In [1]:
```
import pandas as pd import
numpy as np import
matplotlib.pyplot as plt
```

In [2]:
```
df = pd.read_excel('Linear regression project.xlsx')
df.head()
```

Out[2]:

	ID	Amount.Requested	Amount.Funded.By.Investors	Interest.Rate	Loan.Length	Loan.Purpose	Debt.To.Income.Ratio	Home.Ownership	Monthly.Inco
0	81174.0	20000	20000.0	0.09	36 months	debt_consolidation	0.15	MORTGAGE	654
1	99592.0	19200	19200.0	0.12	36 months	debt_consolidation	0.28	MORTGAGE	458
2	80059.0	35000	35000.0	0.22	60 months	debt_consolidation	0.24	MORTGAGE	1150
3	15825.0	10000	9975.0	0.10	36 months	debt_consolidation	0.14	MORTGAGE	383
4	33182.0	12000	12000.0	0.12	36 months	credit_card	0.19	RENT	319

In [3]:
```
df.dtypes
```

Out[3]:

```
ID                          float64
Amount.Requested              int64
Amount.Funded.By.Investors    float64
Interest.Rate                 float64
Loan.Length                    object
Loan.Purpose                   object
Debt.To.Income.Ratio          float64
Home.Ownership                 object
Monthly.Income                float64
Open.CREDIT.Lines               int64
Revolving.CREDIT.Balance        int64
Inquiries.in.the.Last.6.Months  int64
Employment.Length              object dtype:
object
```

In [4]:
```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 2500 entries, 0 to 2499
Data columns (total 13 columns):
 #  Column                      Non-Null Count  Dtype
---  ------                     --------------  -----
0          ID                      2499 non-null  float64
1          Amount.Requested        2500 non-null  int64
2          Amount.Funded.By.Investors  2500 non-null  float64
3          Interest.Rate           2500 non-null  float64
4          Loan.Length             2499 non-null  object
5          Loan.Purpose            2499 non-null  object
6          Debt.To.Income.Ratio    2500 non-null  float64
7          Home.Ownership          2499 non-null  object
8          Monthly.Income          2500 non-null  float64
9          Open.CREDIT.Lines       2500 non-null  int64
10         Revolving.CREDIT.Balance  2500 non-null  int64
11         Inquiries.in.the.Last.6.Months  2500 non-null  int64  12  Employment.Length        2422 non-null  object dtypes: float64(5), int64(4), object(4) memory
usage: 254.0+ KB
```

In [5]:
```
df.isna().sum()
```

Out[5]:

```
ID                          1
Amount.Requested            0
Amount.Funded.By.Investors  0
Interest.Rate               0
Loan.Length                 1
Loan.Purpose                1
Debt.To.Income.Ratio        0
Home.Ownership              1
Monthly.Income              0
Open.CREDIT.Lines           0
Revolving.CREDIT.Balance    0
```

Inquiries.in.the.Last.6.Months 0

	ID	Amount.Requested	Amount.Funded.By.Investors	Interest.Rate	Loan.Length	Loan.Purpose	Debt.To.Income.Ratio	Home.Ownership	Monthly
0	81174.0	20000	20000.00	0.09	36 months	debt_consolidation	0.15	MORTGAGE	
1	99592.0	19200	19200.00	0.12	36 months	debt_consolidation	0.28	MORTGAGE	
2	80059.0	35000	35000.00	0.22	60 months	debt_consolidation	0.24	MORTGAGE	1

Employment.Length 78 dtype:
int64

In [6]:
df

Out[6]:

3	15825.0	10000	9975.00	0.10	36 months	debt_consolidation	0.14	MORTGAGE
4	33182.0	12000	12000.00	0.12	36 months	credit_card	0.19	RENT
...	...	...	...	...	...	...	...	...
2495	23735.0	30000	29950.00	0.17	60 months	debt_consolidation	0.19	MORTGAGE
2496	65882.0	16000	16000.00	0.14	60 months	home_improvement	0.22	OWN
2497	55610.0	10000	10000.00	0.14	36 months	debt_consolidation	0.05	MORTGAGE
2498	38576.0	6000	6000.00	0.12	36 months	major_purchase	0.17	RENT
2499	3116.0	9000	5242.75	0.14	36 months	debt_consolidation	0.07	RENT

