**Multiple linear regression**

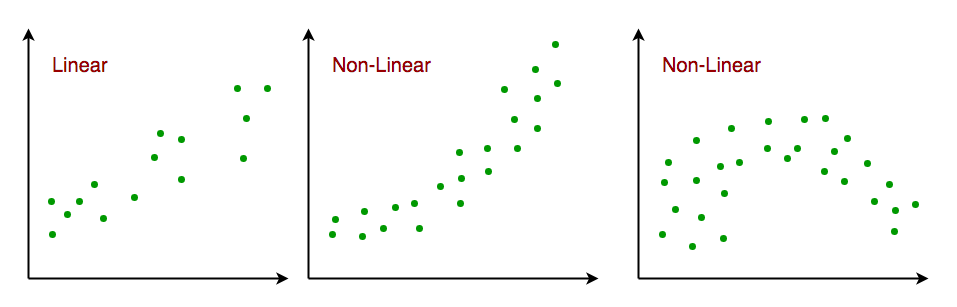
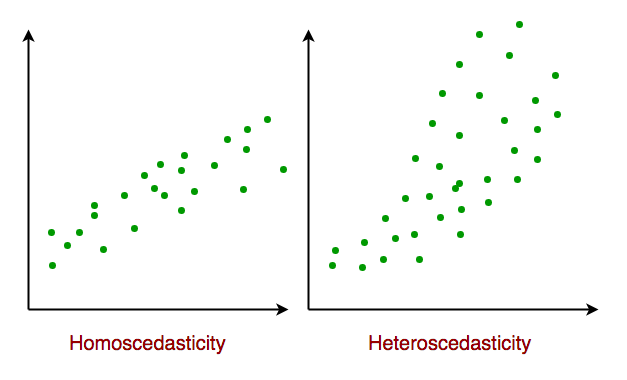
Multiple linear regression attempts to model the relationship between **two or more features** and a response by fitting a linear equation to observed data.

Clearly, it is nothing but an extension of Simple linear regression.

Consider a dataset with **p** features(or independent variables) and one response(or dependent variable).  
Also, the dataset contains **n** rows/observations.

## Assumptions

Given below are the basic assumptions that a linear regression model makes regarding a dataset on which it is applied:

* **Linear relationship**: Relationship between response and feature variables should be linear. The linearity assumption can be tested using scatter plots. As shown below, 1st figure represents linearly related variables where as variables in 2nd and 3rd figure are most likely non-linear. So, 1st figure will give better predictions using linear regression.  
  
* **Little or no multi-collinearity**: It is assumed that there is little or no multicollinearity in the data. Multicollinearity occurs when the features (or independent variables) are not independent from each other.
* **Little or no auto-correlation**: Another assumption is that there is little or no autocorrelation in the data. Autocorrelation occurs when the residual errors are not independent from each other. You can refer here for more insight into this topic.
* **Homoscedasticity**: Homoscedasticity describes a situation in which the error term (that is, the “noise” or random disturbance in the relationship between the independent variables and the dependent variable) is the same across all values of the independent variables. As shown below, figure 1 has homoscedasticity while figure 2 has heteroscedasticity.  
  

As we reach to the end of this article, we discuss some applications of linear regression below.

## Applications:

**1. Trend lines:** A trend line represents the variation in some quantitative data with passage of time (like GDP, oil prices, etc.). These trends usually follow a linear relationship. Hence, linear regression can be applied to predict future values. However, this method suffers from a lack of scientific validity in cases where other potential changes can affect the data

**2. Economics:** Linear regression is the predominant empirical tool in economics. For example, it is used to predict consumption spending, fixed investment spending, inventory investment, purchases of a country’s exports, spending on imports, the demand to hold liquid assets, labor demand, and labor supply.

**3. Finance:** Capital price asset model uses linear regression to analyze and quantify the systematic risks of an investment.

