

Configuration Manual

MSc Research Project
Data Analytics

Aidan Browne
Student ID: 16140818

School of Computing
National College of Ireland

Supervisor: Dr. Catherine Mulwa

National College of Ireland
MSc Project Submission Sheet



School of Computing

Student Name: Aidan Browne

Student ID: 16140818

Programme: MSc Data Analytics **Year:** 2020

Module: Research Project

Lecturer: Dr Catherine Mulwa

Submission Due Date: 28/09/2020

Project Title: Fine-Grained Sentiment Analysis of Yelp Reviews using Deep Learning Models

Word Count: 4,866 **Page Count:** 93

I hereby certify that the information contained in this (my submission) is information pertaining to research I conducted for this project. All information other than my own contribution will be fully referenced and listed in the relevant bibliography section at the rear of the project.

ALL internet material must be referenced in the bibliography section. Students are required to use the Referencing Standard specified in the report template. To use other author's written or electronic work is illegal (plagiarism) and may result in disciplinary action.

Signature *Aidan Browne*
:

Date: 28/09/2020
.....

PLEASE READ THE FOLLOWING INSTRUCTIONS AND CHECKLIST

Attach a completed copy of this sheet to each project (including multiple copies)	<input type="checkbox"/>
Attach a Moodle submission receipt of the online project submission , to each project (including multiple copies).	<input type="checkbox"/>

You must ensure that you retain a HARD COPY of the project , both for your own reference and in case a project is lost or mislaid. It is not sufficient to keep a copy on computer.	<input type="checkbox"/>
--	--------------------------

Assignments that are submitted to the Programme Coordinator Office must be placed into the assignment box located outside the office.

Office Use Only	
Signature:	
Date:	
Penalty Applied (if applicable):	

Configuration Manual

Aidan Browne
Student ID: 16140818

1 Introduction

The Configuration Manual describes the steps taken to successfully produce the results achieved by the 6 models.

2 Hardware Specification

The research was carried out on a Lenovo T440 laptop with the following device and windows operating system specification.

Device specifications	
Device name	DESKTOP-NH7BU6V
Processor	Intel(R) Core(TM) i5-4300U CPU @ 1.90GHz 2.49 GHz
Installed RAM	8.00 GB (7.69 GB usable)
Device ID	055B10A8-4220-4889-A257-FE71117CCE4B
Product ID	00331-20020-00000-AA903
System type	64-bit operating system, x64-based processor
Pen and touch	Touch support with 2 touch points

Figure 1: Laptop Specifications

Windows specifications	
Edition	Windows 10 Pro
Version	1903
Installed on	29/08/2019
OS build	18362.959

Figure 2: Windows Specification

3 Software and Data Installation

To successfully implement the objectives of the project the following data, packages and software were installed on the candidate’s laptop.

3.1 Data Selection

For the purpose of the research undertaken data was downloaded from the Yelp Open Data website¹ to candidate’s local drive. The data is provided in Java Script Object Notation (JSON) files and covers a subset of Yelp’s business, review and user data. The dataset is provided for academic or non-commercial purposes and was updated on the 21st February 2020. Prior to February 2020 the data was used as part of the Yelp Data Challenge with the 13th round finishing on 31st December 2019. The challenge was an annual event for students with a top prize of \$5,000. Compressed the dataset contains over 4 gigabytes (GB) of data and over 9 GB uncompressed. There were 5 JSON files contained in the file but for the research carried out only the Business, Review and User were used. Each Business, Review and User had a unique identifier which enables the files to be merged when creating the two datasets. The period the data covered was between 20/01/2007 – 31/01/2019. Numerical attributes such as review count or review star rating can be used as features for machine learning algorithms. Non-numerical attributes such as review text or list of friends were furthered processed to be able to be used for the research. The business dataset contains features such as location, attributes and category and covered 10 metropolitan areas. The review dataset contains review text, star rating and votes received if a review is useful, funny and cool. The user dataset had attributes pertaining to the user’s social network and all the user’s metadata. A description of the three JSON files can be found in below table

Table 1: Yelp Open Dataset Description

Dataset	Record Count	Attribute Count
Business	209,393	14
Review	8,021,222	9
User	1,968,703	22

The followings steps were taken to download the JSON files.

¹ <https://www.yelp.com/dataset>

1. Enter the highlighted information to agree to the Yelp Open Dataset license agreement before downloading

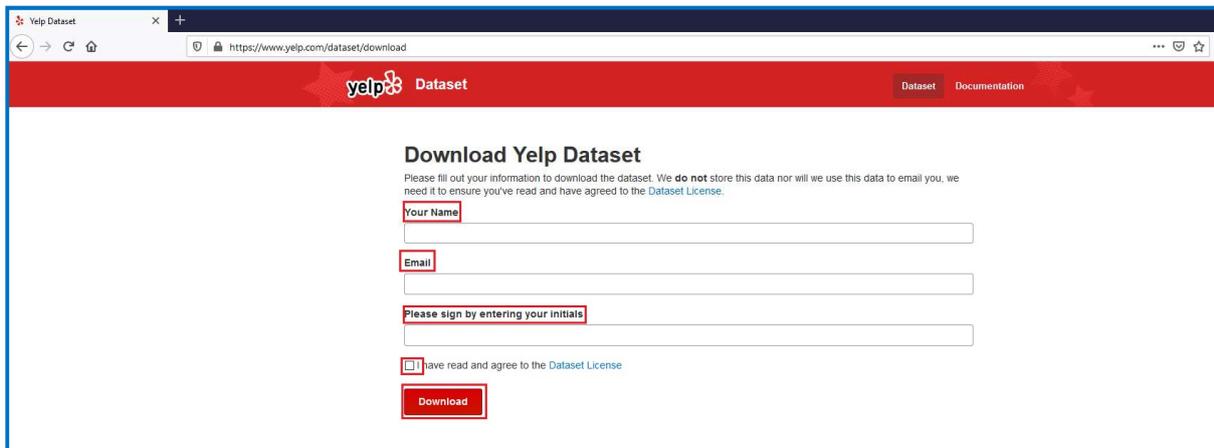


Figure 3: Yelp Open Dataset License

3.2 Azure Databricks

Azure Databricks was chosen to create the two datasets as it offers distributing computing provided by Databricks combined with Microsoft Azure's cloud storage capabilities. This was essential as the total size of the three JSON files used as part of research project totalled 9.6 gigabytes (GB). The largest of these was the Review (6.2GB) which could not be processed by the candidate's laptop. The following steps were followed to set up an Azure Databricks account, create storage containers for the Business, Review and User JSON files and create an Azure Key Vault to access the Azure Blob container from Databricks.

1. Set up an account with your college credentials via the Azure website²

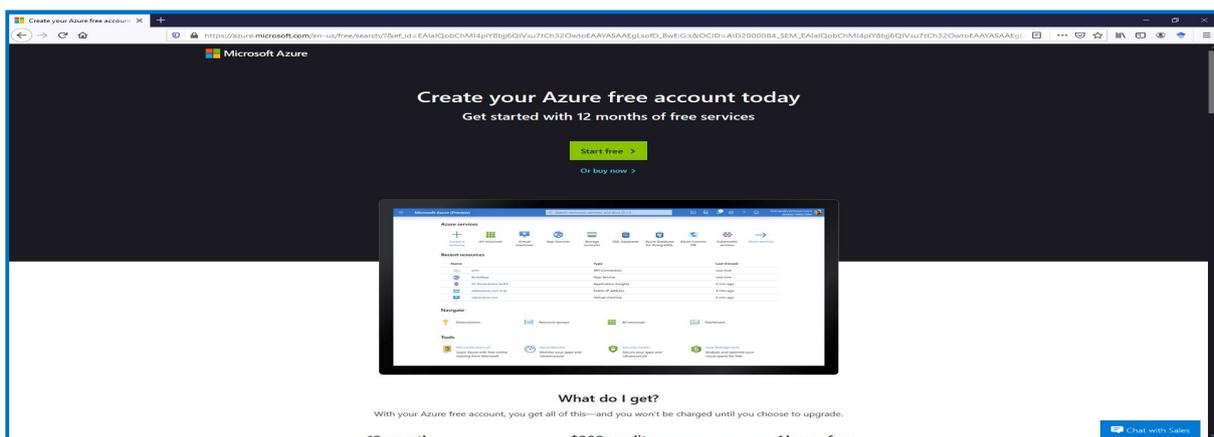


Figure 4: Azure Set UP

² <https://azure.microsoft.com/en-gb/free/>

2. After you have created a free Student Account to unlock the necessary products to be used in the project a Pay-As-You-Go (PAYG) subscriptions needs to be added to the account from the offers page³.

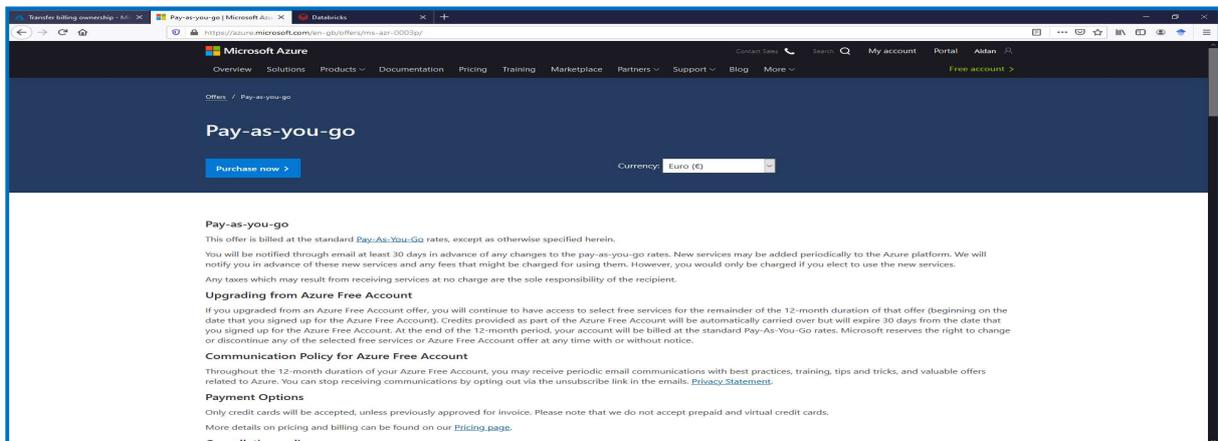


Figure 5: Pay-As-You-Go Subscriptions (1 of 2)

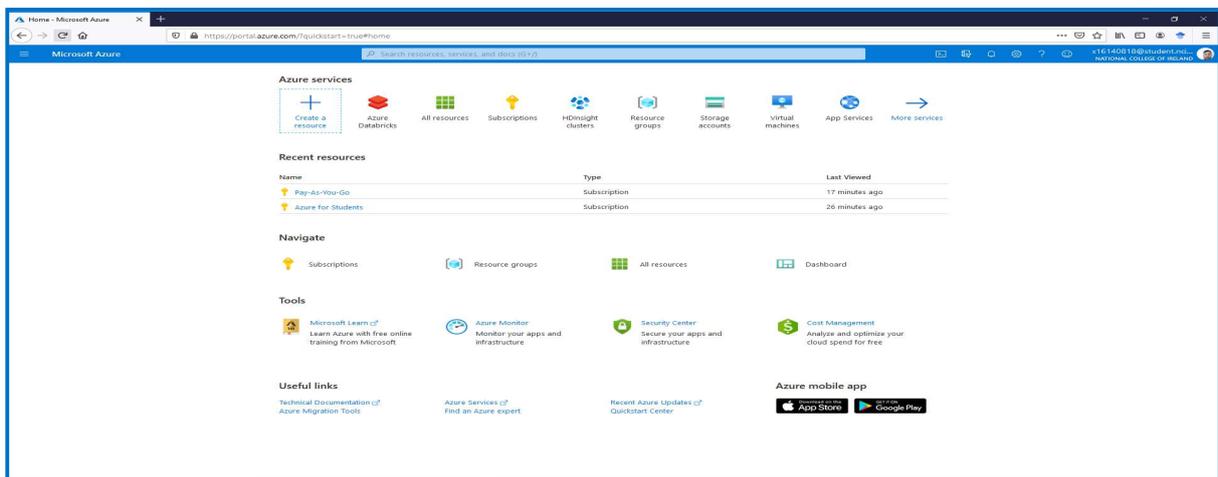


Figure 6: Pay-As-You-Go Subscriptions (2 of 2)

3. After the PAYG subscription has been added a Databricks resource needs to be added to your subscription. From the main portal page click Azure Databricks

³ <https://azure.microsoft.com/en-us/offers/ms-azr-0003p/>

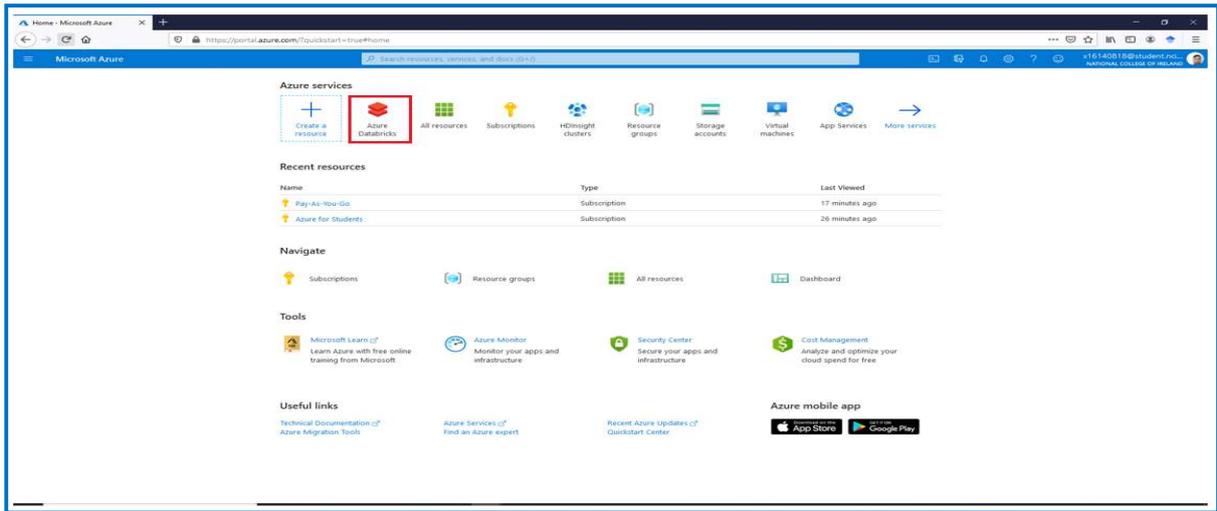


Figure 7: Databricks Set-up

4. Click Add

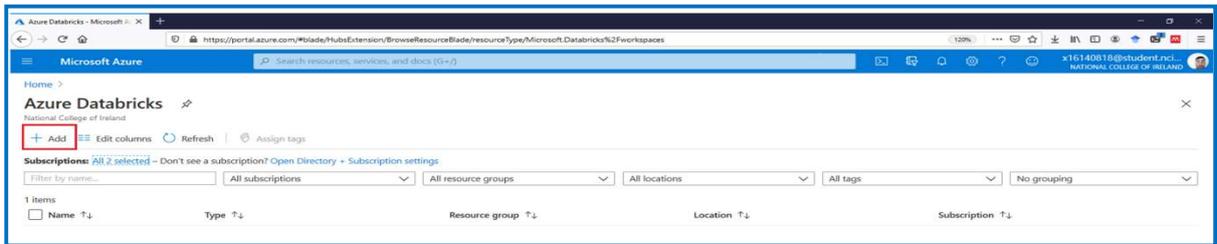


Figure 8: Databricks Set-up

5. Update below information creating new Resource Group, Workspace name, choosing appropriate location and choosing Pricing Tier. When updated click Review + Create

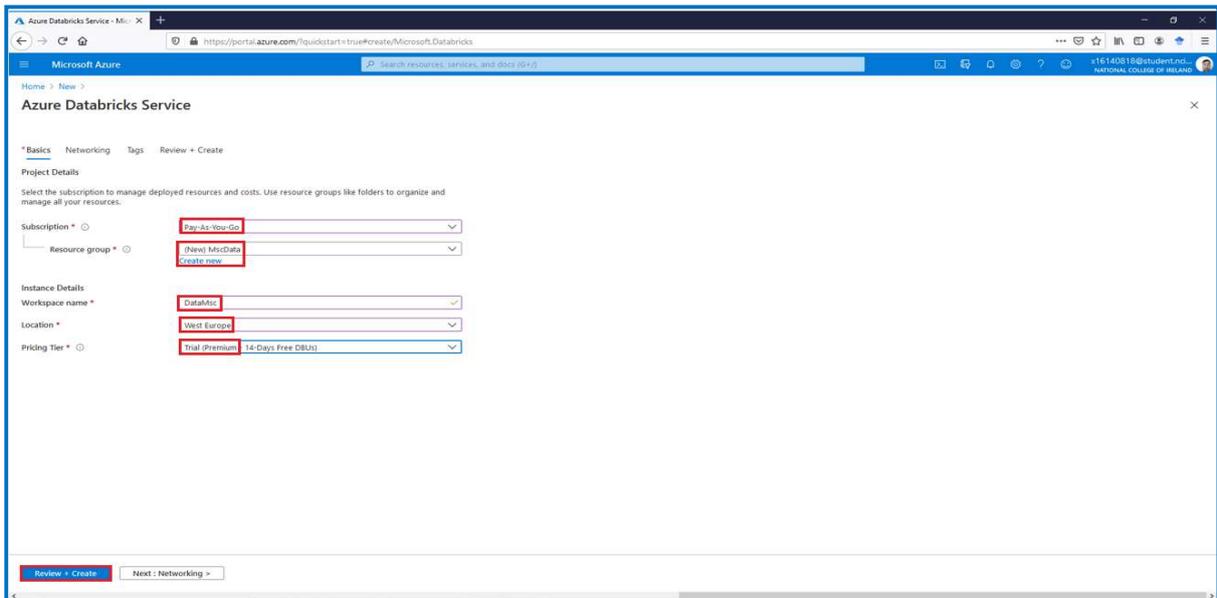


Figure 9: Databricks Set-up

6. If happy with details, click Create

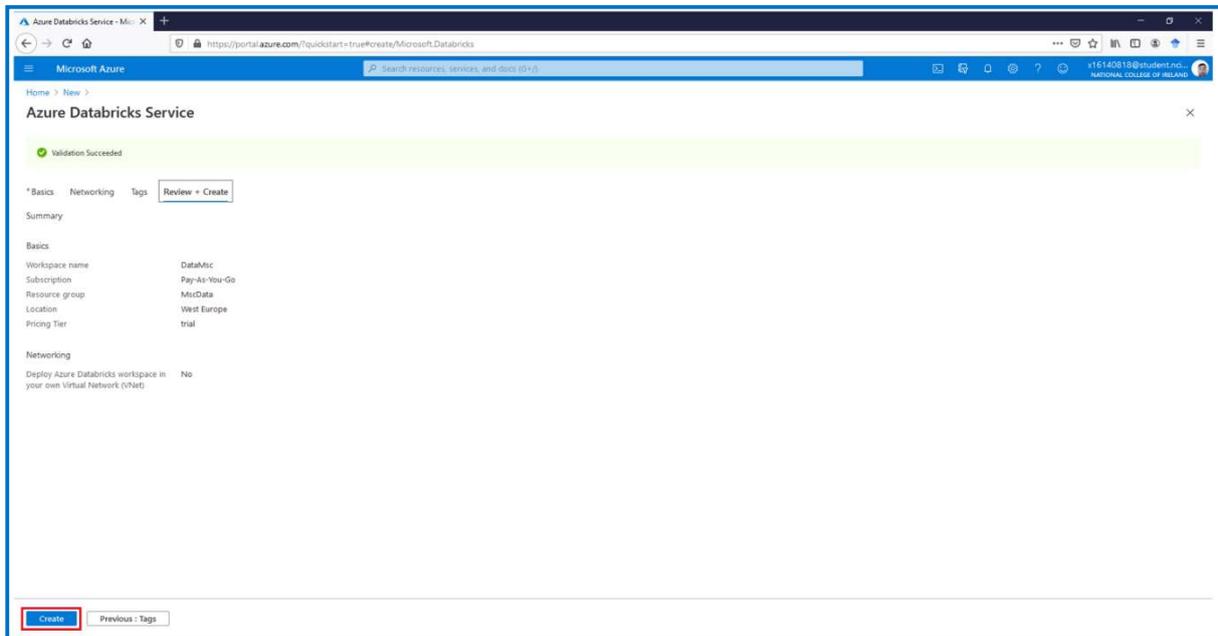


Figure 10: Databricks Set-up

7. If the resource is created successfully you will be brought to below page. Click on Home to return to main page.

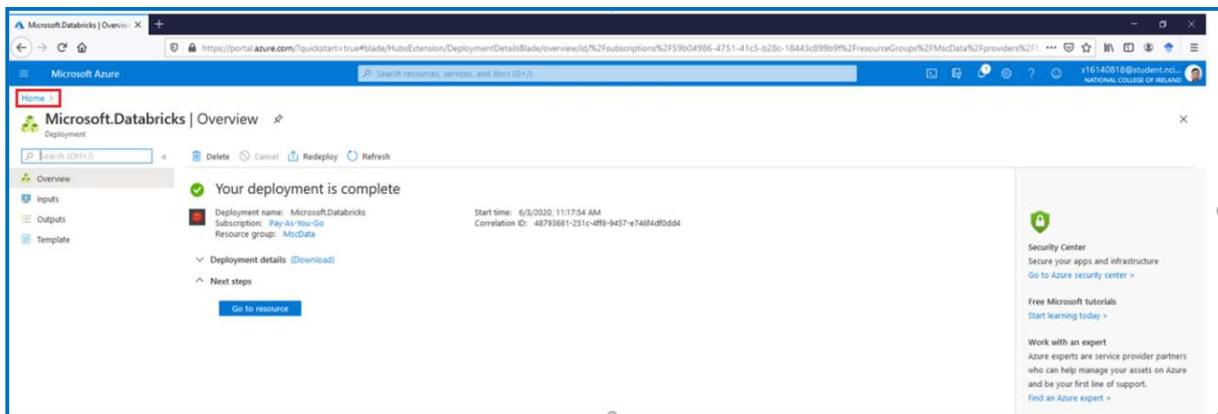


Figure 11: Databricks Set-up

8. Next a Storage Container resource, JSON files and Key Vault need to be added. Return to Home page and click on Storage accounts

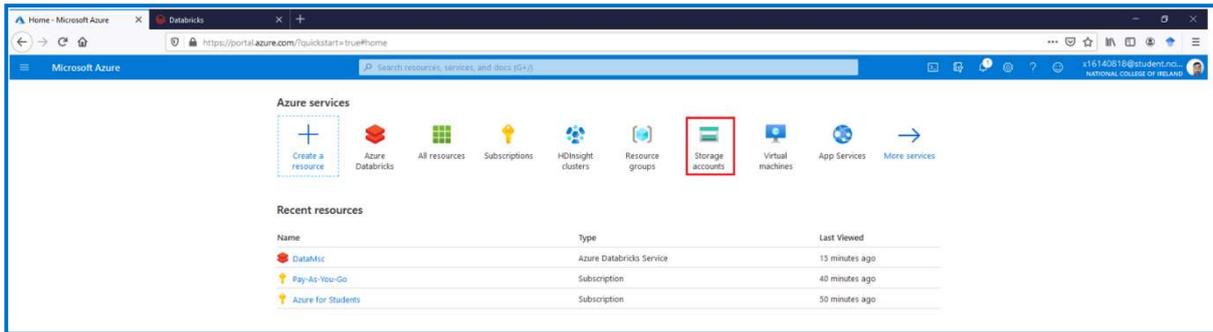


Figure 12: Storage Account Set-up

9. Click Add

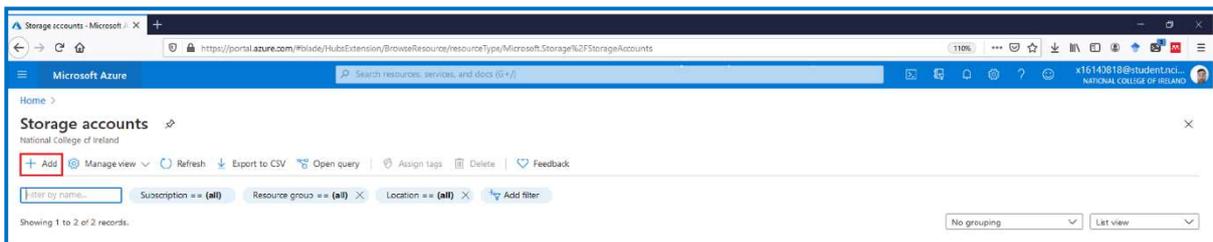


Figure 13: Storage Account Set-up

10. Update below information ensuring Subscription, Resource group and Location matches the Databricks resource. When updated click Review + create

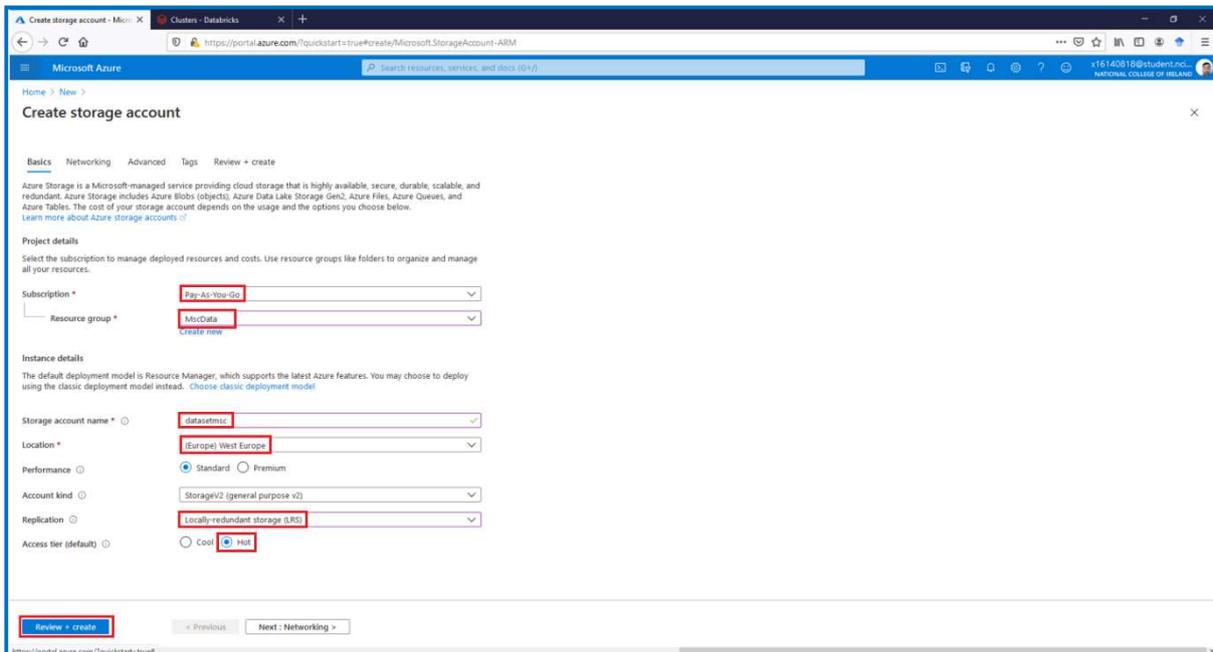


Figure 14: Storage Account Set-up

11. Click Create

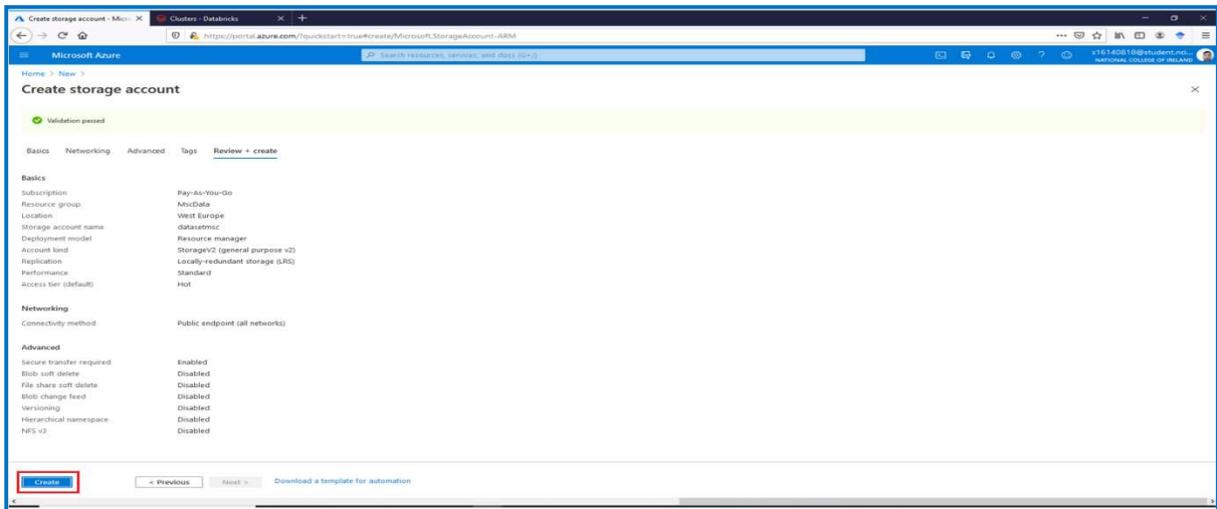


Figure 15: Storage Account Set-up

12. When deployment is complete click Go to Resource to add the JSON files to Azure storage.

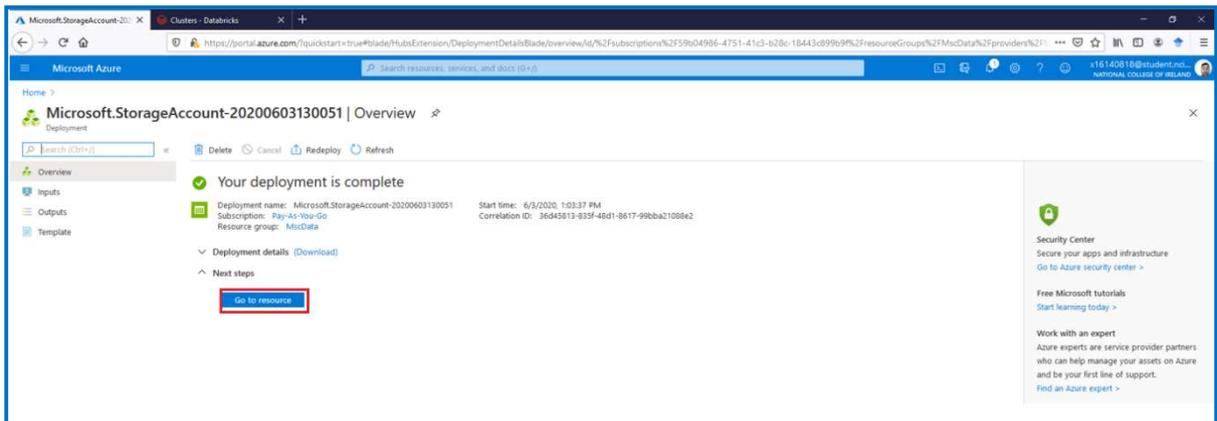


Figure 16: JSON Files Upload

13. When in the Storage account click on Storage Explorer

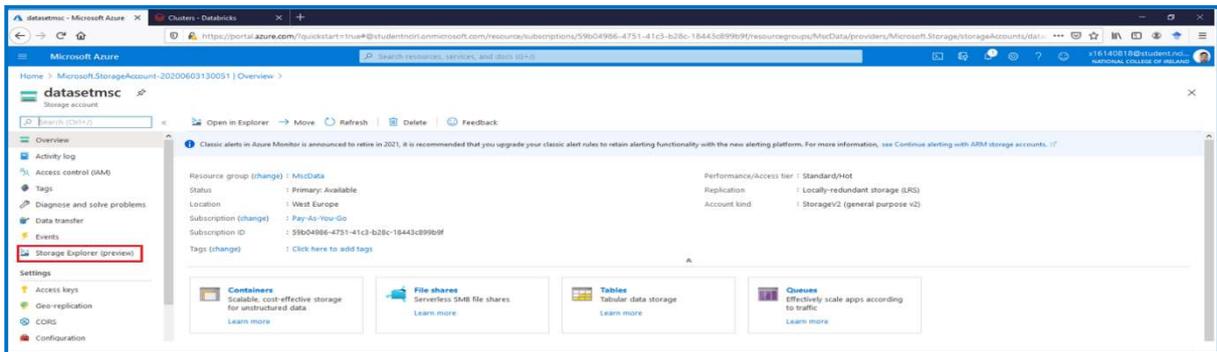


Figure 17: JSON Files Upload

1. Right click on Blob Containers to create the 3 Blobs to hold the Business, Review and Users data

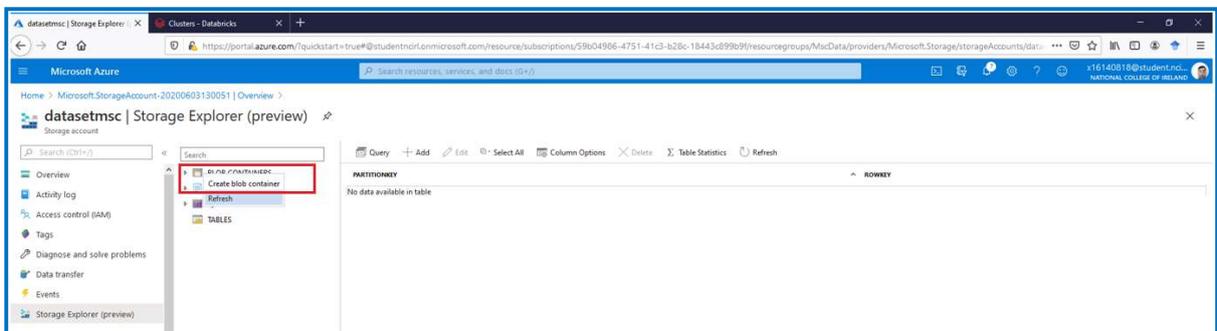


Figure 18: JSON Files Upload

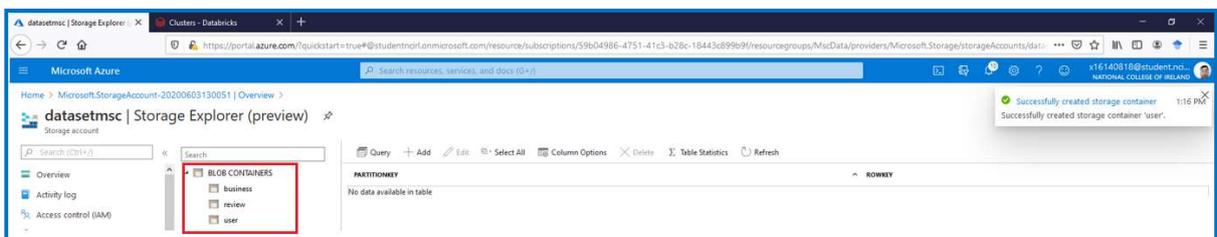


Figure 19: JSON Files Upload

14. Click on each container and then click upload

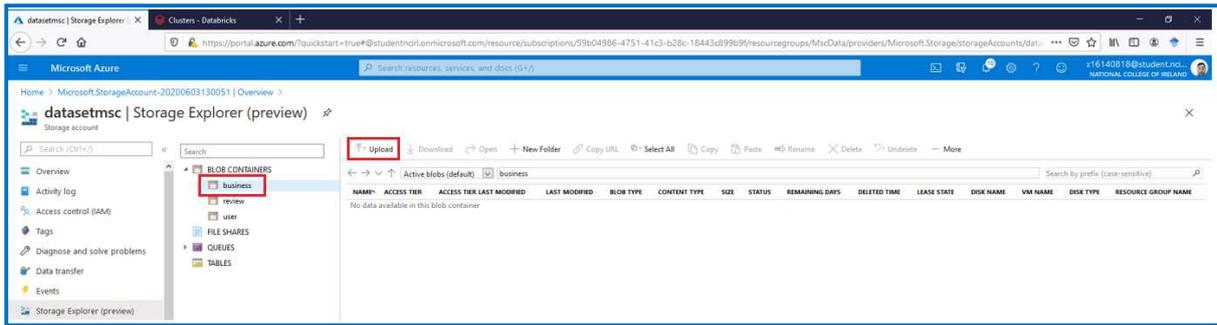


Figure 20: JSON Files Upload

15. Select the corresponding JSON file for each container downloaded from the Yelp Open Dataset website. Once the JSON files have been uploaded click on Access Keys

