# Section I.

Hello! This lesson will focus on very basic yet very important topics in machine learning, like linear regression (and if necessary, logistic regression). It will center on the equally basic yet equally relevant process of data processing/cleaning that must take place before implementation begins. To add a bit of context–I'm going to go with the recommended level of expertise from the assignment doc on canvas. So imagine this audience as being freshman-level members of a data science club who have taken both stat 107 and IS 205 (information science, computer Science, and x + data science majors). That said, they should be familiar with basic syntax and what not and how to work with data, but will need guidance in knowing what to spot in the pre-processing phase, and what to extract in the machine learning phase. Let’s get started.

# Section II.

At its core, machine learning, a major area within the wider area of Artificial Intelligence, is the process by which mathematical/computational processes are automated, leading to them–and the wider processes of which they are a part–becoming quicker, cleaner, and usually more accurate! To us data, computer, and information scientists, machine learning is more than an exciting buzzword. For some of you, it will represent the bread and butter of much of your work going forward, because, as we all know, the realm of computing is becoming increasingly automated! Efficiency goes up across the board when such algorithms are implemented, and these algorithms are present in nearly every computing subfield–from web development to robotics! For the purposes of this club, we’ll focus on a more concise definition of Machine Learning as it applies to data science. From SimpliLearn, “Machine learning as technology helps analyze large chunks of data, easing the tasks of data scientists in an automated process and is gaining a lot of prominence and recognition. Machine learning has changed the way data extraction and interpretation works by involving automatic sets of generic methods that have replaced traditional statistical techniques.”. Sounds awesome, right? What if you’d automated everything in STAT 107? Well, let’s try and think of machine learning itself as more of a *journey,* going forward, because that's what it is. It is not as simple as writing a line of code or two. Something called *data processing* must occur before implementation of even the most basic of machine learning algorithms begins. According to UpGrad, “Data preprocessing in Machine Learning refers to the technique of preparing (cleaning and organizing) the raw data to make it suitable for building and training Machine Learning models.” So, think of machine learning as a party you plan on hosting during this holiday season, and data-processing as the tidying up and planning that must occur if the party is to be at all successful–the two go hand in hand!

# Section III.

For our demonstration today, we’ll be focusing on Linear Regression, a concept of machine learning that serves as a form of predictive analysis. We’ll be using the Sklearn module’s functions to actually implement the linear regression work. In order to implement this properly you’ll need the assistance of the plt module, because you’re all human, and visualization is a must for us to come up with the best ways to approach data problems! We’ll also be working with numpy and pandas, (you can actually use either or), and the sm module, as those three, in tandem with one another or separate, provide different methods of accomplishing the same thing! I want to have as many tools under your belt as possible. Let’s look at the core syntax for the tools we’ll be working with. We’ll use the skills you learned in your classes to clean our data, then base our approach to the Machine learning implementation of of a handy source[[1]](#footnote-0). Let’s get started!

| #sklearn and its accompanying functions from sklearn.linear\_model import LinearRegression from sklearn import preprocessing, svm  from sklearn.model\_selection import train\_test\_split  from sklearn.linear\_model import LinearRegression  import numpy as np import csv import pandas as pd  import seaborn as sns  import matplotlib.pyplot as plt |
| --- |

Feel free to look deeper at the individual imported modules. Each of these is extremely useful in other areas of data science. Make sure to remember to include csv for the dataset you’ll be using so it can be loaded in seamlessly. Per JavatPoint, Linear Regression “ shows the linear relationship between the dependent and independent variables, and shows how the dependent variable(y) changes according to the independent variable (x).” So although we’ll be working with data that can be cleaned, I would caution against using data with no discernible dependent and independent variable, as linear regression will not work , or at least will not provide valuable insight, otherwise. If you need a refresher on independent and dependent variables I would check out this cool source. [[2]](#footnote-1)

# Section IV.

Now we’ll demonstrate this concept a bit more in-depth so you can get more familiar with the syntax and structure of this entire process! Our small example here will demonstrate the minimum work needed on your part in order to make the larger program work as intended. This will entail basic data cleaning methods, and how to prepare your data for use in machine learning. Let’s get to it!