**4. Biology:** Linear regression is used to model causal relationships between parameters in biological systems.

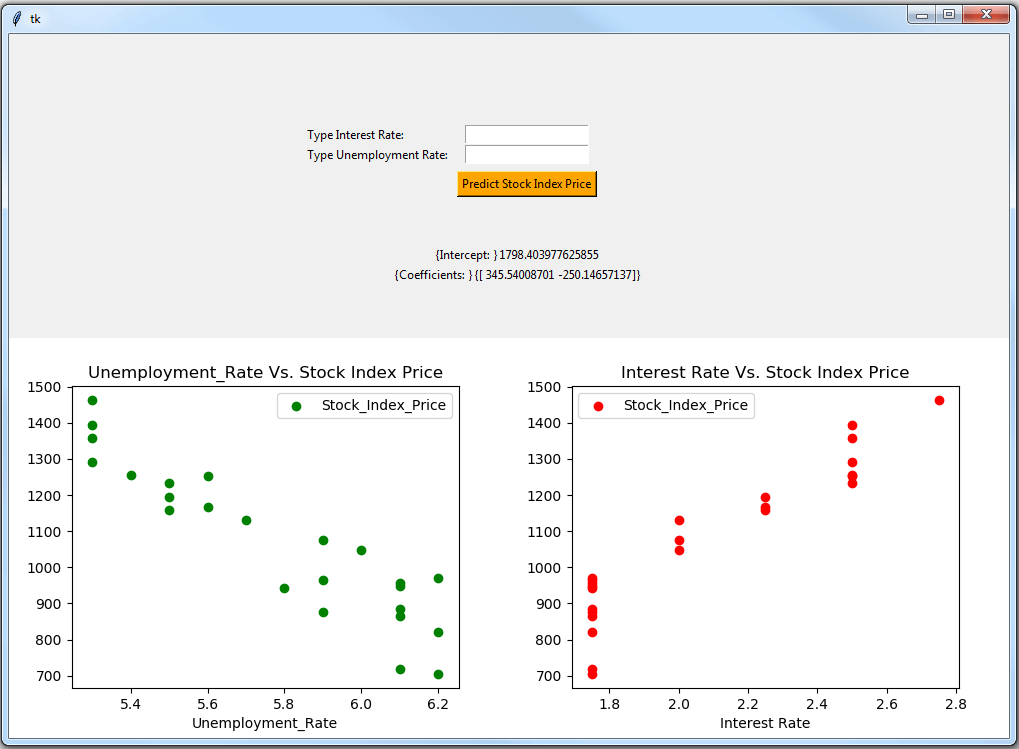
# Example of Multiple Linear Regression in Python

In this tutorial, I’m going to show you 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 would be able to create the following interface in Python:



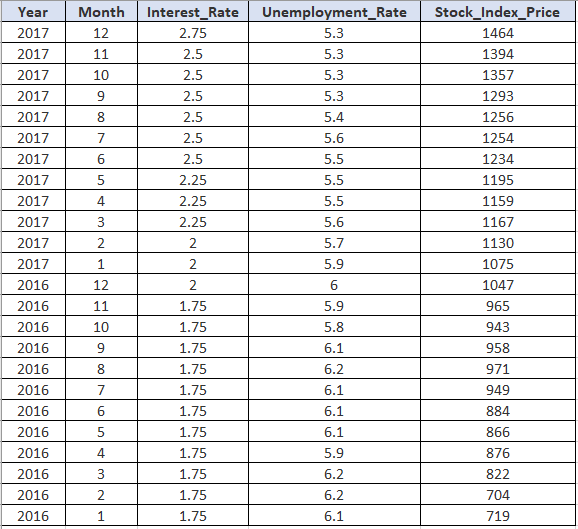
## Example of Multiple Linear Regression in Python

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 linearitysection).