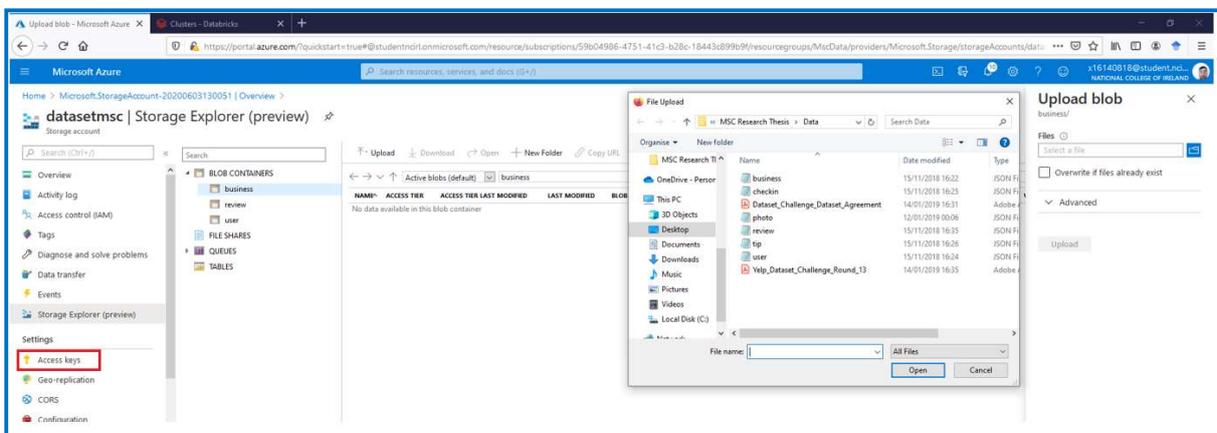


Figure 21: JSON Files Upload

16. Copy Storage account name and Key 1 to notepad or word. When complete return to the homepage

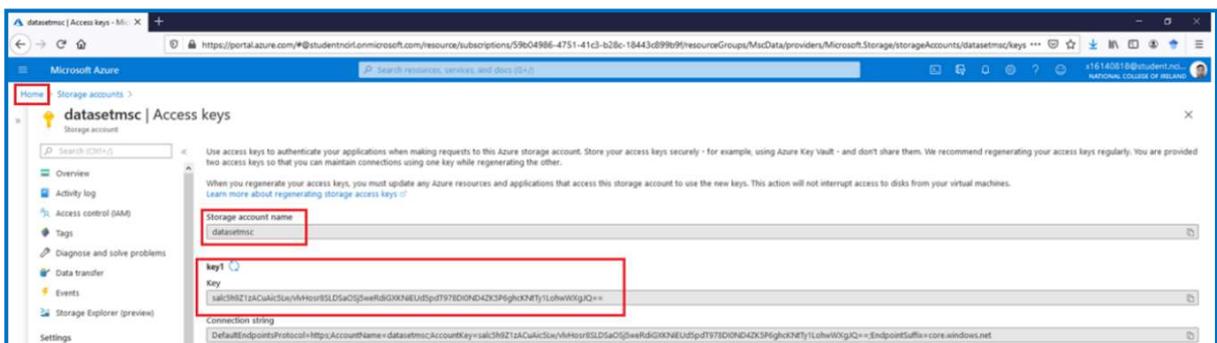


Figure 22: Key Vault and Secret (1 of 17)

17. Next an Azure Key Vault and a secret need to be created. On the homepage click on Create Resource

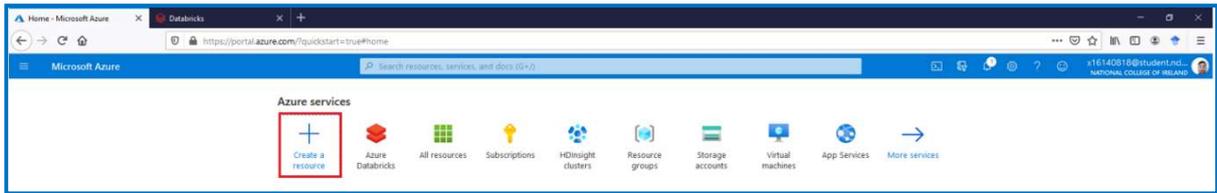


Figure 23: Key Vault and Secret (2 of 17)

18. Search for Key Vault, select Key Vault resource and click create

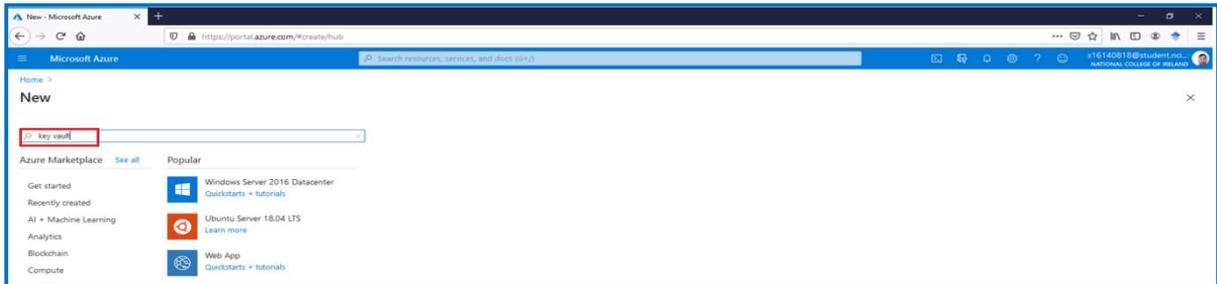


Figure 24: Key Vault and Secret (3 of 17)

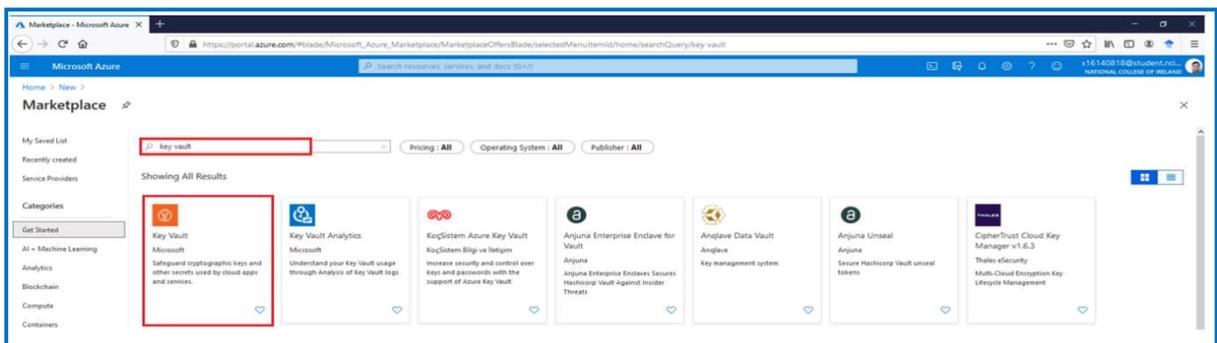


Figure 25: Key Vault and Secret (4 of 17)

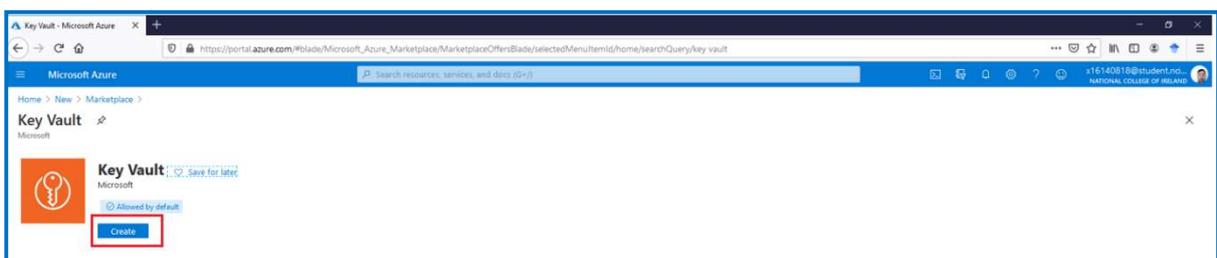


Figure 26: Key Vault and Secret (5 of 17)

19. Enter the following information on the Create Key Vault page again ensuring Resource group and Region are the same as storage account. When complete click Review + create

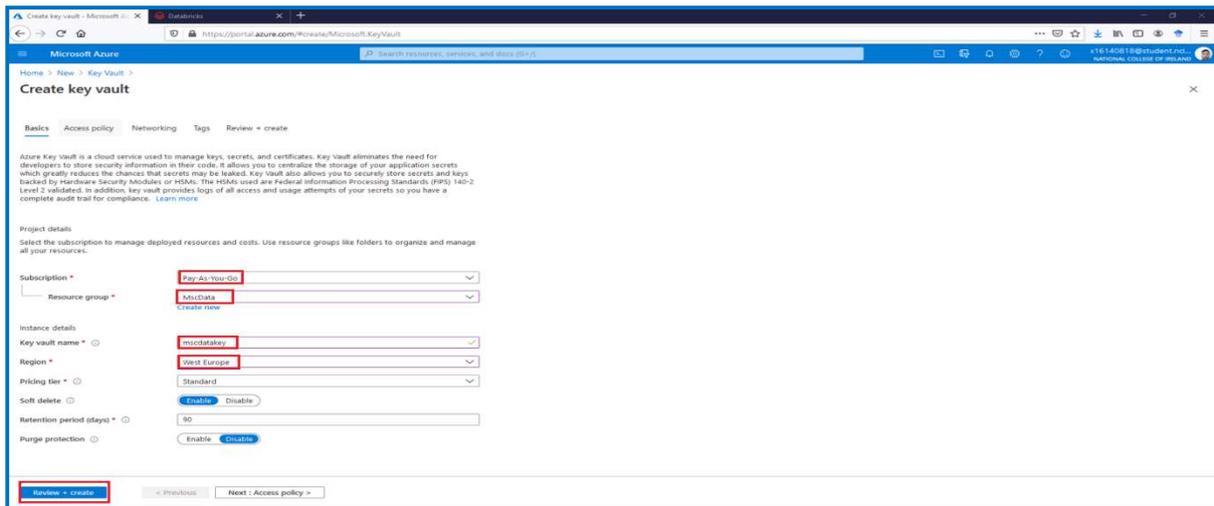


Figure 27: Key Vault and Secret (6 of 17)

20. When validation has passed click Create

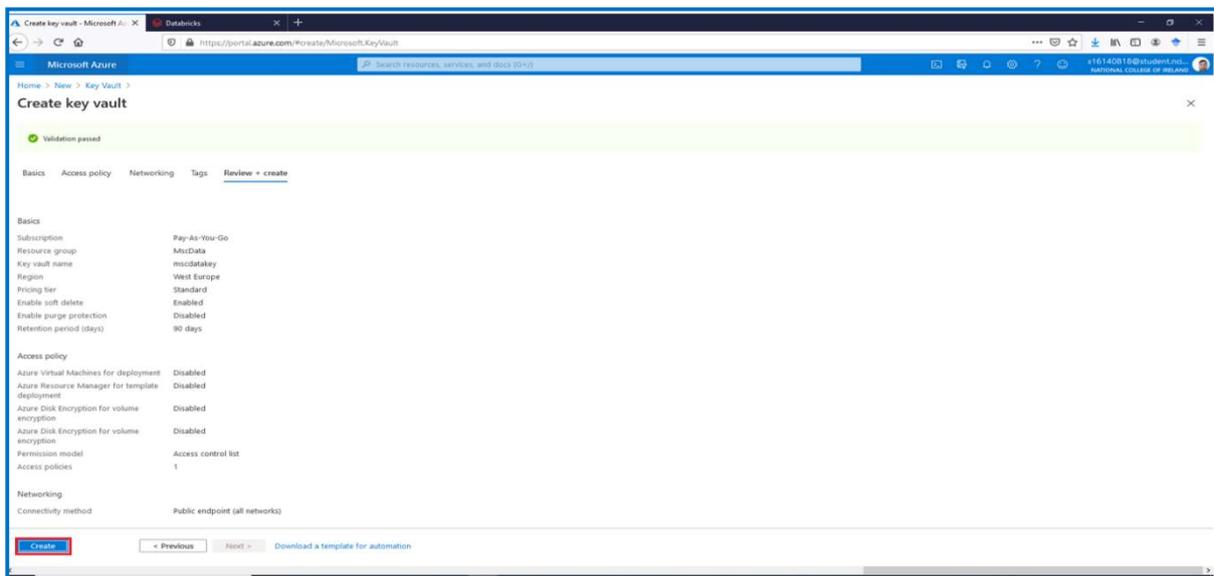


Figure 28: Key Vault and Secret (7 of 17)

21. When the Key Vault has been created navigate to the overview page and click Secrets

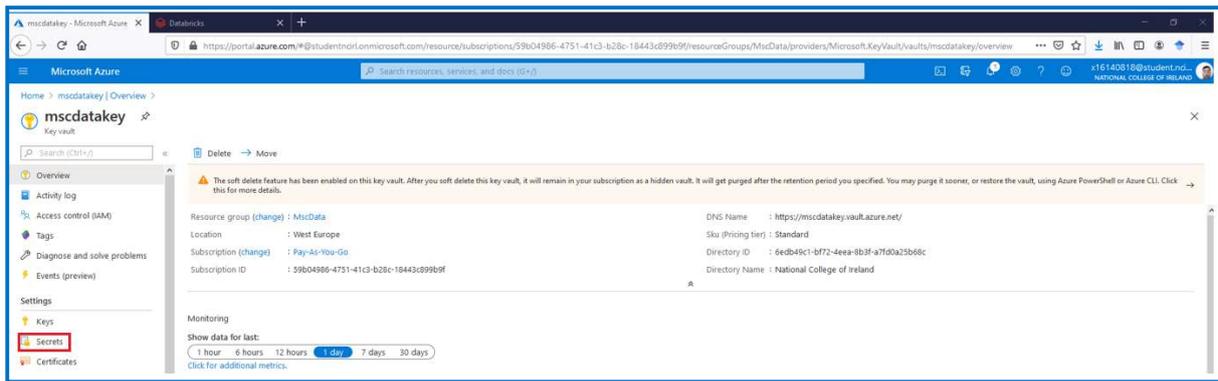


Figure 29: Key Vault and Secret (8 of 17)

22. Click Generate/Import

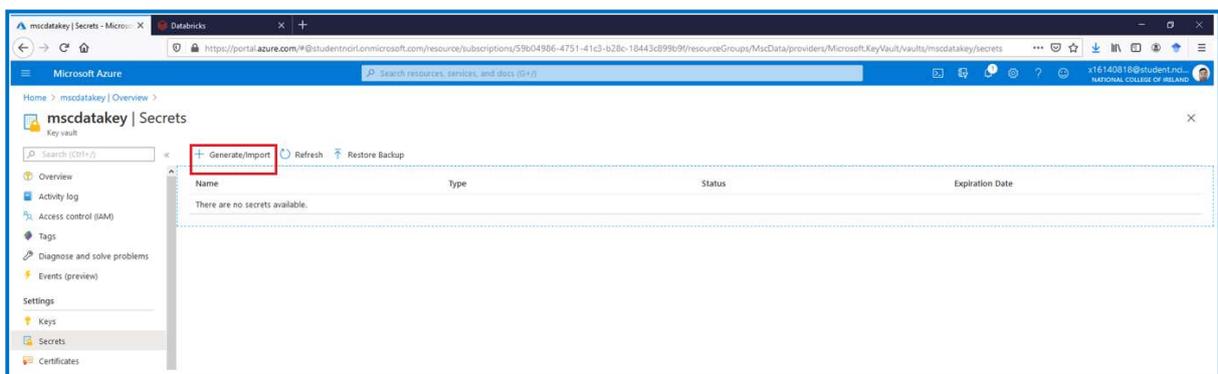


Figure 30: Key Vault and Secret (9 of 17)

23. Update below information. For Value paste Key 1 from Access Keys for Storage account previously copied to notepad. At this point also save the Key name to notepad. Finally click create

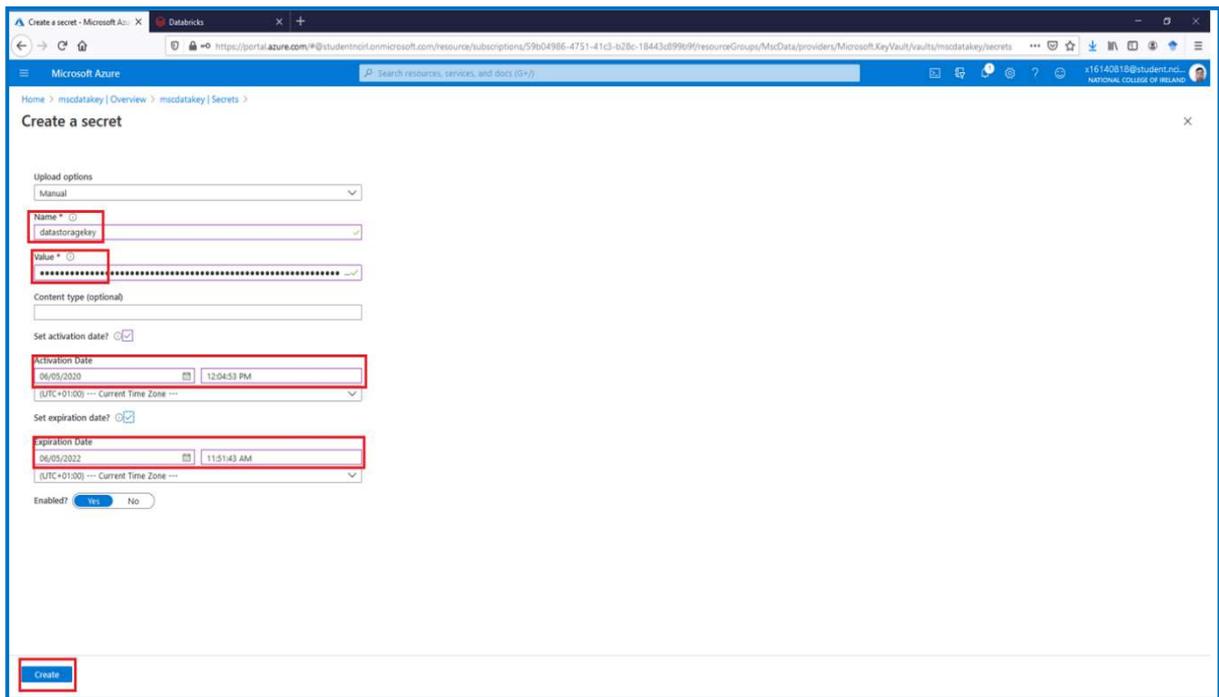


Figure 31: Key Vault and Secret (10 of 17)

24. After the secret has been created navigate to the properties of the Key Vault

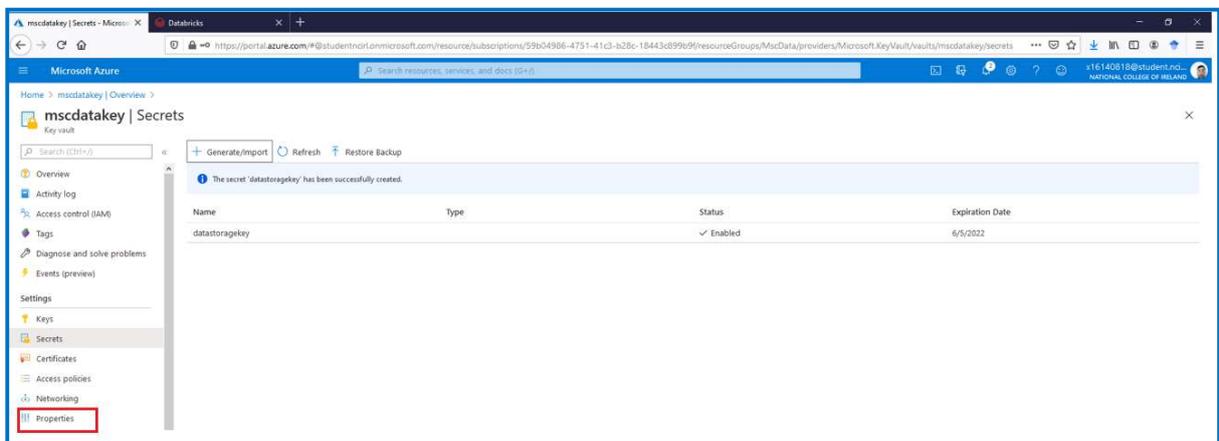


Figure 32: Key Vault and Secret (11 of 17)

25. Copy DNS Name & Resource ID to notepad and return to homepage when finished.

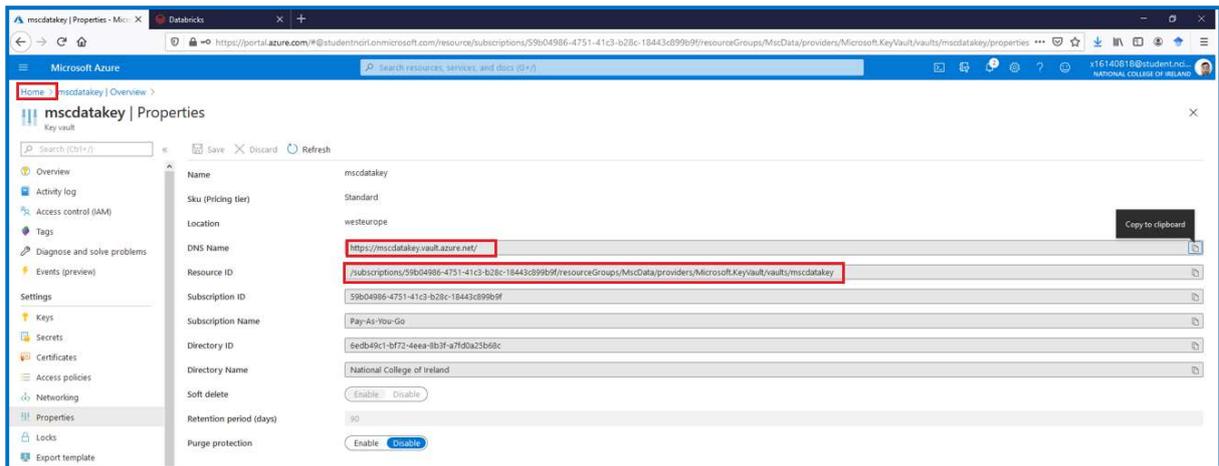


Figure 33: Key Vault and Secret (12 of 17)

26. On the homepage click on the previously created Databricks resource

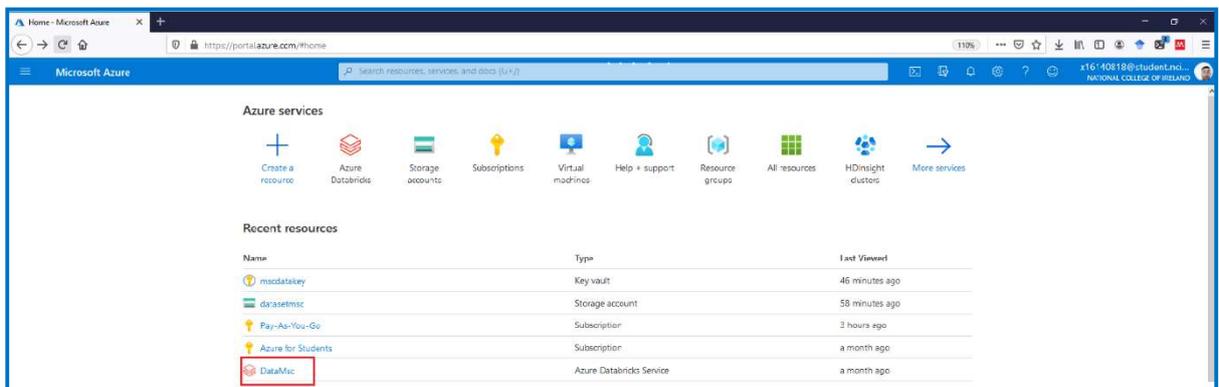


Figure 34: Key Vault and Secret (13 of 17)

27. Click Launch Workspace

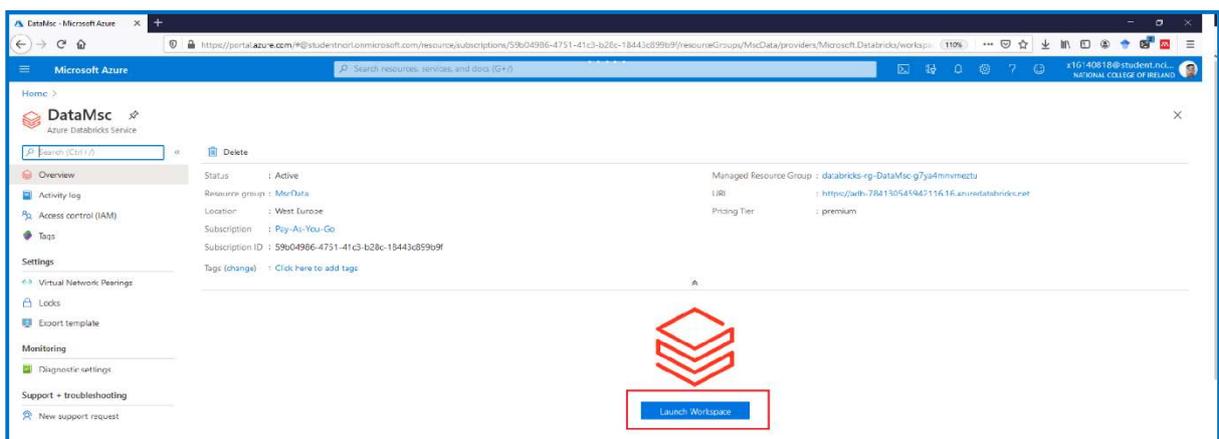


Figure 35: Key Vault and Secret (14 of 17)

28. When the workspace has opened copy the URL and open a new tab.

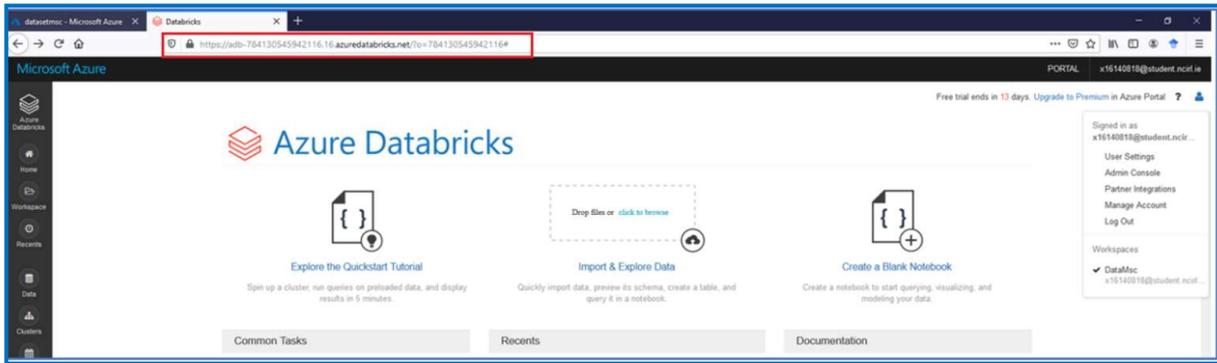


Figure 36: Key Vault and Secret (15 of 17)

29. In the new tab paste the URL copied previously and add #secrets/createScope at the end so that it is in the following format
 https://<location>.azuredatabricks.net/?o=<orgID>#secrets/createScope

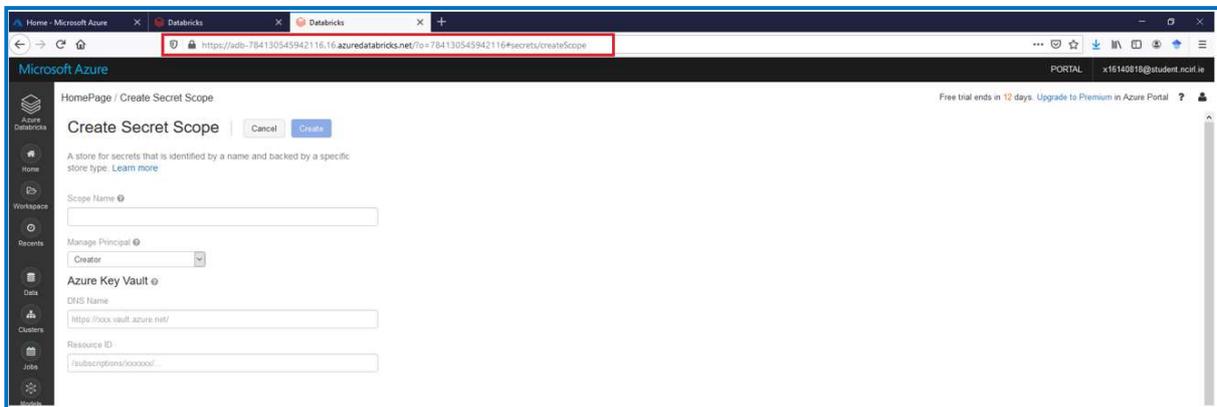


Figure 37: Key Vault and Secret (16 of 17)

30. Update the below information. DNS Name and Resource ID have been previously copied to notepad. When all information has been entered click Create. Azure Databricks has now been successfully configured.

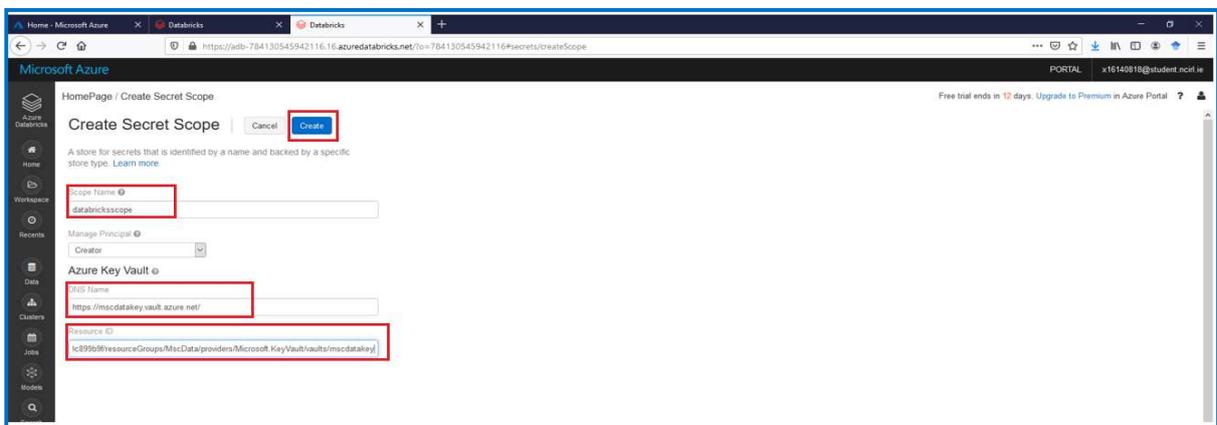


Figure 38: Key Vault and Secret (17 of 17)

3.3 Google

1. A Google account can be set up at Account Setup page⁴. By setting up the account the user will have access to products such as Google Colaboratory (Colab), Google Drive (Drive) and Gmail.

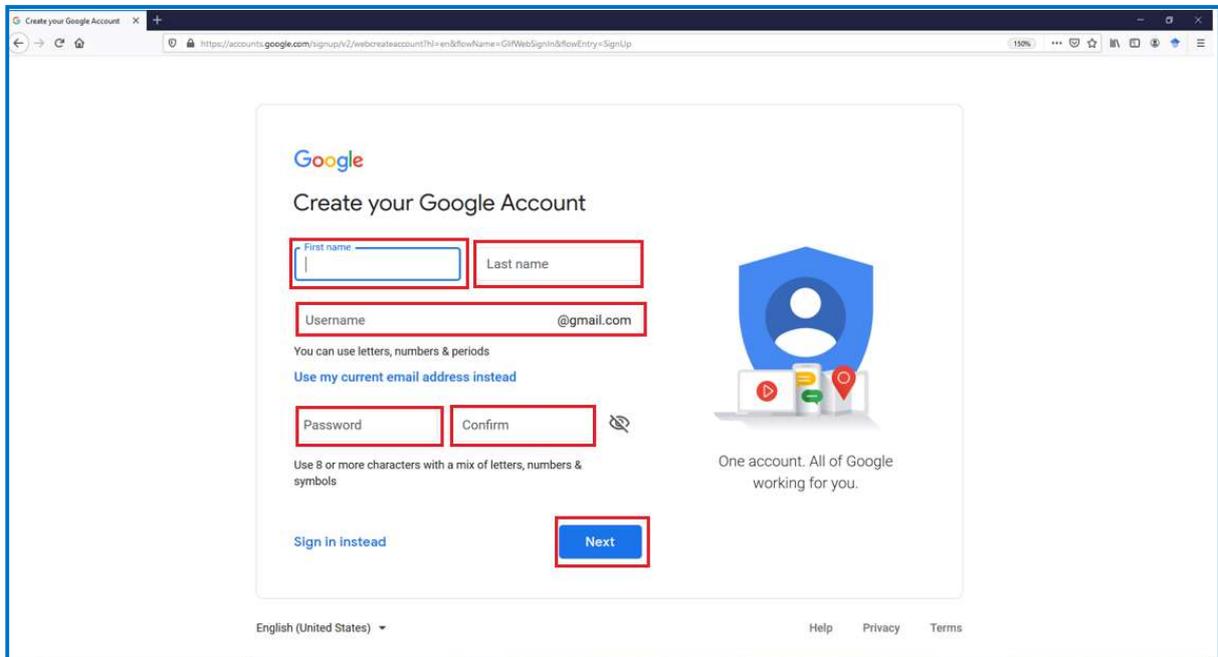


Figure 39: Google Account Set-up

3.4 Genesis Cloud

Genesis Cloud is a cloud Graphic Processing Unit (GPU) service that was required to run the models as the candidate's laptop had limited computational resources. An account can be set up via the Genesis Cloud website⁵. Once the account is created a debit or visa card needs to be added to be able to create virtual instance with the required GPU's. After a credit card has been added the user receives \$50 free credit.

4

<https://accounts.google.com/signup/v2/webcreateaccount?hl=en&flowName=GlifWebSignIn&flowEntry=SignUp>

⁵ <https://www.genesiscloud.com/>

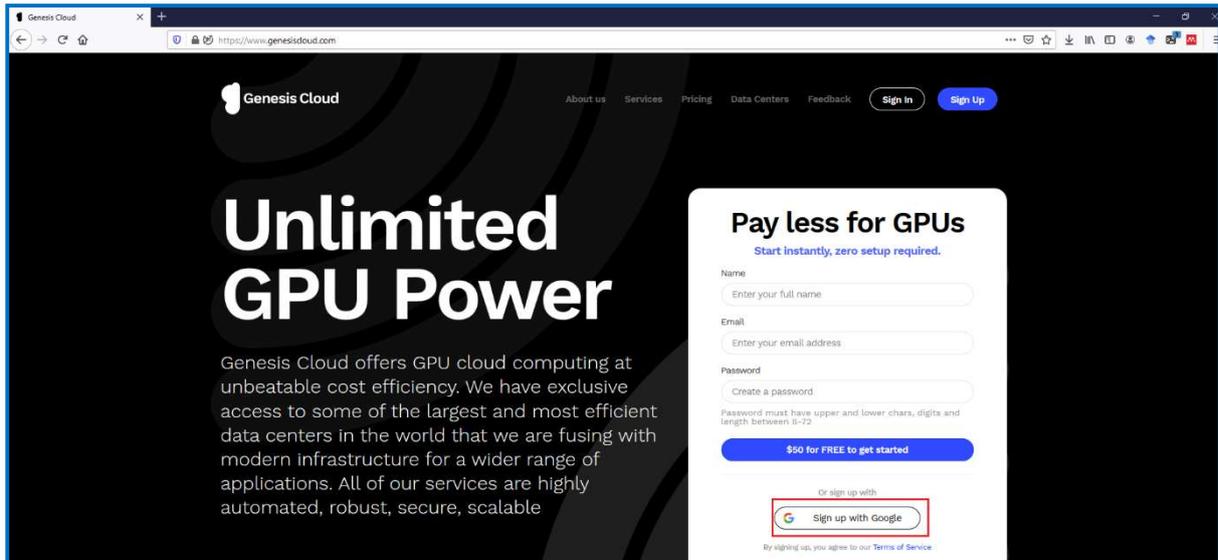


Figure 40: Genesis Cloud Account Set-up

3.5 Anaconda

Anaconda Individual is an open sourced Python distribution for machine learning that includes applications such as JupyterLab and Spyder. Anaconda can be downloaded from the Anaconda website⁶.



