Data to Fish

Example of Multiple Linear Regression in Python

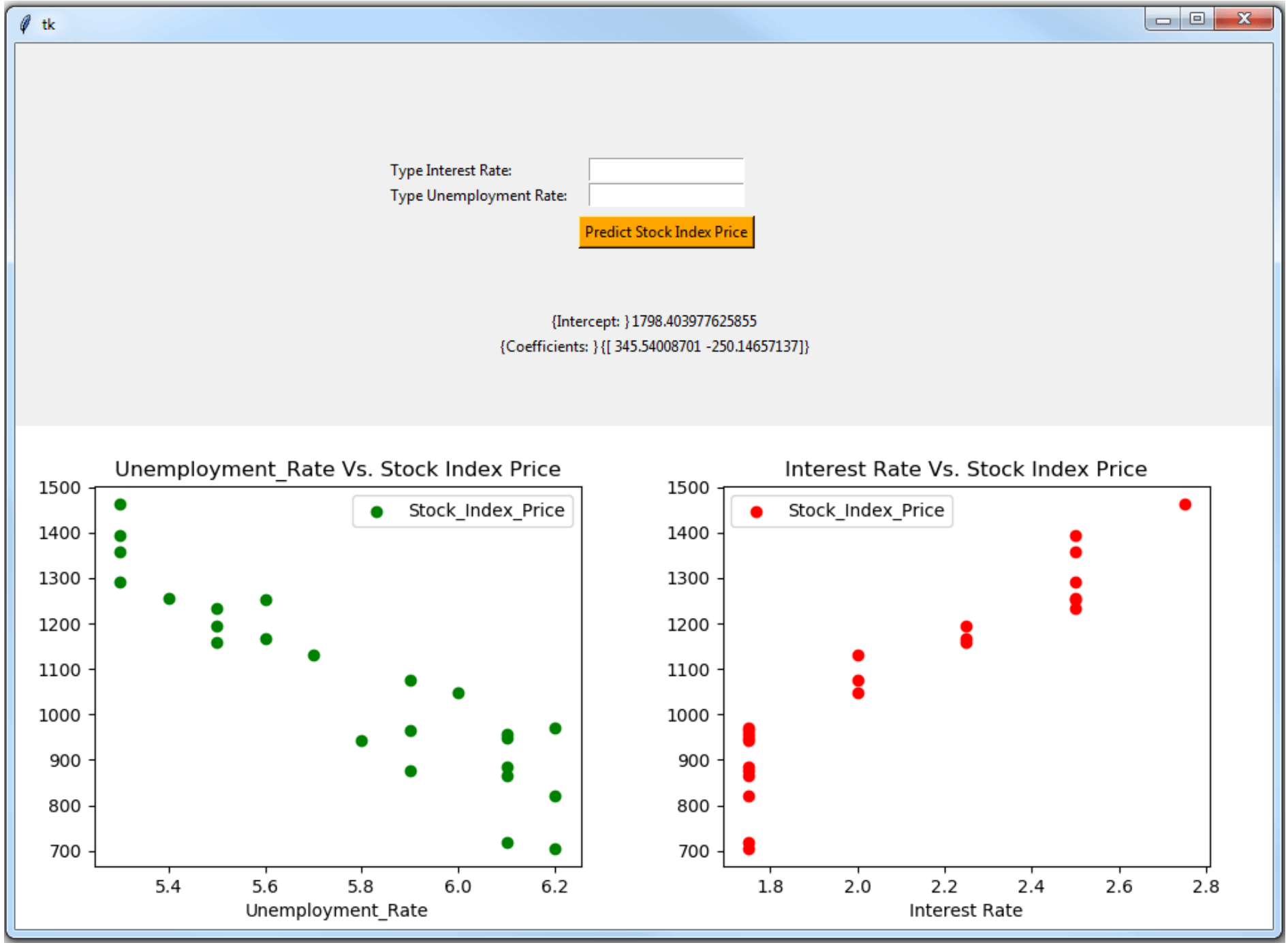
Python / August 13, 2020

In this tutorial, you'll see how to perform multiple linear regression in Python using both *sklearn* and *statsmodels*.