Let’s now jump into the data-set that we’ll be using:



To start, you may capture the above data-set in Python using pandas DataFrame:

from pandas import DataFrame

Stock\_Market = {'Year': [2017,2017,2017,2017,2017,2017,2017,2017,2017,2017,2017,2017,2016,2016,2016,2016,2016,2016,2016,2016,2016,2016,2016,2016],

'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': [2.75,2.5,2.5,2.5,2.5,2.5,2.5,2.25,2.25,2.25,2,2,2,1.75,1.75,1.75,1.75,1.75,1.75,1.75,1.75,1.75,1.75,1.75],

'Unemployment\_Rate': [5.3,5.3,5.3,5.3,5.4,5.6,5.5,5.5,5.5,5.6,5.7,5.9,6,5.9,5.8,6.1,6.2,6.1,6.1,6.1,5.9,6.2,6.2,6.1],

'Stock\_Index\_Price': [1464,1394,1357,1293,1256,1254,1234,1195,1159,1167,1130,1075,1047,965,943,958,971,949,884,866,876,822,704,719]

}

df = DataFrame(Stock\_Market,columns=['Year','Month','Interest\_Rate','Unemployment\_Rate','Stock\_Index\_Price'])

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 the Interest\_Rate (independent variable); and
* The Stock\_Index\_Price (dependent variable) and the Unemployment\_Rate (independent variable)

To perform a quick linearity check, you can use scatter diagrams (utilizing the matplotlib library):

from pandas import DataFrame

import matplotlib.pyplot as plt

Stock\_Market = {'Year': [2017,2017,2017,2017,2017,2017,2017,2017,2017,2017,2017,2017,2016,2016,2016,2016,2016,2016,2016,2016,2016,2016,2016,2016],

'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': [2.75,2.5,2.5,2.5,2.5,2.5,2.5,2.25,2.25,2.25,2,2,2,1.75,1.75,1.75,1.75,1.75,1.75,1.75,1.75,1.75,1.75,1.75],

'Unemployment\_Rate': [5.3,5.3,5.3,5.3,5.4,5.6,5.5,5.5,5.5,5.6,5.7,5.9,6,5.9,5.8,6.1,6.2,6.1,6.1,6.1,5.9,6.2,6.2,6.1],

'Stock\_Index\_Price': [1464,1394,1357,1293,1256,1254,1234,1195,1159,1167,1130,1075,1047,965,943,958,971,949,884,866,876,822,704,719]

}

df = DataFrame(Stock\_Market,columns=['Year','Month','Interest\_Rate','Unemployment\_Rate','Stock\_Index\_Price'])

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()

plt.scatter(df['Unemployment\_Rate'], df['Stock\_Index\_Price'], color='green')

plt.title('Stock Index Price Vs Unemployment Rate', fontsize=14)

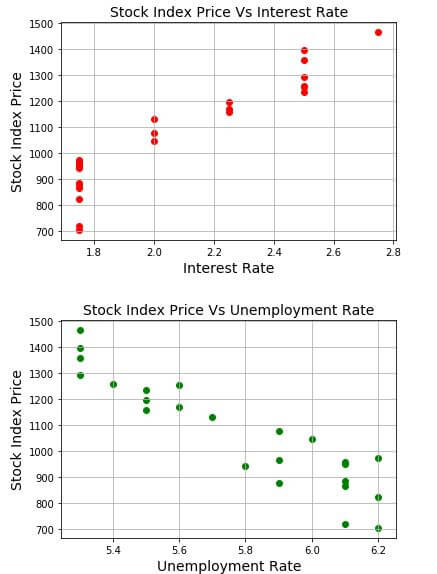
plt.xlabel('Unemployment Rate', fontsize=14)

plt.ylabel('Stock Index Price', fontsize=14)

plt.grid(True)

plt.show()

Once you run the code, you’ll get the following two diagrams:



As you can see, a linear relationship exists in both cases:

* In the first case, when interest rates go up, the stock index price also goes up
* In the second case, when 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 I’ll show you both methods for illustration purposes.