We’re going to demonstrate simple linear regression data-cleaning techniques on a dummy dataset, then see how to apply what we’ve learned to an actual dataset.

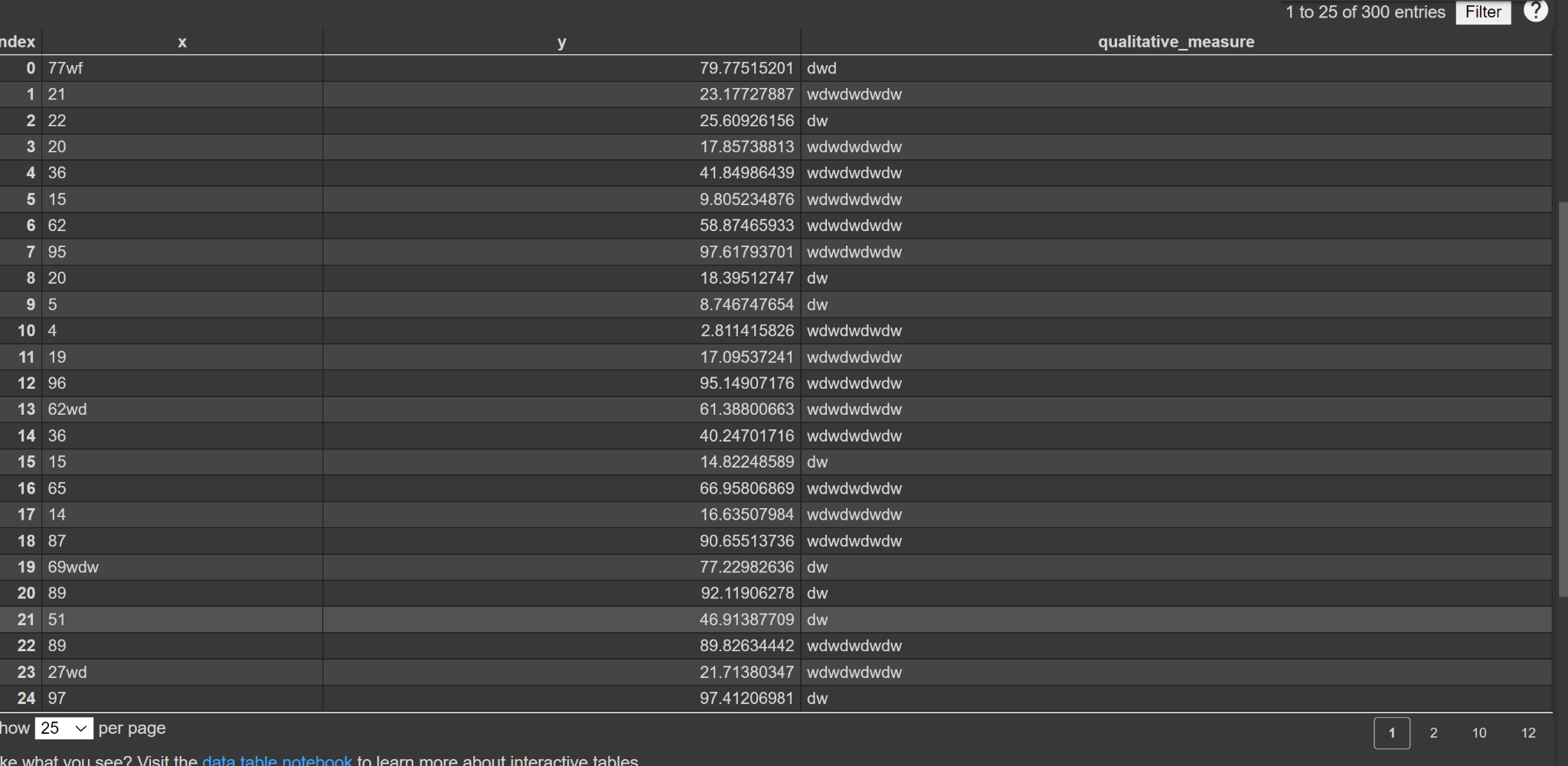
First, let’s load in our data.