Here are the topics to be covered:

1. Reviewing the example to be used in this tutorial
2. Checking for Linearity
3. Performing the multiple linear regression in Python
4. Adding a *tkinter* Graphical User Interface (GUI) to gather input from users, and then display the prediction results

By the end of this tutorial, you'll be able to create the following interface in Python:


tk
Type Interest Rate:
Type Unemployment Rate:
Predict Stock Index Price
{Intercept: }1798.403977625855
{Coefficients: }{[345.54008701 -250.14657137]}
Unemployment_Rate Vs. Stock Index Price
Stock_Index_Price
1500
1400
1300
1200
1100
1000
900
800
700
5.4
5.6
5.8
6.0
6.2
Unemployment_Rate
Interest Rate Vs. Stock Index Price
Stock_Index_Price
1500
1400
1300
1200
1100
1000
900
800
700
1.8
2.0
2.2
2.4
2.6
2.8
Interest Rate

Example of Multiple Linear Regression in Python

In the following example, we will use multiple linear regression to predict the stock index price (i.e., the dependent variable) of a fictitious economy by using 2 independent/input variables:

- Interest Rate
- Unemployment Rate

Please note that you will have to validate that several assumptions are met before you apply linear regression models. Most notably, you have to make sure that a linear relationship exists between the dependent variable and the independent variable/s (more on that under the *checking for linearity* section).

Let’s now jump into the dataset that we’ll be using:

Year	Month	Interest_Rate	Unemployment_Rate	Stock_Index_Price
2017	12	2.75	5.3	1464
2017	11	2.5	5.3	1394
2017	10	2.5	5.3	1357
2017	9	2.5	5.3	1293
2017	8	2.5	5.4	1256
2017	7	2.5	5.6	1254
2017	6	2.5	5.5	1234
2017	5	2.25	5.5	1195
2017	4	2.25	5.5	1159
2017	3	2.25	5.6	1167
2017	2	2	5.7	1130
2017	1	2	5.9	1075
2016	12	2	6	1047
2016	11	1.75	5.9	965
2016	10	1.75	5.8	943
2016	9	1.75	6.1	958
2016	8	1.75	6.2	971
2016	7	1.75	6.1	949
2016	6	1.75	6.1	884
2016	5	1.75	6.1	866
2016	4	1.75	5.9	876
2016	3	1.75	6.2	822
2016	2	1.75	6.2	704
2016	1	1.75	6.1	719

To start, you may capture the above dataset in Python using Pandas DataFrame:

```
import pandas as pd

Stock_Market = {'Year': [2017,2017,2017,2017,2017,2017,2017,2017,2017,2017,2017,2017,2016,2016,201
```

```
                'Month': [12, 11,10,9,8,7,6,5,4,3,2,1,12,11,10,9,8,7,6,5,4,3,2,1],
                'Interest_Rate': [2.75,2.5,2.5,2.5,2.5,2.5,2.5,2.25,2.25,2.25,2,2,2,1.75,1.75,1.75
                'Unemployment_Rate': [5.3,5.3,5.3,5.3,5.4,5.6,5.5,5.5,5.5,5.6,5.7,5.9,6,5.9,5.8,6.
                'Stock_Index_Price': [1464,1394,1357,1293,1256,1254,1234,1195,1159,1167,1130,1075,
                }

df = pd.DataFrame(Stock_Market,columns=['Year','Month','Interest_Rate','Unemployment_Rate','Stock_

print (df)
```

Checking for Linearity

Before you execute a linear regression model, it is advisable to validate that certain assumptions are met.

As noted earlier, you may want to check that a linear relationship exists between the dependent variable and the independent variable/s.