Figure 41: Anaconda Download (1 of 3)

⁶ <https://www.anaconda.com/products/individual>

1. Scroll down the page and select python 3.7 64-bit Graphical Installer for Windows

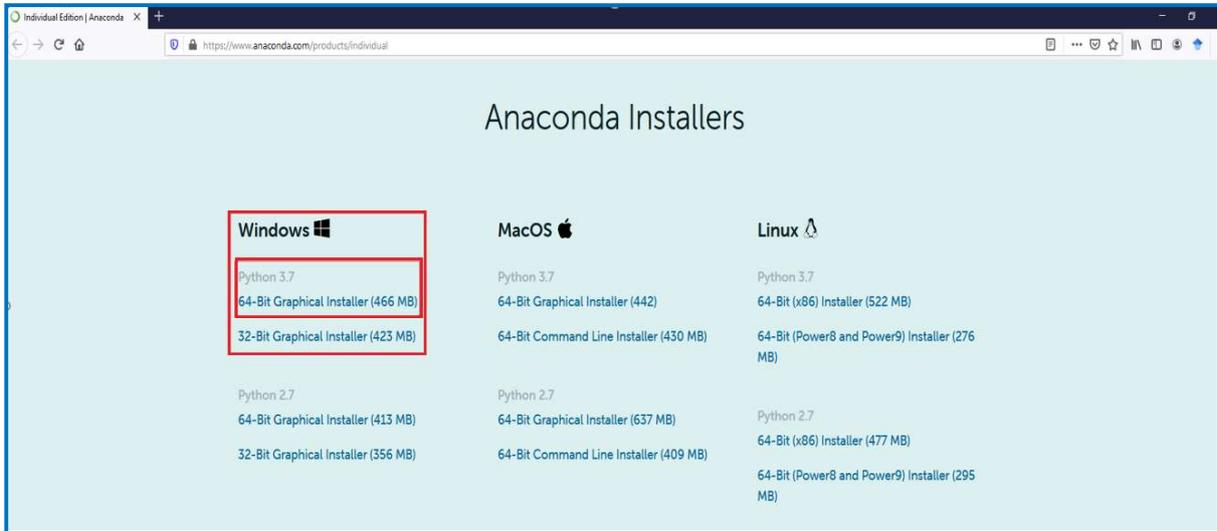


Figure 42: Anaconda Download (2 of 3)

2. Run the file after download.

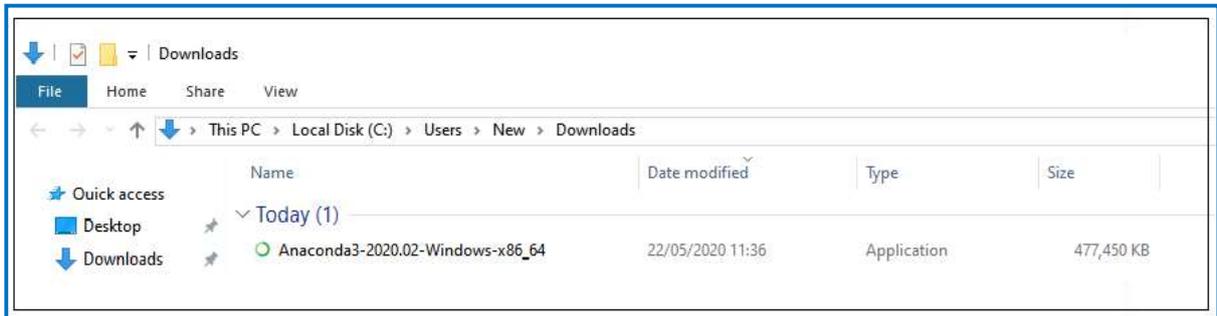


Figure 43: Anaconda Download (3 of 3)

3. After the file has opened click next

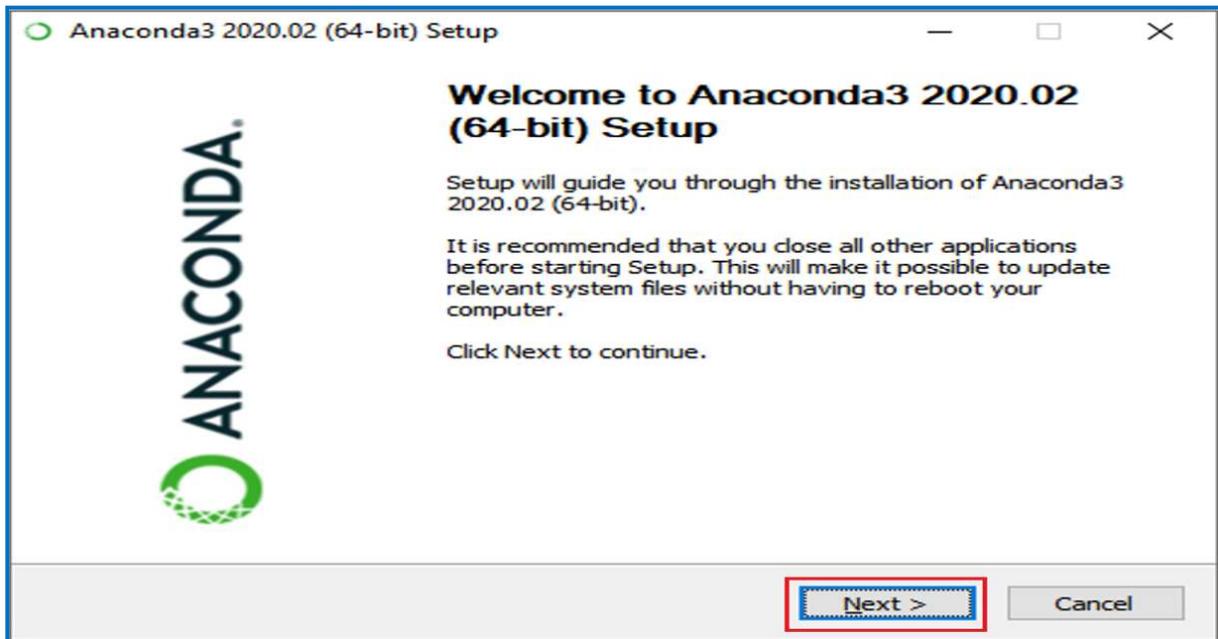


Figure 44: Anaconda Set-up (1 of 21)

4. Click I agree on the Licence Agreement

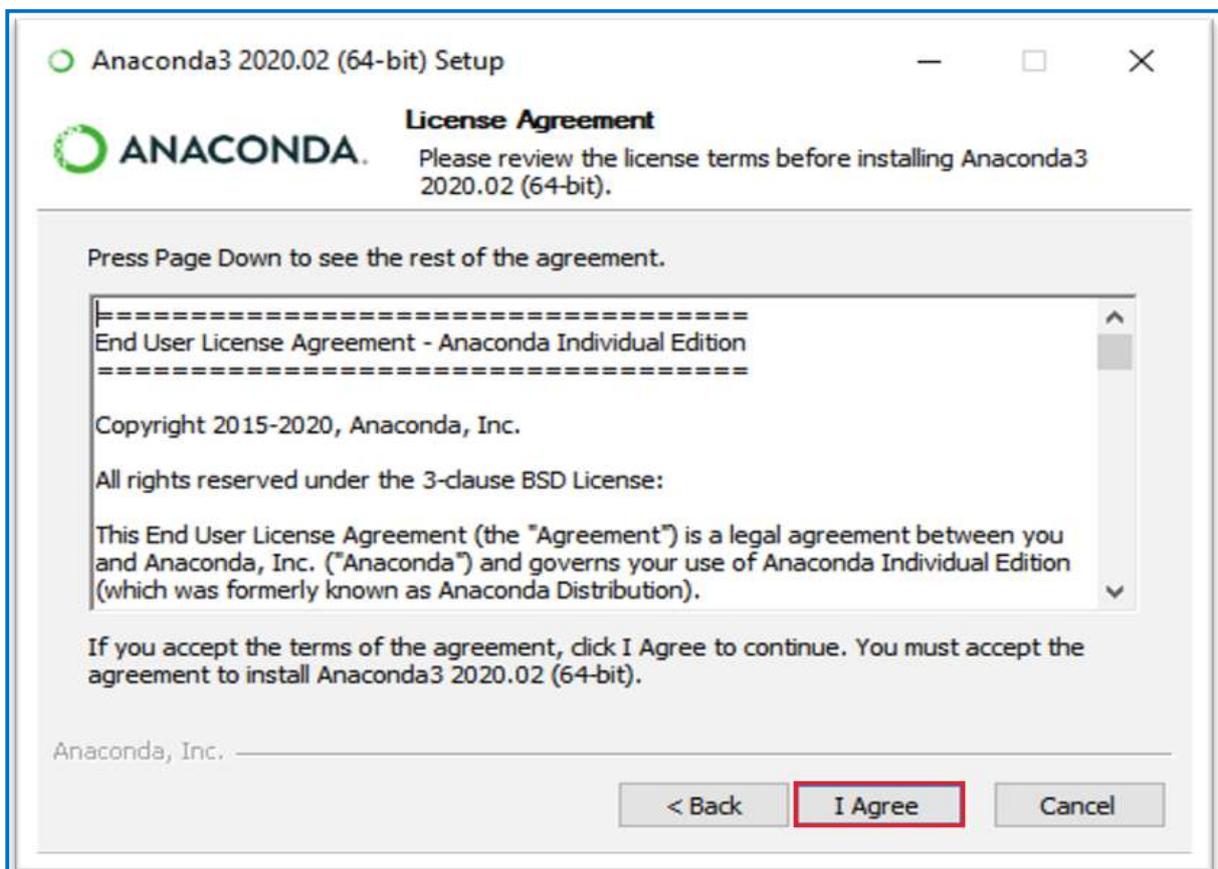


Figure 45: Anaconda Set-up (2 of 21)

5. Click Just Me and click next

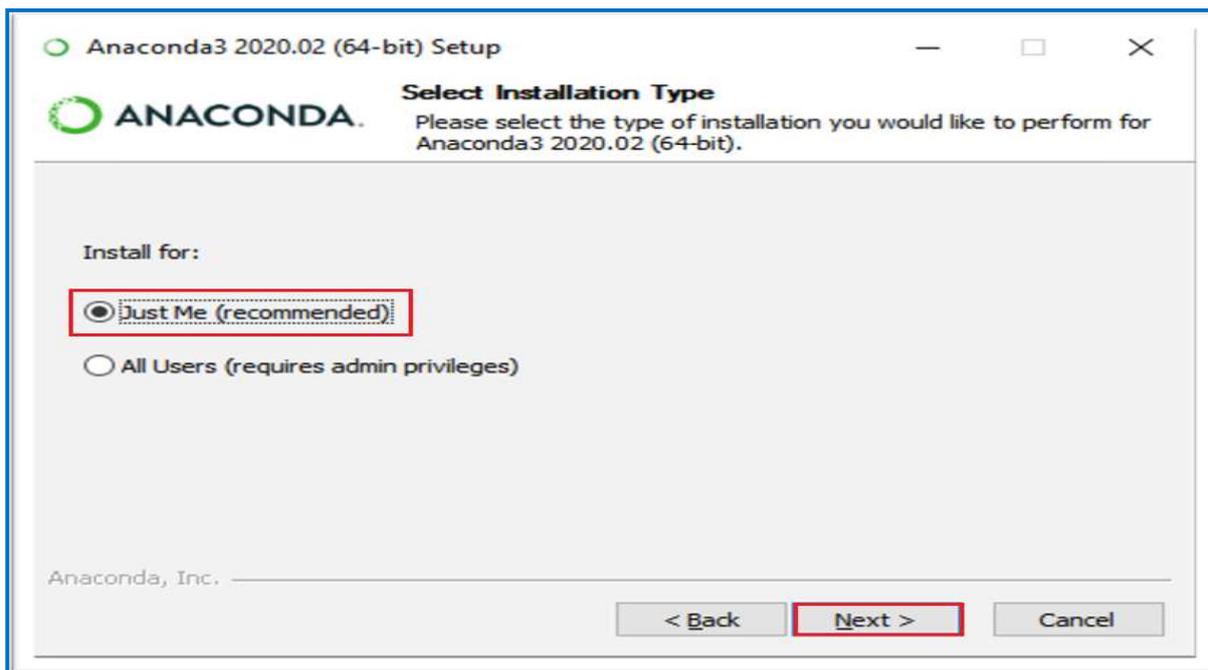


Figure 46: Anaconda Set-up (3 of 21)

6. Leave default destination folder and click next

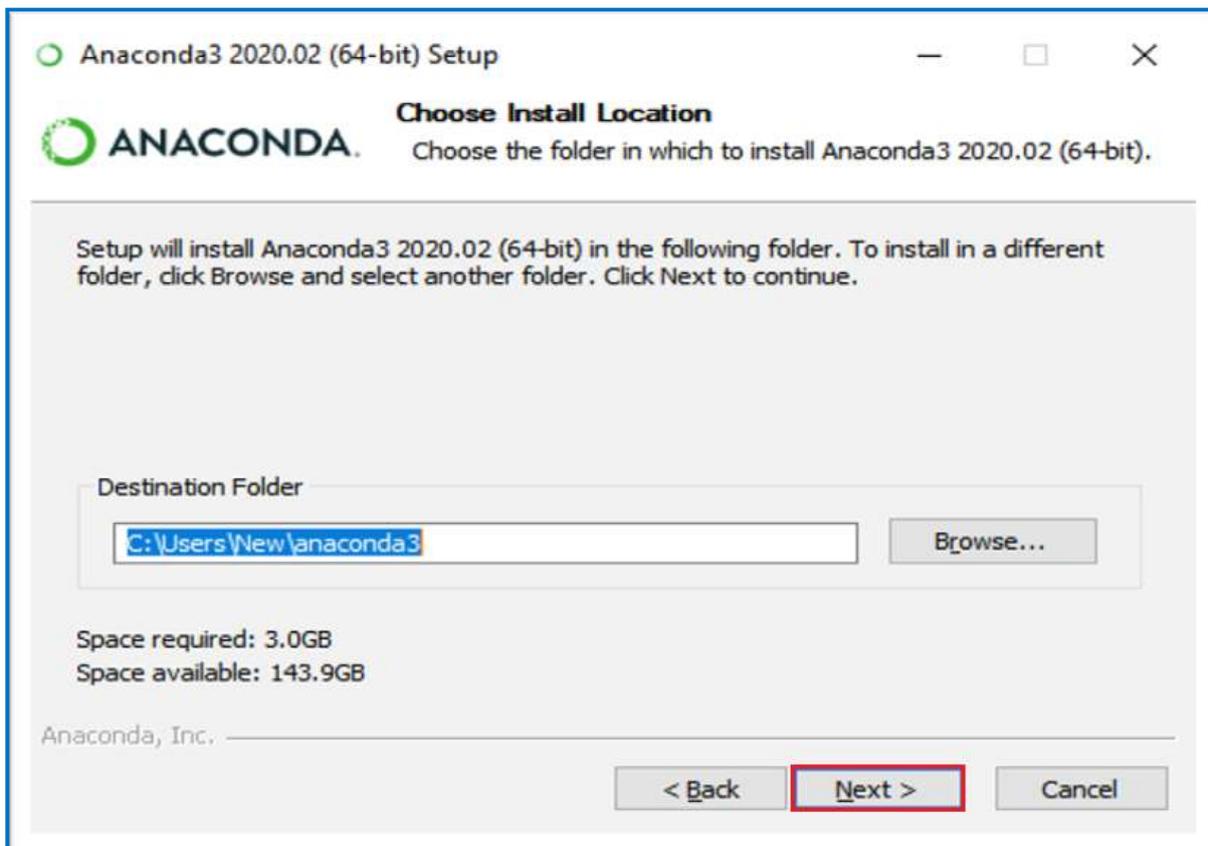


Figure 47: Anaconda Set-up (4 of 21)

7. It is highly recommended to choose Register Anaconda3 as my default Python 3.7. Click Install

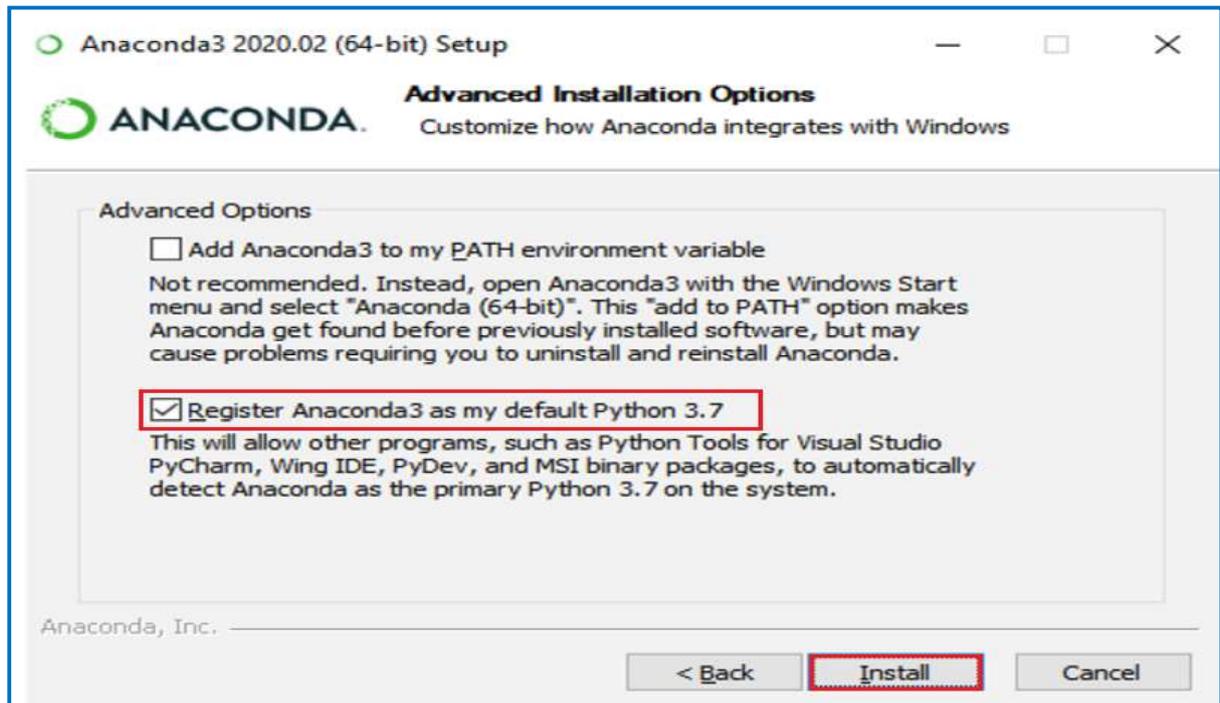


Figure 48: Anaconda Set-up (5 of 21)

8. Once completed click next

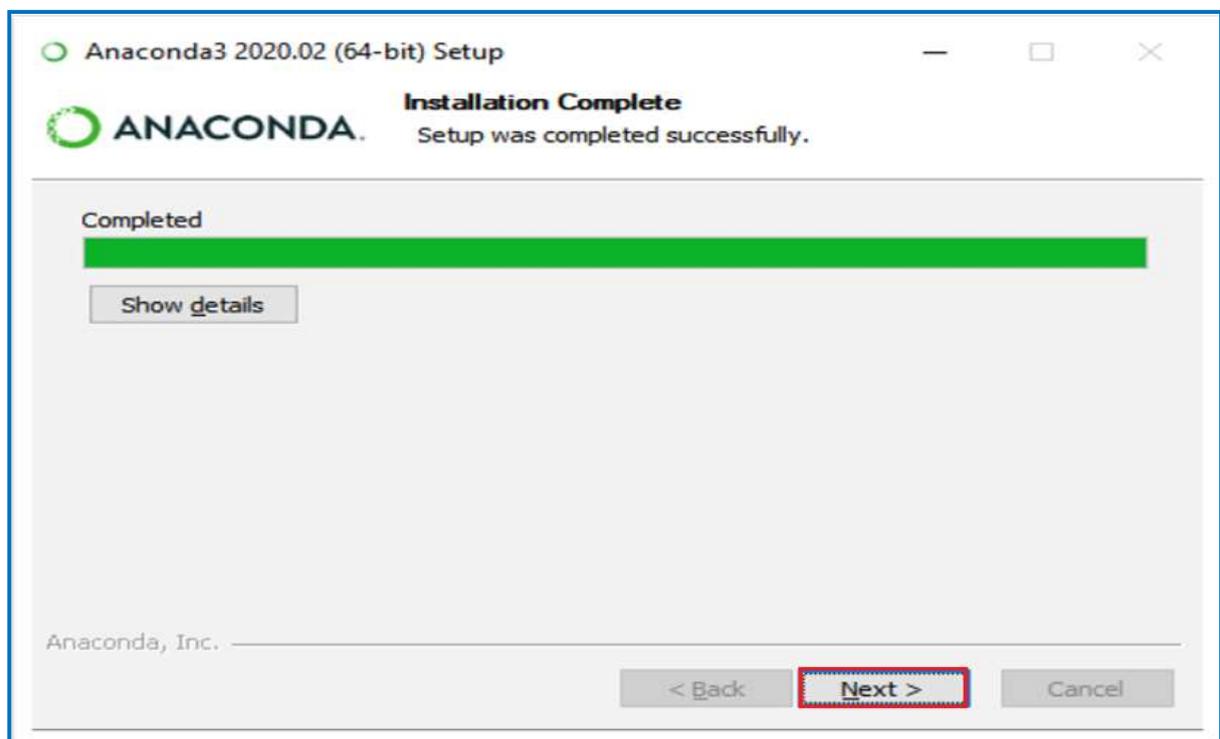


Figure 49: Anaconda Set-up (6 of 21)

9. And next again

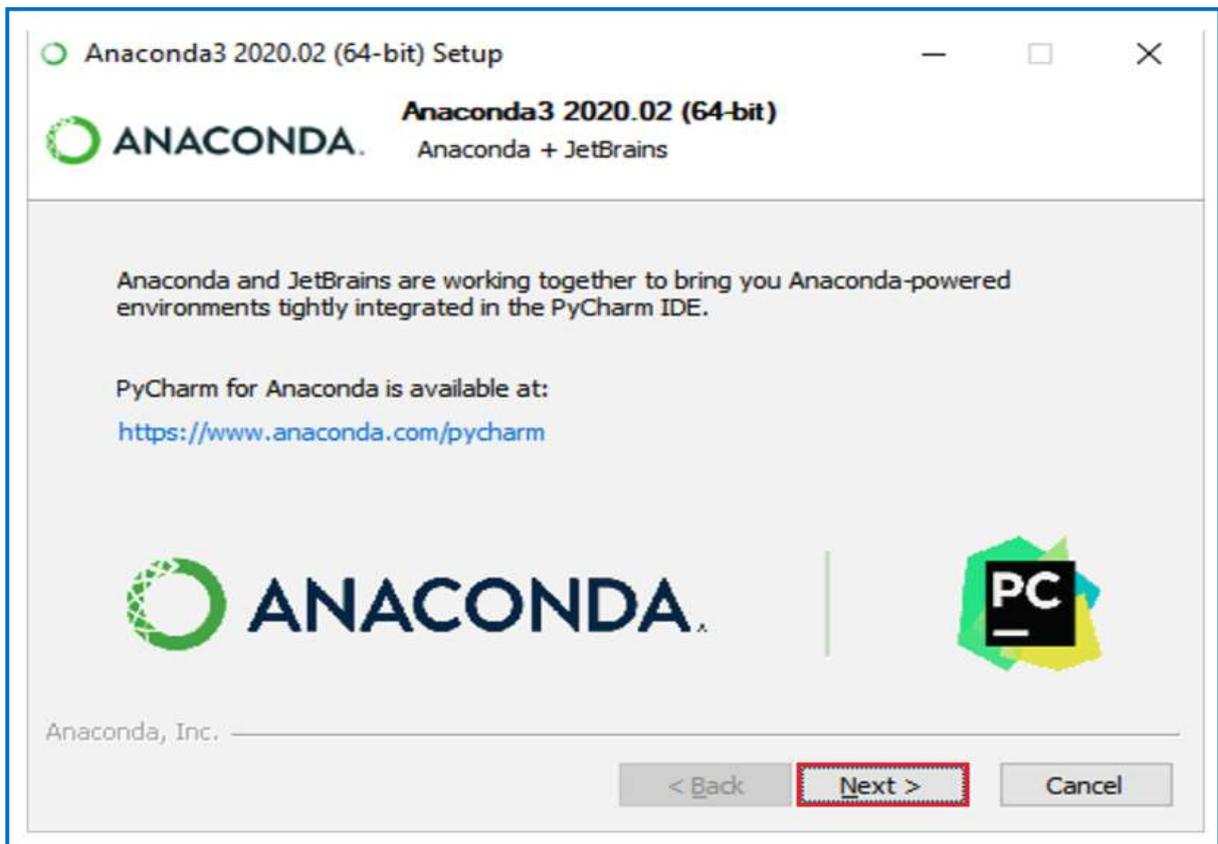


Figure 50: Anaconda Set-up (7 of 21)

10. And finally click finish. Anaconda is now installed

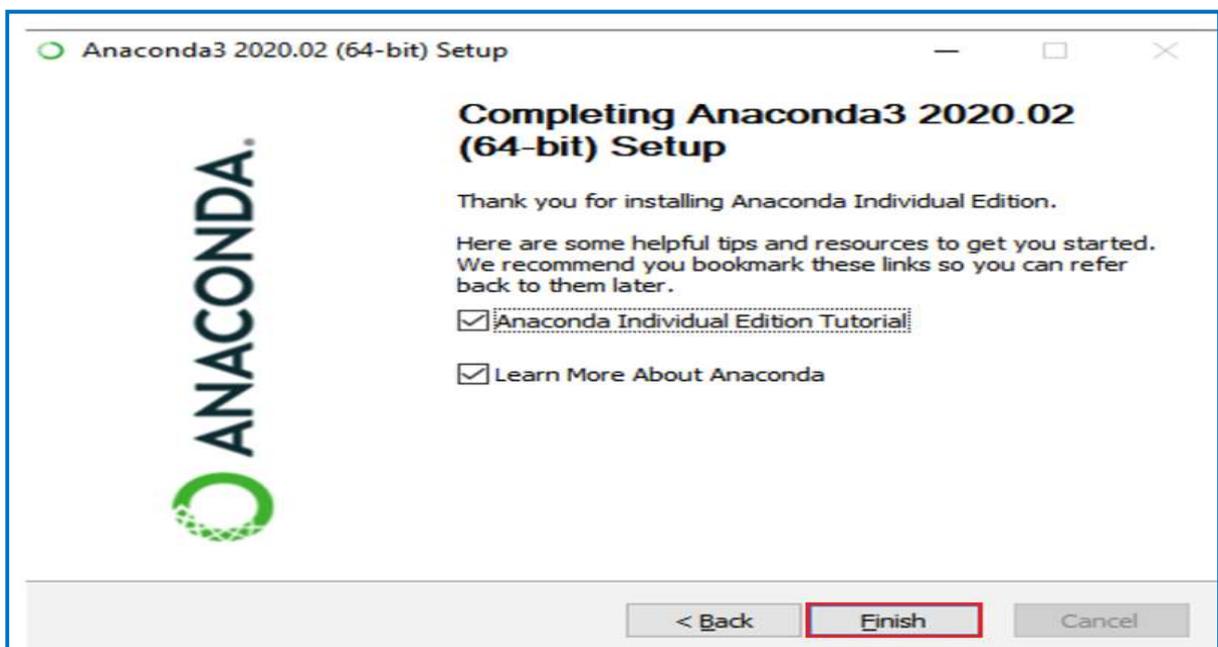


Figure 51: Anaconda Set-up (8 of 21)

11. After installation is complete search for Anaconda Prompt in the search bar.

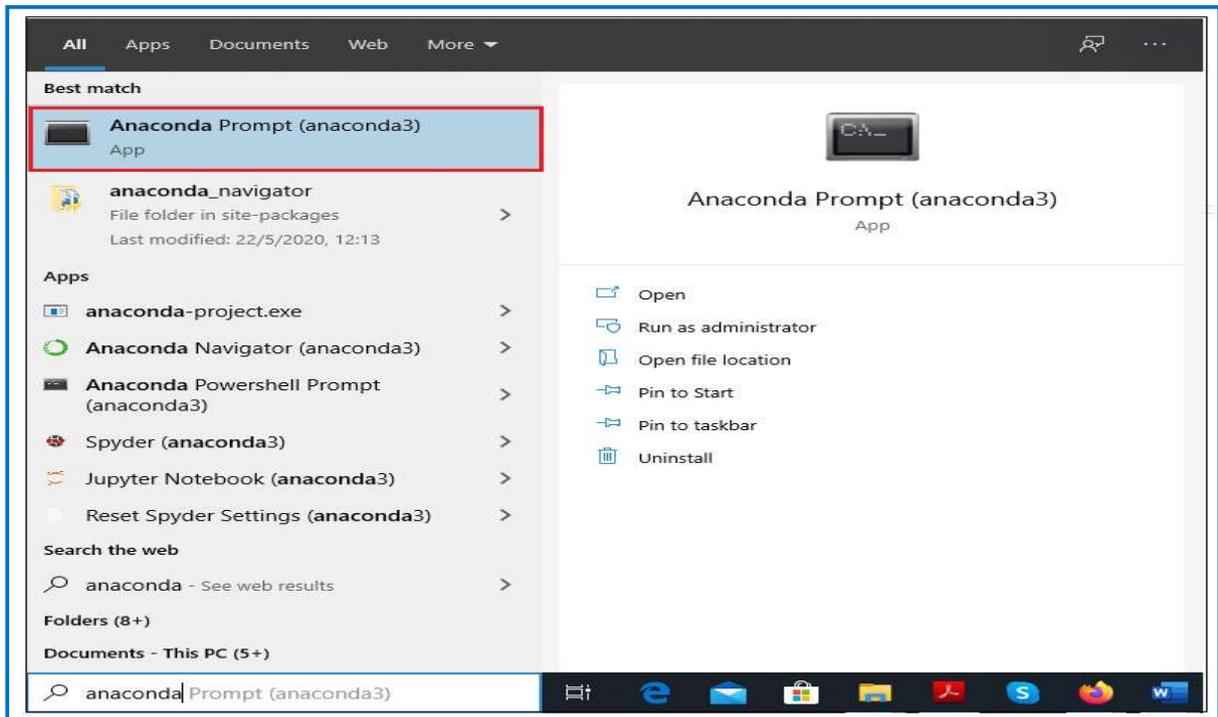


Figure 52: Anaconda Set-up (9 of 21)

12. Open Anaconda prompt, which is similarly to Command Prompt in windows but powered by the Anaconda Distribution. To check if Python has been successfully installed type python.

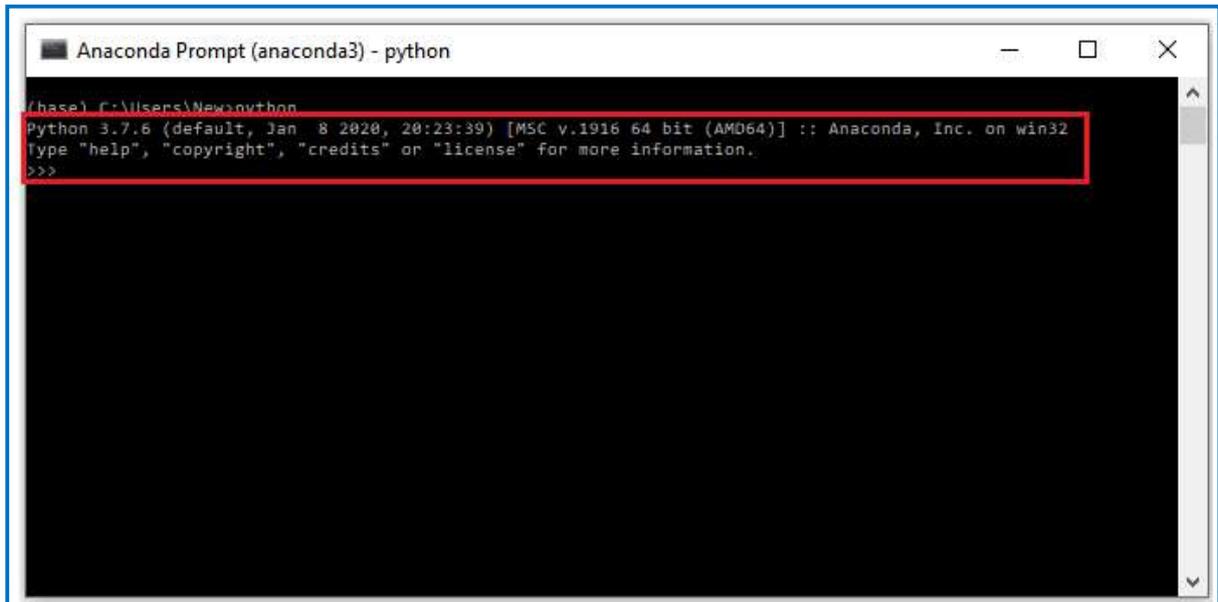
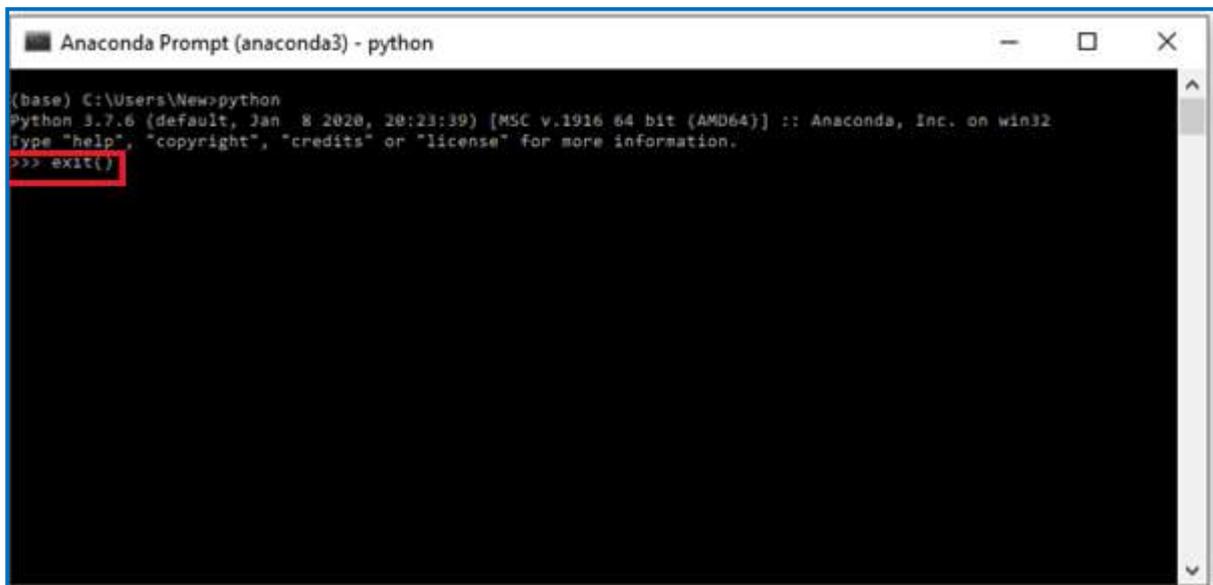


Figure 53: Anaconda Set-up (10 of 21)

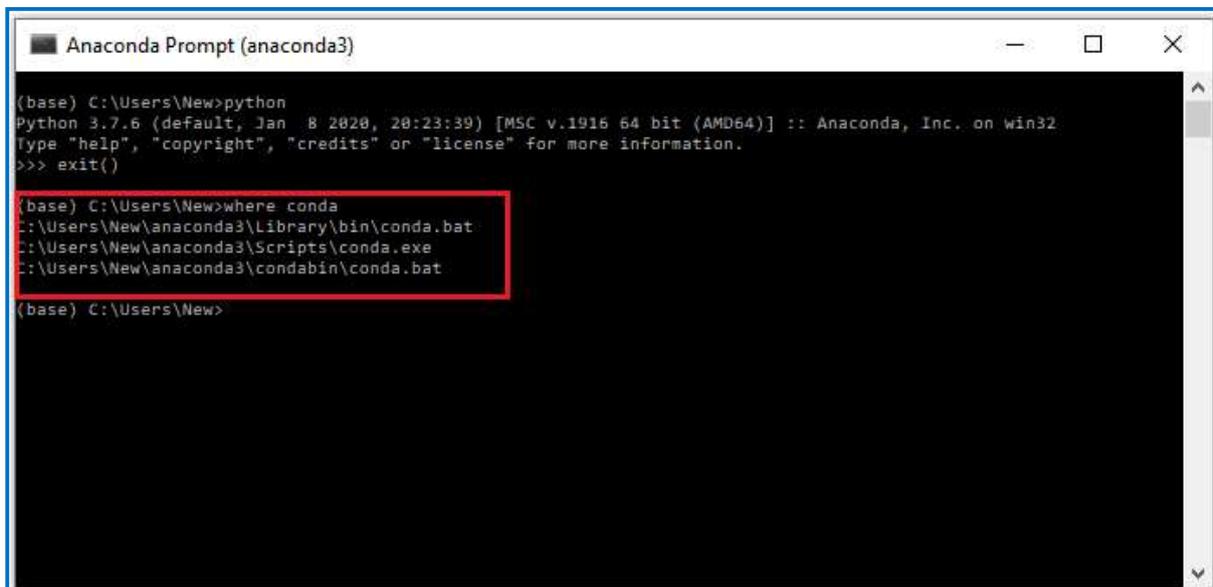
13. To exit type exit()



```
Anaconda Prompt (anaconda3) - python
(base) C:\Users\New>python
Python 3.7.6 (default, Jan 8 2020, 20:23:39) [MSC v.1916 64 bit (AMD64)] :: Anaconda, Inc. on win32
Type "help", "copyright", "credits" or "license" for more information.
>>> exit()
```

Figure 54: Anaconda Set-up (11 of 21)

14. Next check the location of the Anaconda file path. The location is needed to be able to add to Systems properties.



```
Anaconda Prompt (anaconda3)
(base) C:\Users\New>python
Python 3.7.6 (default, Jan 8 2020, 20:23:39) [MSC v.1916 64 bit (AMD64)] :: Anaconda, Inc. on win32
Type "help", "copyright", "credits" or "license" for more information.
>>> exit()

(base) C:\Users\New>where conda
C:\Users\New\anaconda3\Library\bin\conda.bat
C:\Users\New\anaconda3\Scripts\conda.exe
C:\Users\New\anaconda3\condabin\conda.bat

(base) C:\Users\New>
```

Figure 55: Anaconda Set-up (12 of 21)

15. To complete setup the above file paths need to be added to System properties. To open run box press Windows Key + R. When the box opens type sysdm.cpl and click OK

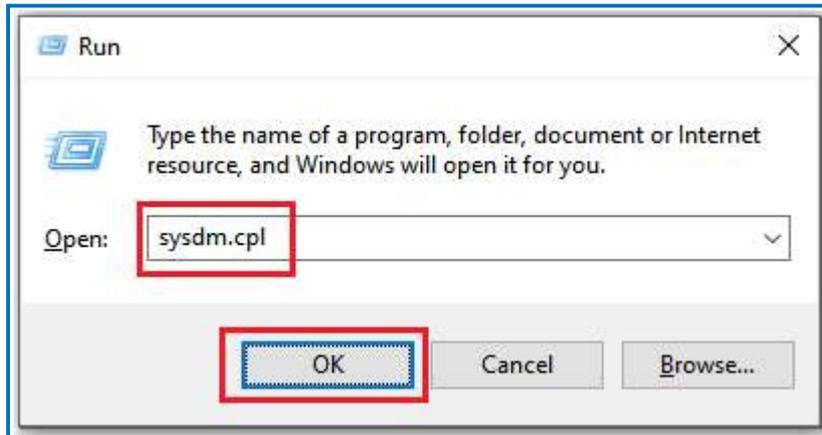


Figure 56: Anaconda Set-up (13 of 21)

16. Click on Advanced

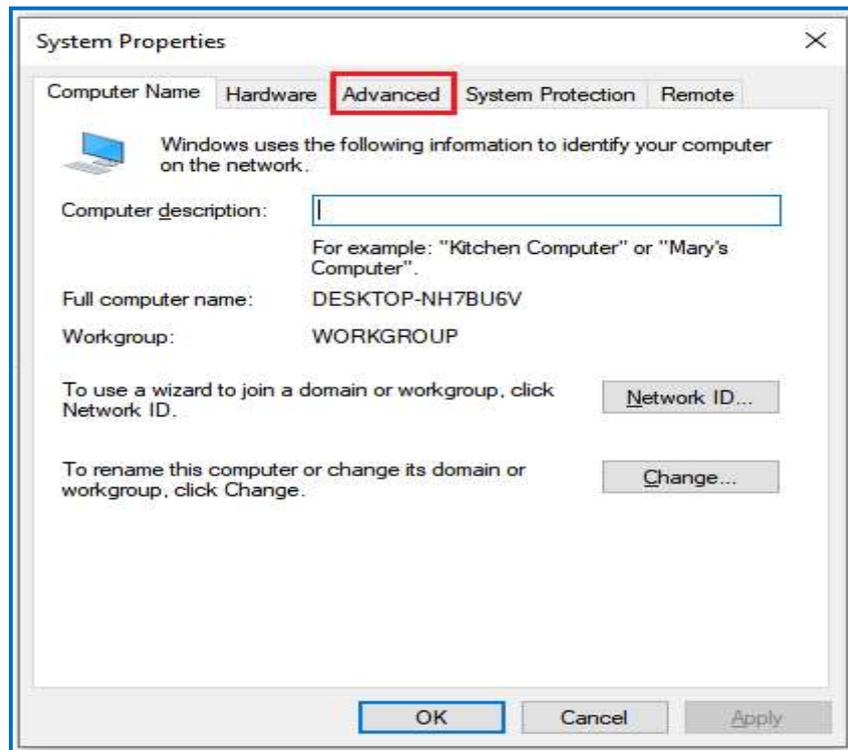


Figure 57: Anaconda Set-up (14 of 21)

17. Then click Environmental Variables

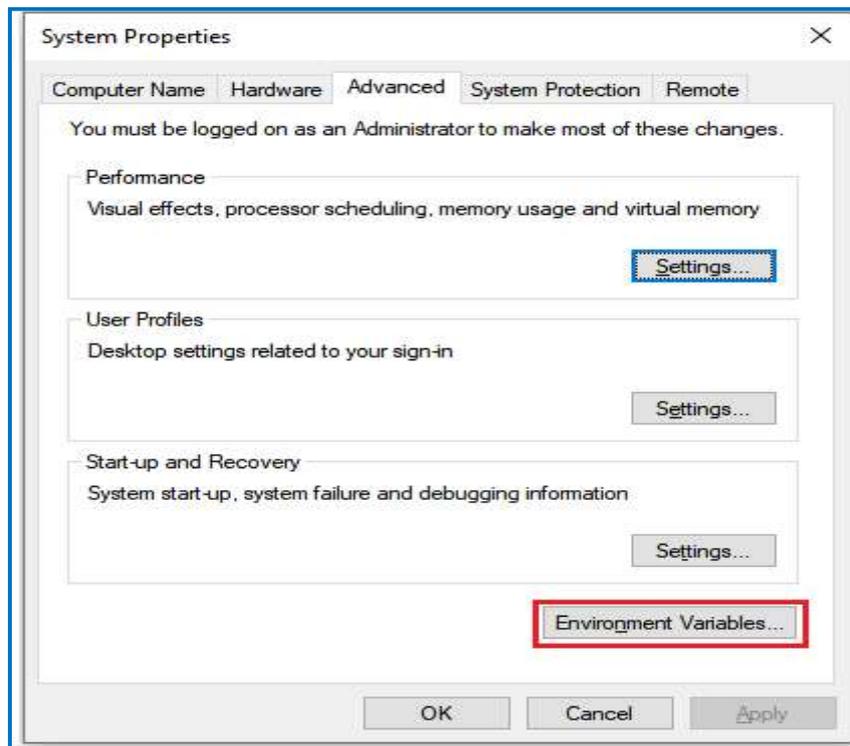


Figure 58: Anaconda Set-up (15 of 21)

18. Click on Path and then click Edit

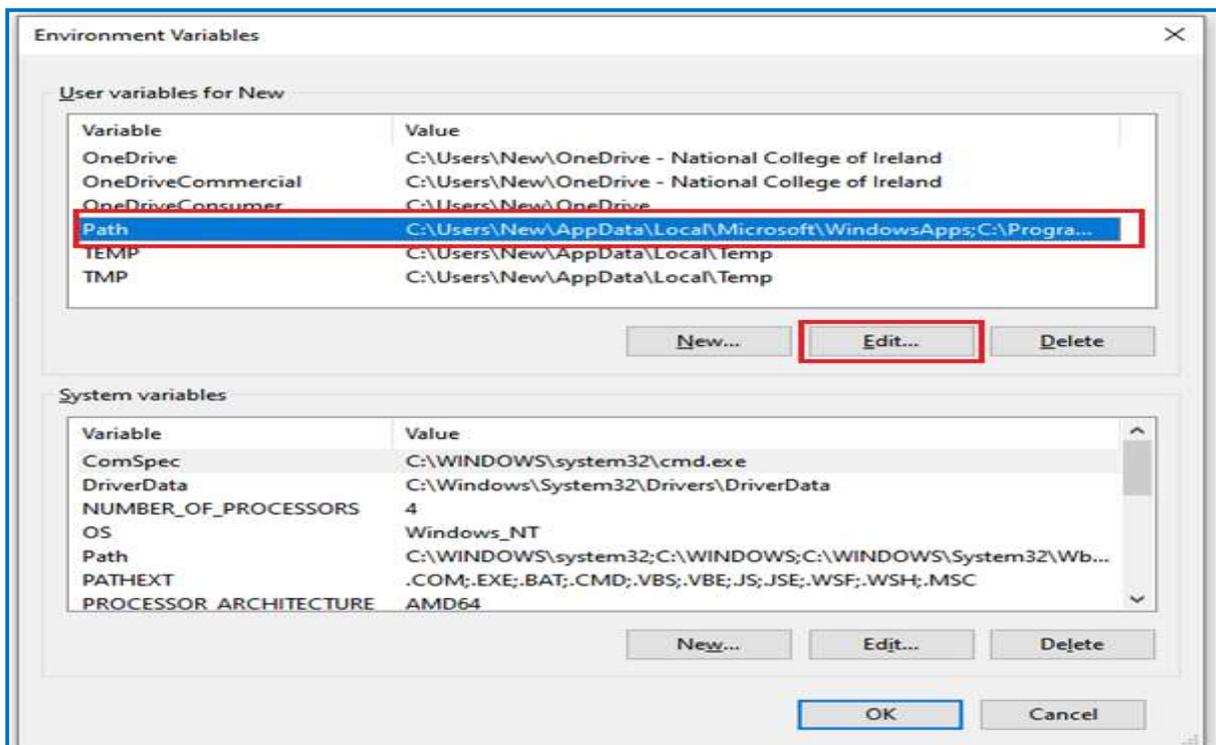


Figure 59: Anaconda Set-up (16 of 21)

19. Click New

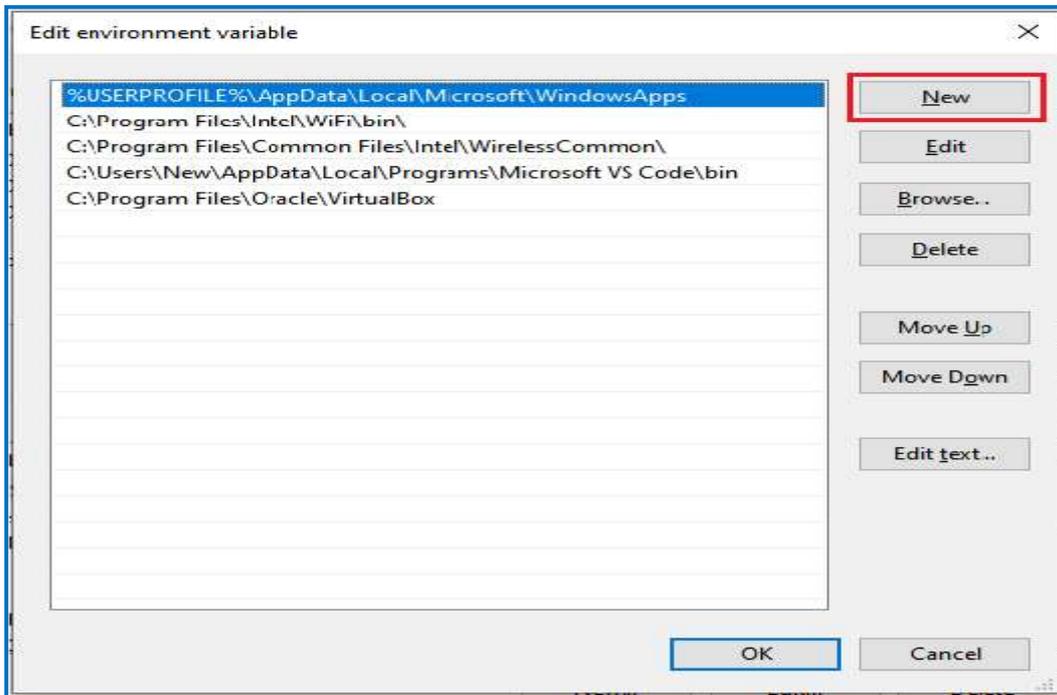


Figure 60: Anaconda Set-up (17 of 21)

20. Then add the locations of files that were returned from Anaconda Prompt where conda. When done click ok and ok for the remaining screens.

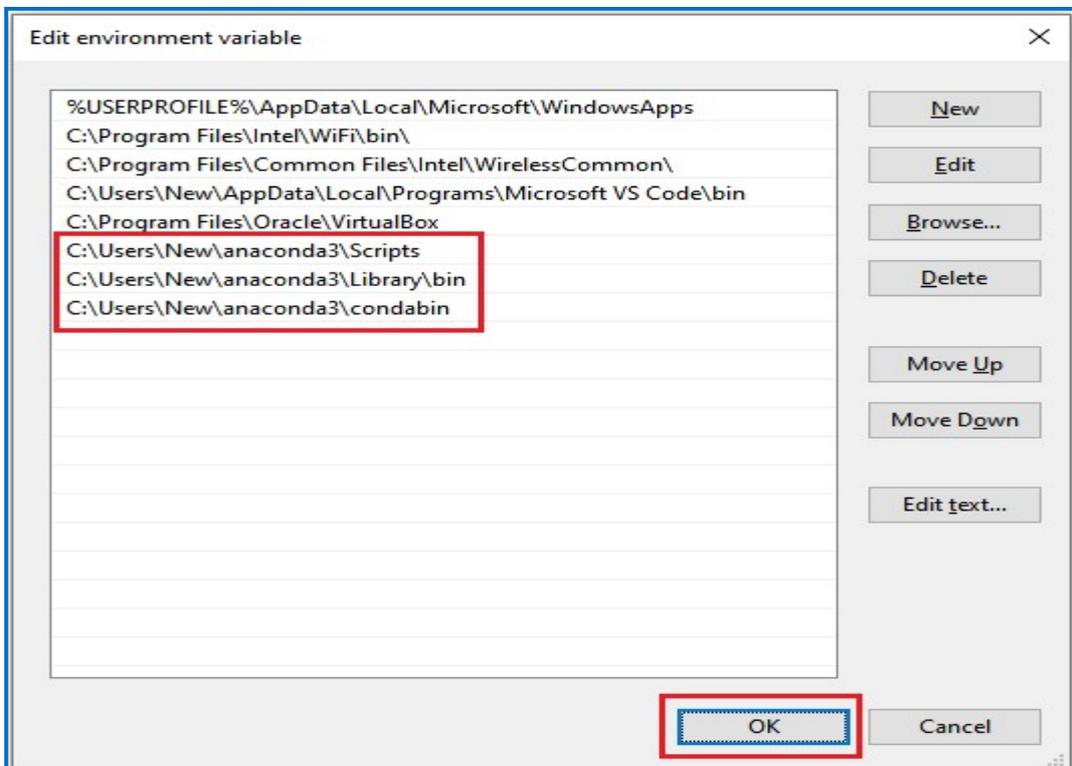


Figure 61: Anaconda Set-up (18 of 21)

21. 20. After above has been completed to check if installation is correct press Windows Key + R again. Enter cmd and click ok.

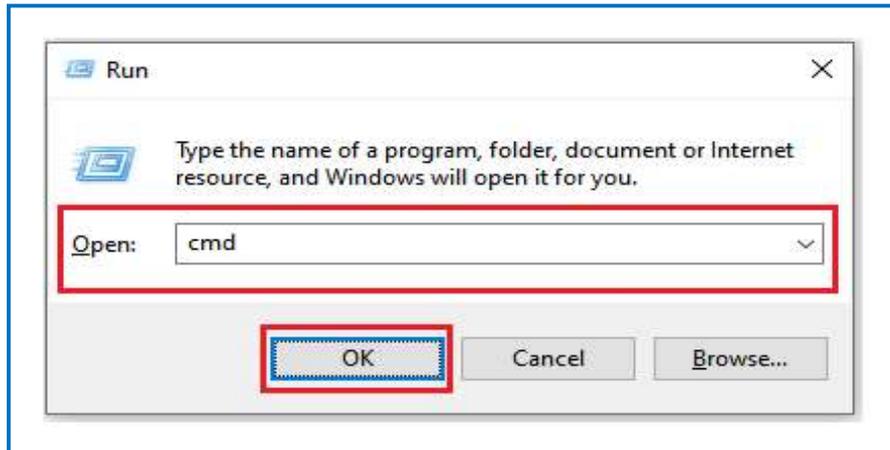


Figure 62: Anaconda Set-up (19 of 21)

22. When Command Prompt opens type conda and the second highlighted box should appear if installed correctly.

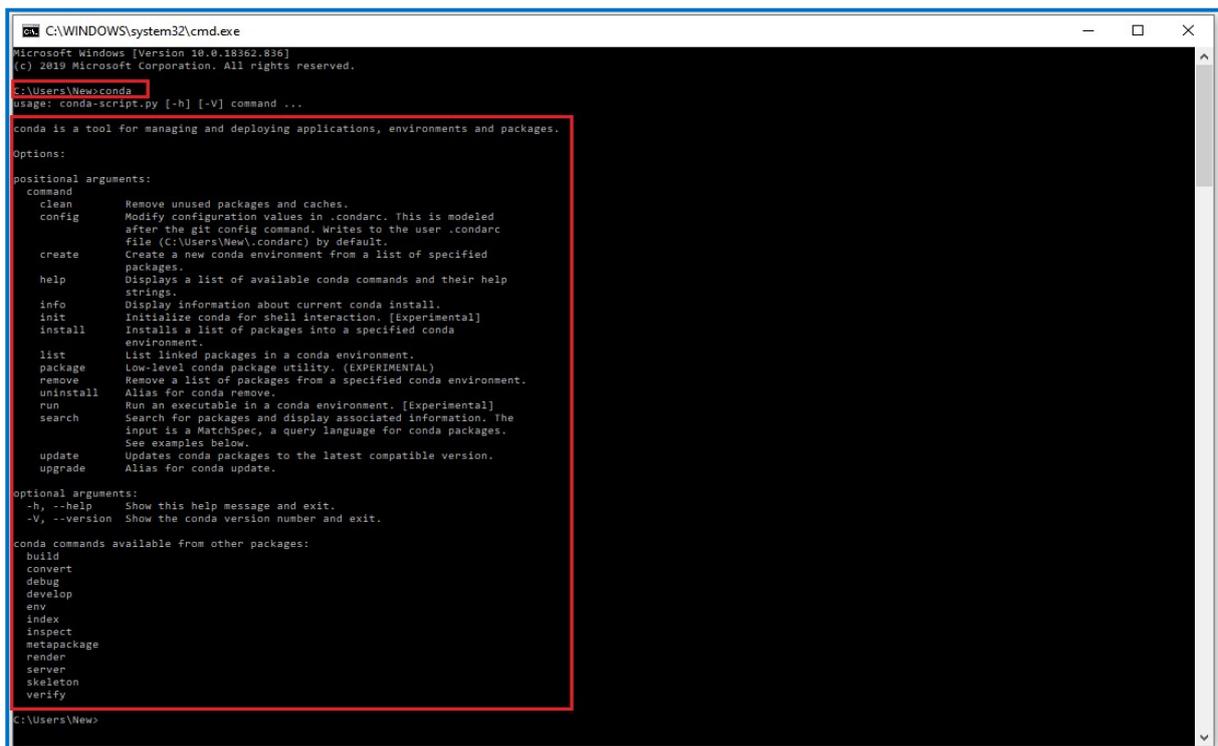
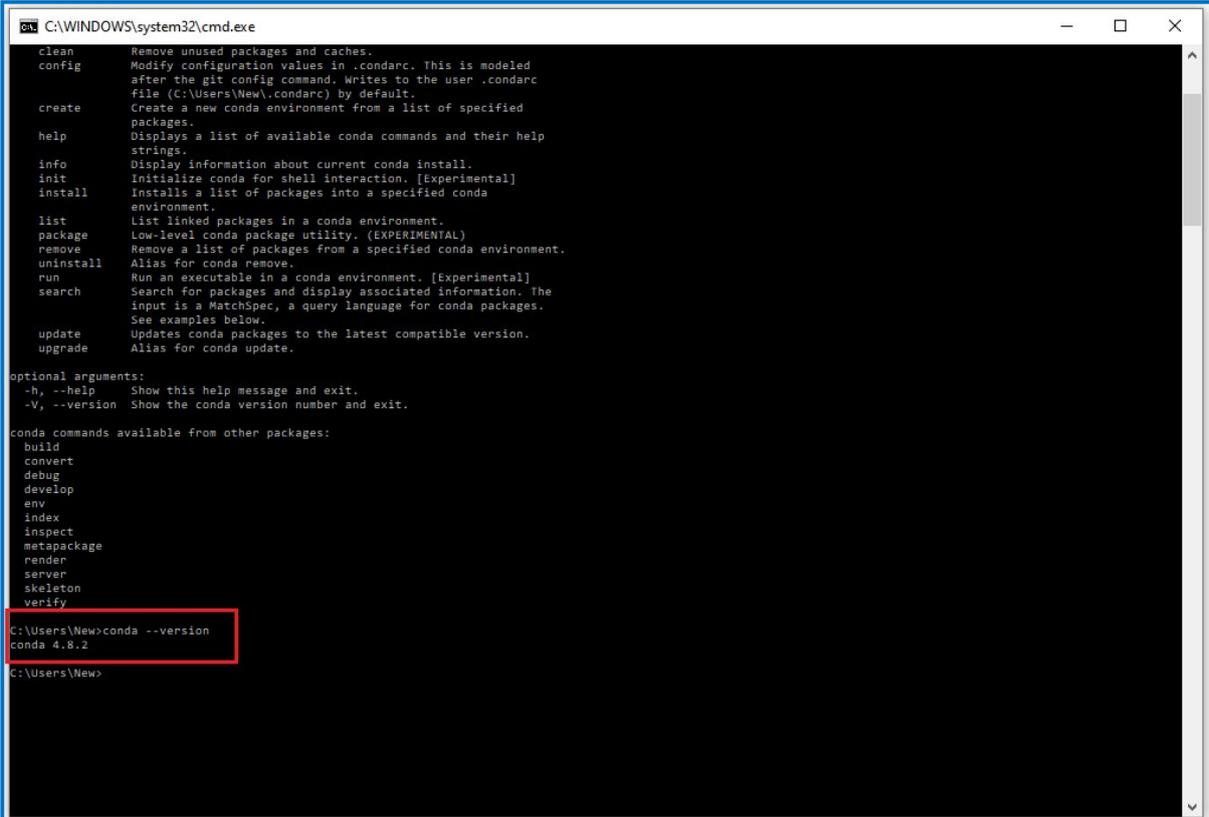


Figure 63: Anaconda Set-up (20 of 21)

23. To check which version, enter conda --version



```
C:\WINDOWS\system32\cmd.exe
clean      Remove unused packages and caches.
config     Modify configuration values in .condarc. This is modeled
           after the git config command. Writes to the user .condarc
           file (C:\Users\New\.condarc) by default.
create     Create a new conda environment from a list of specified
           packages.
help       Displays a list of available conda commands and their help
           strings.
info       Display information about current conda install.
init       Initialize conda for shell interaction. [Experimental]
install    Installs a list of packages into a specified conda
           environment.
list       List linked packages in a conda environment.
package   Low-level conda package utility. (EXPERIMENTAL)
remove     Remove a list of packages from a specified conda environment.
uninstall  Alias for conda remove.
run        Run an executable in a conda environment. [Experimental]
search     Search for packages and display associated information. The
           input is a MatchSpec, a query language for conda packages.
           See examples below.
update     Updates conda packages to the latest compatible version.
upgrade    Alias for conda update.

optional arguments:
-h, --help  Show this help message and exit.
-V, --version Show the conda version number and exit.

conda commands available from other packages:
build
convert
debug
develop
env
index
inspect
metapackage
render
server
skeleton
verify

C:\Users\New>conda --version
conda 4.8.2
C:\Users\New>
```

Figure 64: Anaconda Set-up (21 of 21)

3.6 Weights & Biases

Weights and Biases (W&B) is a developer tool for tracking experiments for deep learning and users can sign up to W&B with their Google account from the applications website⁷

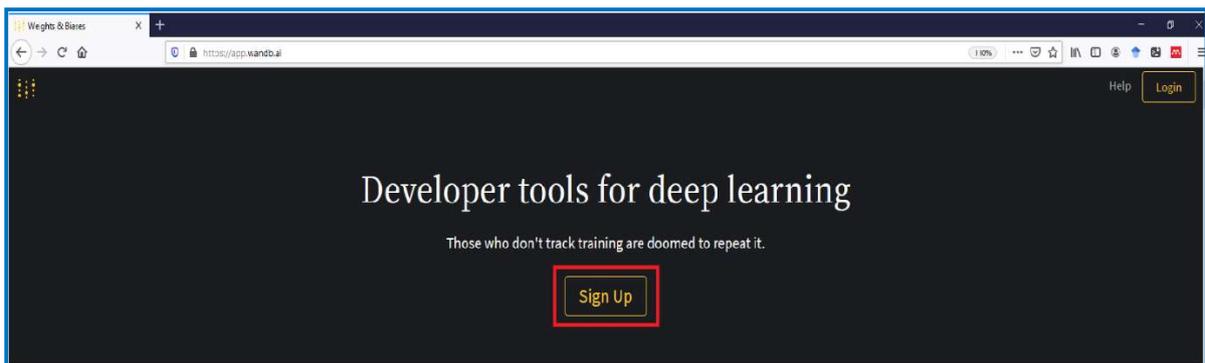


Figure 65: Weights and Biases Set-up

⁷ <https://app.wandb.ai/>

3.7 Putty

PuTTY is a Secure Shell (SSH) software that is utilised for the secure connection between an SSH Client and an SSH Server. For the research project PuTTY was used to tunnel or forward Jupyter Lab from the candidate's laptop to the Genesis Cloud Instance. The software can be downloaded from the PuTTY website⁸.

1. Click here



Figure 66: PuTTY Set-up (1 of 7)

2. Click on below link for 64-bit



Figure 67: PuTTY Set-up (2 of 7)

3. Click save and then run the downloaded file

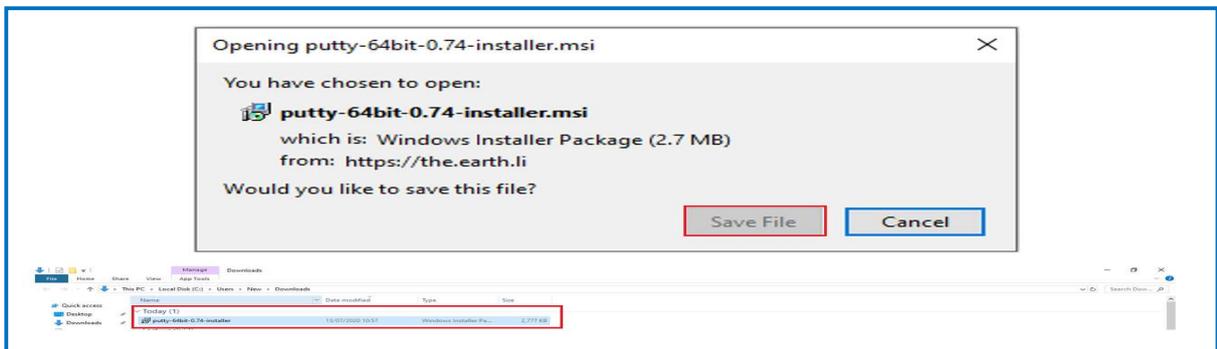


Figure 68: PuTTY Set-up (3 of 7)

⁸ <https://www.putty.org/>

4. Click Next

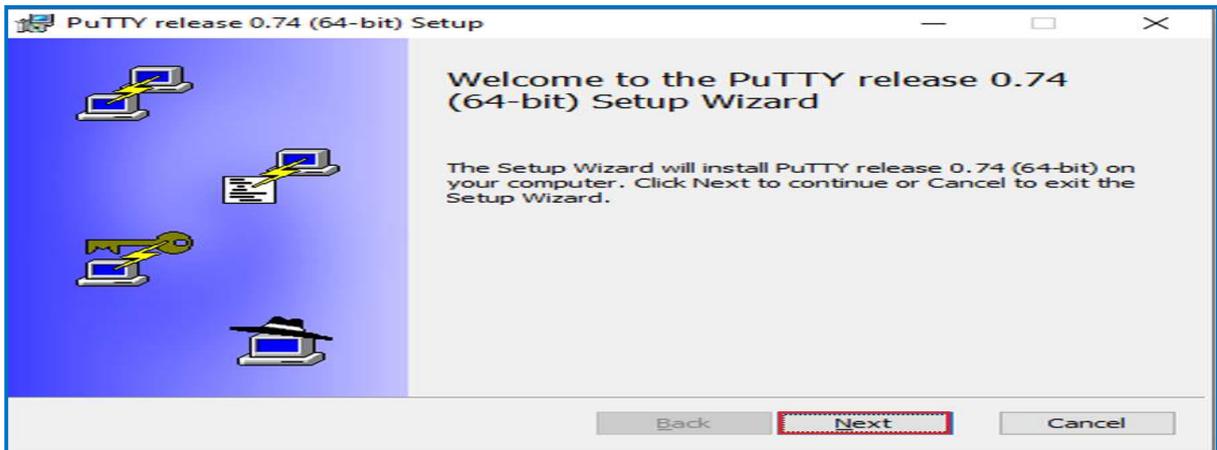


Figure 69: PuTTY Set-up (4 of 7)

5. And click Next again

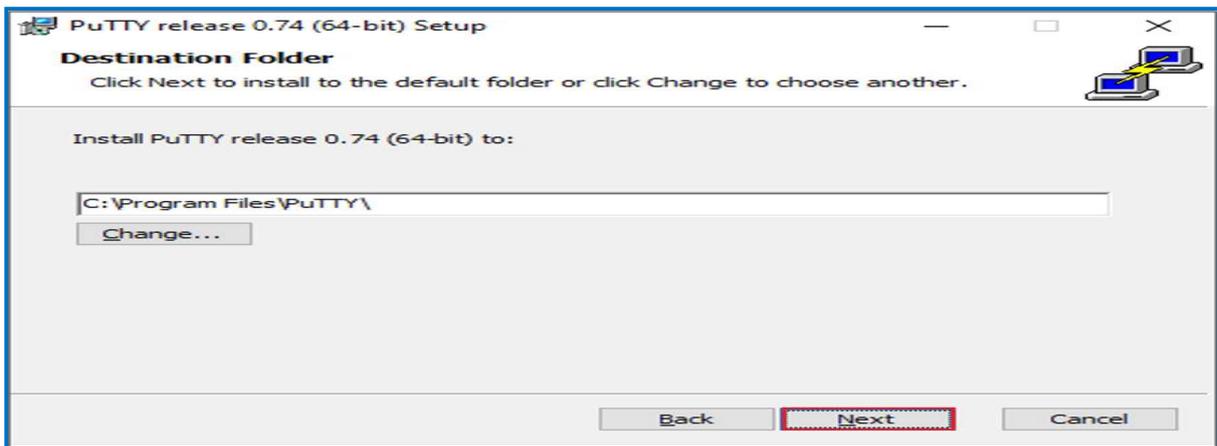


Figure 70: PuTTY Set-up (5 of 7)

6. Click Install

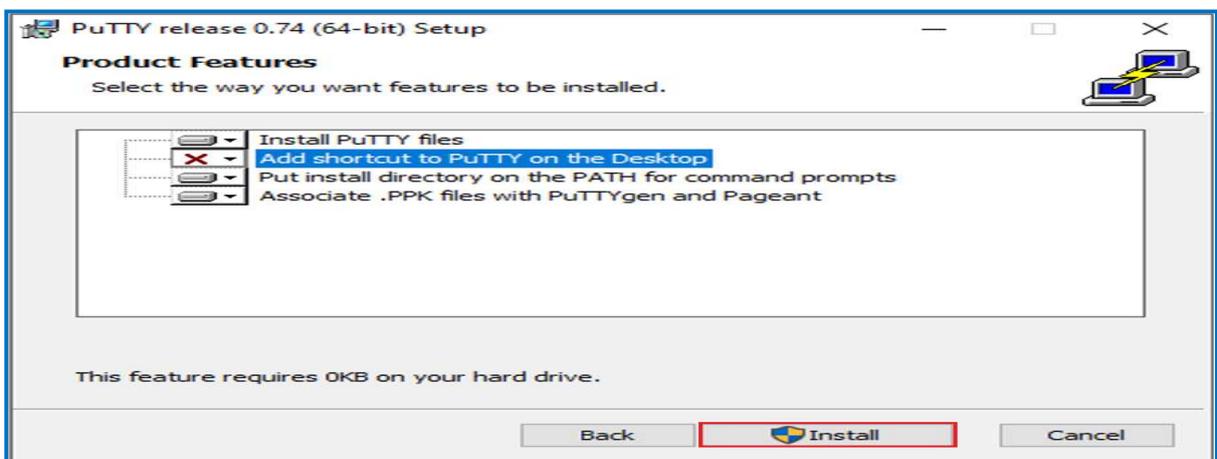


Figure 71: PuTTY Set-up (6 of 7)

7. Finally, click Finish. PuTTY is installed successfully

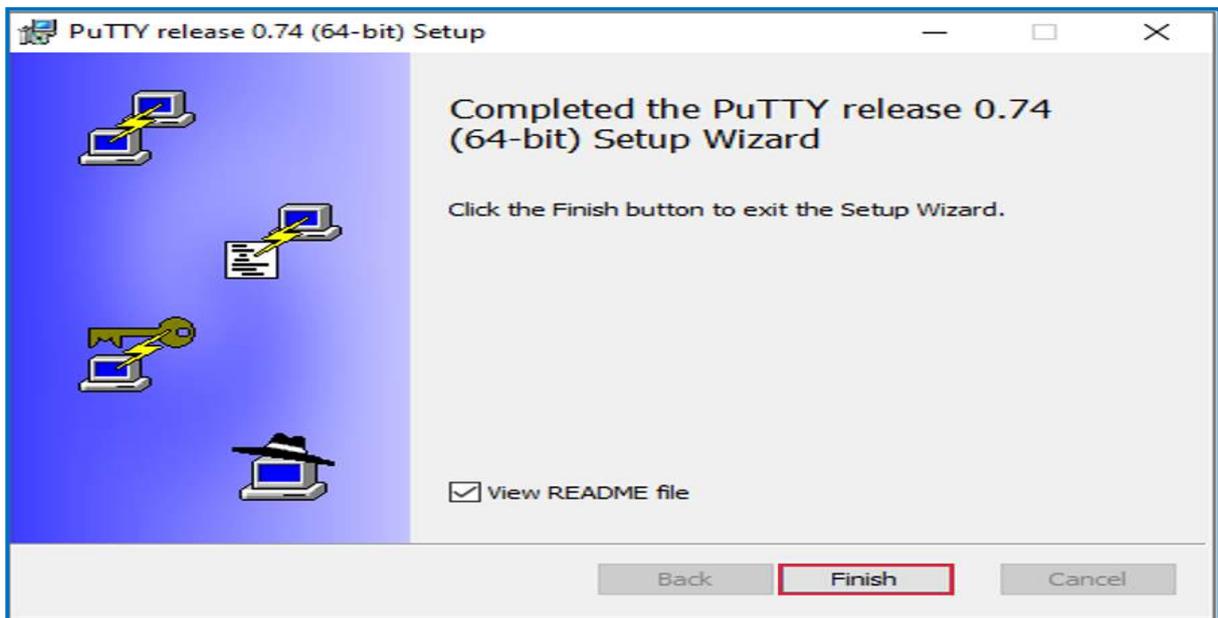


Figure 72: PuTTY Set-up (7 of 7)

3.8 Cliget

Cliget is a Firefox extension that enables the downloading of protected files from sources such as Drive to virtual or remote machines. The add-on adds curl, wget or aria2 to Firefox's download dialog so that user can copy the curl command and run on another machine. The extension can be added to Firefox from the browser add-ons page⁹.

1. Click Add to Firefox

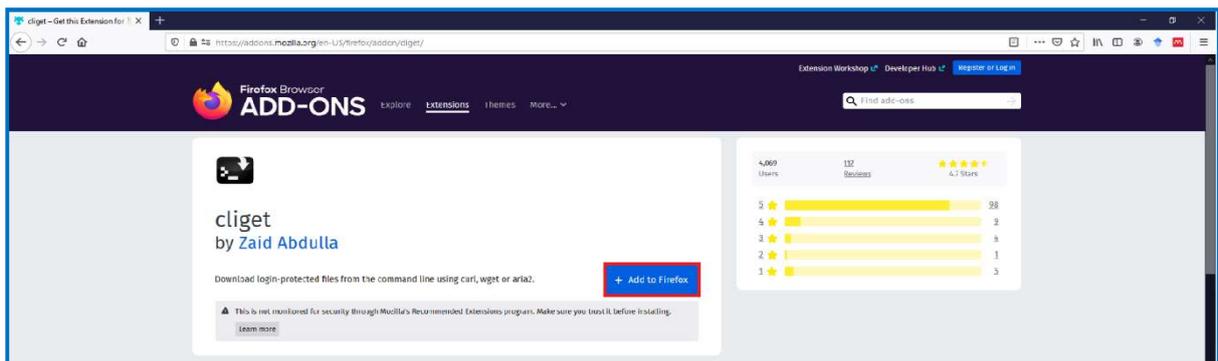


Figure 73: Cliget Add-On

⁹ <https://addons.mozilla.org/en-US/firefox/addon/cliget/>

2. Click Add

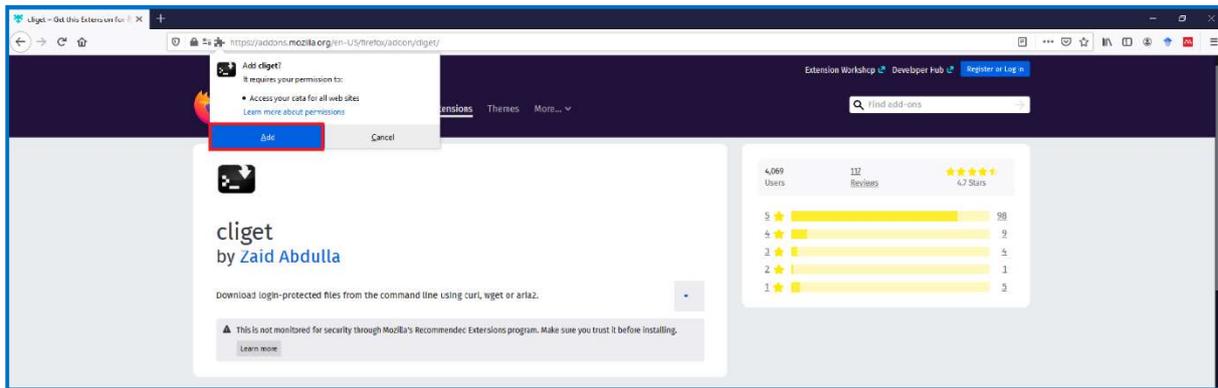


Figure 74: Cliget Download

4 Dataset Creation

The two Datasets were created in Azure Databricks. Launch Databricks as previously demonstrated from the Azure Portal Homepage. First a new cluster needs to be created from the Databricks homepage

1. Click New Cluster

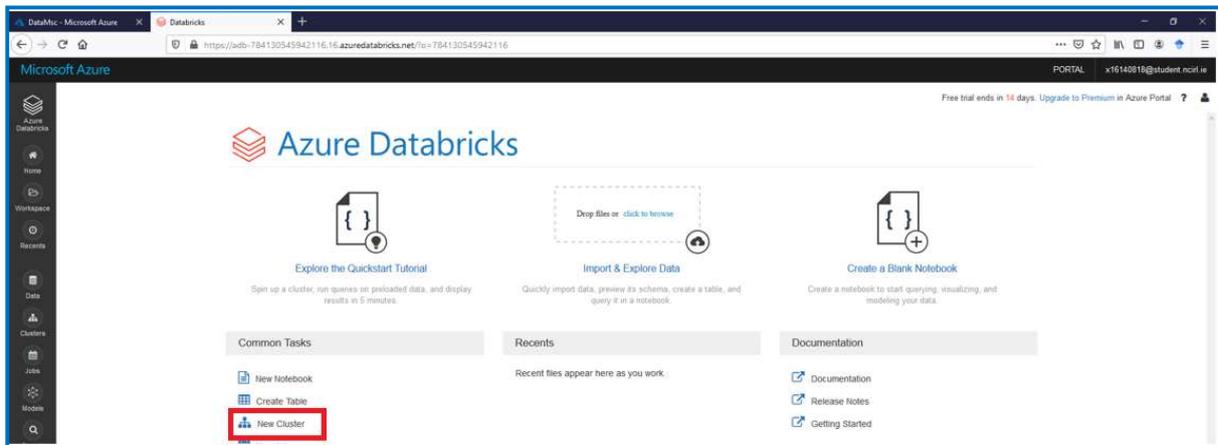


Figure 75: Cluster Set-up (1 of 3)

2. Update below information and click Create Cluster

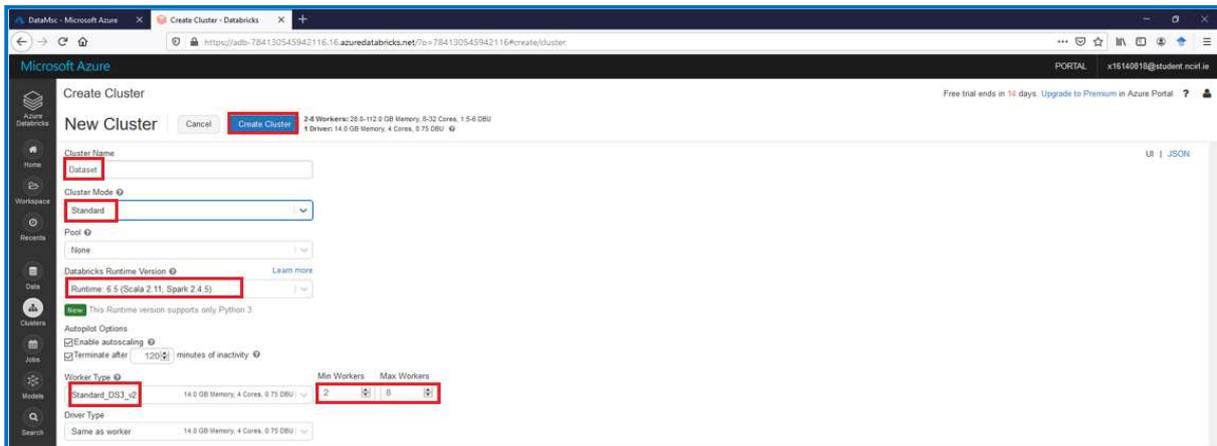


Figure 76: Cluster Set-up (2 of 3)

3. When the Cluster has been approved the State will change to running

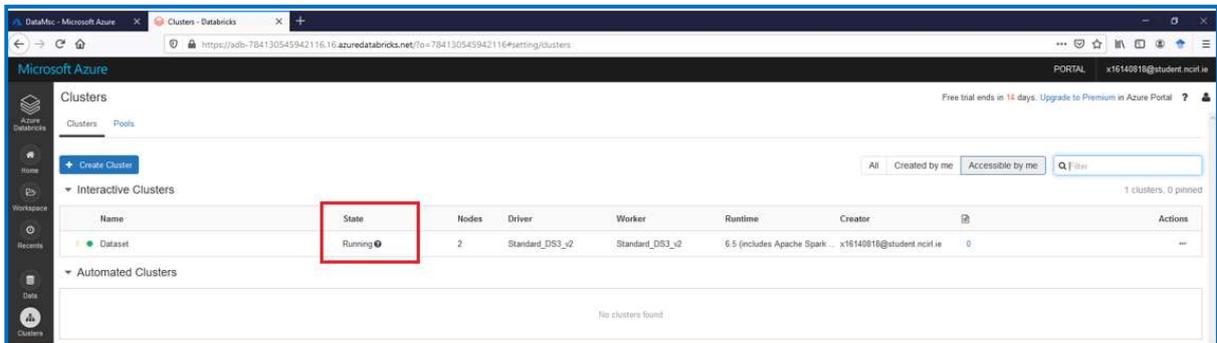


Figure 77: Cluster Set-up (3 of 3)

4. Return to Homepage and launch Jupyter Notebook

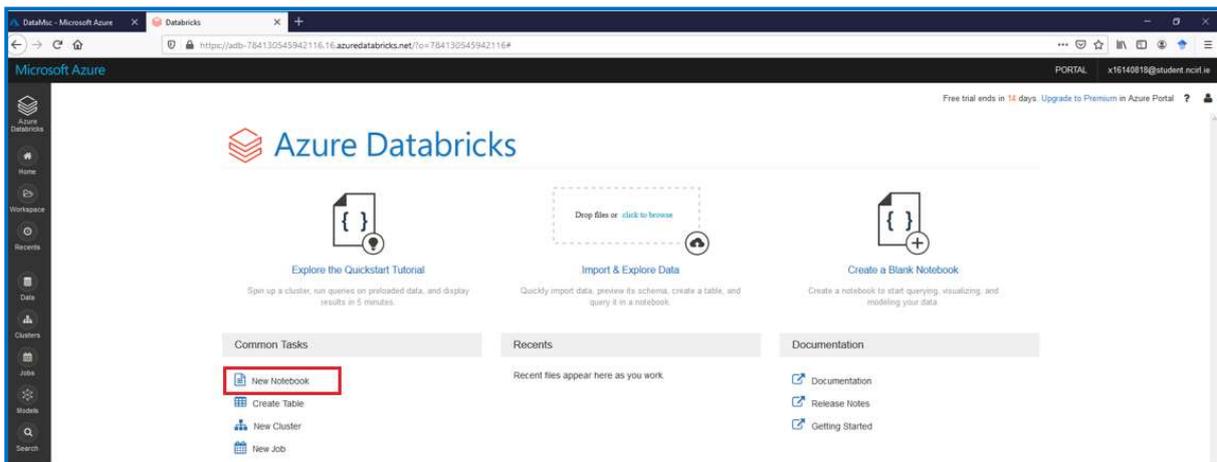


Figure 78: Jupyter Notebook

The following PySpark code was run to create the two datasets

1. Import libraries

```
1 from pyspark.sql.functions import *
2 from pyspark.sql.types import *
3 from graphframes import *
4 from pyspark.mllib.stat import *
5 import pandas as pd
6 from pyspark.ml.classification import LogisticRegression
7 from pyspark.ml.feature import VectorAssembler
8 from pyspark.ml.feature import StandardScaler
9 from pandas.plotting import scatter_matrix
10 import seaborn as sns
11 import matplotlib.pyplot as plt
12 import numpy as np
13 import six
14 from pyspark.ml.evaluation import BinaryClassificationEvaluator
15 from pyspark.ml.classification import RandomForestClassifier
16 from pyspark.ml.tuning import ParamGridBuilder, CrossValidator
17 import mlflow
18 import mlflow.mleap

Command took 0.05 seconds -- by x16140818@student.ncirl.ie at 25/06/2020, 00:31:45 on unknown cluster
```

Figure 79: Python Libraries

2. To import external Libraries such as MLflow click Import Library

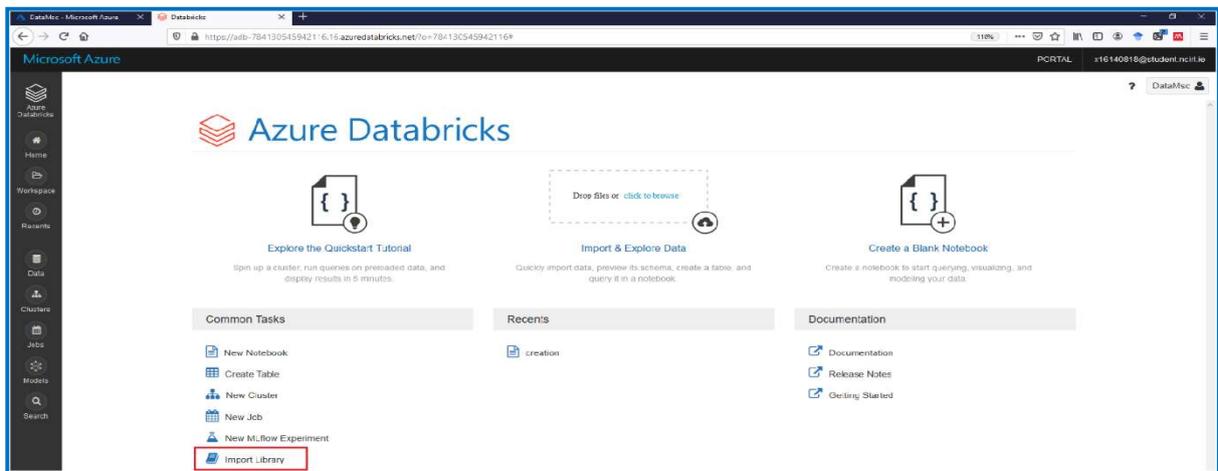


Figure 80: External Libraries (1 of 3)

3. Then libraries from PyPI, Maven and CRAN can be imported



Figure 81: External Libraries (2 of 3)

4. The following external libraries were used

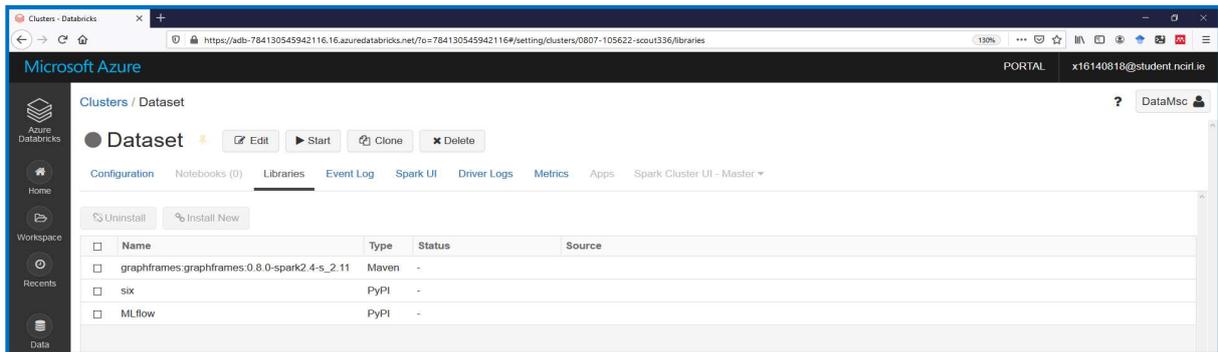


Figure 82: External Libraries (3 of 3)

5. Mount files from Azure to Databricks. This was why the Key Vault and secret were necessary



Figure 83: Mount Azure Files (1 of 3)



Figure 84: Mount Azure Files (2 of 3)



Figure 85: Mount Azure Files (3 of 3)

6. Unmount the files if necessary when all code has been run

```
Cmd 5

1 dbutils.fs.unmount("/mnt/business")

/mnt/business has been unmounted.
Out[13]: True

Command took 26.57 seconds -- by x16140818@student.ncirl.ie at 24/06/2020, 10:09:59 on unknown cluster
```

Figure 86: Unmount Azure Files (1 of 3)

```
Cmd 6

1 dbutils.fs.unmount("/mnt/review")

/mnt/review has been unmounted.
Out[14]: True

Command took 24.05 seconds -- by x16140818@student.ncirl.ie at 24/06/2020, 10:10:02 on unknown cluster
```

Figure 87: Unmount Azure Files (2 of 3)

```
Cmd 7

1 dbutils.fs.unmount("/mnt/user")

/mnt/user has been unmounted.
Out[15]: True

Command took 25.06 seconds -- by x16140818@student.ncirl.ie at 24/06/2020, 10:10:05 on unknown cluster
```

Figure 88: Unmount Azure Files (3 of 3)

7. Read Business, Review and User JSON files as PySpark DataFrames

```
Cmd 8

1 business = spark.read.json("dbfs:/mnt/business/yelp_academic_dataset_business.json")

business: pyspark.sql.dataframe.DataFrame = [address: string, attributes: struct ... 12 more fields]

Command took 7.07 seconds -- by x16140818@student.ncirl.ie at 24/06/2020, 10:21:19 on unknown cluster
```

Figure 89: Read JSON Files (1 of 3)

```

Cmd 9
1 review = spark.read.json("dbfs:/mnt/review/yelp_academic_dataset_review.json")

review: pyspark.sql.dataframe.DataFrame = [business_id: string, cool: long ... 7 more fields]
Command took 1.33 minutes -- by x16140818@student.ncirl.ie at 24/06/2020, 10:21:31 on unknown cluster

```

Figure 90: Read JSON Files (2 of 3)

```

Cmd 10
1 user = spark.read.json("dbfs:/mnt/user/yelp_academic_dataset_user.json")

user: pyspark.sql.dataframe.DataFrame = [average_stars: double, compliment_cool: long ... 20 more fields]
Command took 45.75 seconds -- by x16140818@student.ncirl.ie at 24/06/2020, 10:21:36 on unknown cluster

```

Figure 91: Read JSON Files (3 of 3)

8. Star Distribution

```

Cmd 11
1 display(review.groupBy("stars").count().sort(desc("count")))

```

	stars	count
1	5	3586460
2	4	1673404
3	1	1283897
4	3	842289
5	2	635072

Showing all 5 rows.

```

Command took 1.34 minutes -- by x16140818@student.ncirl.ie at 24/06/2020, 10:21:40 on unknown cluster

```

Figure 92: Star Distribution

9. Filter rows if they have no Text

```

Cmd 12
1 review1 = review.filter(review.text.isNotNull())

review1: pyspark.sql.dataframe.DataFrame = [business_id: string, cool: long ... 7 more fields]
Command took 0.05 seconds -- by x16140818@student.ncirl.ie at 24/06/2020, 10:21:43 on unknown cluster

```

Figure 93: Filter Text

10. Filter rows if they have no Star

```
Cmd 13  
1 rd = review1.filter(review1.stars.isNotNull())  
  
▶ rd: pyspark.sql.dataframe.DataFrame = [business_id: string, cool: long ... 7 more fields]  
Command took 0.04 seconds -- by x16140818@student.ncirl.ie at 24/06/2020, 10:21:45 on unknown cluster
```

Figure 94: Filter Star

11. Display Star Count

```
Cmd 14  
1 display(rd.groupBy("stars").count().sort(desc("count")))  
  
┌───┬───┬───┐  
│   │ stars │ count │  
├───┬───┬───┤  
│ 1 │ 1     │ 1283897 │  
│ 2 │ 2     │ 635072  │  
│ 3 │ 3     │ 842289  │  
│ 4 │ 4     │ 1673404 │  
│ 5 │ 5     │ 3586460 │  
└───┬───┬───┘  
Showing all 5 rows.  
┌───┬───┬───┐  
│ 1 │ 2 │ 1 │  
└───┬───┬───┘  
Command took 1.43 minutes -- by x16140818@student.ncirl.ie at 24/06/2020, 10:21:47 on unknown cluster
```

Figure 95: Display Star Count

12. Apply stratified sampling to create Dataset 1. As stratified sampling in PySpark is approximation this needed to be run more than once.

```
Cmd 15  
1 dataset1 = rd.sampleBy("stars", fractions={1: 0.109043015132834, 2: 0.220447445329034, 3: 0.16621373424086, 4:  
0.0836618055173766, 5: 0.0390357065184053}, seed=434)  
  
▶ dataset1: pyspark.sql.dataframe.DataFrame = [business_id: string, cool: long ... 7 more fields]  
Command took 0.18 seconds -- by x16140818@student.ncirl.ie at 24/06/2020, 10:29:50 on unknown cluster
```

Figure 96: Stratified Sampling Attempt 1



Figure 97: Stratified Sampling Attempt 1 Star Count

13. Attempt 2



Figure 98: Stratified Sampling Attempt 2

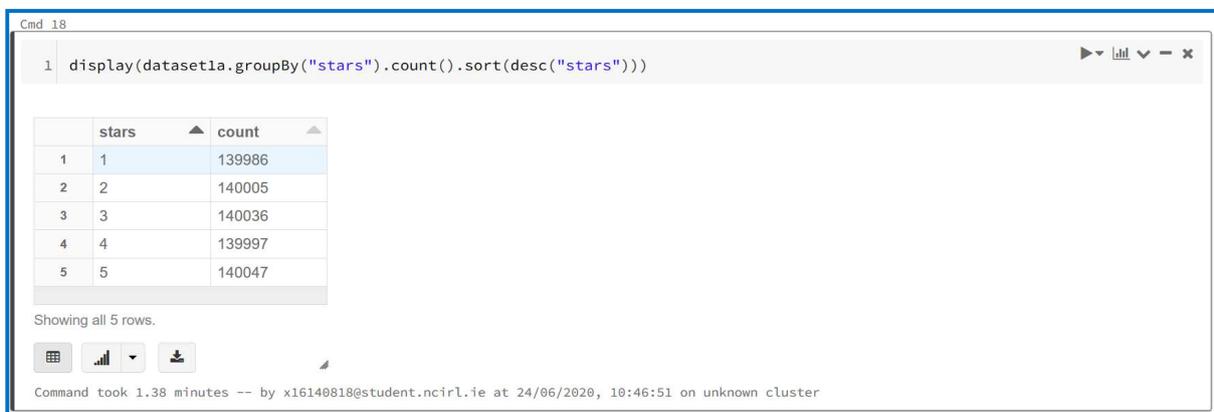


Figure 99: Stratified Sampling Attempt 2 Star Count

14. Attempt 3. Below was chosen to proceed with for the research project

```
Cmd 19
1 dataset1b = rd.sampleBy("stars", fractions=[1: 0.108799795288748, 2: 0.22054354301458, 3: 0.166917363622745, 4: 0.0838919041335896, 5: 0.0391565775051165], seed=434)

▶ dataset1b: pyspark.sql.dataframe.DataFrame = [business_id: string, cool: long ... 7 more fields]
Command took 0.07 seconds -- by x16140818@student.ncirl.ie at 24/06/2020, 10:51:03 on unknown cluster
```

Figure 100: Stratified Sampling Attempt 3

```
Cmd 20
1 display(dataset1b.groupBy("stars").count().sort(desc("stars")))

▶ display: pyspark.sql.dataframe.DataFrame = [stars: string, count: long ... 2 more fields]
Command took 1.43 minutes -- by x16140818@student.ncirl.ie at 24/06/2020, 10:51:07 on unknown cluster
```

	stars	count
1	1	139999
2	2	139998
3	3	140009
4	4	140001
5	5	139981

Showing all 5 rows.

Figure 101: Stratified Sampling Attempt 3 Star Count

15. Select Business Fields

```
Cmd 21
1 bd1 = business.select("business_id", "address", "city", "latitude", "longitude", "name", "state", "categories", "review_count")

▶ bd1: pyspark.sql.dataframe.DataFrame = [business_id: string, address: string ... 7 more fields]
Command took 0.08 seconds -- by x16140818@student.ncirl.ie at 24/06/2020, 10:58:03 on unknown cluster
```

Figure 102: Select Business Fields

16. Merge dataset1b & bd1

```
Cmd 22
1 dataset_one = dataset1b.join(bd1, on="business_id", how="inner")

▶ dataset_one: pyspark.sql.dataframe.DataFrame = [business_id: string, cool: long ... 15 more fields]
Command took 0.09 seconds -- by x16140818@student.ncirl.ie at 24/06/2020, 10:58:04 on unknown cluster
```

Figure 103: Merge DataFrames

17. Replace unnecessary characters, new lines and carriage returns

```
Cmd 23
1 d4 = dataset_one.withColumn("text", regexp_replace(col("text"), "[()&*-,]", ""))

▶ d4: pyspark.sql.dataframe.DataFrame = [business_id: string, cool: long ... 15 more fields]
Command took 0.14 seconds -- by x16140818@student.ncirl.ie at 24/06/2020, 10:58:07 on unknown cluster
```

Figure 104: Remove Whitespaces (1 of 3)

```
Cmd 24
1 d5 = d4.withColumn("text", regexp_replace(col("text"), "[\n\r]", " "))

▶ d5: pyspark.sql.dataframe.DataFrame = [business_id: string, cool: long ... 15 more fields]
Command took 0.07 seconds -- by x16140818@student.ncirl.ie at 24/06/2020, 10:58:09 on unknown cluster
```

Figure 105: Remove Whitespaces (1 of 3)

18. Check Text format is correct

```
Cmd 25
1 d5.select("text").show(5,False)

+-----+
|text
|
+-----+
|Food was piping hot which is great!! Had the buffet...and it truly is all you can eat!! You order whatever you want off the menu except chitlins and they bring it out to you!!! Mac n cheese greens and black eyed peas were pretty good!! Fried chicken was pretty good too! Will come back!!
|
|Bobby Flay's restaurant at Caesar's Palace. Tasty but with mixed reviews. The first time I went there everything I had was delicious. The chicken and pork tenderloin were outstanding. This time I ordered the following - duck crepe 50000 good! shrimp tamale very flavorful and fresh salmon mediocre at best... and our waiter even told us not to order it... but my friend didn't listen and halibut tender flaky not fishy at all... and so flavorful!. The halibut and appetizers were my favorites and I would definitely go and order those again. We also ordered 3 desserts which were totally worth it! Sweet potato pudding and I don't even like sweet potatoes! Banana pie tart thingy and spicy peanut chocolate cake. mmmmm would definitely go there for dessert and the appetizers!!! You need a reservation for this place.
|Pros: Fun atmosphere great for people watching mechanical bull is unique in my experience Cons: I had the chicken tenders and fries and while the tenders were alright the fries were pre
```

Figure 106: (1 of 3)

19. Write Dataset 1 to Databricks File Store (DBFS)

```
Cmd 27
1 d5.coalesce(1).write.csv("FileStore/datasets/d1.csv", header=True)

Command took 1.90 minutes -- by x16140818@student.ncirl.ie at 24/06/2020, 10:58:17 on unknown cluster
```

Figure 107: Write to DBFS

Next the code for creating Dataset 2 is described. First was to generate the Social Network, Content Informativeness and Review Rating features to be used as variables for Logistic Regression (LR). After selecting the features LR is applied to predict review useful score. GraphFrame was used to calculate the Social Network Features of outDegree and PageRank. It requires two inputs Vertices and Edges. To create the Vertices and Edges for GraphFrame using the User Json file the below code was run.

```

Cmd 28
1 ver = user.select("user_id", "name")

▶ ver: pyspark.sql.dataframe.DataFrame = [user_id: string, name: string]
Command took 0.06 seconds -- by x16140818@student.ncirl.ie at 24/06/2020, 11:02:04 on unknown cluster

Cmd 29
1 vertices = ver.withColumnRenamed("user_id", "id")

▶ vertices: pyspark.sql.dataframe.DataFrame = [id: string, name: string]
Command took 0.06 seconds -- by x16140818@student.ncirl.ie at 24/06/2020, 11:02:07 on unknown cluster

Cmd 30
1 vertices.select("id").count()

Out[48]: 1968703
Command took 40.53 seconds -- by x16140818@student.ncirl.ie at 24/06/2020, 11:02:14 on unknown cluster

```

Figure 108: Vertices

1. Create Edges. As the Users friends are in a list, they need to be formatted to have every unique friend on a new row. The columns need to be renamed also to the correct format required by GraphFrames

```

Cmd 31
1 ed = user.select("user_id", "friends")

▶ ed: pyspark.sql.dataframe.DataFrame = [user_id: string, friends: string]
Command took 0.06 seconds -- by x16140818@student.ncirl.ie at 24/06/2020, 11:03:02 on unknown cluster

Cmd 32
1 ed.printSchema()

root
 |-- user_id: string (nullable = true)
 |-- friends: string (nullable = true)

Command took 0.04 seconds -- by x16140818@student.ncirl.ie at 24/06/2020, 11:03:41 on unknown cluster

Cmd 33
1 ed1 = ed.withColumn("friends", explode(split(regexp_replace(col("friends"), "(^\\[\\])(\\]$)", ""), ",")))

▶ ed1: pyspark.sql.dataframe.DataFrame = [user_id: string, friends: string]
Command took 0.13 seconds -- by x16140818@student.ncirl.ie at 24/06/2020, 11:07:26 on unknown cluster

```

Figure 109: Edges

```

Cmd 34
1 ed1.select("user_id","friends").show(3, False)

+-----+
|user_id          |friends          |
+-----+
|ntlvfPzc8eglvk92iDIAw|oeMvJh94PiGQnx_6GlnDPQ |
|ntlvfPzc8eglvk92iDIAw|wm1z1PaJKvHgSDRKfwhfDg|
|ntlvfPzc8eglvk92iDIAw|IkRib6Xs91PPW7pon7VVig|
+-----+
only showing top 3 rows

Command took 0.46 seconds -- by x16140818@student.ncirl.ie at 24/06/2020, 11:07:31 on unknown cluster

Cmd 35
1 edges = (ed1.withColumnRenamed("user_id","src").withColumnRenamed("friends","dst"))

edges: pyspark.sql.dataframe.DataFrame = [src: string, dst: string]

Command took 0.04 seconds -- by x16140818@student.ncirl.ie at 24/06/2020, 11:07:45 on unknown cluster

```

Figure 110: Rename Fields

2. Create GraphFrame

```

Cmd 36
1 g1 = GraphFrame(vertices, edges)

Command took 0.07 seconds -- by x16140818@student.ncirl.ie at 24/06/2020, 11:07:49 on unknown cluster

```

Figure 111: GraphFrame

3. Display outDegree and create DataFrame of id & outDegree

```

Cmd 38
1 display(g1.outDegrees)

Showing the first 1000 rows.

+----+-----+
|id          |outDegree|
+----+-----+
|1|wDheoqPISThML4pShhg3g|57|
|2|ckDvozHDR5hWgrDRTMYZkQ|135|
|3|soG9o5PqmXilKJHpAB777A|279|
|4|tg1Eh5J9iqH5Y0ycb1bejw|123|
|5|jO44Apni7JZVVk4HQ60tA|175|
|6|4ZfcCa4m5RWvO4EFzfYm1A|3436|
+----+-----+

Command took 1.22 minutes -- by x16140818@student.ncirl.ie at 24/06/2020, 11:07:58 on unknown cluster

Cmd 39
1 outd = g1.outDegrees.select("id","outDegree")

outd: pyspark.sql.dataframe.DataFrame = [id: string, outDegree: Integer]

Command took 0.06 seconds -- by x16140818@student.ncirl.ie at 24/06/2020, 11:09:50 on unknown cluster

```

Figure 112: outDegree

4. Run the PageRank algorithm. Create DataFrame of id & PageRank. Merge outDegree and PageRank DataFrames and rename id to user_id as required by the algorithm.

```
Cmd 40
1 rank = g1.pageRank(resetProbability=0.15, tol=0.01)

Command took 22.47 minutes -- by x16140818@student.ncirl.ie at 24/06/2020, 11:09:53 on unknown cluster

Cmd 41
1 pr = rank.vertices.select("id","pagerank")

pr: pyspark.sql.dataframe.DataFrame = [id: string, pagerank: double]

Command took 0.85 seconds -- by x16140818@student.ncirl.ie at 24/06/2020, 11:10:06 on unknown cluster

Cmd 42
1 grapha = outd.join(pr, on="id", how="inner")

grapha: pyspark.sql.dataframe.DataFrame = [id: string, outDegree: integer ... 1 more fields]

Command took 0.86 seconds -- by x16140818@student.ncirl.ie at 24/06/2020, 11:10:12 on unknown cluster

Cmd 43
1 network = grapha.withColumnRenamed("id","user_id")

network: pyspark.sql.dataframe.DataFrame = [user_id: string, outDegree: integer ... 1 more fields]

Command took 0.84 seconds -- by x16140818@student.ncirl.ie at 24/06/2020, 11:10:15 on unknown cluster
```

Figure 113: PageRank

5. Select the necessary fields from User DataFrame. Rename fields to ensure no duplication later. Merge with previous DataFrame

```
Cmd 44
1 uuser = user.select("average_stars","review_count","useful","user_id")

uuser: pyspark.sql.dataframe.DataFrame = [average_stars: double, review_count: long ... 2 more fields]

Command took 0.86 seconds -- by x16140818@student.ncirl.ie at 24/06/2020, 11:42:12 on unknown cluster

Cmd 45
1 uu = uuser.withColumnRenamed("useful","useful_sent").withColumnRenamed("average_stars","useful_avg_stars")

uu: pyspark.sql.dataframe.DataFrame = [useful_avg_stars: double, review_count: long ... 2 more fields]

Command took 0.85 seconds -- by x16140818@student.ncirl.ie at 24/06/2020, 11:42:16 on unknown cluster

Cmd 46
1 social_network = network.join(uu, on="user_id", how="inner")

social_network: pyspark.sql.dataframe.DataFrame = [user_id: string, outDegree: integer ... 4 more fields]

Command took 0.85 seconds -- by x16140818@student.ncirl.ie at 24/06/2020, 11:42:19 on unknown cluster
```

Figure 114: Social Network DataFrame

6. Select necessary fields from Business DataFrame. Again, rename fields to prevent naming duplication

```
Cmd 47
1 bu = business.select("business_id","stars")

▶ bu: pyspark.sql.dataframe.DataFrame = [business_id: string, stars: double]
Command took 0.05 seconds -- by x16140818@student.ncirl.ie at 24/06/2020, 11:42:24 on unknown cluster

Cmd 48
1 buser = bu.withColumnRenamed("stars","business_avg_star")

▶ buser: pyspark.sql.dataframe.DataFrame = [business_id: string, business_avg_star: double]
Command took 0.05 seconds -- by x16140818@student.ncirl.ie at 24/06/2020, 11:42:31 on unknown cluster
```

Figure 115: Business Average Star Rating

7. Select necessary fields from Review DataFrame. Remove spacing and unnecessary characters from text. Final check to see if text is correctly formatted

```
Cmd 49
1 ru = review.select("review_id","user_id","business_id","stars","date","text","useful")

▶ ru: pyspark.sql.dataframe.DataFrame = [review_id: string, user_id: string ... 5 more fields]
Command took 0.07 seconds -- by x16140818@student.ncirl.ie at 24/06/2020, 11:42:52 on unknown cluster

Cmd 50
1 ru2 = ru.withColumn("text", regexp_replace(col("text"), "[\n\r]", " "))

▶ ru2: pyspark.sql.dataframe.DataFrame = [review_id: string, user_id: string ... 5 more fields]
Command took 0.12 seconds -- by x16140818@student.ncirl.ie at 24/06/2020, 11:42:57 on unknown cluster

Cmd 51
1 ru3 = ru2.withColumn("text", trim(ru2.text))

▶ ru3: pyspark.sql.dataframe.DataFrame = [review_id: string, user_id: string ... 5 more fields]
Command took 0.10 seconds -- by x16140818@student.ncirl.ie at 23/06/2020, 17:53:51 on unknown cluster
```

Figure 116: Select Review Fields

```
Cmd 52
1 ru3 = ru2.withColumn("text", regexp_replace(col("text"), "[()&*-,]", ""))

▶ ru3: pyspark.sql.dataframe.DataFrame = [review_id: string, user_id: string ... 5 more fields]
Command took 0.06 seconds -- by x16140818@student.ncirl.ie at 24/06/2020, 11:43:15 on unknown cluster
```

Figure 117: Remove Characters

```

Cmd 53
1 ru3.select("text").show(5,False)
-----
|As someone who has worked with many museums I was eager to visit this gallery on my most recent trip to Las Vegas. When I saw they would be showing infamous eggs of the House of Faberge from the Virginia Museum of Fine Arts VMFA I knew I had to go! Tucked away near the gelateria and the garden the Gallery is pretty much hidden from view. It's what real estate agents would call "cozy" or "charming" - basically any euphemism for small. That being said you can still see wonderful art at a gallery of any size so why the two s you ask? Let me tell you: pricing for this while relatively inexpensive for a Las Vegas attraction is completely over the top. For the space and the amount of art you can fit in there it is a bit much. it's not kid friendly at all. Seriously don't bring them. the security is not trained properly for the show. When the curating and design teams collaborate for exhibitions there is a definite flow. That means visitors should view the art in a certain sequence whether it be by historical period or cultural significance this is how audio guides are usually developed. When I arrived in the gallery I could not tell where to start and security was certainly not helpful. I was told to "just look around" and "do whatever." At such a fine institution I find the lack of knowledge and respect for the art appalling.
|
|I am actually horrified this place is still in business. My 3 year old son needed a haircut this past summer and the lure of the $7 kids cut signs got me in the door. We had to wait a few minutes as both stylists were working on people. The decor in this place is total garbage. It is so tacky. The sofa they had at the time was a pleather sofa with giant holes in it. And my son noticed ants crawling all over the floor and the furniture. It was disgusting and I should have walked out then. Actually I should have turned around and walked out upon entering but I didn't. So the older black male stylist finishes the haircut he was doing and it's our turn. I tell him I want a #2 clipper around the back and sides and then hand cut the top into a standard boys cut. Really freaking simple right? WRONG! Rather than use the clippers and go up to actually cut the hair he went down. Using it moving downward doesn't cut hair it just rubs against it. How does this man who has an alleged cosmetology license not know how to use a set of freaking clippers??? I realized almost immediately that he had no idea what he was doing. No idea at all. After about 10 minutes of watching this guy stumble through it I said "you know what? That's fine." paid and left. ALL I wanted to do was get out of that scummy joint and take my son to a real haircut place. Bottom line: DO NOT GO HERE. RUN THE OTHER WAY!!!!!!
|I love Deagan's. I do. I really do. The atmosphere is cozy and festive. The shrimp tacos and house fries are my standbys. The fries are sometimes good and sometimes great and the spicy dipping sauce they come with is to die for. The beer list is amazing and the cocktails are great. The prices are mid-level so it's not a cheap dive you can go to every week but rather a
Command took 1.34 seconds -- by x16140818@student.ncirl.ie at 24/06/2020, 11:44:29 on unknown cluster

```

Figure 118: Check Text

```

Cmd 54
1 ru4 = ru3.withColumn("text", regexp_replace(col("text"), " ", ""))
ru4: pyspark.sql.dataframe.DataFrame = [review_id: string, user_id: string ... 5 more fields]
Command took 0.06 seconds -- by x16140818@student.ncirl.ie at 24/06/2020, 11:45:57 on unknown cluster

Cmd 55
1 ru5 = ru4.withColumn("text", regexp_replace(col("text"), " ", ""))
ru5: pyspark.sql.dataframe.DataFrame = [review_id: string, user_id: string ... 5 more fields]
Command took 0.05 seconds -- by x16140818@student.ncirl.ie at 24/06/2020, 11:46:52 on unknown cluster

```

Figure 119: Remove Whitespaces

```

Cmd 56
1 ru5.select("text").show(5,False)
-----
|As someone who has worked with many museums I was eager to visit this gallery on my most recent trip to Las Vegas. When I saw they would be showing infamous eggs of the House of Faberge from the Virginia Museum of Fine Arts VMFA I knew I had to go! Tucked away near the gelateria and the garden the Gallery is pretty much hidden from view. It's what real estate agents would call "cozy" or "charming" - basically any euphemism for small. That being said you can still see wonderful art at a gallery of any size so why the two s you ask? Let me tell you: pricing for this while relatively inexpensive for a Las Vegas attraction is completely over the top. For the space and the amount of art you can fit in there it is a bit much. it's not kid friendly at all. Seriously don't bring them. the security is not trained properly for the show. When the curating and design teams collaborate for exhibitions there is a definite flow. That means visitors should view the art in a certain sequence whether it be by historical period or cultural significance this is how audio guides are usually developed. When I arrived in the gallery I could not tell where to start and security was certainly not helpful. I was told to "just look around" and "do whatever." At such a fine institution I find the lack of knowledge and respect for the art appalling.
|
|I am actually horrified this place is still in business. My 3 year old son needed a haircut this past summer and the lure of the $7 kids cut signs got me in the door. We had to wait a few minutes as both stylists were working on people. The decor in this place is total garbage. It is so tacky. The sofa they had at the time was a pleather sofa with giant holes in it. And my son noticed ants crawling all over the floor and the furniture. It was disgusting and I should have walked out then. Actually I should have turned around and walked out upon entering but I didn't. So the older black male stylist finishes the haircut he was doing and it's our turn. I tell him I want a #2 clipper around the back and sides and then hand cut the top into a standard boys cut. Really freaking simple right? WRONG! Rather than use the clippers and go up to actually cut the hair he went down. Using it moving downward doesn't cut hair it just rubs against it. How does this man who has an alleged cosmetology license not know how to use a set of freaking clippers??? I realized almost immediately that he had no idea what he was doing. No idea at all. After about 10 minutes of watching this guy stumble through it I said "you know what? That's fine." paid and left. ALL I wanted to do was get out of that scummy joint and take my son to a real haircut place. Bottom line: DO NOT GO HERE. RUN THE OTHER WAY!!!!!!
|I love Deagan's. I do. I really do. The atmosphere is cozy and festive. The shrimp tacos and house fries are my standbys. The fries are sometimes good and sometimes great
Command took 0.48 seconds -- by x16140818@student.ncirl.ie at 24/06/2020, 11:47:04 on unknown cluster

```

Figure 120: Recheck Text

10. Calculate average review rating.

```
Cmd 62  
1 review_avg = ru12.groupBy("business_id").agg(avg("stars"))  
  
▶ review_avg: pyspark.sql.dataframe.DataFrame = [business_id: string, avg(stars): double]  
Command took 0.06 seconds -- by x1614081@student.ncirl.ie at 24/06/2020, 11:50:41 on unknown cluster
```

Figure 124: Review Average Count

11. Merge ru12 with average review rating

```
Cmd 63  
1 social_data = ru12.join(review_avg, on="business_id", how="inner")  
  
▶ social_data: pyspark.sql.dataframe.DataFrame = [business_id: string, review_id: string ... 10 more fields]  
Command took 0.09 seconds -- by x1614081@student.ncirl.ie at 24/06/2020, 11:50:46 on unknown cluster
```

Figure 125: Merge DataFrames

12. Merge social data with social network DataFrames

```
Cmd 64  
1 usefulness_dataset = social_data.join(social_network, on="user_id", how="inner")  
  
▶ usefulness_dataset: pyspark.sql.dataframe.DataFrame = [user_id: string, business_id: string ... 15 more fields]  
Command took 0.05 seconds -- by x1614081@student.ncirl.ie at 24/06/2020, 11:50:49 on unknown cluster
```

Figure 126: Merge DataFrame

13. Remove timestamp from date field

```
Cmd 65  
1 usefultdata = usefulness_dataset.withColumn("year", split(col("date"), " ")[0])  
  
▶ usefultdata: pyspark.sql.dataframe.DataFrame = [user_id: string, business_id: string ... 16 more fields]  
Command took 0.06 seconds -- by x1614081@student.ncirl.ie at 24/06/2020, 11:50:52 on unknown cluster
```

Figure 127: Remove Field

14. Calculate deviation to average business rating, deviation to average user rating and rename avg(stars) field to review_avg_star.

```
Cmd 66
1 usefu1 = usefu1data.withColumn("abs_Business_star", (abs(usefu1data["avg(stars)"] - usefu1data["business_avg_star"])))
▶ usefu1: pyspark.sql.dataframe.DataFrame = [user_id: string, business_id: string ... 17 more fields]
Command took 0.06 seconds -- by x16140818@student.ncirl.ie at 24/06/2020, 11:51:02 on unknown cluster

Cmd 67
1 usefu2 = usefu1.withColumn("abs_user_star", (abs(usefu1data["avg(stars)"] - usefu1data["usefu1_avg_stars"])))
▶ usefu2: pyspark.sql.dataframe.DataFrame = [user_id: string, business_id: string ... 18 more fields]
Command took 0.05 seconds -- by x16140818@student.ncirl.ie at 24/06/2020, 11:51:06 on unknown cluster

Cmd 68
1 usefu3 = usefu2.withColumnRenamed("avg(stars)", "review_avg_star")
▶ usefu3: pyspark.sql.dataframe.DataFrame = [user_id: string, business_id: string ... 18 more fields]
Command took 0.05 seconds -- by x16140818@student.ncirl.ie at 24/06/2020, 11:51:08 on unknown cluster
```

Figure 128: Deviation to Average Review Star Rating

15. Calculate Question Count and Exclamation Count

```
Cmd 69
1 usefu4 = usefu3.withColumn('questionCount', size(split(col("text"),r"[?]") - 1)
▶ usefu4: pyspark.sql.dataframe.DataFrame = [user_id: string, business_id: string ... 19 more fields]
Command took 0.06 seconds -- by x16140818@student.ncirl.ie at 24/06/2020, 11:51:19 on unknown cluster

Cmd 70
1 usefu5 = usefu4.withColumn('exclamationCount', size(split(col("text"),r"[!]") - 1)
▶ usefu5: pyspark.sql.dataframe.DataFrame = [user_id: string, business_id: string ... 20 more fields]
Command took 0.07 seconds -- by x16140818@student.ncirl.ie at 24/06/2020, 11:51:22 on unknown cluster
```

Figure 129: Question and Exclamation Count

16. Drop unnecessary fields

```
Cmd 71
1 usefu6 = usefu5.drop("stars", "date", "business_avg_star", "usefu1_avg_star", "text")
▶ usefu6: pyspark.sql.dataframe.DataFrame = [user_id: string, business_id: string ... 16 more fields]
Command took 0.05 seconds -- by x16140818@student.ncirl.ie at 24/06/2020, 11:51:45 on unknown cluster
```

Figure 130: Remove Field

17. Convert the year field from string to timestamp and display date range of reviews

```
Cmd 72
1 usefu18 = usefu16.withColumn('review_date', to_date(unix_timestamp(col('year'), 'yyyy-mm-dd').cast("timestamp")))
2

usefu18: pyspark.sql.dataframe.DataFrame = [user_id: string, business_id: string ... 17 more fields]
Command took 0.07 seconds -- by x16140818@student.ncirl.ie at 24/06/2020, 11:51:49 on unknown cluster

Cmd 73
1 display(usefu18.select("review_date"))
```

	review_date
1	2014-01-21
2	2012-01-06
3	2014-01-19
4	2014-01-28
5	2018-01-12
6	2017-01-01
7	2016-01-29

Showing the first 1000 rows.

Command took 5.38 minutes -- by x16140818@student.ncirl.ie at 24/06/2020, 11:51:52 on unknown cluster

Figure 131: Convert Field

18. Calculate average sentence count and drop more unnecessary fields. All fields now have been created for feature selection

```
Cmd 74
1 usefu19 = usefu18.withColumn("avg_sentence_length", (usefu18.wordCount / usefu18.sentenceCount))

usefu19: pyspark.sql.dataframe.DataFrame = [user_id: string, business_id: string ... 18 more fields]
Command took 0.06 seconds -- by x16140818@student.ncirl.ie at 24/06/2020, 11:58:50 on unknown cluster

Cmd 75
1 usefu110 = usefu19.drop("usefu1_sent", "year", "user_id", "business_id", "usefu1_avg_stars")

usefu110: pyspark.sql.dataframe.DataFrame = [review_id: string, usefu1: long ... 13 more fields]
Command took 0.05 seconds -- by x16140818@student.ncirl.ie at 24/06/2020, 11:58:53 on unknown cluster
```

Figure 132: Sentence Count and Remove Field

19. The DataFrame is next filtered for the four-month period

```
Cmd 76
1 usefu111 = usefu110.filter(usefu110["review_date"] >= ("2018-11-01")).filter(usefu110["review_date"] <= ("2019-01-31"))

usefu111: pyspark.sql.dataframe.DataFrame = [review_id: string, usefu1: long ... 13 more fields]
Command took 0.13 seconds -- by x16140818@student.ncirl.ie at 24/06/2020, 11:59:05 on unknown cluster
```

Figure 133: Filter Date

20. A User Defined Function is created to convert reviews with a useful count greater or equal to 5 to 1 and less than 5 to 0. The new label field is converted to an integer

```
Cmd 77
1 def convert_rating(useful):
2   if useful >=5:
3     return 1
4   else:
5     return 0

Command took 0.05 seconds -- by x16140818@student.ncirl.ie at 24/06/2020, 11:59:09 on unknown cluster

Cmd 78
1 useful_convert = udf(lambda x: convert_rating(x))

Command took 0.04 seconds -- by x16140818@student.ncirl.ie at 24/06/2020, 11:59:12 on unknown cluster

Cmd 79
1 useful12 = useful11.withColumn("label", useful_convert("useful"))

▶ useful12: pyspark.sql.dataframe.DataFrame = [review_id: string, useful: long ... 14 more fields]
Command took 0.08 seconds -- by x16140818@student.ncirl.ie at 24/06/2020, 11:59:15 on unknown cluster
```

Figure 134: User Defined Function

```
Cmd 80
1 useful13 = useful12.withColumn('label', useful12["label"].cast(IntegerType()))

▶ useful13: pyspark.sql.dataframe.DataFrame = [review_id: string, useful: long ... 14 more fields]
Command took 0.05 seconds -- by x16140818@student.ncirl.ie at 24/06/2020, 11:59:19 on unknown cluster
```

Figure 135: Cast to Integer

21. The distribution of the 1 & 0's was counted, and stratified sampling was applied to achieve a DataFrame of 21,361 reviews. Again, as stratified sampling is an approximation it takes 2 calculations to achieve the closet possible outcome to the desired DataFrame size

```
Cmd 82
1 display(useful13.groupBy("label").count())
```

	label	count
1	1	39088
2	0	1176748

Showing all 2 rows.

```
Command took 5.72 minutes -- by x16140818@student.ncirl.ie at 24/06/2020, 12:10:35 on unknown cluster
```

Figure 136: Star Count

```

Cmd 83
1 useful_score = useful13.sampleBy("label", fractions={1: 0.54648485468686, 0: 0.0181525696240826}, seed=435)

useful_score: pyspark.sql.dataframe.DataFrame = [review_id: string, usefu: long ... 14 more fields]
Command took 0.13 seconds -- by x1614081@student.ncirl.ie at 24/06/2020, 12:18:04 on unknown cluster

Cmd 84
1 display(useful_score.groupBy("label").count())



|   | label | count |
|---|-------|-------|
| 1 | 1     | 21325 |
| 2 | 0     | 21216 |



Showing all 2 rows.

Command took 6.13 minutes -- by x1614081@student.ncirl.ie at 24/06/2020, 12:18:07 on unknown cluster

```

Figure 137: Starfield Sampling Attempt 1

```

Cmd 85
1 useful_score1 = useful13.sampleBy("label", fractions={1: 0.54740740825163, 0: 0.0182766327177616}, seed=435)

useful_score1: pyspark.sql.dataframe.DataFrame = [review_id: string, usefu: long ... 14 more fields]
Command took 0.15 seconds -- by x1614081@student.ncirl.ie at 24/06/2020, 12:55:55 on unknown cluster

Cmd 86
1 display(useful_score1.groupBy("label").count())



|   | label | count |
|---|-------|-------|
| 1 | 1     | 21347 |
| 2 | 0     | 21358 |



Showing all 2 rows.

Command took 5.88 minutes -- by x1614081@student.ncirl.ie at 24/06/2020, 12:56:03 on unknown cluster

```

Figure 138: Starfield Sampling Attempt 2

22. Drop unnecessary fields

```

Cmd 87
1 useful_score2 = useful_score1.drop("useful", "review_date")

useful_score2: pyspark.sql.dataframe.DataFrame = [review_id: string, wordCount: integer ... 12 more fields]
Command took 0.05 seconds -- by x1614081@student.ncirl.ie at 24/06/2020, 13:05:41 on unknown cluster

```

Figure 139: Remove Field

23. Select independent variables and convert to Pandas DataFrame

```
Cmd 88
1 independent = useful_score2.select(useful_score2.columns[1:13])

independent: pyspark.sql.dataframe.DataFrame = [wordCount: integer, sentenceCount: integer ... 10 more fields]
Command took 0.09 seconds -- by x1614881@student.ncirl.ie at 24/06/2020, 13:05:58 on unknown cluster

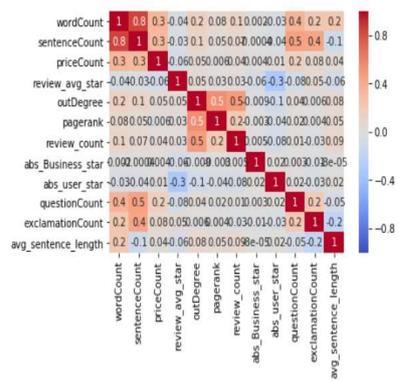
Cmd 89
1 independent_data = independent.toPandas()

Command took 6.29 minutes -- by x1614881@student.ncirl.ie at 24/06/2020, 13:06:04 on unknown cluster
```

Figure 140: Independent Variables

24. Plot Correlation with Seaborn Heatmap for the independent variables. Sentence Count and Word Count are highly correlated

```
Cmd 91
1 plt.figure(figsize=(15, 20))
2 sns.heatmap(independent_data.corr(), annot = True, vmin=-1, vmax=1, center= 0, cmap= 'coolwarm', fmt='.1g')
```



	wordCount	sentenceCount	priceCount	review_avg_star	outDegree	pagerank	review_count	abs_Business_star	abs_user_star	questionCount	exclamationCount	avg_sentence_length
wordCount	1	0.8	0.3	-0.04	0.2	0.08	0.10	0.00	0.03	0.4	0.2	0.2
sentenceCount	0.8	1	0.3	-0.03	0.1	0.05	0.00	0.00	0.04	0.5	0.4	-0.1
priceCount	-0.3	0.3	1	-0.06	0.05	0.00	0.00	0.00	0.01	0.2	0.08	0.04
review_avg_star	-0.04	0.03	0.0	1	0.05	0.03	0.03	0.06	-0.3	-0.08	0.05	-0.06
outDegree	0.2	0.1	0.05	0.0	1	0.5	0.5	0.00	0.90	1.0	0.40	0.00
pagerank	-0.08	0.05	0.00	0.0	0.5	1	0.2	0.00	0.30	0.40	0.00	0.05
review_count	-0.1	0.07	0.04	0.0	0.5	0.2	1	0.00	0.50	0.80	0.10	0.30
abs_Business_star	0.00	0.00	0.00	0.0	0.00	0.00	0.00	1	0.00	0.00	0.00	0.00
abs_user_star	-0.03	0.04	0.01	-0.3	-0.1	-0.04	0.80	0.2	1	0.02	0.03	0.02
questionCount	-0.4	0.5	0.2	-0.08	0.04	0.02	0.30	0.00	0.02	1	0.2	-0.05
exclamationCount	-0.2	0.4	0.08	0.05	0.00	0.00	0.40	0.30	0.10	0.3	1	-0.2
avg_sentence_length	-0.2	-0.1	0.04	0.06	0.08	0.05	0.09	-0.00	0.20	0.05	-0.2	1

Command took 1.15 seconds -- by x1614881@student.ncirl.ie at 24/06/2020, 13:23:42 on unknown cluster

Figure 141: Correlation Heat Map

25. Check correlation between dependent variable and independent variables

```

Cmd 92
1 dependent_corr = useful_score2.select(useful_score2.columns[1:14])

dependent_corr: pyspark.sql.dataframe.DataFrame = [wordCount: integer, sentenceCount: integer ... 11 more fields]
Command took 0.89 seconds -- by x16148818@student.ncirl.ie at 24/06/2020, 13:24:27 on unknown cluster

Cmd 93
1 for i in dependent_corr.columns:
2     if not( isinstance(dependent_corr.select(i).take(1)[0][0], six.string_types)):
3         print( "Correlation to Label for ", i, dependent_corr.stat.corr('label',i))

Correlation to Label for wordCount 0.37859007261711686
Correlation to Label for sentenceCount 0.3403632569098586
Correlation to Label for priceCount 0.15729660179779512
Correlation to Label for review_avg_star -0.025294574712295716
Correlation to Label for outDegree 0.3764977664518094
Correlation to Label for pagerank 0.17735386909378298
Correlation to Label for review_count 0.28741146539021306
Correlation to Label for abs_Business_star 0.012987010501318596
Correlation to Label for abs_user_star -0.10607260657028005
Correlation to Label for questionCount 0.13423651830201377
Correlation to Label for exclamationCount 0.09361952973792916
Correlation to Label for avg_sentence_length 0.09438460600745945
Correlation to Label for label 1.0
Command took 2.50 hours -- by x16148818@student.ncirl.ie at 24/06/2020, 13:24:38 on unknown cluster

```

Figure 142: Correlation Dependent Variable

26. Check correlation if Sentence Count is dropped

```

Cmd 94
1 final_data_useful = useful_score2.drop("sentenceCount", "label", "review_id")

final_data_useful: pyspark.sql.dataframe.DataFrame = [wordCount: integer, priceCount: integer ... 9 more fields]
Command took 0.06 seconds -- by x16148818@student.ncirl.ie at 24/06/2020, 16:01:03 on unknown cluster

Cmd 95
1 independent_fdu = final_data_useful.toPandas()

Command took 6.02 minutes -- by x16148818@student.ncirl.ie at 24/06/2020, 16:01:06 on unknown cluster

```

Figure 143: Remove Sentence Count



Figure 144: Correlation Heat Map

27. Sentence Count was dropped, and all fields are checked for missing values

```
Cmd 97
1 final_data = useful_score2.drop("sentenceCount")

final_data: pyspark.sql.dataframe.DataFrame = [review_id: string, wordCount: integer ... 11 more fields]
Command took 0.04 seconds -- by x16140818@student.ncirl.ie at 24/06/2020, 20:33:26 on unknown cluster

Cmd 98
1 final_data.select([count(when(isnan(c), c)).alias(c) for c in final_data.columns]).toPandas().head()

Out[306]:
  review_id  wordCount  priceCount  review_avg_star  outDegree  pagerank  review_count  abs_Business_star  abs_user_star  questionCount  exclamationCount  avg_sentence_length  label
0          0          0          0          0          0          0          0          0          0          0          0          0          0
```

Figure 145: Sentence Count and Null Values

28. Combine all independent variables to one column

```
Cmd 99
1 numeric_features =
  final_data.select("wordCount", "priceCount", "review_avg_star", "outDegree", "pagerank", "review_count", "abs_Business_star", "abs_user_star", "questionCount", "exclamationCount", "avg_sentence_length")
2
3

numeric_features: pyspark.sql.dataframe.DataFrame = [wordCount: integer, priceCount: integer ... 9 more fields]
Command took 0.09 seconds -- by x16140818@student.ncirl.ie at 24/06/2020, 20:33:51 on unknown cluster

Cmd 100
1 nf = numeric_features.columns

Command took 0.04 seconds -- by x16140818@student.ncirl.ie at 24/06/2020, 20:33:55 on unknown cluster
```

Figure 146: Combine to Single Column

29. Apply VectorAssembler and StandardScaler to the independent variables

```
Cmd 101
1 va = VectorAssembler(inputCols=nf, outputCol="features", handleInvalid="skip")

Command took 0.05 seconds -- by x16140818@student.ncirl.ie at 24/06/2020, 20:34:08 on unknown cluster

Cmd 102
1 lr1 = va.transform(final_data)

lr1: pyspark.sql.dataframe.DataFrame = [review_id: string, wordCount: integer ... 12 more fields]
Command took 0.06 seconds -- by x16140818@student.ncirl.ie at 24/06/2020, 20:34:24 on unknown cluster

Cmd 103
1 scalerII = StandardScaler(inputCol="features", outputCol="scaledFeatures")

Command took 0.05 seconds -- by x16140818@student.ncirl.ie at 24/06/2020, 20:34:43 on unknown cluster

Cmd 104
1 lr2 = scalerII.fit(lr1).transform(lr1)

lr2: pyspark.sql.dataframe.DataFrame = [review_id: string, wordCount: integer ... 13 more fields]
Command took 19.00 minutes -- by x16140818@student.ncirl.ie at 24/06/2020, 20:37:04 on unknown cluster
```

Figure 147: VectorAssembler and StandardScaler

30. Split DataFrame into train and test

```
Cmd 105
1 train, test = lr2.randomSplit([0.8, 0.2], seed=346)

▶ train: pyspark.sql.dataframe.DataFrame = [review_id: string, wordCount: integer ... 13 more fields]
▶ test: pyspark.sql.dataframe.DataFrame = [review_id: string, wordCount: integer ... 13 more fields]

Command took 0.06 seconds -- by x16140818@student.ncirl.ie at 24/06/2020, 20:37:09 on unknown cluster
```

Figure 148: Train and Test

31. Apply Random Forest to Train DataFrame and select most important features. From Feature Importance four features make up 91% of the outcome

```
Cmd 106
1 rf = RandomForestClassifier(labelCol="label", featuresCol="scaledFeatures", numTrees=20)

Command took 0.85 seconds -- by x16140818@student.ncirl.ie at 24/06/2020, 20:37:11 on unknown cluster

Cmd 107
1 feature_selection = rf.fit(train)

Command took 13.51 minutes -- by x16140818@student.ncirl.ie at 24/06/2020, 20:37:14 on unknown cluster

Cmd 108
1 feature_selection.featureImportances

Out[270]: SparseVector(11, {0: 0.1814, 1: 0.0099, 2: 0.0065, 3: 0.2817, 4: 0.1066, 5: 0.3996, 6: 0.0004, 7: 0.0003, 8: 0.0118, 9: 0.0007, 10: 0.0012})

Command took 0.05 seconds -- by x16140818@student.ncirl.ie at 24/06/2020, 20:37:16 on unknown cluster
```

Figure 149: Random Forrest Feature Importance

32. Select 4 important features. Again, apply VectorAssembler and StandardScaler to original DataFrame of 21,361 reviews

```
Cmd 109
1 numeric_features2 = final_data.select("wordCount", "outDegree", "pagerank", "review_count")

▶ numeric_features2: pyspark.sql.dataframe.DataFrame = [wordCount: integer, outDegree: integer ... 2 more fields]

Command took 0.06 seconds -- by x16140818@student.ncirl.ie at 24/06/2020, 20:59:10 on unknown cluster

Cmd 110
1 nf_selected = numeric_features2.columns

Command took 0.04 seconds -- by x16140818@student.ncirl.ie at 24/06/2020, 20:59:13 on unknown cluster

Cmd 111
1 va_selected = VectorAssembler(inputCols=nf_selected, outputCol="features", handleInvalid="skip")

Command took 0.05 seconds -- by x16140818@student.ncirl.ie at 24/06/2020, 20:59:16 on unknown cluster
```

Figure 150: VectorAssembler

```
Cmd 112
1 lr_selected = va_selected.transform(final_data)

▶ lr_selected: pyspark.sql.dataframe.DataFrame = [review_id: string, wordCount: integer ... 12 more fields]
Command took 0.12 seconds -- by x16140818@student.ncirl.ie at 24/06/2020, 20:59:35 on unknown cluster

Cmd 113
1 lr2_selected = scalerII.fit(lr_selected).transform(lr_selected)

▶ lr2_selected: pyspark.sql.dataframe.DataFrame = [review_id: string, wordCount: integer ... 13 more fields]
Command took 5.94 minutes -- by x16140818@student.ncirl.ie at 24/06/2020, 20:59:40 on unknown cluster
```

Figure 151: StandardScaler

33. Split into Train & Test and apply Logistic Regression (LR). Print Area Under a Receiver Operating Characteristic Curve (AUC). The AUC is the default evaluation metric in PySpark for Binary Classification

```
Cmd 114
1 train2, test2 = lr2_selected.randomSplit([0.8, 0.2], seed=348)

▶ train2: pyspark.sql.dataframe.DataFrame = [review_id: string, wordCount: integer ... 13 more fields]
▶ test2: pyspark.sql.dataframe.DataFrame = [review_id: string, wordCount: integer ... 13 more fields]
Command took 0.05 seconds -- by x16140818@student.ncirl.ie at 24/06/2020, 20:59:50 on unknown cluster

Cmd 115
1 logistic = LogisticRegression(labelCol="label", featuresCol="scaledFeatures",maxIter=10)

Command took 0.05 seconds -- by x16140818@student.ncirl.ie at 24/06/2020, 20:59:52 on unknown cluster

Cmd 116
1 cmodel_lr = logistic.fit(train2)

Command took 6.93 minutes -- by x16140818@student.ncirl.ie at 24/06/2020, 20:59:55 on unknown cluster
```

Figure 152: Logistic Regression (1 of 2)

```
Cmd 116
1 cmodel_lr = logistic.fit(train2)

Command took 6.93 minutes -- by x16140818@student.ncirl.ie at 24/06/2020, 20:59:55 on unknown cluster

Cmd 117
1 ctrain_predict = cmodel_lr.transform(train2)

▶ ctrain_predict: pyspark.sql.dataframe.DataFrame = [review_id: string, wordCount: integer ... 16 more fields]
Command took 0.07 seconds -- by x16140818@student.ncirl.ie at 24/06/2020, 21:00:11 on unknown cluster

Cmd 118
1 ctest_predict = cmodel_lr.transform(test2)

▶ ctest_predict: pyspark.sql.dataframe.DataFrame = [review_id: string, wordCount: integer ... 16 more fields]
Command took 0.07 seconds -- by x16140818@student.ncirl.ie at 24/06/2020, 21:11:21 on unknown cluster
```

Figure 153: Logistic Regression (2 of 2)

```
Cmd 119
1 cevaluator_lr = BinaryClassificationEvaluator(rawPredictionCol="rawPrediction")

Command took 0.06 seconds -- by x16140818@student.ncirl.ie at 24/06/2020, 21:11:22 on unknown cluster

Cmd 120
1 print("The area under ROC for train set is {}".format(cevaluator_lr.evaluate(ctrain_predict)))
2 print("The area under ROC for test set is {}".format(cevaluator_lr.evaluate(ctest_predict)))

The area under ROC for train set is 0.886199917984726
The area under ROC for test set is 0.884402684652284

Command took 12.30 minutes -- by x16140818@student.ncirl.ie at 24/06/2020, 21:11:24 on unknown cluster
```

Figure 154: Logistic Regression Results (1 of 5)

34. Calculate Accuracy for LR model

```
Cmd 121
1 accuracy_val = ctest_predict.filter(ctest_predict.label == ctest_predict.prediction).count() /
float(ctest_predict.count())

Command took 12.23 minutes -- by x16140818@student.ncirl.ie at 24/06/2020, 21:11:26 on unknown cluster

Cmd 122
1 print(" The Accuracy Score is: ", accuracy_val)

The Accuracy Score is: 0.8088373183309893

Command took 0.05 seconds -- by x16140818@student.ncirl.ie at 24/06/2020, 21:11:28 on unknown cluster
```

Figure 155: Logistic Regression Results (2 of 5)

35. Calculate True Positive (TP), True Negative (TN), False Positive (FP) and False Negative (FN)

```
Cmd 123
1 ctp = ctest_predict[(ctest_predict.label == 1) & (ctest_predict.prediction == 1)].count()

Command took 6.21 minutes -- by x16140818@student.ncirl.ie at 24/06/2020, 21:11:29 on unknown cluster

Cmd 124
1 ctn = ctest_predict[(ctest_predict.label == 0) & (ctest_predict.prediction == 0)].count()

Command took 6.20 minutes -- by x16140818@student.ncirl.ie at 24/06/2020, 21:11:32 on unknown cluster

Cmd 125
1 cfp = ctest_predict[(ctest_predict.label == 0) & (ctest_predict.prediction == 1)].count()

Command took 6.39 minutes -- by x16140818@student.ncirl.ie at 24/06/2020, 21:11:33 on unknown cluster
```

Figure 156: Logistic Regression Results (3 of 5)

```
Cmd 126
1 cfn = ctest_predict[(ctest_predict.label == 1) & (ctest_predict.prediction == 0)].count()

Command took 6.37 minutes -- by x16140818@student.ncirl.ie at 24/06/2020, 21:11:35 on unknown cluster

Cmd 127
1 print ("True Positives: ", ctp)
2 print ("True Negatives: ", ctn)
3 print ("False Positives: ", cfp)
4 print ("False Negatives: ", cfn)
5 print ("Total: ", ctest_predict.count())

True Positives: 3154
True Negatives: 3747
False Positives: 471
False Negatives: 1160
Total: 8532

Command took 6.50 minutes -- by x16140818@student.ncirl.ie at 24/06/2020, 21:11:37 on unknown cluster
```

Figure 157: Logistic Regression Results (3 of 5)

36. Calculate Recall, Precision and F1 score

```
Cmd 128
1 r1 = float(ctp)/(ctp + cfn)
2 print("Recall: ", r1)

Recall: 0.7311080203987019

Command took 0.04 seconds -- by x16140818@student.ncirl.ie at 24/06/2020, 21:11:40 on unknown cluster

Cmd 129
1 p1 = float(ctp)/(ctp + cfp)
2 print("Precision: ", p1)

Precision: 0.8700689655172413

Command took 0.04 seconds -- by x16140818@student.ncirl.ie at 24/06/2020, 21:11:42 on unknown cluster
```

Figure 158: Logistic Regression Results (4 of 5)

```
Cmd 130
1 F1 = 2*((r1*p1)/(r1+p1))
2 print("F1 Score: ", F1)

F1 Score: 0.7945585086282907

Command took 0.05 seconds -- by x16140818@student.ncirl.ie at 24/06/2020, 21:11:45 on unknown cluster
```

Figure 159: Logistic Regression Results (5 of 5)

37. Next Cross Validation was implemented to try and improve on the test results. It should be noted that MLflow tracks experiments but even though it was imported and had worked on previous runs didn't work on below attempt. The LR achieved slightly better AUC on training and test so it was decided to continue with the LR model. If the number of folds had been increased better results could have possibly been achieved but due to time restrictions this was not possible.

```
Cmd 131
1 paramGrid = (ParamGridBuilder()\
2 .addGrid(logistic.elasticNetParam,[0.0,0.5,1.0])\
3 .addGrid(logistic.maxIter,[1, 5, 10])\
4 .addGrid(logistic.regParam,[0.01,0.5,2.0])\
5 .build())

Command took 0.04 seconds -- by x16140818@student.ncirl.ie at 25/06/2020, 00:28:11 on unknown cluster

Cmd 132
1 cv = CrossValidator(estimator=logistic, estimatorParamMaps=paramGrid, evaluator= cevaluator_lr, numFolds=5)

Command took 0.04 seconds -- by x16140818@student.ncirl.ie at 25/06/2020, 00:28:12 on unknown cluster
```

Figure 160: Cross Validation (1 of 2)

```
Cmd 133
1 cvmodel = cv.fit(train2)

/databricks/spark/python/pyspark/ml/util.py:791: UserWarning: Can not find mlflow. To enable mlflow logging, install MLflow library from PyPi.
  warnings.warn(_MLflowInstrumentation._NO_MLFLOW_WARNING)

Command took 1.49 hours -- by x16140818@student.ncirl.ie at 25/06/2020, 00:32:06 on unknown cluster

Cmd 134
1 train_cv =cvmodel.transform(train2)

▶ train_cv: pyspark.sql.dataframe.DataFrame = [review_id: string, wordCount: integer ... 16 more fields]

Command took 0.17 seconds -- by x16140818@student.ncirl.ie at 25/06/2020, 00:32:13 on unknown cluster
```

Figure 161: Cross Validation (2 of 2)

```
Cmd 135
1 test_cv =cvmodel.transform(test2)

▶ test_cv: pyspark.sql.dataframe.DataFrame = [review_id: string, wordCount: integer ... 16 more fields]

Command took 0.09 seconds -- by x16140818@student.ncirl.ie at 25/06/2020, 00:32:21 on unknown cluster

Cmd 136
1 print("The area under ROC for train set is {}".format(cevaluator_lr.evaluate(train_cv)))
2 print("The area under ROC for test set is {}".format(cevaluator_lr.evaluate(test_cv)))

The area under ROC for train set is 0.8843713651192983
The area under ROC for test set is 0.8821823617043585

Command took 12.87 minutes -- by x16140818@student.ncirl.ie at 25/06/2020, 00:32:39 on unknown cluster
```

Figure 162: Cross Validation Results

38. Next the LR model was applied to the Review DataFrame containing over 8 million reviews.

```
Cmd 137
1 useful15 = useful10.withColumn("label", useful_convert("useful"))
▶ useful15: pyspark.sql.dataframe.DataFrame = [review_id: string, useful: long ... 14 more fields]
Command took 0.06 seconds -- by x16140818@student.ncirl.ie at 24/06/2020, 21:53:08 on unknown cluster

Cmd 138
1 useful15.select("review_id").count()

Out[294]: 8021122
Command took 6.30 minutes -- by x16140818@student.ncirl.ie at 24/06/2020, 21:53:09 on unknown cluster
```

Figure 163: Logistic Regression Full Review DataFrame (1 of 4)

```
Cmd 139
1 numeric_features1 = useful15.select("wordCount", "outDegree", "pagerank", "review_count")
2
▶ numeric_features1: pyspark.sql.dataframe.DataFrame = [wordCount: integer, outDegree: integer ... 2 more fields]
Command took 0.10 seconds -- by x16140818@student.ncirl.ie at 24/06/2020, 21:53:11 on unknown cluster

Cmd 140
1 nf1 = numeric_features1.columns

Command took 0.04 seconds -- by x16140818@student.ncirl.ie at 24/06/2020, 21:53:14 on unknown cluster

Cmd 141
1 val1 = VectorAssembler(inputCols=nf1, outputCol="features", handleInvalid="skip")

Command took 0.06 seconds -- by x16140818@student.ncirl.ie at 24/06/2020, 21:53:17 on unknown cluster
```

Figure 164: Logistic Regression Full Review DataFrame (2 of 4)

```
Cmd 142
1 lru = val1.transform(useful15)
▶ lru: pyspark.sql.dataframe.DataFrame = [review_id: string, useful: long ... 15 more fields]
Command took 0.14 seconds -- by x16140818@student.ncirl.ie at 24/06/2020, 21:53:22 on unknown cluster

Cmd 143
1 lru1 = scalerII.fit(lru).transform(lru)
▶ lru1: pyspark.sql.dataframe.DataFrame = [review_id: string, useful: long ... 16 more fields]
Command took 8.08 minutes -- by x16140818@student.ncirl.ie at 24/06/2020, 21:54:03 on unknown cluster
```

Figure 165: Logistic Regression Full Review DataFrame (3 of 4)

```
Cmd 144
1 lru2 = lru1.withColumn('label', lru1["label"].cast(IntegerType()))

▶ lru2: pyspark.sql.dataframe.DataFrame = [review_id: string, useful: long ... 16 more fields]
Command took 0.05 seconds -- by x16140818@student.ncirl.ie at 24/06/2020, 21:54:19 on unknown cluster

Cmd 145
1 dataset2b = cmodel_lr.transform(lru2)

▶ dataset2b: pyspark.sql.dataframe.DataFrame = [review_id: string, useful: long ... 19 more fields]
Command took 0.10 seconds -- by x16140818@student.ncirl.ie at 24/06/2020, 21:54:21 on unknown cluster
```

Figure 166: Logistic Regression Full Review DataFrame (4 of 4)

39. The DataFrame was filtered for predictions equal to 1 i.e. Useful. The Review ID was selected and merged with Review and Business fields to match format of Dataset 1

```
Cmd 146
1 dataset_lr = dataset2b.filter(dataset2b.prediction == 1)

▶ dataset_lr: pyspark.sql.dataframe.DataFrame = [review_id: string, useful: long ... 19 more fields]
Command took 0.12 seconds -- by x16140818@student.ncirl.ie at 24/06/2020, 22:36:54 on unknown cluster

Cmd 147
1 dataset_lr1 = dataset_lr.select("review_id")

▶ dataset_lr1: pyspark.sql.dataframe.DataFrame = [review_id: string]
Command took 0.05 seconds -- by x16140818@student.ncirl.ie at 24/06/2020, 23:35:40 on unknown cluster
```

Figure 167: Filter for Useful Reviews

```
Cmd 148
1 dataset_2u = dataset_lr1.join(review, on="review_id", how="inner")

▶ dataset_2u: pyspark.sql.dataframe.DataFrame = [review_id: string, business_id: string ... 7 more fields]
Command took 0.05 seconds -- by x16140818@student.ncirl.ie at 24/06/2020, 23:35:55 on unknown cluster

Cmd 149
1 dataset2II = dataset_2u.join(bd1, on="business_id", how="inner")

▶ dataset2II: pyspark.sql.dataframe.DataFrame = [business_id: string, review_id: string ... 15 more fields]
Command took 0.06 seconds -- by x16140818@student.ncirl.ie at 24/06/2020, 23:36:04 on unknown cluster
```

Figure 168: Merge DataFrames

40. Distribution of Star Rating



Figure 169: Star Distribution

41. Apply Stratified Sampling to create Dataset 2



Figure 170: Stratified sampling Attempt 1 (1 of 2)



Figure 171: Stratified sampling Attempt 1 (2 of 2)

42. While checking star distribution in Colab it was noticed that the Display function did not calculate the star count correctly and therefore a pandas dataframe was used to see correct output

```

Cmd 153
1 pd6 = dataset2a.toPandas()

(1) Spark Jobs
/databricks/spark/python/pyspark/sql/types.py:1636: DeprecationWarning: Using or importing the ABCs from 'collections' instead of from 'collections.abc' is deprecated, and
d in 3.8 it will stop working
return pa.schema(fields)
Command took 11.63 minutes -- by x16140818@student.ncirl.ie at 07/08/2020, 19:35:26 on Dataset

Cmd 154
1 pd6.groupby("stars").count()

Out[138]:
  business_id  review_id  cool  date  funny  text  useful  user_id  address  city  latitude  longitude  name  state  categories  review_count
stars
1.0  140591  140591  140591  140591  140591  140591  140591  140591  140591  140591  140591  140591  140591  140480  140591
2.0  139406  139406  139406  139406  139406  139406  139406  139406  139406  139406  139406  139406  139406  139307  139406
3.0  140333  140333  140333  140333  140333  140333  140333  140333  140333  140333  140333  140333  140333  140319  140333
4.0  140640  140640  140640  140640  140640  140640  140640  140640  140640  140640  140640  140640  140640  140630  140640
5.0  139943  139943  139943  139943  139943  139943  139943  139943  139943  139943  139943  139943  139943  139935  139943
Command took 0.60 seconds -- by x16140818@student.ncirl.ie at 07/08/2020, 19:35:28 on Dataset

```

Figure 172: Pandas Groupby

```

Cmd 155
1 dataset2b = dataset2II.sampleBy("stars", fractions=[1: 0.339128080548853, 2: 0.505255836085344, 3: 0.299050803651049, 4: 0.166596829275834, 5: 0.137853843096231],
seed=436)

dataset2b: pyspark.sql.dataframe.DataFrame = [business_id: string, review_id: string ... 15 more fields]
Command took 0.04 seconds -- by x16140818@student.ncirl.ie at 07/08/2020, 20:04:51 on Dataset

Cmd 156
1 display(dataset2b.groupBy("stars").count().sort(desc("count")))

(1) Spark Jobs
  stars  count
1  2  140652
2  5  140485
3  3  139959
4  4  139414
5  1  139113

Showing all 5 rows.
Command took 9.47 minutes -- by x16140818@student.ncirl.ie at 07/08/2020, 17:57:09 on Dataset

```

Figure 173: Stratified sampling Attempt 2 (1 of 2)

```

Cmd 157
1 pd4 = dataset2b.toPandas()

(1) Spark Jobs
Command took 11.60 minutes -- by x16140818@student.ncirl.ie at 07/08/2020, 19:35:41 on Dataset

Cmd 158
1 pd4.groupby("stars").count()

Out[141]:
  business_id  review_id  cool  date  funny  text  useful  user_id  address  city  latitude  longitude  name  state  categories  review_count
stars
1.0  139975  139975  139975  139975  139975  139975  139975  139975  139975  139975  139975  139975  139975  139864  139975
2.0  139979  139979  139979  139979  139979  139979  139979  139979  139979  139979  139979  139979  139979  139900  139979
3.0  139985  139985  139985  139985  139985  139985  139985  139985  139985  139985  139985  139985  139985  139971  139985
4.0  140039  140039  140039  140039  140039  140039  140039  140039  140039  140039  140039  140039  140039  140029  140039
5.0  139989  139989  139989  139989  139989  139989  139989  139989  139989  139989  139989  139989  139989  139961  139989
Command took 0.60 seconds -- by x16140818@student.ncirl.ie at 07/08/2020, 19:35:42 on Dataset

```

Figure 174: Stratified sampling Attempt 2 (2 of 2)

```

Cmd 159
1 dataset2c = dataset2II.sampleBy("stars", fractions=[1: 0.33918864995063, 2: 0.505331635830719, 3: 0.299082848241932, 4: 0.166550433083761, 5: 0.137864675320721],
seed=436)

▶ dataset2c: pyspark.sql.dataframe.DataFrame = [business_id: string, review_id: string ... 15 more fields]
Command took 0.83 seconds -- by x16140818@student.ncirl.ie at 07/08/2020, 20:04:59 on Dataset

Cmd 160
1 pd7 = dataset2c.toPandas()

▶ (1) Spark Jobs
Command took 12.88 minutes -- by x16140818@student.ncirl.ie at 07/08/2020, 20:05:04 on Dataset

Cmd 161
1 pd7.groupby("stars").count()

Out[146]:

```

stars	business_id	review_id	cool	date	funny	text	useful	user_id	address	city	latitude	longitude	name	state	categories	review_count
1.0	140001	140001	140001	140001	140001	140001	140001	140001	140001	140001	140001	140001	140001	140001	139890	140001
2.0	140011	140011	140011	140011	140011	140011	140011	140011	140011	140011	140011	140011	140011	140011	139992	140011
3.0	140007	140007	140007	140007	140007	140007	140007	140007	140007	140007	140007	140007	140007	140007	139993	140007
4.0	139987	139987	139987	139987	139987	139987	139987	139987	139987	139987	139987	139987	139987	139987	139977	139987
5.0	139999	139999	139999	139999	139999	139999	139999	139999	139999	139999	139999	139999	139999	139999	139991	139999

Figure 175: Stratified sampling Attempt 3

43. Clean the text to transfer to candidate's laptop

```

Cmd 162
1 d2 = dataset2c.withColumn("text", regexp_replace(col("text"), "[()&*~]", ""))

▶ d2: pyspark.sql.dataframe.DataFrame = [business_id: string, review_id: string ... 15 more fields]
Command took 0.10 seconds -- by x16140818@student.ncirl.ie at 07/08/2020, 20:18:21 on Dataset

Cmd 163
1 d3 = d2.withColumn("text", regexp_replace(col("text"), "[\n\r]", " "))

▶ d3: pyspark.sql.dataframe.DataFrame = [business_id: string, review_id: string ... 15 more fields]
Command took 0.04 seconds -- by x16140818@student.ncirl.ie at 07/08/2020, 20:18:31 on Dataset

```

Figure 176: Clean Text

```

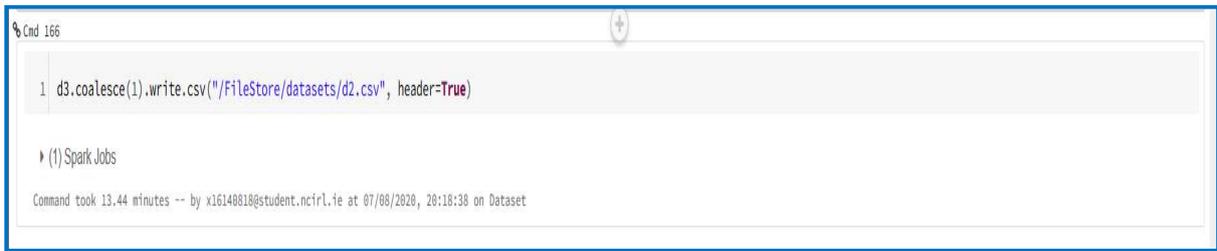
Cmd 164
1 d3.select("text").show(5, False)

▶ (1) Spark Jobs
-----+
|text
|
-----+
|Went there for dinner. Honestly I would not go back again. The food was good nothing special but the prices kinda high and the portions small. You have to order at least 3-4 items from
seafood meat veggies and apps to come close to feeling full. And don't forget the beer. And that's only for 2 people.
|
|Their work is good quality for moderate prices. Niki the owner is the best in terms of workmanship. The store however is generally very messy and the staff are rude and not accommodatin
g. They often quote you one price and charge more because they "quoted wrong" in the first place. Communication is also very challenging since everyone doesn't speak great English or Can
tonese Chinese so it would be hard to get your exact ideas accredited if you don't speak Mandarin Chinese. Appointment times usually run late as well so expect to start getting your nail
s done at 7:15 if you booked 6:30. Expect no apology for lateness...they aint about customer service there :S
|This place is AMAZING! After living in France for years I can assure you that this is the real deal. I cannot say enough wonderful things about this place and the owner. Absolutely the
best crepes in Vegas and in my opinion the best I have had in the US. Do yourself a favor and visit this place soon.
|
|If you are vegetarian or vegan Fern is an excellent choice for a good place to eat. My wife is vegan so I've eaten at Fern several times with her. Even though I'm not vegetarian I've
Command took 9.93 minutes -- by x16140818@student.ncirl.ie at 07/08/2020, 20:35:03 on Dataset

```

Figure 177: Cleaned Text

44. Write data to DBFS



```
1 d3.coalesce(1).write.csv("/FileStore/datasets/d2.csv", header=True)
```

Command took 13.44 minutes -- by x16140818@student.ncirl.ie at 07/08/2020, 20:18:38 on Dataset

Figure 178: Write to Databricks File Store

45. To Transfer files from DBFS to local computer the following steps were taken. Click Data

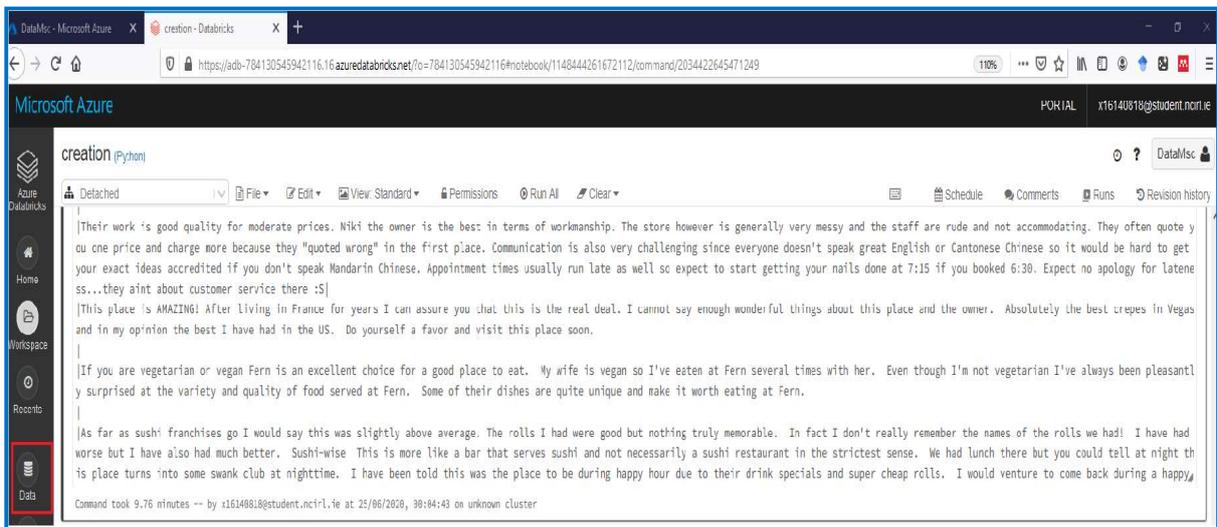


Figure 179: Connect to Data in DBFS

46. Add Data

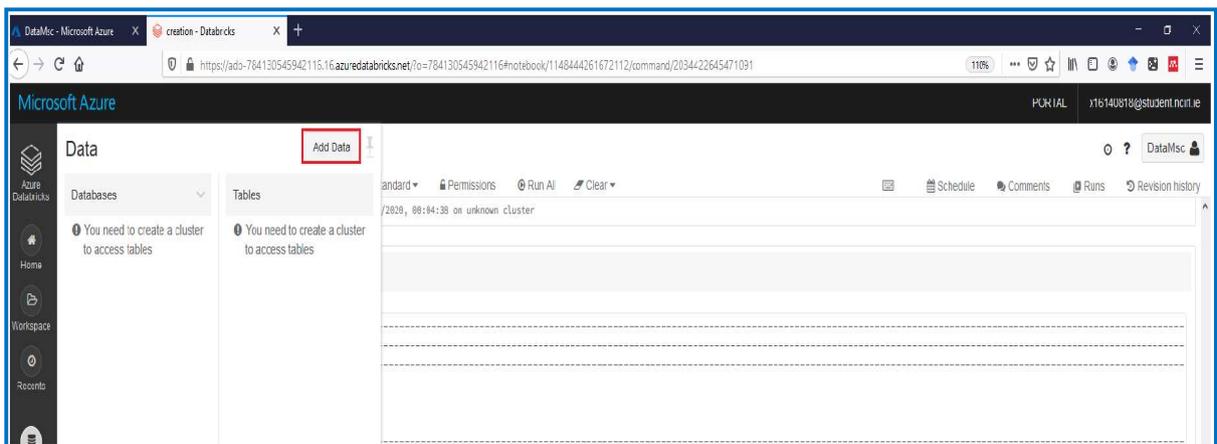


Figure 180: Click Add Data

47. Click DBFS, FileStore, datasets, dataset_one_csv and part-000. Copy the highlighted path for part-000. This needs to be done for Dataset 2 also.

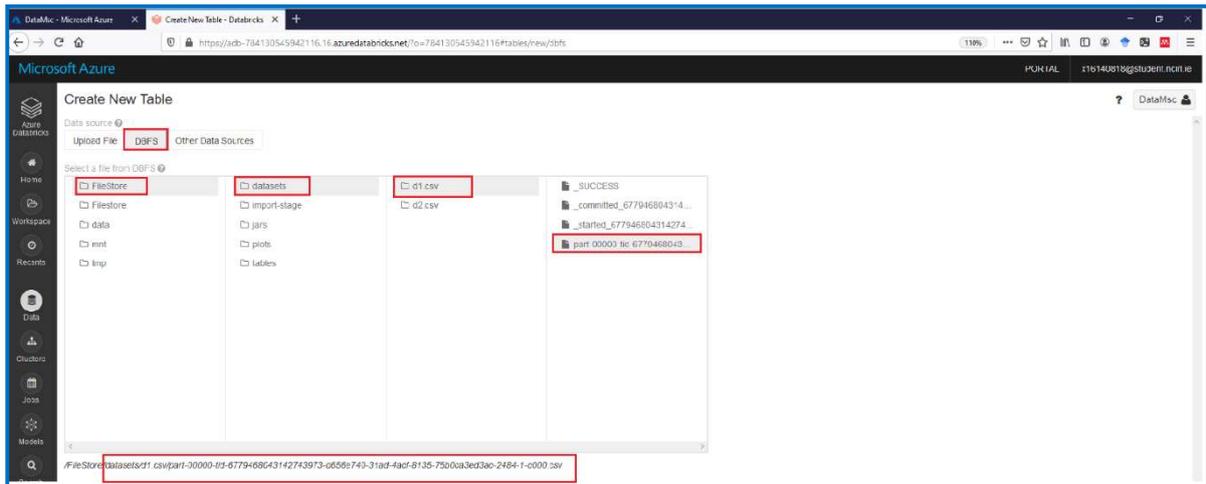


Figure 181: Dataset 1 File Path

48. The following URL was then used to start the download for Dataset 1 ¹⁰

49. The following URL was then used to start the download for Dataset 2 ¹¹

The 2 Datasets have been created and next they were transferred to Colab for further processing

5 Data Pre-processing and Exploration Google Colabartory

Google Colab is free cloud-based environment for machine learning using Jupyter Notebook. Colab provides free access to Tensor Processing Unit (TPU) and GPU. A limit of 12GB of Random Access Memory (RAM) is provided which unfortunately was not enough for running the complete project. As a result, data pre-processing and data exploration was complete with Colab.

2. The first step was to upload the two datasets from local drive to the Google Drive website¹². Click New, File Upload and select the two recently download files from DBFS

¹⁰ <https://westeurope.azuredatabricks.net/files/datasets/d1.csv/part-00000-tid-6779468043142743973-d656e740-31ad-4acf-8135-75b0ca3ed3ac-2484-1-c000.csv>

¹¹ <https://westeurope.azuredatabricks.net/files/datasets/d2.csv/part-00000-tid-662439142231073286-bad7a3a4-8a45-4896-8737-cba2757ef27f-145331-1-c000.csv>

¹² <https://drive.google.com/drive/u/0/my-drive>

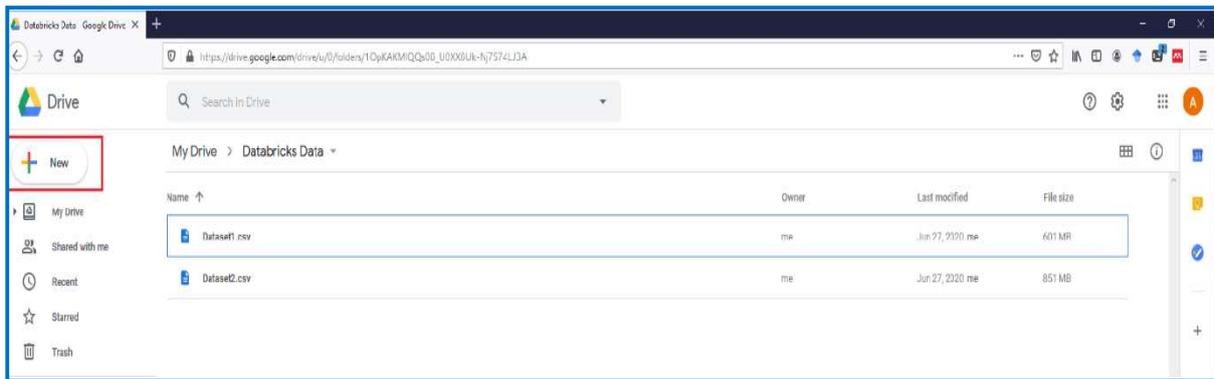


Figure 182: Upload Data from Local

3. Go the Colab website¹³ and click File, New Notebook

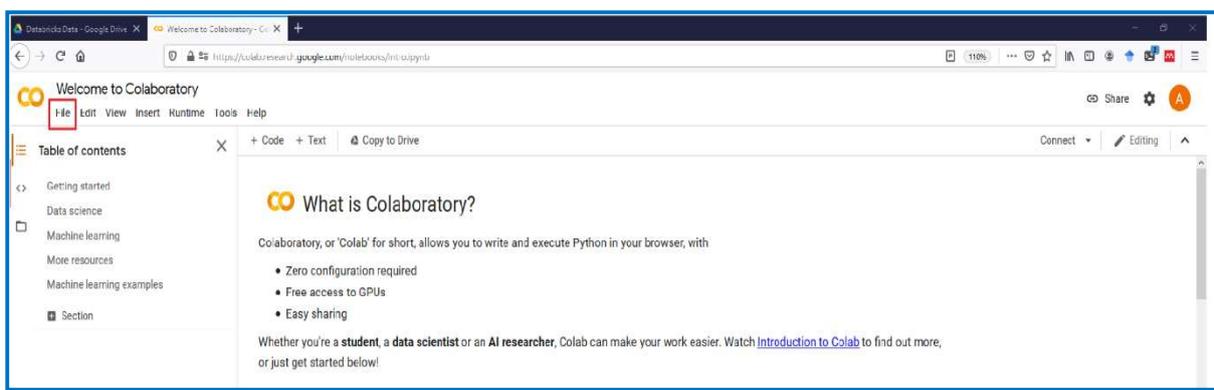


Figure 183: Launch Notebook

4. Select Tensor Processing Unit (TPU) by clicking Runtime, Change Runtime Type, select Hardware accelerator as TPU and then click save.

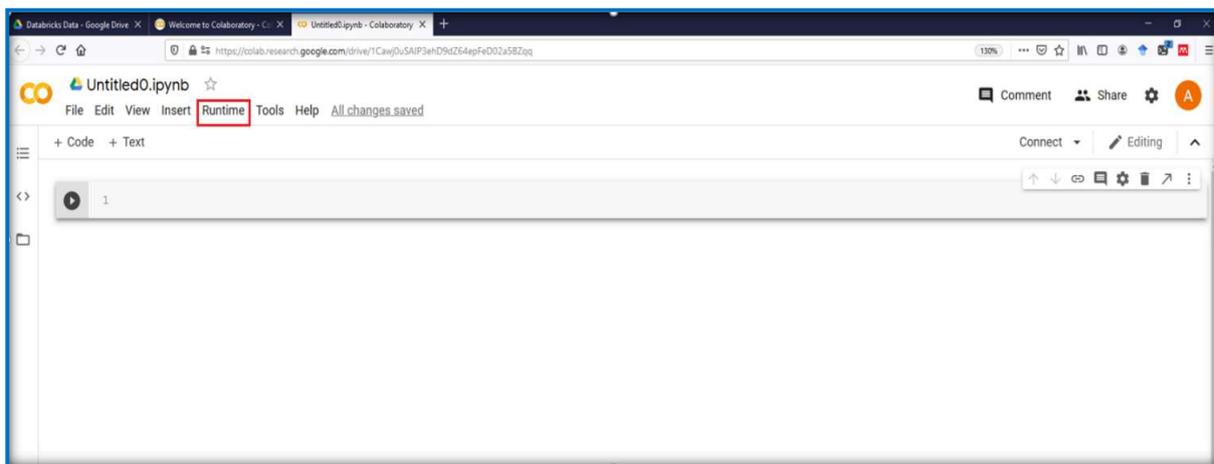


Figure 184: TPU Set-up (1 of 2)

¹³ <https://colab.research.google.com/notebooks/intro.ipynb>

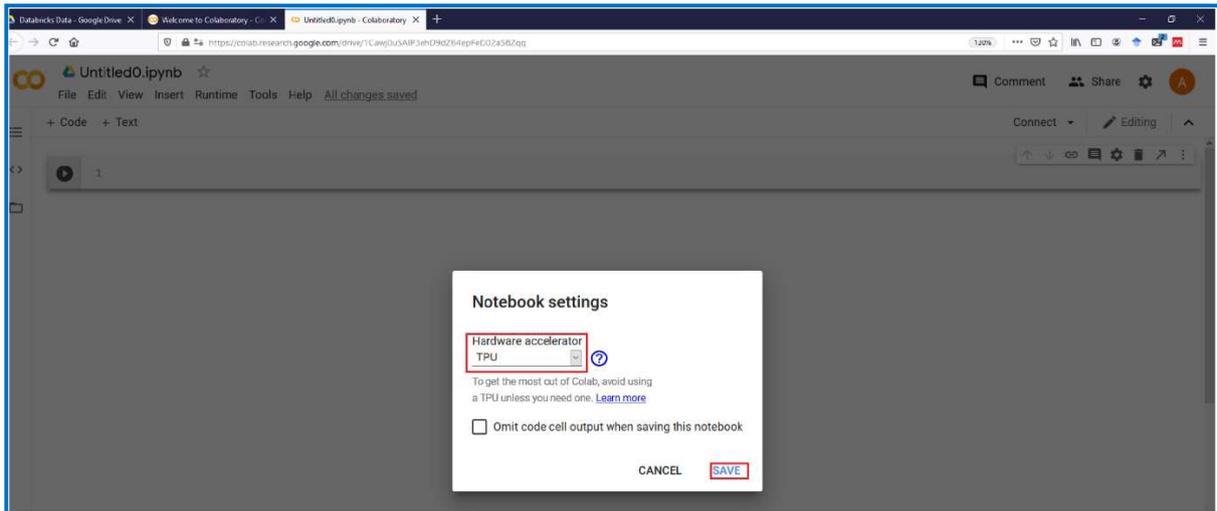


Figure 185: TPU Set-up (2 of 2)

- Once the TPU has been assigned the following Python code was run to process Dataset 1 further. Install necessary libraries.



Figure 186: Install Transformers Library



Figure 187: Import Libraries

6. Mount Google Drive to access Dataset 1. Go to the URL link to copy authentication code

```
1 from google.colab import drive
2 drive.mount('/content/drive')
```

Go to this URL in a browser: https://accounts.google.com/o/oauth2/auth?client_id=947318989803-6bn6qk8qdyf4n4g3pfee6491hc0brc4i.apps.googleusercontent.com&redirect_uri=urn%3aietf

Enter your authorization code:
.....
Mounted at /content/drive

```
[ ] 1 dataset1 = pd.read_csv('/content/drive/My Drive/Colab Notebooks/Dataset1.csv', error_bad_lines=False)
```

b'Skipping line 132889: expected 17 fields, saw 21\n'
/usr/local/lib/python3.6/dist-packages/IPython/core/interactiveshell.py:2718: DtypeWarning: Columns (7,11,12) have mixed types.Specify dtype option on import or set low_memory=interactivity=interactivity, compiler=compiler, result=result)

Figure 188: Mount Google Drive

7. Select text and stars, check for null values, convert star column to integer and display star distribution

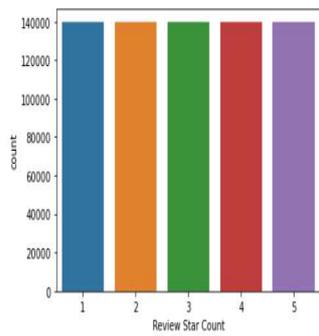
```
[ ] 1 d1 = dataset1[["text","stars"]].copy()
```

```
[ ] 1 d1['text'].isnull().sum()
```

0

```
[ ] 1 d1["stars"] = d1["stars"].apply(np.int)
```

```
[ ] 1 star1 = sns.countplot(d1.stars)
2 plt.xlabel("Review Star Count");
```



Review Star Count	count
1	140000
2	135000
3	130000
4	125000
5	120000

Figure 189: Star Distribution

8. Reduce star rating by 1, rename stars column labels, split train & test and reset index



Figure 190: Format Labels

9. Split the training dataset into training and validation. Display star distribution for training, validation and testing and reset index



Figure 191: Training Star Distribution

```
[ ] 1 star6 = sns.countplot(d1_eval.labels)
     2 plt.xlabel('Review Star Count')
     3 plt.title('Star Distribution');
```

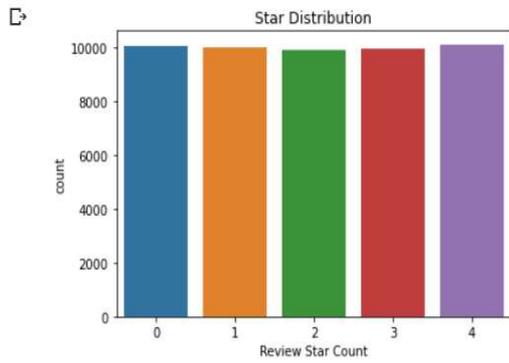


Figure 192: Evaluation Star Distribution

```
[ ] 1 star7 = sns.countplot(d1_test.labels)
     2 plt.xlabel('Review Star Count')
     3 plt.title('Star Distribution');
```

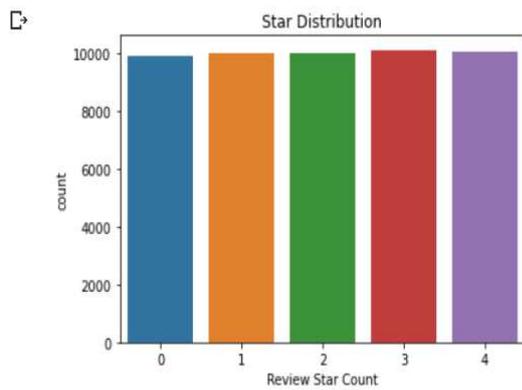


Figure 193: Test Star Distribution

```
[ ] 1 d1_train.reset_index(drop=True)
```

	text	labels
0	I was admittedly a bit skeptical about the Duc...	3
1	Came here for lunch on a weekday. Heard mostly...	2
2	This place is weird. It's in a beautiful old b...	2

Figure 194: Reset Train Index

```
[ ] 1 d1_eval.reset_index(drop=True)
```

	text	labels
0	Le service quoique sympathique est si lent que...	1
1	We went back for a second time really looking ...	2
2	My boyfriend and I visited this buffet on a Tu...	3

Figure 195: Reset Evaluation Index

```
[ ] 1 d1_test.reset_index(drop=True)
```

	text	labels
0	I really liked the beer and patio seating here...	3
1	I was very excited when I saw the 'Coming Soon...	1
2	This gets a one star because there's nothing l...	0

Figure 196: Reset Test Index

10. Write train, validation and test to csv in Google Drive

```
[ ] 1 d1_train.to_csv('/content/drive/My Drive/Dataset 1 files/d1_train.csv',index=False, header=True)
[ ] 1 d1_eval.to_csv('/content/drive/My Drive/Dataset 1 files/d1_eval.csv',index=False, header=True)
[ ] 1 d1_test.to_csv('/content/drive/My Drive/Dataset 1 files/d1_test.csv',index=False, header=True)
```

Figure 197: Write to Google Drive

11. Import BertTokenizer, select text column of Dataset 1 and reset index

```
[ ] 1 from transformers import BertTokenizer
[ ] 1 tokenizer = BertTokenizer.from_pretrained('bert-base-uncased',do_lower_case=True)
```

Downloading: 100%  232k/232k [00:00<00:00, 1.47MB/s]

```
[ ] 1 tokens1 = d1[['text']].copy().sample(n=500000)
[ ] 1 tokens1.reset_index(drop=True)
```

	text
0	This place is so good!!! They have quite the s...
1	Tried this little French place last night for ...
2	It was soft opening the ordering was a disaste...

Figure 198: Bert Tokenizer

12. Token Distribution

```
[ ] 1 input_ids = []

[ ] 1 lengths = []

[ ] 1 for sentences in tokens1.text:
2   encoded_sent = tokenizer.encode(
3     sentences,
4     add_special_tokens=True,
5   )
6   input_ids.append(encoded_sent)
7   lengths.append(len(encoded_sent))
```

Streaming output truncated to the last 5000 lines.

Token indices sequence length is longer than the specified maximum sequence length for this model (688 > 512). Running this sequence through the model will result in indexing

Token indices sequence length is longer than the specified maximum sequence length for this model (544 > 512). Running this sequence through the model will result in indexing

Token indices sequence length is longer than the specified maximum sequence length for this model (516 > 512). Running this sequence through the model will result in indexing

Token indices sequence length is longer than the specified maximum sequence length for this model (737 > 512). Running this sequence through the model will result in indexing

Token indices sequence length is longer than the specified maximum sequence length for this model (516 > 512). Running this sequence through the model will result in indexing

Token indices sequence length is longer than the specified maximum sequence length for this model (523 > 512). Running this sequence through the model will result in indexing

Figure 199: Token Function

13. Print Min, Max, Mean, Median and percentage of sentences over 512 Tokens and display token distribution

```
[ ] 1 print(' Min Length: {:} tokens'.format(min(lengths)))
2 print(' Max Length: {:,} tokens'.format(max(lengths)))
3 print(' Mean Length: {:,} tokens'.format(np.mean(lengths)))
4 print(' Median Length: {:,} tokens'.format(np.median(lengths)))
```

Min Length: 3 tokens
Max Length: 3,090 tokens
Mean Length: 149.940786 tokens
Median Length: 109.0 tokens

```
[ ] 1 number_trunc = sum(i >=512 for i in lengths)
2
3 number_sent= len(lengths)
4
5 percent = float(number_trunc) / float(number_sent)
6
7 print('{:,} of {:,} sentences({:.1%}) in Dataset 1 are longer the 512 tokens.'.format(number_trunc,number_sent,percent))
```

13,065 of 500,000 sentences(2.6%) in Dataset 1 are longer the 512 tokens.

Figure 200: Token Distribution Statistics

```
[ ] 1 sns.set(style='darkgrid')
2
3 sns.set(font_scale=1.5)
4 plt.rcParams["figure.figsize"] = (10,5)
5
6 lengths1 = [min(l, 512) for l in lengths]
7
8 sns.distplot(lengths1)
9
10 plt.xlabel('Token Count')
11
12 plt.title('Token Distribution');
```

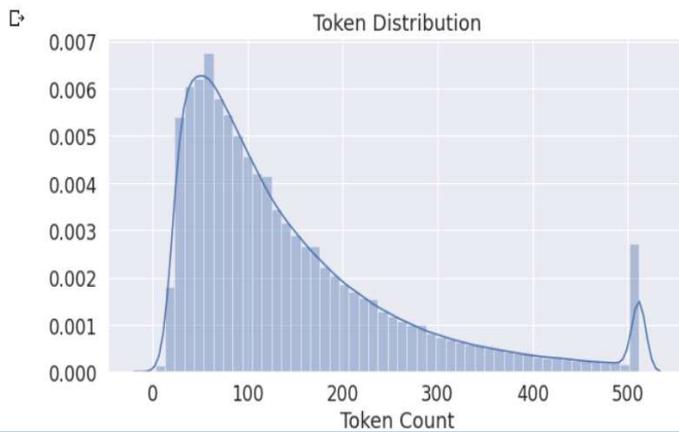


Figure 201: Token Distribution

14. The same process as above was applied to Dataset 2

6 Fine-grained Yelp Models

The 6 Fine-grained Yelp models were run using JupyterLab and Genesis Cloud. The following steps were taken to set-up an Instance and run the models

1. Create Instance via Genesis Cloud website¹⁴. Click Create New Instance

¹⁴ <https://compute.genesiscloud.com/dashboard/instances>

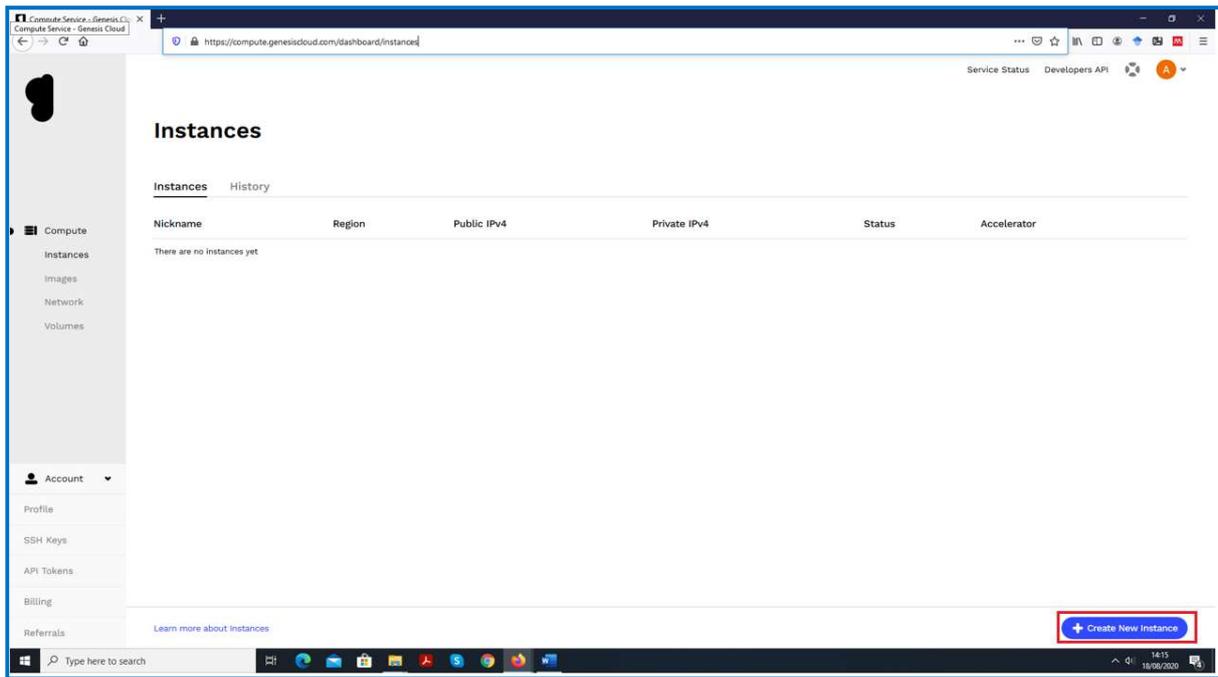


Figure 202: Instance Webpage

2. Required information for Instance. Hostname, 2 GPU's, Preconfigured image of PyTorch 1.5, Password (remember to copy to notepad), Install NVIDIA GPU driver 430.50. When previous has been done click create instance.

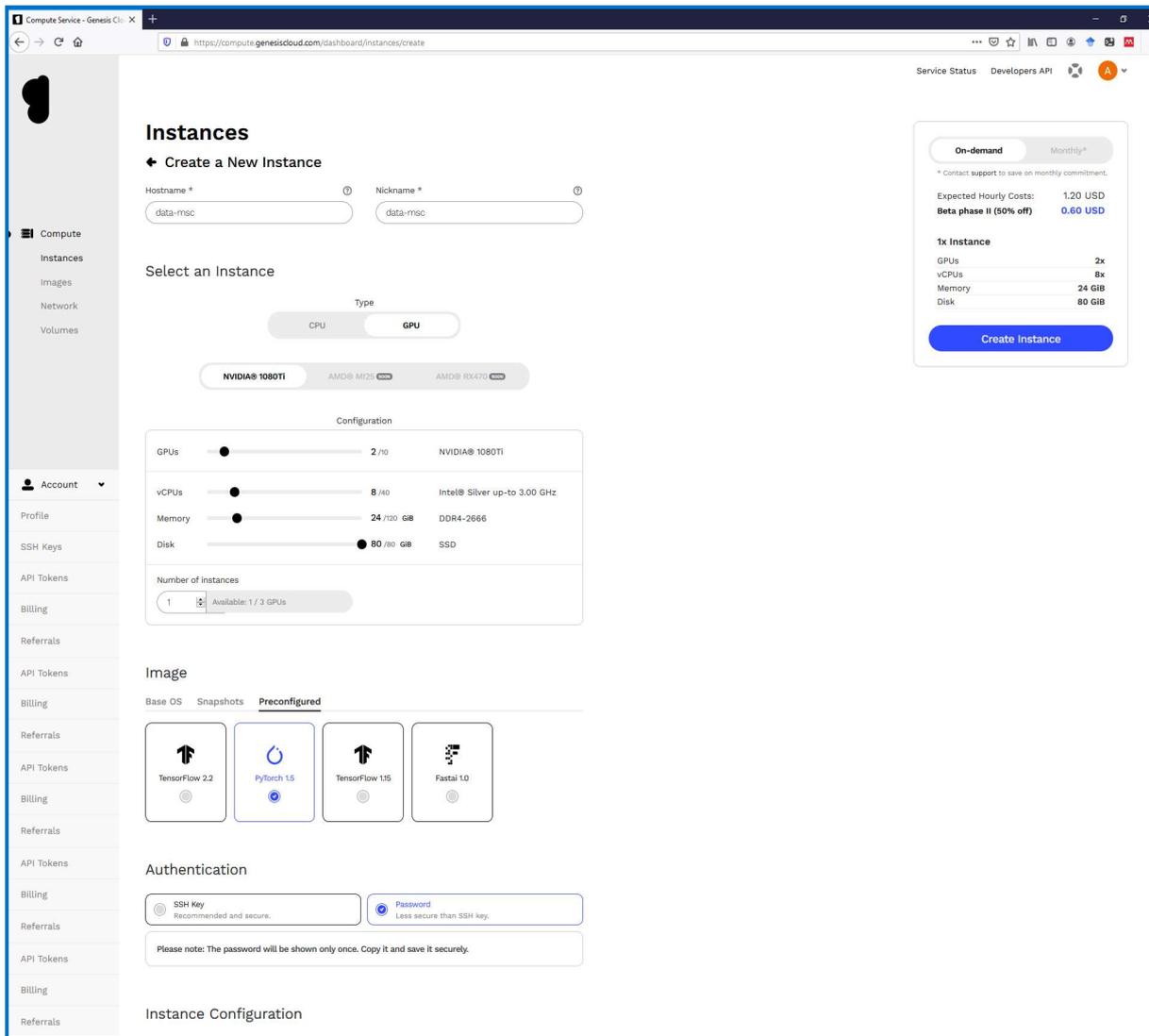


Figure 203: Create New Instance (1 of 2)

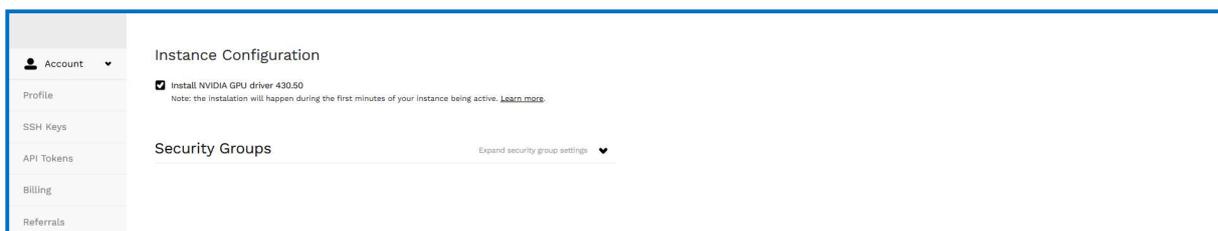


Figure 204: Create New Instance (2 of 2)

3. When the Instance is running copy IPv4 without the SSH and open PuTTY. Paste the copied IPv4 under Session Host Name. Click Tunnels. Under Source port enter 888 and Destination localhost:8888. Click Add and then open and then yes.

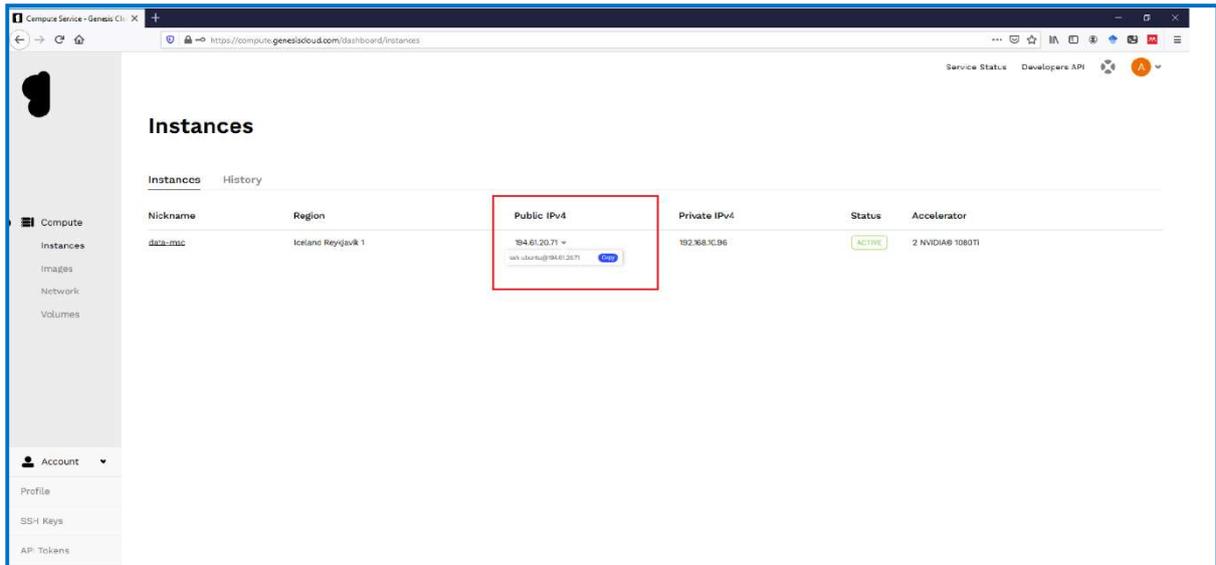


Figure 205: Ipv4

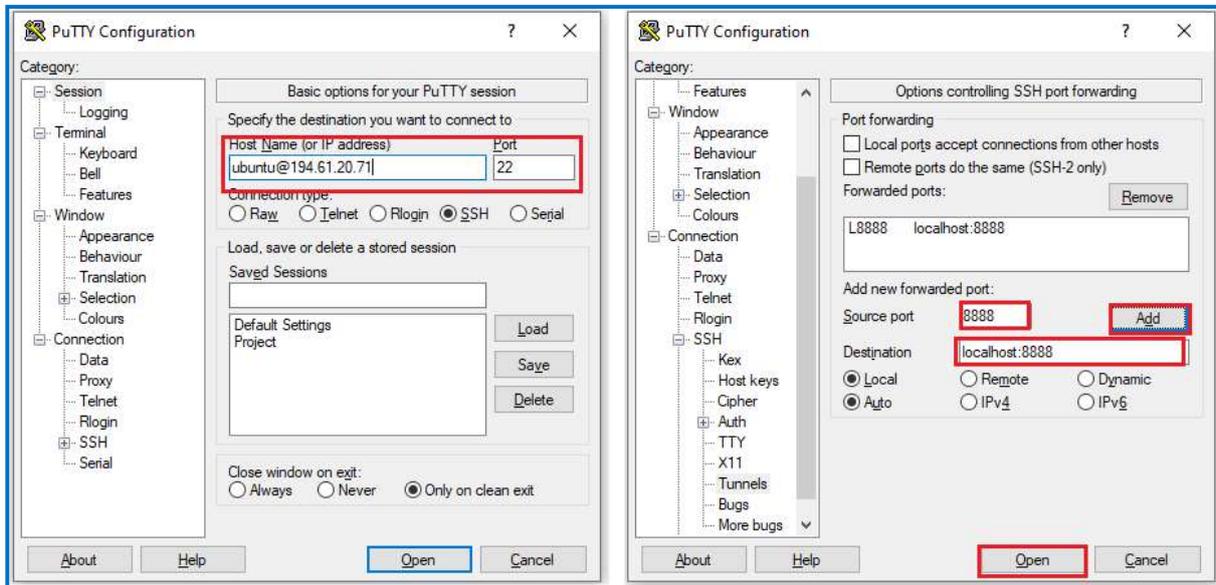


Figure 206: PuTTY Configuration

- When PuTTY opens paste the password previously copied and hit enter. After the password has been accepted to start a browserless JupyterLab enter `jupyter lab --no-browser`. The URL for accessing JupyterLab will now be generated but first open command prompt and enter `ssh -CNL localhost:8888:localhost:8888 ubuntu@194.61.20.71`. The IP address at the end will change for every new Instance. Enter the password and if correct JupyterLab has now been forwarded from a user local machine to the Genesis Cloud GPU's.


```

[ ]: !pip install transformers
!pip install simpletransformers
!pip install wandb

[ ]: !git clone https://github.com/NVIDIA/apex

[ ]: cd apex

[ ]: !pip install -v --no-cache-dir --global-option="--cpp_ext" --global-option="--cuda_ext" ./

[ ]: import logging
import pandas as pd
from sklearn.metrics import accuracy_score, f1_score
import wandb
from simpletransformers.classification import ClassificationArgs, ClassificationModel

wandb.login()

logging.basicConfig(level=logging.INFO)
transformers_logger = logging.getLogger("transformers")
transformers_logger.setLevel(logging.WARNING)

# Preparing train data
train_df = pd.read_csv('/home/ubuntu/d1_train.csv', error_bad_lines=False)
eval_df = pd.read_csv('/home/ubuntu/d1_eval.csv', error_bad_lines=False)
test_df = pd.read_csv('/home/ubuntu/d1_test.csv', error_bad_lines=False)

# Args for model
model_args2 = ClassificationArgs()
model_args2.evaluate_during_training = True
model_args2.evaluate_during_training_steps = 1000
model_args2.manual_seed = 4
model_args2.eval_batch_size = 32
model_args2.max_seq_length = 150
model_args2.train_batch_size = 64
model_args2.num_train_epochs = 3
model_args2.overwrite_output_dir = True
model_args2.reprocess_input_data = True
model_args2.best_model_dir = '/home/ubuntu/outputs2/best_model'
model_args2.output_dir = '/home/ubuntu/outputs2/output'
model_args2.cache_dir = '/home/ubuntu/outputs2/cache_dir'
model_args2.wandb_project = "Msc"
model_args2.wandb_kwargs = {"name": "AlbertD1_2"}
model_args2.n_gpu = 2
model_args2.use_early_stopping = True
model_args2.early_stopping_metric = "mcc"
model_args2.early_stopping_metric_minimize = False
model_args2.early_stopping_patience = 5
model_args2.learning_rate = 4e-5

# Create a TransformerModel
model_a2 = ClassificationModel("albert", "albert-base-v2", use_cuda=True, num_labels=5, args=model_args2)

# Train the model
model_a2.train_model(
    train_df,
    eval_df=eval_df,
    accuracy=accuracy_score)

# Evaluate the model
result, model_outputs, wrong_predictions = model_a2.eval_model(test, accuracy=accuracy_score)

wandb.log({'AlbertD1_2': result})

[ ]: result, model_outputs, wrong_predictions = model_a2.eval_model(test_df, accuracy=accuracy_score)

[ ]: print(result)

[ ]: print(model_outputs)

[ ]: wandb.log({'AlbertD1_2': result})

```

Figure 208: ALBERT-base D1 Code

ALBERT-base D1

What makes this run special?

Privacy: PRIVATE

Tags: +

Author: browner55

State: finished

Start time: July 18th, 2020 at 11:25:17 pm

Duration: 1m 52s

Run path: browner55/Msc/Ziq6A0i9

Hostname: deep-learning-msc

OS: Linux-5.0.0-41-generic-x86_64-with-debian-buster-sid

Python version: 3.7.6

Python executable: /usr/local/share/anaconda3/bin/python

Git repository: git clone https://github.com/NVIDIA/apex

Git state: git checkout -b "ALBERT-base-D1" 3104fd59776a470a5dee3f0d5f3168cf9ab35b53

Command: <code>-python with no main file> -f /home/ubuntu/.local/share/jupyter/runtime/kernel-a8ce8503-310a-4f2b-8df9-c9310dca6799.json</code>

System Hardware: GPU type GeForce GTX 1080 Ti

W&B CLI Version: 0.9.3

Config Raw

Config parameters describe your model's inputs. [Learn more](#)

Name	Value
adam_epsilon	1.000e-8
best_model_dir	/home/ubuntu/outputs2/best_model
cache_dir	/home/ubuntu/outputs2/cache_dir
custom_layer_parameters	[]
custom_parameter_groups	[]
do_lower_case	false
early_stopping_consider_epochs	false
early_stopping_delta	0
early_stopping_metric	mcc
early_stopping_metric_minimize	false
early_stopping_patience	5
encoding	-
eval_batch_size	32
evaluate_during_training	true
evaluate_during_training_silent	true

Page 1 of 4

Summary Raw

Summary metrics describe your results. [Learn more](#)

Name	Value
AlbertD1_2.accuracy	0.6749
AlbertD1_2.eval_loss	0.7661
AlbertD1_2.mcc	0.5939
confusion_matrix	table-file
pr	table-file
roc	table-file

Page 1 of 1

Figure 209: ALBERT-base D1 Results

6.2 Experiment 2 ALBERT-base D2

```
[1]: !pip install transformers
!pip install simpletransformers==0.45.5
!pip install wandb

[2]: import wandb

[3]: wandb.login()

True

[4]: import torch

[5]: !git clone https://github.com/NVIDIA/apex

[6]: cd apex

[7]: !pip install -v --no-cache-dir --global-option="--cpp_ext" --global-option="--cuda_ext" ./

[8]: import logging
import pandas as pd
from sklearn.metrics import accuracy_score
from simpletransformers.classification import ClassificationArgs, ClassificationModel

logging.basicConfig(level=logging.INFO)
transformers_logger = logging.getLogger("transformers")
transformers_logger.setLevel(logging.WARNING)

# Preparing train data
train_df1 = pd.read_csv('/home/ubuntu/d2_train.csv', error_bad_lines=False)
eval_df1 = pd.read_csv('/home/ubuntu/d2_eval.csv', error_bad_lines=False)
test_df1 = pd.read_csv('/home/ubuntu/d2_test.csv', error_bad_lines=False)

# Args for model
args4 = ClassificationArgs()
args4.evaluate_during_training = True
args4.evaluate_during_training_steps = 1000
args4.manual_seed = 4
args4.eval_batch_size = 32
args4.max_seq_length = 235
args4.train_batch_size = 64
args4.num_train_epochs = 3
args4.overwrite_output_dir = True
args4.reprocess_input_data = True
args4.best_model_dir = '/home/ubuntu/outputs/best_model'
args4.output_dir = '/home/ubuntu/outputs/output'
args4.cache_dir = '/home/ubuntu/outputs/cache_dir'
args4.wandb_project = "Msc"
args4.wandb_kwargs = {"name": "AlbertD2"}
args4.n_gpu = 2
args4.use_early_stopping = True
args4.early_stopping_metric = "mcc"
args4.early_stopping_metric_minimize = False
args4.early_stopping_patience = 5
args4.learning_rate = 4e-5

# Create a TransformerModel
model_a4 = ClassificationModel("albert", "albert-base-v2", use_cuda=True, num_labels=5, args=args4)

# Train the model
model_a4.train_model(
    train_df1,
    eval_df=eval_df1,
    accuracy=accuracy_score)

# Evaluate the model
result4, model_outputs4, wrong_predictions4 = model_a4.eval_model(test_df1, accuracy=accuracy_score)

wandb.log({'AlbertD2': result4})
```

Figure 210: ALBERT-base D2 Code

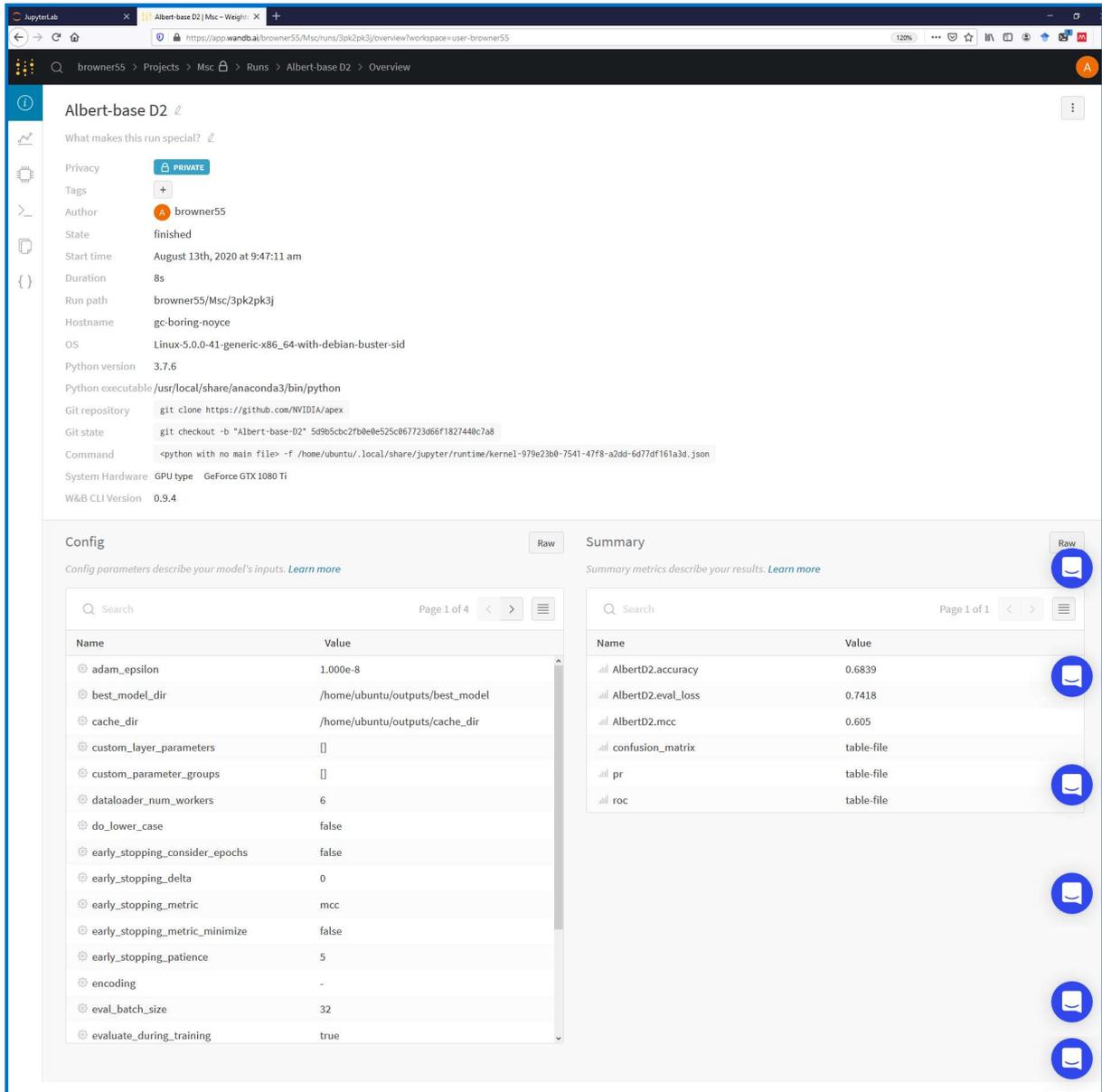


Figure 211: ALBERT-base D2 Results

6.3 Experiment 3 ELECTRA-base D1

```
[ ]: !pip install transformers
!pip install simpletransformers
!pip install wandb

[ ]: import wandb

[ ]: wandb.login()
True

[ ]: import torch

[ ]: !git clone https://github.com/NVIDIA/apex

[ ]: cd apex

[ ]: !pip install -v --no-cache-dir --global-option="--cpp_ext" --global-option="--cuda_ext" ./

[ ]: import logging
import pandas as pd
from sklearn.metrics import accuracy_score
from simpletransformers.classification import ClassificationArgs, ClassificationModel

logging.basicConfig(level=logging.INFO)
transformers_logger = logging.getLogger("transformers")
transformers_logger.setLevel(logging.WARNING)

# Preparing train data
train_df = pd.read_csv('/home/ubuntu/d1_train.csv', error_bad_lines=False)
eval_df = pd.read_csv('/home/ubuntu/d1_eval.csv', error_bad_lines=False)
test_df = pd.read_csv('/home/ubuntu/d1_test.csv', error_bad_lines=False)

# Args for model
model_args11 = ClassificationArgs()
model_args11.evaluate_during_training = True
model_args11.evaluate_during_training_steps = 1000
model_args11.manual_seed = 4
model_args11.eval_batch_size = 32
model_args11.max_seq_length = 150
model_args11.train_batch_size = 64
model_args11.num_train_epochs = 3
model_args11.override_output_dir = True
model_args11.reprocess_input_data = True
model_args11.best_model_dir = '/home/ubuntu/outputs/best_model'
model_args11.output_dir = '/home/ubuntu/outputs/output'
model_args11.cache_dir = '/home/ubuntu/outputs/cache_dir'
model_args11.wandb_project = "Msc"
model_args11.wandb_kwargs = {"name": "ELECTRA D1"}
model_args11.n_gpu = 2
model_args11.use_early_stopping = True
model_args11.early_stopping_metric = "mcc"
model_args11.early_stopping_metric_minimize = False
model_args11.early_stopping_patience = 5
model_args11.learning_rate = 4e-5

# Create a TransformerModel
model11 = ClassificationModel("electra", "google/electra-base-discriminator", use_cuda=True, num_labels=5,

# Train the model
model11.train_model(
    train_df,
    eval_df=eval_df,
    accuracy=accuracy_score)

# Evaluate the model
result11, model_outputs11, wrong_predictions11 = model11.eval_model(test_df, accuracy=accuracy_score)

wandb.log({'ELECTRA D1': result11})
```

Figure 212: ELECTRA-base D1 Code

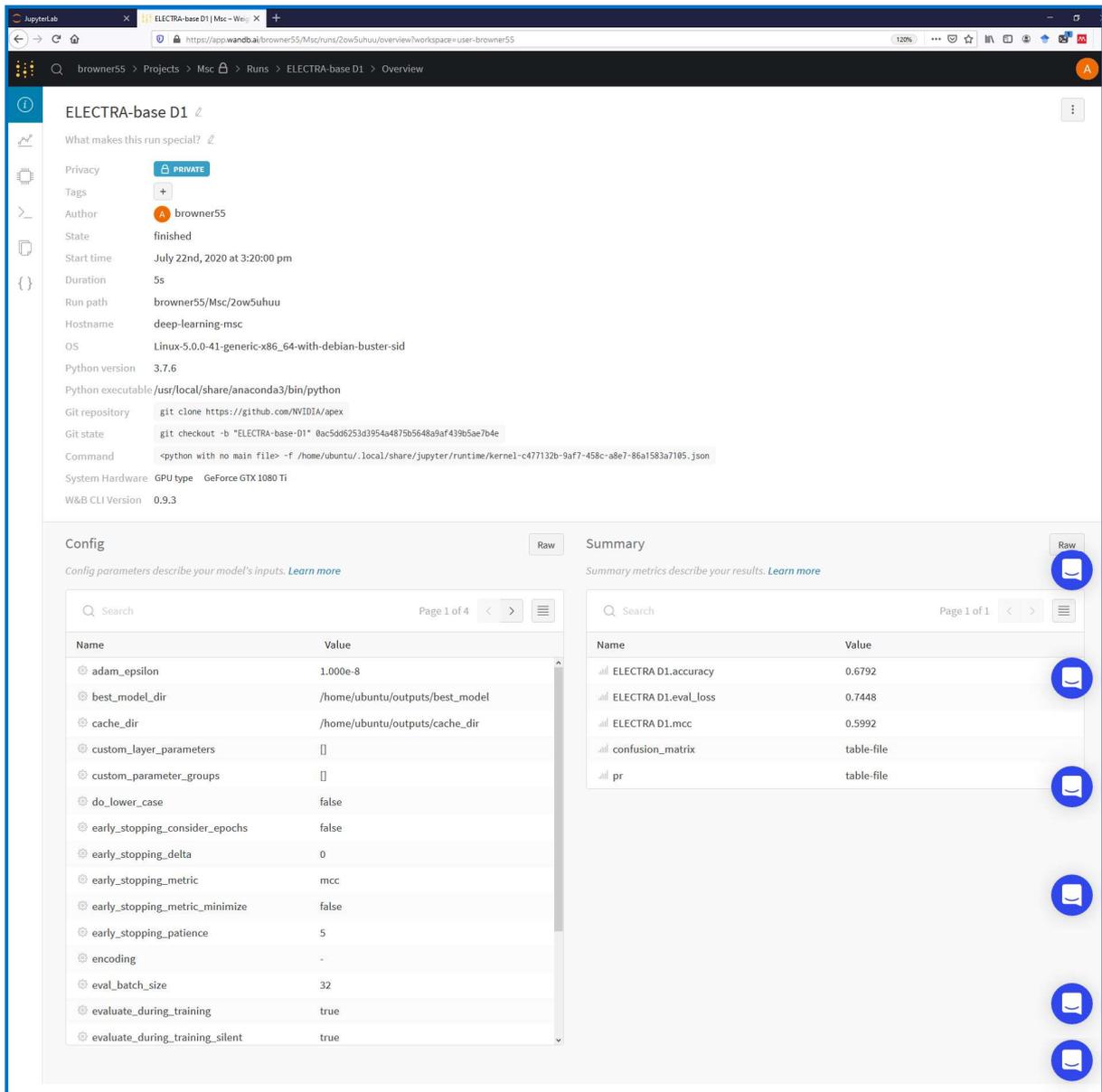


Figure 213: ELECTRA-base D1 Results

6.4 Experiment 4 ELECTRA-base D2

```
[1]: !pip install transformers
!pip install simpletransformers==0.45.5
!pip install wandb

[2]: import wandb

[3]: wandb.login()
True

[4]: import torch

[5]: !git clone https://github.com/NVIDIA/apex

[6]: cd apex

[7]: !pip install -v --no-cache-dir --global-option="--cpp_ext" --global-option="--cuda_ext" ./

[8]: import logging
import pandas as pd
from sklearn.metrics import accuracy_score
from simpletransformers.classification import ClassificationArgs, ClassificationModel

logging.basicConfig(level=logging.INFO)
transformers_logger = logging.getLogger("transformers")
transformers_logger.setLevel(logging.WARNING)

# Preparing train data
train_df1 = pd.read_csv('/home/ubuntu/d2_train.csv', error_bad_lines=False)
eval_df1 = pd.read_csv('/home/ubuntu/d2_eval.csv', error_bad_lines=False)
test_df1 = pd.read_csv('/home/ubuntu/d2_test.csv', error_bad_lines=False)

# Args for model
model_args12 = ClassificationArgs()
model_args12.evaluate_during_training = True
model_args12.evaluate_during_training_steps = 1000
model_args12.manual_seed = 4
model_args12.eval_batch_size = 32
model_args12.max_seq_length = 235
model_args12.train_batch_size = 64
model_args12.num_train_epochs = 3
model_args12.override_output_dir = True
model_args12.reprocess_input_data = True
model_args12.best_model_dir = '/home/ubuntu/outputs/best_model'
model_args12.output_dir = '/home/ubuntu/outputs/output'
model_args12.cache_dir = '/home/ubuntu/outputs/cache_dir'
model_args12.wandb_project = "Msc"
model_args12.wandb_kwargs = {"name": "ELECTRA D2"}
model_args12.n_gpu = 2
model_args12.use_early_stopping = True
model_args12.early_stopping_metric = "mcc"
model_args12.early_stopping_metric_minimize = False
model_args12.early_stopping_patience = 5
model_args12.learning_rate = 4e-5

# Create a TransformerModel
model12 = ClassificationModel("electra", "google/electra-base-discriminator", use_cuda=True, num_labels=5,

# Train the model
model12.train_model(
train_df1,
eval_df=eval_df1,
accuracy=accuracy_score)

# Evaluate the model
result12, model_outputs12, wrong_predictions12 = model12.eval_model(test_df1, accuracy=accuracy_score)

wandb.log({'ELECTRA D2': result12})
```

Figure 214: ELECTRA-base D2 Code

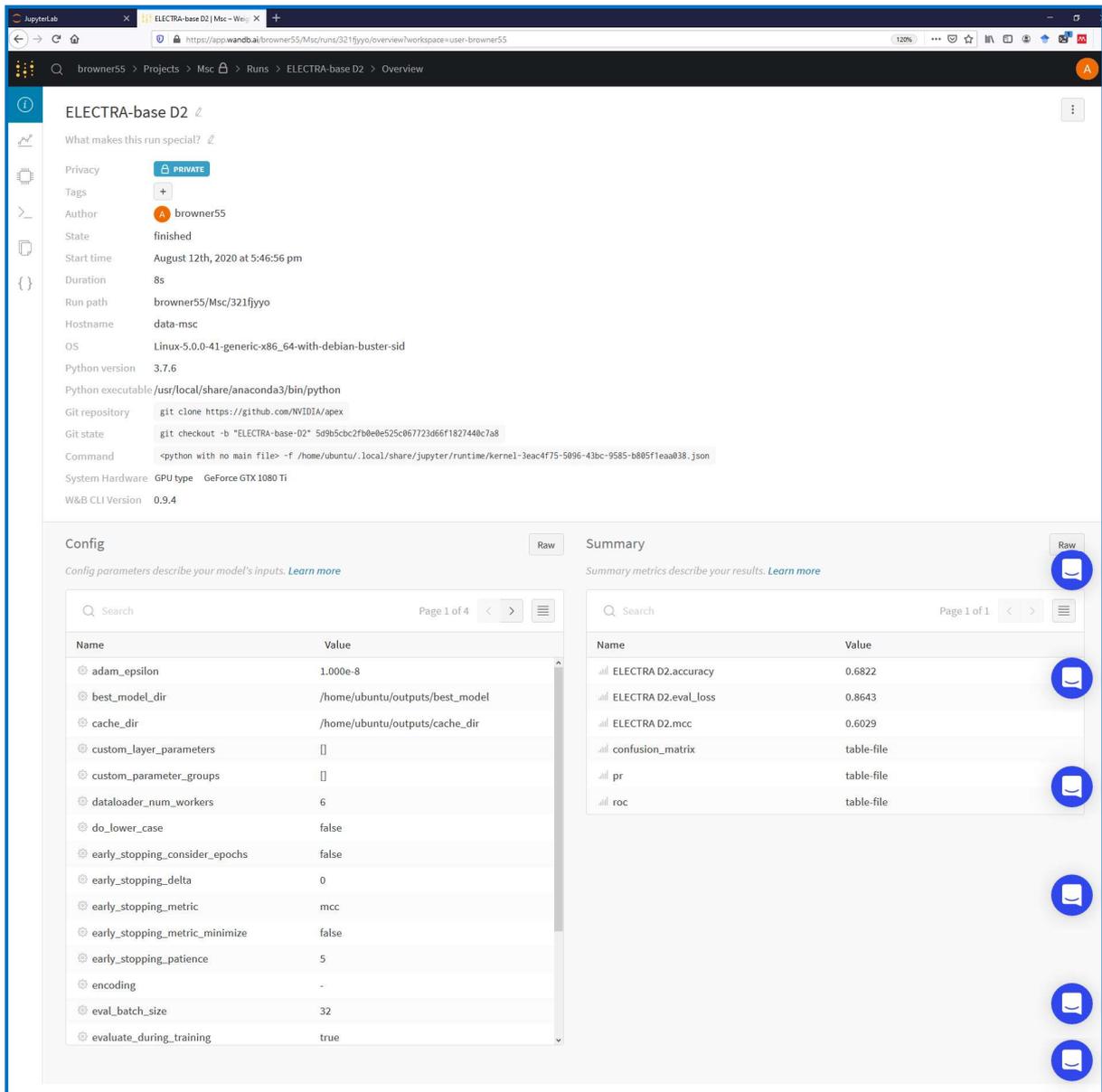


Figure 215: ELECTRA-base D2 Results

6.5 Smaller Bert D1

```
[ ]: !pip install transformers
!pip install simpletransformers
!pip install wandb

[ ]: import wandb

[ ]: wandb.login()
True

[ ]: import torch

[ ]: !git clone https://github.com/NVIDIA/apex

[ ]: cd apex

[ ]: !pip install -v --no-cache-dir --global-option="--cpp_ext" --global-option="--cuda_ext" ./

[ ]: import logging
import pandas as pd
from sklearn.metrics import accuracy_score
from simpletransformers.classification import ClassificationArgs, ClassificationModel

logging.basicConfig(level=logging.INFO)
transformers_logger = logging.getLogger("transformers")
transformers_logger.setLevel(logging.WARNING)

# Preparing train data
train_df = pd.read_csv('/home/ubuntu/d1_train.csv', error_bad_lines=False)
eval_df = pd.read_csv('/home/ubuntu/d1_eval.csv', error_bad_lines=False)
test_df = pd.read_csv('/home/ubuntu/d1_test.csv', error_bad_lines=False)

# Args for model
args7 = ClassificationArgs()
args7.evaluate_during_training = True
args7.evaluate_during_training_steps = 1000
args7.manual_seed = 4
args7.eval_batch_size = 32
args7.max_seq_length = 150
args7.train_batch_size = 64
args7.num_train_epochs = 3
args7.override_output_dir = True
args7.reprocess_input_data = True
args7.best_model_dir = '/home/ubuntu/outputs/best_model'
args7.output_dir = '/home/ubuntu/outputs/output'
args7.cache_dir = '/home/ubuntu/outputs/cache_dir'
args7.wandb_project = "Msc"
args7.wandb_kwargs = {"name": "BertMinD1"}
args7.n_gpu = 2
args7.use_early_stopping = True
args7.early_stopping_metric = "mcc"
args7.early_stopping_metric_minimize = False
args7.early_stopping_patience = 5
args7.learning_rate = 4e-5
args7.do_lower_case = True

# Create a TransformerModel
model_a7 = ClassificationModel("bert", "google/bert_uncased_L-10_H-768_A-12", use_cuda=True, num_labels=5,

# Train the model
model_a7.train_model(
    train_df,
    eval_df=eval_df,
    accuracy=accuracy_score)

# Evaluate the model
result7, model_outputs7, wrong_predictions7 = model_a7.eval_model(test_df, accuracy=accuracy_score)

wandb.log({'BertMinD1': result7})
```

Figure 216: Smaller Bert D1 Code

The screenshot displays the Wandb interface for a run named "Smaller BERT D1". The overview section provides details about the run's status, author, and environment. The configuration section lists various hyperparameters used during training. The summary section shows key performance metrics for the model.

Overview:

- What makes this run special? [?](#)
- Privacy: PRIVATE
- Tags: +
- Author: ▲ browner55
- State: finished
- Start time: July 30th, 2020 at 3:49:05 pm
- Duration: 7s
- Run path: browner55/Msc/rmegp432
- Hostname: gc-elated-hypatia
- OS: Linux-5.0.0-41-generic-x86_64-with-debian-buster-sid
- Python version: 3.7.6
- Python executable: /usr/local/share/anaconda3/bin/python
- Git repository: git clone https://github.com/NVIDIA/apex
- Git state: git checkout -b "Smaller-BERT-D1" 459de22d59c64e30fd4b368c368c5b74e269f3dd
- Command: `<python with no main file> -f /home/ubuntu/.local/share/jupyter/runtime/kernel-d5a57472-3a9c-40c0-aad5-81ed977443a1.json`
- System Hardware: GPU type GeForce GTX 1080 Ti
- W&B CLI Version: 0.9.4

Config:

Name	Value
adam_epsilon	1.000e-8
best_model_dir	/home/ubuntu/outputs/best_model
cache_dir	/home/ubuntu/outputs/cache_dir
custom_layer_parameters	[]
custom_parameter_groups	[]
dataloader_num_workers	6
do_lower_case	true
early_stopping_consider_epochs	false
early_stopping_delta	0
early_stopping_metric	mcc
early_stopping_metric_minimize	false
early_stopping_patience	5
encoding	-
eval_batch_size	32
evaluate_during_training	true

Summary:

Name	Value
BertMinD1.accuracy	0.6719
BertMinD1.eval_loss	0.912
BertMinD1.mcc	0.5901
confusion_matrix	table-file
pr	table-file
roc	table-file

Figure 217: Smaller Bert D1 Results

6.6 Experiment 6 Smaller Bert D2

```
[ ]: !pip install transformers
!pip install simpletransformers==0.45.5
!pip install wandb

[ ]: import wandb

[ ]: wandb.login()
True

[ ]: import torch

[ ]: !git clone https://github.com/NVIDIA/apex

[ ]: cd apex

[ ]: !pip install -v --no-cache-dir --global-option="--cpp_ext" --global-option="--cuda_ext" ./

[ ]: import logging
import pandas as pd
from sklearn.metrics import accuracy_score
from simpletransformers.classification import ClassificationArgs, ClassificationModel

logging.basicConfig(level=logging.INFO)
transformers_logger = logging.getLogger("transformers")
transformers_logger.setLevel(logging.WARNING)

# Preparing train data
train_df1 = pd.read_csv('/home/ubuntu/d2_train.csv', error_bad_lines=False)
eval_df1 = pd.read_csv('/home/ubuntu/d2_eval.csv', error_bad_lines=False)
test_df1 = pd.read_csv('/home/ubuntu/d2_test.csv', error_bad_lines=False)

# Args for model
model_args5 = ClassificationArgs()
model_args5.evaluate_during_training = True
model_args5.evaluate_during_training_steps = 1000
model_args5.manual_seed = 4
model_args5.eval_batch_size = 32
model_args5.max_seq_length = 235
model_args5.train_batch_size = 64
model_args5.num_train_epochs = 3
model_args5.override_output_dir = True
model_args5.reprocess_input_data = True
model_args5.best_model_dir = '/home/ubuntu/outputs/best_model'
model_args5.output_dir = '/home/ubuntu/outputs/output'
model_args5.cache_dir = '/home/ubuntu/outputs/cache_dir'
model_args5.wandb_project = "Msc"
model_args5.wandb_kwargs = {"name": "BertMin2"}
model_args5.n_gpu = 2
model_args5.use_early_stopping = True
model_args5.early_stopping_metric = "mcc"
model_args5.early_stopping_metric_minimize = False
model_args5.early_stopping_patience = 5
model_args5.learning_rate = 4e-5
model_args5.do_lower_case = True

# Create a TransformerModel
model_a5 = ClassificationModel("bert", "google/bert_uncased_L-12_H-512_A-8", use_cuda=True, num_labels=5, a

# Train the model
model_a5.train_model(
    train_df1,
    eval_df=eval_df1,
    accuracy= accuracy_score)

# Evaluate the model
result, model_outputs, wrong_predictions = model_a5.eval_model(test_df1, accuracy=accuracy_score)

wandb.log({'BertMinD2': result})
```

Figure 218: Smaller Bert D2 Code

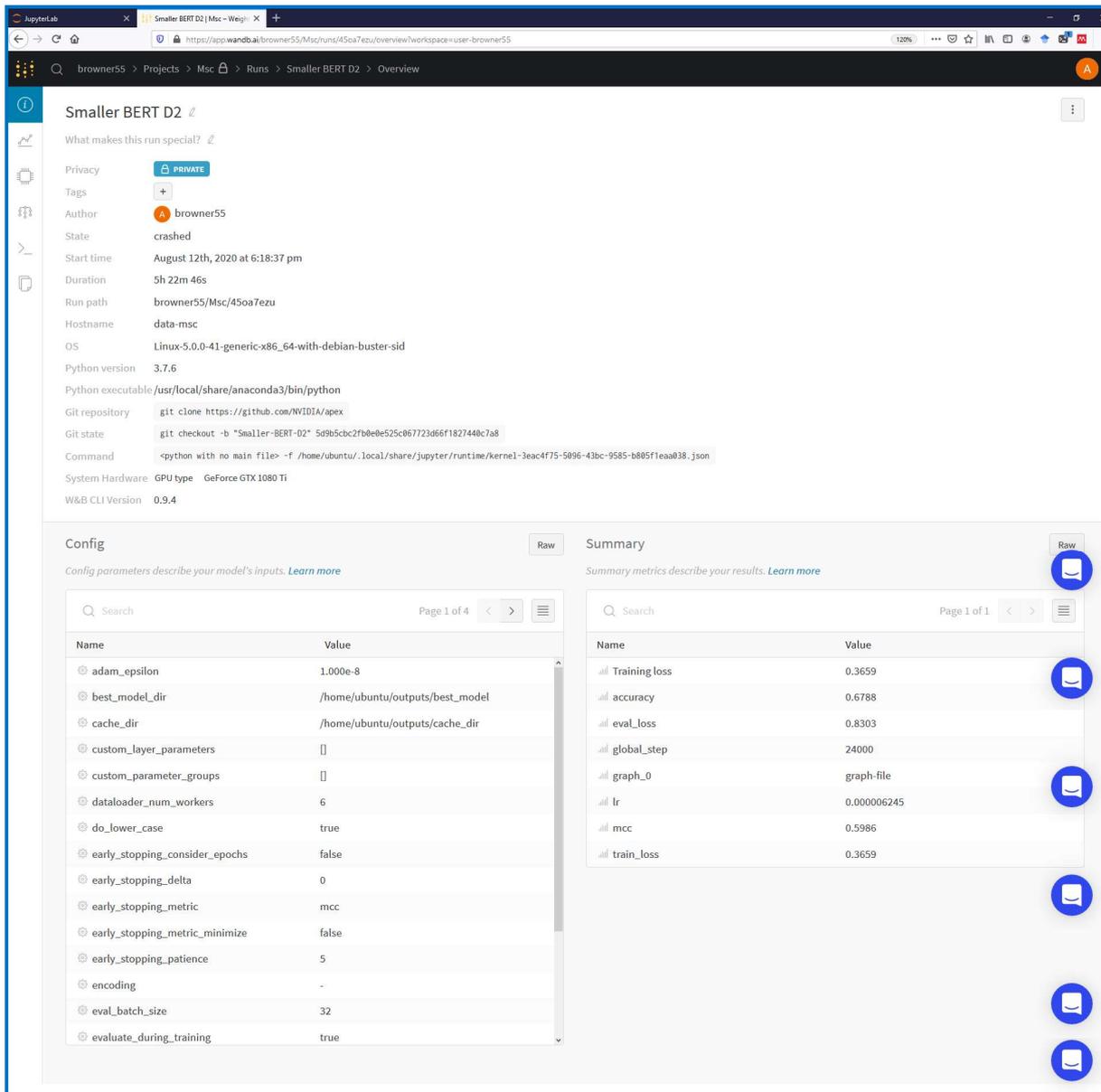


Figure 219: Smaller Bert D2 Results

7 Conclusion

By utilizing the information above the process of implementing the candidates project can be achieved. It should be noted that due to the randomness of Neural Networks the exact results may not be achieved. To control this the candidate used manual seed for the Simple Transformers library however the candidate didn't realize that another library used randomness also. Therefore, there will be a slight difference in the results achieved to the Technical Report when the code is re-run. Finally, the project on Weights and Biases has been made public until the end of September and can be viewed here¹⁵.

¹⁵ <https://app.wandb.ai/browner55/Msc?workspace=user-browner55>