DATA SCIENCE (HTTPS://WWW.ANALYTICSVIDHYA.COM/BLOG/CATEGORY/DATA-SCIENCE/)
MACHINE LEARNING (HTTPS://WWW.ANALYTICSVIDHYA.COM/BLOG/CATEGORY/MACHINE-LEARNING/)
PYTHON (HTTPS://WWW.ANALYTICSVIDHYA.COM/BLOG/CATEGORY/PYTHON-2/)

A Complete Python Tutorial to Learn Data Science from Scratch

KUNAL JAIN (HTTPS://WWW.ANALYTICSVIDHYA.COM/BLOG/AUTHOR/KUNALJ/), JANUARY 14, 2016 LOGIN TO BOOKMARK THIS AR...

Overview

- This article is a complete tutorial to learn data science using python from scratch
- It will also help you to learn basic data analysis methods using python
- You will also be able to enhance your knowledge of machine learning algorithms

Introduction

It happened a few years back. After working on SAS for more than 5 years, I decided to move out of my comfort zone. Being a data scientist (https://courses.analyticsvidhya.com/courses/introduction-to-data-science-2?utm_source=blog&utm_medium=learnDSwithPythonScratcharticle), my hunt for other useful tools was ON! Fortunately, it didn't take me long to decide – Python was my appetizer.

I always had an inclination for coding. This was the time to do what I really loved. Code. Turned out, coding was actually quite easy!

I learned the basics of Python (https://courses.analyticsvidhya.com/courses/introduction-to-data-science?utm_source=blog&utm_medium=learnDSwithPythonScratcharticle) within a week. And, since then, I've not only explored this language to the depth, but also have helped many other to learn this language. Python was originally a general purpose language. But, over the years, with strong community support, this language got dedicated library for data analysis and predictive modeling.

Due to lack of resource on python for data science (https://courses.analyticsvidhya.com/courses/introduction-to-data-science-2?utm_source=blog&utm_medium=learnDSwithPythonScratcharticle), I decided to create this tutorial to help many others to learn python faster. In this tutorial, we will take bite sized information about how to use Python for Data Analysis, chew it till we are comfortable and practice it at our own end.

A complete python tutorial from scratch for data science. (https://s3-ap-south-1.amazonaws.com/av-blog-media/wp-content/uploads/2016/01/Learn-Data-Science-with-Python-from-Scratch.png)

You can also check out the 'Introduction to Data Science (https://courses.analyticsvidhya.com/courses/introduction-to-data-science-2?utm_source=blog&utm_medium=learnDSwithPythonScratch oduction to the world of data science. It includes modules on Python, ng with multiple practical projects to get your hands dirty.

Table of Contents

Let's get started!

1. Basics of Python for Data Analysis

Why learn Python for data analysis?

Python has gathered a lot of interest recently as a choice of language for data analysis. I had basics of Python (https://www.analyticsvidhya.com/blog/2014/03/sas-vs-vs-python-tool-learn/utm_source=blog&utm_medium=learnDSwithPythonScratcharticle) some time back. Here are some reasons which go in favour of learning Python:

- Open Source – free to install
- Awesome online community
- Very easy to learn
- Can become a common language for data science and production of web based analytics products.

Needless to say, it still has few drawbacks too:

- It is an interpreted language rather than compiled language – hence might take up more CPU time. However, given the savings in programmer time (due to ease of learning), it might still be a good choice.

Python 2.7 v/s 3.4

This is one of the most debated topics in Python. You will invariably cross paths with it, specially if you are a beginner. There is no right/wrong choice here. It totally depends on the situation and your need to use. I will try to give you some pointers to help you make an informed choice.

Why Python 2.7?

1. Awesome community support! This is something you'd need in your early days. Python 2 was released in late 2000 and has been in use for more than 15 years.
2. Plethora of third-party libraries! Though many libraries have provided 3.x support but still a large number of modules work only on 2.x versions. If you plan to use Python for specific applications like web-development with high reliance on external modules, you might be better off with 2.7.
3. Some of the features of 3.x versions have backward compatibility and can work with 2.7 version.

Why Python 3.4?

1. Cleaner and faster! Python developers have fixed some inherent glitches and minor drawbacks in order to set a stronger foundation for the future. These might not be very relevant initially, but will matter eventually.
2. It is the future! 2.7 is the last release for the 2.x family and eventually everyone has to shift to 3.x versions. Python 3 has released stable versions for past 5 years and will continue the same.

There is no clear winner but I suppose the bottom line is that you should focus on learning Python as a language. Shifting between versions should just be a matter of time. Stay tuned for a dedicated article on Python 2.x vs 3.x in the near future!

How to install Python?

There are 2 approaches to install Python:

- You can download Python dir[illegible]t site (https://www.python.org/download/releases/2.7/) and install individual components and libraries you want

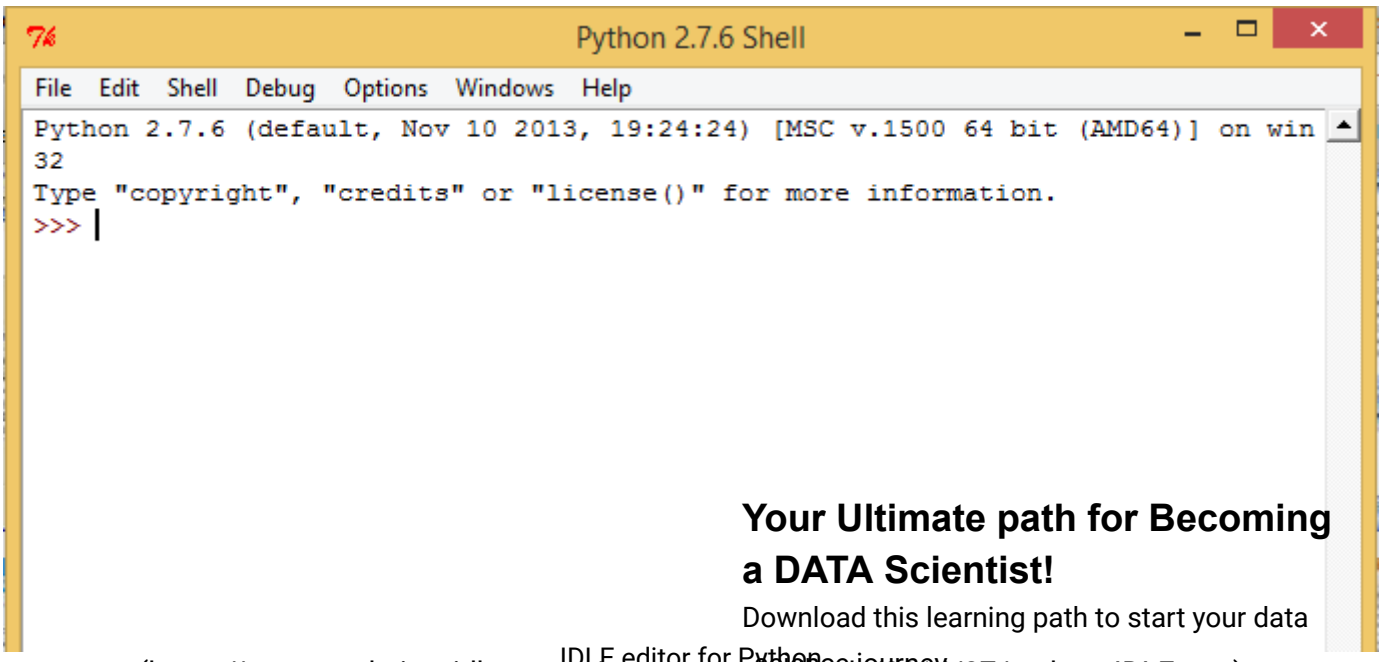
- Alternately, you can download and install a package, which comes with pre-installed libraries. I would recommend downloading Anaconda (https://www.continuum.io/downloads). Another option could be Enthought Canopy Express (https://www.enthought.com/downloads/).

Second method provides a hassle free installation and hence I'll recommend that to beginners. The imitation of this approach is you have to wait for the entire package to be upgraded, even if you are interested in the latest version of a single library. It should not matter until and unless, until and unless, you are doing cutting edge statistical research.

Choosing a development environment

Once you have installed Python, there are various options for choosing an environment. Here are the 3 most common options:

- Terminal / Shell based
- IDLE (default environment)
- iPython notebook – similar to markdown in R



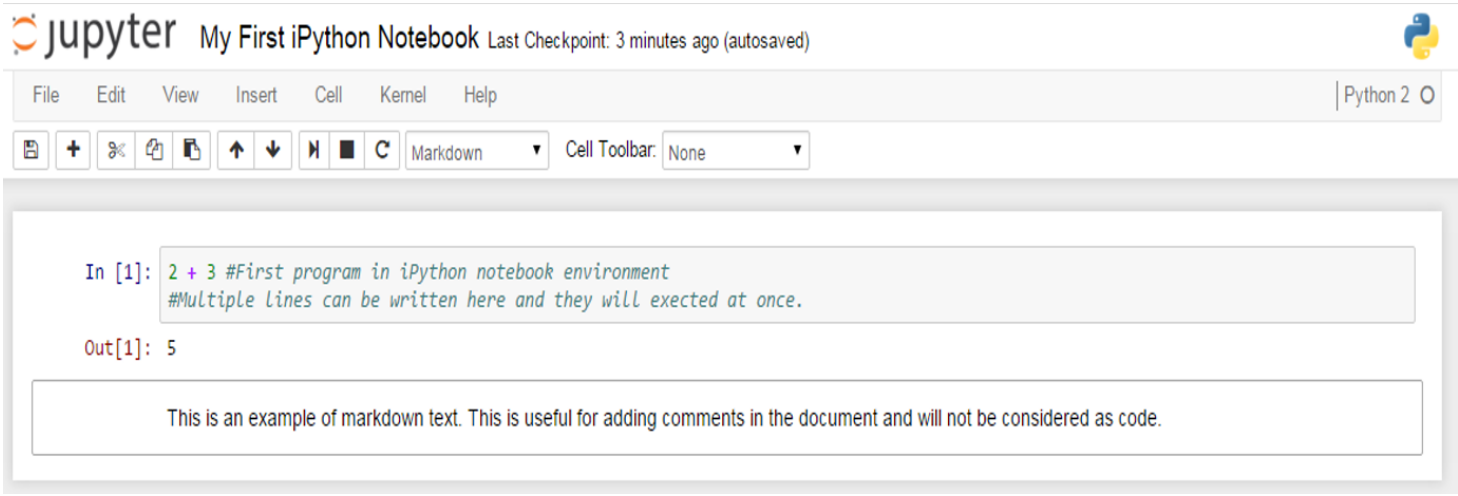
IDLE editor for Python (https://www.analyticsvidhya.com/wp-content/uploads/2014/07/python_IDLE.png)

While the right environment depends on your need, I personally prefer iPython Notebooks a lot. It provides a lot of good features for documenting while writing the code itself and you can choose to run the code in blocks (rather than the line by line execution)

We will use iPython environment for this complete tutorial.

Warming up: Running your first Python program

You can use Python as a simple calculator to start with:



(https://www.analyticsvidhya.com/wp-content/uploads/2016/01/jupyter1.png)

Few things to note

- You can start iPython notebook by writing “ipython notebook” on your terminal / cmd, depending on the OS you are working on
- You can name a iPython notebook by simply clicking on the name – UntitledO in the above screenshot
- The interface shows In [*] for inputs and Out[*] for output.
- You can execute a code by pressing “Shift + Enter” or “ALT + Enter”, if you want to insert an additional row after.

Before we deep dive into problem solving, lets take a step back and understand the basics of Python. As we know that data structures and iteration and conditional constructs form the crux of any language. In Python, these include lists, strings, tuples, dictionaries, for-loop, while-loop, if-else, etc. Let’s take a look at some of these.

2. Python libraries and Data Structures

Python Data Structures

Following are some data structures, which are used in Pytho order to use them as appropriate.

- **Lists** – Lists are one of the most versatile data structure in Python. A list can simply be defined by writing a list of comma separated values in square brackets. Lists might contain items of different types, but usually the items all have the same type. Python lists are mutable and individual elements of a list can be changed.

Here is a quick example to define a list and then access it:

Lists

A list can be simply defined by writing comma separated values in square brackets.

```
In [1]: squares_list = [0,1,4,9,16,25]

In [2]: squares_list
Out[2]: [0, 1, 4, 9, 16, 25]
```

Individual elements of a list can be accessed by writing the index number in square bracket. Please note that the first index of a list is 0 and not 1

```
In [3]: squares_list[0] #Indexing returns the item
Out[3]: 0
```

A range of script can be accessed by having first index and last index

```
In [4]: squares_list[2:4] #Slicing returns a new list
Out[4]: [4, 9]
```

A Negative index accesses the list from end

```
In [5]: squares_list[-2] #It should return the second last element in the list
Out[5]: 16
```

A few common methods applicable to lists include: append() extend() insert() remove() pop() count() sort() reverse()

(https://www.analyticsvidhya.com/wp-content/uploads/2014/07/python_lists.png)

- **Strings** – Strings can simply be defined by use of single ('), double (") or triple (''') inverted commas. Strings enclosed in tripe quotes (''') can span over multiple lines and are used frequently in docstrings (Python's way of documenting functions). \ is used as an escape character. Please note that Python strings are immutable, so you can not change part of strings.

Strings

A string can be simply defined by using single ('), double (") or triple (""") quotation

```
In [6]: greeting = 'Hello'
        print greeting[1]          # Return character on the index 1
        print len(greeting)        # Prints length of string
        print greeting + 'World'   # String Concatenation

        e
        5
        HelloWorld
```

Raw strings can be used to pass on string as is. Python interpretter does not alter the string, if you specify a string to be raw. Raw strings can be defined by adding r to the string

```
In [8]: stmt = r'\n is a newline character by default.'
        print stmt

        \n is a newline character by default.
```

Python strings are immutable and hence can be changed. Doing so will result in an error

```
In [9]: greeting[1:] = 'i' #Trying to change Hello to Hi. Should result in an error

        ---------------------------------------------------------------------------
        TypeError                                 Traceback (most recent call last)
        <ipython-input-9-2da872e33998> in <module>()
        ----> 1 greeting[1:] = 'i' #Trying to change Hello to Hi. Should result in an error

        TypeError: 'str' object does not support item assignment
```

Common string methods include lower(), upper(), strip(), isdigit(), isspace(), find(), replace(), split() and join(). These are usually very helpful when you need to perform data manipulations or cleaning on text fields.

(https://www.analyticsvidhya.com/wp-content/uploads/2014/07/python_strings.png)

- **Tuples** – A tuple is represented by a number of values separated by commas. Tuples are immutable and the output is surrounded by parentheses so that nested tuples are processed correctly. Additionally, even though tuples are immutable, they can hold mutable data if needed.

Since Tuples are immutable and can not change, they are faster in processing as compared to lists. Hence, if your list is unlikely to change, you should use tuples, instead of lists.

Tuples

A tuple is represented by a number of values separated by commas.

```
In [10]: tuple_example = 0, 1, 4, 9, 16, 25

In [11]: tuple_example #output would be enclosed in paranthesis

Out[11]: (0, 1, 4, 9, 16, 25)

In [12]: tuple_example[2] #Single elements can be accessed in similar fashion

Out[12]: 4

In [13]: tuple_example[2] = 6 #Tuples are immutable and hence this should result in error

---------------------------------------------------------------------------
TypeError                                 Traceback (most recent call last)
<ipython-input-13-6c410e816018> in <module>()
----> 1 tuple_example[2] = 6 #Tuples are immutable and hence this should result in error

TypeError: 'tuple' object does not support item assignment
```

(https://www.analyticsvidhya.com/wp-content/uploads/2014/07/Python_tuples.png)

- **Dictionary –** Dictionary is an unordered set of *key: value* pairs, with the requirement that the keys are unique (within one dictionary). A pair of braces creates an empty dictionary: {}.

Dictionary

A dictionary is an unordered set of key: value pairs, with the requirement that the keys are unique (within one dictionary). A pair of braces creates an empty dictionary: {}.

```
In [20]: extensions = {'Kunal': 9073, 'Tavish' : 9128, 'Sunil' : 9223, 'Nitin' : 9330}
         extensions

Out[20]: {'Kunal': 9073, 'Nitin': 9330, 'Sunil': 9223, 'Tavish': 9128}

In [22]: extensions['Mukesh'] = 9150
         extensions

Out[22]: {'Kunal': 9073, 'Mukesh': 9150, 'Nitin': 9330, 'Sunil': 9223, 'Tavish': 9128}

In [23]: extensions.keys()

Out[23]: ['Sunil', 'Tavish', 'Kunal', 'Mukesh', 'Nitin']
```

(https://www.analyticsvidhya.com/wp-content/uploads/2014/07/Python_dictionary.png)

Python Iteration and Conditional Constructs

Like most languages, Python also has a FOR-loop which is the most widely used method for iteration. It has a simple syntax:

```
for i in [Python Iterable]:
  expression(i)
```

Here “Python Iterable” can be a list, tuple or other advanced data structures which we will explore in later sections. Let’s take a look at a simple example, determining the factorial of a number.

```
fact=1
for i in range(1,N+1):
  fact *= i
```

Coming to conditional statements, these are used to execute code fragments based on a condition. The most commonly used construct is if-else, with following syntax:

```
if [condition]:
  __execution if true__
else:
  __execution if false__
```

For instance, if we want to print whether the number N is even or odd:

```
if N%2 == 0:
  print ('Even')
else:
  print ('Odd')
```

Now that you are familiar with Python fundamentals, let’s take a step further. What if you have to perform the following tasks:

1. Multiply 2 matrices
2. Find the root of a quadratic equation
3. Plot bar charts and histograms
4. Make statistical models
5. Access web-pages

If you try to write code from scratch, its going to be a nightmare and you won’t stay on Python for more than 2 days! But lets not worry about that. Thankfully, there are many libraries with predefined which we can directly import into our code and make our life easy.

For example, consider the factorial example we just saw. We

```
math.factorial(N)
```

Off-course we need to import the math library for that. Lets explore the various libraries next.

Python Libraries

Lets take one step ahead in our journey to learn Python by getting acquainted with some useful libraries. The first step is obviously to learn to import them into our environment. There are several ways of doing so in Python:

```
import math as m
```

```
from math import *
```

In the first manner, we have defined an alias m to library math. We can now use various functions from math library (e.g. factorial) by referencing it using the alias m.factorial().

In the second manner, you have imported the entire name space in math i.e. you can directly use factorial() without referring to math.

> Tip: Google recommends that you use first style of importing libraries, as you will know where the functions have come from.

Following are a list of libraries, you will need for any scientific computations and data analysis:

- **NumPy** stands for Numerical Python. The most powerful feature of NumPy is n-dimensional array. This library also contains basic linear algebra functions, Fourier transforms, advanced random number capabilities and tools for integration with other low level languages like Fortran, C and C++
- **SciPy** stands for Scientific Python. SciPy is built on NumPy. It is one of the most useful library for variety of high level science and engineering modules like discrete Fourier transform, Linear Algebra, Optimization and Sparse matrices.
- **Matplotlib** for plotting vast variety of graphs, starting from histograms to line plots to heat plots.. You can use Pylab feature in ipython notebook (ipython notebook –pylab = inline) to use these plotting features inline. If you ignore the inline option, then pylab converts ipython environment to an environment, very similar to Matlab. You can also use [illegible] plot.
- **Pandas** for structured data operations and manipulat[illegible]unging and preparation. Pandas were added relatively recently to Python and have been instrumental in boosting Python's usage in data scientist community.

- **Scikit Learn** for machine learning. Built on NumPy, SciPy and matplotlib, this library contains a lot of effiecient tools for machine learning and statistical modeling including classification, regression, clustering and dimensionality reduction.
- **Statsmodels** for statistical modeling. Statsmodels is a Python module that allows users to explore data, estimate statistical models, and perform statistical tests. An extensive list of descriptive statistics, statistical tests, plotting functions, and result statistics are available for different types of data and each estimator.
- **Seaborn** for statistical data visualization. Seaborn is a library for making attractive and informative statistical graphics in Python. It is based on matplotlib. Seaborn aims to make visualization a central part of exploring and understanding data.
- **Bokeh** for creating interactive plots, dashboards and data applications on modern web-browsers. It empowers the user to generate elegant and concise graphics in the style of D3.js. Moreover, it has the capability of high-performance interactivity over very large or streaming datasets.
- **Blaze** for extending the capability of Numpy and Pandas to distributed and streaming datasets. It can be used to access data from a multitude of sources including Bcolz, MongoDB, SQLAlchemy, Apache Spark, PyTables, etc. Together with Bokeh, Blaze can act as a very powerful tool for creating effective visualizations and dashboards on huge chunks of data.
- **Scrapy** for web crawling. It is a very useful framework for getting specific patterns of data. It has the capability to start at a website home url and then dig through web-pages within the website to gather information.
- **SymPy** for symbolic computation. It has wide-ranging capabilities from basic symbolic arithmetic to calculus, algebra, discrete mathematics and quantum physics. Another useful feature is the capability of formatting the result of the computations as LaTeX code.
- **Requests** for accessing the web. It works similar to the the standard python library urllib2 but is much easier to code. You will find subtle differences with urllib2 but for beginners, Requests might be more convenient.

Additional libraries, you might need:

- **os** for Operating system and file operations
- **networkx** and **igraph** for graph based data manipulations
- **regular expressions** for finding patterns in text data
- **BeautifulSoup** for scrapping web. It is inferior to Scrapy as it will extract information from just a single webpage in a run.

Now that we are familiar with Python fundamentals and additional libraries, lets take a deep dive into problem solving through Python. Yes I mean making a predictive model! In the process, we use some powerful libraries and also come across the next level of data structures. We will take you through the 3 key phases:

1. Data Exploration – finding out more about the data we
2. Data Munging – cleaning the data and playing with it to make it better suit statistical modeling
3. Predictive Modeling – running the actual algorithms and having fun

3. Exploratory analysis in Python using Pandas

In order to explore our data further, let me introduce you to another animal (as if Python was not enough!) – Pandas

(https://www.analyticsvidhya.com/wp-content/uploads/2014/08/pandas.jpg) Image Source: Wikipedia

Pandas is one of the most useful data analysis library in Python (I know these names sounds weird, but hang on!). They have been instrumental in increasing the use of Python in data science community. We will now use Pandas to read a data set from an Analytics Vidhya competition, perform exploratory analysis and build our first basic categorization algorithm for solving this problem.

Before loading the data, lets understand the 2 key data structures in Pandas – Series and DataFrames

Introduction to Series and Dataframes

Series can be understood as a 1 dimensional labelled / indexed array. You can access individual elements of this series through these labels.

A dataframe is similar to Excel workbook – you have column names referring to columns and you have rows, which can be accessed with use of row numbers. The essential difference being that column names and row numbers are known as column and row index, in case of dataframes.

Series and dataframes form the core data model for Pandas in Python. The data sets are first to read into these dataframes and then various operations (e.g. group by, aggregation etc.) can be applied very easily to its columns.

More: 10 Minutes to Pandas (http://pandas.pydata.org/panc

Practice data set – Loan Prediction Problem

You can download the dataset from here (http://datahack.analyticsvidhya.com/contest/practice-problem-loan-prediction-iii). Here is the description of the variables:

```
VARIABLE DESCRIPTIONS:
Variable                  Description
Loan_ID                   Unique Loan ID
Gender                    Male/ Female
Married                   Applicant married (Y/N)
Dependents                Number of dependents
Education                 Applicant Education (Graduate/ Under Graduate)
Self_Employed             Self employed (Y/N)
ApplicantIncome           Applicant income
CoapplicantIncome         Coapplicant income
LoanAmount                Loan amount in thousands
Loan_Amount_Term          Term of loan in months
Credit_History            credit history meets guidelines
Property_Area             Urban/ Semi Urban/ Rural
Loan_Status               Loan approved (Y/N)
```

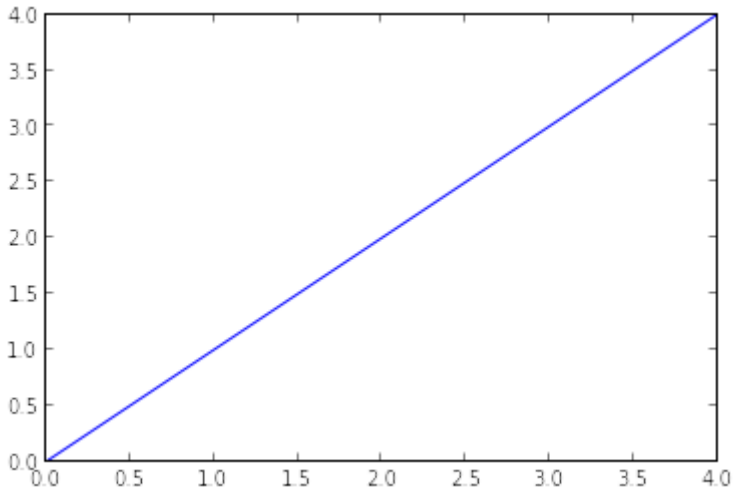
Let's begin with the exploration

To begin, start iPython interface in Inline Pylab mode by typing following on your terminal/windows command prompt:

```
ipython notebook --pylab=inline
```


This opens up iPython notebook in pylab environment, which has a few useful libraries already imported. Also, you will be able to plot your data inline, which makes this a really good environment for interactive data analysis. You can check whether the environment has loade mand (and getting the output as seen in the figure below):

```
plot(arange(5))
```



(https://www.analyticsvidhya.com/wp-content/uploads/2014/08/ipython_pylab_check.png)

I am currently working in Linux, and have stored the dataset in the following location:

/home/kunal/Downloads/Loan_Prediction/train.csv

Importing libraries and the data set:

Following are the libraries we will use during this tutorial:

- numpy
- matplotlib
- pandas

Please note that you do not need to import matplotlib and numpy because of Pylab environment. I have still kept them in the code, in case you use the code in a different environment.

After importing the library, you read the dataset using function read_csv(). This is how the code looks like till this stage:

```
import pandas as pd
import numpy as np
import matplotlib as plt
%matplotlib inline

df = pd.read_csv("/home/kunal/Downloads/Loan_Prediction/train.csv") #Reading the dataset in a
dataframe using Pandas
```

Quick Data Exploration

Once you have read the dataset, you can have a look at few top rows by using the function head()

```
df.head(10)
```

```
In [3]: df.head(10)  #Printing first 10 rows of dataset
```

Out[3]:

	Loan_ID	Gender	Married	Dependents	Education	Self_Employed	ApplicantIncome	CoapplicantIncome	LoanAmount	Loan_Amount_Term	Cr
0	LP001002	Male	No	0	Graduate	No	5849	0	NaN	360	1
1	LP001003	Male	Yes	1	Graduate	No	4583	1508	128	360	1
2	LP001005	Male	Yes	0	Graduate	Yes	3000	0	66	360	1
3	LP001006	Male	Yes	0	Not Graduate	No	2583	2358	120	360	1
4	LP001008	Male	No	0	Graduate	No	6000	0	141	360	1
5	LP001011	Male	Yes	2	Graduate	Yes	5417	4196	267	360	1
6	LP001013	Male	Yes	0	Not Graduate	No	2333	1516	95	360	1
7	LP001014	Male	Yes	3+	Graduate	No	3036	2504	158	360	0
8	LP001018	Male	Yes	2	Graduate	No	4006	1526	168	360	1
9	LP001020	Male	Yes	1	Graduate	No	12841	10968	349	360	1

(https://www.analyticsvidhya.com/wp-content/uploads/2016/01/1.-head.png)

This should print 10 rows. Alternately, you can also look at more rows by printing the dataset

Next, you can look at summary of numerical fields by using describe() function

```
df.describe()
```

In [4]:
```
df.describe()  #Get summary of numerical variables
```

Out[4]:

	ApplicantIncome	CoapplicantIncome	LoanAmount	Loan_Amount_Term	Credit_History
count	614.000000	614.000000	592.000000	600.00000	564.000000
mean	5403.459283	1621.245798	146.412162	342.00000	0.842199
std	6109.041673	2926.248369	85.587325	65.12041	0.364878
min	150.000000	0.000000	9.000000	12.00000	0.000000
25%	2877.500000	0.000000	100.000000	360.00000	1.000000
50%	3812.500000	1188.500000	128.000000	360.00000	1.000000
75%	5795.000000	2297.250000	168.000000	360.00000	1.000000
max	81000.000000	41667.000000	700.000000	480.00000	1.000000

(https://www.analyticsvidhya.com/wp-content/uploads/2016/01/2.-describe.png)

describe() function would provide count, mean, standard deviation (std), min, quartiles and max in its output (Read this article (https://www.analyticsvidhya.com/blog/2014/07/statistics/) to refresh basic statistics to understand population distribution)

Here are a few inferences, you can draw by looking at the output of describe() function:

1. LoanAmount has (614 – 592) 22 missing values.
2. Loan_Amount_Term has (614 – 600) 14 missing values.
3. Credit_History has (614 – 564) 50 missing values.
4. We can also look that about 84% applicants have a credit_history. How? The mean of Credit_History field is 0.84 (Remember, Credit_History has value 1 for those who have a credit history and 0 otherwise)
5. The ApplicantIncome distribution seems to be in line with expectation. Same with CoapplicantIncome

Please note that we can get an idea of a possible skew in the data by comparing the mean to the median, i.e. the 50% figure.

For the non-numerical values (e.g. Property_Area, Credit_History etc.), we can look at frequency distribution to understand whether they make sense or not. The frequency table can be printed by following command:

```
df['Property_Area'].value_counts()
```

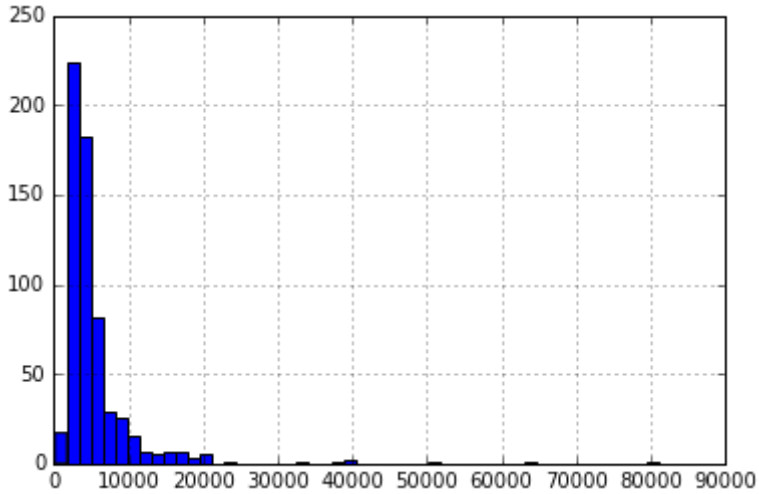
Similarly, we can look at unique values of port of credit history. Note that dfname['column_name'] is a basic indexing technique to acess a particular column of the dataframe. It can be a list of columns as well. For more information, refer to the "10 Minutes to Pandas" resource shared above.

Distribution analysis

Now that we are familiar with basic data characteristics, let us study distribution of various variables. Let us start with numeric variables – namely ApplicantIncome and LoanAmount

Lets start by plotting the histogram of ApplicantIncome using the following commands:

```
df['ApplicantIncome'].hist(bins=50)
```

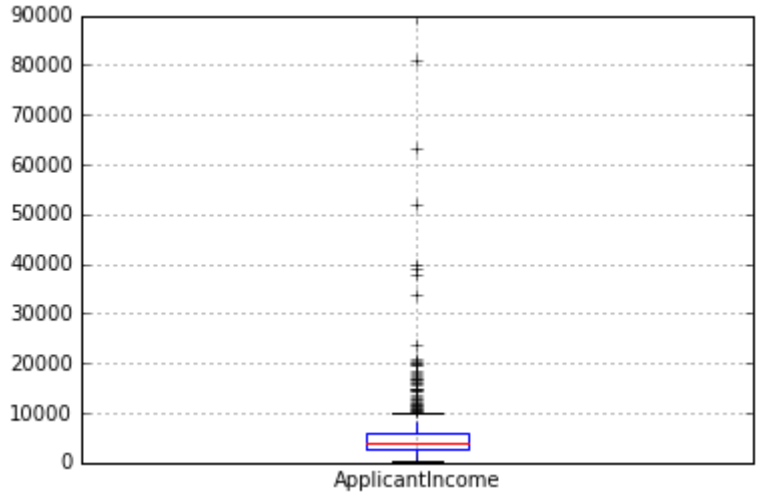


(https://www.analyticsvidhya.com/wp-content/uploads/2016/01/output_6_1.png)

Here we observe that there are few extreme values. This is also the reason why 50 bins are required to depict the distribution clearly.

Next, we look at box plots to understand the distributions. Box plot for fare can be plotted by:

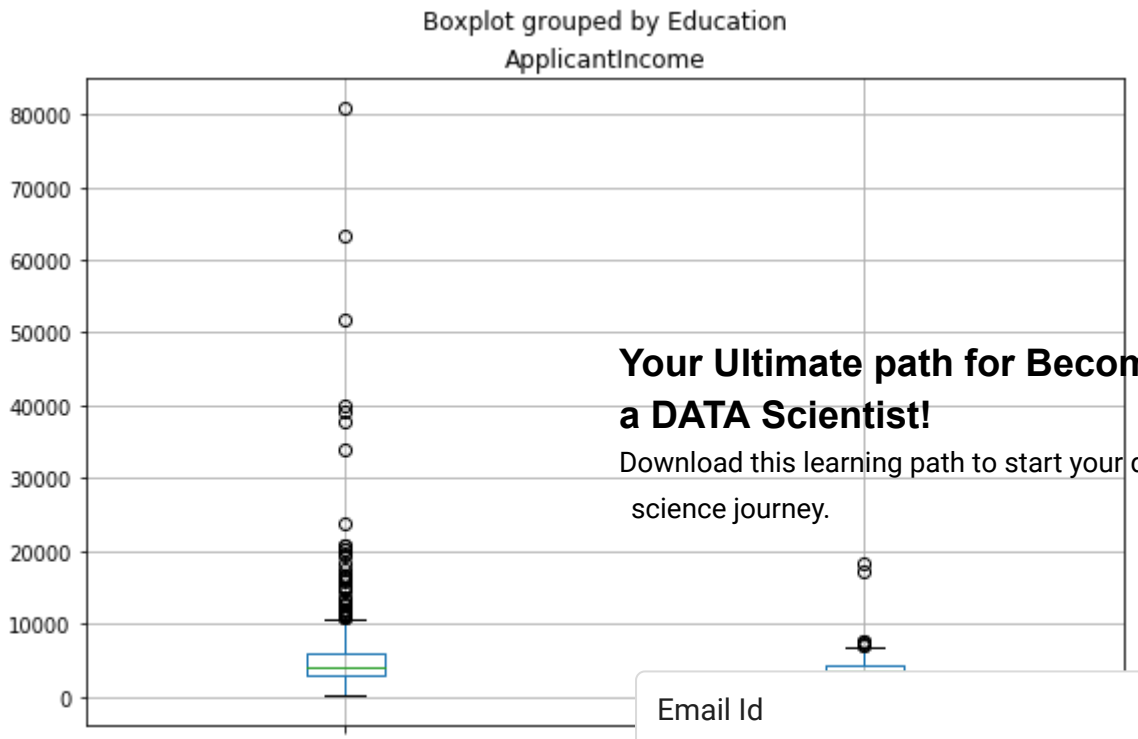
```
df.boxplot(column='ApplicantIncome')
```



(https://www.analyticsvidhya.com/wp-content/uploads/2016/01/output_7_1.png)

This confirms the presence of a lot of outliers/extreme values. This can be attributed to the income disparity in the society. Part of this can be driven by the fact that we are looking at people with different education levels. Let us segregate them by Education:

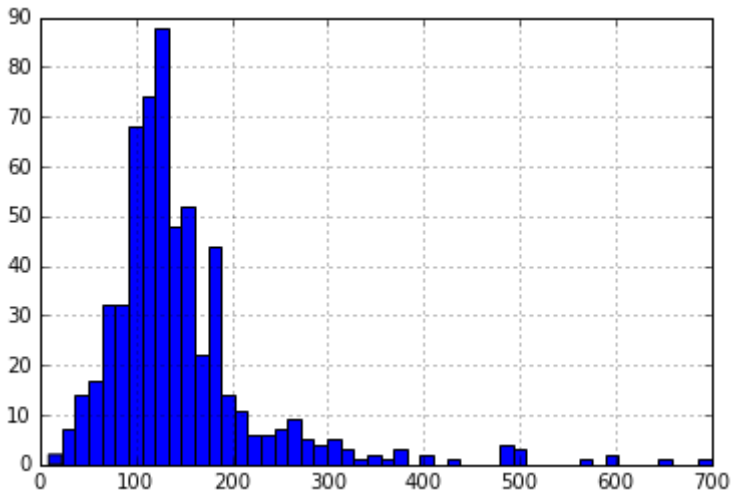
```
df.boxplot(column='ApplicantIncome', by = 'Education')
```



We can see that there is no substantial different between the mean income of graduate and non-graduates. But there are a higher number of graduates with very high incomes, which are appearing to be the outliers.

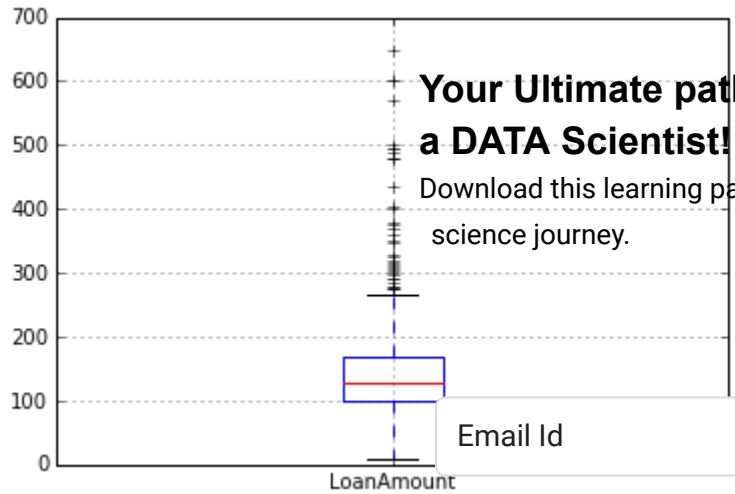
Now, Let's look at the histogram and boxplot of LoanAmount using the following command:

```
df['LoanAmount'].hist(bins=50)
```



(https://www.analyticsvidhya.com/wp-content/uploads/2016/01/output_13_1.png)

```
df.boxplot(column='LoanAmount')
```

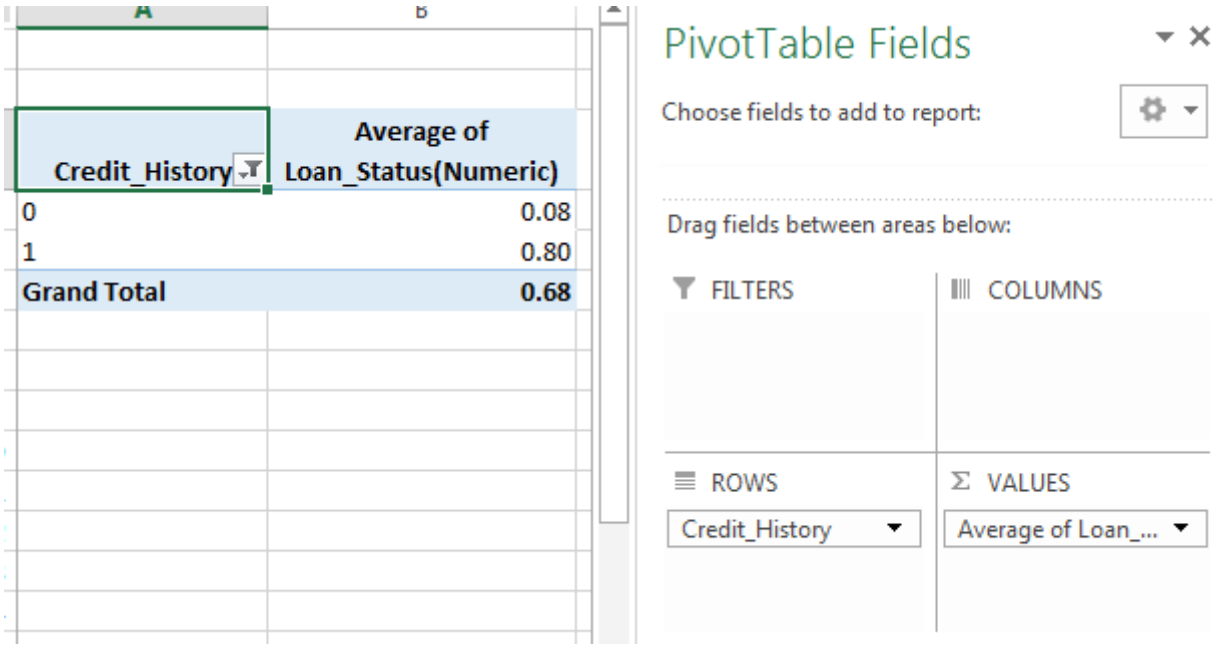


(https://www.analyticsvidhya.com/wp-content/uploads/2016/01/output_14_1.png)

Again, there are some extreme values. Clearly, both ApplicantIncome and LoanAmount require some amount of data munging. LoanAmount has missing and well as extreme values values, while ApplicantIncome has a few extreme values, which demand deeper understanding. We will take this up in coming sections.

Categorical variable analysis

Now that we understand distributions for ApplicantIncome and LoanIncome, let us understand categorical variables in more details. We will use Excel style pivot table and cross-tabulation. For instance, let us look at the chances of getting a loan based on credit history. This can be achieved in MS Excel using a pivot table as:



(https://www.analyticsvidhya.com/wp-content/uploads/2016/01/10.-pivot_table3.png)

Note: here loan status has been coded as 1 for Yes and 0 for No. So the mean represents the probability of getting loan.

Now we will look at the steps required to generate a similar insight using Python. Please refer to this article (https://www.analyticsvidhya.com/blog/2016/01/12-pandas-techniques-python-data-manipulation/) for getting a hang of the different data manipulation techniques in Pandas.

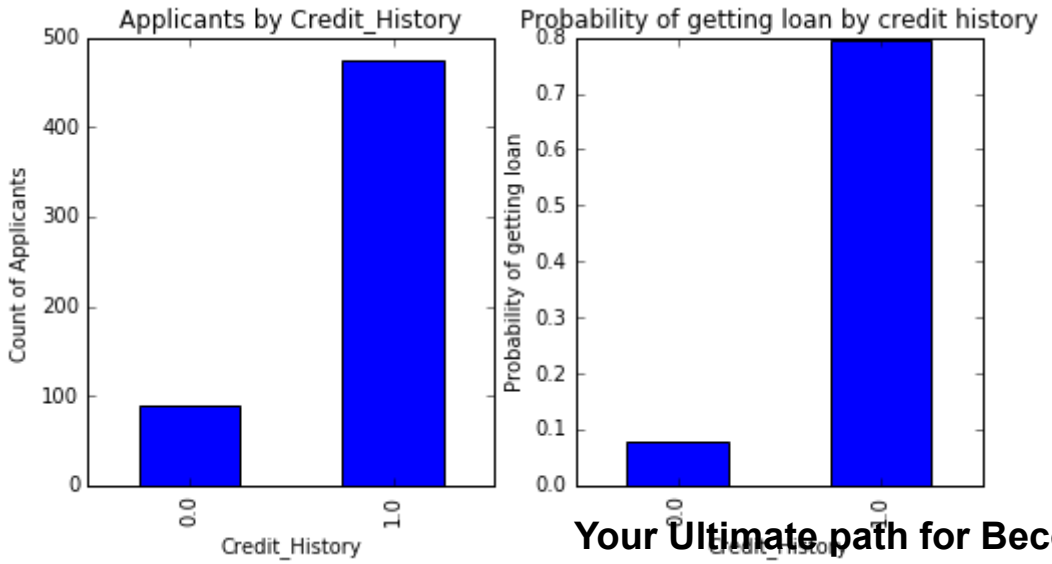
```
temp1 = df['Credit_History'].value_counts(ascending=True)
temp2 = df.pivot_table(values='Loan_Status',index=['Credit_History'],aggfunc=lambda x: x.map
({'Y':1,'N':0}).mean())
print ('Frequency Table for Credit History:')
print (temp1)

print ('\nProbility of getting loan for each Credit History class:')
print (temp2)
```

```
Frequency Table for Credit History:
0     89
1    475
Name: Credit_History, dtype: int64

Probility of getting loan for each Credit History class:
Credit_History
0    0.078652
1    0.795789
Name: Loan_Status, dtype: float64
```

(https://www.analyticsvidhya.com/wp-content/uploads/2016/01/11.-pivot_python.png)

Now we can observe that we get a similar pivot_table like the MS Excel one. This can be plotted as a bar chart using the “matplotlib” library with following code:

```
import matplotlib.pyplot as plt
fig = plt.figure(figsize=(8,4))
ax1 = fig.add_subplot(121)
ax1.set_xlabel('Credit_History')
ax1.set_ylabel('Count of Applicants')
ax1.set_title("Applicants by Credit_History")
temp1.plot(kind='bar')

ax2 = fig.add_subplot(122)
temp2.plot(kind = 'bar')
ax2.set_xlabel('Credit_History')
ax2.set_ylabel('Probability of getting loan')
ax2.set_title("Probability of getting loan by credit history")
```

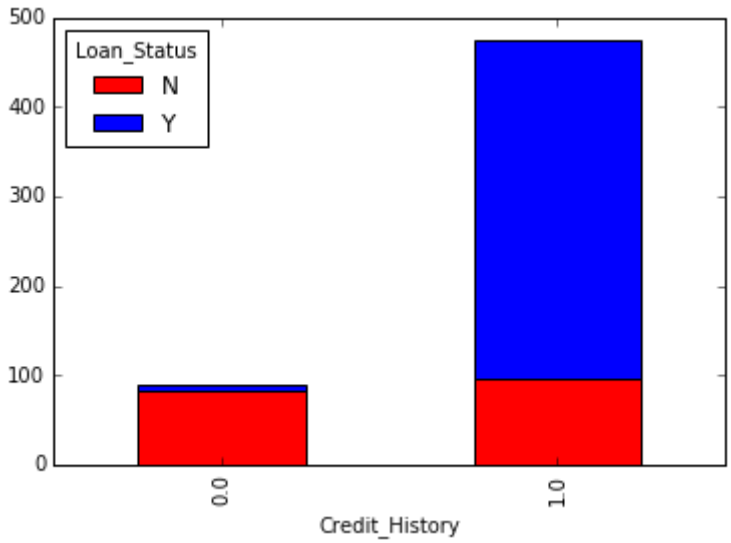


(https://www.analyticsvidhya.com/wp-content/uploads/2016/01/output_16_1.png)

This shows that the chances of getting a loan are eight-fold if the applicant has a valid credit history. You can plot similar graphs by Married, Self-Employed, Property_Area, etc.

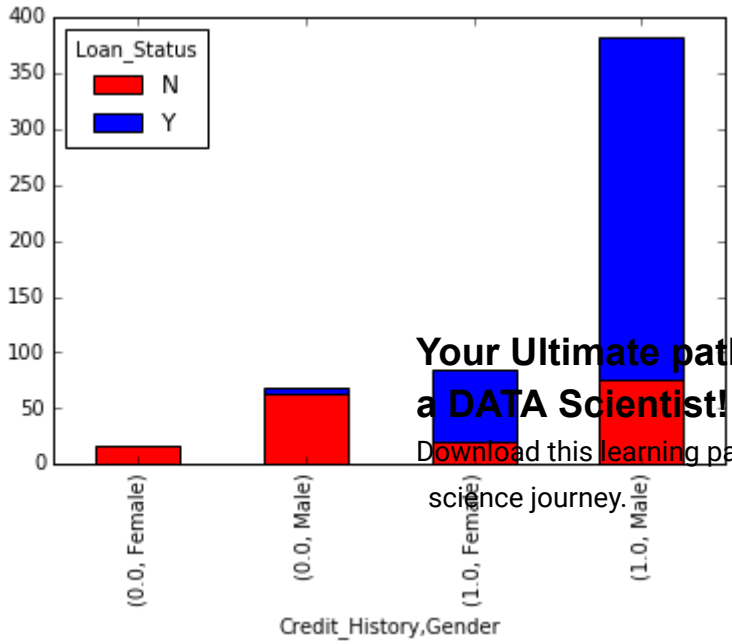
Alternately, these two plots can also be visualized by combining them in a stacked chart::

```
temp3 = pd.crosstab(df['Credit_History'], df['Loan_Status'])
temp3.plot(kind='bar', stacked=True, color=['red','blue'], grid=False)
```



(https://www.analyticsvidhya.com/wp-content/uploads/2016/01/output_17_1.png)

You can also add gender into the mix (similar to the pivot table in Excel):



(https://www.analyticsvidhya.com/wp-content/uploads/201

If you have not realized already, we have just created two basic classification algorithms here, one based on credit history, while other on 2 categorical variables (including gender). You can quickly code this to create your first submission on AV Datahacks.

We just saw how we can do exploratory analysis in Python using Pandas. I hope your love for pandas (the animal) would have increased by now – given the amount of help, the library can provide you in analyzing datasets.

Next let's explore ApplicantIncome and LoanStatus variables further, perform data munging (https://www.analyticsvidhya.com/blog/2014/09/data-munging-python-using-pandas-baby-steps-python/) and create a dataset for applying various modeling techniques. I would strongly urge that you take another dataset and problem and go through an independent example before reading further.

4. Data Munging in Python : Using Pandas

For those, who have been following, here are your must wear shoes to start running.
(https://www.analyticsvidhya.com/wp-content/uploads/2014/09/Orange-Shoes.jpg)

Data munging – recap of the need

While our exploration of the data, we found a few problems in the data set, which needs to be solved before the data is ready for a good model. This exercise is typically referred as "Data Munging". Here are the problems, we are already aware of:

1. There are missing values in some variables. We should estimate those values wisely depending on the amount of missing values and the expected importance of variables.
2. While looking at the distributions, we saw that ApplicantIncome and LoanAmount seemed to contain extreme values at either end. Though they might make intuitive sense, but should be treated appropriately.

In addition to these problems with numerical fields, we should also look at the non-numerical fields i.e. Gender, Property_Area, Married, Education and Dependents to see, if they contain any useful information.

If you are new to Pandas, I would recommend reading this article (https://www.analyticsvidhya.com/blog/2016/01/12-pandas-techniques-python-data-manipulation/) before moving on. It details some useful techniques of data manipulation.

Check missing values in the dataset

Let us look at missing values in all the variables because most models don't work with missing data and even if they do, imputing them helps more often than not. So, let us check the number of nulls / NaNs in the dataset

```
df.apply(lambda x: sum(x.isnull()),axis=0)
```

This command should tell us the number of missing values in each column as isnull() returns 1, if the value is null.

```
In [14]: df.apply(lambda x: sum(x.isnull()),axis=0)
Out[14]: Loan_ID              0
         Gender              13
         Married              3
         Dependents          15
         Education            0
         Self_Employed       32
         ApplicantIncome      0
         CoapplicantIncome    0
         LoanAmount          22
         Loan_Amount_Term    14
         Credit_History      50
         Property_Area        0
         Loan_Status          0
         dtype: int64
```

(https://www.analyticsvidhya.com/wp-content/uploads/2016/01/4.-missing.png)

Though the missing values are not very high in number, but many variables have them and each one of these should be estimated and added in the data. Get a detailed view on different imputation techniques through this article (https://www.analyticsvidhya.com/blog/2016/01/guide-data-exploration/).

Note: Remember that missing values may not always be NaNs. For instance, if the Loan_Amount_Term is 0, does it makes sense or would you consider that missing? I suppose your answer is missing and you're right. So we should check for values which are unpractical.

How to fill missing values in LoanAmount?

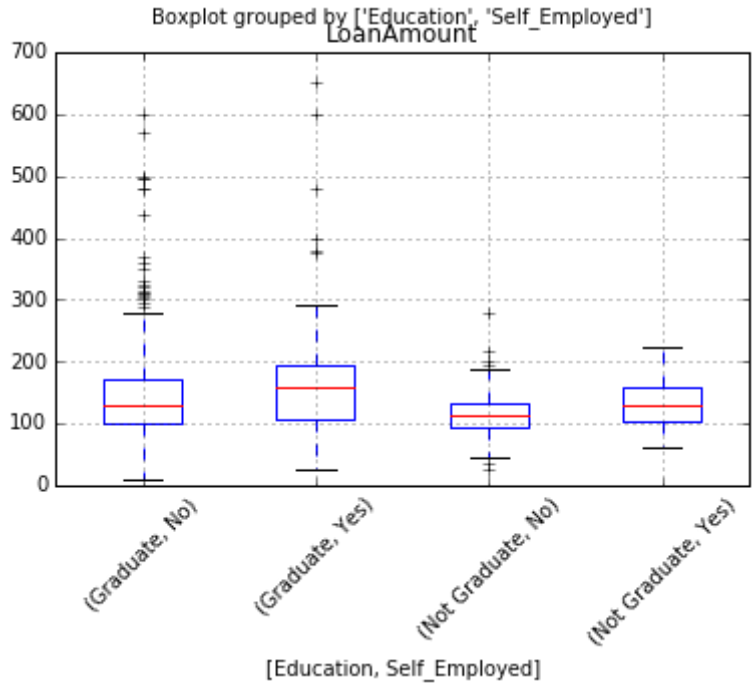
There are numerous ways to fill the missing values of loan amount – the simplest being replacement by mean, which can be done by following code:


```
df['LoanAmount'].fillna(df['LoanAmount'].mean(), inplace=True)
```

The other extreme could be to build a supervised learning model to predict loan amount on the basis of other variables and then use age along with other variables to predict survival.

Since, the purpose now is to bring out the steps in data munging, I'll rather take an approach, which lies some where in between these 2 extremes. A key hypothesis is that the whether a person is educated or self-employed can combine to give a good estimate of loan amount.

First, let's look at the boxplot to see if a trend exists:



(https://www.analyticsvidhya.com/wp-content/uploads/2016/01/5.-loan-amount-boxplot.png)

Thus we see some variations in the median of loan amount for each group and this can be used to impute the values. But first, we have to ensure that each of Self_Employed and Education variables should not have a missing values.

As we say earlier, Self_Employed has some missing values. Let's look at the frequency table:

```
In [40]: df['Self_Employed'].value_counts()

Out[40]: No     500
         Yes     82
         Name: Self_Employed, dtype: int64
```


(https://www.analyticsvidhya.com/wp-content/uploads/201

Since ~86% values are "No", it is safe to impute the missing values as "No" as there is a high probability of success. This can be done using the following code:

```
df['Self_Employed'].fillna('No',inplace=True)
```

Now, we will create a Pivot table, which provides us median values for all the groups of unique values of Self_Employed and Education features. Next, we define a function, which returns the values of these cells and apply it to fill the missing values of loan amount:

```
table = df.pivot_table(values='LoanAmount', index='Self_Employed' ,columns='Education', aggfu
nc=np.median)
# Define function to return value of this pivot_table
def fage(x):
 return table.loc[x['Self_Employed'],x['Education']]
# Replace missing values
df['LoanAmount'].fillna(df[df['LoanAmount'].isnull()].apply(fage, axis=1), inplace=True)
```

This should provide you a good way to impute missing values of loan amount.

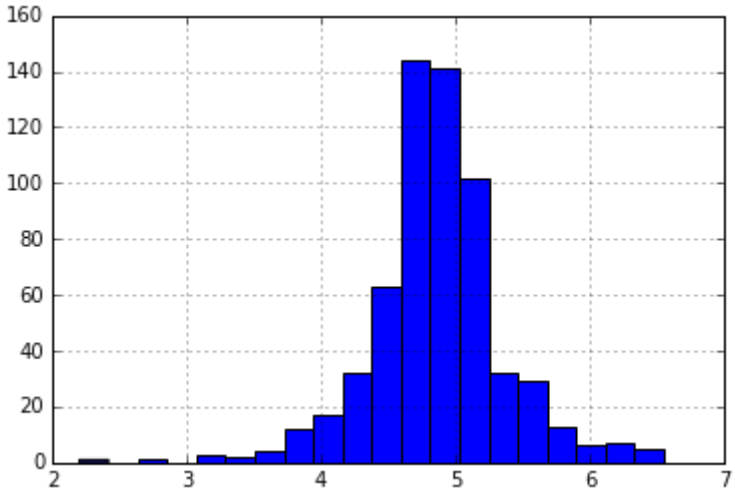
NOTE : This method will work only if you have not filled the missing values in Loan_Amount variable using the previous approach, i.e. using mean.

How to treat for extreme values in distribution of LoanAmount and ApplicantIncome?

Let's analyze LoanAmount first. Since the extreme values are practically possible, i.e. some people might apply for high value loans due to specific needs. So instead of treating them as outliers, let's try a log transformation to nullify their effect:


```
df['LoanAmount_log'] = np.log(df['LoanAmount'])
df['LoanAmount_log'].hist(bins=20)
```

Looking at the histogram again:

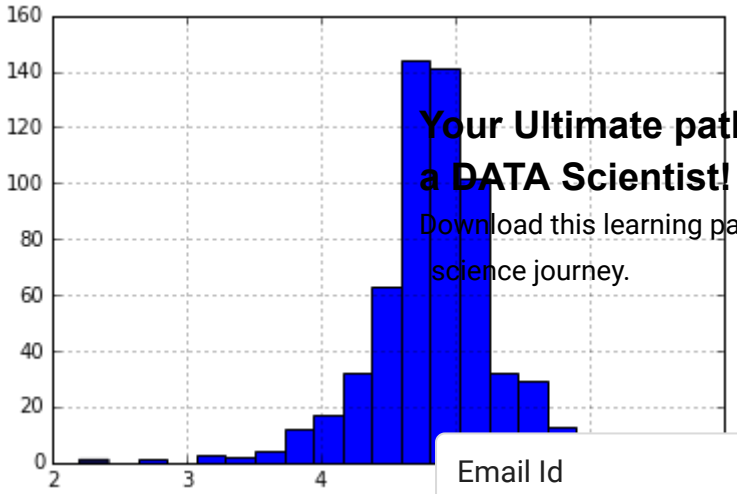


(https://www.analyticsvidhya.com/wp-content/uploads/2016/01/7.-loan-log.png)

Now the distribution looks much closer to normal and effect of extreme values has been significantly subsided.

Coming to ApplicantIncome. One intuition can be that some applicants have lower income but strong support Co-applicants. So it might be a good idea to combine both incomes as total income and take a log transformation of the same.

```
df['TotalIncome'] = df['ApplicantIncome'] + df['CoapplicantIncome']
df['TotalIncome_log'] = np.log(df['TotalIncome'])
df['LoanAmount_log'].hist(bins=20)
```



(https://www.analyticsvidhya.com/wp-content/uploads/2016/01/8.-total-income-log.png)

Now we see that the distribution is much better than before. I will leave it upto you to impute the missing values for Gender, Married, Dependents, Loan_Amount_Term, Credit_History. Also, I encourage you to think about possible additional information which can be derived from the data. For example, creating a column for LoanAmount/TotalIncome might make sense as it gives an idea of how well the applicant is suited to pay back his loan.

Next, we will look at making predictive models.

5. Building a Predictive Model in Python

After, we have made the data useful for modeling, let's now look at the python code to create a predictive model on our data set. Skicit-Learn (sklearn) is the most commonly used library in Python for this purpose and we will follow the trail. I encourage you to get a refresher on sklearn through this article (https://www.analyticsvidhya.com/blog/2015/01/scikit-learn-python-machine-learning-tool/).

Since, sklearn requires all inputs to be numeric, we should convert all our categorical variables into numeric by encoding the categories. Before that we will fill all the missing values in the dataset. This can be done using the following code:

```
df['Gender'].fillna(df['Gender'].mode()[0], inplace=True)
df['Married'].fillna(df['Married'].mode()[0], inplace=True)
df['Dependents'].fillna(df['Dependents'].mode()[0], inplace=True)
df['Loan_Amount_Term'].fillna(df['Loan_Amount_Term'].mode()[0], inplace=True)
df['Credit_History'].fillna(df['Credit_History'].mode()[0], inplace=True)
```

```
from sklearn.preprocessing import LabelEncoder
var_mod = ['Gender','Married','Dependents','Education', 'Self_Employed', 'Property_Area', 'Loan_
Status']
le = LabelEncoder()
for i in var_mod:
    df[i] = le.fit_transform(df[i])
df.dtypes
```


Next, we will import the required modules. Then we will define a generic classification function, which takes a model as input and determines the Accuracy and Cross-Validation scores. Since this is an introductory article, I will not go into the details of coding. Please refer to this article

(https://www.analyticsvidhya.com/blog/2015/08/common-machine-learning-algorithms/) for getting details of the algorithms with R and Python codes. Also, it'll be good to get a refresher on cross-validation through this article (https://www.analyticsvidhya.com/blog/2015/11/improve-model-performance-cross-validation-in-python-r/), as it is a very important measure of power performance.

```
#Import models from scikit learn module:
from sklearn.linear_model import LogisticRegression
from sklearn.cross_validation import KFold   #For K-fold cross validation
from sklearn.ensemble import RandomForestClassifier
from sklearn.tree import DecisionTreeClassifier, export_graphviz
from sklearn import metrics

#Generic function for making a classification model and accessing performance:
def classification_model(model, data, predictors, outcome):
  #Fit the model:
  model.fit(data[predictors],data[outcome])

  #Make predictions on training set:
  predictions = model.predict(data[predictors])

  #Print accuracy
  accuracy = metrics.accuracy_score(predictions,data[outcome])
  print ("Accuracy : %s" % "{0:.3%}".format(accuracy))

  #Perform k-fold cross-validation with 5 folds
  kf = KFold(data.shape[0], n_folds=5)
  error = []
  for train, test in kf:
    # Filter training data
    train_predictors = (data[predictors].iloc[train,:])

    # The target we're using to train the algorithm.
    train_target = data[outcome].iloc[train]

    # Training the algorithm using the predictors and target.
    model.fit(train_predictors, train_target)

    #Record error from each cross-validation run
    error.append(model.score(data[predictors].iloc[test,:], data[outcome].iloc[test]))
```

```
    print ("Cross-Validation Score : %s" % "{0:.3%}".format(np.mean(error)))

    #Fit the model again so that it can be refered outside the function:
    model.fit(data[predictors],data[outcome])
```

Logistic Regression

Let's make our first Logistic Regression model. One way would be to take all the variables into the model but this might result in overfitting (don't worry if you're unaware of this terminology yet). In simple words, taking all variables might result in the model understanding complex relations specific to the data and will not generalize well. Read more about Logistic Regression (https://www.analyticsvidhya.com/blog/2015/11/beginners-guide-on-logistic-regression-in-r/).

We can easily make some intuitive hypothesis to set the ball rolling. The chances of getting a loan will be higher for:

1. Applicants having a credit history (remember we observed this in exploration?)
2. Applicants with higher applicant and co-applicant incomes
3. Applicants with higher education level
4. Properties in urban areas with high growth perspectives

So let's make our first model with 'Credit_History'.

```
outcome_var = 'Loan_Status'
model = LogisticRegression()
predictor_var = ['Credit_History']
classification_model(model, df,predictor_var,outcome_var)
```


Accuracy : 80.945% Cross-Validation Score : 80.946%

```
#We can try different combination of variables:
predictor_var = ['Credit_History','Education','Married','Self_Employed','Property_Area']
classification_model(model, df,predictor_var,outcome_var)
```

Accuracy : 80.945% Cross-Validation Score : 80.946%

Generally we expect the accuracy to increase on adding variables. But this is a more challenging case. The accuracy and cross-validation score are not getting impacted by less important variables. Credit_History is dominating the mode. We have two options now:

1. Feature Engineering: dereive new information and try to predict those. I will leave this to your creativity.
2. Better modeling techniques. Let's explore this next.

Decision Tree

Decision tree is another method for making a predictive model. It is known to provide higher accuracy than logistic regression model. Read more about Decision Trees (https://www.analyticsvidhya.com/blog/2015/01/decision-tree-simplified/).

```
model = DecisionTreeClassifier()
predictor_var = ['Credit_History','Gender','Married','Education']
classification_model(model, df,predictor_var,outcome_var)
```

Accuracy : 81.930% Cross-Validation Score : 76.656%

Here the model based on categorical variables is unable to have an impact because Credit History is dominating over them. Let's try a few numerical variables:

```
#We can try different combination of variables:
predictor_var = ['Credit_History','Loan_Amount_Term','LoanAmount_log']
classification_model(model, df,predictor_var,outcome_var)
```


Accuracy : 92.345% Cross-Validation Score : 71.009%

Here we observed that although the accuracy went up on adding variable, the cross-validation error went down. This is the result of model over-fitting the data. Let's try an algorithm and see if it helps:

Random Forest

Random forest is another algorithm for solving the classification problem. Read more about Random Forest (https://www.analyticsvidhya.com/blog/2015/09/random-forest-algorithm-multiple-challenges/).

An advantage with Random Forest is that we can make it work with all the features and it returns a feature importance matrix which can be used to select features.

```
model = RandomForestClassifier(n_estimators=100)
predictor_var = ['Gender', 'Married', 'Dependents', 'Education',
       'Self_Employed', 'Loan_Amount_Term', 'Credit_History', 'Property_Area',
        'LoanAmount_log','TotalIncome_log']
classification_model(model, df,predictor_var,outcome_var)
```

Accuracy : 100.000% Cross-Validation Score : 78.179%

Here we see that the accuracy is 100% for the training set. This is the ultimate case of overfitting and can be resolved in two ways:

1. Reducing the number of predictors
2. Tuning the model parameters

Let's try both of these. First we see the feature importance matrix from which we'll take the most important features.

```
#Create a series with feature importances:
featimp = pd.Series(model.feature_importances_, index=predictor_var).sort_values(ascending=Fa
lse)
print (featimp)
```

```
Credit_History      0.273094
TotalIncome_log     0.264433
LoanAmount_log      0.229032
Dependents          0.050138
Property_Area       0.048979
Loan_Amount_Term    0.042681
Married             0.025823
Education           0.022426
Gender              0.021895
Self_Employed       0.021500
dtype: float64
```

(https://www.analyticsvidhya.com/wp-content/uploads/2016/01/9.-rf-feat-imp.png)

Let's use the top 5 variables for creating a model. Also, we will modify the parameters of random forest model a little bit:

```
model = RandomForestClassifier(n_estimators=25, min_samples_split=25, max_depth=7, max_featur
es=1)
predictor_var = ['TotalIncome_log','LoanAmount_log','Credit_History','Dependents','Property_A
rea']
classification_model(model, df,predictor_var,outcome_var)
```

Accuracy : 82.899% Cross-Validation Score : 81.461%

Notice that although accuracy reduced, but the cross-validation score is improving showing that the model is generalizing well. Remember that random forest models are not exactly repeatable. Different runs will result in slight variations because of randomization. But the output should stay in the ballpark.

You would have noticed that even after some basic parameter tuning on random forest, we have reached a cross-validation accuracy only slightly better than the original logistic regression model. This exercise gives us some very interesting and unique learning:

1. Using a more sophisticated model does not guarantee better results.
2. Avoid using complex modeling techniques as a black box without understanding the underlying concepts. Doing so would increase the tendency of overfitting thus making your models less interpretable
3. Feature Engineering (https://www.analyticsvidhya.com/...riable-transformation-creation/) is the key to success. Every... real art and creativity lies in enhancing your features to better suit the model.

You can access the dataset and problem statement used in this post at this link: Loan Prediction Challenge (https://datahack.analyticsvidhya.com/contest/practice-problem-loan-prediction-iii/?utm_source=complete-tutorial-learn-data-science-python-scratch-2&utm_medium=blog)

Projects

Now, its time to take the plunge and actually play with some other real datasets. So are you ready to take on the challenge? Accelerate your data science journey with the following Practice Problems:

 (https://datahack.analyticsvidhya.com/contest/genpact-machine-learning-hackathon-1/?utm_source=complete-tutorial-learn-data-science-python-scratch-2&utm_medium=blog)	Practice Problem: Food Demand Forecasting Chall (https://datahack.analyticsvidhya.com/contest/ge machine-learning-hackathon-1/?utm_source=comp tutorial-learn-data-science-python-scratch-2&utm_medium=blog)
 (https://datahack.analyticsvidhya.com/contest/wns-analytics-hackathon-2018-1/?utm_source=complete-tutorial-learn-data-science-python-scratch-2&utm_medium=blog)	Practice Problem: HR Analytics Challenge (https://datahack.analyticsvidhya.com/contest/wn analytics-hackathon-2018-1/?utm_source=comple tutorial-learn-data-science-python-scratch-2&utm_medium=blog)

 (https://datahack.analyticsvidhya.com/contest/enigma-codefest-machine-learning-1/?utm_source=complete-tutorial-learn-data-science-python-scratch-2&utm_medium=blog)	Practice Problem: Predict Number of Upvotes (https://datahack.analyticsvidhya.com/contest/encodefest-machine-learning-1/?utm_source=comple tutorial-learn-data-science-python-scratch-2&utm_medium=blog)

End Notes

I hope this tutorial will help you maximize your efficiency when starting with data science in Python. I am sure this not only gave you an idea about basic data analysis methods but it also showed you how to implement some of the more sophisticated techniques available today.

You should also check out our free Python course (https://courses.analyticsvidhya.com/courses/introduction-to-data-science?utm_source=blog&utm_medium=learnDSwithPythonScratcharticle) and then jump over to learn how to apply it for Data Science (https://courses.analyticsvidhya.com/courses/introduction-to-data-science-2?utm_source=blog&utm_medium=learnDSwithPythonScratcharticle).

Python is really a great tool and is becoming an increasingly popular language among the data scientists. The reason being, it's easy to learn, integrates well with other databases and tools like Spark and Hadoop. Majorly, it has the great computational intensity and has powerful data analytics libraries.

So, learn Python to perform the full life-cycle of any data science project. It includes reading, analyzing, visualizing and finally making predictions.

If you come across any difficulty while practicing Python, or you have any thoughts /suggestions/feedback on the post, please feel free to post them through comments below.

Note – The discussions of this article are going on at AV's Discuss portal. Join here (https://discuss.analyticsvidhya.com/t/discussions-for-article-a-complete-tutorial-to-learn-data-science-with-python-from-scratch/65186?u=jalfaizy)!

If you like what you just read & want to continue your analytics learning, subscribe to our emails (http://feedburner.google.com/fb/a/mailverify?uri=analyticsvidhya), follow us on twitter (http://twitter.com/analyticsvidhya) or like our facebook page (http://facebook.com/analyticsvidhya).

You can also read this article on Analytics Vidhya's Android APP GET IT ON Google Play

(//play.google.com/store/apps/details?id=com.analyticsvidhya.android&utm_source=blog_article&utm_campaign=blog&pcampaignid=MKT-Other-global-all-co-prtnr-py-PartBadge-Mar2515-1)

Share this:

(https://www.analyticsvidhya.com/blog/2016/01/complete-tutorial-learn-data-science-python-scratch-2/?share=linkedin&nb=1)

(https://www.analyticsvidhya.com/blog/2016/01/complete-tutorial-learn-data-science-python-scratch-2/?share=facebook&nb=1)

(https://www.analyticsvidhya.com/blog/2016/01/complete-tutorial-learn-data-science-python-scratch-2/?share=twitter&nb=1)

(https://www.analyticsvidhya.com/blog/2016/01/complete-tutorial-learn-data-science-python-scratch-2/?share=pocket&nb=1)

(https://www.analyticsvidhya.com/blog/2016/01/complete-tutorial-learn-data-science-python-scratch-2/?share=reddit&nb=1)

Related Articles

(https://www.analyticsvidhya.com/blog/2016/09/most-active-data-scientists-free-books-notebooks-tutorials-on-github/)
Most Active Data Scientists, Free Books, Notebooks & Tutorials on Github (https://www.analyticsvidhya.com/blog/2016/09/most-active-data-scientists-free-books-notebooks-tutorials-on-github/)
September 30, 2016
In "Machine Learning"

(https://www.analyticsvidhya.com/blog/2015/07/github-special-data-scientists-to-follow-best-tutorials/)
Top Data Scientists to Follow & Best Data Science Tutorials on GitHub (https://www.analyticsvidhya.com/blog/2015/07/github-special-data-scientists-to-follow-best-tutorials/)
July 21, 2015
In "Business Analytics"

(https://www.analyticsvidhya.com/blog/2015/07/certification-courses-sas-r-python-machine-learning-big-data-spark-2015-16/)
Top Certification Courses in SAS, R, Python, Machine Learning, Big Data, Spark (https://www.analyticsvidhya.com/blog/2016/01/top-certification-courses-sas-r-python-machine-learning-big-data-spark-2015-16/)

TAGS : APPLY() (HTTPS://WWW.ANALYTICSVIDHYA.COM/BLOG/TAG/APPLY/), BOX PLOTS (HTTPS://WWW.ANALYTICSVIDHYA.COM/BLOG/TAG/BOX-PLOTS/), CATEGORICAL VARIABLE (HTTPS://WWW.ANALYTICSVIDHYA.COM/BLOG/TAG/CATEGORICAL-VARIABLE/), DATA CLEANING (HTTPS://WWW.ANALYTICSVIDHYA.COM/BLOG/TAG/DATA-CLEANING/), DATA EXPLORATION (HTTPS://WWW.ANALYTICSVIDHYA.COM/BLOG/TAG/DATA-EXPLORATION/), DATA FRAMES (HTTPS://WWW.ANALYTICSVIDHYA.COM/BLOG/TAG/DATA-FRAMES/), DATA MINING (HTTPS://WWW.ANALYTICSVIDHYA.COM/BLOG/TAG/DATA-MINING/), DATA MUNGING (HTTPS://WWW.ANALYTICSVIDHYA.COM/BLOG/TAG/DATA-MUNGING/), DATA SCIENCE (HTTPS://WWW.ANALYTICSVIDHYA.COM/BLOG/TAG/DATA-SCIENCE/), DATA STRUCTURE (HTTPS://WWW.ANALYTICSVIDHYA.COM/BLOG/TAG/DATA-STRUCTURE/), DATA WRANGLING (HTTPS://WWW.ANALYTICSVIDHYA.COM/BLOG/TAG/DATA-WRANGLING/), DECISION TREE (HTTPS://WWW.ANALYTICSVIDHYA.COM/BLOG/TAG/DECISION-TREE/), DICTIONARY (HTTPS://WWW.ANALYTICSVIDHYA.COM/BLOG/TAG/DICTIONARY/), DISTRIBUTION ANALYSIS (HTTPS://WWW.ANALYTICSVIDHYA.COM/BLOG/TAG/DISTRIBUTION-ANALYSIS/), INSTALLING PYTHON (HTTPS://WWW.ANALYTICSVIDHYA.COM/BLOG/TAG/INSTALLING-PYTHON/), IPYTHON (HTTPS://WWW.ANALYTICSVIDHYA.COM/BLOG/TAG/IPYTHON/), LISTS (HTTPS://WWW.ANALYTICSVIDHYA.COM/BLOG/TAG/LISTS/), LOGISTIC REGRESSION (HTTPS://WWW.ANALYTICSVIDHYA.COM/BLOG/TAG/LOGISTIC-REGRESSION/), MATPLOTLIB (HTTPS://WWW.ANALYTICSVIDHYA.COM/BLOG/TAG/MATPLOTLIB/), MEAN (HTTPS://WWW.ANALYTICSVIDHYA.COM/BLOG/TAG/MEAN/), MERGE (HTTPS://WWW.ANALYTICSVIDHYA.COM/BLOG/TAG/MERGE/), NUMPY (HTTPS://WWW.ANALYTICSVIDHYA.COM/BLOG/TAG/NUMPY/), PANDAS (HTTPS://WWW.ANALYTICSVIDHYA.COM/BLOG/TAG/PANDAS/), PYTHON (HTTPS://WWW.ANALYTICSVIDHYA.COM/BLOG/TAG/PYTHON/), PYTHON 3.7 (HTTPS://WWW.ANALYTICSVIDHYA.COM/BLOG/TAG/PYTHON-3-7/), PYTHON DOWNLOAD (HTTPS://WWW.ANALYTICSVIDHYA.COM/BLOG/TAG/PYTHON-DOWNLOAD/), PYTHON TUTORIAL (HTTPS://WWW.ANALYTICSVIDHYA.COM/BLOG/TAG/PYTHON-TUTORIAL/), SCIKIT-LEARN (HTTPS://WWW.ANALYTICSVIDHYA.COM/BLOG/TAG/SCIKIT-LEARN/), SCIPY (HTTPS://WWW.ANALYTICSVIDHYA.COM/BLOG/TAG/SCIPY/), SETS (HTTPS://WWW.ANALYTICSVIDHYA.COM/BLOG/TAG/SETS/), SPLIT (HTTPS://WWW.ANALYTICSVIDHYA.COM/BLOG/TAG/SPLIT/), STACKED CHART (HTTPS://WWW.ANALYTICSVIDHYA.COM/BLOG/TAG/STACKED-CHART/), STARTING PYTHON (HTTPS://WWW.ANALYTICSVIDHYA.COM/BLOG/TAG/STARTING-PYTHON/), STATISTICS (HTTPS://WWW.ANALYTICSVIDHYA.COM/BLOG/TAG/STATISTICS/), STRINGS (HTTPS://WWW.ANALYTICSVIDHYA.COM/BLOG/TAG/STRINGS/), TUPLE (HTTPS://WWW.ANALYTICSVIDHYA.COM/BLOG/TAG/TUPLE/), TUPLES (HTTPS://WWW.ANALYTICSVIDHYA.COM/BLOG/TAG/TUPLES/), TUTORIAL (HTTPS://WWW.ANALYTICSVIDHYA.COM/BLOG/TAG/TUTORIAL/)

NEXT ARTICLE

[Infographic] 10 Popular TV Shows on Data Science and Artificial Intelligence

(https://www.analyticsvidhya.com/blog/2016/01/10-popular-tv-shows-data-science-artificial-intelligence/)

•••

PREVIOUS ARTICLE

Model Monitoring Senior Business Analyst/Assistant Manager – Gurgaon (5-6 years of experience)

(https://www.analyticsvidhya.com/blog/2016/01/model-monitoring-senior-business-analystassistant-manager-gurgaon-5-6-years-experience/)

(https://www.analyticsvidhya.com/blog/author/kunalj/)

Kunal Jain (Https://Www.Analyticsvidhya.Com/Blog/Author/Kunalj/)

Kunal is a post graduate from IIT Bombay in Aerospace Engineering. He has spent more than 10 years in field of Data Science. His work experience ranges from mature markets like UK to a developing market like India. During this period he has lead teams of various sizes and has worked on various tools like SAS, SPSS, Qlikview, R, Python and Matlab.

This article is quite old and you might not get a prompt response from the author. We request you to post this comment on Analytics Vidhya's Discussion portal (https://discuss.analyticsvidhya.com/) to get your queries resolved

54 COMMENTS

MOUMITA MITRA

January 15, 2016 at 4:41 am (https://www.analyticsvidhya.com/blog/2016/01/complete-tutorial-learn-data-science-python-scratch-2/#comment-103788)

can you please suggest me good data analysis book on python

PARITOSH GUPTA

January 15, 2016 at 6:15 am (https://www.analyticsvidhya.com/blog/2016/01/complete-tutorial-learn-data-science-python-scratch-2/#comment-103796)

There is a very good book on Python for Data Analysis, O Reily — Python for Data Analysis

KUNAL JAIN (HTTP://WWW.ANALYTICSVIDHYA.COM)

January 18, 2016 at 7:06 pm (https://www.analyticsvidhya.com/blog/2016/01/complete-tutorial-learn-data-science-python-scratch-2/#comment-104034)

Moumita,

The book mentioned by Paritosh is a good place to start. You can also refer some of the books mentioned here:

http://www.analyticsvidhya.com/blog/2014/06/books-data-scientists-or-aspiring-ones/ (http://www.analyticsvidhya.com/blog/2014/06/books-data-scientists-or-aspiring-ones/)

Hope this helps.