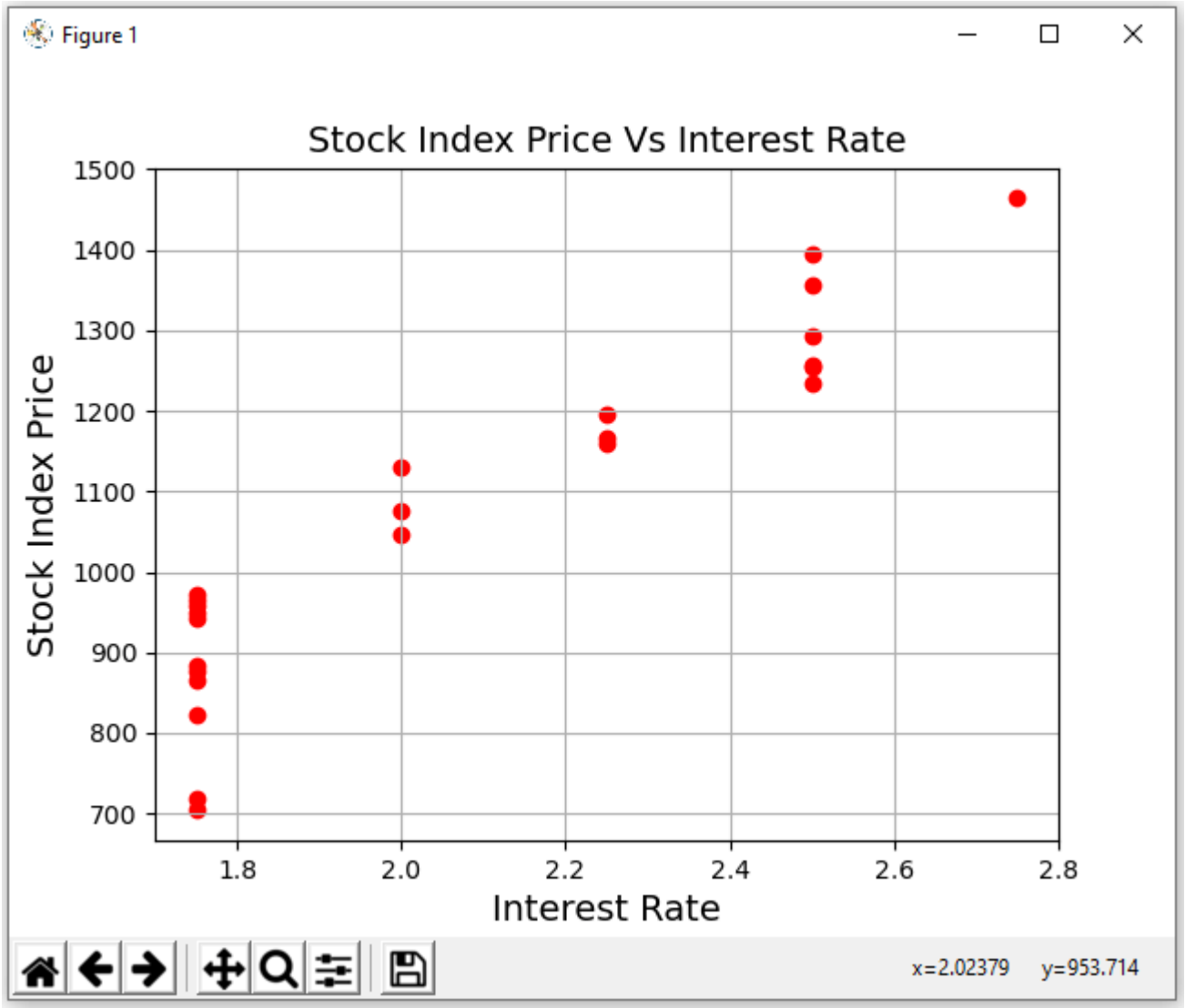
In our example, you may want to check that a linear relationship exists between the:

- Stock_Index_Price (dependent variable) and Interest_Rate (independent variable)
- Stock_Index_Price (dependent variable) and Unemployment_Rate (independent variable)

To perform a quick linearity check, you can use scatter diagrams (utilizing the *matplotlib* library). For example, you can use the code below in order to plot the relationship between the Stock_Index_Price and the Interest_Rate:

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

Stock_Market = {'Year': [2017,2017,2017,2017,2017,2017,2017,2017,2017,2017,2017,2017,2016,2016,201
                'Month': [12, 11,10,9,8,7,6,5,4,3,2,1,12,11,10,9,8,7,6,5,4,3,2,1],
                'Interest_Rate': [2.75,2.5,2.5,2.5,2.5,2.5,2.5,2.25,2.25,2.25,2,2,2,1.75,1.75,1.75
                'Unemployment_Rate': [5.3,5.3,5.3,5.3,5.4,5.6,5.5,5.5,5.5,5.6,5.7,5.9,6,5.9,5.8,6.
                'Stock_Index_Price': [1464,1394,1357,1293,1256,1254,1234,1195,1159,1167,1130,1075,
                }

df = pd.DataFrame(Stock_Market,columns=['Year','Month','Interest_Rate','Unemployment_Rate','Stock_

plt.scatter(df['Interest_Rate'], df['Stock_Index_Price'], color='red')
plt.title('Stock Index Price Vs Interest Rate', fontsize=14)
plt.xlabel('Interest Rate', fontsize=14)
plt.ylabel('Stock Index Price', fontsize=14)
plt.grid(True)
plt.show()
```

You'll notice that indeed a linear relationship exists between the Stock_Index_Price and the Interest_Rate. Specifically, when interest rates go up, the stock index price also goes up:



And for the second case, you can use this code in order to plot the relationship between the Stock_Index_Price and the Unemployment_Rate:

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

Stock_Market = {'Year': [2017,2017,2017,2017,2017,2017,2017,2017,2017,2017,2017,2017,2016,2016,201
                'Month': [12, 11,10,9,8,7,6,5,4,3,2,1,12,11,10,9,8,7,6,5,4,3,2,1],
                'Interest_Rate': [2.75,2.5,2.5,2.5,2.5,2.5,2.5,2.25,2.25,2.25,2,2,2,1.75,1.75,1.75
                'Unemployment_Rate': [5.3,5.3,5.3,5.3,5.4,5.6,5.5,5.5,5.5,5.6,5.7,5.9,6,5.9,5.8,6.
                'Stock_Index_Price': [1464,1394,1357,1293,1256,1254,1234,1195,1159,1167,1130,1075,
                }

df = pd.DataFrame(Stock_Market,columns=['Year','Month','Interest_Rate','Unemployment_Rate','Stock_

plt.scatter(df['Unemployment_Rate'], df['Stock_Index_Price'], color='green')
plt.title('Stock Index Price Vs Unemployment Rate', fontsize=14)
plt.xlabel('Unemployment Rate', fontsize=14)
plt.ylabel('Stock Index Price', fontsize=14)
plt.grid(True)
plt.show()
```

As you can see, a linear relationship also exists between the Stock_Index_Price and the Unemployment_Rate – when the unemployment rates go up, the stock index price goes down (here we still have a linear relationship, but with a negative slope):

Next, we are going to perform the actual multiple linear regression in Python.

Performing the Multiple Linear Regression

Once you added the data into Python, you may use both sklearn and statsmodels to get the regression results.

Either method would work, but let's review both methods for illustration purposes.

You may then copy the code below into Python:

```
import pandas as pd
from sklearn import linear_model
import statsmodels.api as sm

Stock_Market = {'Year': [2017,2017,2017,2017,2017,2017,2017,2017,2017,2017,2017,2017,2016,2016,201
                'Month': [12, 11,10,9,8,7,6,5,4,3,2,1,12,11,10,9,8,7,6,5,4,3,2,1],
                'Interest_Rate': [2.75,2.5,2.5,2.5,2.5,2.5,2.5,2.25,2.25,2.25,2,2,2,1.75,1.75,1.75
                'Unemployment_Rate': [5.3,5.3,5.3,5.3,5.4,5.6,5.5,5.5,5.5,5.6,5.7,5.9,6,5.9,5.8,6.
                'Stock_Index_Price': [1464,1394,1357,1293,1256,1254,1234,1195,1159,1167,1130,1075,
                }

df = pd.DataFrame(Stock_Market,columns=['Year','Month','Interest_Rate','Unemployment_Rate','Stock_

X = df[['Interest_Rate','Unemployment_Rate']] # here we have 2 variables for multiple regression.
Y = df['Stock_Index_Price']

# with sklearn
regr = linear_model.LinearRegression()
regr.fit(X, Y)
```

```
print('Intercept: \n', regr.intercept_)
print('Coefficients: \n', regr.coef_)

# prediction with sklearn
New_Interest_Rate = 2.75
New_Unemployment_Rate = 5.3
print ('Predicted Stock Index Price: \n', regr.predict([[New_Interest_Rate ,New_Unemployment_Rate]

# with statsmodels
X = sm.add_constant(X) # adding a constant

model = sm.OLS(Y, X).fit()
predictions = model.predict(X)

print_model = model.summary()
print(print_model)
```

Once you run the code in Python, you'll observe three parts:

(1) The first part shows the output generated by *sklearn:*

```
Intercept:
 1798.4039776258564
Coefficients:
 [ 345.54008701 -250.14657137]
```

This output includes the intercept and coefficients. You can use this information to build the multiple linear regression equation as follows:

$$\text{Stock\_Index\_Price} = (\text{Intercept}) + (\text{Interest\_Rate coef})*X_1 + (\text{Unemployment\_Rate coef})*X_2$$

And once you plug the numbers:

$$\text{Stock\_Index\_Price} = (1798.4040) + (345.5401)*X_1 + (-250.1466)*X_2$$

(2) The second part displays the predicted output using *sklearn:*

```
Predicted Stock Index Price:
 [1422.86238865]
```

Imagine that you want to predict the stock index price after you collected the following data:

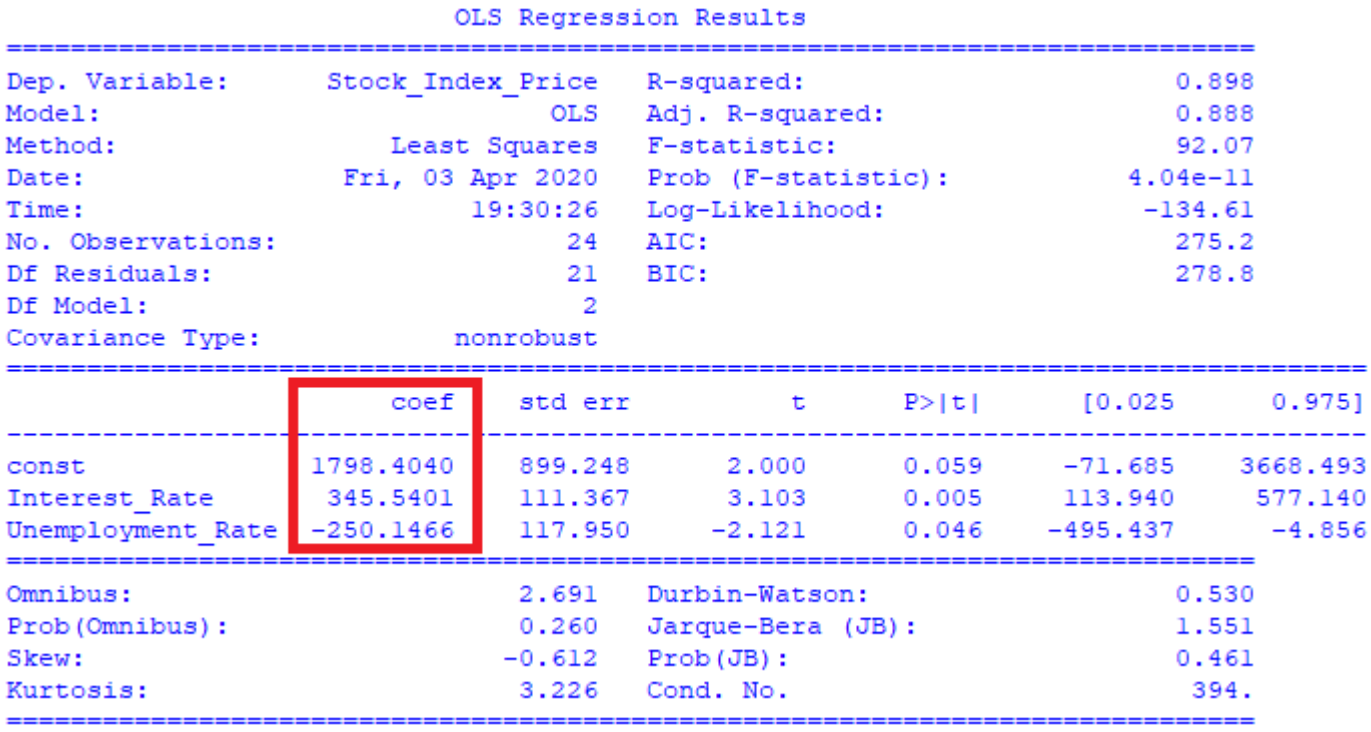
- Interest Rate = 2.75 (i.e., X_1= 2.75)
- Unemployment Rate = 5.3 (i.e., X_2= 5.3)

If you plug that data into the regression equation, you'll get the same predicted result as displayed in the second part:

$$\text{Stock\_Index\_Price} = (1798.4040) + (345.5401)*(2.75) + (-250.1466)*(5.3) = 1422.86$$

(3) The third part displays a comprehensive table with statistical info generated by *statsmodels*.

This information can provide you additional insights about the model used (such as the fit of the model, standard errors, etc):



OLS Regression Results

Dep. Variable:	Stock_Index_Price	R-squared:	0.898
Model:	OLS	Adj. R-squared:	0.888
Method:	Least Squares	F-statistic:	92.07
Date:	Fri, 03 Apr 2020	Prob (F-statistic):	4.04e-11
Time:	19:30:26	Log-Likelihood:	-134.61
No. Observations:	24	AIC:	275.2
Df Residuals:	21	BIC:	278.8
Df Model:	2		
Covariance Type:	nonrobust		

	coef	std err	t	P>|t|	[0.025	0.975]
const	1798.4040	899.248	2.000	0.059	-71.685	3668.493
Interest_Rate	345.5401	111.367	3.103	0.005	113.940	577.140
Unemployment_Rate	-250.1466	117.950	-2.121	0.046	-495.437	-4.856

Omnibus:	2.691	Durbin-Watson:	0.530
Prob(Omnibus):	0.260	Jarque-Bera (JB):	1.551
Skew:	-0.612	Prob(JB):	0.461
Kurtosis:	3.226	Cond. No.	394.

Notice that the coefficients captured in this table (highlighted in red) match with the coefficients generated by sklearn.

That's a good sign! we got consistent results by applying both sklearn and statsmodels.

Next, you'll see how to create a GUI in Python to gather input from users, and then display the prediction results.

GUI used for the Multiple Linear Regression in Python

This is where the real fun begins!

Why not create a Graphical User Interface (GUI) that will allow users to input the independent variables in order to get the predicted result?

It may be that some of the users may not know much about inputting the data in the Python code itself, so it makes sense to create them a simple interface where they can manage the data in a simplified manner.

You can even create a batch file to launch the Python program, and so the users will just need to double-click on the batch file in order to launch the GUI.

Here is the full Python code for your ultimate Regression GUI:

```
import pandas as pd
from sklearn import linear_model
import tkinter as tk
import matplotlib.pyplot as plt
from matplotlib.backends.backend_tkagg import FigureCanvasTkAgg

Stock_Market = {'Year': [2017,2017,2017,2017,2017,2017,2017,2017,2017,2017,2017,2017,2016,2016,201
                'Month': [12, 11,10,9,8,7,6,5,4,3,2,1,12,11,10,9,8,7,6,5,4,3,2,1],
                'Interest_Rate': [2.75,2.5,2.5,2.5,2.5,2.5,2.5,2.25,2.25,2.25,2,2,2,1.75,1.75,1.75
                'Unemployment_Rate': [5.3,5.3,5.3,5.3,5.4,5.6,5.5,5.5,5.5,5.6,5.7,5.9,6,5.9,5.8,6.
                'Stock_Index_Price': [1464,1394,1357,1293,1256,1254,1234,1195,1159,1167,1130,1075,
                }
```