You may then copy the below code into Python, before we dive into the results:

from pandas import DataFrame

from sklearn import linear\_model

import statsmodels.api as sm

Stock\_Market = {'Year': [2017,2017,2017,2017,2017,2017,2017,2017,2017,2017,2017,2017,2016,2016,2016,2016,2016,2016,2016,2016,2016,2016,2016,2016],

'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': [2.75,2.5,2.5,2.5,2.5,2.5,2.5,2.25,2.25,2.25,2,2,2,1.75,1.75,1.75,1.75,1.75,1.75,1.75,1.75,1.75,1.75,1.75],

'Unemployment\_Rate': [5.3,5.3,5.3,5.3,5.4,5.6,5.5,5.5,5.5,5.6,5.7,5.9,6,5.9,5.8,6.1,6.2,6.1,6.1,6.1,5.9,6.2,6.2,6.1],

'Stock\_Index\_Price': [1464,1394,1357,1293,1256,1254,1234,1195,1159,1167,1130,1075,1047,965,943,958,971,949,884,866,876,822,704,719]

}

df = DataFrame(Stock\_Market,columns=['Year','Month','Interest\_Rate','Unemployment\_Rate','Stock\_Index\_Price'])

X = df[['Interest\_Rate','Unemployment\_Rate']] # here we have 2 variables for multiple regression. If you just want to use one variable for simple linear regression, then use X = df['Interest\_Rate'] for example.Alternatively, you may add additional variables within the brackets

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()

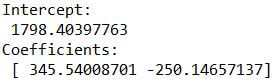
predictions = model.predict(X)

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:



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

Stock\_Index\_Price = (Intercept) + (Interest\_Rate coef)\*X1 + (Unemployment\_Rate coef)\*X2

And once you plug the numbers:

Stock\_Index\_Price = (1798.4040) + (345.5401)\*X1 + (-250.1466)\*X2

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

Linear Regression

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

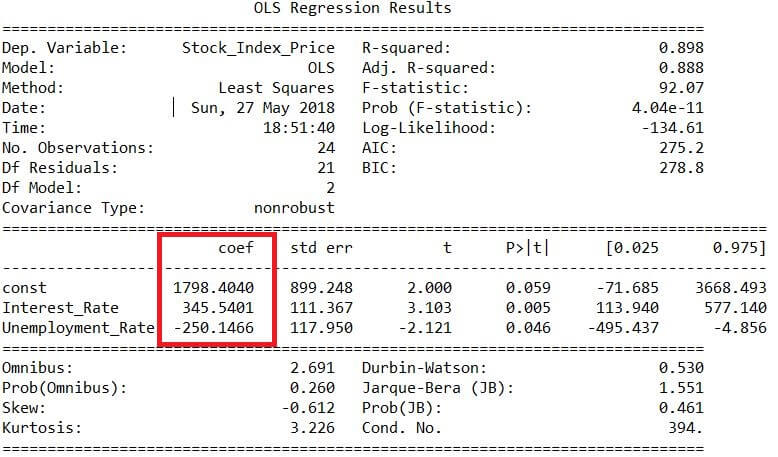
* Interest Rate = 2.75 (i.e., X1= 2.75)
* Unemployment Rate = 5.3 (i.e., X2= 5.3)

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

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.)



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, I’ll show you 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 GUI that will allow users 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 the batch file in order to display the GUI.

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

from pandas import DataFrame

from sklearn import linear\_model

import tkinter as tk

import statsmodels.api as sm

Stock\_Market = {'Year': [2017,2017,2017,2017,2017,2017,2017,2017,2017,2017,2017,2017,2016,2016,2016,2016,2016,2016,2016,2016,2016,2016,2016,2016],

'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': [2.75,2.5,2.5,2.5,2.5,2.5,2.5,2.25,2.25,2.25,2,2,2,1.75,1.75,1.75,1.75,1.75,1.75,1.75,1.75,1.75,1.75,1.75],

'Unemployment\_Rate': [5.3,5.3,5.3,5.3,5.4,5.6,5.5,5.5,5.5,5.6,5.7,5.9,6,5.9,5.8,6.1,6.2,6.1,6.1,6.1,5.9,6.2,6.2,6.1],

'Stock\_Index\_Price': [1464,1394,1357,1293,1256,1254,1234,1195,1159,1167,1130,1075,1047,965,943,958,971,949,884,866,876,822,704,719]