Kunal

DEEPAK

January 31, 2016 at 5:49 pm (https://www.analyticsvidhya.com/blog/2016/01/complete-tutorial-learn-data-science-python-scratch-2/#comment-104988)

Hey Kunal

Im trying to follow your lesson however I am stuck at reading the CSV file. Im using Ipython and trying to read it. I am following the syntax that you have provided but it still doesnt work.
Can you please help me if its possible I would really appreciate it

Thanks

Deepak

PRANESH

January 15, 2016 at 5:00 am (https://www.analyticsvidhya.com/blog/2016/01/complete-tutorial-learn-data-science-python-scratch-2/#comment-103789)

Hi Kunal,

When you are planning to schedule next data science meetup in Bangalore. I have missed the previous session due to conflict

---

KUNAL JAIN (HTTP://WWW.ANALYTICSVIDHYA.COM)

January 18, 2016 at 7:08 pm (https://www.analyticsvidhya.com/blog/2016/01/complete-tutorial-learn-data-science-python-scratch-2/#comment-104036)

Pranesh,

We will have a meetup some time in early March. We will announce the dates on DataHack platform and our meetup group page.

Hope to see you around this time.

Regards,
Kunal

GIANFRANCO

January 15, 2016 at 4:16 pm (https://www.analyticsvidhya.com/blog/2016/01/complete-tutorial-learn-data-science-python-scratch-2/#comment-103830)

Little error just matter for newbe as I'm:

import matplotlib.pyplot as plt
fig = plt.pyplot.figure(figsize=(8,4)) Error

import matplotlib.pyplot as plt
fig = plt.figure(figsize=(8,4)) Right

KUNAL JAIN (HTTP://WWW.ANALYTICSVIDHYA.COM)

January 18, 2016 at 7:12 pm (https://www.analyticsvidhya.com/blog/2016/01/complete-tutorial-learn-data-science-python-scratch-2/#comment-104037)

Thanks Gianfranco for highlighting it. Have corrected the same.

Regards,
Kunal

DHEERAJ PATTA

January 15, 2016 at 5:19 pm (https://www.analyticsvidhya.com/blog/2016/01/complete-tutorial-learn-data-science-python-scratch-2/#comment-103834)

Thank you so much Kunal, this is indeed a great start for any Python beginner.
Really appreciate your team's effort in bringing Data Science to a wider audience.

I strongly suggest "A Byte of Python" by Swaroop CH. It may be bit old now but helped me in getting a good start in Python.

HIGHSPIRITS

January 15, 2016 at 5:31 pm (https://www.analyticsvidhya.com/blog/2016/01/complete-tutorial-learn-data-science-python-scratch-2/#comment-103835)

Awesome!!! This is one area where I was looking for help and AV has provided it!!! Thanks a lot for the quick guide Kunal…very much helpful…

KUNAL JAIN (HTTP://WWW.ANALYTICSVIDHYA.COM)

January 18, 2016 at 7:13 pm (https://www.analyticsvidhya.com/blog/2016/01/complete-tutorial-learn-data-science-python-scratch-2/#comment-104038)

Glad that you liked it HighSpirits!

KAMI888

January 16, 2016 at 3:33 am (https://www.analyticsvidhya.com/blog/2016/01/complete-tutorial-learn-data-science-python-scratch-2/#comment-103865)

Great!!!!!! Thank you!. I was just looking around for this.

KUNAL JAIN (HTTP://WWW.ANALYTICSVIDHYA.COM)

January 18, 2016 at 7:14 pm (https://www.analyticsvidhya.com/blog/2016/01/complete-tutorial-learn-data-science-python-scratch-2/#comment-104039)

Thanks Kami888 for your comment.

Do let us know how you progress with this.

Regards,
Kunal

DR.D.K.SAMUEL

January 16, 2016 at 3:59 am (https://www.analyticsvidhya.com/blog/2016/01/complete-tutorial-learn-data-science-python-scratch-2/#comment-103868)

Really well written, will be nice if it is made available as a pdf for download (with all supporting references). I will print and refer till I learn in full. Thanks

---

SMRUTIRANJAN TRIPATHY

January 17, 2016 at 10:04 am (https://www.analyticsvidhya.com/blog/2016/01/complete-tutorial-learn-data-science-python-scratch-2/#comment-103955)

Hi kunal ji ,

Can you please guide (for a newbie)who dont have any software background , how can acquire big data knowledge. whether is it necessary to learn SQL , JAVA ?Before stepping in the big data practically, how can i warm up my self without getting in touch with the bias. Can you please suggest good blog regarding big data for newbie.

KUNAL JAIN (HTTP://WWW.ANALYTICSVIDHYA.COM)

January 18, 2016 at 7:15 pm (https://www.analyticsvidhya.com/blog/2016/01/complete-tutorial-learn-data-science-python-scratch-2/#comment-104040)

Smrutiranjan,
:
Kindly post this question on our discussion portal – http://discuss.analyticsvidhya.com (http://discuss.analyticsvidhya.com)

This is not relevant to the article above.

Regards,
Kunal

FALKOR

January 18, 2016 at 9:41 pm (https://www.analyticsvidhya.com/blog/2016/01/complete-tutorial-learn-data-science-python-scratch-2/#comment-104049)

This was good until, the fact hit me. I am using IDLE and don't have the libraries installed. Now, how do I get these Pandas, Numpy etc installed for IDLE on Windows!?

Its been a long complicated browsing session. Only solution I seem to get is to ditch IDLE and move to Spyder or move to Python 3.5 altogether.

Any solutions will be helpful, thank you.

DIGVIJAY

March 7, 2016 at 3:02 pm (https://www.analyticsvidhya.com/blog/2016/01/complete-tutorial-learn-data-science-python-scratch-2/#comment-106809)

I suggest installing anaconda. Its better to start with as it contains most of the commonly used libraries for data analysis. Once anaconda is up and working, you can use any IDE of your choice.

FALKOR

January 19, 2016 at 1:26 am (https://www.analyticsvidhya.com/blog/2016/01/complete-tutorial-learn-data-science-python-scratch-2/#comment-104056)

Got Pandas to finally install and work, here is how I did it. Just in case it helps somebody else.

Download pip-installer from here: https://bootstrap.pypa.io/get-pip.py (https://bootstrap.pypa.io/get-pip.py)

Put it on to desktop or some known path

Open Command prompt and point to the path or open the path

Execute the file in command prompt with: python get-pip.py

Check if you got it right using: python -m pip install -U pip

This will ensure that you are on the current version

Restart the system, just for the heck of it. To be on safer side.

In Command prompt, set the path using this: C:\users\yourname>set PATH = %PATH%;C:\python27\scripts

Still in command prompt, install a library like: pip install num

Should work (maybe)

*

I had a C++ compiler error, installing this resolved it: https://www.microsoft.com/en-us/download/details.aspx?id=44266 (https://www.microsoft.com/en-us/download/details.aspx?id=44266)

Try installing libraries again
*

**Sources from all over the place!

ERIK (HTTP://WWW.MARSJA.SE)

January 19, 2016 at 9:20 am (https://www.analyticsvidhya.com/blog/2016/01/complete-tutorial-learn-data-science-python-scratch-2/#comment-104093)

Thank you for a real comprehensive post. Personally, I am mainly using Python for creating Psychology experiments but I would like to start doing some analysis with Python (right now I mainly use R). Some of the libraries (e.g., Seaborn) was new to me.

GT_67

January 19, 2016 at 12:15 pm (https://www.analyticsvidhya.com/blog/2016/01/complete-tutorial-learn-data-science-python-scratch-2/#comment-104100)

Hello !
I can't let this piece of code to work:

```
table = df.pivot_table(values='LoanAmount', index='Self_Employed' ,columns='Education',
aggfunc=np.median)
# Define function to return value of this pivot_table
def fage(x):
return table.loc[x['Self_Employed'],x['Education']]
# Replace missing values
df['LoanAmount'].fillna(df[df['LoanAmount'].isnull()].apply(fage, axis=1), inplace=True)
```

I've this error:

ValueError: invalid fill value with a

I checked the null values of the columns "LoanAmount", "Self_Employed" and "Education" and nothing wrong shows out. 614 values as others full columns.

Someone else had the same error ?

MOHAMED

January 20, 2016 at 1:02 am (https://www.analyticsvidhya.com/blog/2016/01/complete-tutorial-learn-data-science-python-scratch-2/#comment-104156)

Mr. gt_67,
I have same the error do you have any idea what that could be?
if Kunal can help understand and fix this piece of code will be great.

DIGNITY

February 29, 2016 at 6:14 am (https://www.analyticsvidhya.com/blog/2016/01/complete-tutorial-learn-data-science-python-scratch-2/#comment-106345)

Missing values are already replaced by the mean with this line of code (1.st way)
df['LoanAmount'].fillna(df['LoanAmount'].mean(), inplace=True)

before.

This part is the second way of replacing missing values so
if you skip above line the code should work.

MOHAMED

January 19, 2016 at 12:40 pm (https://www.analyticsvidhya.com/blog/2016/01/complete-tutorial-learn-data-science-python-scratch-2/#comment-104108)

This is a great, great resource. Thanks Kunal. But let me ask you for curiosity is this how data scientist do at work, I mean it is like using a command like to get insight from the data, isn't there GUI with python so you can be more productive?

Keep up the good work.

KISHORE

January 19, 2016 at 12:51 pm (https://www.analyticsvidhya.com/blog/2016/01/complete-tutorial-learn-data-science-python-scratch-2/#comment-104110)

Hi Kunal,

Thanks for the excellent tutorial using python. It would be great if you could do a similar tutorial using R.

Regards,
Kishore

ERIK MARSJA (HTTP://WWW.MARSJA.SE)

January 20, 2016 at 5:33 pm (https://www.analyticsvidhya.com/blog/2016/01/complete-tutorial-learn-data-science-python-scratch-2/#comment-104206)

Thank you Kunal for a real comprehensive tutorial on doing data science in Python! I really appreciated the list of libraires. Really useful. I have, my self, started to look more and more on doing data analysis with Python. I have tested pandas some and your exploratory analysis with-pandas part was also helpful.

VENU

January 24, 2016 at 5:39 am (https://www.analyticsvidhya.com/blog/2016/01/complete-tutorial-learn-data-science-python-scratch-2/#comment-104498)

Good One

HEMANTH

January 24, 2016 at 3:00 pm (https://www.analyticsvidhya.com/blog/2016/01/complete-tutorial-learn-data-science-python-scratch-2/#comment-104522)

Is there a python library for performing OCR on PDF files? or for converting a raw scanned PDF to a 'searchable' PDF? To perform Text Analytics…

ABHI

January 24, 2016 at 4:38 pm (https://www.analyticsvidhya.com/blog/2016/01/complete-tutorial-learn-data-science-python-scratch-2/#comment-104534)

Hey, great article. I find my self getting hiccups the moment probability and statistics start appearing. Can you suggest a book that takes me through these easily just like in this tutorial. Both of these seem to be the lifeline of ML.

DEEPAK

January 31, 2016 at 5:51 pm (https://www.analyticsvidhya.com/blog/2016/01/complete-tutorial-learn-data-science-python-scratch-2/#comment-104989)

Hey Kunal

Im trying to follow your lesson however I am stuck at reading ng to read it. I am following the syntax that you have provided but it still doesnt work.
Can you please help me if its possible I would really appreciate it.

Thanks

Deepak

DEEPAK

February 2, 2016 at 2:21 am (https://www.analyticsvidhya.com/blog/2016/01/complete-tutorial-learn-data-science-python-scratch-2/#comment-105054)

Hello Kunal i have started your tutorial but i am having difficulty at importing pandas an opening the csv file do you mind assisting me
Thanks

KUNAL JAIN (HTTP://WWW.ANALYTICSVIDHYA.COM)

February 3, 2016 at 11:35 am (https://www.analyticsvidhya.com/blog/2016/01/complete-tutorial-learn-data-science-python-scratch-2/#comment-105143)

Deepak,

What is the problem you are facing? Can you attach a screenshot?

Also, tell me which OS are you working on and which Python installation are you working on?

Regards,
Kunal

DEEPAK

February 3, 2016 at 4:14 pm (https://www.analyticsvidhya.com/blog/2016/01/complete-tutorial-learn-data-science-python-scratch-2/#comment-105160)

Hey Thanks for replying no i do not think i can attach a screen shot on this Blog Wall. I would love to email it to you but do not have your email address though.

But the problem i am having is trying to open the .csv file (train). I have opened pylab inline
My code is like this :

Line 1: %pylab inline
Populating the interactive namespace from numpy and matplotlib

Line 2: import pandas as pd

df = pd.read_csv("/Desktop/Studying_Tools/AV/train.csv")

When i click run in Ipython Notebook. It gives me an error Like this:

```
OSError Traceback (most recent call last)
in ()
1 import pandas as pd
2
—-> 3 df = pd.read_csv("/Desktop/Studying_Tools/AV/train.csv")

C:\Users\Deepak Mahtani\Anaconda3\lib\site-packages\pandas\io\parsers.py in parser_f(filepath_or_buffer, sep, dialect, compression, doublequote, escapechar, quotechar, quoting, skipinitialspace, lineterminator, header, index_col, names, prefix, skiprows, skipfooter, skip_footer, na_values, true_values, false_values, delimiter, converters, dtype, usecols, engine, delim_whitespace, as_recarray, na_filter, compact_ints, use_unsigned, low_memory, buffer_lines, warn_bad_lines, error_bad_lines, keep_default_na, thousands, comment, decimal, parse_dates, keep_date_col, dayfirst, date_parser, memory_map, float_precision, nrows, iterator, chunksize, verbose, encoding, squeeze, mangle_dupe_cols, tupleize_cols, infer_datetime_format, skip_blank_lines)
496 skip_blank_lines=skip_blank_lines)
497
–> 498 return _read(filepath_or_buffer, kwds)
499
500 parser_f.__name__ = name

C:\Users\Deepak Mahtani\Anaconda3\lib\site-packages\pandas\io\parsers.py in _read(filepath_or_buffer, kwds)
273
274 # Create the parser.
–> 275 parser = TextFileReader(filepath_or_buffer, **kwds)
276
277 if (nrows is not None) and (chunksize is not None):

C:\Users\Deepak Mahtani\Anaconda3\lib\site-packages\pandas\io\parsers.py in __init__(self, f, engine, **kwds)
588 self.options['has_index_names'] = kwds['has_index_names']
589
–> 590 self._make_engine(self.engine)
591
592 def _get_options_with_defaults(self, engine):

C:\Users\Deepak Mahtani\Anaconda3\lib\site-packages\pandas\io\parsers.py in _make_engine(self, engine)
729 def _make_engine(self, engine='c'):
730 if engine == 'c':
–> 731 self._engine = CParserWrapper(self.f, **self.options)
732 else:
733 if engine == 'python':
```

C:\Users\Deepak Mahtani\Anaconda3\lib\site-packages\pandas\io\parsers.py in __init__(self, src, **kwds)
1101 kwds['allow_leading_cols'] = self.index_col is not False
1102
-> 1103 self._reader = _parser.TextReader(src, **kwds)
1104
1105 # XXX

pandas\parser.pyx in pandas.parser.TextReader.__cinit__ (pandas\parser.c:3246)()

pandas\parser.pyx in pandas.parser.TextReader._setup_parser_source (pandas\parser.c:6111)()

OSError: File b'/Desktop/Studying_Tools/AV/train.csv' does not exist

Im using Anaconda
Ipython Notebook (Jupyter)- version 4.0.4
Im running it on my Windows 8 Laptop

Please try Help , and thanks again

JAINI
February 7, 2016 at 12:35 am (https://www.analyticsvidhya.com/blog/2016/01/complete-tutorial-learn-data-science-python-scratch-2/#comment-105300)

Hi Kunal

Sincere apologies for a very basic question. I have installed python per above instructions. Unfortunately I am unable to launch ipython notebook. Have spent hours but I guess I missing something. Could you please kindly guide.

Thank you

Jaini

KUNAL JAIN (HTTP://WWW.ANALYTICSVIDHYA.COM)
February 7, 2016 at 10:46 am (https://www.analyticsvidhya.com/blog/2016/01/complete-tutorial-learn-data-science-python-scratch-2/#comment-105313)

Jaini,

What is the error you are getting? Which OS you are on? And what happens when you type ipython notebook in shell / terminal / cmd ?

Regards,
Kunal

EMANUEL WOISKI

February 7, 2016 at 8:18 pm (https://www.analyticsvidhya.com/blog/2016/01/complete-tutorial-learn-data-science-python-scratch-2/#comment-105342)

Nice article!
A few remarks:
1- "–pylab=inline" is not recommended any more. Use "%matplotlib inline" for each notebook.
2- You can start a jupyter server using "jupyter notebook" instead of "ipython notebook". For me, notebooks open faster that way.
3- For plotting, use "import matplotlib.pyplot as plt".
Regards
woiski

JAINI

February 8, 2016 at 1:11 am (https://www.analyticsvidhya.com/blog/2016/01/complete-tutorial-learn-data-science-python-scratch-2/#comment-105350)

Thank you. I sincerely appreciate your instant response. I just reinstalled and went through command prompt and it worked.

NGNIKHILGOYAL

February 20, 2016 at 9:45 pm (https://www.analyticsvidhya.com/blog/2016/01/complete-tutorial-learn-data-science-python-scratch-2/#comment-105981)

IT would be good if you explained the code as you went along the exercise. For someone unfamiliar with some of the methods and functions, it is difficult to understand what is going on. Eg.: While creating the pivot table, you introduced aggfunc=lambda x: x.map({'Y':1,'N':0}).mean()) without explaining it. Intuitively I know you are coding Y as 1 and N as 0 and taking mean of each but you still need to explain what is lambda x: x.map . . . .

OLGA

February 26, 2016 at 1:06 pm (https://www.analyticsvidhya.com/blog/2016/01/complete-tutorial-learn-data-science-python-scratch-2/#comment-106263)

There seems to be a bit of confusion, when you plot histogram. Histogram, by definition, is a plot of occurrence frequency of some variable. So, when you do manipulation with ApplicantIncome, transforming to a TotalIncome by adding CoapplicantIncome, the outcome does not affect the histogram of LoanAmount,

because the outcome of this manipulation does not change the occurrence frequency or the values of LoanAmount. If you compare both of your plots, they will look exactly the same for mentioned above reason. So, it will be, probably, better to correct this part of the article.

ADULL KKU

February 29, 2016 at 3:40 am (https://www.analyticsvidhya.com/blog/2016/01/complete-tutorial-learn-data-science-python-scratch-2/#comment-106333)

Thanks

VLAD

February 29, 2016 at 11:13 pm (https://www.analyticsvidhya.com/blog/2016/01/complete-tutorial-learn-data-science-python-scratch-2/#comment-106401)

Hi Kunal – first off thanks for this informative tutorial. Great stuff. Unfortunately I'm unable to download the dataset – I need to be signed up on AV, and I get an invalid request on signup. Thank you again for this material.

VLAD

March 1, 2016 at 3:37 am (https://www.analyticsvidhya.com/blog/2016/01/complete-tutorial-learn-data-science-python-scratch-2/#comment-106408)

Worked when I tried again after a few hours. Nevermind!

DORINEL

March 10, 2016 at 1:28 pm (https://www.analyticsvidhya.com/blog/2016/01/complete-tutorial-learn-data-science-python-scratch-2/#comment-107006)

Hi Kunal,
Dont you give us access to the data set any more? I am reading your tutorial and want to repeat your steps for data analysis!
Thanks,
Dorinel

SAM

March 11, 2016 at 10:43 am (https://www.analyticsvidhya.com/blog/2016/01/complete-tutorial-learn-data-science-python-scratch-2/#comment-107088)

when running this code :

table = df.pivot_table(values='LoanAmount', index='Self_Employed' ,columns='Education', aggfunc=np.median)
# Define function to return value of this pivot_table
def fage(x):
return table.loc[x['Self_Employed'],x['Education']]
Replace missing values
df['LoanAmount'].fillna(df[df['LoanAmount'].isnull()].apply(fage, axis=1), inplace=True)

i am getting this error:
KeyError: ('the label [Graduate] is not in the [index]', u'occurred at index 0')

Any ideas?

Thanks In Advance

MARC

March 12, 2016 at 12:59 am (https://www.analyticsvidhya.com/blog/2016/01/complete-tutorial-learn-data-science-python-scratch-2/#comment-107119)

Thanks for this. Is there a way to get access to the dataset that was used for this? seems like it became unavailable from March 7!

PAVAN KUMAR

May 29, 2016 at 6:29 am (https://www.analyticsvidhya.com/blog/2016/01/complete-tutorial-learn-data-science-python-scratch-2/#comment-111563)

Really great and would start following – I am a new entry to the data analysis stream

---

HARNEET

July 28, 2016 at 4:20 am (https://www.analyticsvidhya.com/blog/2016/01/complete-tutorial-learn-data-science-python-scratch-2/#comment-114171)

Hi Kunal,

I have trying to get some validations in python for logistic regression as available for SAS, like Area Under Curve, Concordant, Discordant and Tied pairs, Ginni Value etc. But I am unable to find it through google, what ever I was able to find was very confusing.

Can you please help me with this?

Regards,
Harneet.

UMARYUSUF

August 8, 2016 at 9:54 am (https://www.analyticsvidhya.com/blog/2016/01/complete-tutorial-learn-data-science-python-scratch-2/#comment-114561)

Very well written tutorial to learn data science with python.

ASIF AMEER

August 19, 2016 at 5:06 pm (https://www.analyticsvidhya.com/blog/2016/01/complete-tutorial-learn-data-science-python-scratch-2/#comment-114960)

Really awesome Kunal Jain, I appreciate your work….

PETER FRECH

August 23, 2016 at 9:08 am (https://www.analyticsvidhya.com/blog/2016/01/complete-tutorial-learn-data-science-python-scratch-2/#comment-115048)

Hello,

very good article. I just stumbled upon one piece of code where I am not quite sure if I just don’ interpret the arguments well, or whether there is truely a mistake in your code. It is the following:

metrics.accuracy_score(predictions,data[outcome])

Isn’t “predictions” the true predictions, which should be placed as the argument “y_pred” of the accuracy_score method, and “data[outcome]” are the real values which should be associated with the argument “y_true”?

If that is so, then I think the order of passing the arguments is wrong, because the method is defined as following (according to doc): confusion_matrix(y_true, y_pred[, labels]) –> that means y_true comes as 1st argument. You have it the other way arround.

or doesn’t make it a difference at all? Anyways.

Best regards,
Peter

NICOLA

September 4, 2016 at 8:36 am (https://www.analyticsvidhya.com/blog/2016/01/complete-tutorial-learn-data-science-python-scratch-2/#comment-115563)

Hi!
And thank you very much for your tutorial

Unfortunately there is no way to find the .csv file for the loan prediction problem in https://datahack.analyticsvidhya.com/contest/practice-problem-loan-prediction-iii/ (https://datahack.analyticsvidhya.com/contest/practice-problem-loan-prediction-iii/)

WAYNE

September 18, 2016 at 1:32 am (https://www.analyticsvidhya.com/blog/2016/01/complete-tutorial-learn-data-science-python-scratch-2/#comment-116168)

Hello,

Thank you for the tutorial.
But as already mentioned by Nicola, there is no way to download the DataSet.
Could you please check it?

Thanks

GOPALANKAILASH

October 28, 2016 at 3:49 pm (https://www.analyticsvidhya.com/blog/2016/01/complete-tutorial-learn-data-science-python-scratch-2/#comment-117605)

The amount of effort you guys put into these article is a true inspiration for folks like me to learn!
Thanks for all this!

JACK MA

November 8, 2016 at 10:52 pm (https://www.analyticsvidhya.com/blog/2016/01/complete-tutorial-learn-data-science-python-scratch-2/#comment-118148)

Great one. Thank you.

When I type in “df.describe() ” , it works, but it gives me a warning information :
“user\AppData\Local\Continuum\Anaconda3\lib\site-packa
RuntimeWarning: Invalid value encountered in percentile
RuntimeWarning) ”
What is it means?

Secondly, when I running “df[‘ApplicantIncome’].hist(bins=50)”
It tells me “”, so I can not see the chart.

Anyone can helps? Thank you.

POPULAR POSTS

7 Innovative Machine Learning GitHub Projects you Should Try Out in Python (https://www.analyticsvidhya.com/blog/2019/08/7-innovative-machine-learning-github-projects-in-python/)

24 Ultimate Data Science Projects To Boost Your Knowledge and Skills (& can be accessed freely) (https://www.analyticsvidhya.com/blog/2018/05/24-ultimate-data-science-projects-to-boost-your-knowledge-and-skills/)

Commonly used Machine Learning Algorithms (with Python and R Codes) (https://www.analyticsvidhya.com/blog/2017/09/common-machine-learning-algorithms/)

A Complete Python Tutorial to Learn Data Science from Scratch (https://www.analyticsvidhya.com/blog/2016/01/complete-tutorial-learn-data-science-python-scratch-2/)

7 Regression Techniques you should know! (https://www.analyticsvidhya.com/blog/2015/08/comprehensive-guide-regression/)

Stock Prices Prediction Using Machine Learning and Deep Learning Techniques (with Python codes) (https://www.analyticsvidhya.com/blog/2018/10/predicting-stock-price-machine-learningnd-deep-learning-techniques-python/)

Complete Guide to Parameter Tuning in XGBoost with codes in Python (https://www.analyticsvidhya.com/blog/2016/03/complete-guide-parameter-tuning-xgboost-with-codes-python/)

Understanding Support Vector Machine algorithm from e (https://www.analyticsvidhya.com/blog/2017/09/understaing-support-vector-machine-example-code/)

RECENT POSTS

Master Dimensionality Reduction with these 5 Must-Know Applications of Singular Value Decomposition (SVD) in Data Science (https://www.analyticsvidhya.com/blog/2019/08/5-applications-singular-value-decomposition-svd-data-science/)

AUGUST 5, 2019

7 Innovative Machine Learning GitHub Projects you Should Try Out in Python (https://www.analyticsvidhya.com/blog/2019/08/7-innovative-machine-learning-github-projects-in-python/)

AUGUST 2, 2019

A Data Science Leader's Guide to Managing Stakeholders (https://www.analyticsvidhya.com/blog/2019/08/data-science-leader-guide-managing-stakeholders/)

AUGUST 1, 2019

Building a Recommendation System using Word2vec: A Unique Tutorial with Case Study in Python (https://www.analyticsvidhya.com/blog/2019/07/how-to-build-recommendation-system-word2vec-python/)

JULY 30, 2019

(https://courses.analyticsvidhya.com/courses/natural-language-processing-nlp?utm_source=Sticky_banner1&utm_medium=display&utm_campaign=NLPcourse)

(https://datamin.analyticsvidhya.com/?utm_source=Sticky_banner2&utm_medium=display&utm_campaign=datamin)

-
(http://play.google.com/store/apps/details?id=com.analyticsvidhya.android)
ت