**LOGISTIC REGRESSION: Bank\_Personal\_Loan\_modelling**

#import pandas

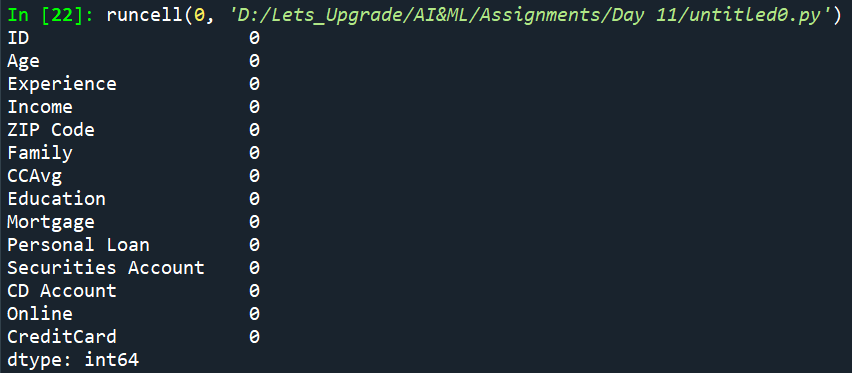
import pandas as pd

#Load dataset

dataset=pd.read\_excel("Bank\_Personal\_Loan\_Modelling.xlsx",sheet\_name=1)

#checking for null values

print(dataset.isna().sum())



#removing unimportant columns

dataset=dataset.drop('ID',axis=1)

dataset=dataset.drop('ZIP Code',axis=1)

#declaring dependent variable

y=dataset['Personal Loan']

#declaring independent variable

x=dataset[['Age','Experience','Income','Family','CCAvg','Education','Mortgage','Securities Account','CD Account','Online','CreditCard']]

#import statsmodels package to apply Logistic Regression

import statsmodels.api as sm

#adding constant to the independent variables

x1=sm.add\_constant(x)

#perform Logistic regression

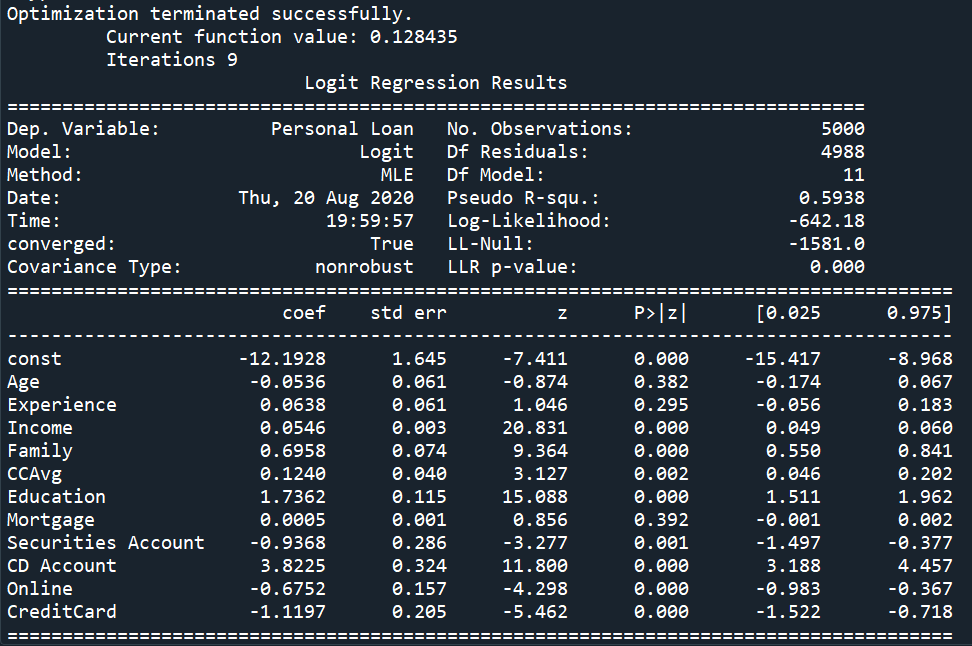
Logistic=sm.Logit(y,x1)

#developing the model

result=Logistic.fit()

#viewing the result

print(result.summary())



['Age','Experience','Income','Family','CCAvg','Education','Mortgage','Securities Account','CD Account','Online','CreditCard'] => All the independent variables have P<0.05. Therefore all the variables are significantly important.

**LOGISTIC REGRESSION: Attrition Analysis**

#import packages

import pandas as pd

import numpy as np

from sklearn import preprocessing

#load dataset

dataset=pd.read\_csv("general\_data.csv")

#drop unimportant columns

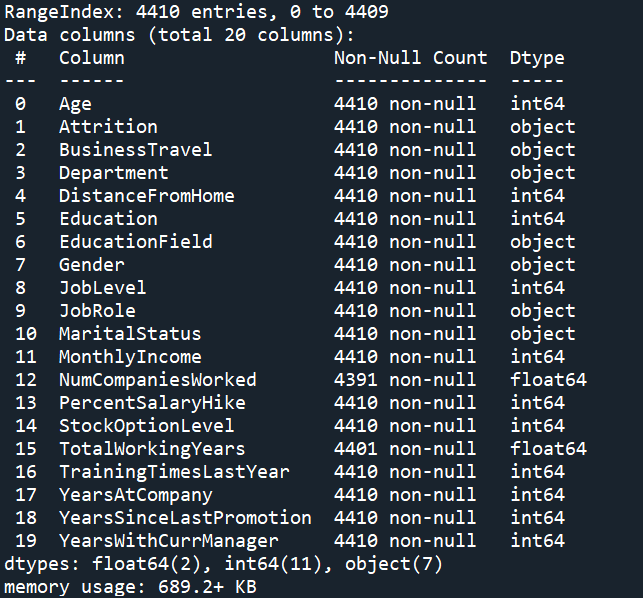
dataset=dataset.drop('EmployeeCount',axis=1)

dataset=dataset.drop('EmployeeID',axis=1)

dataset=dataset.drop('Over18',axis=1)

dataset=dataset.drop('StandardHours',axis=1)

dataset.info()



#converting string variables into Numerical

le=preprocessing.LabelEncoder()

le.fit(dataset["Gender"])

dataset["Gender"]=le.transform(dataset["Gender"])

le.fit(dataset["Attrition"])

dataset["Attrition"]=le.transform(dataset["Attrition"])

le.fit(dataset["BusinessTravel"])

dataset["BusinessTravel"]=le.transform(dataset["BusinessTravel"])

le.fit(dataset["Department"])

dataset["Department"]=le.transform(dataset["Department"])

le.fit(dataset["EducationField"])

dataset["EducationField"]=le.transform(dataset["EducationField"])

le.fit(dataset["JobRole"])

dataset["JobRole"]=le.transform(dataset["JobRole"])

le.fit(dataset["MaritalStatus"])

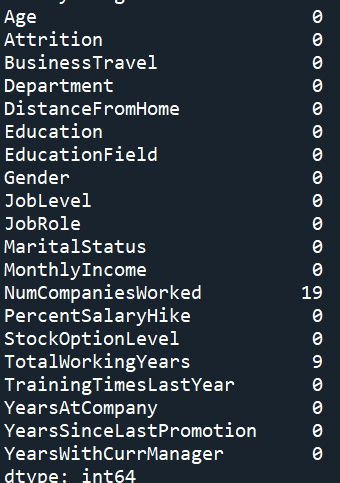
dataset["MaritalStatus"]=le.transform(dataset["MaritalStatus"])

le.fit(dataset["MaritalStatus"])

dataset["MaritalStatus"]=le.transform(dataset["MaritalStatus"])

#checking for null values

print(dataset.isna().sum())



#replacing null values with average of variable

print("mean of NumCompaniesWorked: ",dataset["NumCompaniesWorked"].mean())

new\_NCW\_var=np.where(dataset["NumCompaniesWorked"].isnull(),2,dataset["NumCompaniesWorked"])

dataset["NumCompaniesWorked"]=new\_NCW\_var

print("mean of TotalWorkingYears: ",dataset["TotalWorkingYears"].mean())

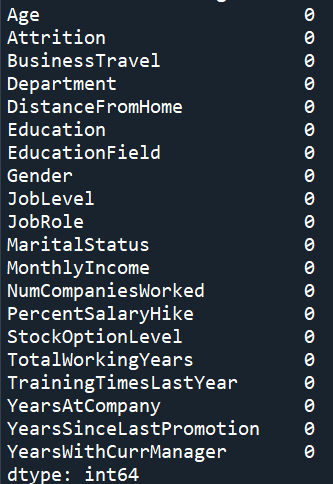
new\_TWY\_var=np.where(dataset["TotalWorkingYears"].isnull(),11,dataset["TotalWorkingYears"])

dataset["TotalWorkingYears"]=new\_TWY\_var



#recheck for null values

print(dataset.isna().sum())



print(dataset.columns)

#declaring dependent variable

Y=dataset['Attrition']

#declaring independent variable

X=dataset[['Age','BusinessTravel','Department','DistanceFromHome','Education','EducationField', 'Gender', 'JobLevel', 'JobRole','MaritalStatus','MonthlyIncome','NumCompaniesWorked','PercentSalaryHike', 'StockOptionLevel', 'TotalWorkingYears','TrainingTimesLastYear', 'YearsAtCompany', 'YearsSinceLastPromotion','YearsWithCurrManager']]

#import statsmodels package to apply Logistic Regression

import statsmodels.api as sm

#adding constant to the independent variables

X1=sm.add\_constant(X)

#perform Logistic regression

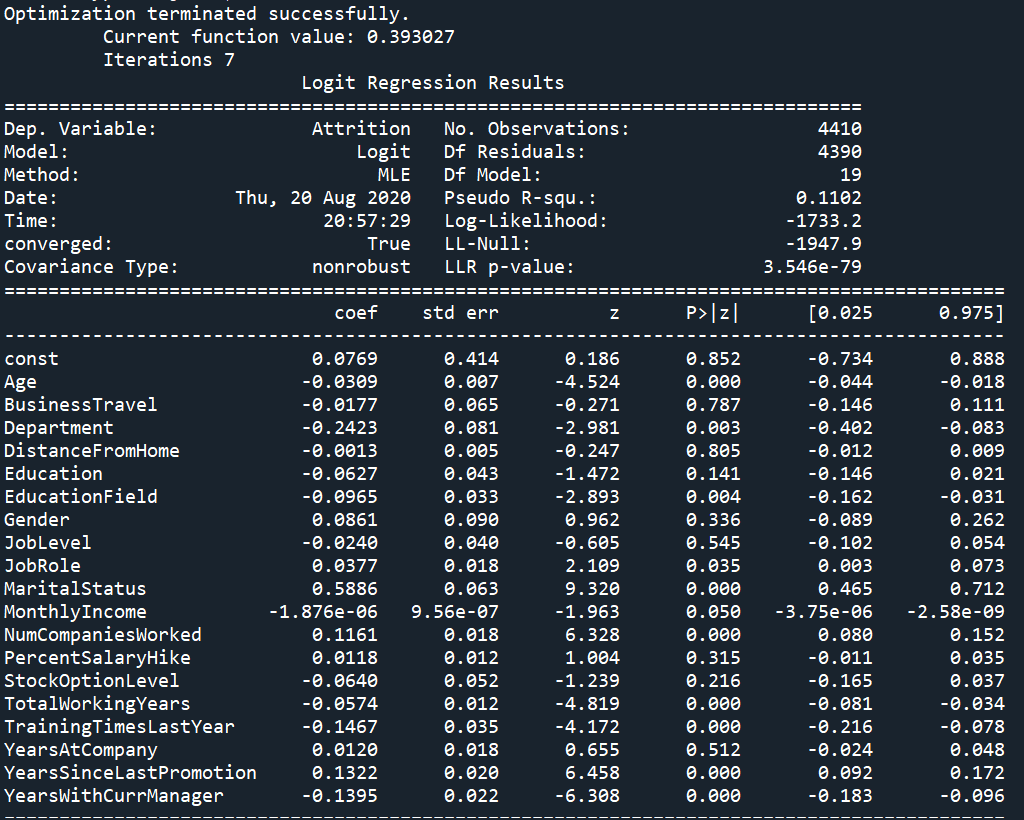
Logistic=sm.Logit(Y,X1)

#developing the model

result=Logistic.fit()

#viewing the result

print(result.summary())



Age, Department, EducationField, JobeRole, MaritialStatus, Monthly Income, NumCompaniesWorked, TotalWorkingYears, TrainingTimesLastYear, YearsSinceLastPromotion, YearsWithCurrManager => P<0.05. Therefore, the variables are significantly important.

BusinessTravel, DistanceFromHome, Education, Gender, JobLevel, PercentSalaryHike, StockOptionLevel, YearsAtCompany => P>0.05. Therefore, the variables are not significantly important.

**LINEAR REGRESSION: price v\s sqft\_living**

#Import numpy as np, pandas as pd,matplotlib.pyplot as plt and seaborn as sns.

import numpy as np

import pandas as pd

import matplotlib.pyplot as plt

import seaborn as sns

#Import the data using pandas

data=pd.read\_excel("Linear Regression.xlsx",sheet\_name=0)

#dropping unimportant columns

data=data.drop('bedrooms',axis=1)

data=data.drop('bathrooms',axis=1)

data=data.drop('floors',axis=1)

print(data.head())

print()

print(data.dtypes)

print()

#EXploratory Data Analysis-EDA

data.hist()

print(data.corr())

sns.scatterplot(data['sqft\_living'],data['price'])

#assign features to X and Y

X=data.iloc[:,1:]

Y=data.iloc[:,:1]

#visualize dataset

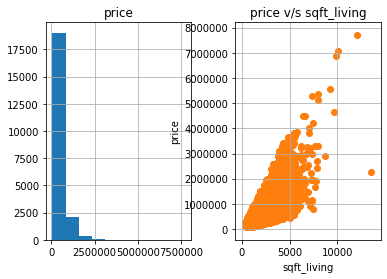
plt.scatter(X,Y)

plt.title('price v/s sqft\_living')

plt.xlabel('sqft\_living')

plt.ylabel('price')

plt.show()



#split records for training and testing

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=0)

print(X\_train.shape)

print(X\_test.shape)

#model building with sklearn

from sklearn.linear\_model import LinearRegression

lin\_reg=LinearRegression()

#train the model

lin\_reg.fit(X\_train,Y\_train)

print(lin\_reg.coef\_)

print(lin\_reg.intercept\_)

#visualize training set result

plt.scatter(X\_train,Y\_train,color='red')

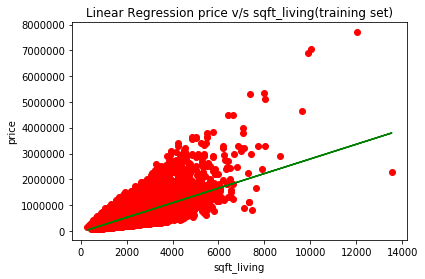
plt.plot(X\_train,lin\_reg.predict(X\_train),color='green')

plt.title('Linear Regression price v/s sqft\_living(training set)')

plt.xlabel('sqft\_living')

plt.ylabel('price')

plt.show()



#test the model(for X\_test value predict y value)

y\_pred=lin\_reg.predict(X\_test)

print(y\_pred)

print(X\_test.head())

#compare initial data to predicted data

print(data.head())

#vizualise the test dataset

plt.scatter(X\_test,Y\_test,color='green')

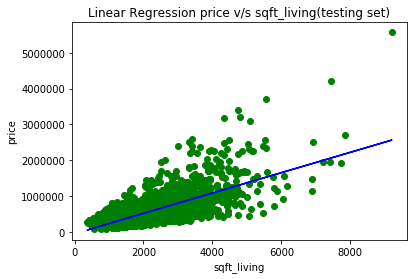
plt.plot(X\_test,lin\_reg.predict(X\_test),color='blue')

plt.title('Linear Regression price v/s sqft\_living(testing set)')

plt.xlabel('sqft\_living')

plt.ylabel('price')

plt.show()



#Estimate the cost

from sklearn.metrics import mean\_squared\_error,r2\_score

RSME=np.sqrt(mean\_squared\_error(Y\_test, y\_pred))

r\_square=r2\_score(Y\_test, y\_pred)

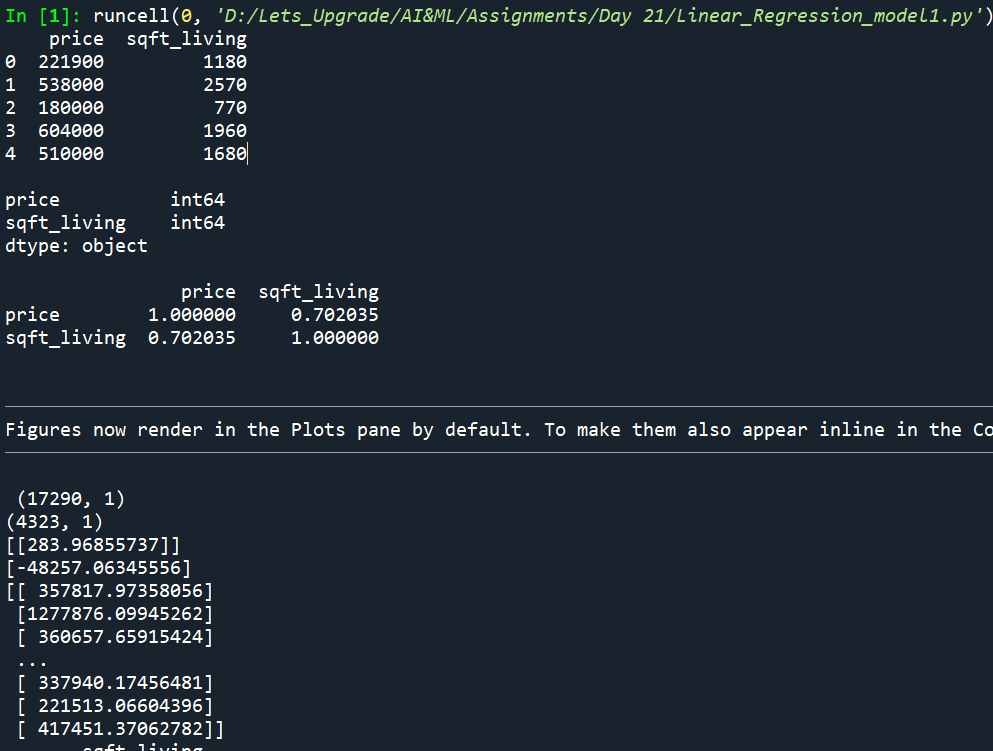
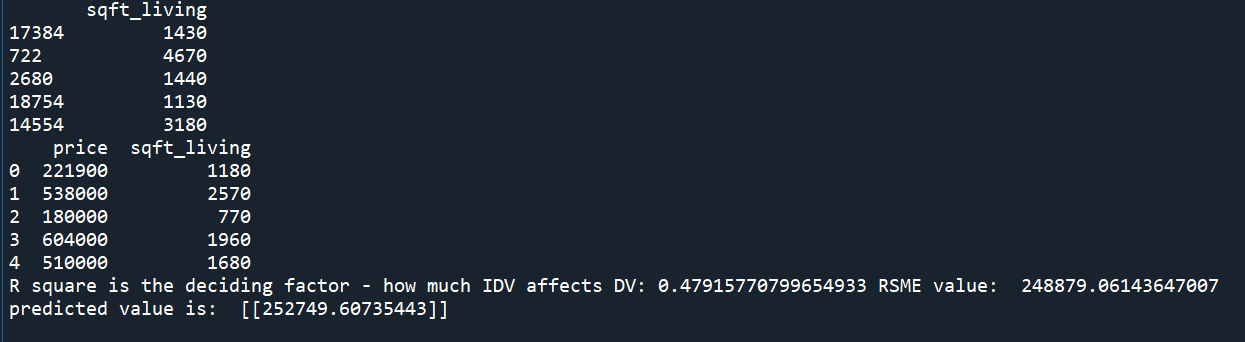
print("R square is the deciding factor - how much IDV affects DV:",r\_square,"RSME value: ",RSME)

#predicting unseen value

unseen\_predict=lin\_reg.predict(np.array([[1060]]))

print("predicted value is: ",unseen\_predict)

**Output:**

**LINEAR REGRESSION: price v\s bedrooms**

#Import numpy as np, pandas as pd,matplotlib.pyplot as plt and seaborn as sns.

import numpy as np

import pandas as pd

import matplotlib.pyplot as plt

import seaborn as sns

#Import the data using pandas

data=pd.read\_excel("Linear Regression.xlsx",sheet\_name=0)

#dropping unimportant columns

data=data.drop('sqft\_living',axis=1)

data=data.drop('bathrooms',axis=1)

data=data.drop('floors',axis=1)

print(data.head())

print()

print(data.dtypes)

print()

#EXploratory Data Analysis-EDA

data.hist()

print(data.corr())

sns.scatterplot(data['bedrooms'],data['price'])

#assign features to X and Y

X=data.iloc[:,1:]

Y=data.iloc[:,:1]

#visualize dataset

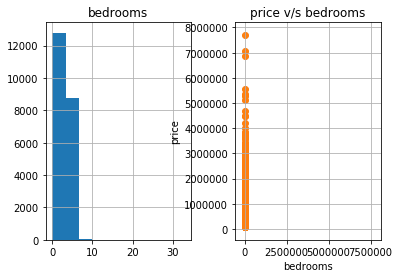
plt.scatter(X,Y)

plt.title('price v/s bedrooms')

plt.xlabel('bedrooms')

plt.ylabel('price')

plt.show()



#split records for training and testing

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=0)

print(X\_train.shape)

print(X\_test.shape)

#model building with sklearn

from sklearn.linear\_model import LinearRegression

lin\_reg=LinearRegression()

#train the model

lin\_reg.fit(X\_train,Y\_train)

print(lin\_reg.coef\_)

print(lin\_reg.intercept\_)

#visualize training set result

plt.scatter(X\_train,Y\_train,color='red')

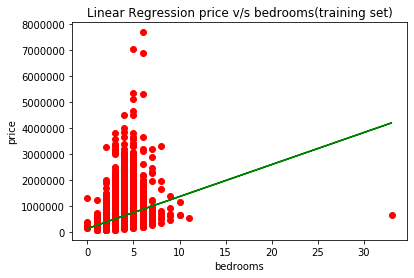
plt.plot(X\_train,lin\_reg.predict(X\_train),color='green')

plt.title('Linear Regression price v/s bedrooms(training set)')

plt.xlabel('bedrooms')

plt.ylabel('price')

plt.show()



#test the model(for X\_test value predict y value)

y\_pred=lin\_reg.predict(X\_test)

print(y\_pred)

print(X\_test.head())

#compare initial data to predicted data

print(data.head())

#vizualise the test dataset

plt.scatter(X\_test,Y\_test,color='green')

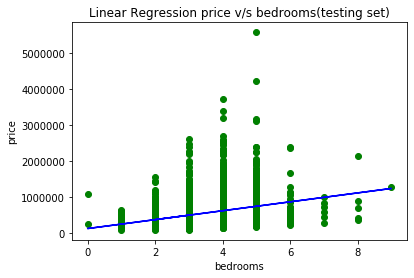
plt.plot(X\_test,lin\_reg.predict(X\_test),color='blue')

plt.title('Linear Regression price v/s bedrooms(testing set)')

plt.xlabel('bedrooms')

plt.ylabel('price')

plt.show()



#Estimate the cost

from sklearn.metrics import mean\_squared\_error,r2\_score

RSME=np.sqrt(mean\_squared\_error(Y\_test, y\_pred))

r\_square=r2\_score(Y\_test, y\_pred)

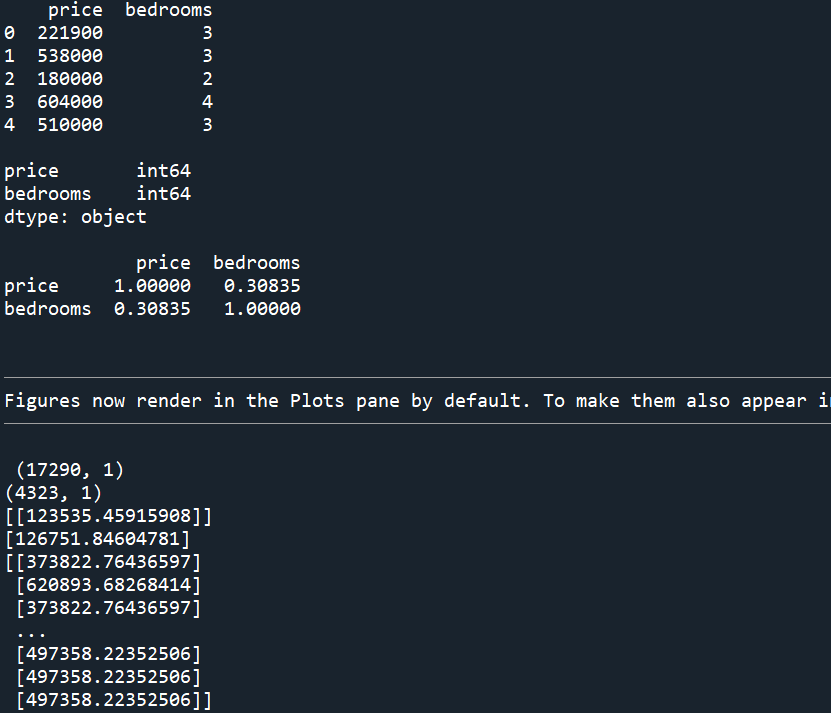
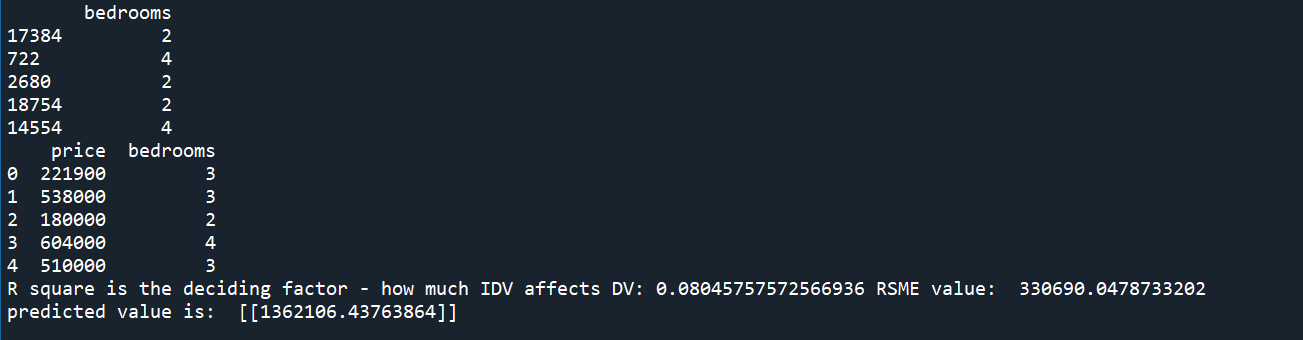
print("R square is the deciding factor - how much IDV affects DV:",r\_square,"RSME value: ",RSME)

#predicting unseen value

unseen\_predict=lin\_reg.predict(np.array([[10]]))

print("predicted value is: ",unseen\_predict)

**Output:**

**LINEAR REGRESSION: price v\s bathrooms**

#Import numpy as np, pandas as pd,matplotlib.pyplot as plt and seaborn as sns.

import numpy as np

import pandas as pd

import matplotlib.pyplot as plt

import seaborn as sns

#Import the data using pandas

data=pd.read\_excel("Linear Regression.xlsx",sheet\_name=0)

#dropping unimportant columns

data=data.drop('sqft\_living',axis=1)

data=data.drop('bedrooms',axis=1)

data=data.drop('floors',axis=1)

print(data.head())

print()

print(data.dtypes)

print()

#EXploratory Data Analysis-EDA

data.hist()

print(data.corr())

sns.scatterplot(data['bathrooms'],data['price'])

#assign features to X and Y

X=data.iloc[:,1:]

Y=data.iloc[:,:1]

#visualize dataset

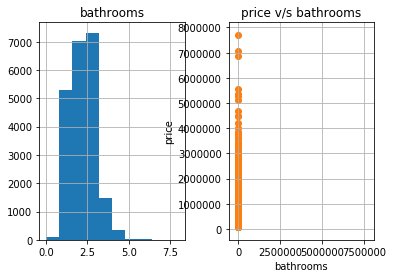
plt.scatter(X,Y)

plt.title('price v/s bathrooms')

plt.xlabel('bathrooms')

plt.ylabel('price')

plt.show()



#split records for training and testing

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=0)

print(X\_train.shape)

print(X\_test.shape)

#model building with sklearn

from sklearn.linear\_model import LinearRegression

lin\_reg=LinearRegression()

#train the model

lin\_reg.fit(X\_train,Y\_train)

print(lin\_reg.coef\_)

print(lin\_reg.intercept\_)

#visualize training set result

plt.scatter(X\_train,Y\_train,color='red')

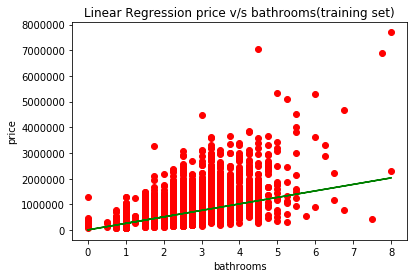
plt.plot(X\_train,lin\_reg.predict(X\_train),color='green')

plt.title('Linear Regression price v/s bathrooms(training set)')

plt.xlabel('bathrooms')

plt.ylabel('price')

plt.show()



#test the model(for X\_test value predict y value)

y\_pred=lin\_reg.predict(X\_test)

print(y\_pred)

print(X\_test.head())

#compare initial data to predicted data

print(data.head())

#vizualise the test dataset

plt.scatter(X\_test,Y\_test,color='green')

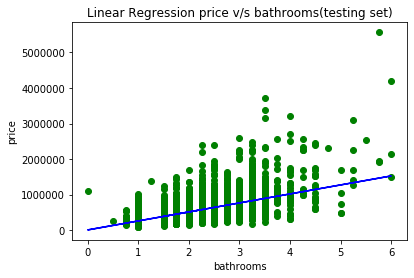
plt.plot(X\_test,lin\_reg.predict(X\_test),color='blue')

plt.title('Linear Regression price v/s bathrooms(testing set)')

plt.xlabel('bathrooms')

plt.ylabel('price')

plt.show()



#Estimate the cost

from sklearn.metrics import mean\_squared\_error,r2\_score

RSME=np.sqrt(mean\_squared\_error(Y\_test, y\_pred))

r\_square=r2\_score(Y\_test, y\_pred)

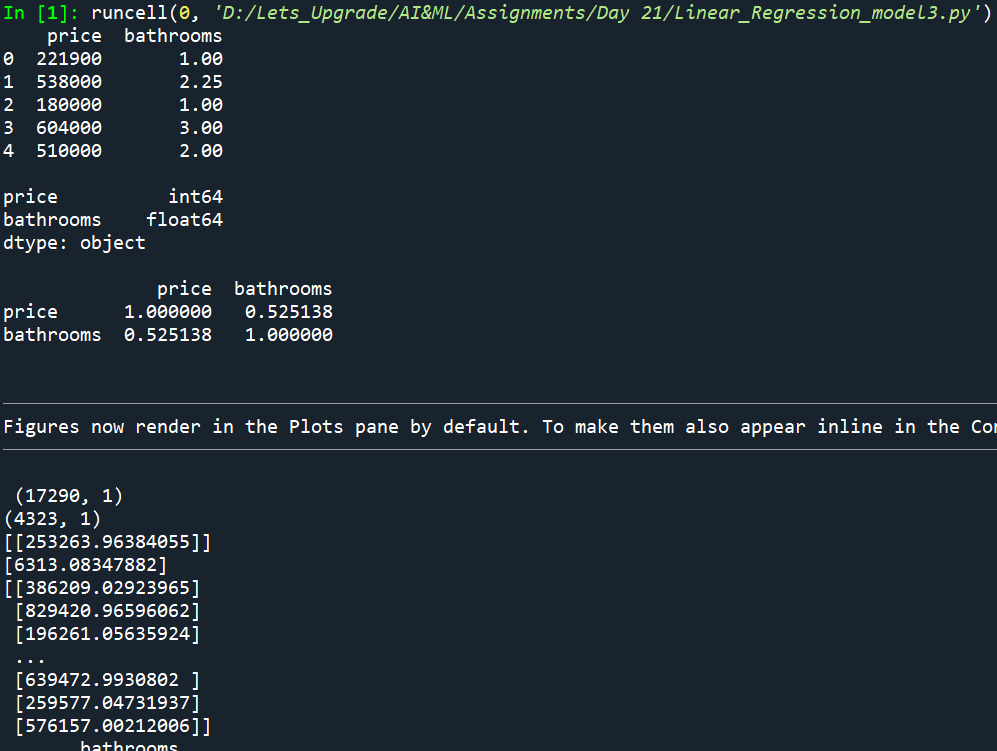
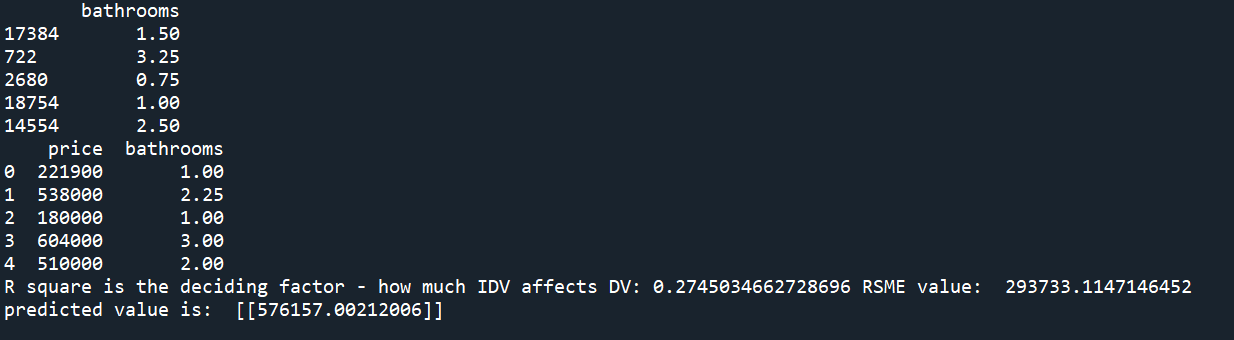
print("R square is the deciding factor - how much IDV affects DV:",r\_square,"RSME value: ",RSME)

#predicting unseen value

unseen\_predict=lin\_reg.predict(np.array([[2.25]]))

print("predicted value is: ",unseen\_predict)

**Output:**

**LINEAR REGRESSION: price v\s floors**

#Import numpy as np, pandas as pd,matplotlib.pyplot as plt and seaborn as sns.

import numpy as np

import pandas as pd

import matplotlib.pyplot as plt

import seaborn as sns

#Import the data using pandas

data=pd.read\_excel("Linear Regression.xlsx",sheet\_name=0)

#dropping unimportant columns

data=data.drop('sqft\_living',axis=1)

data=data.drop('bedrooms',axis=1)

data=data.drop('bathrooms',axis=1)

print(data.head())

print()

print(data.dtypes)

print()

#EXploratory Data Analysis-EDA

data.hist()

print(data.corr())

sns.scatterplot(data['floors'],data['price'])

#assign features to X and Y

X=data.iloc[:,1:]

Y=data.iloc[:,:1]

#visualize dataset

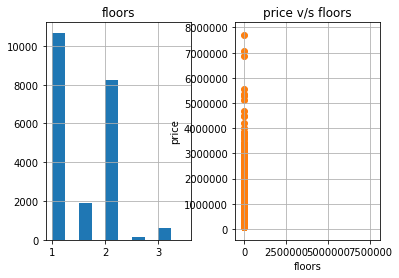
plt.scatter(X,Y)

plt.title('price v/s floors')

plt.xlabel('floors')

plt.ylabel('price')

plt.show()



#split records for training and testing

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=0)

print(X\_train.shape)

print(X\_test.shape)

#model building with sklearn

from sklearn.linear\_model import LinearRegression

lin\_reg=LinearRegression()

#train the model

lin\_reg.fit(X\_train,Y\_train)

print(lin\_reg.coef\_)

print(lin\_reg.intercept\_)

#visualize training set result

plt.scatter(X\_train,Y\_train,color='red')

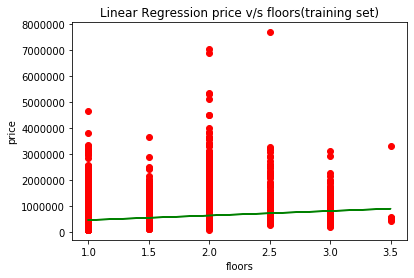
plt.plot(X\_train,lin\_reg.predict(X\_train),color='green')

plt.title('Linear Regression price v/s floors(training set)')

plt.xlabel('floors')

plt.ylabel('price')

plt.show()



#test the model(for X\_test value predict y value)

y\_pred=lin\_reg.predict(X\_test)

print(y\_pred)

print(X\_test.head())

#compare initial data to predicted data

print(data.head())

#vizualise the test dataset

plt.scatter(X\_test,Y\_test,color='green')

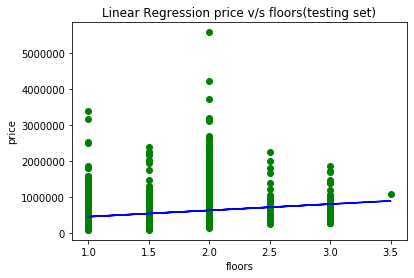
plt.plot(X\_test,lin\_reg.predict(X\_test),color='blue')

plt.title('Linear Regression price v/s floors(testing set)')

plt.xlabel('floors')

plt.ylabel('price')

plt.show()



#Estimate the cost

from sklearn.metrics import mean\_squared\_error,r2\_score

RSME=np.sqrt(mean\_squared\_error(Y\_test, y\_pred))

r\_square=r2\_score(Y\_test, y\_pred)

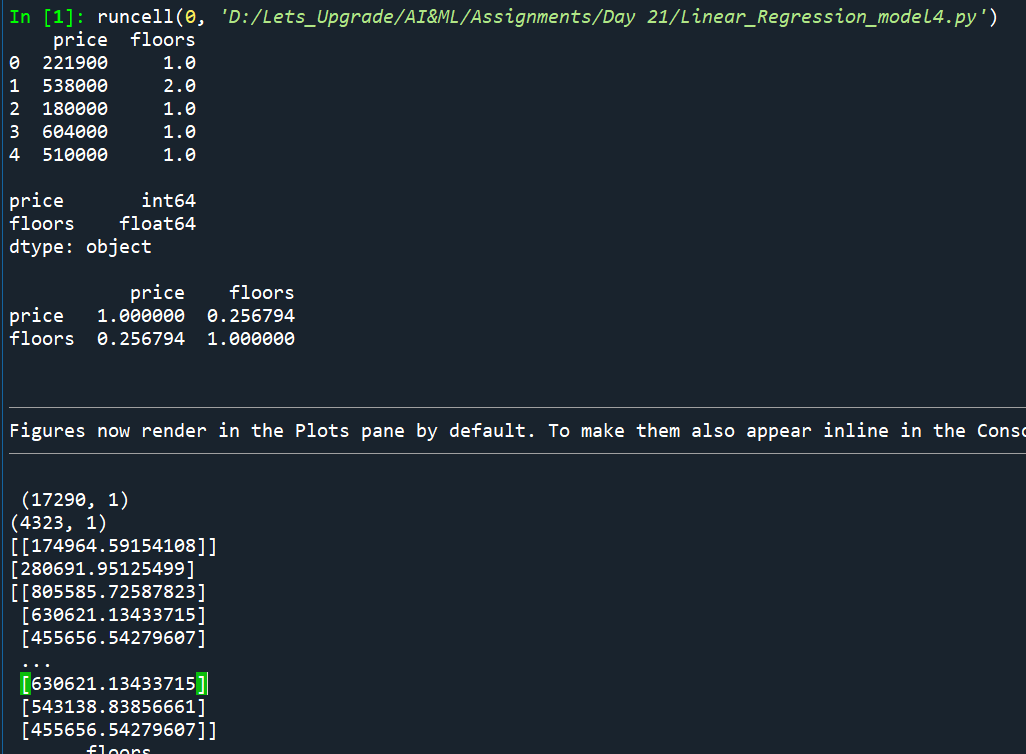
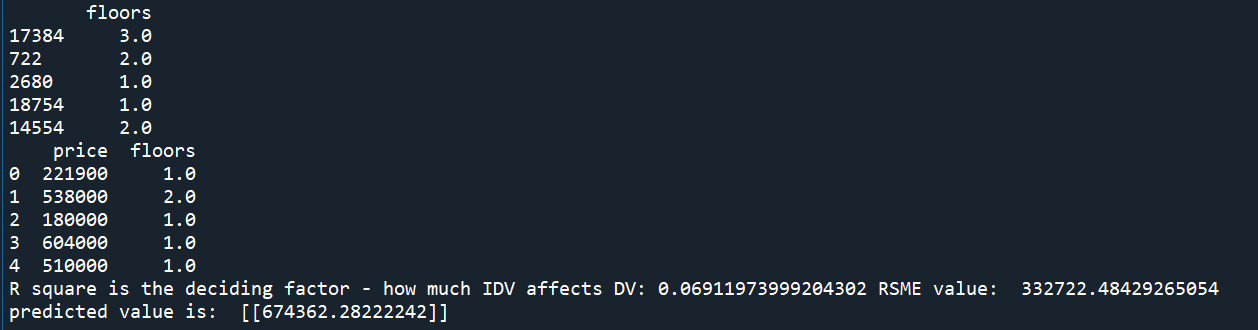
print("R square is the deciding factor - how much IDV affects DV:",r\_square,"RSME value: ",RSME)

#predicting unseen value

unseen\_predict=lin\_reg.predict(np.array([[2.25]]))

print("predicted value is: ",unseen\_predict)

**Output:**

**MULTIPLE LINEAR REGRESSION: (DV🡪price) v\s (IDV🡪sqft\_living, bedrooms, bathrooms, floors)**

#import packages for Multiple Linear Regression

import numpy as np

import pandas as pd

import matplotlib.pyplot as plt

import seaborn as sns

#load data

data=pd.read\_excel("Linear Regression.xlsx",sheet\_name=0)

#check for string value

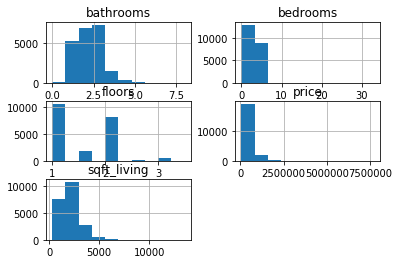
print(data.dtypes)

#check for null values in dataset

print(data.isnull().sum())

print(data.describe())

data.hist()



#data normalization(not required as no bid diff in data values)

#from sklearn.preprocessing import StandardScaler

#sc=StandardScaler()

#data=sc.fit\_transform(data)

#before splitting assign input variables(IDV) to X and output variable(DV) to Y

X=data.iloc[:,1:5]

Y=data.iloc[:,:1]

#split records for training and testing in ration 75:25

from sklearn.model\_selection import train\_test\_split

x\_train,x\_test,y\_train,y\_test=train\_test\_split(X,Y,test\_size=0.25,random\_state=2)

#training the model by calling linear regression algorithm from sklearn

from sklearn.linear\_model import LinearRegression

mul\_reg=LinearRegression()

mul\_reg.fit(x\_train,y\_train)

#testing the model

ypred=mul\_reg.predict(x\_test)

#forecasting by trained data

unseen\_pred=mul\_reg.predict(np.array([[1180,3,1,1]]))

print(unseen\_pred)

#evaluation

from sklearn.metrics import r2\_score,mean\_squared\_error

RSME=np.sqrt(mean\_squared\_error(y\_test, ypred))

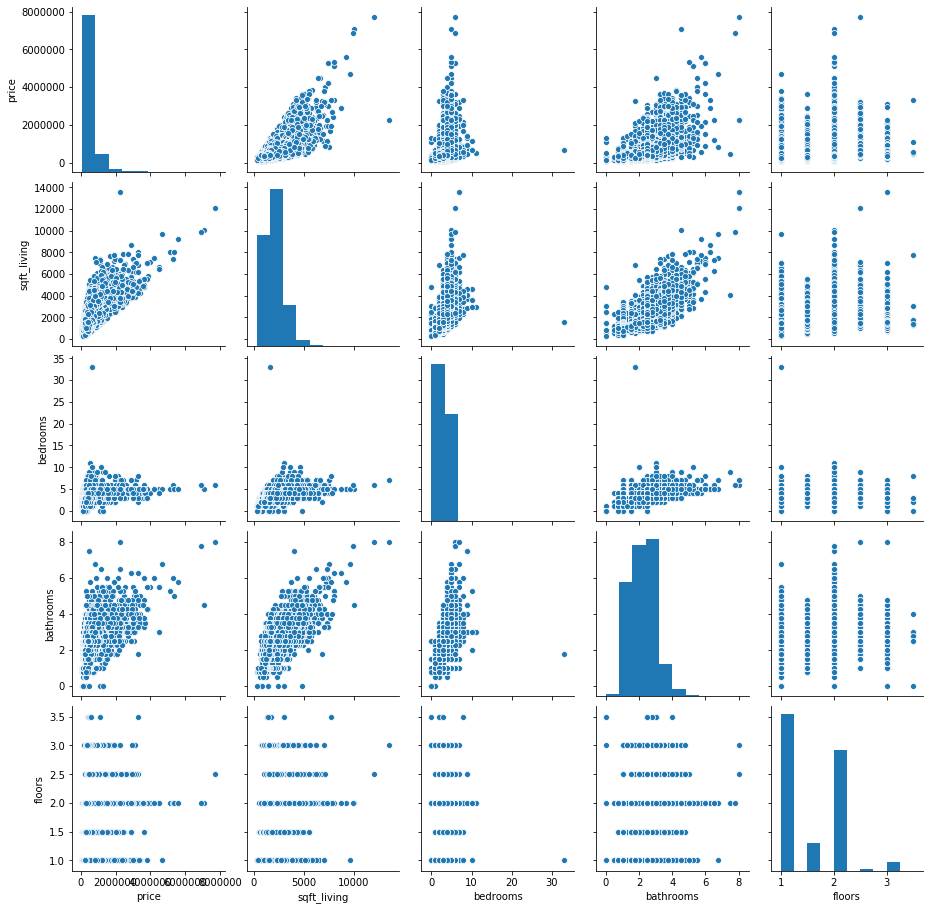
r\_square=r2\_score(y\_test, ypred)

print("R square is the deciding factor - how much IDV affects DV:",r\_square,"RSME value: ",RSME)

#inferences

print(data.corr())

sns.pairplot(data)



**Output:**

