DATA SCIENCE

> by Dr Juan H Klopper

- Research Fellow
- School for Data Science and Computational Thinking
- Stellenbosch University

INTRODUCTION

Welcome to this course resource, created for you by the School of Data Science and Computational thinking at Stellenbosch University.

There is little doubt that you have not come across the term **Data Science**. The ability to learn from data defines our scientific endevour. **Data hides useful information**, whether that information be in spreadsheets, images, sounds, and even the written word. Modern tools are making extracting knowledge from data ever more powerful, and perhaps more importantly, easier.

Gaining access to the information in data has evolved. We currently have an abundance of riches, both is software and in hardware. Every day increases the amount of data that we generate and capture. We live in an age of information. Never before has it been so important to learn to tap into this information.

This course is all about tapping into that information using one of the most popular tools in existance. Python is an open source computer language. It is ideally suited to data science and its popularity has skyrocketed over the last decade. It is fast becoming one of the most important languages to speak, arguably starting to rival the importance of being able to communicate with the spoken and written word.

Great efforts are made all over the world to democratise access to this new language. Leading Universities have campus wide courses in data science using Python. Multiple online companies have sprung up teaching Data Science using Python. A quick YouTube search will show an endless supply of tutorials on data science and Python. Access to data science has truly been democratised.

This freedom of access to resources for what is a free and open language, is one of the cornerstones of the success of Python and Data Science as a whole. You will never get stuck on this new journey that you are about to embark upon. Help is available everywhere. Developing this new skill has never been easier.

In this course we will define Data Science, understand common terms and defintions used in Data Science, explore the tools of trade, and learn the most usefull techniques in our pursuit to understand and learn fro our data.

DEFINING DATA SCIENCE

Data Science is about the exploration of data in order to learn from it, draw conclusion from it, predict with it, and use it to infer results to new situations, remaining cognisant of the uncertainty in our results.

As a science it brings together an array of approaches to understanding and using data. This includes data generation, data capture and storage, data verification and manipulation, statistics, and a variety of modern learning algorithms such as machine learning.

The common techniques and principles of this science makes it applicable to an enormous array of real-world situations. From earth and climate science to astronomy, cosmology, physics, biology, chemistry, engineering, healthcare, economics, politics, and so many more fields and subfields.

Alll of this made possible by today's modern computing infrastructure and access to the internet. This course is produced to run in the cloud. No need to even install anything on your local computer. These services for Data Science are available free of charge. Access to the internet is the only barrier to entry. With this statement I add my voice to the international plea to governments to see access to the internet as a basic human right. The more minds we add to our problems, the easier they will be to solve. With these solutions will come the solutions to poverty, hunger, climate change, illness, and so many more of our current and future challenges as a species on *a mote of dust suspended in a sunbeam*. *Carl Sagan*.

THE TOOLS OF DATA SCIENCE

COMPUTER LANGUAGES

Open source computer languages such as Python, R, and Julia are languages that allow us to interact with and instruct our computers to perform useful tasks. As with the spoken languages they have structure and rules.

The advantages of these languages for us as Data Scientists lie in their ease of access and use. As open source languages they are freely available, not contrlled by some corporation. Instead they thrive on a community of developers and mainainers from all over the world that, in most cases, spend their free time to continuously update the tools. It is a tremendous achievement of humanity that such goodwill can lead to such success.

Newer versions of these languages are introduced at regular intervals, much like your cellphone operating system. The decision on what new changes are incorporated is decided by groups of developers. In the case of Python this use to be lead by the creator of the languge, Guido van Rossum, who perceived of the language all the way back in the 1980's. Julia is maintained by its developers out of the Massachusetts Institute of Technology.

All of these languages are expanded by packages, modules, and libraries. These are pieces of software written, once again mostly, by volunteers. They expand on the core language by providing convenient and useful additions to language. This course will teach you about the most common and powerful Python packages for Data Science.

Any computer language requires a piece of software in which we can type our code. The majority are integrated development environments (IDE). These environments are programs, much like Microsoft Word.

It has also become popular to use modern web browsers as coding environments. This allows us to write code in the cloud, without using the local recources of our machines. For the purposes of security and data safety, these browser environments can also be created on local machines as easy installs. For the purposes of this course, we will use Google Drive as there are no sensitive data that will be used.

GOOGLE COLAB

Google Drive provides many tools such as Google Docs for writing reports and letters and Google Sheets for data capture as spreadsheets. It also provides a tool called **Colaboratory**.

Once you haev created a free Google account, you will have access to Google email and a suite of other online apps. While you can use most browsers, Google Chrome will work best. Once signed into your account, you can type the following URL into the address bar of the browser: https://colab.research.google.com

This will make Google Colab available as another Google online app in your account. A Google Colab file or document is termed a **notebook**. It is based on the Jupyter notebook (which you can install and use on your local machine).

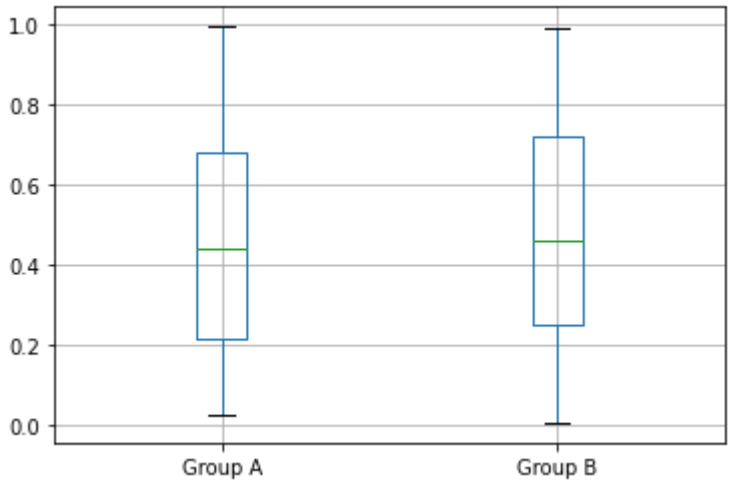
COLAB NOTEBOOK

This document was created as a Colab notebook. It allows for writing of formatted words, sentences, paragraphs, and sections as you can see here. We can even import and show images (as seen by our logo at the top of this notebook). It also allows for the typing and execution of Python code as can be seen below. Don't be concerned about the code. We will learn all about it in upcoming notebooks.

```
#%config InlineBackend.figure_format = "retina"

import numpy as np
import pandas as pd

df = pd.DataFrame(
    np.random.rand(100, 2),
    columns=['Group A', 'Group B']
)

df.describe()
```

```
   Group A    Group B
```

```
1 df.boxplot();
```



Both *normal written language* and code are entered into **cells**. A cell can be a text cell or a code cell. In the two code cells above, we generated some random variables and first calculated summary statistics for the two variables and then generated a box-and-whisker plot of the data.

When hovering in the middle of the page, between two cells (or at the bottom of the last cell), two buttons pop up. They are marked `+ Code` and `+ Text`. They generate new cell right below the buttons.

The text cells have a tool bar, much like a simple program. The buttons in the toolbar allow for formatting of text and insertion of images and links.

Text cells can format text using Markdown language. Makdown language uses special characters to format text. For instance adding an underscore before and after a word or words, prints them in *italics*. Double underscores print in **bold**.

Text cells can also print LaTex. LaTex is a language for matematical typesetting. In (1) below, we see the equation for the normal distribution given a mean, μ, and a standard deviation, σ.

$$f\left(x|\mu,\sigma\right)=\frac{1}{\sigma\sqrt{2\pi}}e^{-\frac{1}{2}\left(\frac{x-\mu}{\sigma}\right)^2} \tag{1}$$

The top of the notebook also has a toolbar with the usual File, Edit, View, and other buttons. We will become familiar with these throughout the course.

Since this is a Google app, there is also some familiar buttons on the top right. These allow us to share a notebook with our coworkers and collaborators. We can also leave comments, just as

with a Google Doc.

At the top of the left pane (which is collapsable), are four icons for a table of content, search, code snippets, and a folder explorer. The first is a very convenient *table of contents*. When we use titles and subtitles, these appear in the table of contents.

The beauty of the notebook format as coding environment, is that is generates a document that serves as a very useful report. We can include *normal text*, images, links, formatted input, and our code. It makes for expressive research documents that can be used for collaboration, sharing, and presentation.

Below, is a brief example of a research project. The data comes from a real-life survey run by an organisation called Kaggle. You can sign up for a free account. Kaggle even provides a very similar notebook environment to Google Colab.

The survey is run every year and is completed by Data Scientists from across the Globe. It gives a unique insigth into who Data Scientists are, what tools they use, and even includes their salaries.

This example is for demonstration purposes only. As mentioned above, do not be concerned about the code. Appreciate the results and the use of a notebook as research document, combining text, code, and results to gain insight.

DATA SCIENCE SURVEY EXAMPLE

PACKAGES USED IN THIS NOTEBOOK

Packages are the extension to the langauge developed by enthusiasts, acadmics, and professionals. Below, we import these for use in the example project. You will notice pound symbols (hashtags) followed by some green text. This symbol in Python denotes a code comment. Python ignores everything in a line after a pound symbol. It is good practice to leave code comments, both to your future self, and to others who might read your code.

```
# Statistics module of the scipy package
from scipy import stats
```

```
# Plotting package import
import plotly.graph_objects as go
import plotly.express as px
```

```
import plotly.figure_factory as ff
import plotly.io as pio
pio.templates.default = 'plotly_white'
```

```
# Format tables printed to the screen (don't put this on the same line as the co
%load_ext google.colab.data_table
```

```
# For connecting to Google Drive
from google.colab import drive
```

CONNECTING TO GOOGLE DRIVE

The data is in the Google Drive in which this notebook was created. It was saved in a subfolder. We need to log into our account again to generate a security key which we copy and paste into the cell below. Simply follow the on-screen instructions and allow the required permissions. There is also some code to navigate to the subfolder.

```
# Connect to your Google Drive
drive.mount('/gdrive', force_remount=True)
```

```
Mounted at /gdrive
```

```
'/gdrive/My Drive/Stellenbosch University/School for Data Science and Computati
```

```
/gdrive/My Drive/Stellenbosch University/School for Data Science and Computat
```

IMPORTING THE DATA

The data is saved in a comma seperated values (CSV) spreadsheet file.

```
df = pd.read_csv('kaggle_survey_2020_responses.csv', low_memory=False)
```

The data is displayed as a spreadsheet to make sure that it imported properly.

```
# Drop first row containing metadata
df.drop(0, axis=0, inplace=True)
# First five rows of data
df[:5]
```

	Time from Start to Finish (seconds)	Q1	Q2	Q3	Q4	Q5	Q6	Q7_Part_1	Q7_Part_2
1	1838	35-39	Man	Colombia	Doctoral degree	Student	5-10 years	Python	R
2	289287	30-34	Man	United States of America	Master's degree	Data Engineer	5-10 years	Python	R
3	860	35-39	Man	Argentina	Bachelor's degree	Software Engineer	10-20 years	NaN	NaN
4	507	30-34	Man	United States of America	Master's degree	Data Scientist	5-10 years	Python	NaN
		30-			Master's	Software	3-5		

The number of observations and variables are returned to give us an indication of the dimensions of our data set.

```
1 # The shape of the dataframe object
2 df.shape
```

```
(20036, 355)
```

We note that there are 200036 observations (respondents to the survey), each completing data for 355 variables.

RESPONDENT QUALIFICATIONS

Calculating the frequency of different (highest) qualifications.

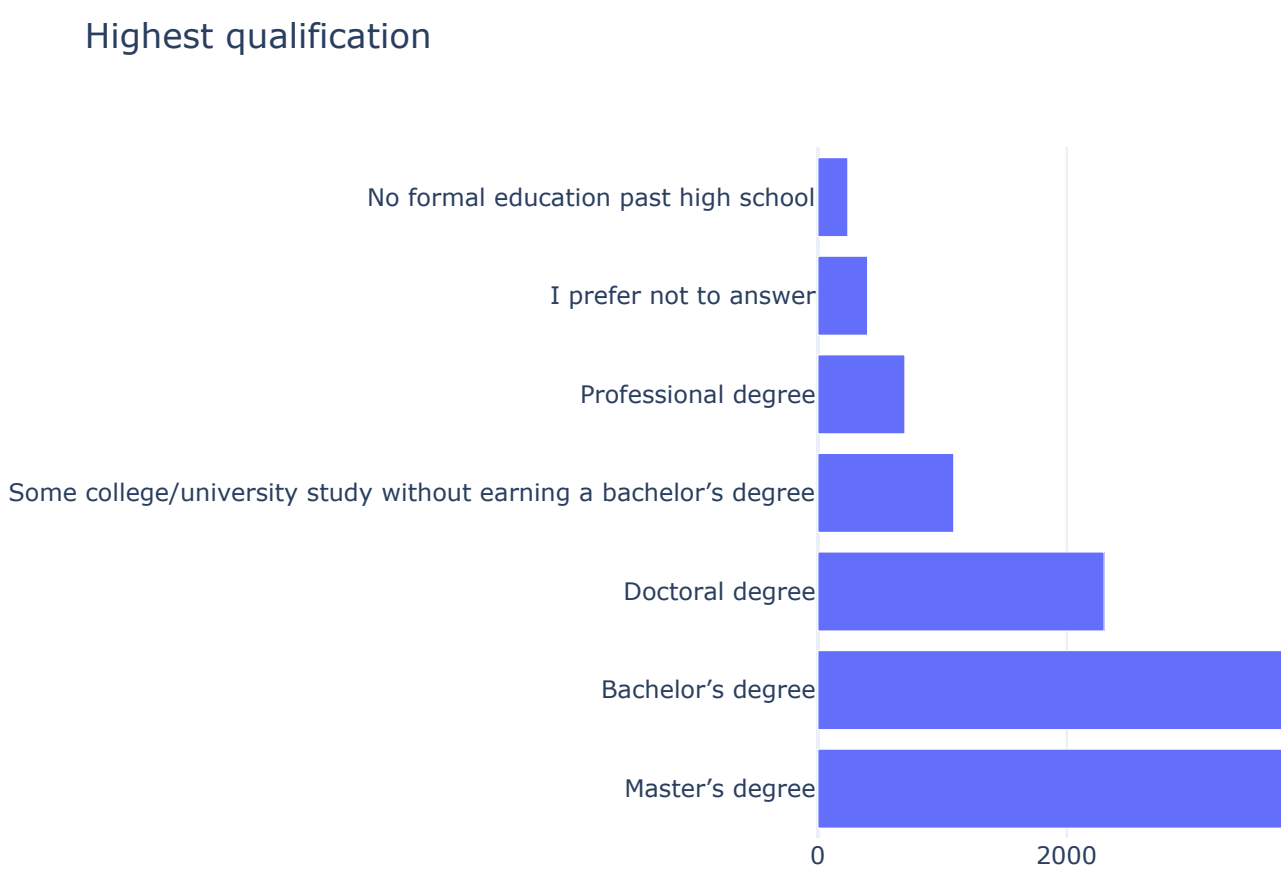
```
1 # Frequency of sample space elements for Q4 (highest qualification)
2 df.Q4.value_counts()
```

```
Master's degree                                                   7859
Bachelor's degree                                                 6978
Doctoral degree                                                   2302
Some college/university study without earning a bachelor's degree 1092
Professional degree                                                699
I prefer not to answer                                             399
No formal education past high school                               240
Name: Q4, dtype: int64
```

The majority of candidates have Master's and Bachelor's degrees. More than 2000 have a PhD.

A bar plot provides a visual analysis of the data. With the plotly packages, we can hover over elements in the plot for more information. We can also pan, zoom, and save the image to disk.

```
go.Figure(
    go.Bar(
        y=df.Q4.value_counts().index.tolist(),
        x=df.Q4.value_counts().values.tolist(),
        orientation='h'
    )
).update_layout(
    title='Highest qualification',
    xaxis={'title':'Count'}
)
```



PYTHON AND R

Next up, we investigate the number of Data Scientist that use Python and R (and the previous leader, C) in their daily work. We also include Structured Query Language used in databases.

```
go.Figure(
```

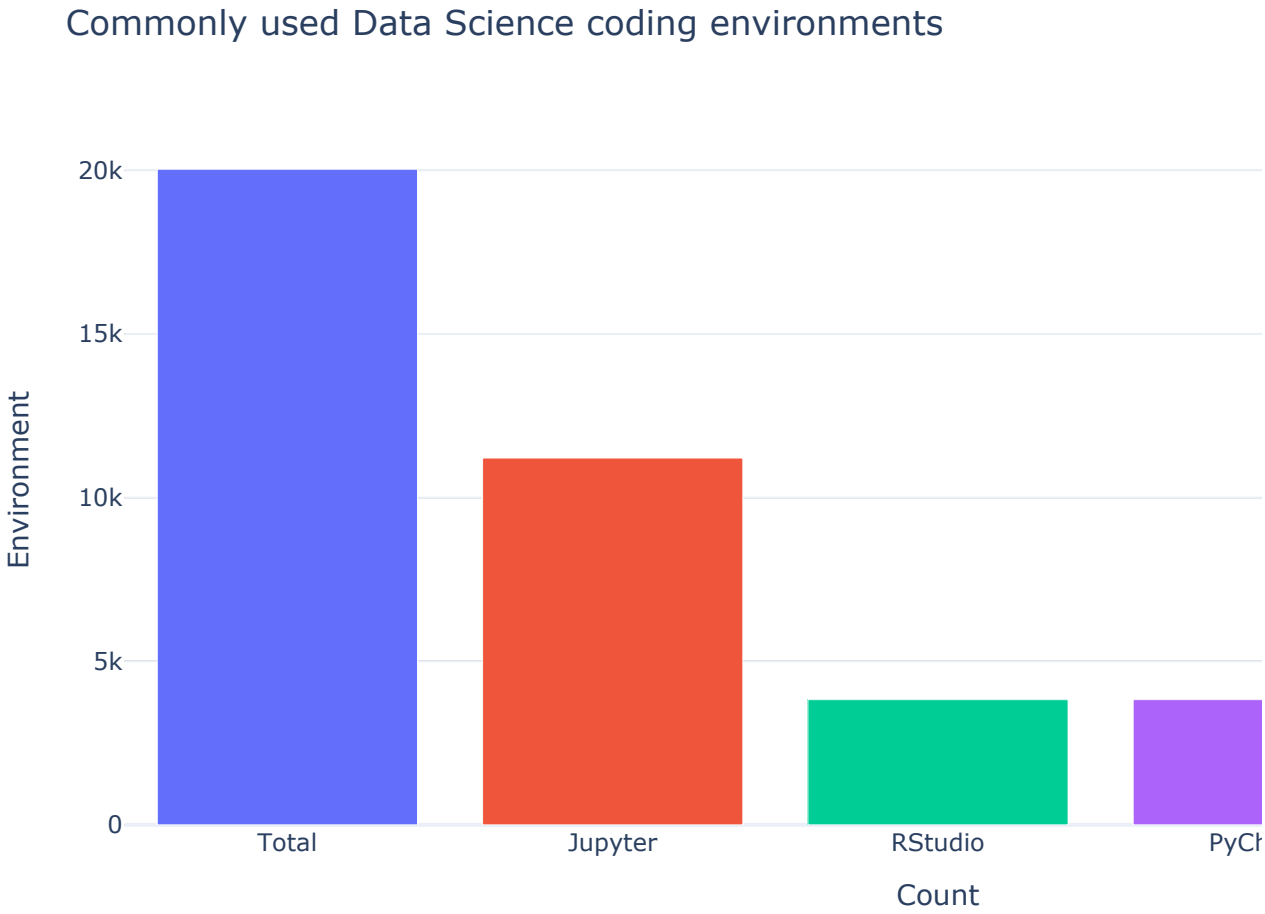
```
    go.Bar(
        x=['Total'],
        y=[df.shape[0]],
        name='Total'
    )
).add_trace(
    go.Bar(
        x=['Python'],
        y=df.Q7_Part_1.value_counts().values.tolist(),
        name='Python'
    )
).add_trace(
    go.Bar(
        x=['R'],
        y=df.Q7_Part_2.value_counts().values.tolist(),
        name='R'
    )
).add_trace(
    go.Bar(
        x=['C'],
        y=df.Q7_Part_4.value_counts().values.tolist(),
        name='R'
    )
).add_trace(
    go.Bar(
        x=['SQL'],
        y=df.Q7_Part_3.value_counts().values.tolist(),
        name='SQL'
    )
).update_layout(title='Commonly used Data Science languages',
                xaxis={'title':'Count'},
                yaxis={'title':'Language'})
```

We note that Python is the leading language used in Data Science.

CODING ENVIRONMENTS

We investigate teh comonly used coding environments and compare Jupyter notebooks (on which Google Colab is based), as well as RStudio, PyCharm, and Microsoft Visual Studio Cloud. Free versions of these environments are either in the cloud or can be installed on your local machine.

```
go.Figure(
    go.Bar(
        x=['Total'],
        y=[df.shape[0]],
        name='Total'
    )
).add_trace(
    go.Bar(
        x=['Jupyter'],
        y=df.Q9_Part_1.value_counts().values.tolist(),
        name='Jupyter'
    )
).add_trace(
    go.Bar(
        x=['RStudio'],
        y=df.Q9_Part_2.value_counts().values.tolist(),
        name='R'
    )
).add_trace(
    go.Bar(
        x=['PyCharm'],
        y=df.Q7_Part_5.value_counts().values.tolist(),
        name='R'
    )
).add_trace(
    go.Bar(
        x=['VS Code'],
        y=df.Q7_Part_4.value_counts().values.tolist(),
        name='R'
    )
).update_layout(title='Commonly used Data Science coding environments',
                xaxis={'title':'Count'},
                yaxis={'title':'Environment'})
```



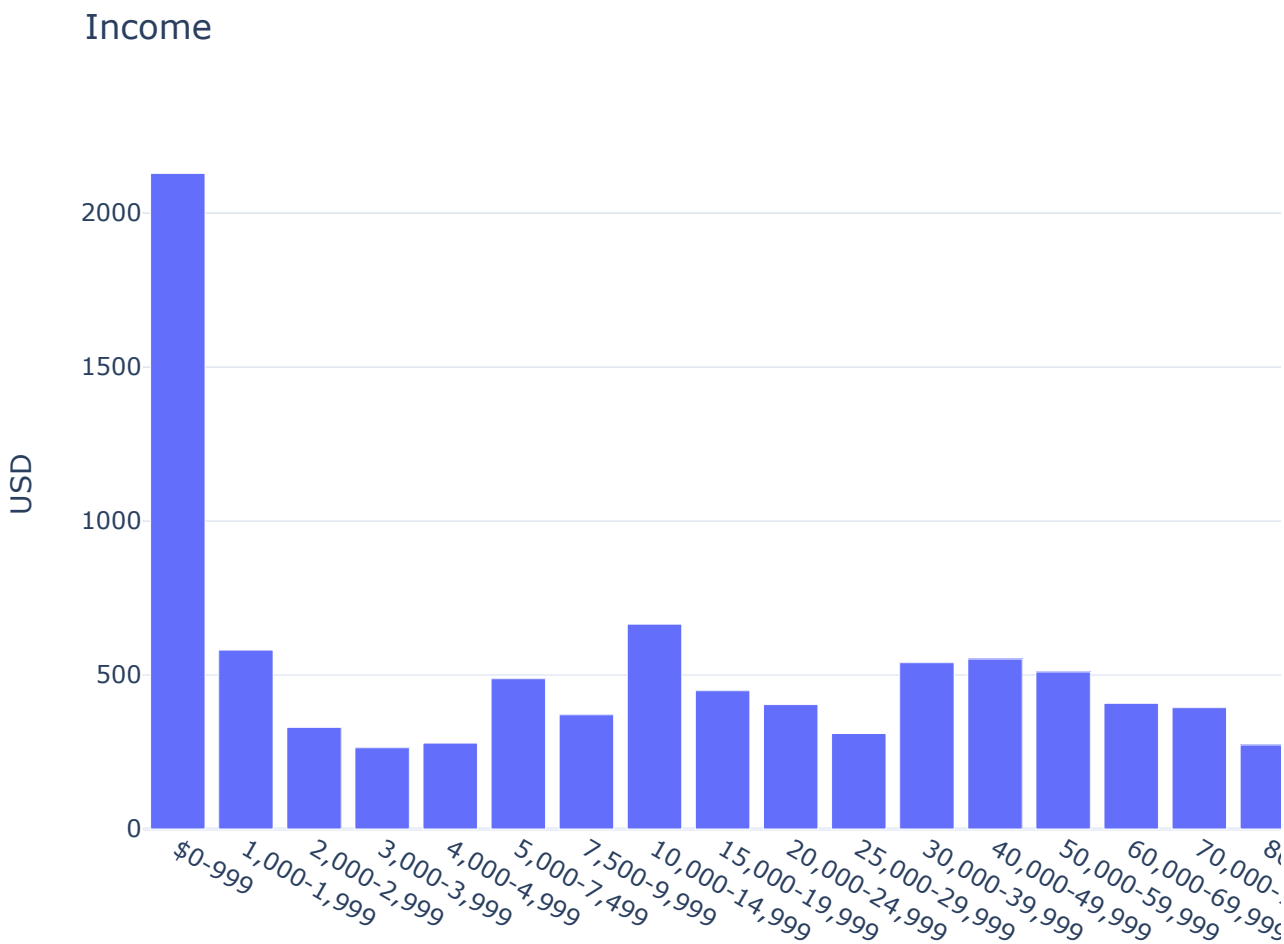
Jupyter notebooks are very popular. They provide an excellent coding envirnoment for the analysis of data.

INCOME

The survey asked respondents to select a salary bracket. We visualise the frequency of all of the brackets.

```
# Saving an ordered list of the brackets as dictionary key-value pairs
income_brackets = {
    '$0-999':2128,
    '1,000-1,999':581,
    '2,000-2,999':330,
    '3,000-3,999':264,
    '4,000-4,999':279,
    '5,000-7,499':488,
    '7,500-9,999':371,
    '10,000-14,999':665,
    '15,000-19,999':449,
    '20,000-24,999':404,
    '25,000-29,999':310,
    '30,000-39,999':540,
    '40,000-49,999':552,
```

```
    '50,000-59,999':510,
    '60,000-69,999':408,
    '70,000-79,999':394,
    '80,000-89,999':273,
    '90,000-99,999':280,
    '100,000-124,999':573,
    '125,000-149,999':315,
    '150,000-199,999':347,
    '200,000-249,999':115,
    '250,000-299,999':48,
    '300,000-500,000':55,
    '> $500,000':50
}

go.Figure(
    go.Bar(
        x=list(income_brackets.keys()),
        y=list(income_brackets.values())
    )
).update_layout(
    title='Income',
    yaxis={'title':'USD'}
)
```



There is a definite bump at the 100000 USD mark. There are some well paid Data Scientist.

INCOME DIFFERENCES FOR WOMEN

The underrespresentation of groups other than men is a continued problem in Data Science. We use the 100000 USD mark to distinguish high income individuals from the rest.

```
# All high income brackets (above $100000)
high_income = ['100,000-124,999',
               '125,000-149,999',
               '150,000-199,999',
               '200,000-249,999',
               '250,000-299,999',
               '300,000-500,000',
               '> $500,000']
```

```
# Add a new column for income above and below $100000
df['HighIncome'] = np.where(df.Q24.isin(high_income), 'Yes', 'No')
```

For the sake of simplicity and to answer our specific research question regarding women, we only extract those that identified as *Man* or *Woman*.

```
# Select only binary sex participants
binary_gender = df.loc[(df.Q2 == 'Man') | (df.Q2 == 'Woman')]
```

A contingency table of observed values shows the scale of the problem.

```
# Contigency table of observed values
pd.crosstab(
    binary_gender.Q2,
    binary_gender.HighIncome,
    margins=True
)
```

1 to 3 of 3 entries Filter

Q2	No	Yes	All
Man	14494	1295	15789
Woman	3706	172	3878
All	18200	1467	19667

Show 25 per page

```
# Relative frequency observed value contingency table
pd.crosstab(
    binary_gender.Q2,
    binary_gender.HighIncome,
    margins=True,
    normalize=True
```

```
7 )
```

1 to 3 of 3 entries Filter

Q2	No	Yes	All
Man	0.73697055982102	0.06584634158743072	0.8028169014084507
Woman	0.1884374841104388	0.00874561448111049	0.19718309859154928
All	0.9254080439314588	0.07459195606854122	1.0

Show 25 per page

When selecting only those who identified as either man or women, only 19.7% were woman. Of these, 4.4% were in the high income brakcet. For men, the figure as almost double at 8.2%.

A χ^2 test for independence can investigate the dependence between sex and income.

```
1 stats.chi2_contingency([[14494, 1295], [3706, 172]])
```

```
(63.444674249142516,
 1.6493472192256754e-15,
 1,
 array([[14611.26760563,  1177.73239437],
        [ 3588.73239437,   289.26760563]]))
```

There was a significant association between gender and high income status, χ^2 value $63,4$, p value < 0.01.

UNITED STATES OF AMERICA VS SOUTH AFRICA

We investigate of there is a dependence between country (United States of America and South Africa) and income level (again using the 100000 USD cut-off.

```
1 usa_sa_df = df.loc[df.Q3.isin(['South Africa', 'United States of America'])]
```

```
1 pd.crosstab(
2     usa_sa_df.Q3,
3     usa_sa_df.HighIncome,
4     margins=True
5 )
```

1 to 3 of 3 entries Filter

Q3	No	Yes	All
South Africa	133	8	141
United States of America	1406	831	2237
All	1539	839	2378

Show 25 per page

```
stats.chi2_contingency(
    pd.crosstab(
        usa_sa_df.Q3,
        usa_sa_df.HighIncome
    )
)
```

```
(56.17463007230668,
 6.631168127655343e-14,
 1,
 array([[  91.25273339,   49.74726661],
        [1447.74726661,  789.25273339]]))
```

A total of 2237 respondents where from the USA compared to only 141 from South Africa. Thirty-seven percent of the American respondents where in the high income bracket compared to only 5.7% of the South African respondents, χ^2 value 56.2, p value < 0.01.

PREDICTING INCOME BRACKET USING MACHINE LEARNING

We generate a random forest machine learning model and use age, sex, country, highest qualification, and whether the respondent uses Python as feature variable in a supervised learning application, predicting income level (high or not).

```
# Selecting the feature and target variables
variables = [
             'Q1',
             'Q2',
             'Q3',
             'Q4',
             'Q7_Part_1',
             'HighIncome'
]

rf_data = df[variables]
```

We will use the TensorFlow DecisionForest module from Google to create the model.

```
# Install this package if required
!pip install tensorflow_decision_forests
```

```
Collecting tensorflow_decision_forests
  Downloading https://files.pythonhosted.org/packages/3a/34/10f20fe95d9882b82
     |████████████████████████████████| 6.2MB 6.5MB/s
Requirement already satisfied: tensorflow~=2.5 in /usr/local/lib/python3.7/di
Requirement already satisfied: absl-py in /usr/local/lib/python3.7/dist-packa
Requirement already satisfied: numpy in /usr/local/lib/python3.7/dist-package
Requirement already satisfied: wheel in /usr/local/lib/python3.7/dist-package
Requirement already satisfied: pandas in /usr/local/lib/python3.7/dist-packag
Requirement already satisfied: six in /usr/local/lib/python3.7/dist-packages
Requirement already satisfied: grpcio~=1.34.0 in /usr/local/lib/python3.7/dis
```

```
Requirement already satisfied: flatbuffers~=1.12.0 in /usr/local/lib/python3.
Requirement already satisfied: gast==0.4.0 in /usr/local/lib/python3.7/dist-p
Requirement already satisfied: termcolor~=1.1.0 in /usr/local/lib/python3.7/d
Requirement already satisfied: keras-nightly~=2.5.0.dev in /usr/local/lib/pyt
Requirement already satisfied: astunparse~=1.6.3 in /usr/local/lib/python3.7/
Requirement already satisfied: keras-preprocessing~=1.1.2 in /usr/local/lib/p
Requirement already satisfied: tensorboard~=2.5 in /usr/local/lib/python3.7/d
Requirement already satisfied: h5py~=3.1.0 in /usr/local/lib/python3.7/dist-p
Requirement already satisfied: opt-einsum~=3.3.0 in /usr/local/lib/python3.7/
Requirement already satisfied: wrapt~=1.12.1 in /usr/local/lib/python3.7/dist
Requirement already satisfied: tensorflow-estimator<2.6.0,>=2.5.0rc0 in /usr/
Requirement already satisfied: google-pasta~=0.2 in /usr/local/lib/python3.7/
Requirement already satisfied: typing-extensions~=3.7.4 in /usr/local/lib/pyt
Requirement already satisfied: protobuf>=3.9.2 in /usr/local/lib/python3.7/di
Requirement already satisfied: pytz>=2017.2 in /usr/local/lib/python3.7/dist-
Requirement already satisfied: python-dateutil>=2.7.3 in /usr/local/lib/pytho
Requirement already satisfied: markdown>=2.6.8 in /usr/local/lib/python3.7/di
Requirement already satisfied: tensorboard-plugin-wit>=1.6.0 in /usr/local/li
Requirement already satisfied: tensorboard-data-server<0.7.0,>=0.6.0 in /usr/
Requirement already satisfied: werkzeug>=0.11.15 in /usr/local/lib/python3.7/
Requirement already satisfied: requests<3,>=2.21.0 in /usr/local/lib/python3.
Requirement already satisfied: setuptools>=41.0.0 in /usr/local/lib/python3.7
Requirement already satisfied: google-auth-oauthlib<0.5,>=0.4.1 in /usr/local
Requirement already satisfied: google-auth<2,>=1.6.3 in /usr/local/lib/python
Requirement already satisfied: cached-property; python_version < "3.8" in /us
Requirement already satisfied: importlib-metadata; python_version < "3.8" in
Requirement already satisfied: idna<3,>=2.5 in /usr/local/lib/python3.7/dist-
Requirement already satisfied: urllib3!=1.25.0,!=1.25.1,<1.26,>=1.21.1 in /us
Requirement already satisfied: certifi>=2017.4.17 in /usr/local/lib/python3.7
Requirement already satisfied: chardet<4,>=3.0.2 in /usr/local/lib/python3.7/
Requirement already satisfied: requests-oauthlib>=0.7.0 in /usr/local/lib/pyt
Requirement already satisfied: cachetools<5.0,>=2.0.0 in /usr/local/lib/pytho
Requirement already satisfied: rsa<5,>=3.1.4; python_version >= "3.6" in /usr
Requirement already satisfied: pyasn1-modules>=0.2.1 in /usr/local/lib/python
Requirement already satisfied: zipp>=0.5 in /usr/local/lib/python3.7/dist-pac
Requirement already satisfied: oauthlib>=3.0.0 in /usr/local/lib/python3.7/di
Requirement already satisfied: pyasn1>=0.1.3 in /usr/local/lib/python3.7/dist-
Installing collected packages: tensorflow-decision-forests
Successfully installed tensorflow-decision-forests-0.1.7
```

```
# Importing the required packages
import tensorflow_decision_forests as tfdf
import tensorflow as tf
```

```
classes = rf_data.HighIncome.unique().tolist() # A list of the two income classe
classes
```

```
['No', 'Yes']
```

```
rf_data.HighIncome = rf_data.HighIncome.map(classes.index)
```

The data set is split into a training set, which the decision forest model will use to *learn* from, and a test set, against whichh we can evaluate the trained machine learning model.

```
def split(ds, r=0.3):
```

```
def split(ds, r=0.8):
  test_ind = np.random.rand(len(ds)) < r

  return ds[~test_ind], ds[test_ind]
```

```
rf_data_train, rf_data_test = split(rf_data)
```

There is class imbalance with 92% of respondents not in the high income brakcet. Simply predicting *NOT HIGH INCLOME BRACKET* for all cases, makes you 93% correct all the time. Our model will have to beat this metric.

```
rf_data_train.HighIncome.value_counts(normalize=True)
```

```
0    0.925234
1    0.074766
Name: HighIncome, dtype: float64
```

The decision forest class is instantiated and the model compiles, prior to passing the training data to the model.

```
rf_model = tfdf.keras.RandomForestModel()
```

```
rf_model.compile(
    metrics=['accuracy']
)
```

```
rf_data_train = tfdf.keras.pd_dataframe_to_tf_dataset(
    rf_data_train,
    label='HighIncome'
)
rf_data_test = tfdf.keras.pd_dataframe_to_tf_dataset(
    rf_data_test,
    label='HighIncome'
)
```

```
rf_model.fit(
    x=rf_data_train
)
```

```
219/219 [==============================] - 5s 1ms/step
<tensorflow.python.keras.callbacks.History at 0x7fbc2022a5d0>
```

The model is now evaluated on data it has never seen (the test data).

```
evaluation = rf_model.evaluate(
    rf_data_test,
    return_dict=True # Returning a dictionary of metrics
)
```

```
95/95 [==============================] - 1s 5ms/step - loss: 0.0000e+00 - acc
```

```
evaluation.keys() # Metric dictionary keys
```

```
dict_keys(['loss', 'accuracy'])
```

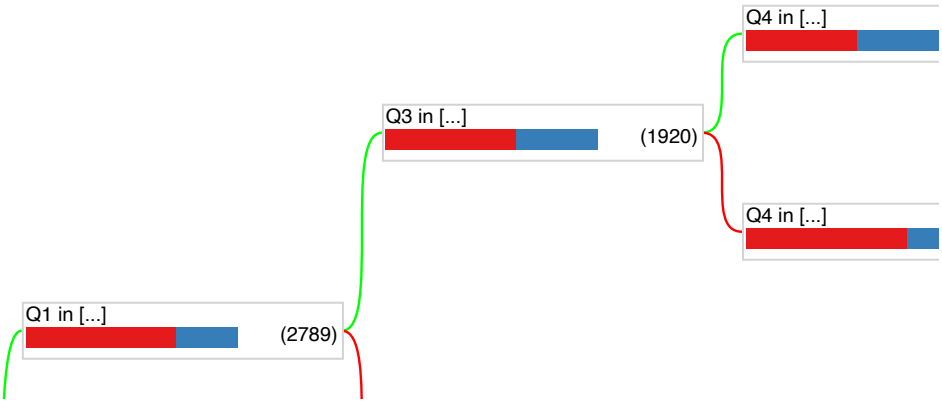
```
evaluation.values() # Metric values
```

```
dict_values([0.0, 0.9239148497581482])
```

We see an accuracy in excess of our baseline model.

Below, we plot the decision algorithm learned by this model.

```
tfdf.model_plotter.plot_model_in_colab(
    rf_model,
    tree_idx=0,
    max_depth=4
)
```




```
1 rf_model.make_inspector().variable_importances()
```

```
{'NUM_AS_ROOT': [("Q3" (4; #2), 300.0)]}
```

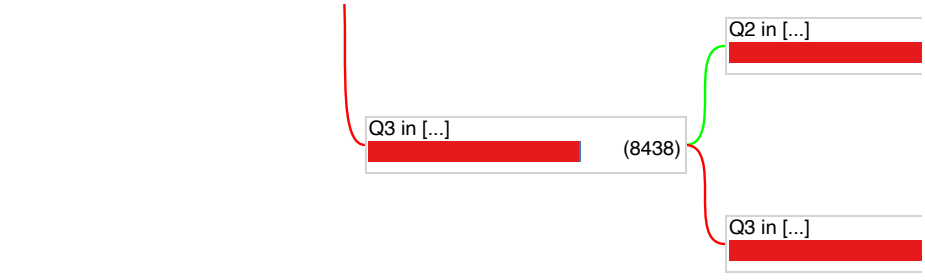
We note that the country of origin was the *most important variable* in the model.

CONCLUSION

This notebook introduced the topic, its tools, and a brief example of an Data Science project, incorporating a machine learning model.

At the end of this course you will be empowered to create all of this and much, much more.

```
1
```



✓ 0s completed at 14:18