}

df = DataFrame(Stock\_Market,columns=['Year','Month','Interest\_Rate','Unemployment\_Rate','Stock\_Index\_Price'])

X = df[['Interest\_Rate','Unemployment\_Rate']] # here we have 2 input variables for multiple regression. If you just want to use one variable for simple linear regression, then use X = df['Interest\_Rate'] for example.Alternatively, you may add additional variables within the brackets

Y = df['Stock\_Index\_Price'] # 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\_)

# with statsmodels

X = sm.add\_constant(X) # adding a constant

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

predictions = model.predict(X)

# tkinter GUI

root= tk.Tk()

canvas1 = tk.Canvas(root, width = 1200, height = 450)

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)

# with statsmodels

print\_model = model.summary()

label\_model = tk.Label(root, text=print\_model, justify = 'center', relief = 'solid', bg='LightSkyBlue1')

canvas1.create\_window(800, 220, window=label\_model)

# 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\_Unemployment\_Rate]]))

label\_Prediction = tk.Label(root, text= Prediction\_result, bg='orange')

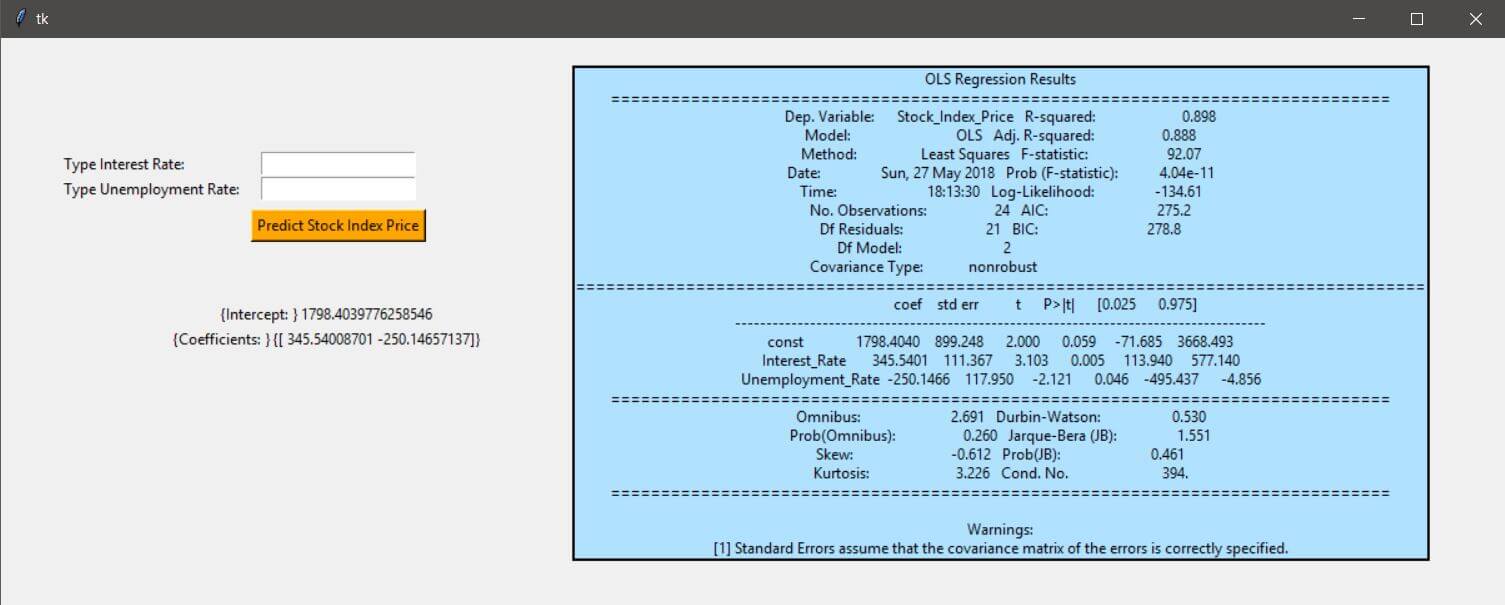
canvas1.create\_window(260, 280, window=label\_Prediction)

button1 = tk.Button (root, text='Predict Stock Index Price',command=values, bg='orange') # button to call the 'values' command above

canvas1.create\_window(270, 150, window=button1)

root.mainloop()