#Load in data

df = pd.read\_csv("slr\_data.csv")

df

Notice how we didn’t use traditional boilerplate python to load in the csv. Be sure to take advantage of the powerful functions within your imported modules, like with Pandas! Loading in the csv files you want to analyze through pandas makes it both easier and faster to work with csv files, as it loads the data straight into a data frame and also allows you to view the dataset within your working environment (assuming you’re using a notebook, like jupyter, which works very well with the pandas module.)

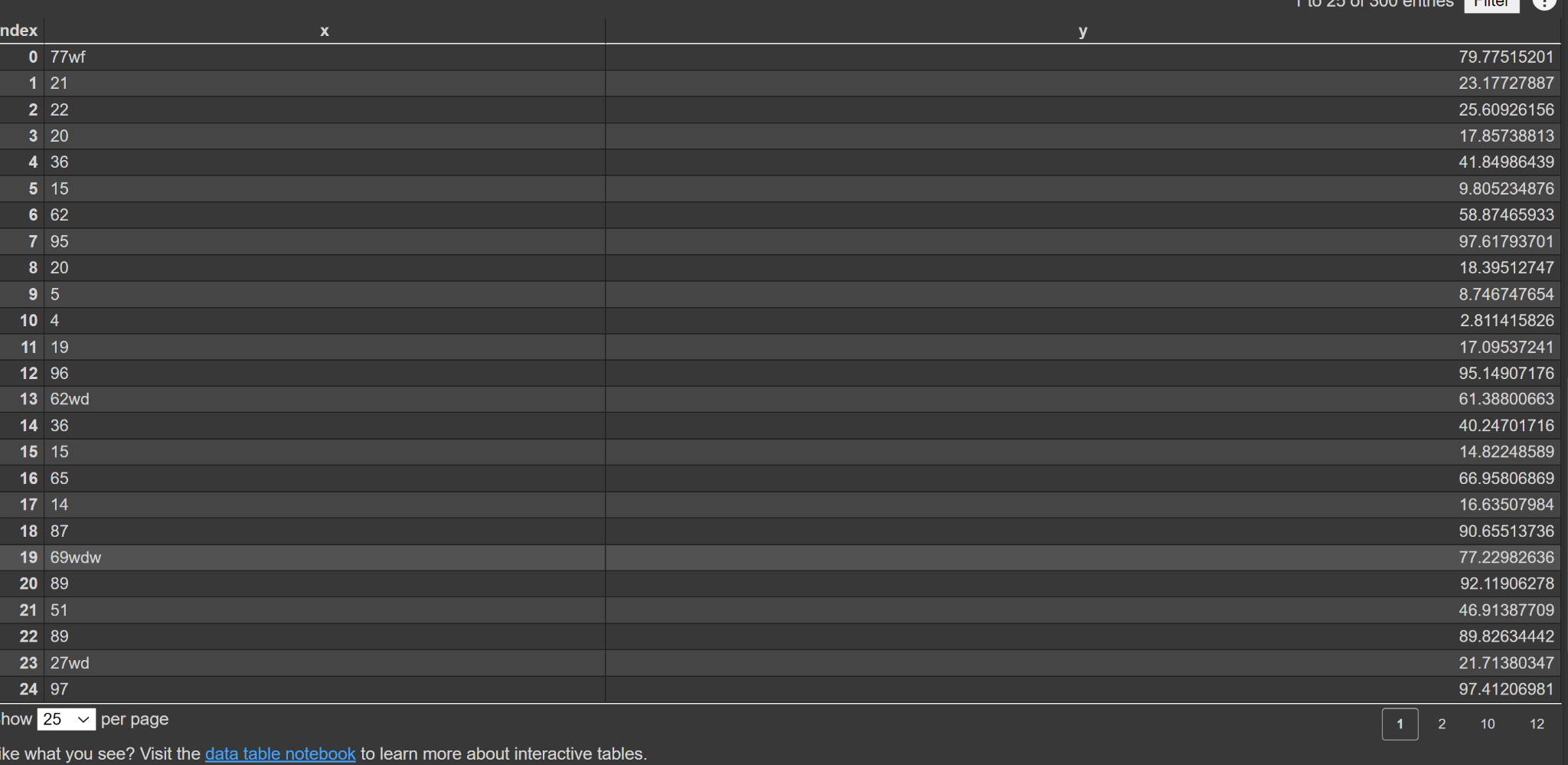


When we look at our data set, we can see that it’s not *exactly* as messy as much of the data you might see in the real-world, but it certainly is ugly enough to require a cleaning. Think for a second what should be done. While you think, look back to the definitions of linear regression outlined in section III. Judging by what you see, what particular columns would one actually need to implement the algorithm? If you said the more quantitative ones, you’d be correct. In a real data set, “x” and “y” will have an identity, and it will be easy to distinguish between the independent and dependent variables. For the purposes of this demonstration, x will be the independent variable and y will be the dependent variable. Let’s reduce our dataset to that particular portion.

#we will construct a new dataframe from the original

df\_x\_y = df[["x","y"]]

df\_x\_y



It’s great that we finally have the variables (columns) we’ll be working with, but the data is still messy. The presence of unwanted characters is one of the most prominent issues you’ll be dealing with in data cleaning outside of irrelevant data (like the column we got rid of in the last step). How might we go about removing the irrelevant characters within some of our data points? Think about what you’ve learned so far in your courses.

What if I told you that there was a way to make it extremely simple? Let’s call on the regex module, a useful pythonic tool for us data science clubbers (especially when it comes to data cleaning).

import re

ex = []

for i in df\_x\_y["x"]:

i = str(i)

i = re.sub("[^0-9]", "", i)

i = int(i)

ex.append(i)

y = []

for x in df\_x\_y["y"]:

x = str(i)

x = re.sub("[^0-9]", "", x)

x = int(x)

y.append(x)