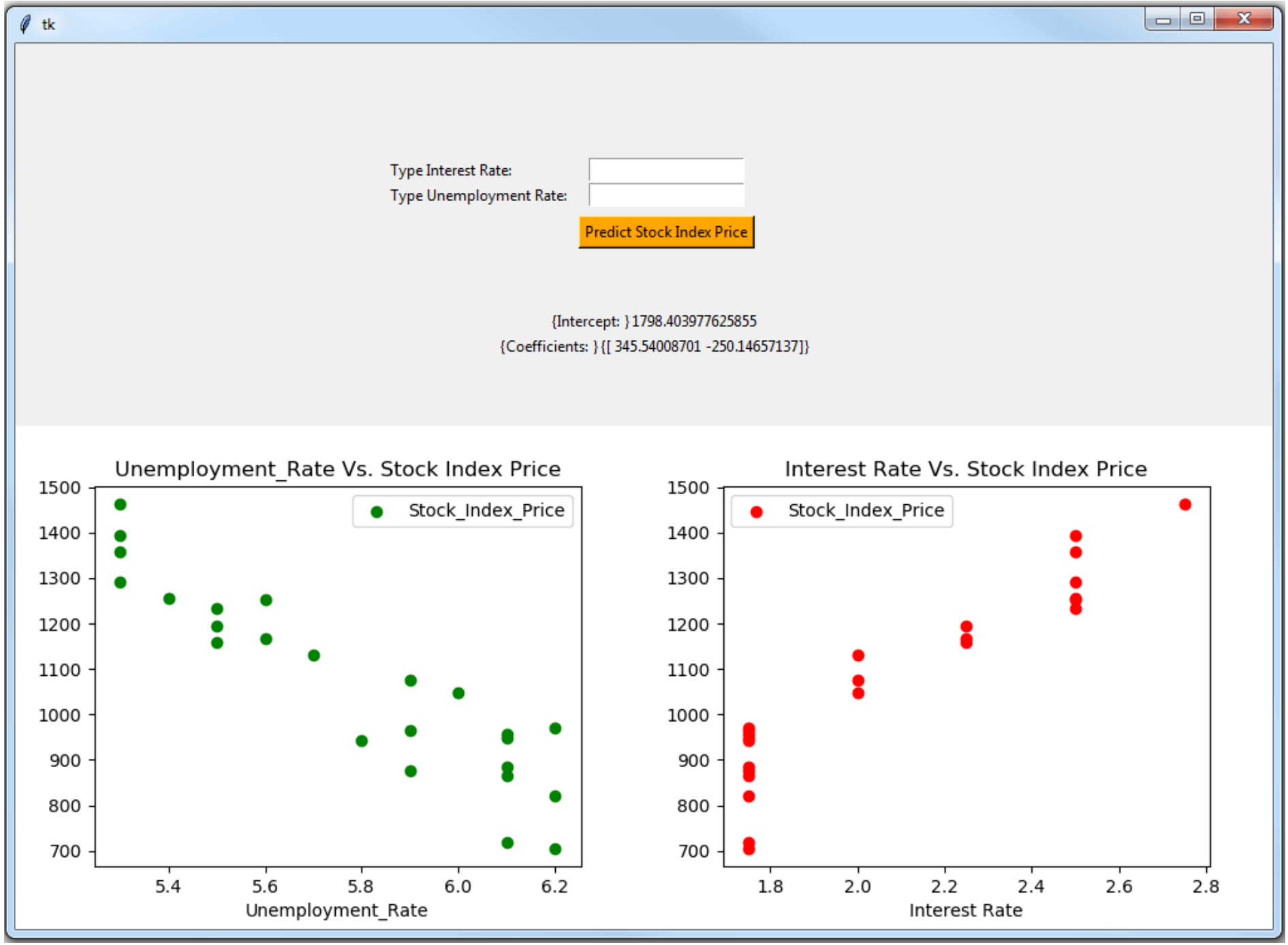
```
df = pd.DataFrame(Stock_Market,columns=['Year','Month','Interest_Rate','Unemployment_Rate','Stock_

X = df[['Interest_Rate','Unemployment_Rate']].astype(float) # here we have 2 input variables for m
Y = df['Stock_Index_Price'].astype(float) # output variable (what we are trying to predict)

# with sklearn
regr = linear_model.LinearRegression()
regr.fit(X, Y)

print('Intercept: \n', regr.intercept_)
print('Coefficients: \n', regr.coef_)

# tkinter GUI
root= tk.Tk()

canvas1 = tk.Canvas(root, width = 500, height = 300)
canvas1.pack()

# with sklearn
Intercept_result = ('Intercept: ', regr.intercept_)
label_Intercept = tk.Label(root, text=Intercept_result, justify = 'center')
canvas1.create_window(260, 220, window=label_Intercept)
```

```
# with sklearn
Coefficients_result  = ('Coefficients: ', regr.coef_)
label_Coefficients = tk.Label(root, text=Coefficients_result, justify = 'center')
canvas1.create_window(260, 240, window=label_Coefficients)

# New_Interest_Rate label and input box
label1 = tk.Label(root, text='Type Interest Rate: ')
canvas1.create_window(100, 100, window=label1)

entry1 = tk.Entry (root) # create 1st entry box
canvas1.create_window(270, 100, window=entry1)

# New_Unemployment_Rate label and input box
label2 = tk.Label(root, text=' Type Unemployment Rate: ')
canvas1.create_window(120, 120, window=label2)

entry2 = tk.Entry (root) # create 2nd entry box
canvas1.create_window(270, 120, window=entry2)

def values():
    global New_Interest_Rate #our 1st input variable
    New_Interest_Rate = float(entry1.get())
```

```
    global New_Unemployment_Rate #our 2nd input variable
    New_Unemployment_Rate = float(entry2.get())

    Prediction_result  = ('Predicted Stock Index Price: ', regr.predict([[New_Interest_Rate ,New_U
    label_Prediction = tk.Label(root, text= Prediction_result, bg='orange')
    canvas1.create_window(260, 280, window=label_Prediction)

button1 = tk.Button (root, text='Predict Stock Index Price',command=values, bg='orange') # button
canvas1.create_window(270, 150, window=button1)

#plot 1st scatter
figure3 = plt.Figure(figsize=(5,4), dpi=100)
ax3 = figure3.add_subplot(111)
ax3.scatter(df['Interest_Rate'].astype(float),df['Stock_Index_Price'].astype(float), color = 'r')
scatter3 = FigureCanvasTkAgg(figure3, root)
scatter3.get_tk_widget().pack(side=tk.RIGHT, fill=tk.BOTH)
ax3.legend(['Stock_Index_Price'])
ax3.set_xlabel('Interest Rate')
ax3.set_title('Interest Rate Vs. Stock Index Price')

#plot 2nd scatter
figure4 = plt.Figure(figsize=(5,4), dpi=100)
```

```
ax4 = figure4.add_subplot(111)
ax4.scatter(df['Unemployment_Rate'].astype(float),df['Stock_Index_Price'].astype(float), color = '
scatter4 = FigureCanvasTkAgg(figure4, root)
scatter4.get_tk_widget().pack(side=tk.RIGHT, fill=tk.BOTH)
ax4.legend(['Stock_Index_Price'])
ax4.set_xlabel('Unemployment_Rate')
ax4.set_title('Unemployment_Rate Vs. Stock Index Price')

root.mainloop()
```

