Week 5 Laboratory Activity

In the past few weeks, you have experienced some of the different skills needed as a Data Scientist, these are mainly on collecting data, wrangling data and presenting the data through descriptive statistics and visualization. From Week 5 to Week 7, we will be doing some activities relating to the basics of Machine Learning.

Many of the principles and theories of statistical machine learning are about the behavior of different learning algorithms with more, or with less data. We can demonstrate one of the fundamental machine learning algorithms, Linear Regression, using functions of one variable and we can examine these behaviors by changing our data and our models. This exploration gives us a guided introduction to the concepts behind the theory. In this week's activity, you need to explore the linear regression and you are encouraged to have open discussions on the Moodle Ed Forums The activities are separated into 3 topics:

Activity 1: Examines basics to linear algorithms with an example.
Activity 2: Examines linear regression and the generation of noisy data from a "true" model.
Activity 3: Examines linear regression fitting different orders of the polynomial.

Activity 1: Basics of Linear Algorithms

Introduction

Machine Learning (ML) is a method of data analysis to perform a specific task based on the existence of patterns in data and hence infer (predict) some outcome, instead of having explicit instructions. The statistics needed for this is called inferential statistics, as opposed to descriptive statistics that you have been introduced to. This field is considered as a subset of the Pattern Recognition field.

Detecting patterns is an important task in Machine Learning, and it is usually the first step needed. Much like how we (humans) look at descriptive statistics to get an idea of what we can infer from it. Let's start by looking at patterns.

Patterns

We start with our regular importing of libraries for Python.

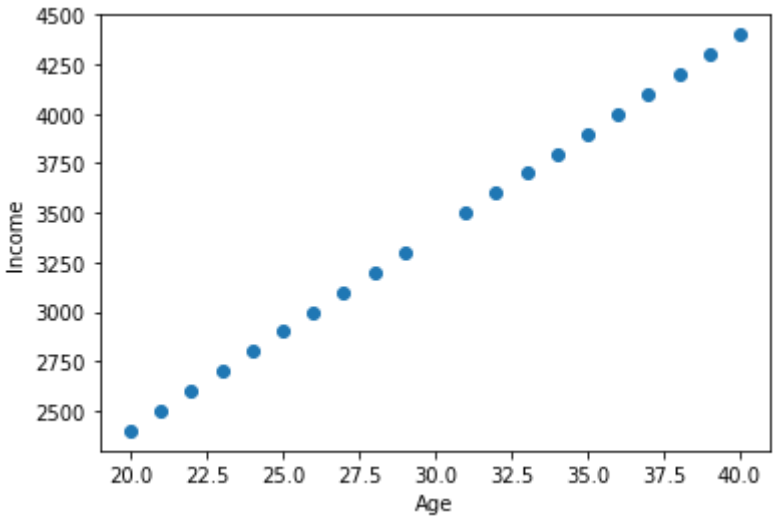
```
import pandas as pd
import random                          # added for data generation
import matplotlib.pyplot as plt
%matplotlib inline             # don't forget this line
```

Assume you have the following dataset (in a DataFrame form) about information of age and income of 40 people.

```
df = pd.DataFrame(
    {'Age' : [20,21,22,23,24,25,26,27,28,29,31,32,33,34,35,36,37,38,39,40],
'Income' :[2400,2500,2600,2700,2800,2900,3000,3100,3200,3300,3500,
  3600,3700,3800,3900,4000,4100,4200,4300,4400]}
)
df.head()
```

Do you see any relation between `Age` and `Income`? We have already learned how to create different plots. Let's use a plot which can show the relation between two variables. By relation we mean how the values of `Income` would change with the changes in the values of `Age`. We can show this relationship with a Scatter plot. Let's show a scatter plot of Age versus Income.

```
plt.scatter(df['Age'],df['Income'])
# plt.scatter(df.Age,df.Income)      # if you like to use the '.' notation
plt.xlabel('Age')
plt.ylabel('Income')
plt.show()
```



Considering the plot above, can you estimate what the Income of a 41 year old is?