df\_x\_y2 = pd.DataFrame(

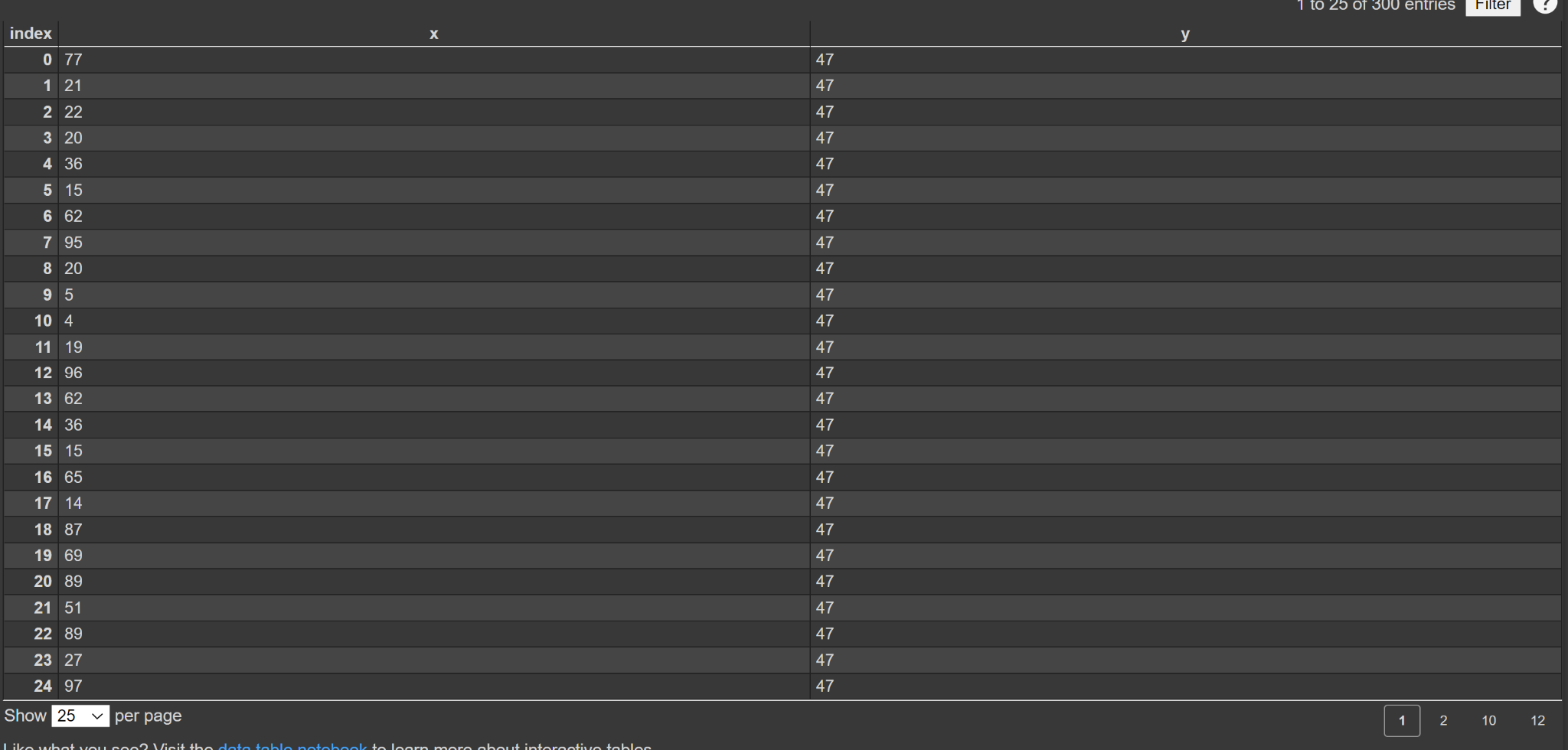
{'x': ex,

'y': y,})

df\_x\_y2

Step through everything carefully. What do you see? Let’s go step-by-step:

1. First, we had to create two different lists and loop through the 2 different columns.
2. In each loop, we had cast each element to a string so that it could be used in regex
3. We then used re’s “sub” function to eliminate all non-numeric characters from the stringed element
4. We then converted each element back to an int–can’t analyze stringed numbers!
5. We used the lists we made to append each element, all numeric and ready to go, to the lists.
6. We created another iteration of our previous dataframe (note: you will do this alot when modifying data frames in pandas). Were we successful? Is everything clean and ready to go? Let’s check.



Looks like everything is clean and character-free! Keep this method in mind when you encounter messy data when mining and scraping out there. Now, on to machine learning.

# Section V (Final Section)

At last, we begin implementing our concepts of machine learning. As you saw in section IV, data-cleaning is a must for any machine learning algorithm, especially the one we’ll be working with. Even if you’re not explicitly looking for an independent or dependent variable (not all models will always involve predictive analysis) you want to be certain that the data you’re working with, which will almost always be in the form of a dataframe, is neat and sorted nicely, free of characters or anything of that sort that cannot be used in computations. You also learned how to eliminate unnecessary columns or sections of data that cannot be worked with. With that, you’re ready to tackle most basic in-need-of-cleaning data sets going forward! Now for our algorithm. Make sure you have the imports ready to go from section II. Let’s dive in:

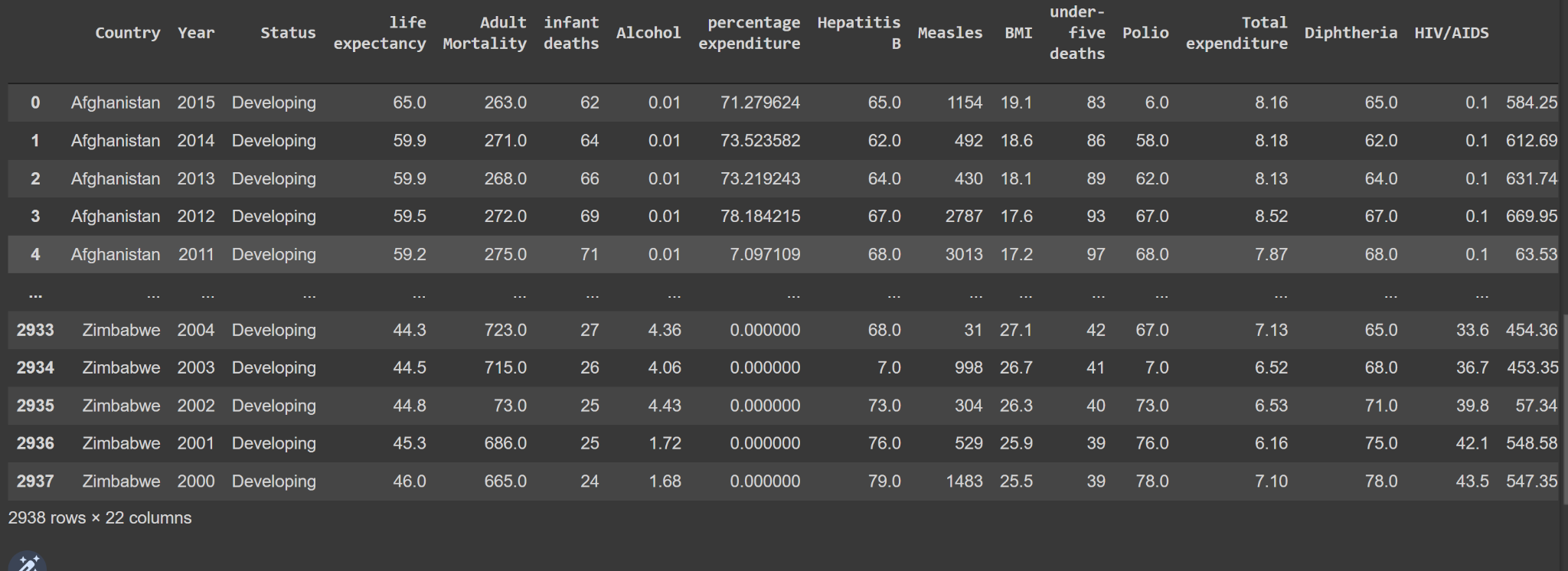
We’ll be working with a dataset from the World Health Organization[[3]](#footnote-2), pre-pandemic, which has a plethora of data to work with, including life expectancy and GDP per capita! For the purposes of this demonstration, we’ll be focusing on the United States, but feel free to work with any other nation once you implement your model. We’ll be using “year” (independent variable) as a predictor for “life expectancy” (dependent variable) to try and investigate whether or not Americans can look forward to increasing life spans in the future or not. Now, since this data set is slightly aged, the predictions we make might not necessarily align with what actually went down, or will go down in the future, but the skills and methods you learn can be applied to an exact data set of that kind, once WHO pushes out their updated statistics for 2022 and so on.

## Step 1:

For this step, we want to first load in our data and prepare our dataframe.

df = pd.read\_csv('Life Expectancy Data.csv')

df



As you can see, all types of data from all types of countries are available. Let’s narrow it down to our country in question, with the independent and dependent variables in question.

df1 = df[["Country","Year","life expectancy "]]

df1

df2 = df1.loc[df1["Country"] == "United States of America"]

df2

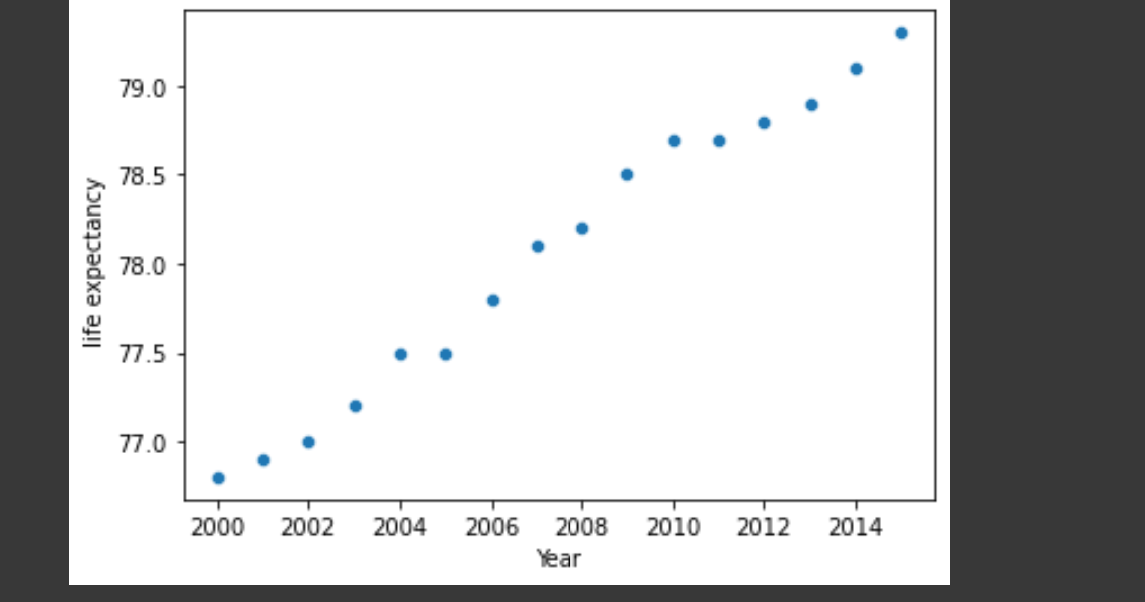


Great! We have the relevant data and are ready to push forward with our linear regression algorithm.

Let’s take a look at our data and the sort of relationship that currently exists between our independent and dependent variables.

df2 = df2[:][:500]

sns.scatterplot(x ="Year", y ="life expectancy ", data = df2)



What do you see? What stands out? There’s already a sort of discernible relationship between our variables, isn’t there? Obviously, you’re not going to have as much luck out there in the wild with your data, but for the purposes of this demonstration, we won't be too hard on you.

Next, we need to reshape our columns, which is technically a part of the data cleaning process, but can be done right before you train your data for the model! Afterward, we need to split our dataset into two different sets. One set will be used for creating the linear regression model we’ve been so eager to implement, and the other will be used to test the accuracy of our model. See the code below!

X = np.array(df2['Year']).reshape(-1, 1)

y = np.array(df2['life expectancy ']).reshape(-1, 1)

X\_train, X\_test, y\_train, y\_test = train\_test\_split(X, y, test\_size = 0.25)

The Sklearn module will now come to our aid. Traditionally, data scientists would have to write these convoluted machine learning algorithms by hand, but, as time passed, modules were developed that deal with some of the basics! Of course, as you become more advanced in your computing career, you’ll learn how to write these and other algorithms on your own, but until then, we’ll favor convenience over the traditional route.

regr = LinearRegression()

regr.fit(X\_train, y\_train)

We use the module to create our LinearRegression object and then use it to fit our training sets!

We will now use the score function, which will allow us to evaluate our model on the training data first (train the model first, always!) and then on our test data.

train\_score = regr.score(X\_train, y\_train)

print("Our model's training score is: ", train\_score)

test\_score = regr.score(X\_test, y\_test)

print("Our model's test score is:", test\_score )

These are our results:

Our model's training score is: 0.9810499040482212

Our model's test score is: 0.98867909715825

Pretty accurate? Think of the score as a percentage. According to our results, both of our scores are pretty accurate. The scores are also very close to one another, which means our model is not “over-fitted”. Per the corporate finance institute, “Overfitting is a modeling error that introduces bias to the model because it is too closely related to the data set.”. Many of you in this club are engineers and scientists, and so know what bias means for a model like this. Too much bias means too little generalizability, a.k.a we could not make many accurate predictions about future data with this particular model!

Remember: We’re human and so need to visualize our work. Always visualize before you dig into the data, and be sure to visualize again once the model is successfully created and tested.