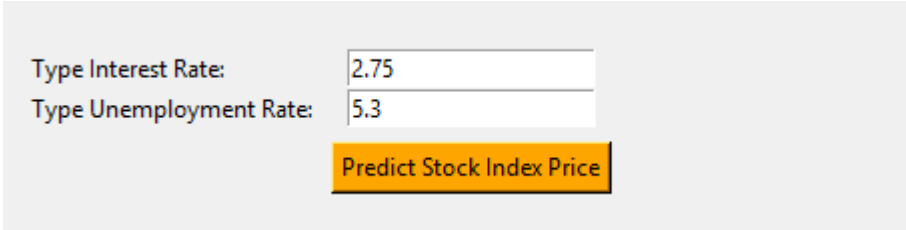
Once you run the code, you'll see this GUI, which includes the output generated by sklearn and the scatter diagrams:


tk
Type Interest Rate:
Type Unemployment Rate:
Predict Stock Index Price
{Intercept: }1798.403977625855
{Coefficients: }{[345.54008701 -250.14657137]}
Unemployment_Rate Vs. Stock Index Price
Stock_Index_Price
1500
1400
1300
1200
1100
1000
900
800
700
5.4
5.6
5.8
6.0
6.2
Unemployment_Rate
Interest Rate Vs. Stock Index Price
Stock_Index_Price
1500
1400
1300
1200
1100
1000
900
800
700
1.8
2.0
2.2
2.4
2.6
2.8
Interest Rate

Recall that earlier we made a prediction by using the following values:

- Interest Rate = 2.75
- Unemployment Rate = 5.3

Type those values in the input boxes, and then click on the ‘Predict Stock Index Price’ button:



You’ll now see the predicted result of 1422.86, which matches with the value you saw before.


tk
Type Interest Rate:
2.75
Type Unemployment Rate:
5.3
Predict Stock Index Price
{Intercept: } 1798.403977625855
{Coefficients: }{[345.54008701 -250.14657137]}
{Predicted Stock Index Price: }{[1422.86238865]}
Unemployment_Rate Vs. Stock Index Price
Stock_Index_Price
Unemployment_Rate
Interest Rate Vs. Stock Index Price
Stock_Index_Price
Interest Rate

You may also want to check the following tutorial to learn more about embedding charts on a tkinter GUI.

Conclusion

Linear regression is often used in Machine Learning. You have seen some examples of how to perform multiple linear regression in Python using both sklearn and statsmodels.

Before applying linear regression models, make sure to check that a *linear* relationship exists between the dependent variable (i.e., what you are trying to predict) and the independent variable/s (i.e., the input variable/s).

← Previous Post

Next Post →

Tutorials

Python Tutorials

R Tutorials

Julia Tutorials

Batch Scripts Tutorials

MS Access Tutorials

Excel Tutorials

Copyright © 2020 | Data to Fish

Privacy Policy

Terms of Service

All rights reserved ©