A. 2000
B. 3400
C. 3500
D. 4000
E. 4500

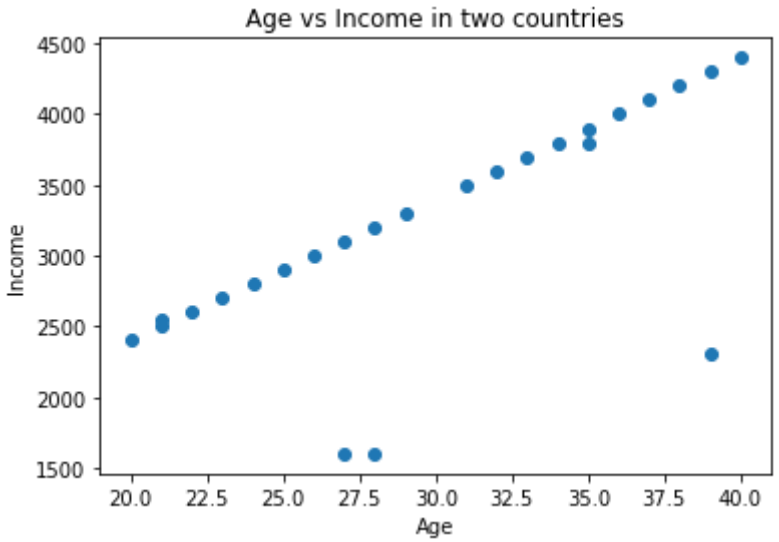
If you chose 4500, you are right! As you can see by increasing the age values, the income values also increase as well. Congratulations! You have detected the general pattern in this data set. The pattern in this data set is simple, as you increase the `Age`, `Income` also increases.

Now let us change the data set slightly and add information of other people from a different country to see if the general pattern changes or not.

```
dfNewCountry= pd.DataFrame(
    {'Age' : [21,27,28,39,35],
     'Income' :[2550,1600,1600,2300,3800]}
)
dfAllCountries=pd.concat([df,dfNewCountry])
```

Before we proceed, we have introduced a “new” pandas function called `concat()`. What does it do? It merges the two dataframes on the row axis. Compare this with the `merge()` function. Let’s now plot the scatter plot.

```
plt.scatter(dfAllCountries['Age'],dfAllCountries['Income'])
plt.xlabel('Age')
plt.title('Age vs Income in two countries')
plt.ylabel('Income')
plt.show()
```



Again, try to answer this question. Considering the plot above, can you guess which of the following Income would be the income of a person who is 41?

A. 2000
B. 3400
C. 3500
D. 4000
E. 4500

If your answer is 4500, you are right! Otherwise, please bear in your mind the following tip:

- In ML, Pattern refers to a ***general pattern*** and we ignore the minority of data points which have different values from the majority when we detect the pattern. In the plot 'Age vs Income in two countries', we ignore information of 3 people whose income values were far from the rest. We just determine the pattern based on the general increasing relation between values of Age and Income. It seems that by detecting the general pattern, we are able to predict the income of people of varying ages.

In the plot above, *Age* is called an ***independent variable*** or ***predictor*** whereas *Income* is a dependent ***variable*** or ***responding*** variable. We use the independent variable to predict the value of the dependent variable, e.g, we use variable 'Age' (independent variable) to predict variable 'Income' (dependent Variable).

What you did to guess the income of a person who has 41, was an example of prediction. You looked at the data and understood the relation between Age and Income for those who are between 18 and 40 and predicted the Income of a person who is 41. That's exactly what ML does. We give a dataset to a machine learning algorithm, it finds general patterns (by building models) and does predictions based on the given dataset. Then, the model predicts the income on the detected general pattern.

Introduction to the Concept of Linear Regression

Regression for continuous dependent variables with the help of independent variables is called Linear Regression. This is as opposed to Logistic Regression where it is used to predict the categorical dependent variable with the help of independent variables. Logistic regression is used mainly for classification. We will investigate classification later.

Let’s take a gentle path to understanding linear regression. First, create a data set and then we define a linear function to represent the general pattern between dependent and independent variables.

```
random.seed(123)
indVar=random.sample(range(1,  30),  20)
indVar
```

The first thing to note in any programming is that when we use functions to generate random numbers, the generation of the random numbers are pseudo random. What this means, in short, is that computers generate the random numbers using some mathematical function and are based on a starting point. If this starting point is known, then the “random” numbers generated can be reproduced. This is where the term setting the seed number means in computing. Hence, to ensure that all of you have the same random numbers generated, we set the seed for the `random` library. Remember, you imported the library earlier and we set it by calling `random.seed(123)`.

After that, we can assign the variable `indVar` by generating 20 numbers, within the range of 1 to 30 using `random.sample(range(1, 30), 20)`.

Note that some of you may still be getting a different range of numbers as the setting of the seed may not be equal for all OS or machine architecture (32-bit, 64-bit, processor type, etc.).

Now we have a variable `IndVar` that contains a list of 20 numbers.

```
indVar
```

Should return the following output.

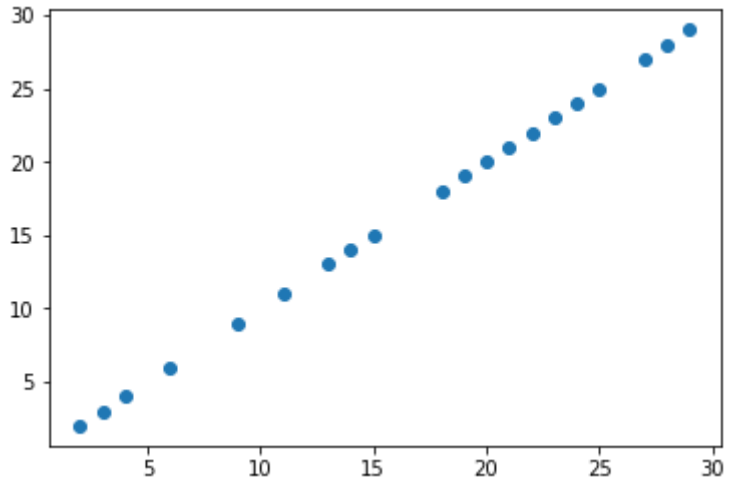
```
[2, 9, 3, 25, 14, 28, 4, 29, 13, 18, 20, 11, 19, 22, 27, 15, 6, 24, 21, 23]
```

Let's create a set of dependent variables that are of the same values, for illustration purposes only.

```
depVar = indVar[:]
```

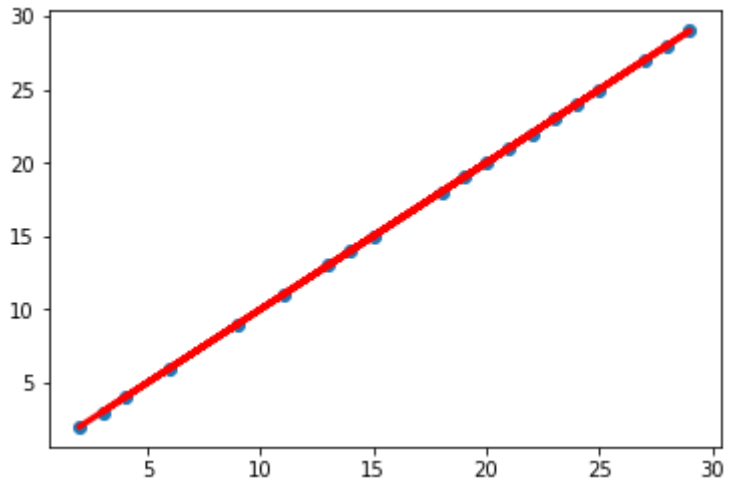
Now we have 2 sets of data points, where the independent and dependent data are the same. The purpose of this is purely for illustration only. Let's plot the data using a scatter plot to view it.

```
plt.scatter(indVar,depVar)
plt.show()
```



Let's plot a line across those points.

```
plt.scatter(indVar,depVar)
plt.plot(indVar,depVar,'r-', linewidth=3)  # red line, of thickness 3
plt.show()
```



What we are trying to illustrate so far is that if the numbers are accurately related in a linear manner, we can draw a straight line through it. Let's now assume that this relationship is the true relationship between the independent and dependent variables. Let's call these the true numbers.

```
trueX = indVar[:]
trueY = depVar[:]
```

Let's generate what we may find as real data (again, this is just simulated). Usually, in any data collection, the data that is recorded may not be the exact figure, for example measurements from analog devices, or measuring non-digital measures, or measurements that are recorded manually by humans. We simulate this by adding some noise (noise in this case means additional data that are random (unpredictable) and carries no real useful information).

Let's firstly add additional data points to our independent variable.

```
noiseXVals=random.sample(range(1,30),10)
indVar.extend(noiseXVals)
```

We now have 30 data points for our independent variable. You can have a look at the `indVar` values. What we need to do now is to add another 10 values to our dependent variables, `depVar` but the values should be not exactly that of the new independent variables but values that are slightly different. What we are going to do here is to pick those new values from noiseXVals using the for "`x in noiseXVals`" and add some minor noise to it.

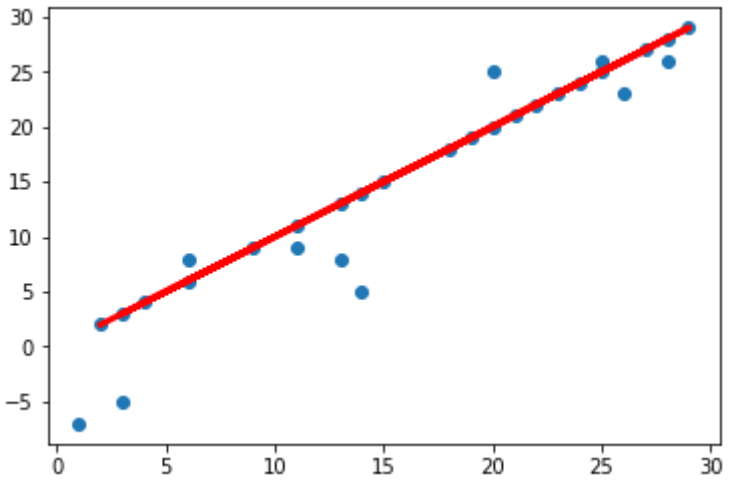
```
noiseYVals=[x+round(random.uniform(-10,6)) for x in noiseXVals]
depVar.extend(noiseYVals)
```

Before you execute the `extend()`, you may want to have a look at the values of `noiseXVals` and `noiseYVals`. The `random.uniform()` function, creates floating

point numbers between -10 and 8 (you can change the parameters if you like) and the `round()` function just rounds up the number to the nearest integer. Now,

we replot the graph and have a look at it.

```
plt.scatter(indVar,depVar)
# the red line represents the true data
plt.plot(trueX,trueY,'r-', linewidth=3)
plt.show()
```



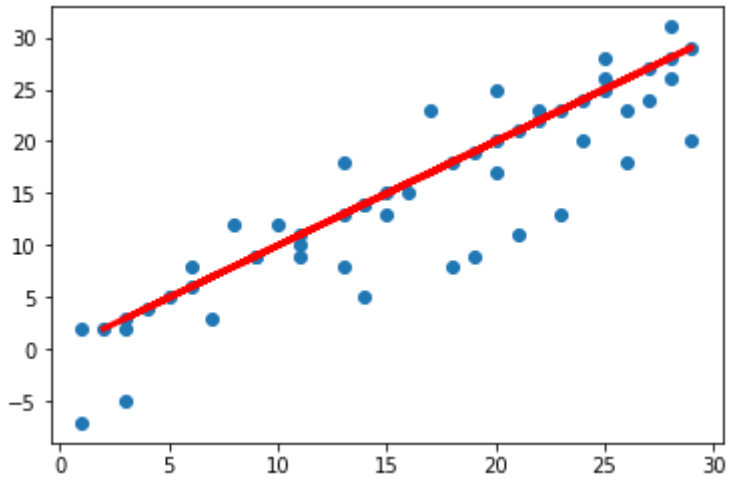
Assuming that the new dots are the data that you have. The red straight line now doesn't seem to be the best fit. The red line seems to be a bit higher than it should be. Are you able to find a mathematical function which represents the general pattern in your data set?

What if data is scattered more?! Let us add more noise to this data set by following commands and see if we can still represent the general trend by a line.

```
noiseXVals=random.sample(range(1,30),25)
indVar.extend(noiseXVals)

noiseYVals=[x+round(random.uniform(-10,6))for x in noiseXVals]
depVar.extend(noiseYVals)

plt.scatter(indVar,depVar)
plt.plot(trueX,trueY,'r-', linewidth=3)
plt.show()
```

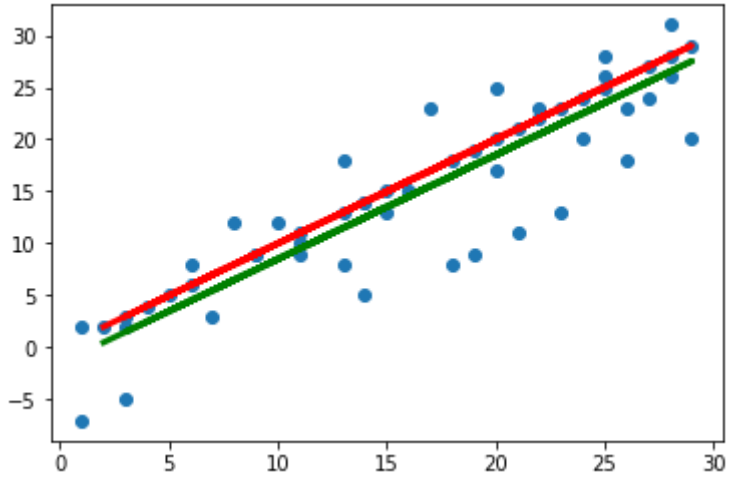


Do you think that the red line which you plotted is the best line? Let's plot another line which has a y-axis value that is smaller (by 1.5).

```
trueYNew=[x-1.5 for x in trueY]
```

Let's plot the alternate line.

```
plt.scatter(indVar,depVar)
plt.plot(trueX,trueY,'r-',    linewidth=3)
plt.plot(trueX,trueYNew,'g-', linewidth=3)
plt.show()
```



Does the green line look like a better fit? How can you evaluate it?

Linear Regression in Python

We have taken great lengths to get a non-mathematical approach to understanding the best fit, with the high level idea of what a linear regression is like. Now, we are going to look at a

library in Python that we can use to create linear regression models. In your week 1 lectures, we mentioned that the following are the basic libraries:

- Numpy
- Pandas
- MatPlotLib
- SciPy (also Scikit-Learn, which is built on-top of SciPy)

Although there are other packages in Python, we use the SciPy library to create our model.

```
from scipy.stats import linregress
```

The function `scipy.stats.linregress()` returns 5 values, and they are the slope (of the linear regression line), the intercept (where the line crosses on the y-axis), the r-value (the correlation - we called this the r-value as you would have seen in our earlier lectures), the p-value (the statistical relevance value) and the standard error (which is the computed error based on the estimated gradient). Some of you will have many questions on this, but let's leave it to another course for the time being (you can actually read a lot about this). We assign the returned values to 5 variables after calling the `linregress()` function by providing the independent and dependent variables.

```
slope, intercept, r_value, p_value, std_err = linregress(indVar,depVar)
```

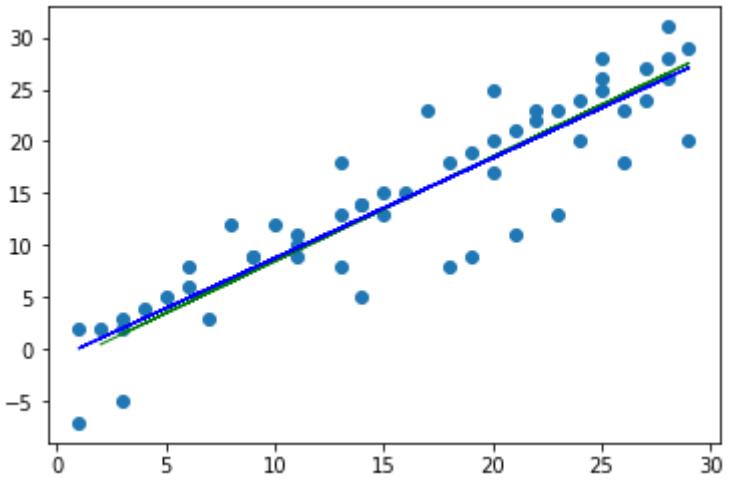
You can view the values returned as:

```
print("slope: %f         intercept: %f" % (slope, intercept))
print("r-value: %f" % r_value)
print("p-value: %f" % p_value)
```

With the slope and intercept, we can compute the values that we needed for the new line that is the result of the linear regression function. We use the "`for xi in indVar`" to go through each value of the `indVar` to obtain the corresponding points for the line.

```
line = [slope*xi + intercept for xi in indntVar]

# our 'raw' data
plt.scatter(indVar, depVar)
# our earlier manual estimated line (in green)
plt.plot(trueX,trueYNew,'g-', linewidth=1)
# the line created by the linregress() function (in blue)
plt.plot(indVar,line,'b-', linewidth=1)
plt.show()
```



You will see the best line is slightly different from the earlier estimated line that we had earlier. You have completed your first linear regression, the very first step towards Machine Learning.

Practice 1:
Create a linear regression model for the following data, which is a data for baseball runs by the different batters. You will firstly need to determine which variable should be considered as independent and which variable should be considered as dependent.

```
df = pd.DataFrame(
    {'Name' : ['Mike','Aaron','Brad','Steve','George','Mitchell',
               'Shaun','Glenn','Pat','Robert','David'],
     'Age' : [22, 25, 25, 26, 28, 28, 31, 32, 33, 39, 44],
     'Runs' :[1000, 980, 900, 900, 828, 700, 672, 662, 655, 600, 557]}
)
```

Practice 2:
Create another linear regression model using our earlier DataFrame `dfAllCountries`?

Practice 3:
Do you think that you can use your model to predict income of a 70 years old person? Why or why not?

I hope that it has been easy to follow for this week, as we will slowly have laboratories that are not step by step (nor explained) and you would need to put in effort to search for the meaning of the errors and debug the issues that come along.

For the rest of week 5, you are to go through Activities 2 and 3 on your own (provided as .ipynb files). There is no practice in them, but you have to study it carefully.