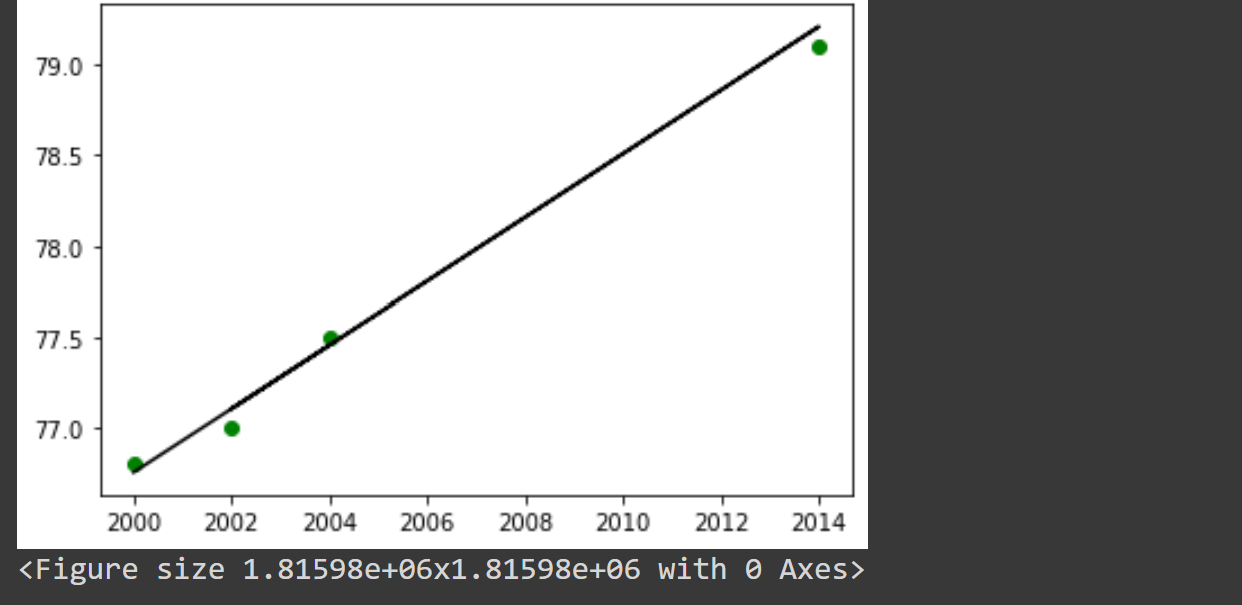
y\_pred = regr.predict(X\_test)

plt.scatter(X\_test, y\_test, color ='g')

plt.plot(X\_test, y\_pred, color ='k')

plt.figure(figsize= (25222,25222))

plt.show()



Here we have it– a model exhibiting the sort of linear relationship that exists between years and American life expectancy. As I mentioned before, be sure to try this and some of the other countries, and with more modern data if possible!

Conclusion

Now, for the send off. Today, you learned about what it takes to implement a machine learning algorithm. Linear regression is certainly one of the most basic of machine learning algorithms, but any data scientist or AI or machine learning engineer worth their salt will tell you that data problems out there in the world are sometimes solved by not overthinking their solutions. Though there exist a wide-variety of tools out there for you to use, and although as computing professionals you understand that there are certainly multiple paths to approach the same problem, with multiple solutions to solve it, sometimes it's best to work smarter instead of harder. Today, you were exposed to the process of data-cleaning/processing that one must undergo before they even attempt to identify patterns and relationships within their data, and how clean everything must be before actual computations are carried out. You were encouraged to visualize your data both before and after the creation of your model, and learned that visualization is key in data work. Finally, you implemented the model itself using some of Python’s most powerful libraries. It's no wonder that Python is often the go-to for data science work. Can you imagine doing this in C++? Wait, you said yes? Perhaps we’ll cover that next time!

1. <https://machinelearningknowledge.ai/linear-regression-in-python-sklearn-with-example/> [↑](#footnote-ref-0)
2. <https://www.thoughtco.com/independent-and-dependent-variable-examples-606828> [↑](#footnote-ref-1)
3. <https://www.kaggle.com/kumarajarshi/life-expectancy-who?ref=hackernoon.com> [↑](#footnote-ref-2)