2500 rows × 13 columns

In [7]:

```
df.nunique()
```

Out[7]:

```
ID                          2499
Amount.Requested             380
Amount.Funded.By.Investors   708
Interest.Rate                 21
Loan.Length                    3
Loan.Purpose                  14
Debt.To.Income.Ratio          36
Home.Ownership                 5
Monthly.Income               632
Open.CREDIT.Lines             29
Revolving.CREDIT.Balance    2349
Inquiries.in.the.Last.6.Months  10
Employment.Length             10 dtype: int64
```

In [8]:

```
gp=df.groupby('Amount.Requested').size()
gp.head()
```

Out[8]:

```
Amount.Requested
1000   18
1125    1
1200    6
1400    3 1450    1 dtype: int64
```

In [9]:

```
df.drop_duplicates(subset=['Amount.Requested'
df
```

Out[9]:

	ID	Amount.Requested	Amount.Funded.By.Investors	Interest.Rate	Loan.Length	Loan.Purpose	Debt.To.Income.Ratio	Home.Ownership
0	81174.0	20000		0.09	36 months	debt_consolidation	0.15	MORTGAGE
1	99592.0	19200		0.12	36 months	debt_consolidation	0.28	MORTGAGE
2	80059.0	35000		0.22	60 months	debt_consolidation	0.24	MORTGAGE
3	15825.0	10000		0.10	36 months	debt_consolidation	0.14	MORTGAGE
4	33182.0	12000		0.12	36 months	credit_card	0.19	RENT
...	...	...		...	...	...	...	...
2447	79820.0	25875		0.18	60 months	debt_consolidation	0.14	MORTGAGE
2466	92038.0	22750		0.22	60 months	home_improvement	0.29	OWN
2472	49533.0	17300		0.22	60 months	wedding	0.04	MORTGAGE
2489	63256.0	19075		0.19	36 months	debt_consolidation	0.15	RENT
2491	78043.0	8475		0.08	36 months	debt_consolidation	0.16	RENT

380 rows × 13 columns

],inplace=**True**)

In [10]:
```
df.describe()
```

Out[10]:

In [11]:
```
df.corr()
```

Out[11]:

In [12]:
```
import seaborn as sns
```

In [13]:

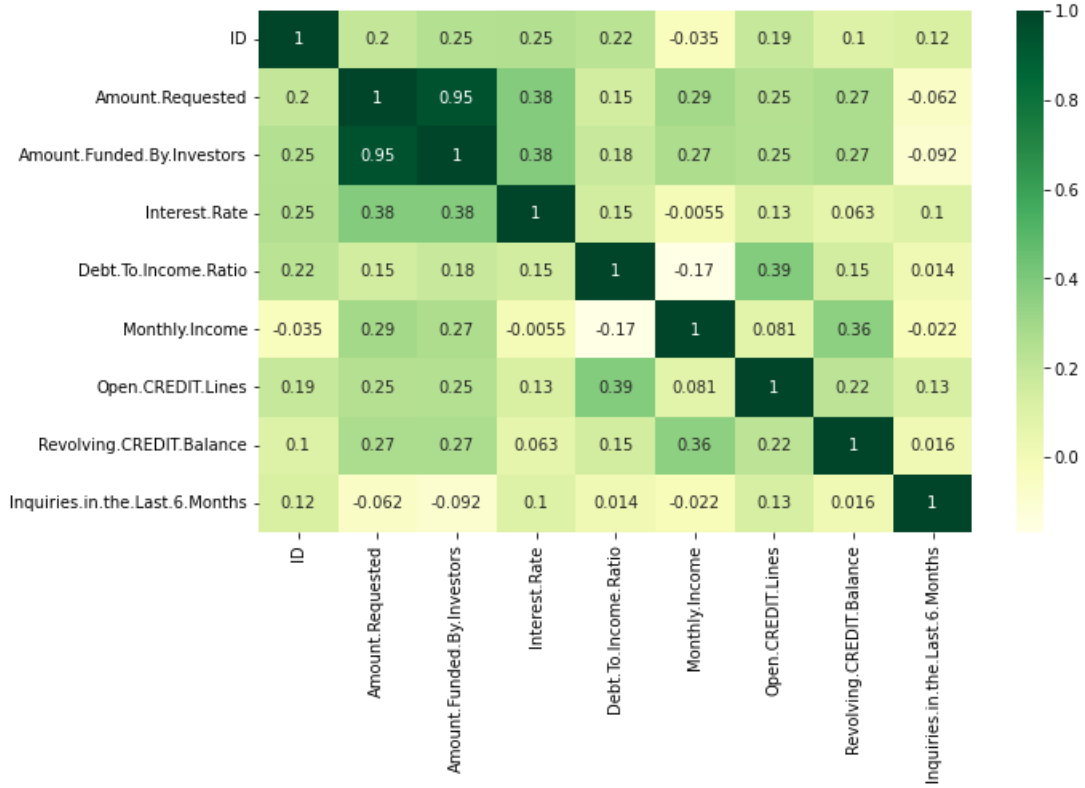
	ID	Amount.Requested	Amount.Funded.By.Investors	Interest.Rate	Debt.To.Income.Ratio	Monthly.Income	Open.CREDIT.Lines	Revolving.CRE
count	380.000000	380.000000	380.000000	380.000000	380.000000	380.000000	380.000000	
mean	56423.807895	13231.973684	12799.127105	0.136421	0.165658	5348.496895	10.057895	1
std	28890.272016	8239.491632	8310.901170	0.043292	0.075625	5682.379787	4.329053	1
min	17.000000	1000.000000	200.000000	0.050000	0.000000	666.670000	2.000000	
25%	33037.750000	6562.500000	6056.250000	0.100000	0.110000	3333.330000	7.000000	
50%	58928.000000	11912.500000	11362.500000	0.140000	0.160000	4583.330000	9.000000	1
75%	79733.000000	18631.250000	18406.250000	0.170000	0.220000	6025.000000	13.000000	1
max	103218.000000	35000.000000	35000.000000	0.240000	0.340000	102750.000000	25.000000	21

```
plt.figure(figsize=(10,6))
sns.heatmap(df.corr(),annot=True,cmap='YlGn')
```

Out[13]:

<AxesSubplot:>

	ID	Amount.Requested	Amount.Funded.By.Investors	Interest.Rate	Debt.To.Income.Ratio	Monthly.Income	Open.CREDIT.Line
ID	1.000000	0.197388	0.247795	0.254870	0.224371	-0.034803	0.18800
Amount.Requested	0.197388	1.000000	0.946643	0.381223	0.145515	0.293385	0.24522
Amount.Funded.By.Investors	0.247795	0.946643	1.000000	0.383232	0.181764	0.268437	0.24714
Interest.Rate	0.254870	0.381223	0.383232	1.000000	0.150943	-0.005498	0.12626
Debt.To.Income.Ratio	0.224371	0.145515	0.181764	0.150943	1.000000	-0.168431	0.38810
Monthly.Income	-0.034803	0.293385	0.268437	-0.005498	-0.168431	1.000000	0.08115
Open.CREDIT.Lines	0.188001	0.245228	0.247140	0.126266	0.388106	0.081158	1.00000
Revolving.CREDIT.Balance	0.104385	0.273258	0.270189	0.063045	0.149911	0.355870	0.21693
Inquiries.in.the.Last.6.Months	0.124743	-0.061913	-0.092030	0.103001	0.014198	-0.021957	0.13369

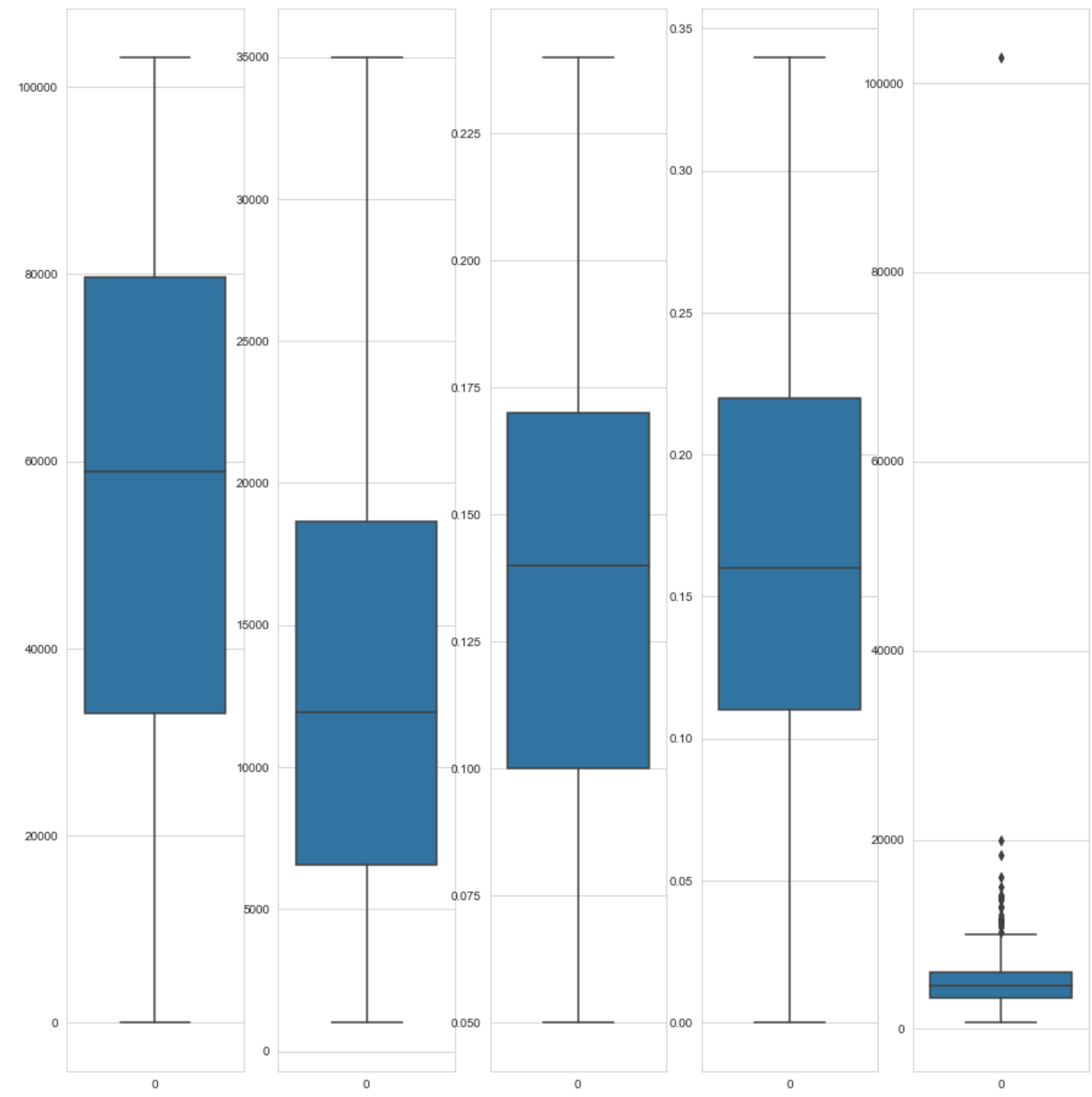


In [14]:

```
plt.figure(figsize=(15,16))
sns.set_style('whitegrid') plt.subplot(1,5,1)
sns.boxplot(data=df['ID']) plt.subplot(1,5,2)
sns.boxplot(data=df['Amount.Requested'])
plt.subplot(1,5,3)
sns.boxplot(data=df['Interest.Rate'])
plt.subplot(1,5,4)
sns.boxplot(data=df['Debt.To.Income.Ratio']
) plt.subplot(1,5,5)
sns.boxplot(data=df['Monthly.Income'])
```

Out[14]:

<AxesSubplot:>



In [15]:
```
df.shape
```

Out[15]:

(380, 13)

In [16]:
```
df.isna()
```

Out[16]:

	ID	Amount.Requested	Amount.Funded.By.Investors	Interest.Rate	Loan.Length	Loan.Purpose	Debt.To.Income.Ratio	Home.Ownership	Monthly.Incom
0	False	False	False	False	False	False	False	False	Fals
1	False	False	False	False	False	False	False	False	Fals
2	False	False	False	False	False	False	False	False	Fals
3	False	False	False	False	False	False	False	False	Fals
4	False	False	False	False	False	False	False	False	Fals
...	...	...	...	...	...	...	...	...	...
2447	False	False	False	False	False	False	False	False	Fals
2466	False	False	False	False	False	False	False	False	Fals
2472	False	False	False	False	False	False	False	False	Fals

2489	False	False	False	False	False	False	False	False	Fals
2491	False	False	False	False	False	False	False	False	Fals

380 rows × 13 columns

In [17]:
```
x=pd.DataFrame(data=df['Amount.Requested'],columns=('Amount.Requested',)) x.head()
```

Out[17]:

	Amount.Requested
0	20000
1	19200
2	35000
3	10000
4	12000

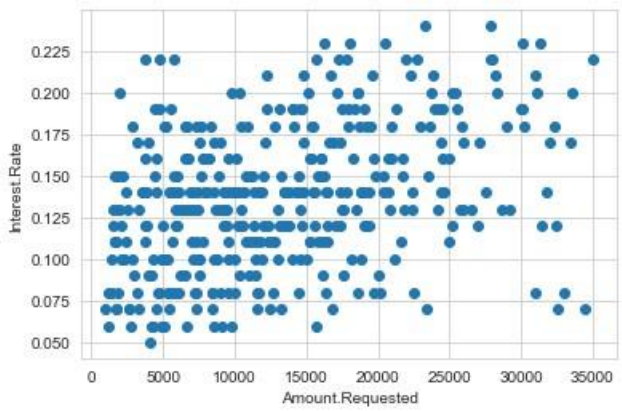
In [18]:
```
y=pd.DataFrame(data=df['Interest.Rate'],columns=('Interest.Rate',)) y.head()
```

Out[18]:

	Interest.Rate
0	0.09
1	0.12
2	0.22
3	0.10
4	0.12

In [19]:
```
plt.scatter(x,y) plt.xlabel("Amount.Requested")
plt.ylabel("Interest.Rate")
```

Out[19]: Text(0, 0.5, 'Interest.Rate')



In [20]:
```
from sklearn.model_selection import train_test_split x_train,x_test,y_train,y_test =
train_test_split(x,y,train_size=0.2,
                                 random_state=4)
```

In [21]:
```
from sklearn.linear_model import LinearRegression
```

In [22]:
```
lr = LinearRegression()
```

In [23]:
```
lr.fit(x_train,y_train)
```

Out[23]:

LinearRegression()

In [24]:
```
x_test
```

Out[24]: **Amount.Requested**

2472	17300
17	14000
1186	7900
29	14500
316	31075
...	...
393	21250
10	2000
170	11500
1508	31500
42	5500

304 rows × 1 columns

In [25]:
```
y_test
```

Out[25]:

	Interest.Rate
2472	0.22
17	0.10
1186	0.14
29	0.08
316	0.20
...	...
393	0.19
10	0.20
170	0.11
1508	0.12
42	0.19

304 rows × 1 columns

In [26]:
```
x_train
```

Out[26]:

	Amount.Requested
2380	15575
163	2800
1629	14100
538	1500
80	7450
...	...
218	15200
2239	5175
743	19600
604	2400
382	18400

76 rows × 1 columns

In [27]:
```
y_train
```

Out[27]:

	Interest.Rate
2380	0.14
163	0.10
1629	0.19
538	0.13

80	0.13
...	...
218	0.11
2239	0.18
743	0.21
604	0.11
382	0.19

76 rows × 1 columns

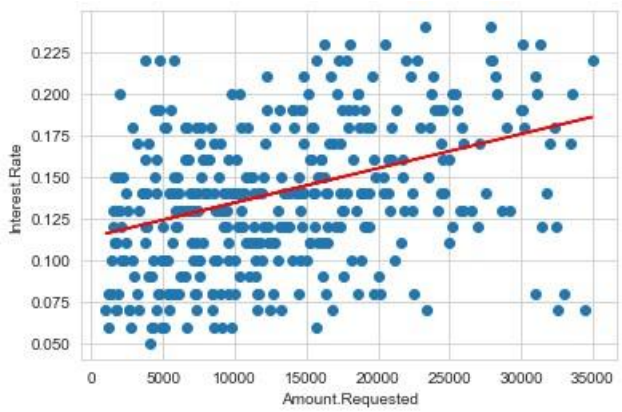
In [28]:
```
lr.predict(x_test.iloc[0].values.reshape(1,1))
```

```
C:\Users\kiransindam\AppData\Local\Programs\Python\Python310\lib\site-packages\sklearn\base.py:450: UserWarning: X does not have valid feature names
, but LinearRegression was fitted with feature names  warnings.warn(
```