And when you run the code, you’ll see this GUI:



The left-hand-side of the GUI displays the output generated by sklearn:

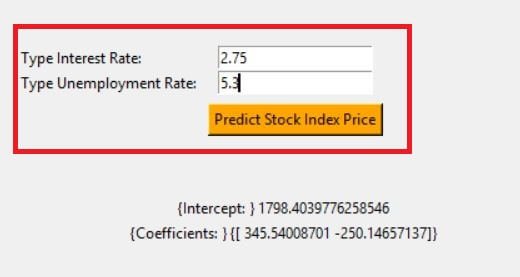
* It includes 2 input boxes, so that the user may type values for the interest and unemployment rates to get the predicted result
* It also includes the intercept and coefficients generated by sklearn

While, the right-hand side of the GUI displays the output generated by statsmodels.

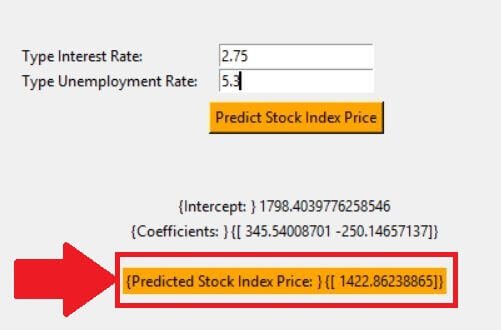
Recall that earlier we made a prediction by using the following values:

* Interest Rate = 2.75
* Unemployment Rate = 5.3

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



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



## Yet another GUI

To be gentle, the GUI that we just saw is not the most appealing one.

And so, in this section, I’ll share the code that will contain:

* the output generated by sklearn; and
* the scatter diagrams that we used earlier to check for linearity

And here is the full Python code:

from pandas import DataFrame

from sklearn import linear\_model

import tkinter as tk

import matplotlib.pyplot as plt

from matplotlib.backends.backend\_tkagg import FigureCanvasTkAgg

Stock\_Market = {'Year': [2017,2017,2017,2017,2017,2017,2017,2017,2017,2017,2017,2017,2016,2016,2016,2016,2016,2016,2016,2016,2016,2016,2016,2016],

'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': [2.75,2.5,2.5,2.5,2.5,2.5,2.5,2.25,2.25,2.25,2,2,2,1.75,1.75,1.75,1.75,1.75,1.75,1.75,1.75,1.75,1.75,1.75],

'Unemployment\_Rate': [5.3,5.3,5.3,5.3,5.4,5.6,5.5,5.5,5.5,5.6,5.7,5.9,6,5.9,5.8,6.1,6.2,6.1,6.1,6.1,5.9,6.2,6.2,6.1],

'Stock\_Index\_Price': [1464,1394,1357,1293,1256,1254,1234,1195,1159,1167,1130,1075,1047,965,943,958,971,949,884,866,876,822,704,719]

}

df = DataFrame(Stock\_Market,columns=['Year','Month','Interest\_Rate','Unemployment\_Rate','Stock\_Index\_Price'])

X = df[['Interest\_Rate','Unemployment\_Rate']].astype(float) # here we have 2 input variables for multiple regression. If you just want to use one variable for simple linear regression, then use X = df['Interest\_Rate'] for example.Alternatively, you may add additional variables within the brackets

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\_Unemployment\_Rate]]))

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 to call the 'values' command above

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()

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 = 'g')

scatter4 = FigureCanvasTkAgg(figure4, root)

scatter4.get\_tk\_widget().pack(side=tk.RIGHT, fill=tk.BOTH)

ax4.legend()

ax4.set\_xlabel('Unemployment\_Rate')

ax4.set\_title('Unemployment\_Rate Vs. Stock Index Price')

root.mainloop()

Once you run the code, you would get the GUI below:

