.099725 -0.904132 -1.5915
           1
               1.585988 -0.904132 -1.6451
               -1.198204 -1.511224 -0.936971 0.095422 -3.385781
                                                              3.376463 -0.903881 -1.5022
           2
           3
               -1.029353 -1.511224 2.002033 -0.484445
           4
                                                    1.082501 0.243131 -0.901875 -1.2431
         53935 -0.163993 -0.538173 -1.524772 -1.064312 -0.662921 -0.204488 -0.294437 0.0164
               -0.163993 -1.511224 -1.524772 -1.064312 0.942868 -1.099725 -0.294437 -0.0371
         53936
         53937 -0.206205 1.407928 -1.524772 -1.064312 0.733417 1.138369 -0.294437 -0.0639
         53938
               0.131496  0.434877  0.826431  -0.484445  -0.523288  0.243131  -0.294437  0.3737
         53939 -0.100674 -0.538173 -1.524772 -0.484445 0.314515 -1.099725 -0.294437 0.0878
        53920 rows × 10 columns
```

```
View recommended plots
 Next steps: Generate code with traindata
# dividing the data wherein xt retains all the features while yt retains the target (price)
xt = traindata.drop('price',axis =1)
yt = traindata['price']
```

xt.head() # checking if data is split properly

```
carat
                              color clarity
                                                 depth
                                                          table
      0 -1.198204 -0.538173 -0.936971 -0.484445 -0.174203 -1.099725 -1.591573 -1.539219
      1.585988 -1.645173 -1.662014 -
       2 \quad \text{-1.198204} \quad \text{-1.511224} \quad \text{-0.936971} \quad 0.095422 \quad \text{-3.385781} \quad 3.376463 \quad \text{-1.502241} \quad \text{-1.460280} \quad \cdot 
      4 -1.029353 -1.511224 2.002033 -0.484445 1.082501 0.243131 -1.243176 -1.214690 ·
______
 Next steps: Generate code with xt
                                   View recommended plots
yt.head() # checking if data is split properly
    0
        -0.904132
        -0.904132
        -0.903881
        -0.902125
        -0.901875
     Name: price, dtype: float64
from sklearn.model_selection import train_test_split #importing important library for training a model, to split the data properly acco
Xt,Xtt,Yt,Ytt = train_test_split(xt,yt,test_size=0.3,random_state=42)
# Multiple Linear Regression fitting training data into model
model = LinearRegression()
model mlr = model.fit(Xt,Yt)
# making prediction after fitting data into mdoel
YpMLR = model_mlr.predict(Xtt)
# calculating the mean square error of the model with actual data
mse MLR = mean squared error(Ytt,YpMLR)
print('The mean squared error for Multiple Linear Regression: ',mse_MLR)
 The mean squared error for Multiple Linear Regression: 0.11096621872138686
# calculating the mean absolute error of the model with actual data
mae_MLR = mean_absolute_error(Ytt,YpMLR)
print('The mean absolute error for Multiple Linear Regression: ',mae_MLR)
 The mean absolute error for Multiple Linear Regression: 0.21303238468283664
# fitting data into Random forest
rfModel = RandomForestRegressor()
model_rf = rfModel.fit(Xt,Yt)
# making prediction after fitting data
Yprf = model_rf.predict(Xtt)
# calculating the mean square error of the model with actual data
mse_RF = mean_squared_error(Ytt,Yprf)
print('The mean squared error for Random Forest: ',mse_RF)
 The mean squared error for Random Forest: 0.017906355767287833
# calculating the mean absolute error of the model with actual data
mae_RF = mean_absolute_error(Ytt,Yprf)
print('The mean absolute error for Random Forest: ',mae_RF)
 The mean absolute error for Random Forest: 0.06718954037281256
# fitting data into LASSO model
lassoModel = lasso()
model_ls = LassoModel.fit(Xt,Yt)
# making prediction using testing data
YpLs = model_ls.predict(Xtt)
# calculating the mean square error of the model with actual data
mse_Ls = mean_squared_error(Ytt,YpLs)
print('The mean squared error for Lasso: ',mse Ls)
```

0.0

LASSO

```
The mean squared error for Lasso: 0.9810294848199543
# calculating the mean absolute error of the model with actual data
mae_Ls = mean_absolute_error(Ytt,YpLs)
print('The mean absolute error for Lasso: ',mae_Ls)
The mean absolute error for Lasso: 0.7532785276594522
# saving all the scores of mean squared error
scoresSE = [('MLR',mse_MLR),('Random Forest',mse_RF),('LASSO',mse_Ls)]
# saving all the scores of mean absolute error
scoresAE = [('MLR',mae_MLR),('Random Forest',mae_RF),('LASSO',mae_Ls)]
# Visualizing MSE and MAE in table format
mse = pd.DataFrame(data = scoresSE,columns=['Model','MSE Score'])
mae = pd.DataFrame(data = scoresAE,columns=['Model','MAE Score'])
cdf = pd.merge(mse,mae,on='Model')
cdf
                Model MSE Score MAE Score
      0
                 MLR
                        0.110966
                                   0.213032
      1 Random Forest
                        0.017906
                                   0.067190
               LASSO
                        0.981029
                                   0.753279
 Next steps:
             Generate code with cdf
                                       View recommended plots
# Visualizing findings in diagram
mse.sort_values(by=(['MSE Score']), ascending=False, inplace=True)
f,axe = plt.subplots(1,1,figsize=(10,7))
sns.barplot(x=mse['Model'],y=mse['MSE Score'],ax = axe)
axe.set_xlabel('Model',size=20)
axe.set_ylabel('Mean Squared Error',size=20)
plt.show
       matplotlib.pyplot.show
       def show(*args, **kwargs)
       /usr/local/lib/python3.10/dist-packages/matplotlib/pyplot.py
       Display all open figures.
       Parameters
       block : bool, optional
         1.0
         0.8
      Mean Squared Error
         0.2
```

MLR

Model

Random Forest

```
# Visualizing findings in diagram
mae.sort_values(by=(['MAE Score']), ascending=False, inplace=True)
f,axe = plt.subplots(1,1,figsize=(10,7))
sns.barplot(x=mae['Model'],y=mae['MAE Score'],ax = axe)
axe.set_xlabel('Model',size=20)
axe.set_ylabel('Mean Abdolute Error',size=20)
plt.show
matplotlib.pyplot.show
```