Out[28]:
```
array([[0.14950488]])
```

In [29]:
```
plt.scatter(df['Amount.Requested'],df['Interest.Rate'])
plt.plot(x_test,lr.predict(x_test),color='red')
plt.xlabel('Amount.Requested') plt.ylabel('Interest.Rate')
```

Out[29]: Text(0, 0.5, 'Interest.Rate')



In [30]:
```
lr.coef_
```

Out[30]:
```
array([[2.0621461e-06]])
```

In [31]:
```
lr.intercept_
```

Out[31]:
```
array([0.11382976])
```

In [32]:
```
#y=mx + b
#m*17300 +b
```

In [33]:
```
y_predicte= lr.predict(x_train) y_predicte
```

Out[33]:

```
array([[0.14594768],
[0.11960377],
       [0.14290602],
       [0.11692298],
       [0.12919274],
       [0.16409457],
       [0.12955362],
       [0.12187213],
       [0.12310941],
[0.13393568],
       [0.1334717 ],
       [0.18054018],
       [0.13919415],
       [0.14331845],
       [0.12816167],
       [0.12166591],
       [0.14599924],
       [0.15862988],
       [0.17120897],
       [0.14068921],
       [0.14744274],
       [0.14780361],
       [0.15383539],
       [0.13032693],
       [0.11769628],
       [0.11841803],
       [0.12898653],
       [0.11774783],
       [0.13909105],
       [0.13934881],
       [0.12847099],
       [0.15754725],
       [0.13527608],
[0.12769769],
       [0.1525981 ],
       [0.13651336],
       [0.12939896],
       [0.14764895],
[0.13445122],
       [0.1461539 ],
       [0.11718074],
       [0.15589754],
       [0.14733963],
       [0.15223723],
       [0.12785235],
       [0.12321252],
       [0.14888624],
       [0.12403738],
       [0.14007057],
       [0.15316519],
       [0.13775065],
       [0.16265107],
       [0.11836648],
       [0.14367932],
       [0.12867721],
       [0.16229019],
       [0.15507268],
       [0.11712919],
       [0.13455432],
       [0.14228737],
       [0.14476195],
       [0.14352466],
       [0.16074358],
       [0.13125489],
       [0.15074217],
       [0.14115319],
[0.14986576],
       [0.1215628 ],
       [0.16703313],
       [0.13017226],
       [0.17590035],
       [0.14517438],
       [0.12450136],
```

[0.15424782],
[0.11877891],
[0.15177324]])

In [34]:
```
from sklearn.metrics import mean_squared_error,r2_score
```

In [35]:
```
mse_train = mean_squared_error(y_train,y_predicte)
mse_train
```

Out[35]: 0.0014173329130899262

In [36]:
```
score_train = r2_score(y_train,y_predicte) score_train
```

Out[36]:

0.14031578165778436

In [37]:
```
y_predicted = lr.predict(x_test) y_predicted
```

Out[37]:

```
array([[0.14950488],
       [0.1426998 ],
       [0.13012071],
       [0.14373087],
       [0.17791095],
       [0.12888342],
       [0.16435234],
       [0.13439966],
       [0.16332126],
       [0.14574147],
       [0.15301053],
       [0.15713482],
       [0.13960658],
       [0.12965673],
       [0.17156985],
       [0.13151266],
       [0.14440107],
       [0.12908964],
       [0.12661506],
       [0.13352325],
       [0.12352184],
       [0.16579584],
       [0.11785094],
       [0.12048018],
       [0.16125912],
       [0.12372806],
       [0.14161718],
       [0.13177043],
       [0.1257902 ],
       [0.12584176],
       [0.15254655],
       [0.1646101 ],
       [0.12594486],
       [0.13460588],
       [0.13383257],
       [0.12599642],
       [0.15058751],
       [0.12151125],
       [0.14388554],
       [0.12414049],
       [0.13898794],
       [0.17837493],
       [0.1243467 ],
       [0.18291165],
       [0.14847381],
       [0.13130644],
       [0.13084246],
       [0.14527748],
       [0.15094839],
       [0.16187776],
       [0.15141237],
       [0.13140955],
       [0.14697876],
       [0.14285446],
       [0.12491379],
       [0.16538341],
       [0.1256871 ],
       [0.11929444],
```

[0.1532683],
[0.17069344],
[0.1378022],
[0.13610093],
[0.12609953],
[0.15197946],
[0.12285165],
[0.14079232],
[0.12723371],
[0.13831774],
[0.14636011],
[0.13795687],
[0.14166873],
[0.15342296],
[0.13738978],
[0.14754585],
[0.14393709],
[0.15264966],
[0.1567224],
[0.15053596],
[0.13522452],
[0.14419486],
[0.13362636],
[0.12914119],
[0.14558681],
[0.14218427],
[0.14785517],
[0.1321313],
[0.12393427],
[0.13274995],
[0.14105009],
[0.11743851],
[0.12475913],
[0.15837211],
[0.1355854],
[0.15857833],
[0.12522311],
[0.164507],
[0.13238907],
[0.12130504],
[0.14187494],
[0.17399287],
[0.12362495],
[0.15455714],
[0.16022804],
[0.17945756],
[0.15089683],
[0.15017508],
[0.12826478],
[0.15136082],
[0.14063766],
[0.14929867],
[0.16636293],
[0.11754162],
[0.13259529],
[0.14027678],
[0.14723652],
[0.17363199],
[0.11749007],
[0.1376991],
[0.15888765],
[0.13692579],
[0.1278008],
[0.13305927],
[0.1638368],
[0.17198228],
[0.16718779],
[0.13878172],
[0.12836789],
[0.15919697],
[0.1441433],
[0.12486224],
[0.12063484],
[0.14795827],
[0.15584598],
[0.12104727],

[0.14641166],
[0.12805856],
[0.12893498],
[0.12553244],
[0.17569414],
[0.11939755],
[0.12125348],
[0.12207834],
[0.13599783],
[0.14022523],
[0.15228878],
[0.13805997],
[0.16368214],
[0.14383398],
[0.13486365],
[0.11846959],
[0. 13156421], [0.16522875],
[0.1849738],
[0.12728526],
[0.16280573],
[0.12749147],
[0.1758488],
[0.1214597],
[0.15646463],
[0.12429515],
[0.13723511],
[0.11913978],
[0.1299145],
[0.12635729],
[0.1186758],
[0.15404161],
[0.1285741],
[0.13718356],
[0.16208398],
[0.1497111],
[0.17775629],
[0.13517297],
[0.12929585],
[0.13671958],
[0.14295757],
[0.13506986],
[0.15703172],
[0.14001901],
[0.167394],
[0.15651618],
[0.12171747],
[0.13852396],
[0.11975843],
[0.1158919],
[0.1328015],
[0.16002183],
[0.11728385],
[0.13620404],
[0.12280009],
[0.12053173],
[0.12001619],
[0.18311787],
[0.13233752],
[0.12671817],
[0.12981139],
[0.13342014],
[0.15450559],
[0.16847663],
[0.14703031],
[0.12506845],
[0.13053314],
[0.13331704],
[0.131358],
[0.12331563],
[0.12249077],
[0.14424641],
[0.13537918],
[0.14682409],
[0.16414612],
[0.12548088],
[0.13434811],

[0.13857551],
[0.12274854],
[0.14991731],
[0.17141519],
[0.1695077],
[0.12702749],
[0.1229032],
[0.12743992],
[0.1384724],
[0.15971251],
[0.12326407],
[0.14538059],
[0.16744556],
[0.14512282],
[0.17615812],
[0.16631137],
[0.1250169],
[0.15533045],
[0.1722916],
[0.12496535],
[0.15466025],
[0.12764614], [0.18600487],
[0.14084387],
[0.13197664],
[0.13290461],
[0.12042862],
[0.12254232],
[0.17785939],
[0.11681987],
[0.12620263],
[0.15991872],
[0.17981843],
[0.16301194],
[0.14610234],
[0.11702608],
[0.14878313],
[0.14579302],
[0.14955644],
[0.13548229],
[0.1454837],
[0.12537777],
[0.12945051],
[0.15012353],
[0.14156562],
[0.12738837],
[0.13115178],
[0.15610375],
[0.11898512],
[0.13574006],
[0.14656633],
[0.14409175],
[0.1483707],
[0.12573865],
[0.13326548],
[0.12754303],
[0.11810871],
[0.11614967],
[0.12377961],
[0.14110164],
[0.152495],
[0.11630433],
[0.18064329],
[0.15388694],
[0.122233],
[0.18095261],
[0.17285869],
[0.12455292],
[0.12212989],
[0.13099712],
[0.1173354],
[0.14960799],
[0.16961081],
[0.13661647],
[0.13403879],
[0.11816026],
[0.12883187],

[0.11671676],
[0.12852255],
[0.14429797],
[0.14775206],
[0.13733822],
[0.16584739],
[0.13249218],
[0.15311364],
[0.18188058],
[0.12640885],
[0.15208257],
[0.15765036],
[0.11795405],
[0.13754444],
[0.17878736],
[0.12517156]])

In [38]:

```
mse_test = mean_squared_error(y_train,y_predicte)
mse_test
```

Out[38]: 0.0014173329130899262

In [39]:

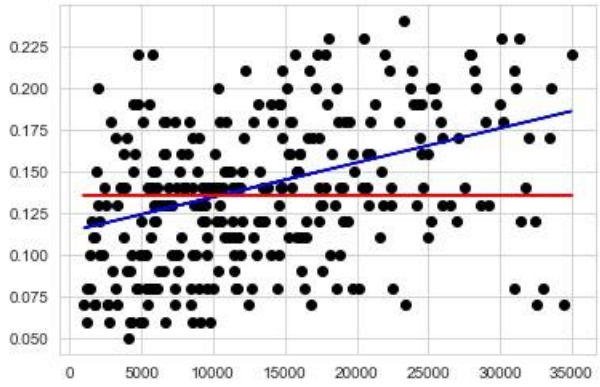
```
score_test = r2_score(y_train,y_predicte) score_test
```

Out[39]:

0.14031578165778436

In [40]:

```
plt.scatter(x_test, y_test, color='black')
plt.plot(x_test ,y_predicted , color='blue') a
= y_test.to_numpy() k = np.mean(a)
plt.plot(x_test, [k]*len(a), color='red') plt.show()
```



In [41]:

```
import statsmodels.api as sm x2 =
sm.add_constant(x)
est=sm.OLS(y,x2)
est2=est.fit() est2.summary()
```

Out[41]: OLS Regression Results

Dep. Variable:	Interest.Rate	**R-squared:**	0.145
Model:	OLS	**Adj. R-squared:**	0.143
Method:	Least Squares	**F-statistic:**	64.28
Date:	Sun, 04 Sep 2022	**Prob (F-statistic):**	1.36e-14
Time:	18:45:08	**Log-Likelihood:**	684.26
No. Observations:	380	**AIC:**	-1365.
Df Residuals:	378	**BIC:**	-1357.
Df Model:	1		
Covariance Type:	nonrobust		

	coef	std err	t	P>|t|	[0.025	0.975]
const	0.1099	0.004	28.234	0.000	0.102	0.118
Amount.Requested	2.003e-06	2.5e-07	8.017	0.000	1.51e-06	2.49e-06

Omnibus:	4.723	**Durbin-Watson:**	1.992
Prob(Omnibus):	0.094	**Jarque-Bera (JB):**	3.337
Skew:	0.066	**Prob(JB):**	0.189
Kurtosis:	2.560	**Cond. No.**	2.95e+04

Notes:

[1] Standard Errors assume that the covariance matrix of the errors is correctly specified.

[2] The condition number is large, 2.95e+04. This might indicate that there are strong multicollinearity or other numerical problems.

In [ ]:

In [ ]: