National College of Ireland

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: 

Date: 28/09/2020

PLEASE READ THE FOLLOWING INSTRUCTIONS AND CHECKLIST

Attach a completed copy of this sheet to each project (including multiple copies)	□
Attach a Moodle submission receipt of the online project submission, to each project (including multiple copies).	□

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.	□

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.

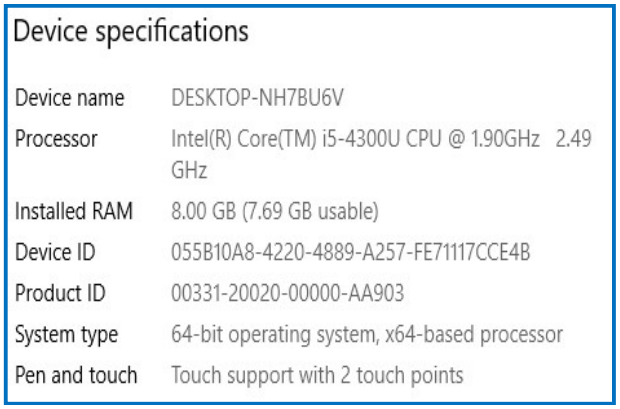


Figure 1: Laptop Specifications

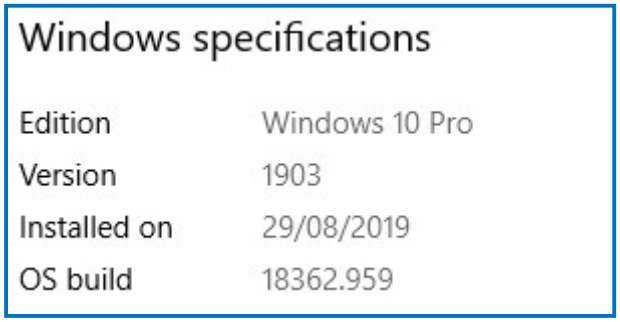


Figure 2: Windows Specification

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 website[1] 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

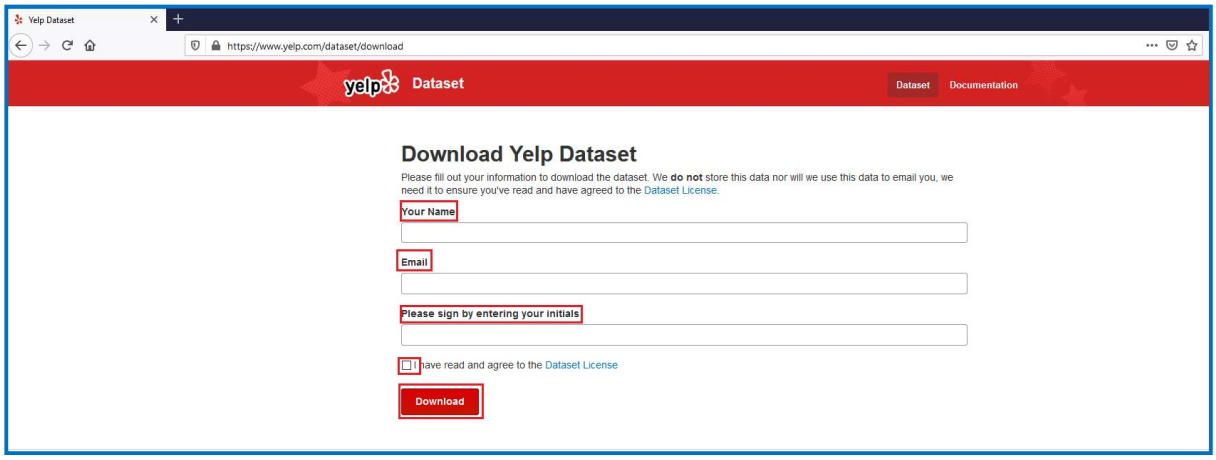


Figure 3: Yelp Open Dataset License

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 website[2]

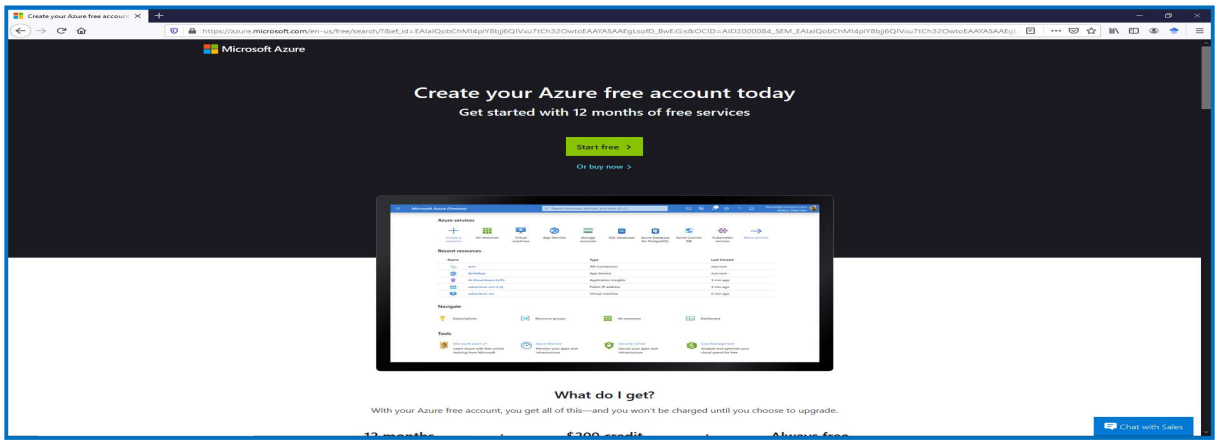


Figure 4: Azure Set UP

[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 page[3].

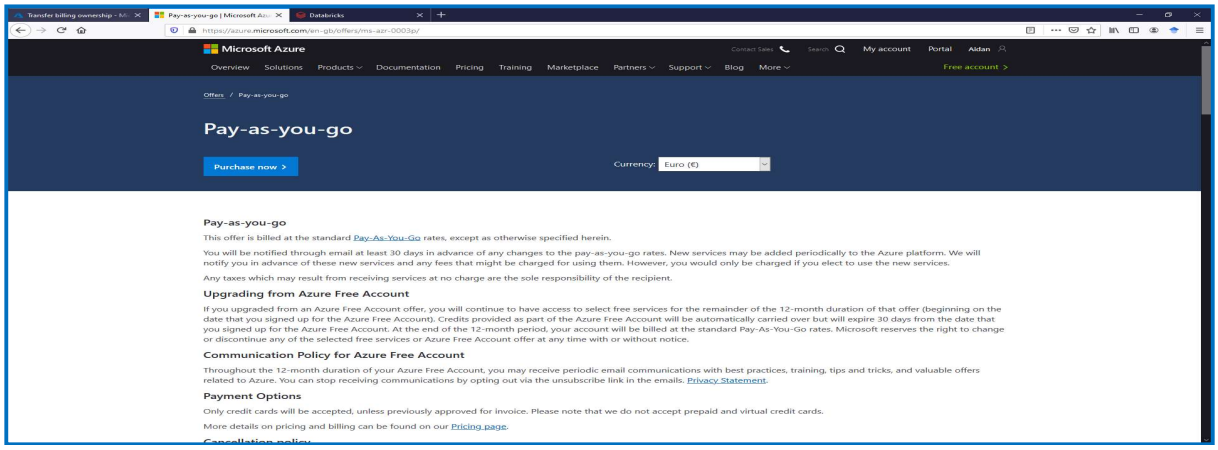


Figure 5: Pay-As-You-Go Subscriptions (1 of 2)

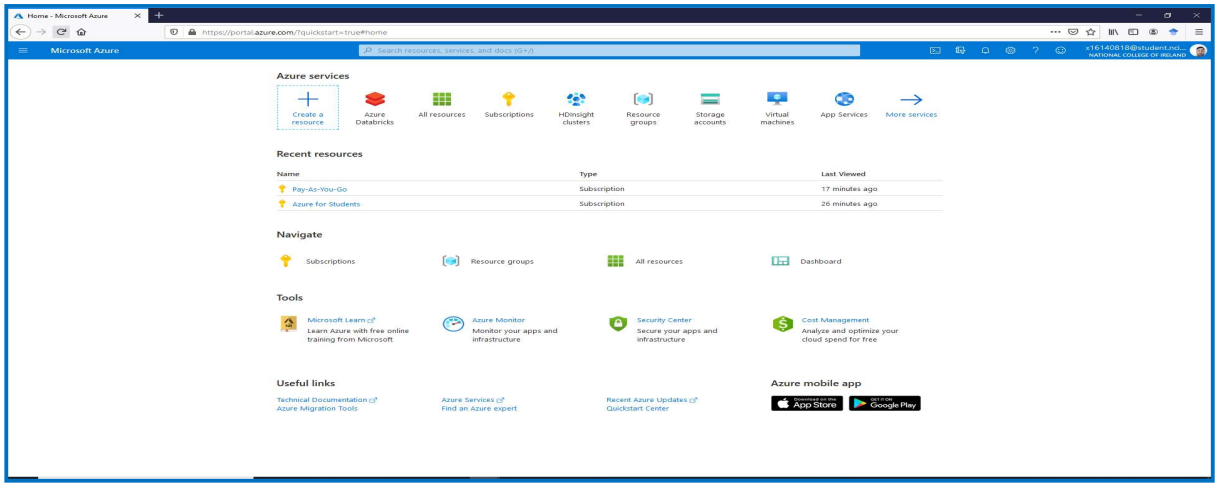


Figure 6: Pay-As-You-Go Subscriptions (2 of 2)

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/

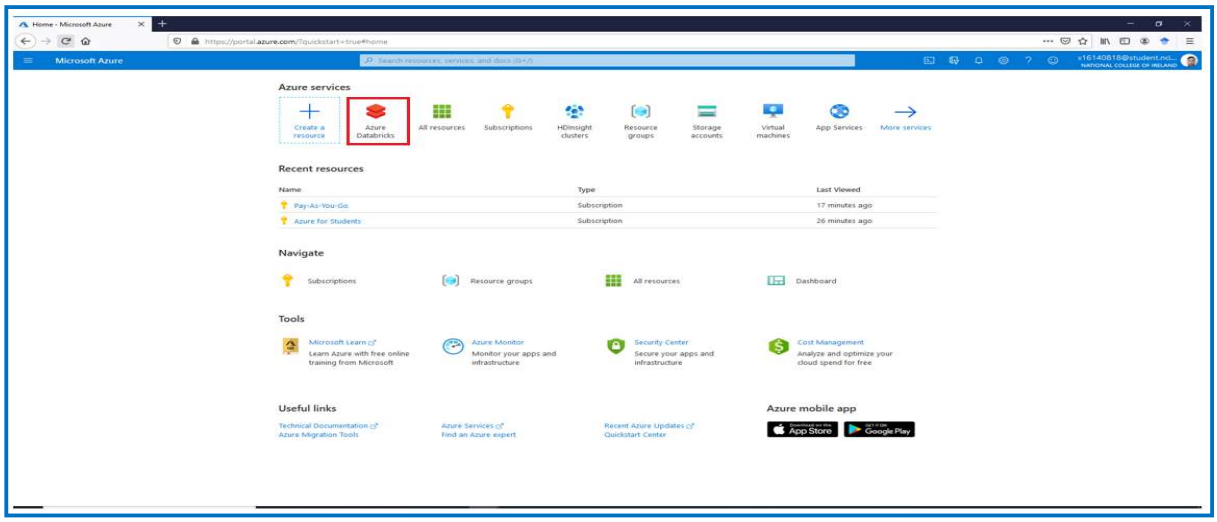


Figure 7: Databricks Set-up

4. Click Add

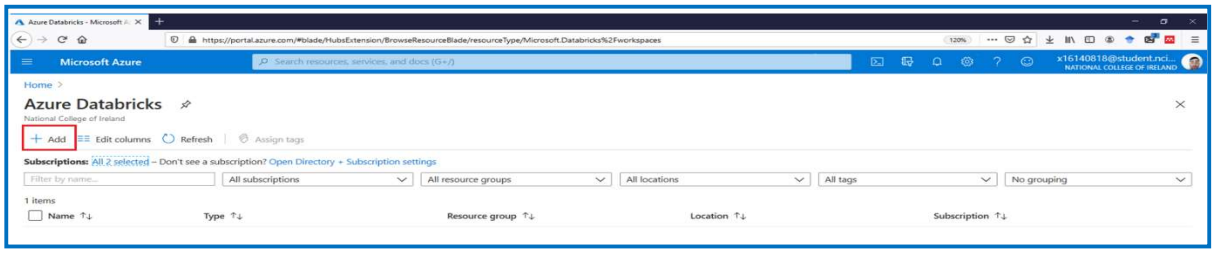


Figure 8: Databricks Set-up

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

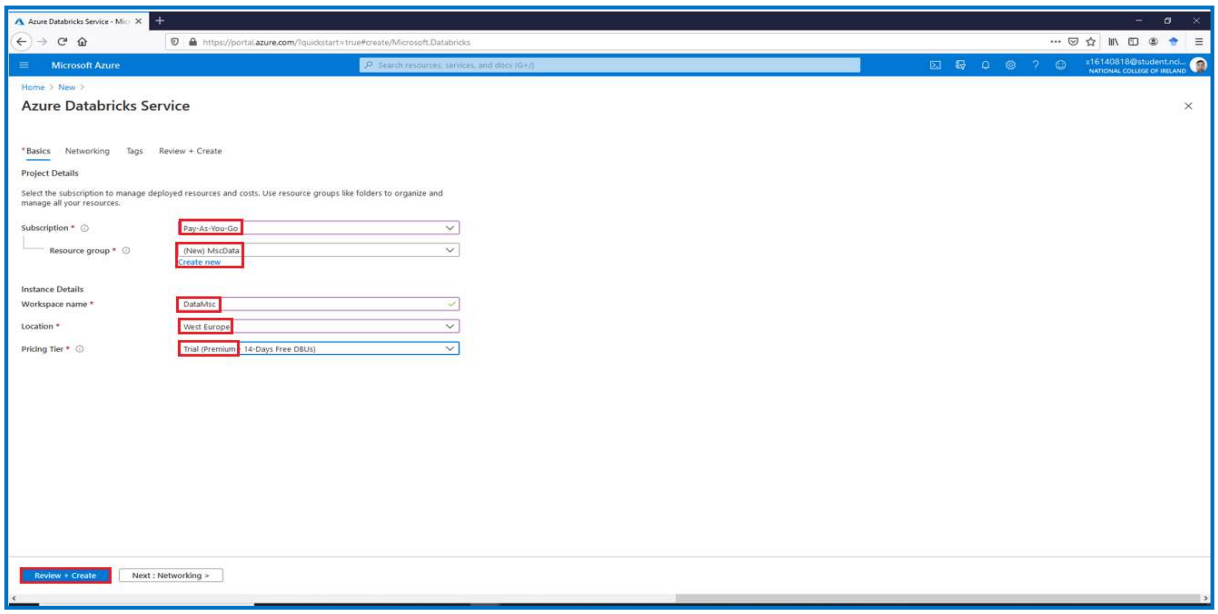


Figure 9: Databricks Set-up

6. If happy with details, click Create

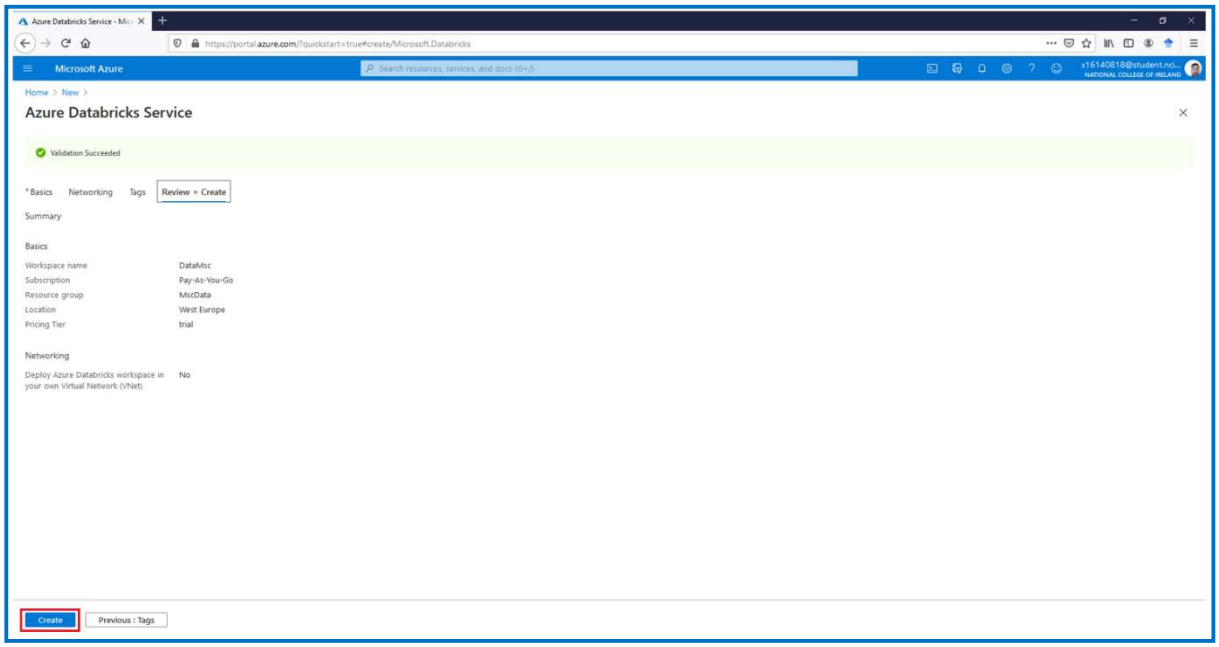


Figure 10: Databricks Set-up

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

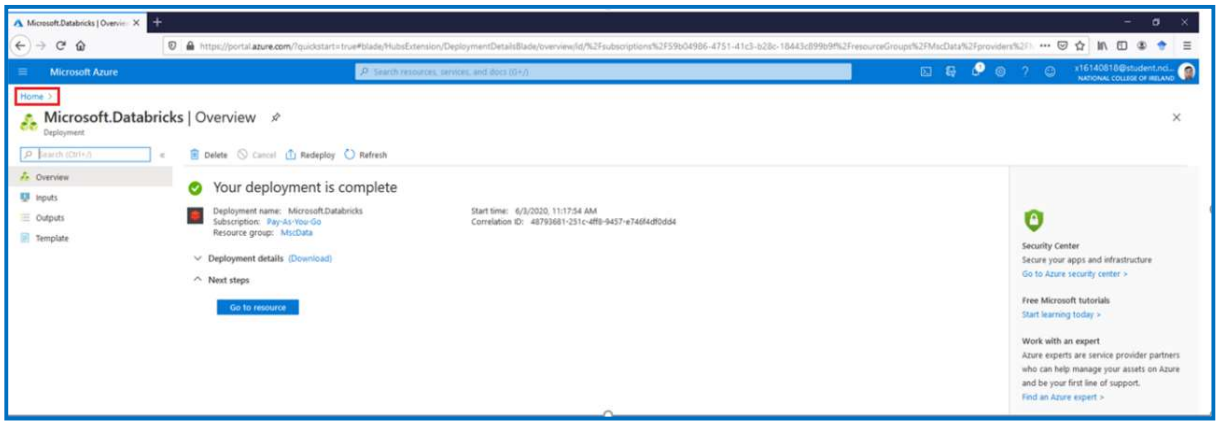


Figure 11: Databricks Set-up

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

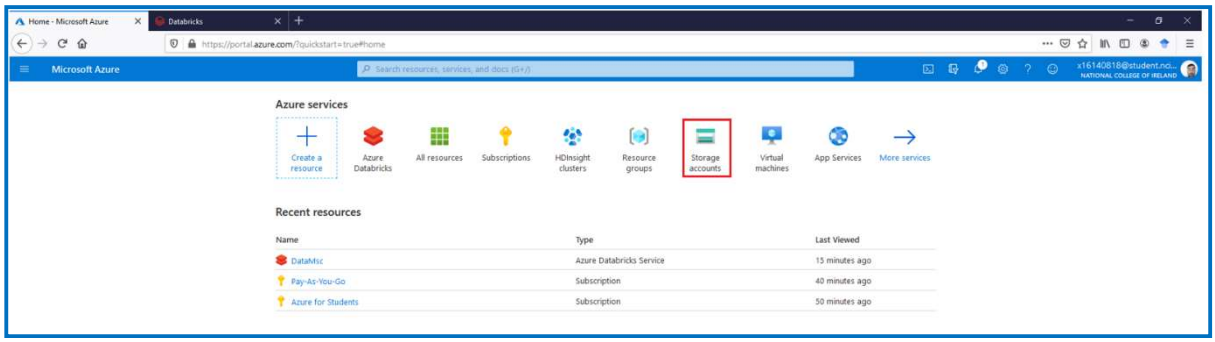


Figure 12: Storage Account Set-up

9. Click Add

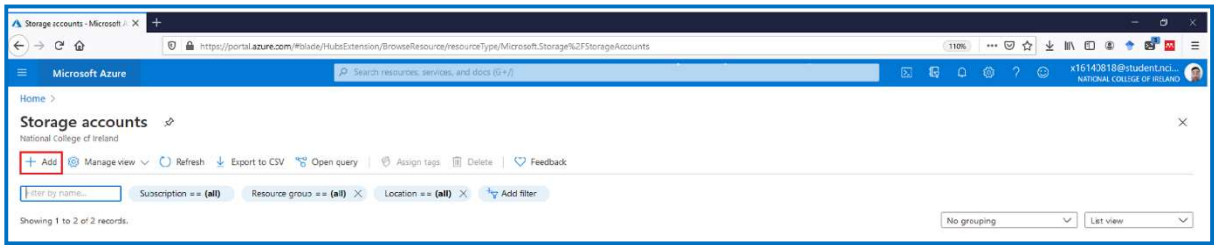


Figure 13: Storage Account Set-up

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

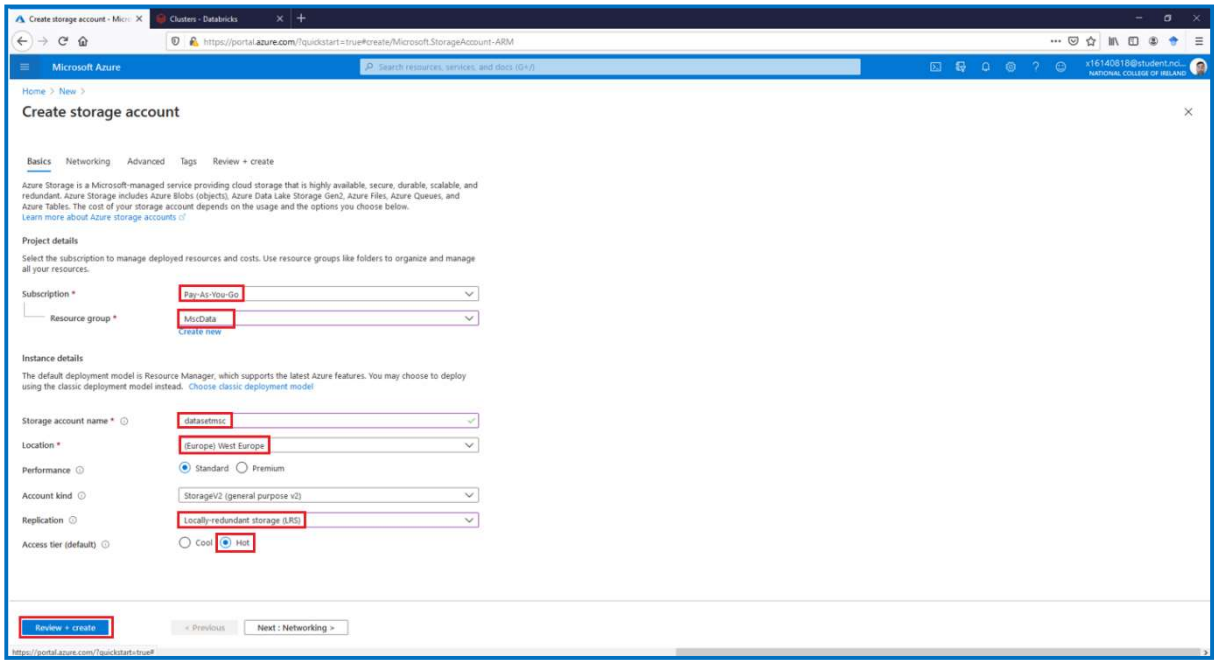


Figure 14: Storage Account Set-up

11. Click Create

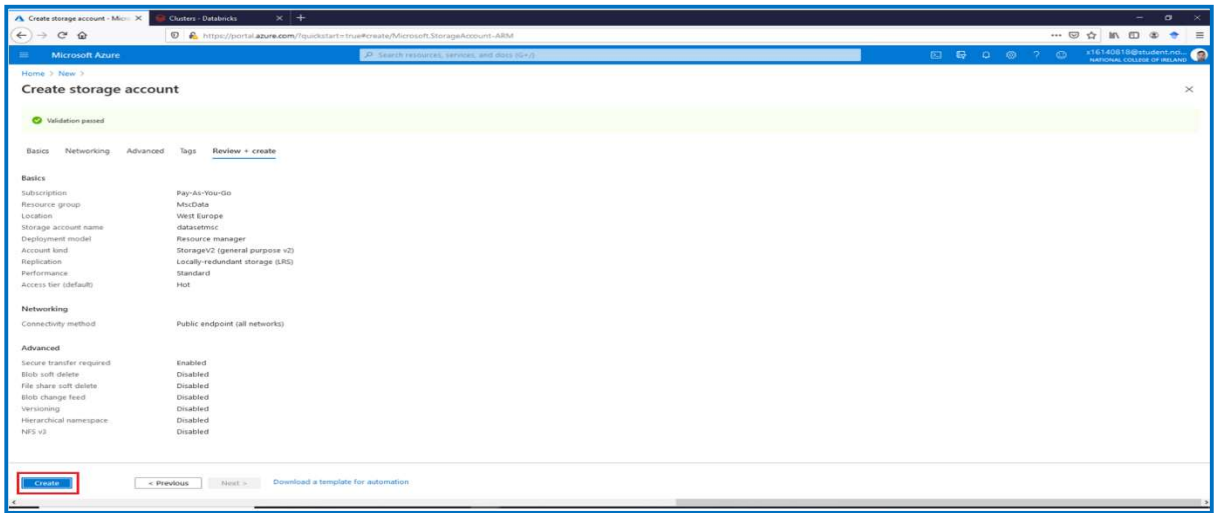


Figure 15: Storage Account Set-up

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

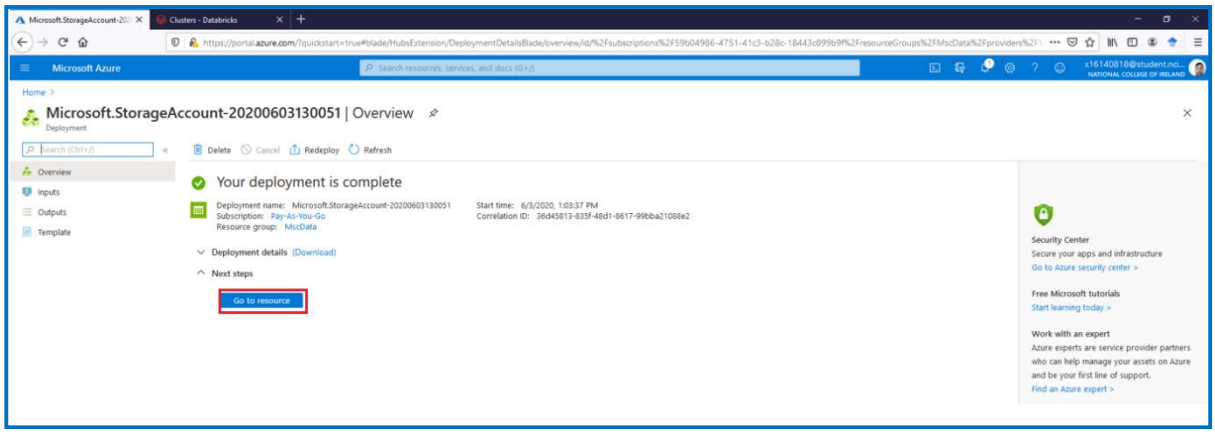


Figure 16: JSON Files Upload

13. When in the Storage account click on Storage Explorer

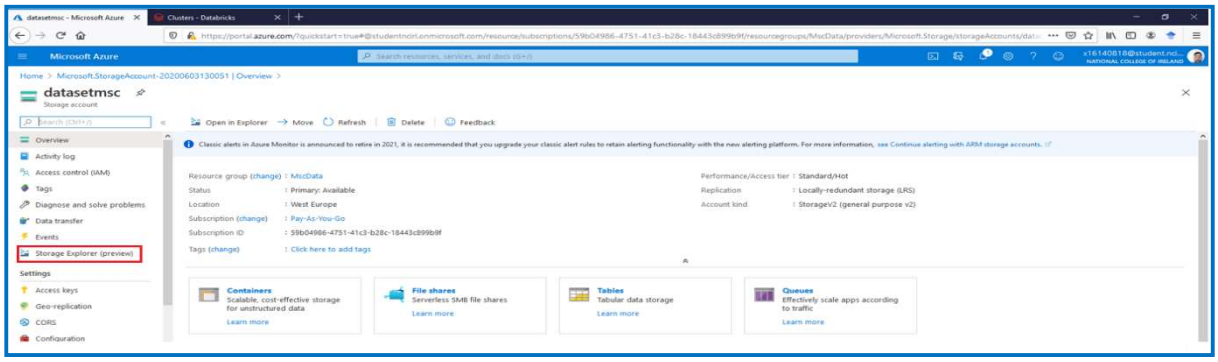


Figure 17: JSON Files Upload

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

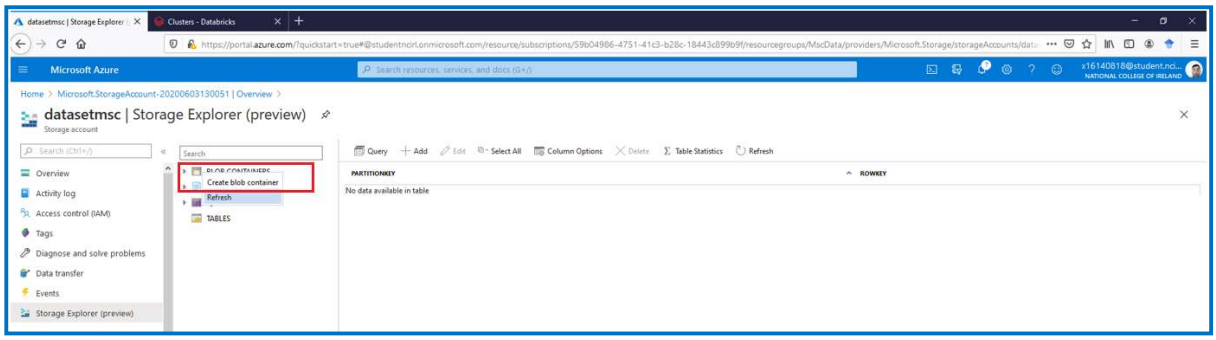


Figure 18: JSON Files Upload

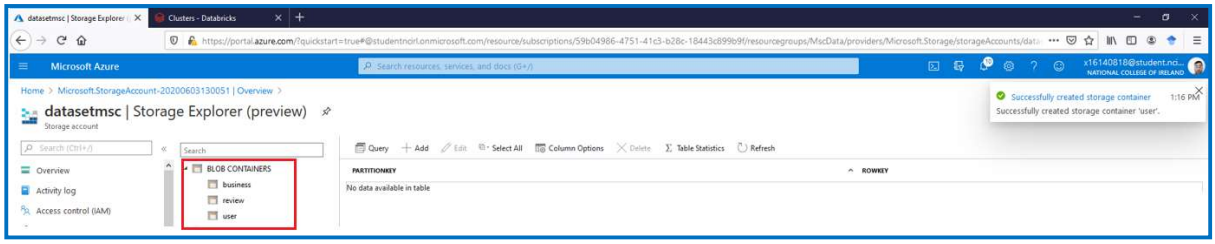


Figure 19: JSON Files Upload

14. Click on each container and then click upload

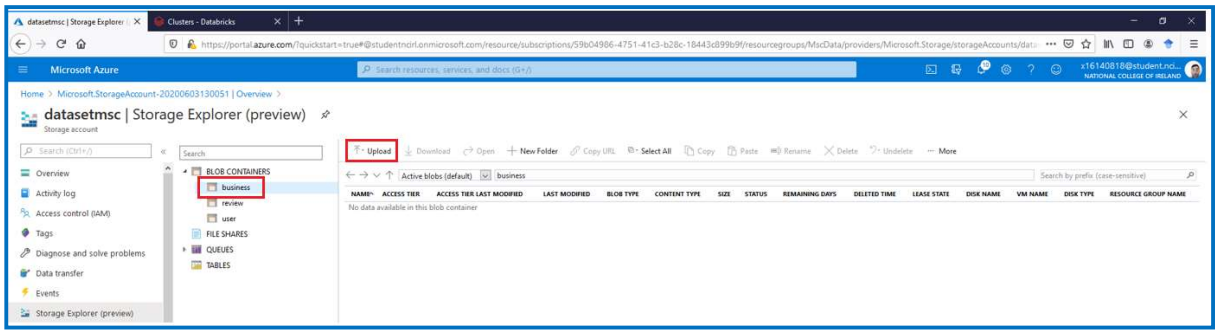


Figure 20: JSON Files Upload

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

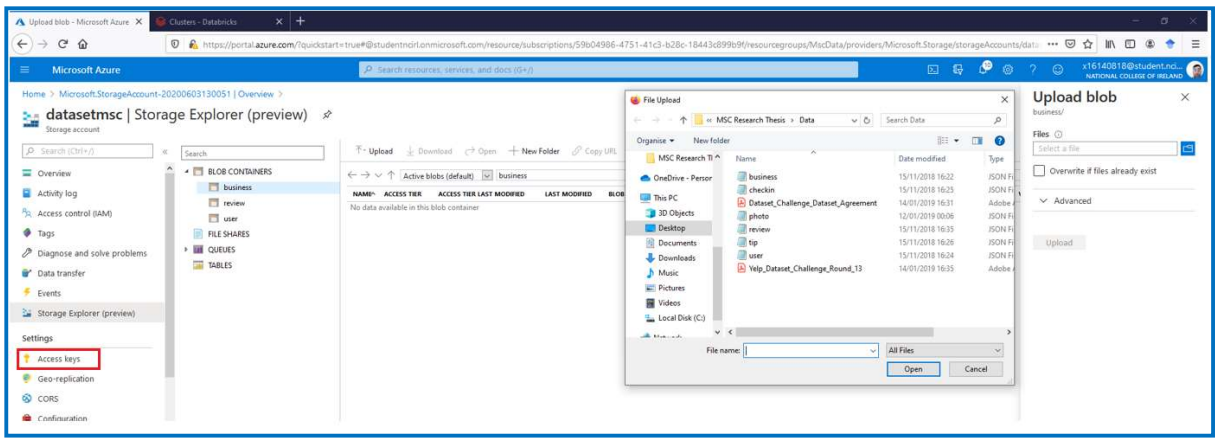


Figure 21: JSON Files Upload

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

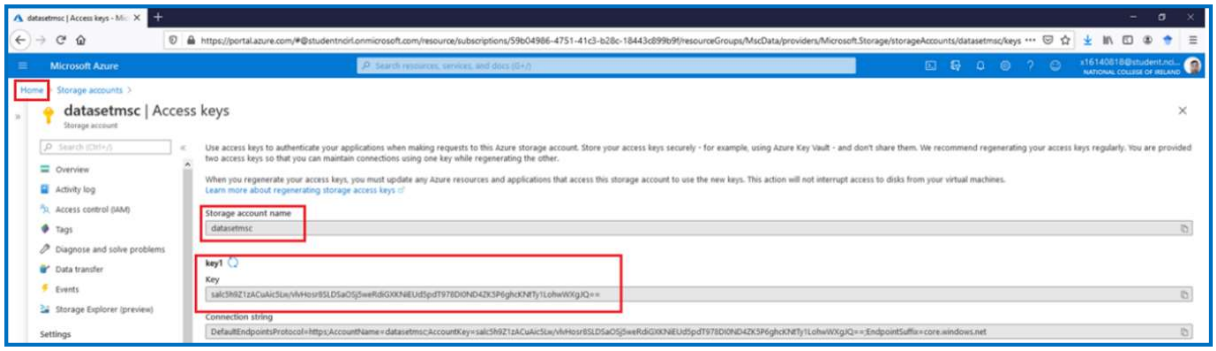


Figure 22: Key Vault and Secret (1 of 17)

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

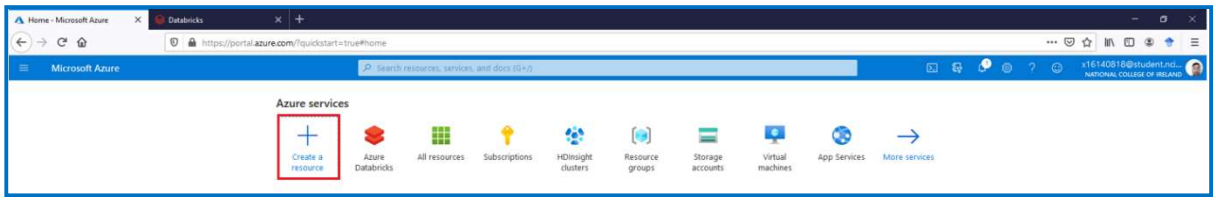


Figure 23: Key Vault and Secret (2 of 17)

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

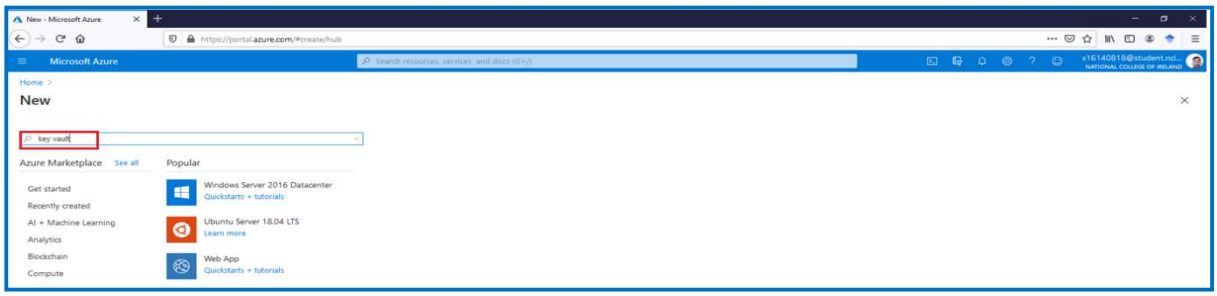


Figure 24: Key Vault and Secret (3 of 17)

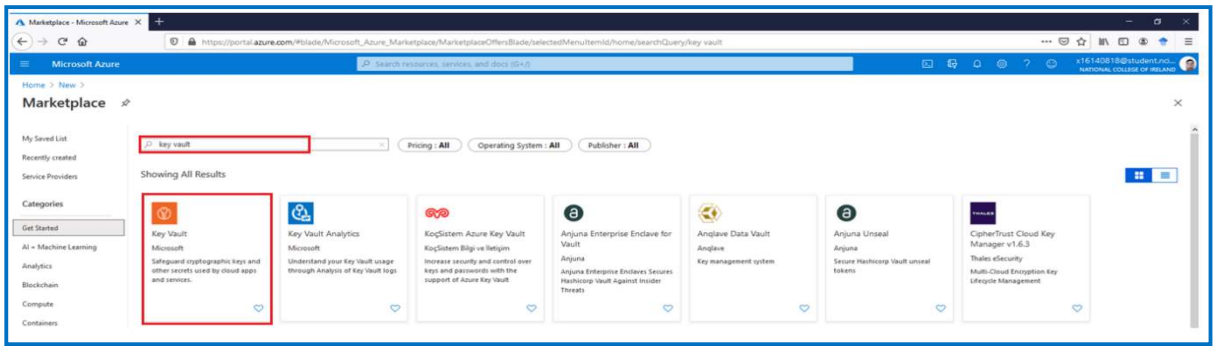


Figure 25: Key Vault and Secret (4 of 17)

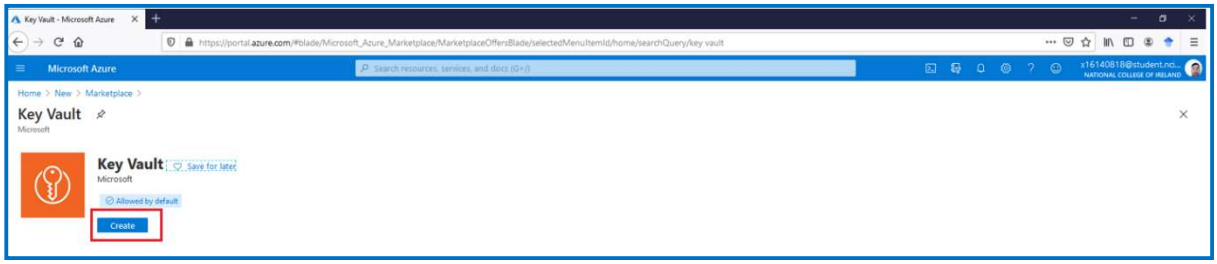


Figure 26: Key Vault and Secret (5 of 17)

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

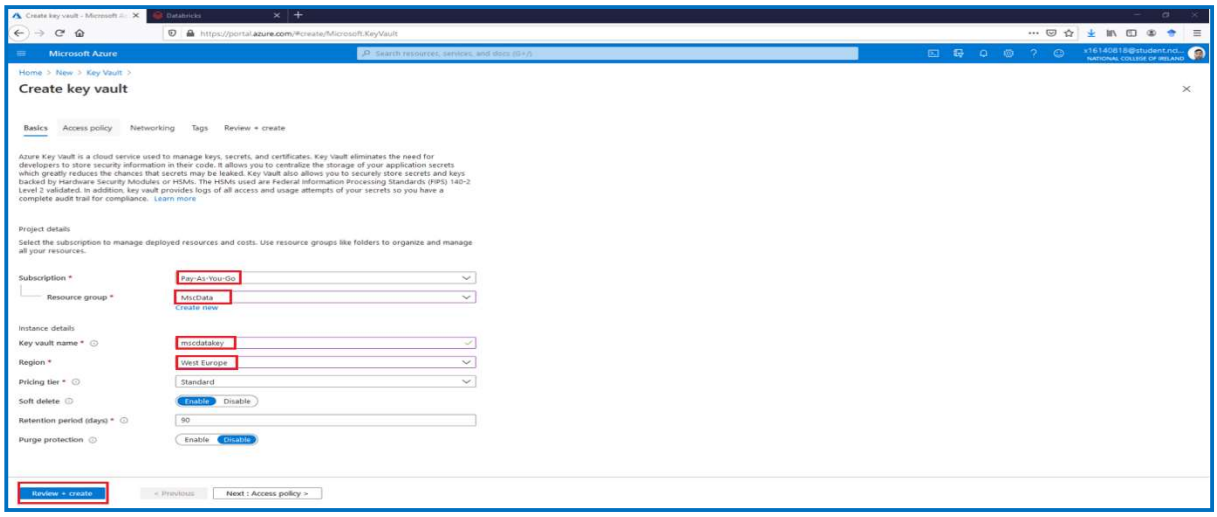


Figure 27: Key Vault and Secret (6 of 17)

20. When validation has passed click Create

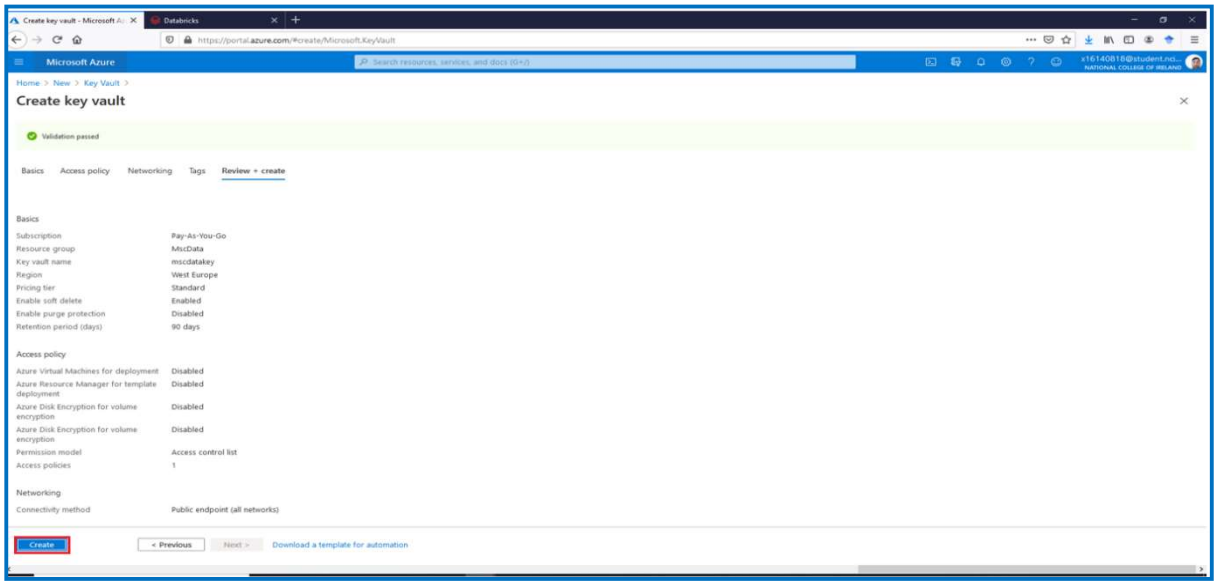


Figure 28: Key Vault and Secret (7 of 17)

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

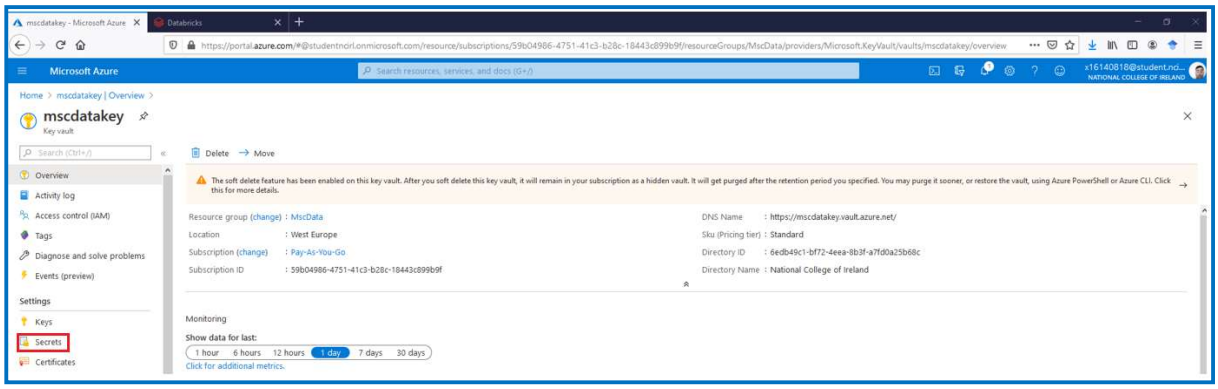


Figure 29: Key Vault and Secret (8 of 17)

22. Click Generate/Import

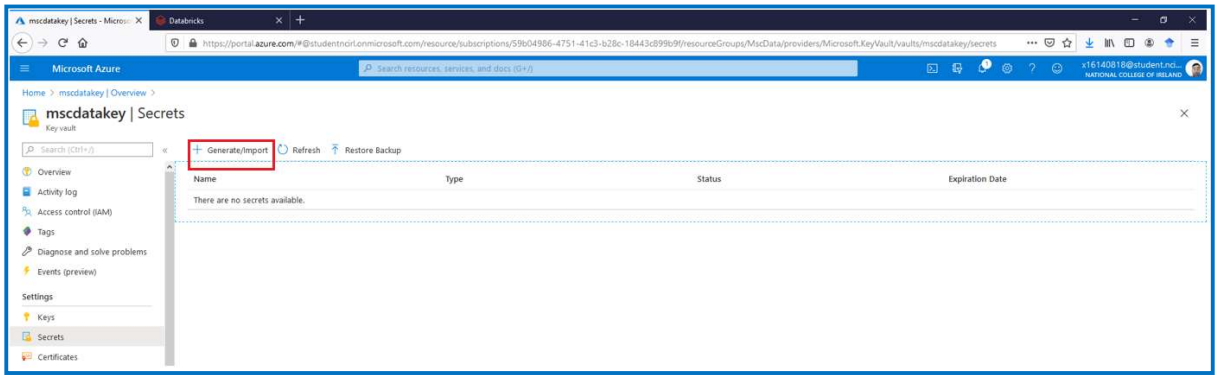


Figure 30: Key Vault and Secret (9 of 17)

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

.

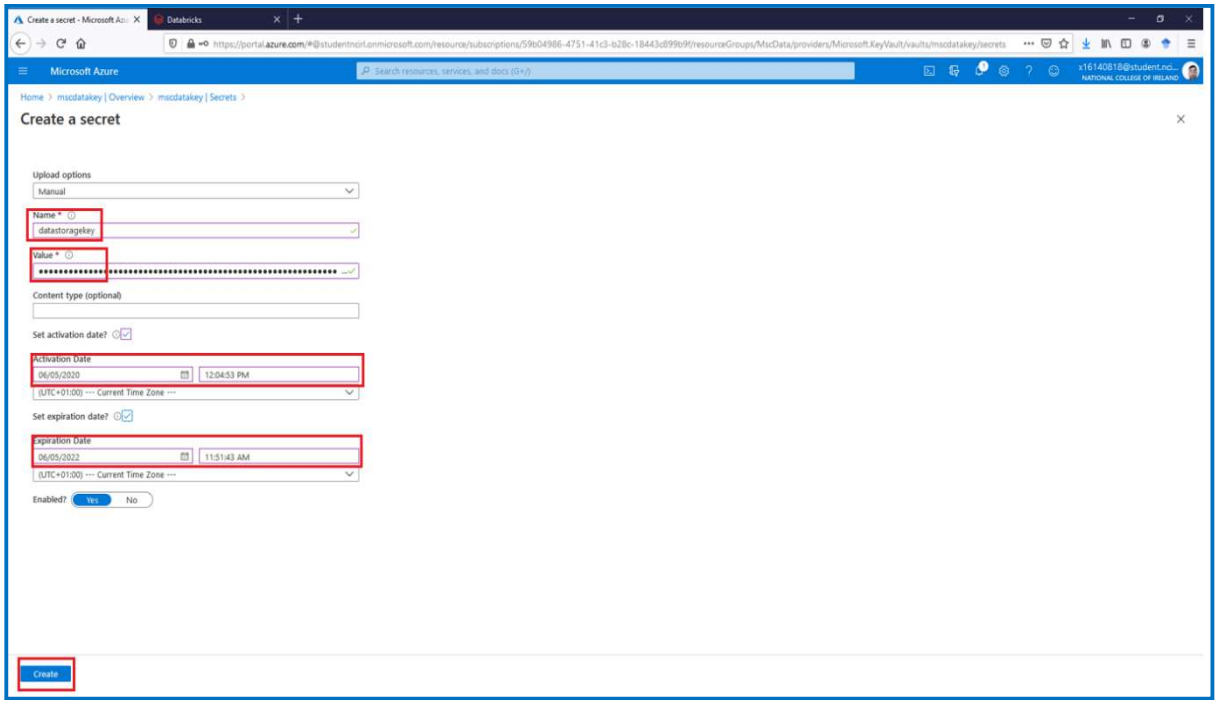


Figure 31: Key Vault and Secret (10 of 17)

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

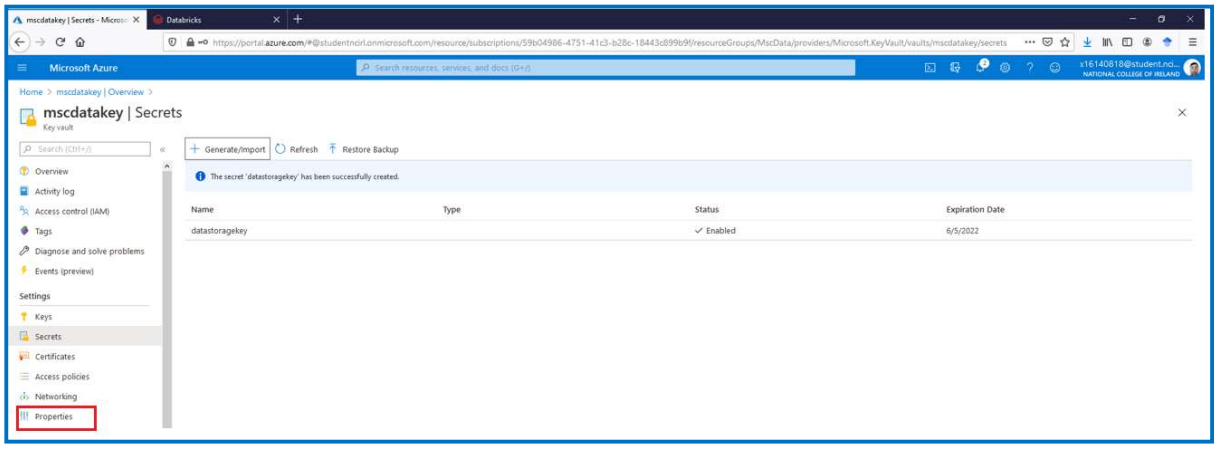


Figure 32: Key Vault and Secret (11 of 17)

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

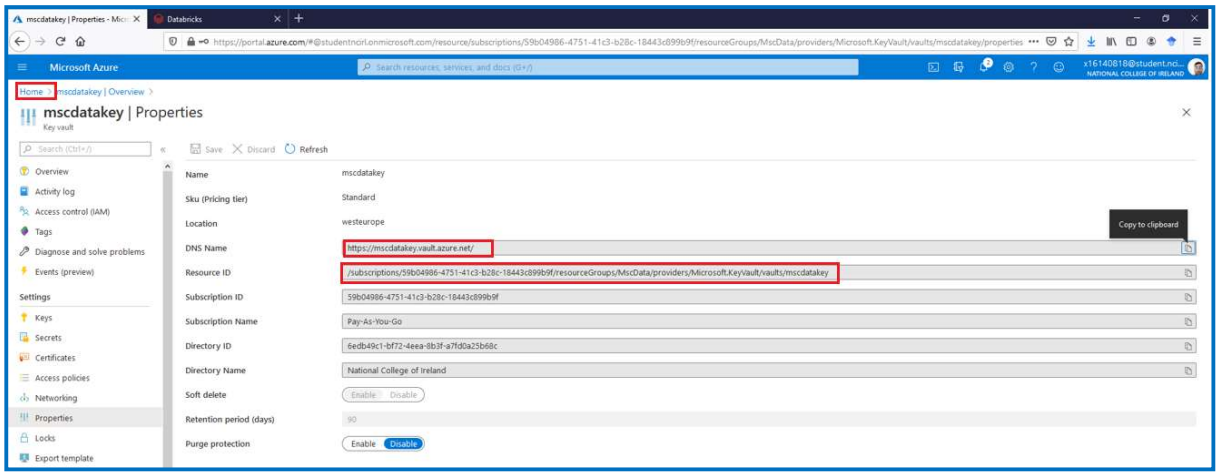


Figure 33: Key Vault and Secret (12 of 17)

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

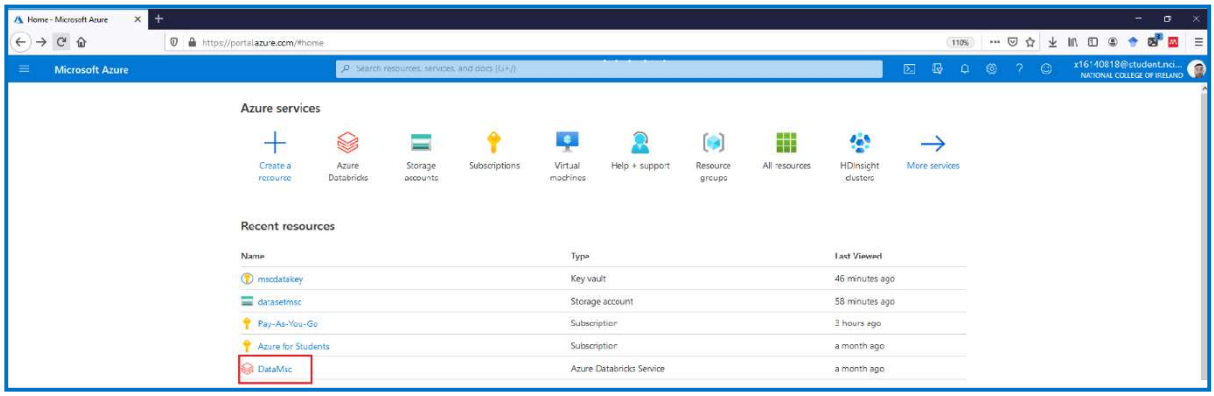


Figure 34: Key Vault and Secret (13 of 17)

27. Click Launch Workspace

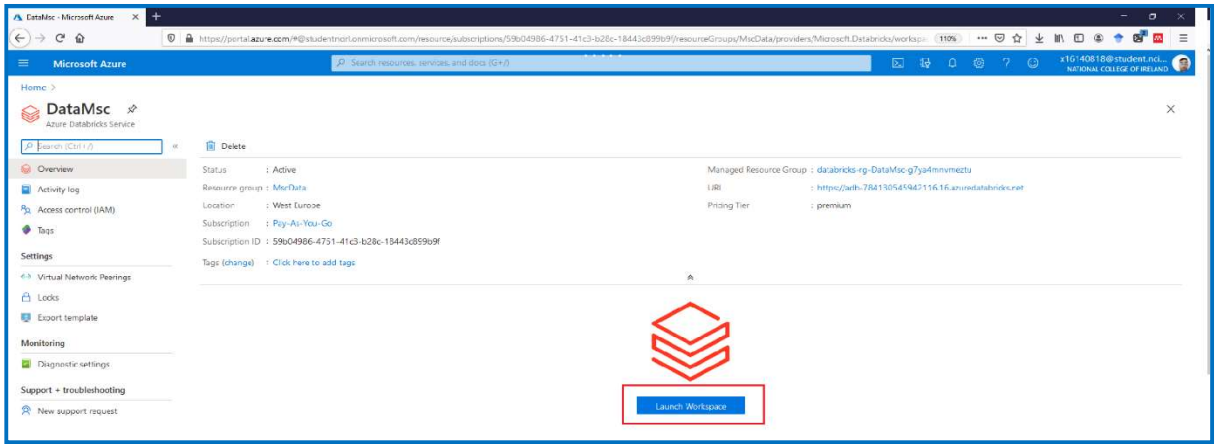


Figure 35: Key Vault and Secret (14 of 17)

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

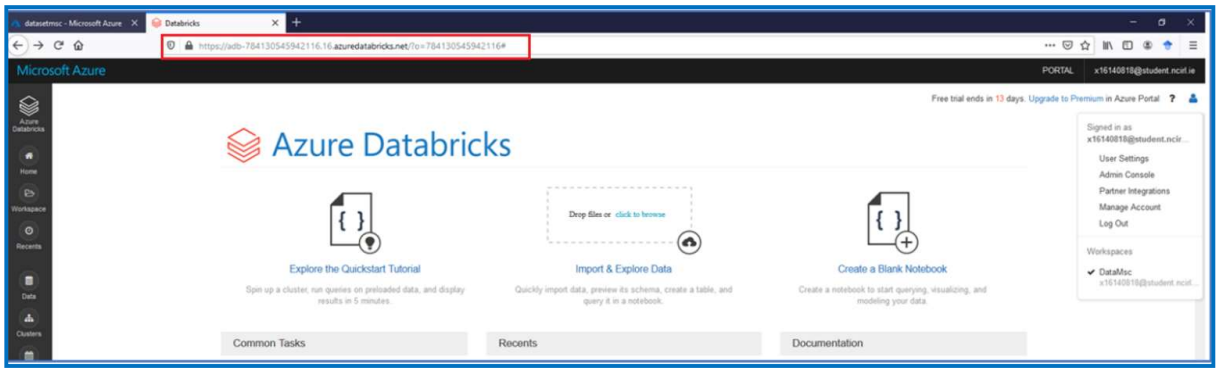


Figure 36: Key Vault and Secret (15 of 17)

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

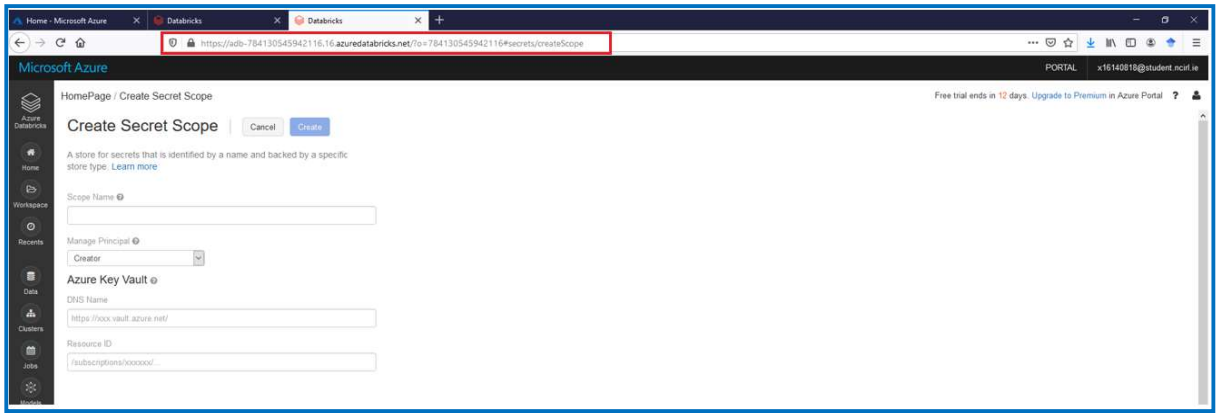


Figure 37: Key Vault and Secret (16 of 17)

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.

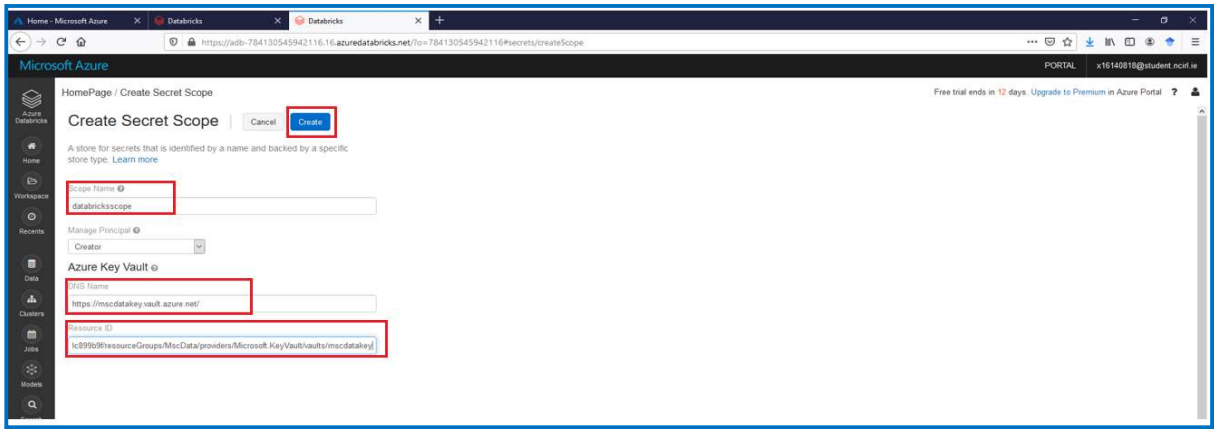


Figure 38: Key Vault and Secret (17 of 17)

3.3 Google

1. A Google account can be set up at Account Setup page[4]. By setting up the account the user will have access to products such as Google Colaboratory (Colab), Google Drive (Drive) and Gmail.

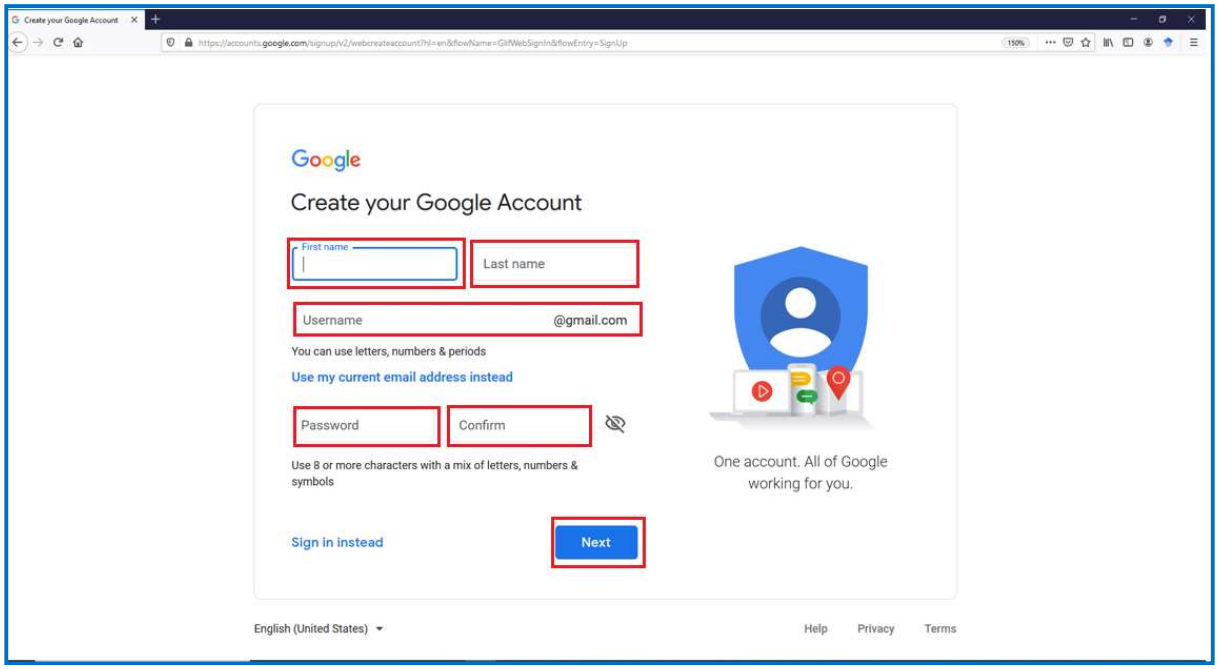


Figure 39: Google Account Set-up

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 website[5]. 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/

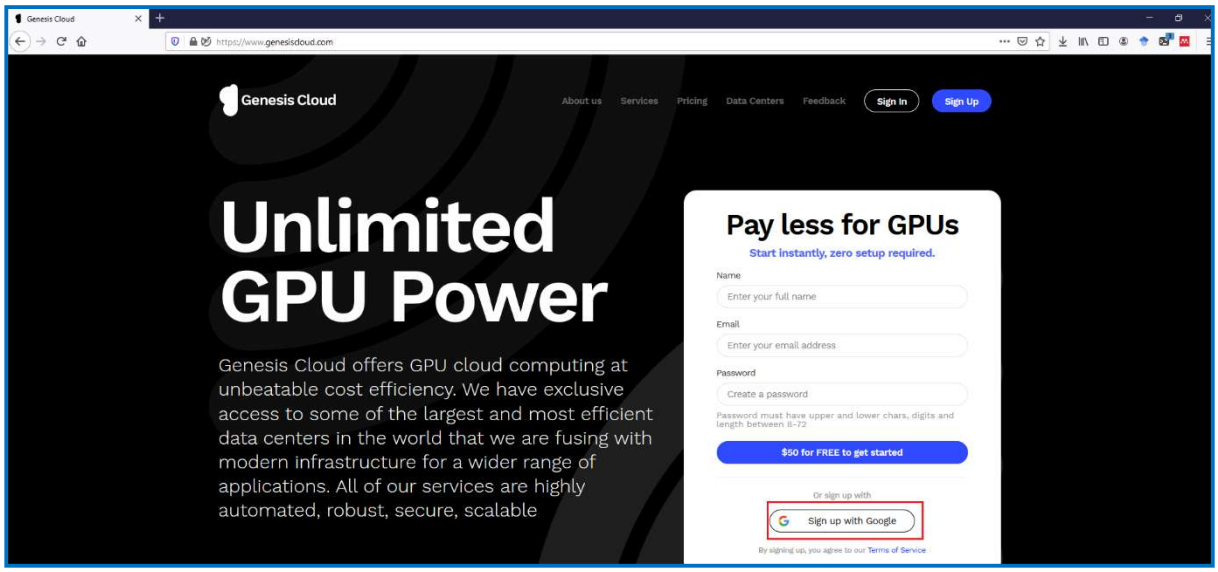


Figure 40: Genesis Cloud Account Set-up

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 website[6].

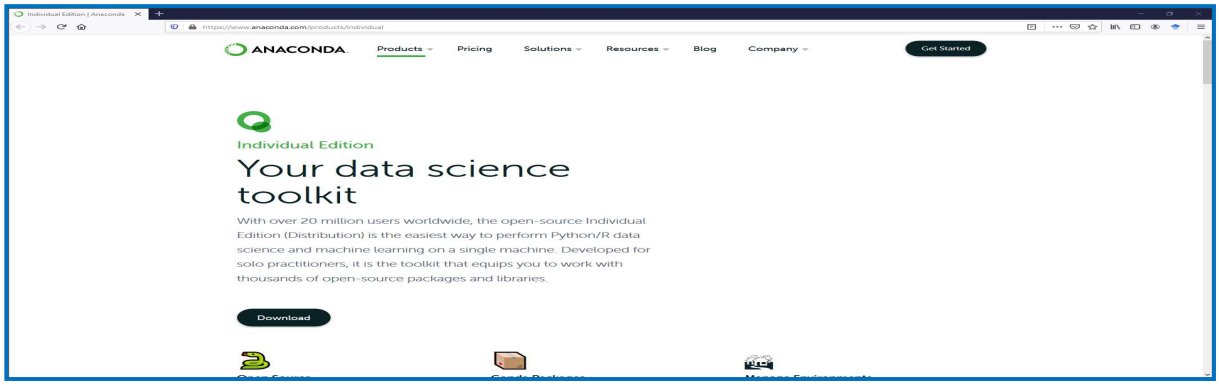


Figure 41: Anaconda Download (1 of 3)

[6] https://www.anaconda.com/products/individual

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

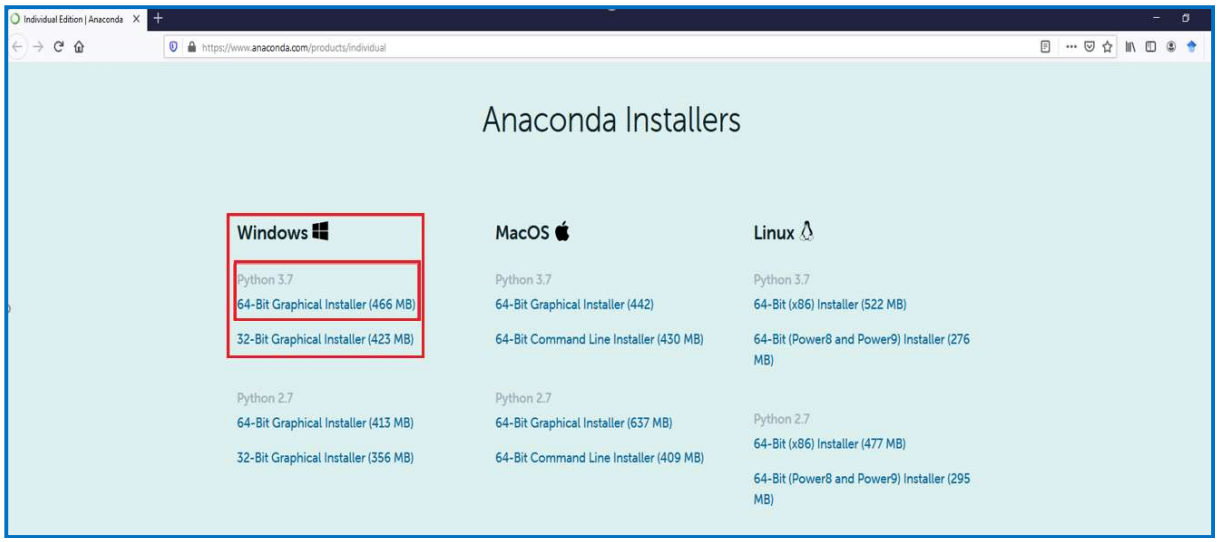


Figure 42: Anaconda Download (2 of 3)

2. Run the file after download.

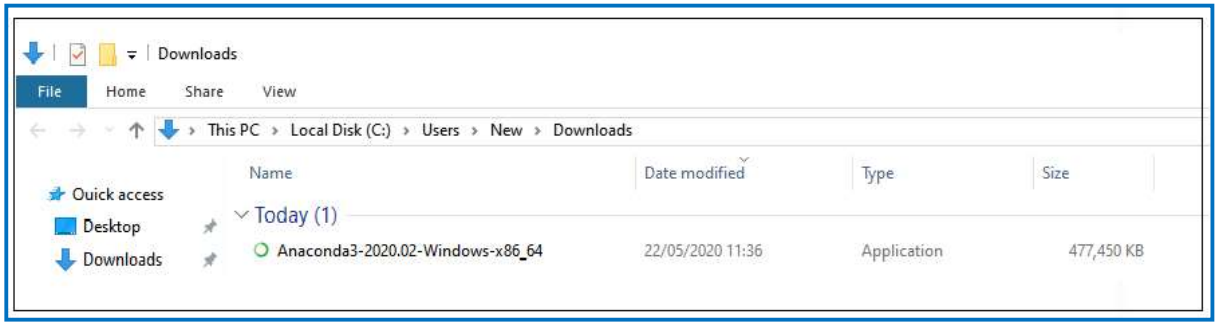


Figure 43: Anaconda Download (3 of 3)

3. After the file has opened click next

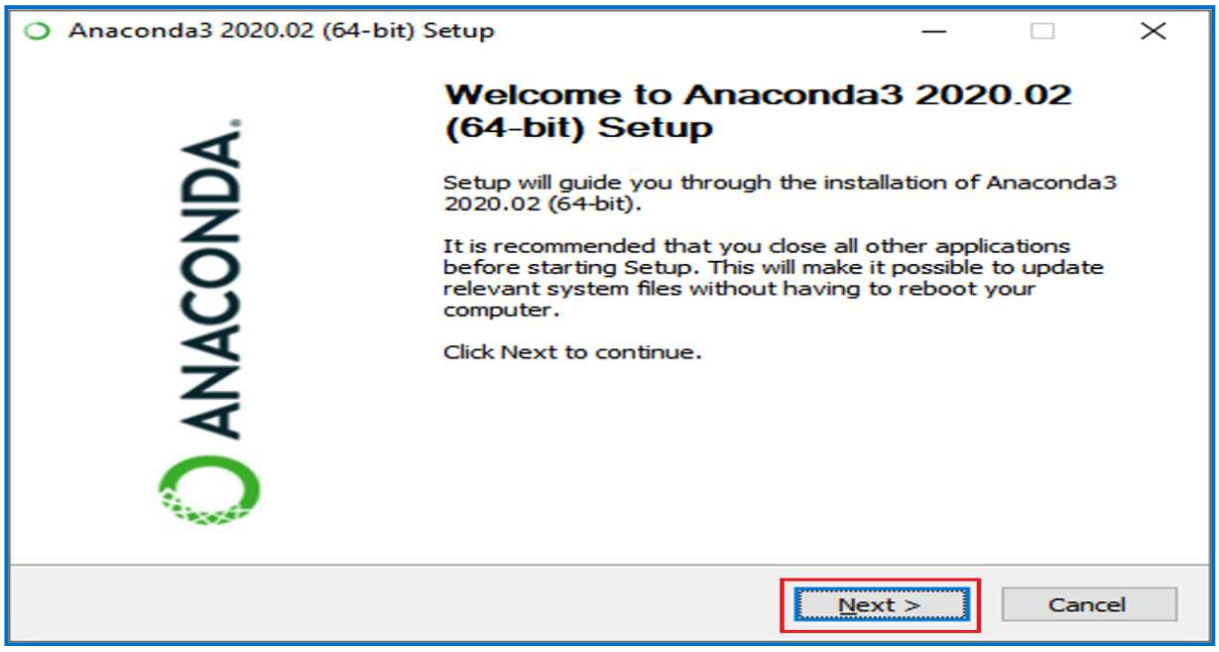


Figure 44: Anaconda Set-up (1 of 21)

4. Click I agree on the Licence Agreement

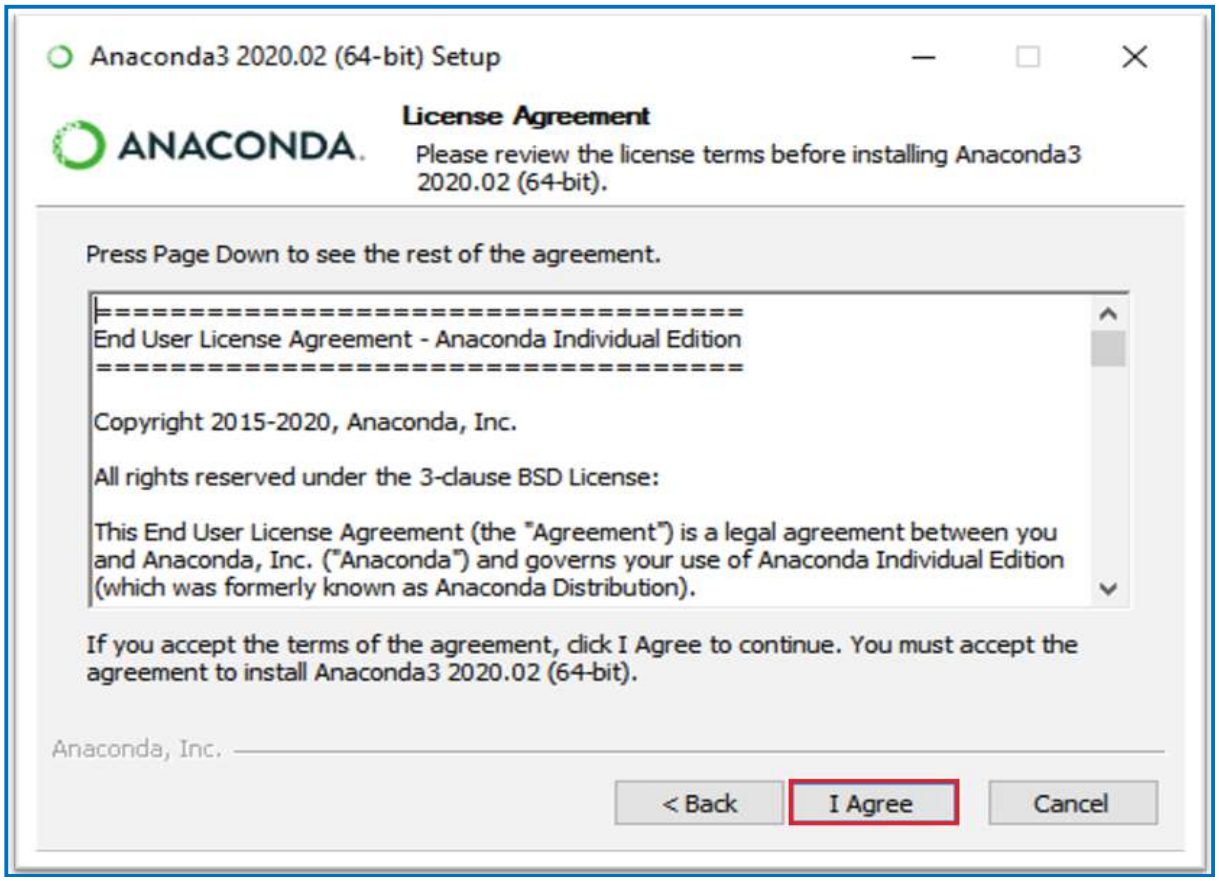


Figure 45: Anaconda Set-up (2 of 21)

5. Click Just Me and click next

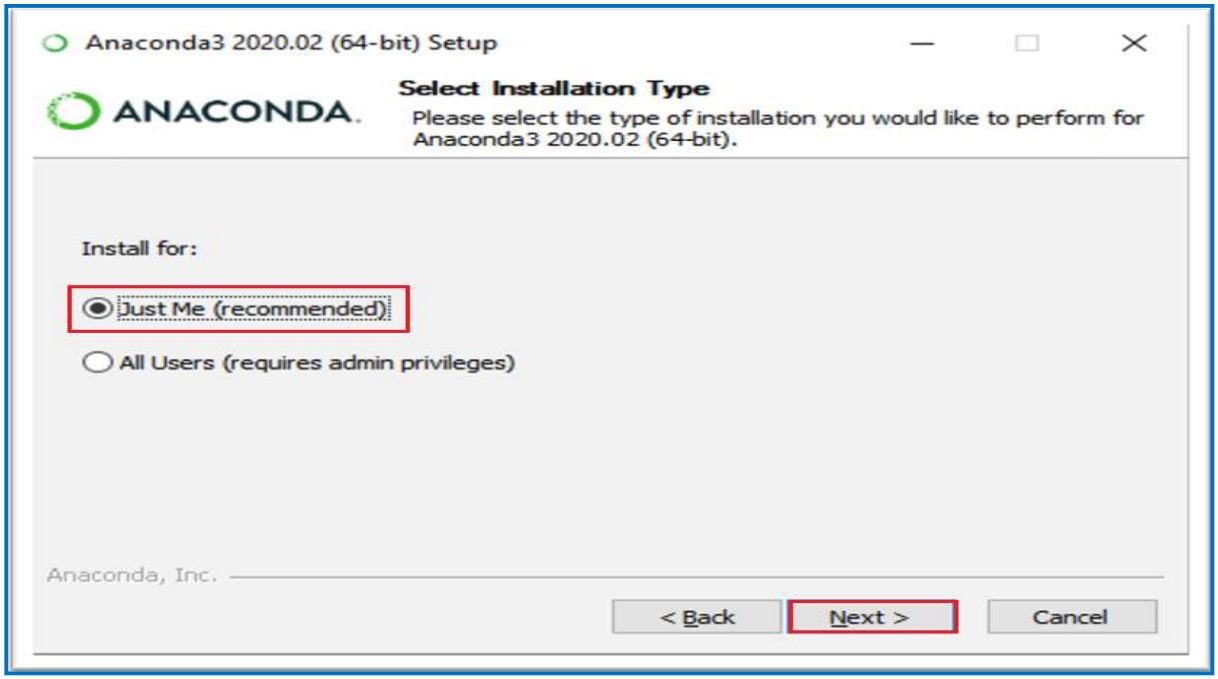


Figure 46: Anaconda Set-up (3 of 21)

6. Leave default destination folder and click next

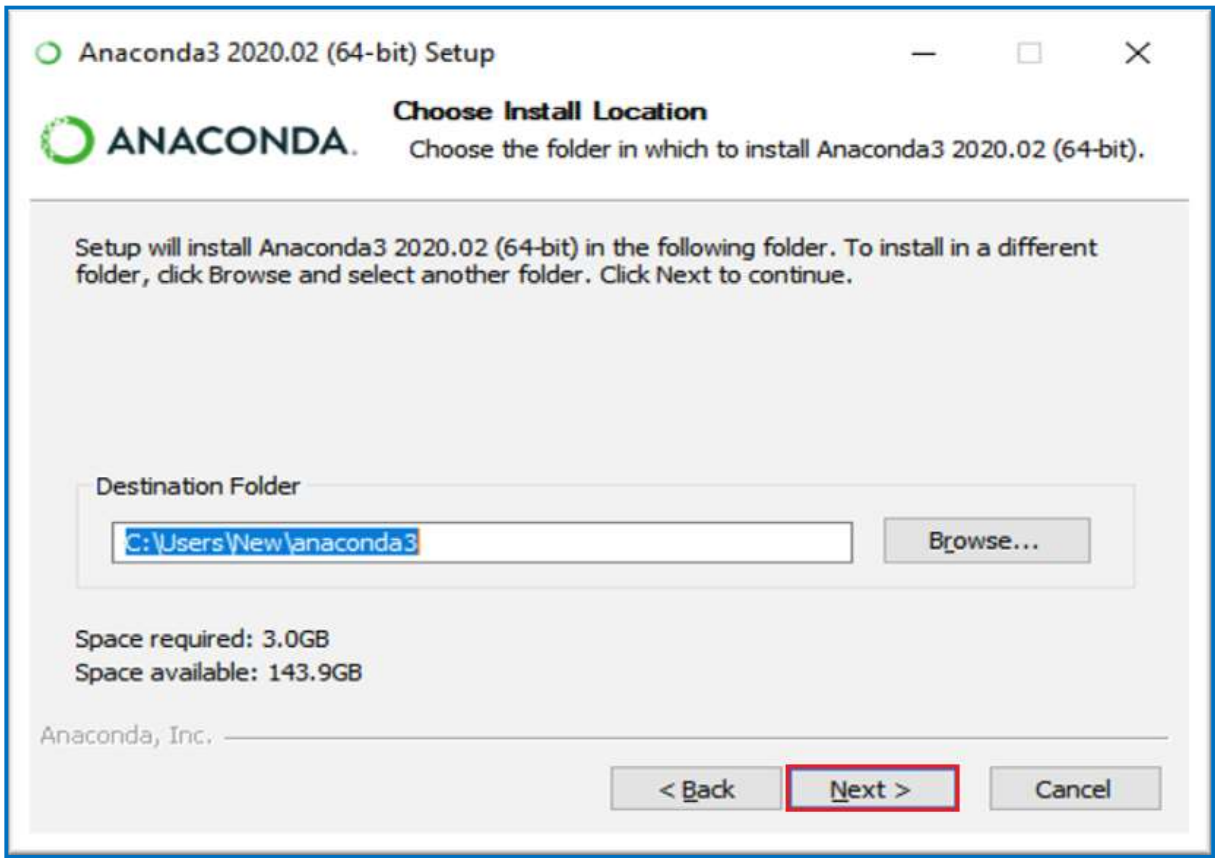


Figure 47: Anaconda Set-up (4 of 21)

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

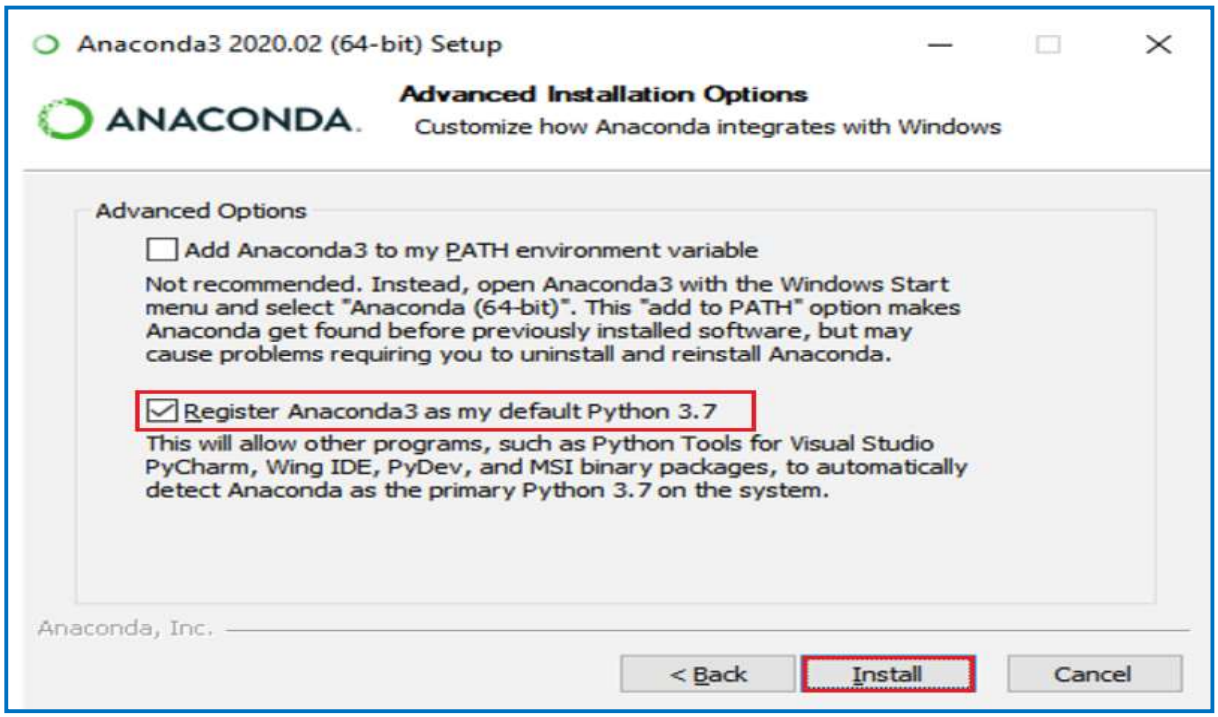


Figure 48: Anaconda Set-up (5 of 21)

8. Once completed click next

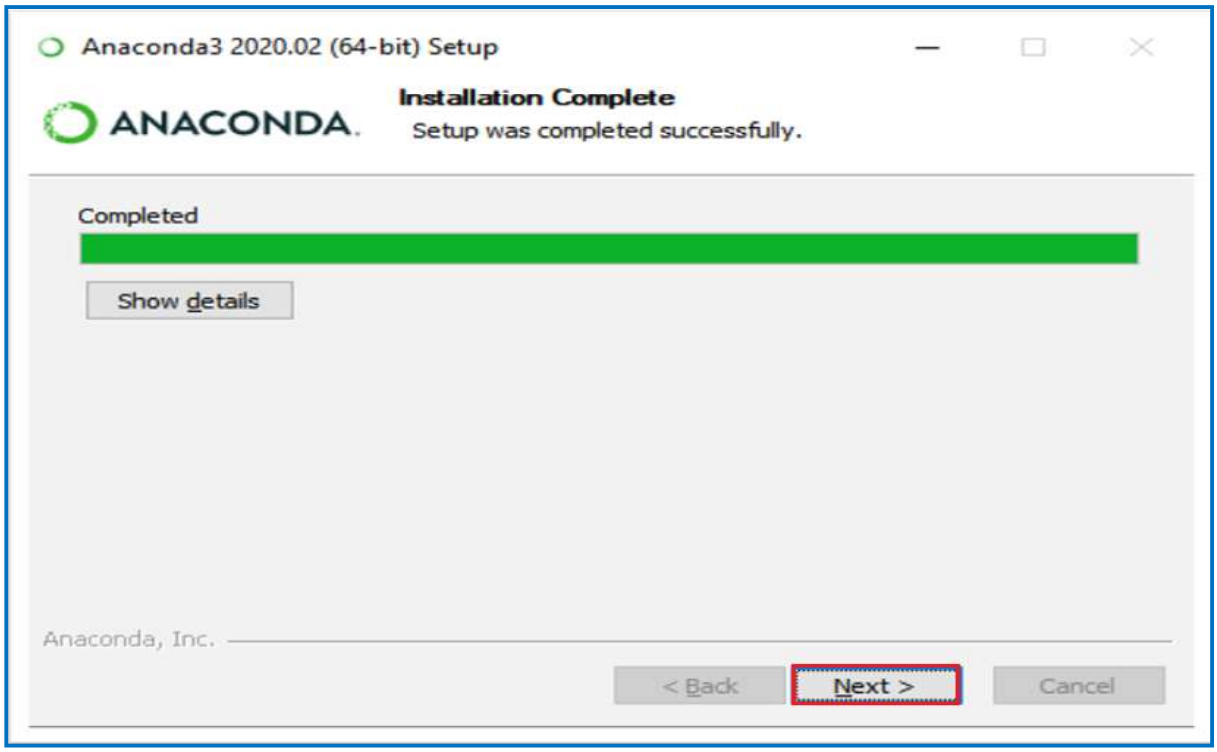


Figure 49: Anaconda Set-up (6 of 21)

9. And next again

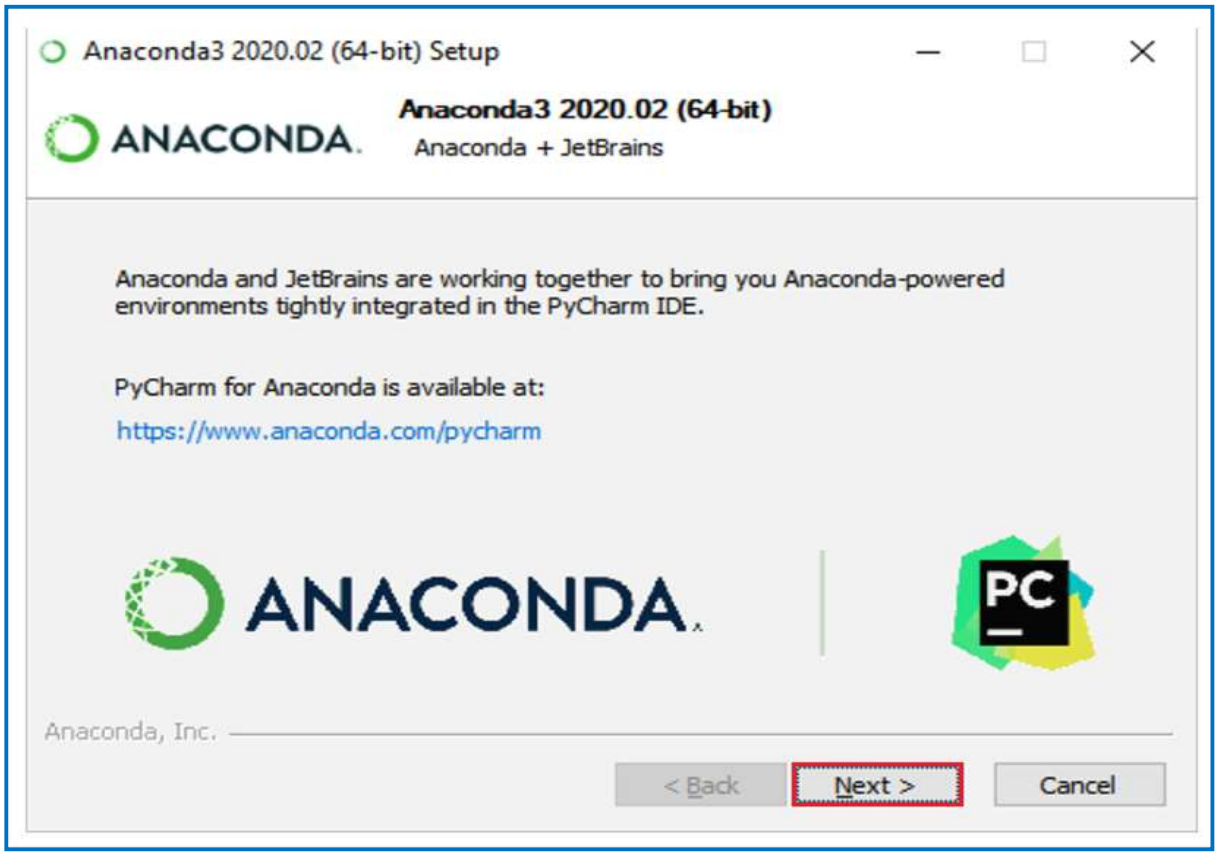


Figure 50: Anaconda Set-up (7 of 21)

10. And finally click finish. Anaconda is now installed

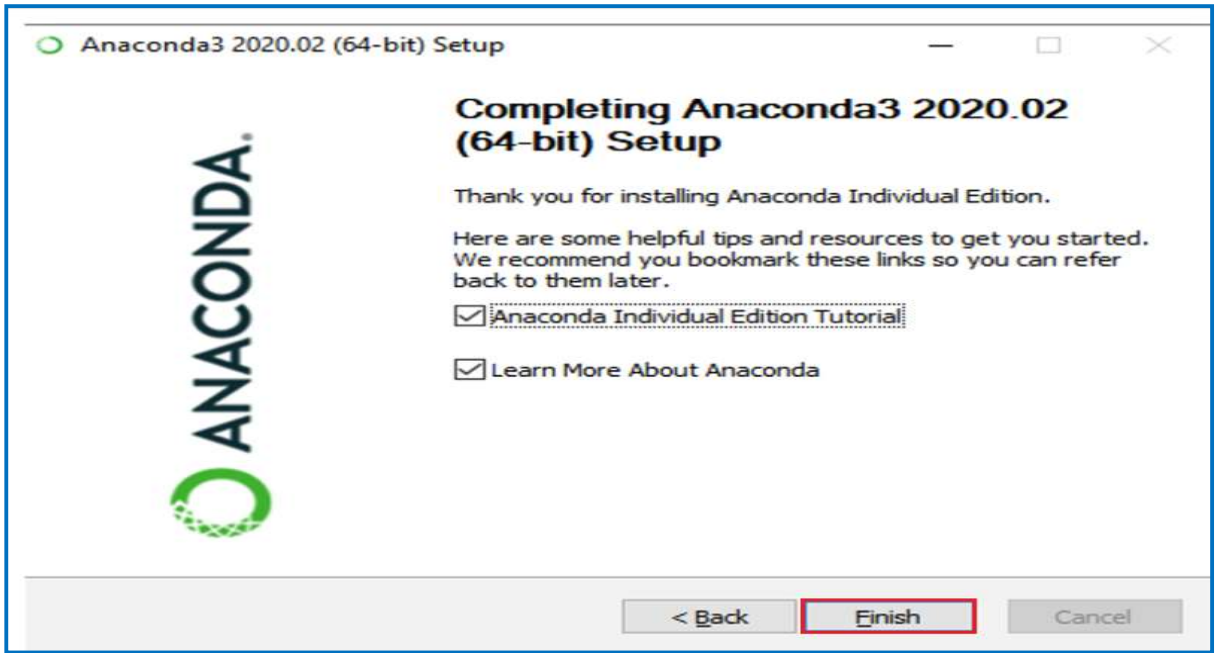


Figure 51: Anaconda Set-up (8 of 21)

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

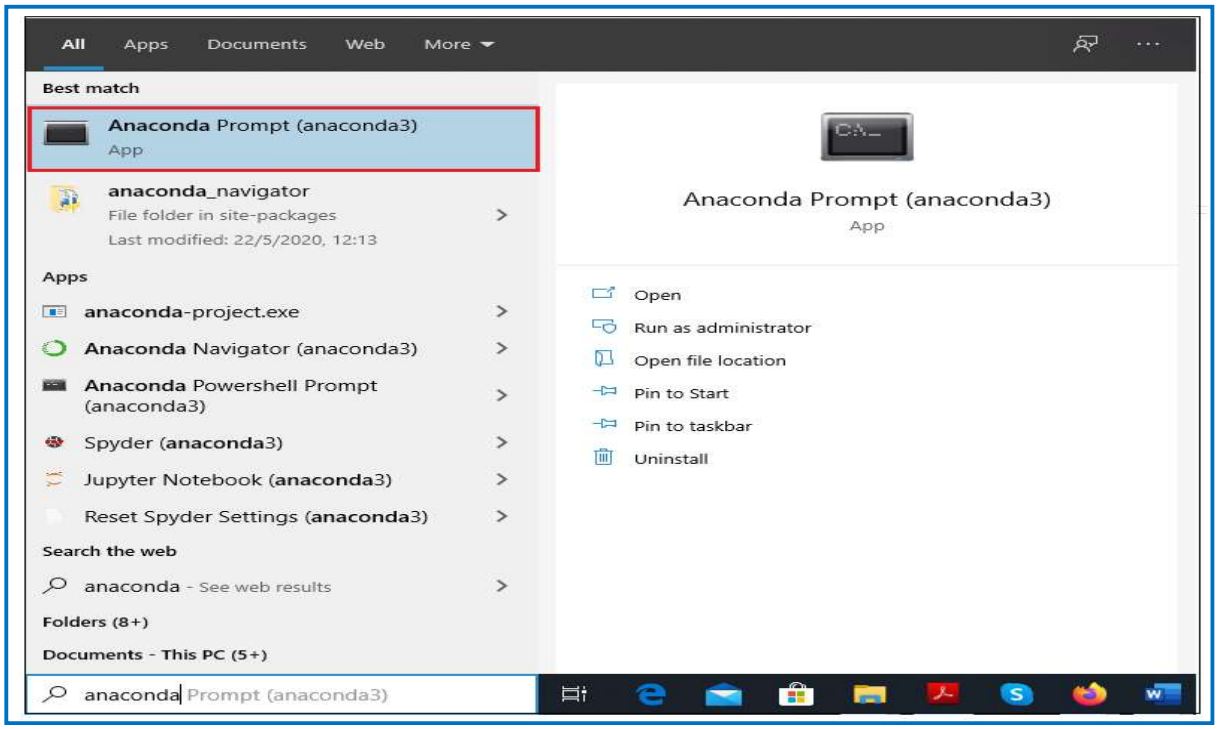


Figure 52: Anaconda Set-up (9 of 21)

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.

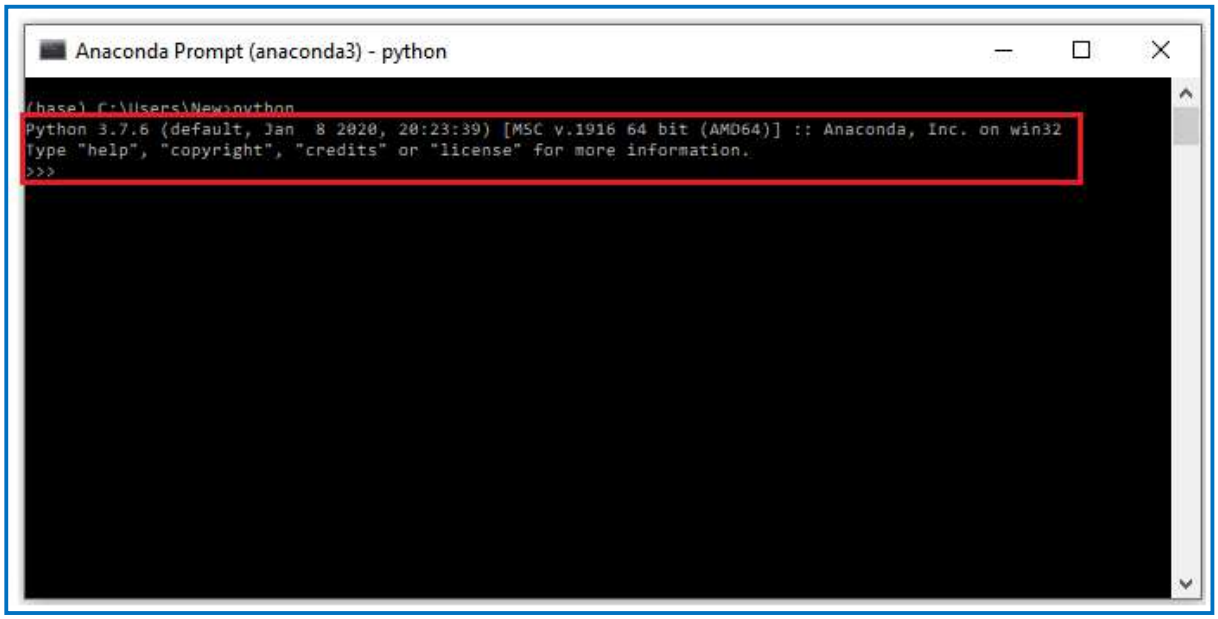


Figure 53: Anaconda Set-up (10 of 21)

13. To exit type exit()

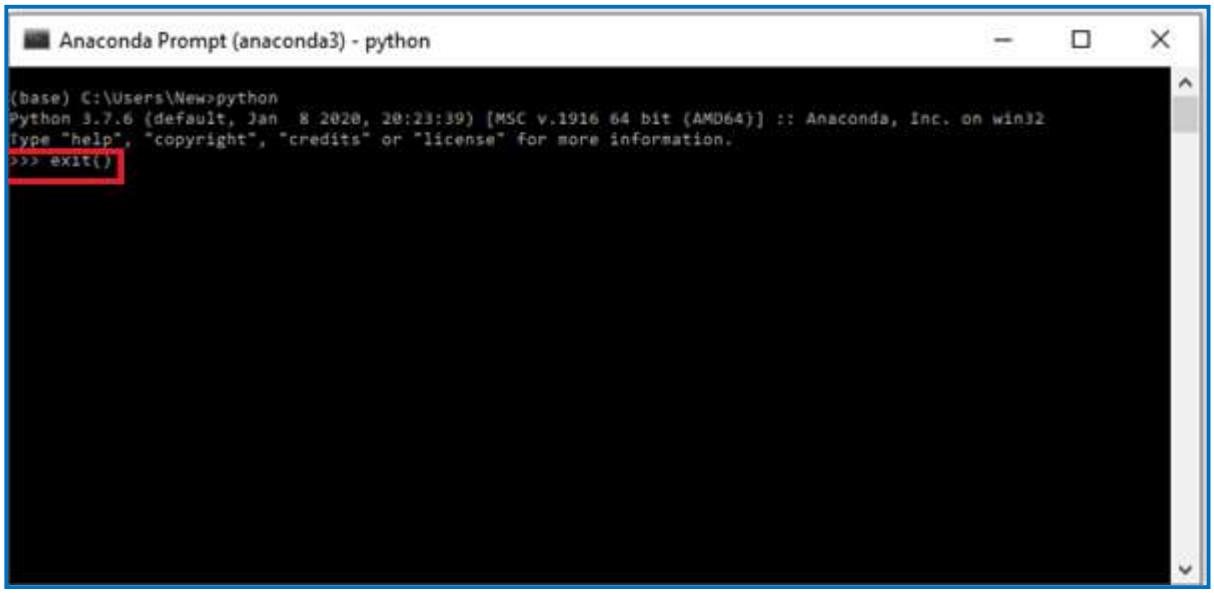


Figure 54: Anaconda Set-up (11 of 21)

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

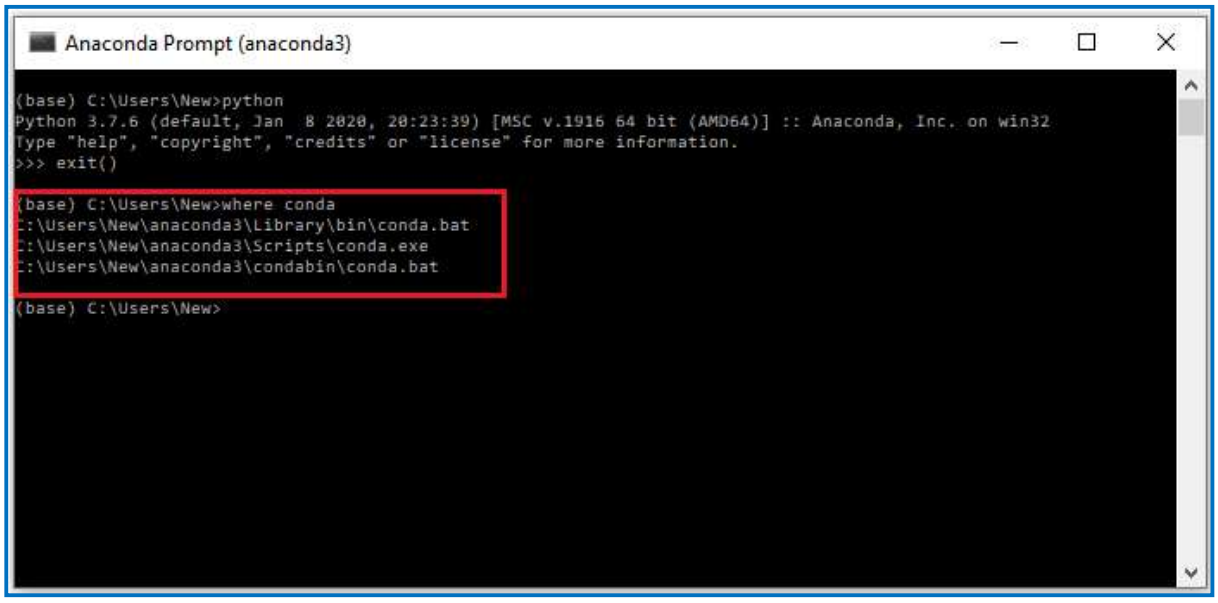


Figure 55: Anaconda Set-up (12 of 21)

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

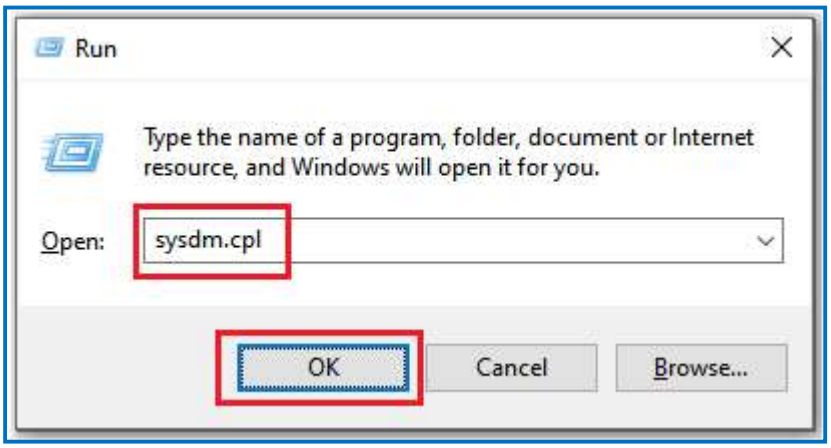


Figure 56: Anaconda Set-up (13 of 21)

16. Click on Advanced

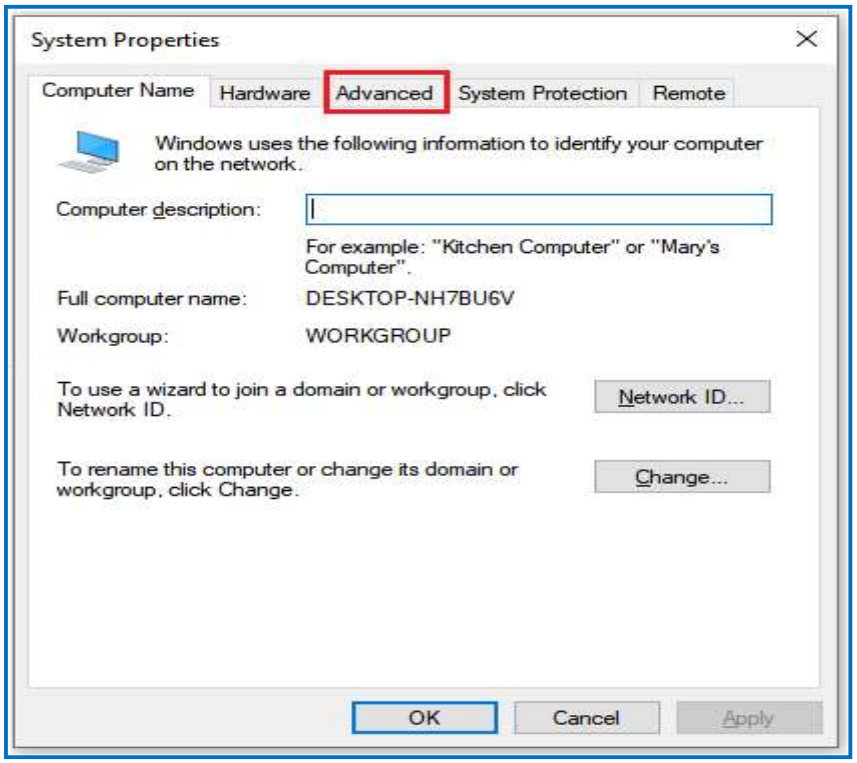


Figure 57: Anaconda Set-up (14 of 21)

17. Then click Environmental Variables

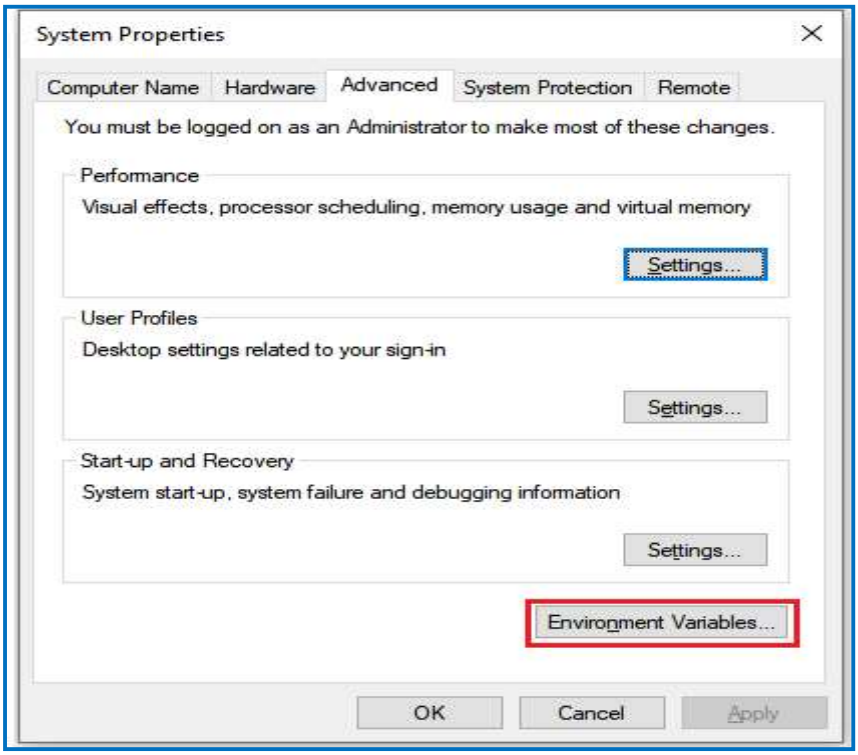


Figure 58: Anaconda Set-up (15 of 21)

18. Click on Path and then click Edit

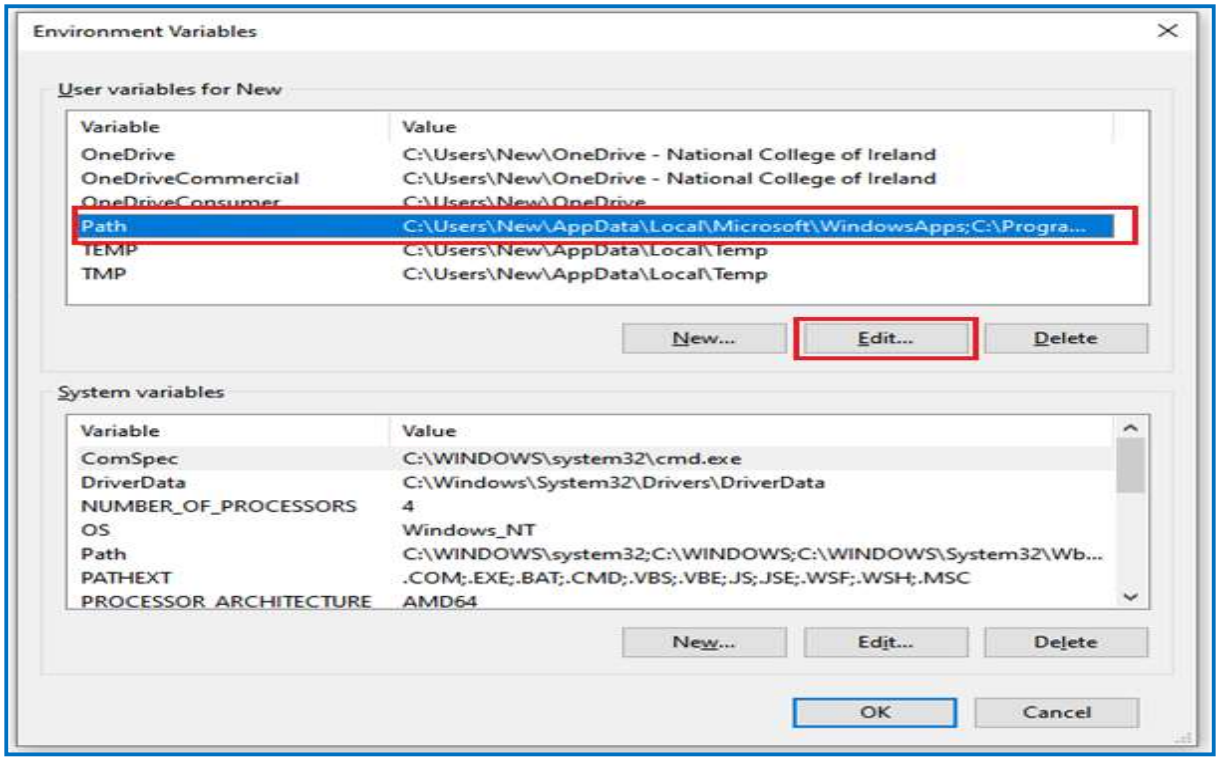


Figure 59: Anaconda Set-up (16 of 21)

19. Click New

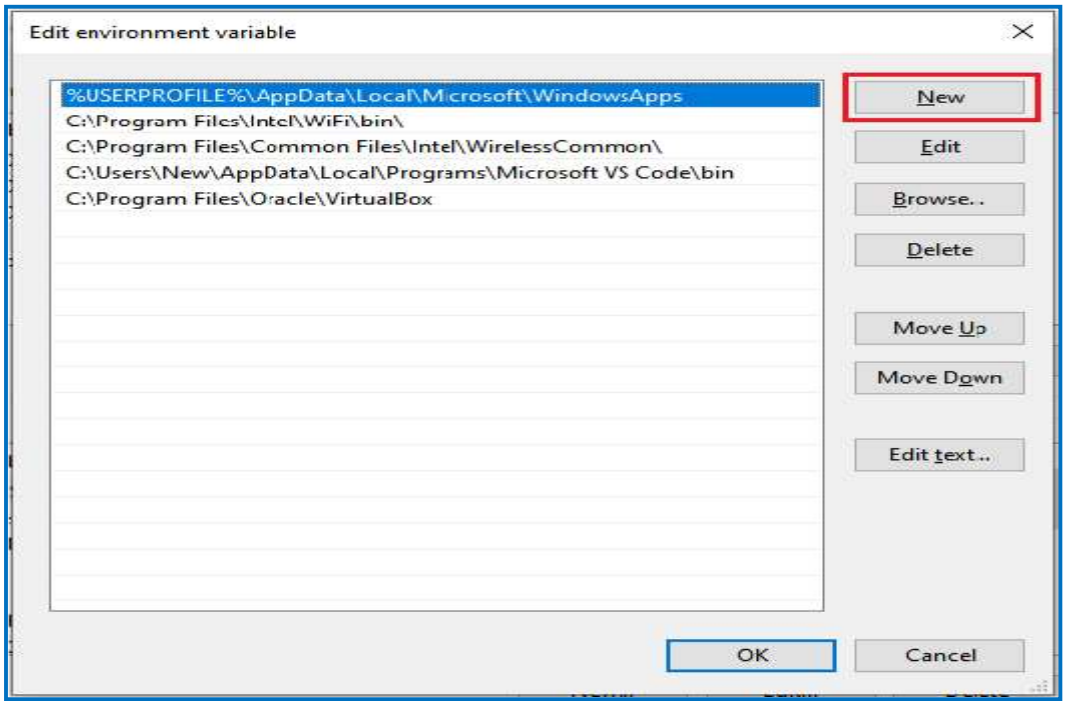


Figure 60: Anaconda Set-up (17 of 21)

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.

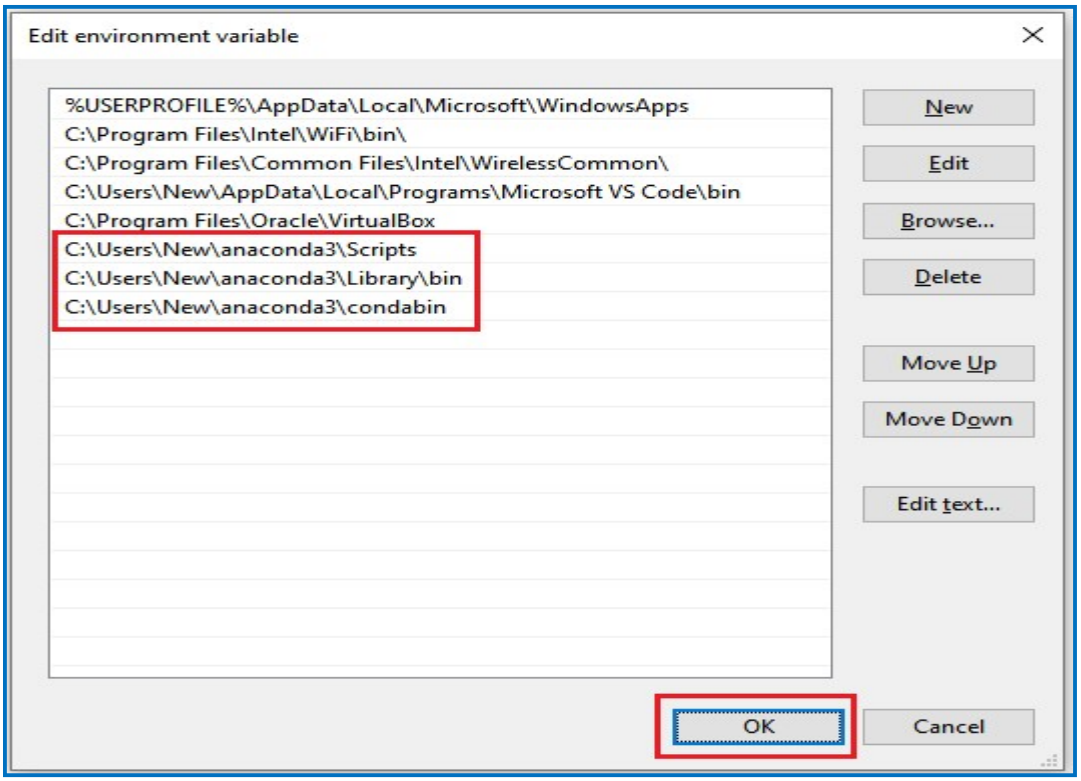


Figure 61: Anaconda Set-up (18 of 21)

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

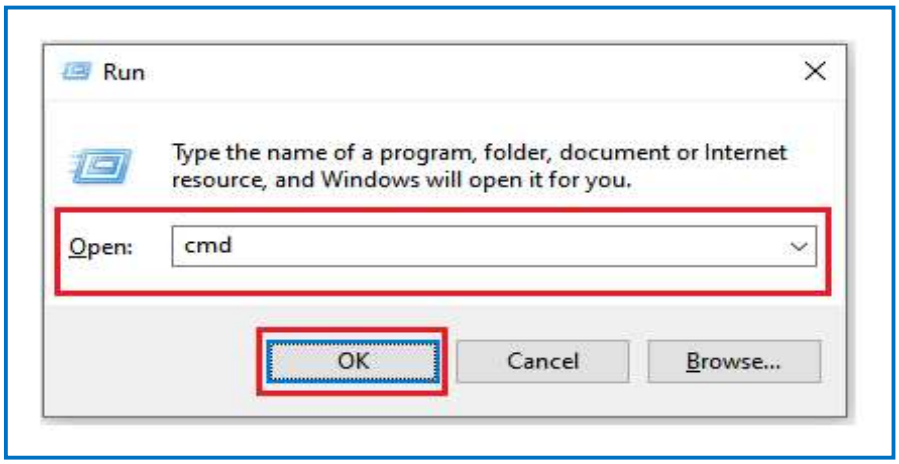


Figure 62: Anaconda Set-up (19 of 21)

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

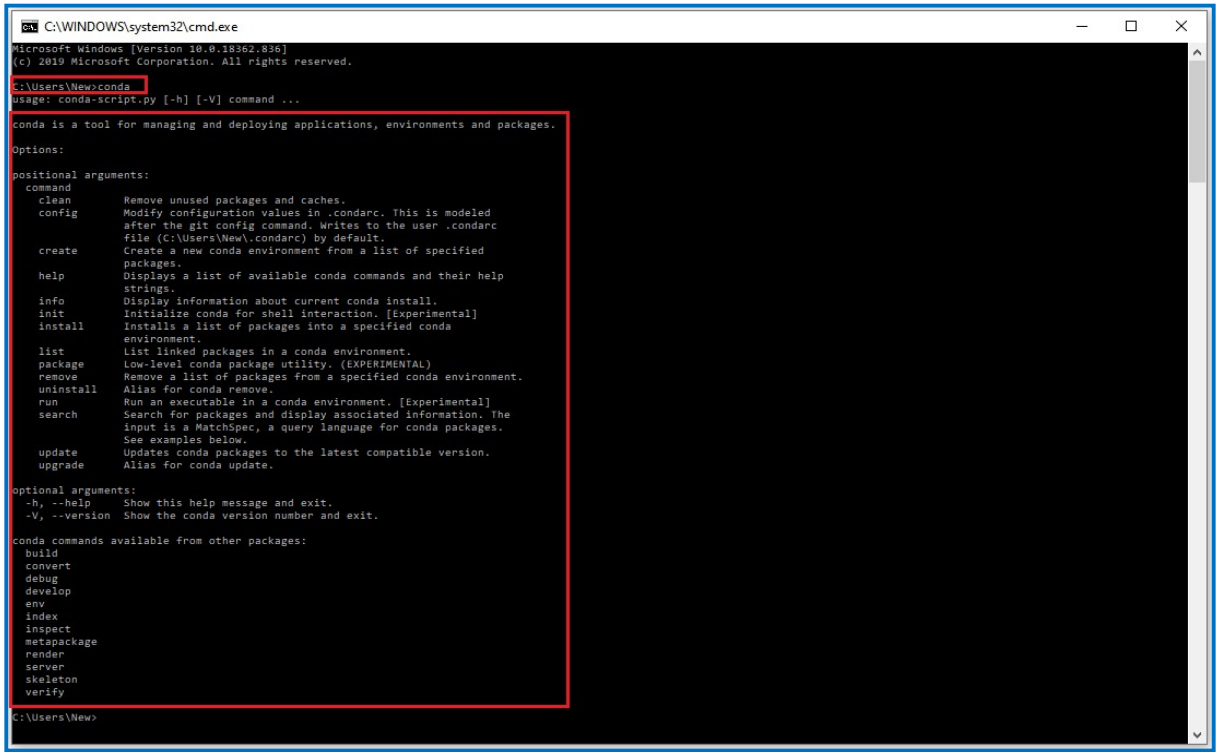


Figure 63: Anaconda Set-up (20 of 21)

23. To check which version, enter conda –version

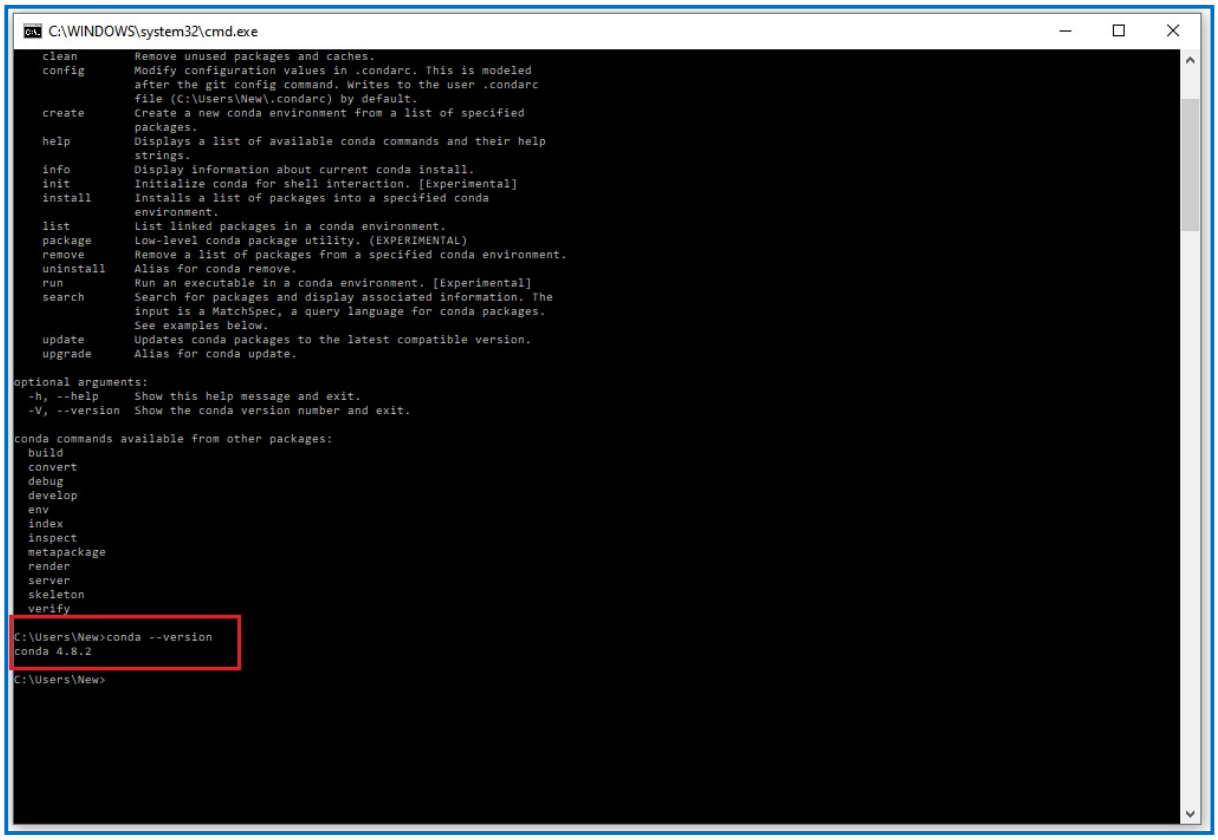


Figure 64: Anaconda Set-up (21 of 21)

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 website[7]

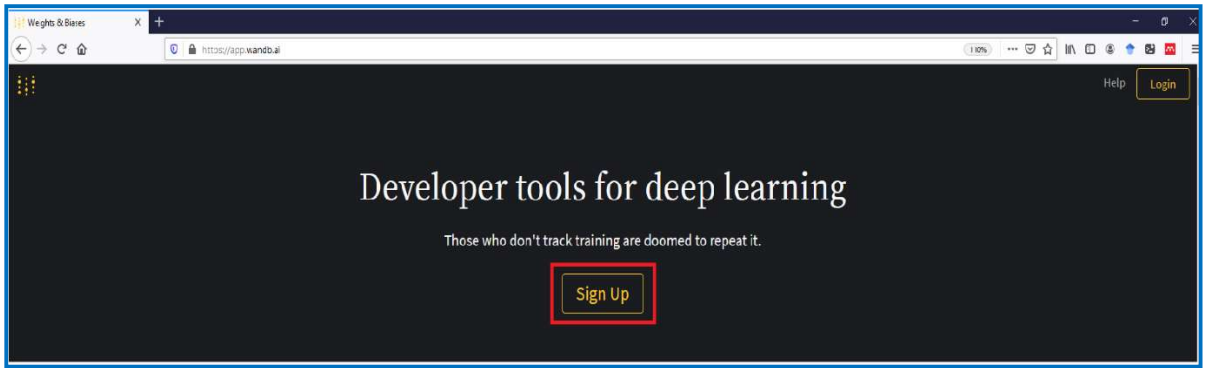


Figure 65: Weights and Biases Set-up

[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 website[8].

1. Click here

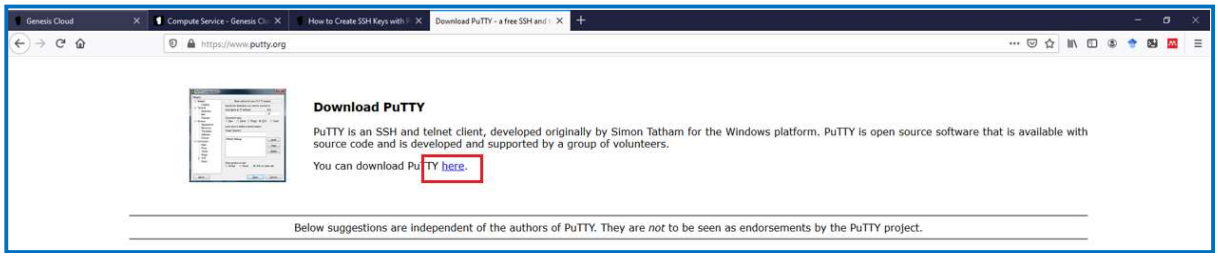


Figure 66: PuTTY Set-up (1 of 7)

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

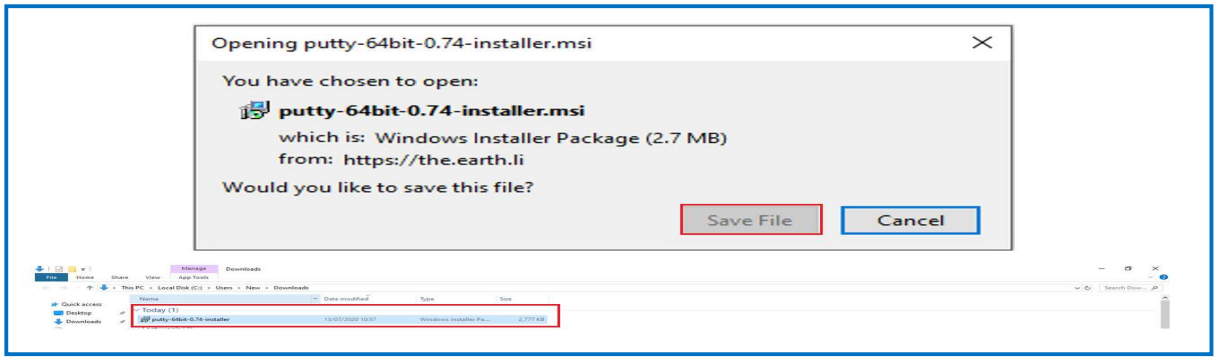


Figure 68: PuTTY Set-up (3 of 7)

[8] https://www.putty.org/

4. Click Next

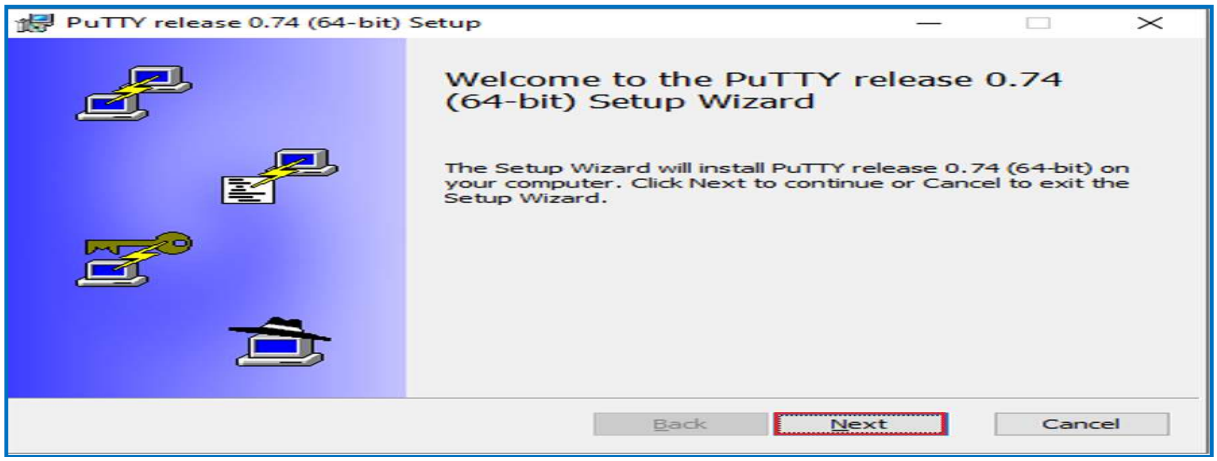


Figure 69: PuTTY Set-up (4 of 7)

5. And click Next again

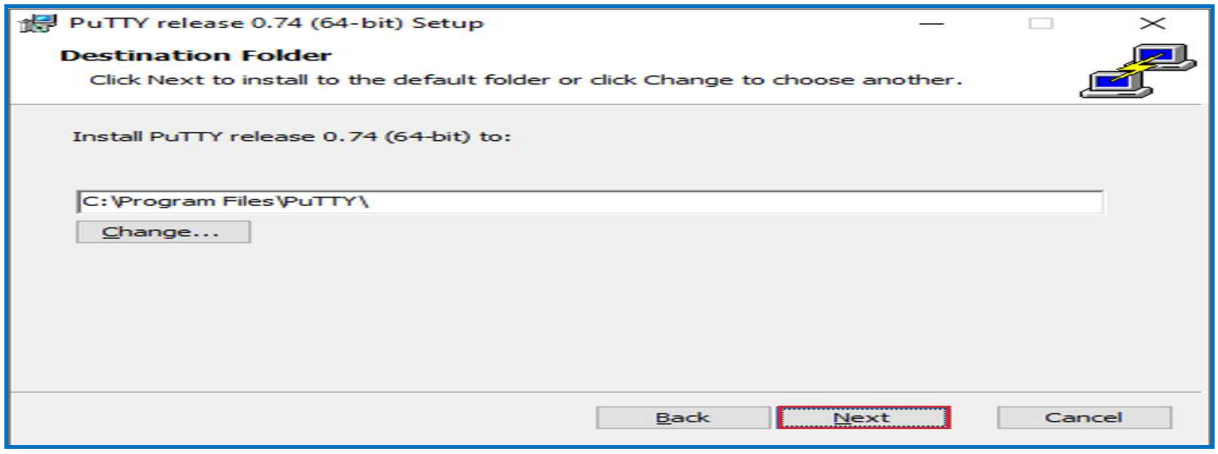


Figure 70: PuTTY Set-up (5 of 7)

6. Click Install

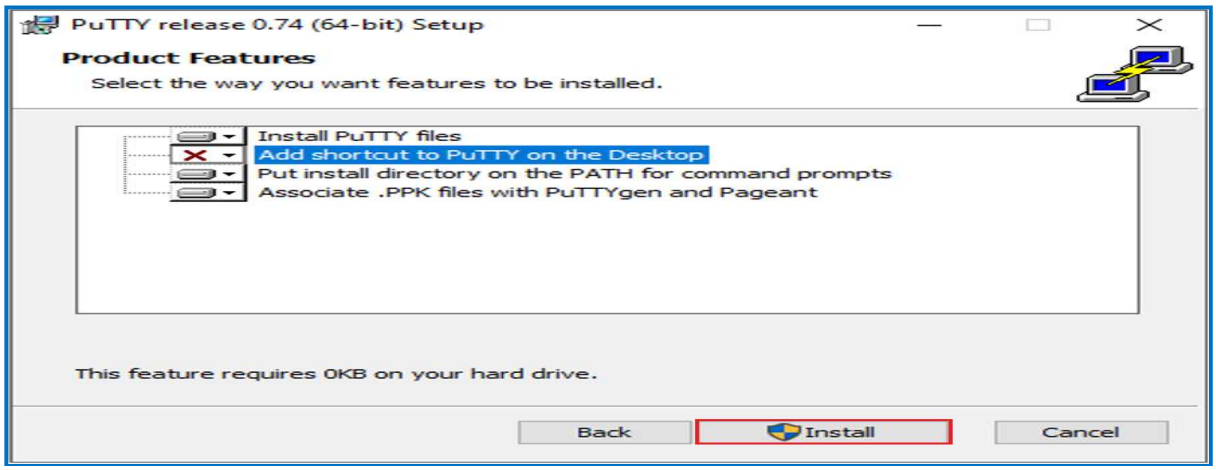


Figure 71: PuTTY Set-up (6 of 7)

7. Finally, click Finish. Putty is installed successfully

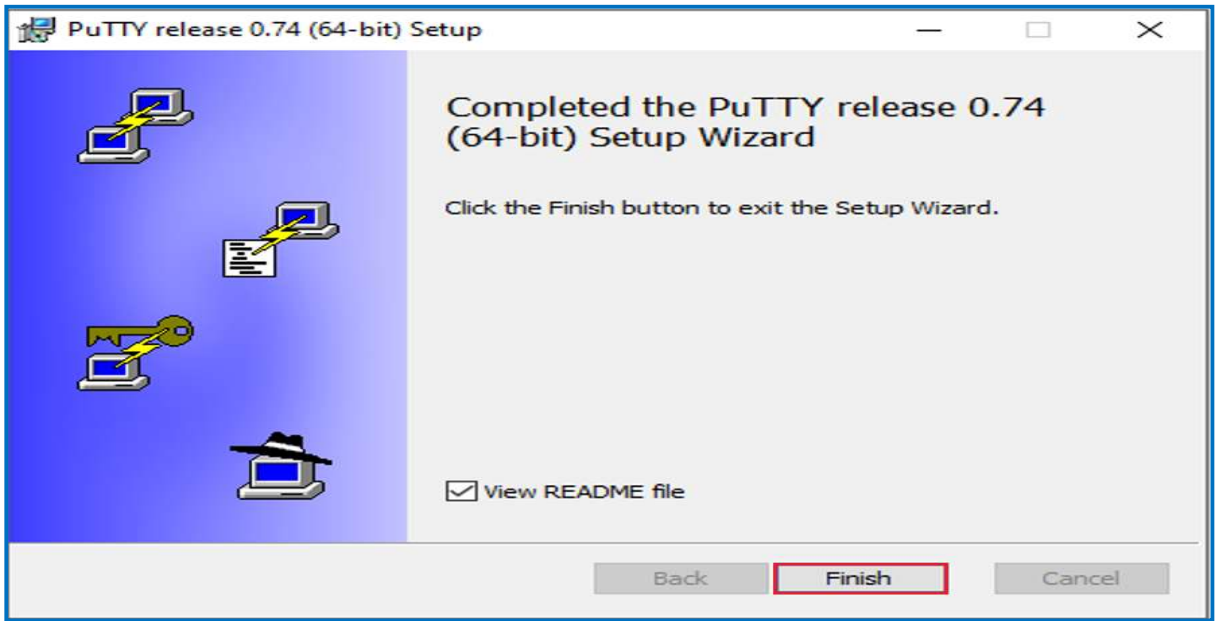


Figure 72: PuTTY Set-up (7 of 7)

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 page[9].

1. Click Add to Firefox

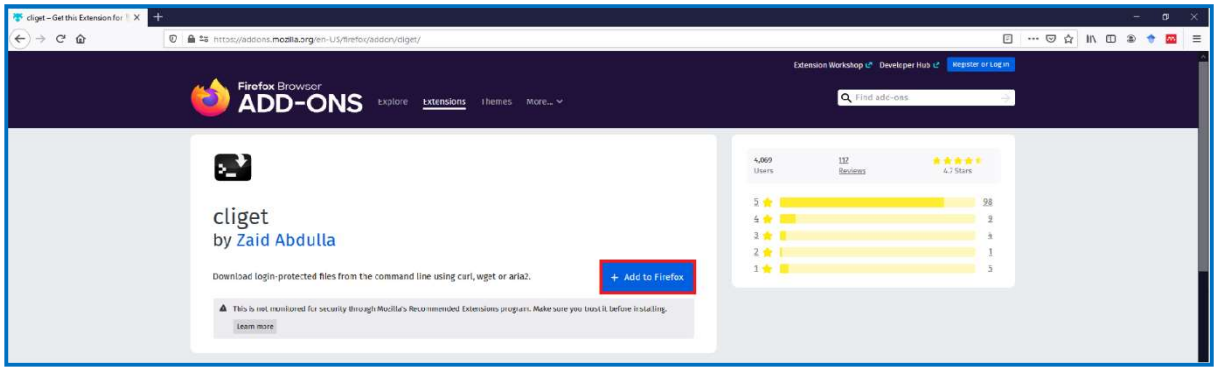


Figure 73: Cliget Add-On

[9] https://addons.mozilla.org/en-US/firefox/addon/cliget/

2. Click Add

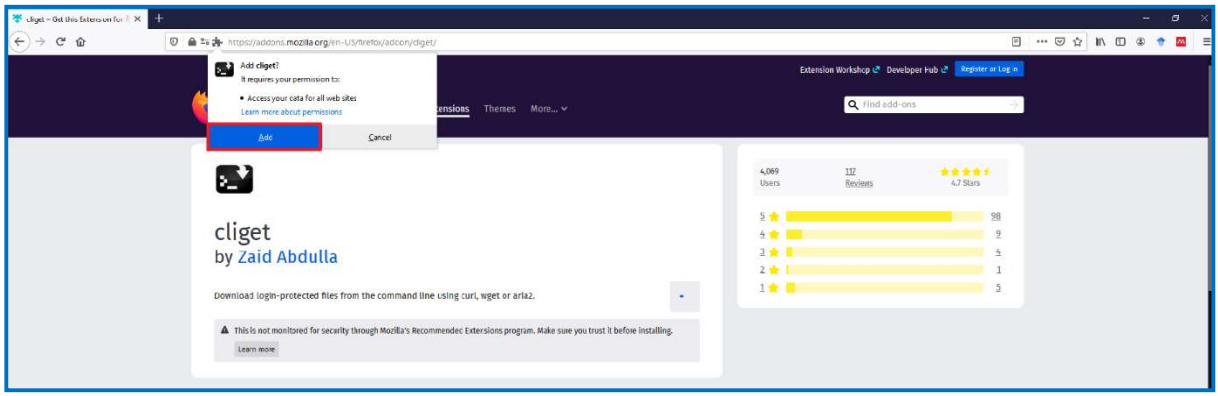


Figure 74: Cliget Download

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

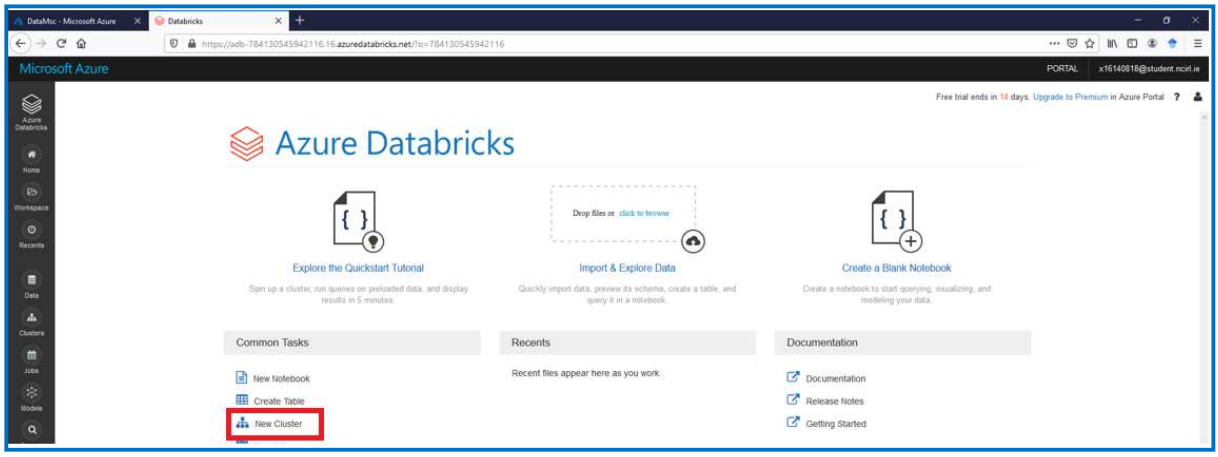


Figure 75: Cluster Set-up (1 of 3)

2. Update below information and click Create Cluster

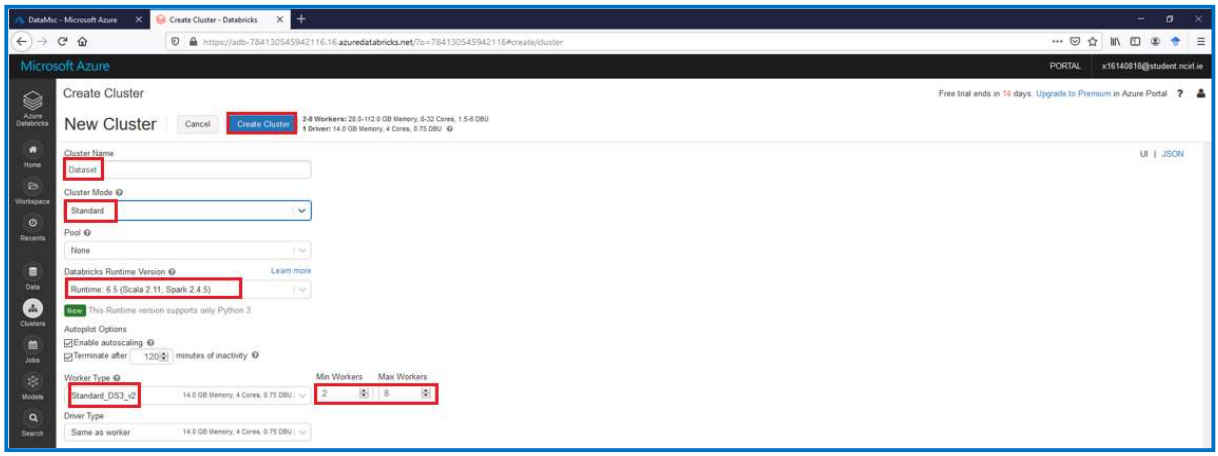


Figure 76: Cluster Set-up (2 of 3)

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

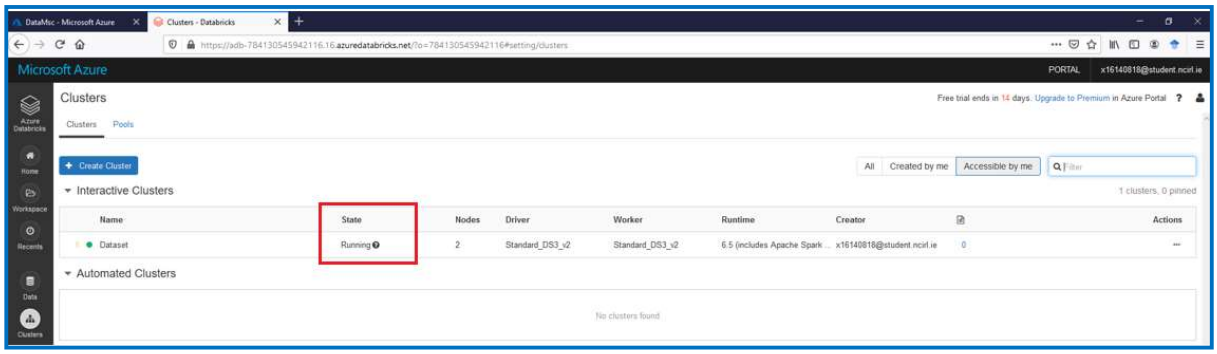


Figure 77: Cluster Set-up (3 of 3)

4. Return to Homepage and launch Jupyter Notebook

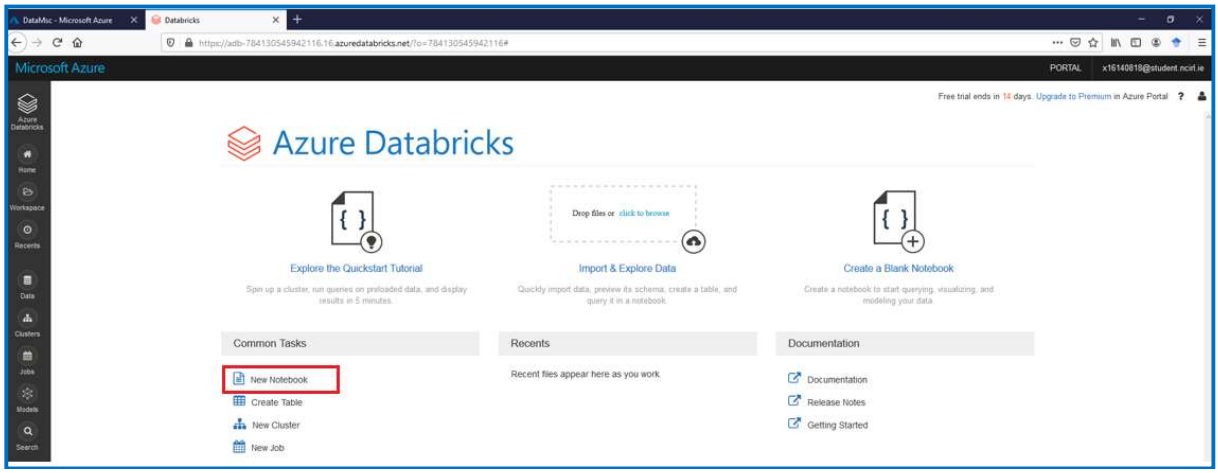


Figure 78: Jupyter Notebook

The following PySpark code was run to create the two datasets

1. Import libraries

```
Cmd 1

from pyspark.sql.functions import *
from pyspark.sql.types import *
from graphframes import *
from pyspark.mllib.stat import *
import pandas as pd
from pyspark.ml.classification import LogisticRegression
from pyspark.ml.feature import VectorAssembler
from pyspark.ml.feature import StandardScaler
from pandas.plotting import scatter_matrix
import seaborn as sns
import matplotlib.pyplot as plt
import numpy as np
import six
from pyspark.ml.evaluation import BinaryClassificationEvaluator
from pyspark.ml.classification import RandomForestClassifier
from pyspark.ml.tuning import ParamGridBuilder, CrossValidator
import mlflow
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

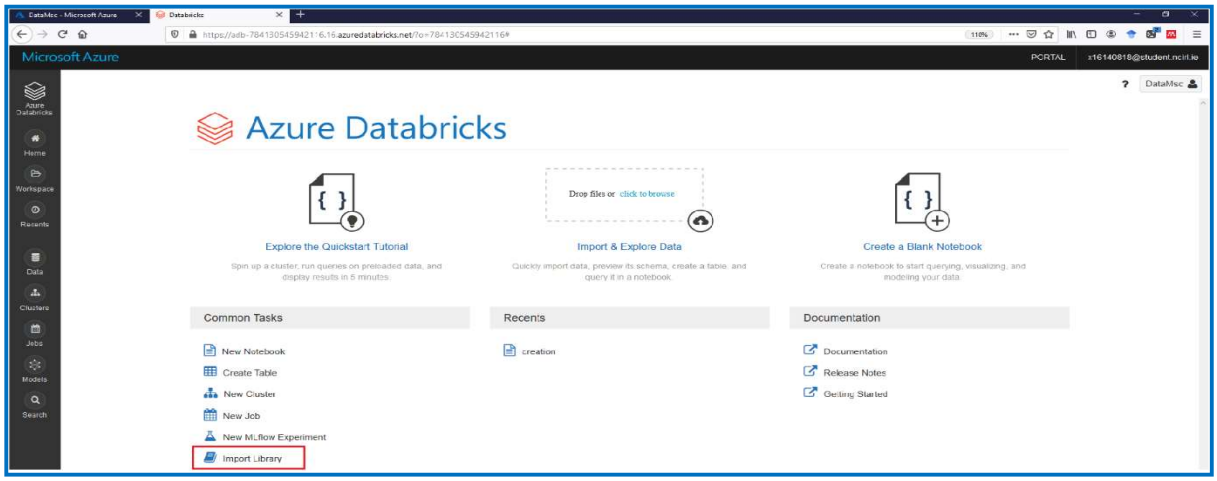


Figure 80: External Libraries (1 of 3)

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

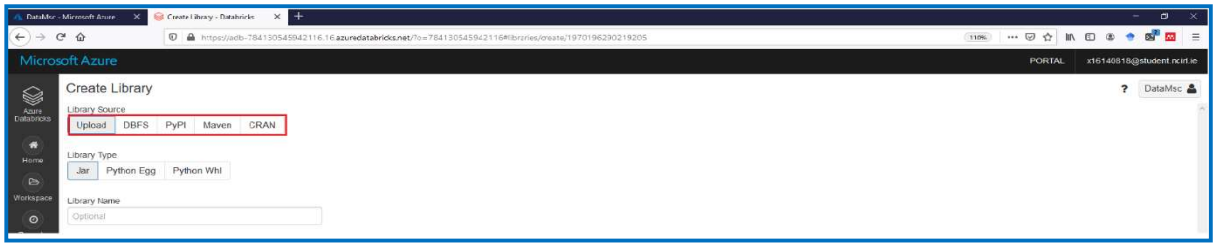


Figure 81: External Libraries (2 of 3)

4. The following external libraries were used

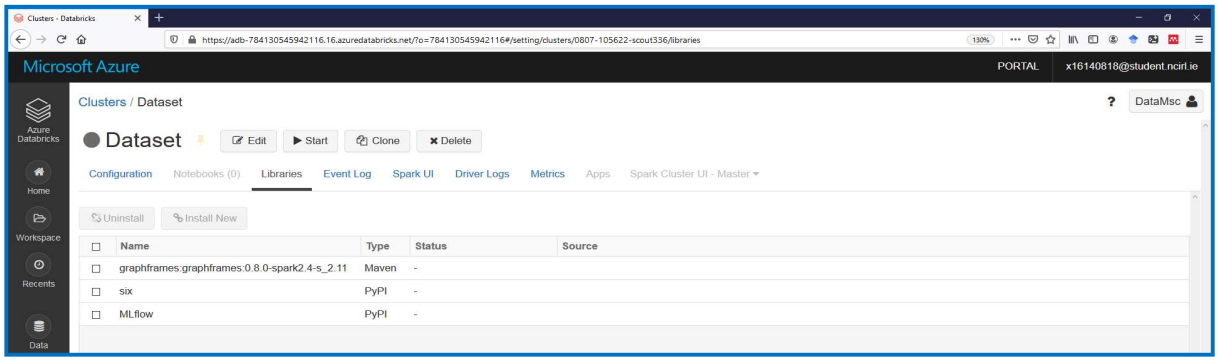


Figure 82: External Libraries (3 of 3)

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

Figure 86: Unmount Azure Files (1 of 3)

Figure 87: Unmount Azure Files (2 of 3)

Figure 88: Unmount Azure Files (3 of 3)

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

Figure 89: Read JSON Files (1 of 3)

Figure 90: Read JSON Files (2 of 3)

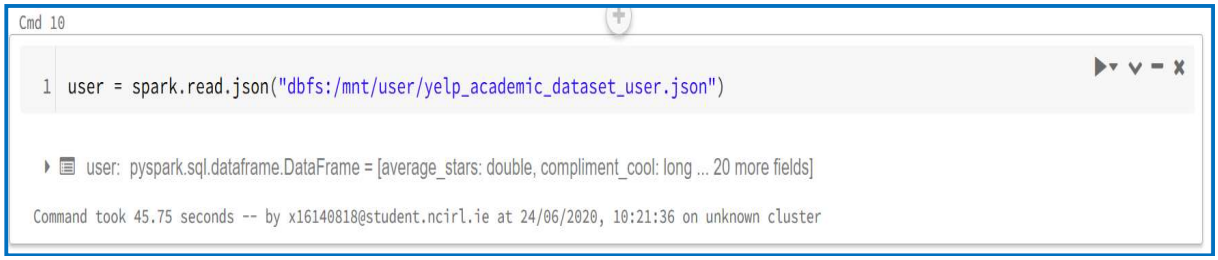


Figure 91: Read JSON Files (3 of 3)

8. Star Distribution

Figure 92: Star Distribution

9. Filter rows if they have no Text

Figure 93: Filter Text

10. Filter rows if they have no Star

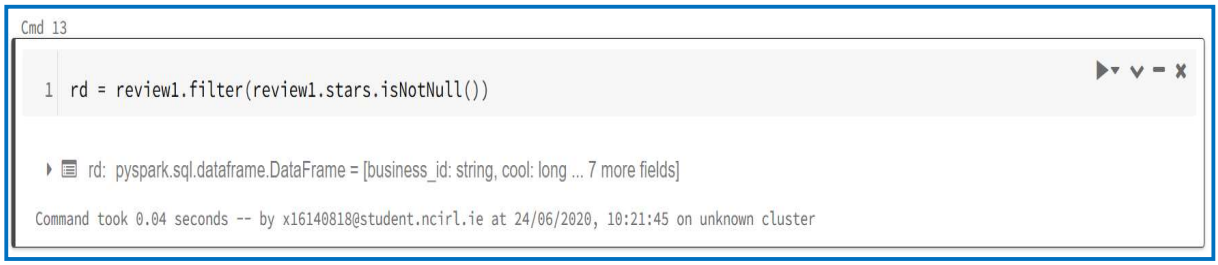


Figure 94: Filter Star

11. Display Star Count

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.

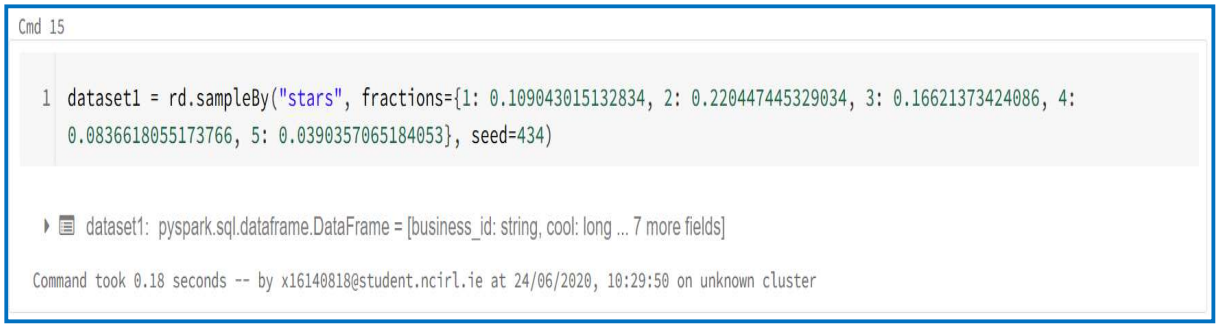


Figure 96: Stratified Sampling Attempt 1

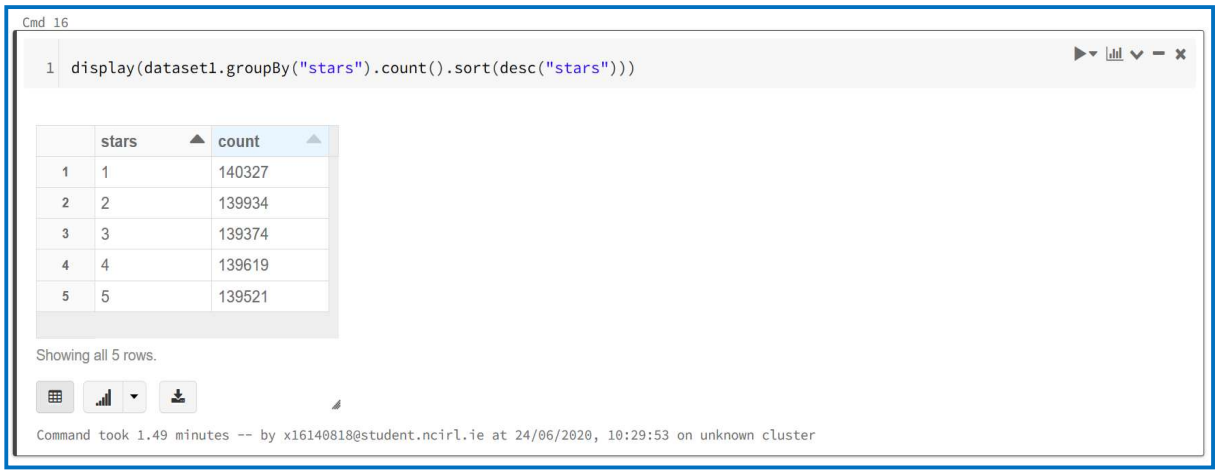


Figure 97: Stratified Sampling Attempt 1 Star Count

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

Figure 100: Stratified Sampling Attempt 3

Figure 101: Stratified Sampling Attempt 3 Star Count

15. Select Business Fields

Figure 102: Select Business Fields

16. Merge dataset1b & bd1

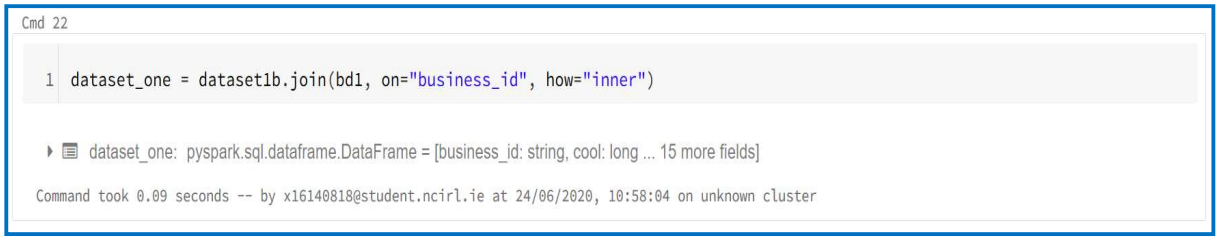


Figure 103: Merge DataFrames

17. Replace unnecessary characters, new lines and carriage returns

Figure 104: Remove Whitespaces (1 of 3)

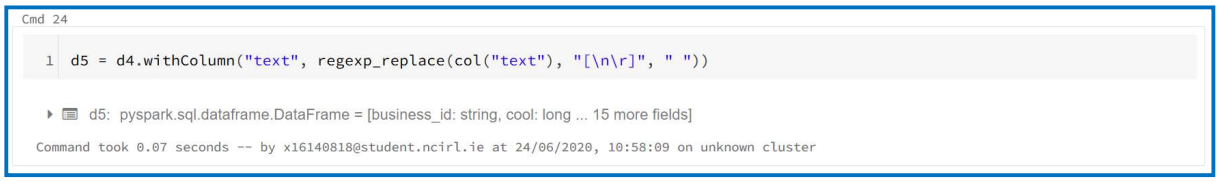


Figure 105: Remove Whitespaces (1 of 3)

18. Check Text format is correct

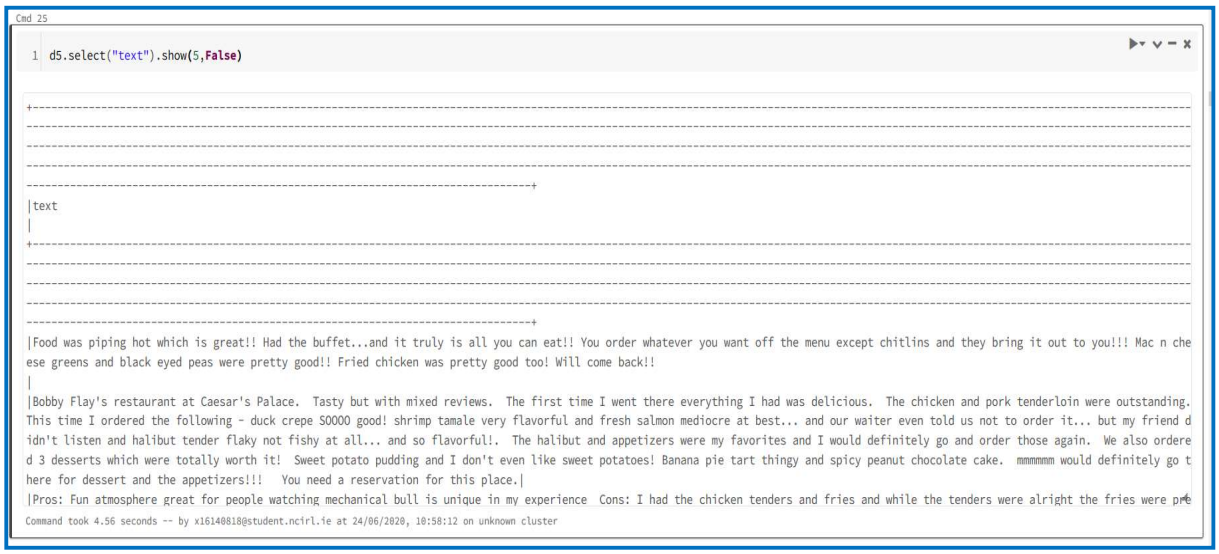


Figure 106: (1 of 3)

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

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.

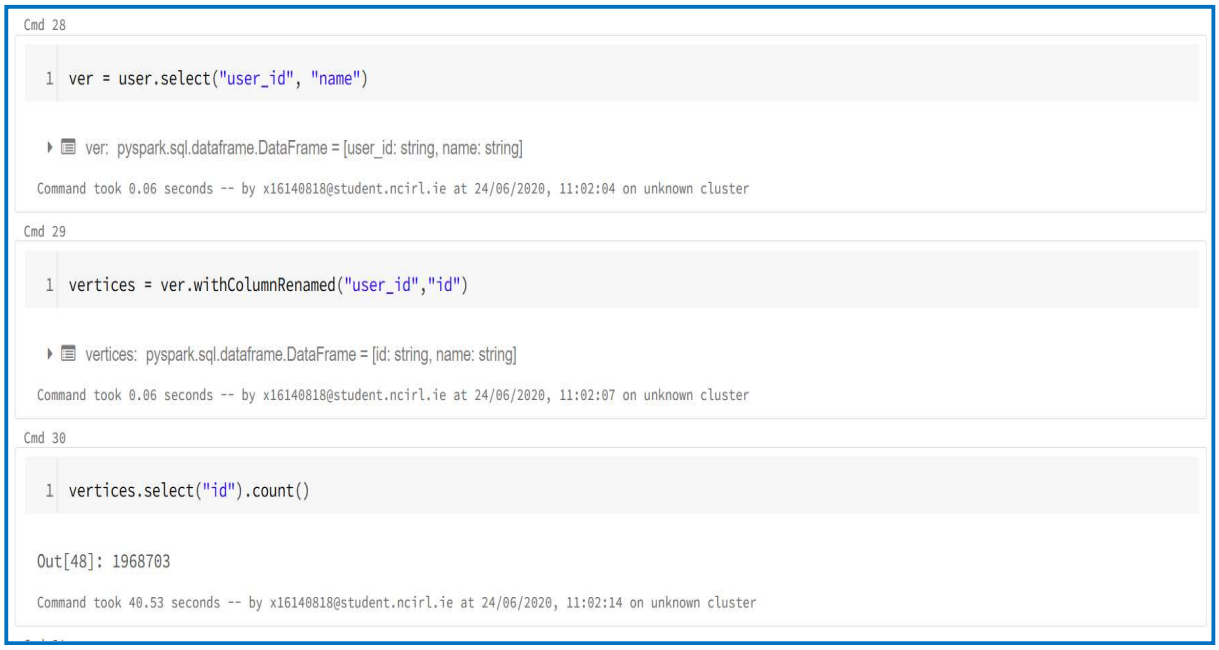


Figure 108: Vertices

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

Figure 109: Edges

Figure 110: Rename Fields

2. Create GraphFrame

Figure 111: GraphFrame

3. Display outDegree and create DataFrame of id & outDegree

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.

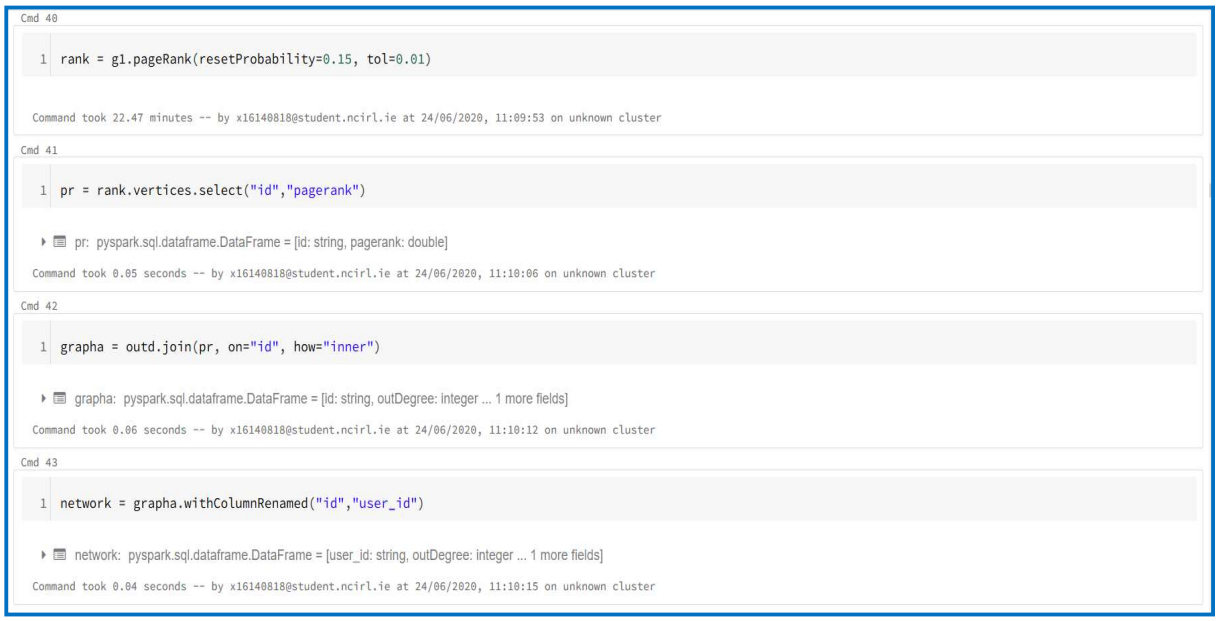


Figure 113: PageRank

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

Figure 114: Social Network DataFrame

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

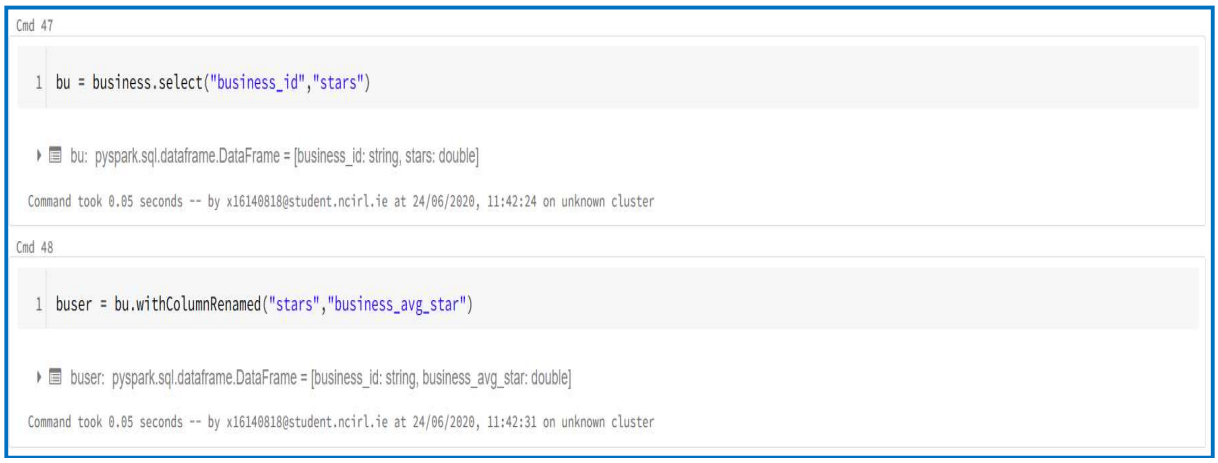


Figure 115: Business Average Star Rating

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

Figure 116: Select Review Fields

Figure 117: Remove Characters

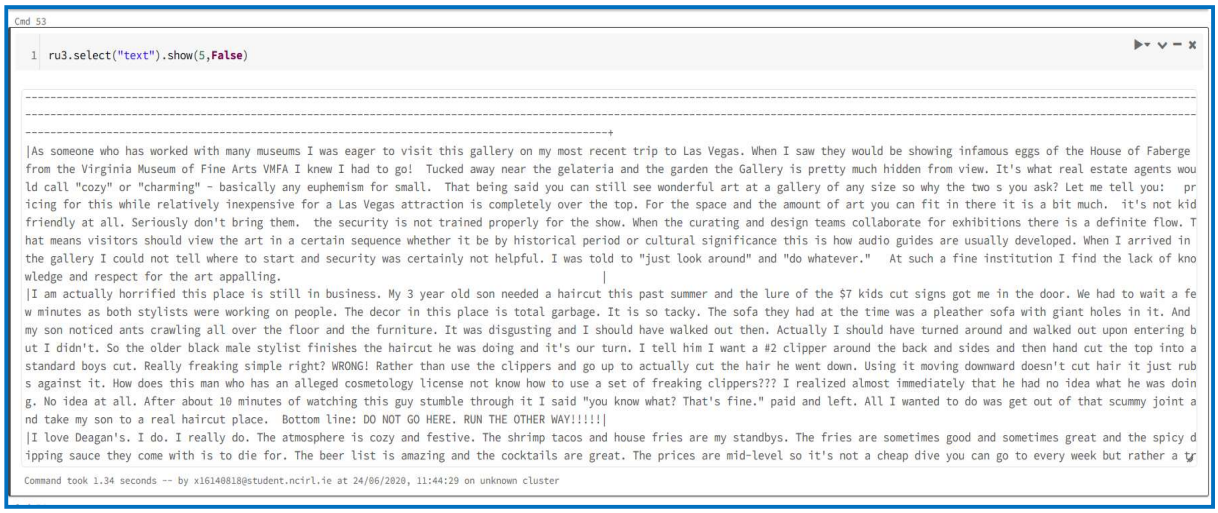


Figure 118: Check Text

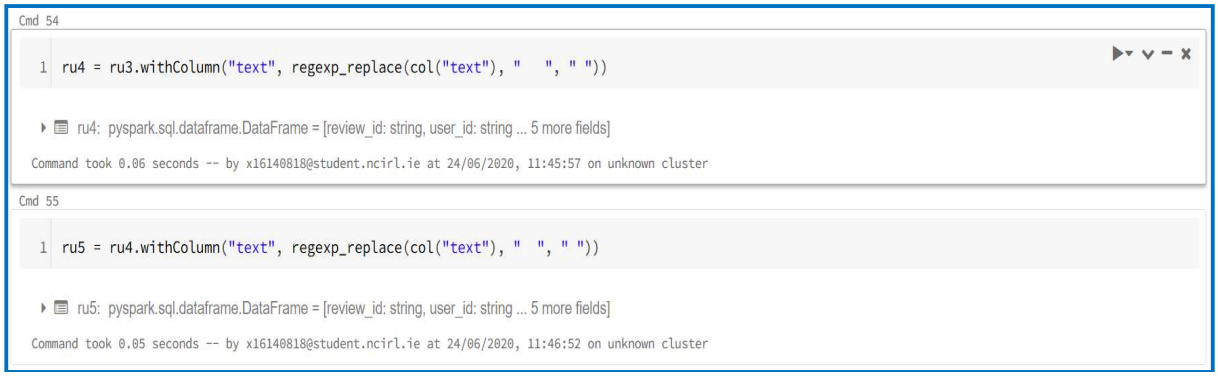


Figure 119: Remove Whitespaces

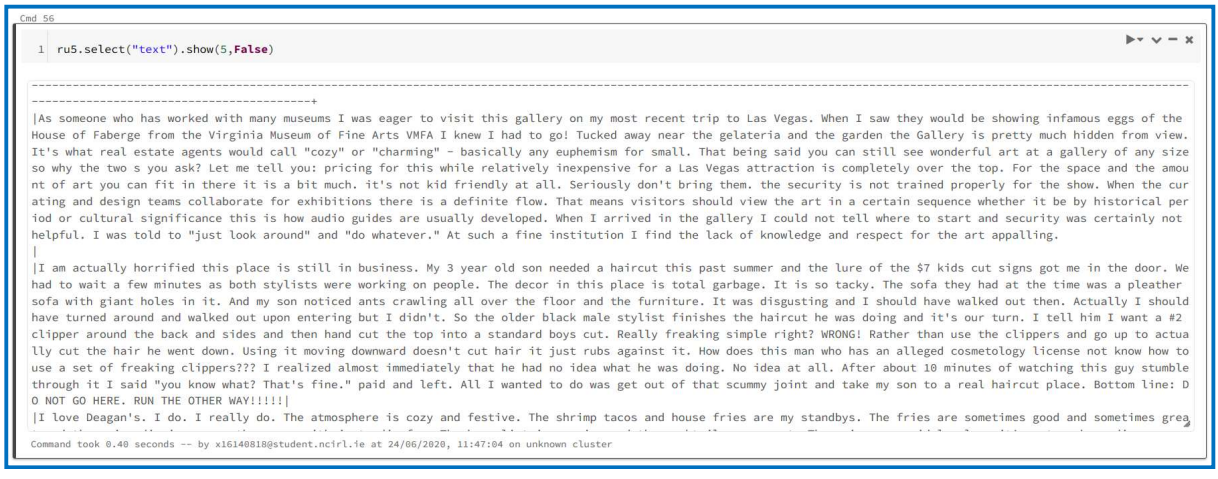


Figure 120: Recheck Text

8. Calculate Word Count, Sentence Count and Price Count. Check to see if Price Count is correct

Figure 121: Word, Sentence, Price Count

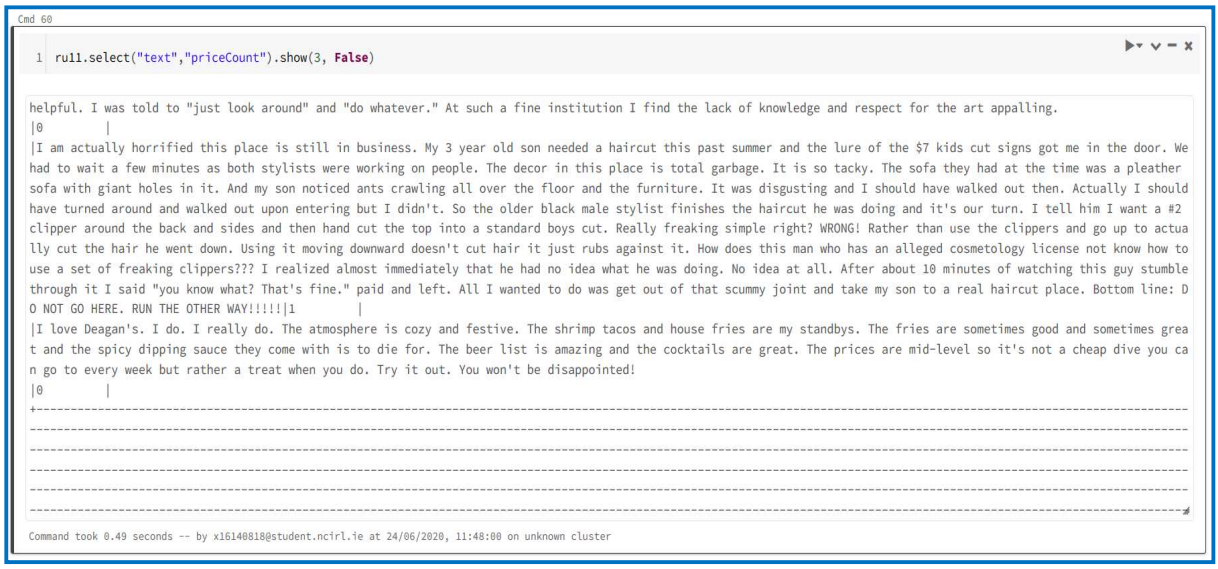


Figure 122: Check Price Count

9. Merge ru11 and buser DataFrames.

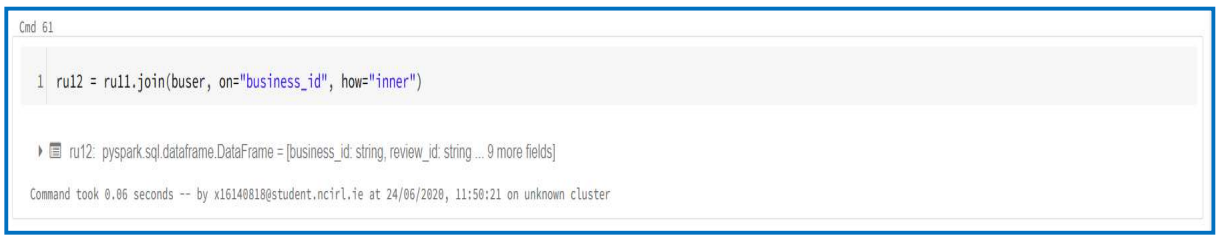


Figure 123: Merge DataFrames

10. Calculate average review rating.

Figure 124: Review Average Count

11. Merge ru12 with average review rating

Figure 125: Merge DataFrames

12. Merge social data with social network DataFrames

Figure 126: Merge DataFrame

13. Remove timestamp from date field

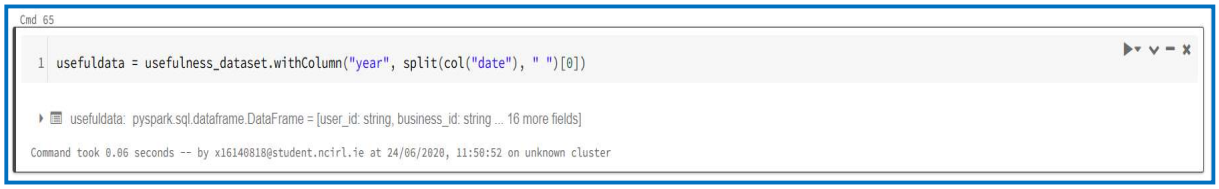


Figure 127: Remove Field

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

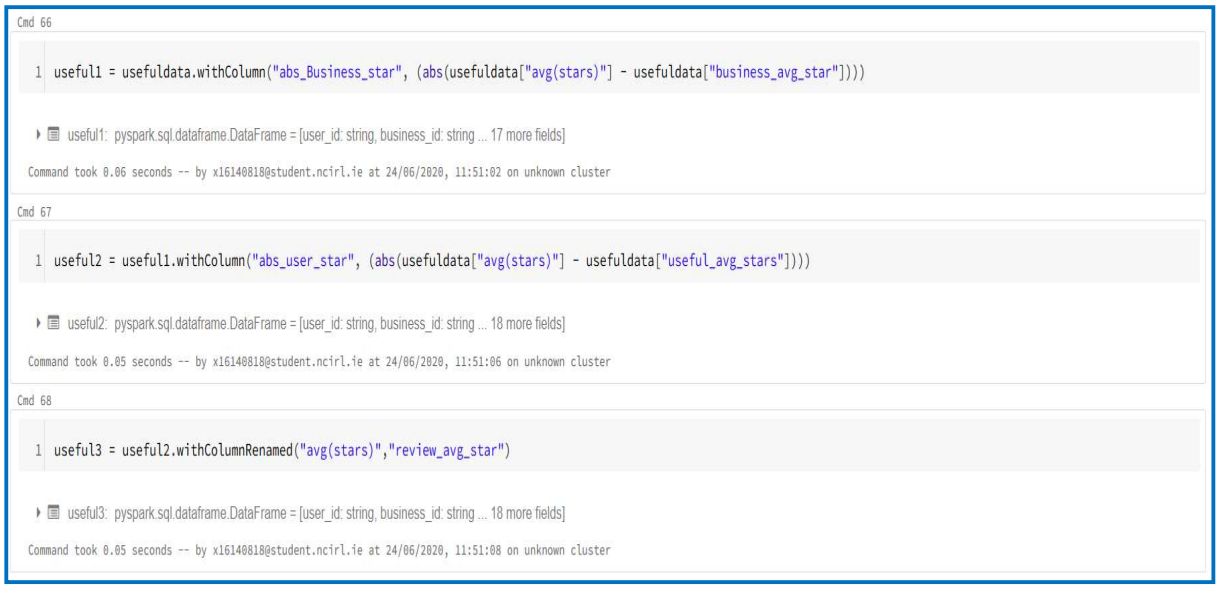


Figure 128: Deviation to Average Review Star Rating

15. Calculate Question Count and Exclamation Count

Figure 129: Question and Exclamation Count

16. Drop unnecessary fields

Figure 130: Remove Field

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

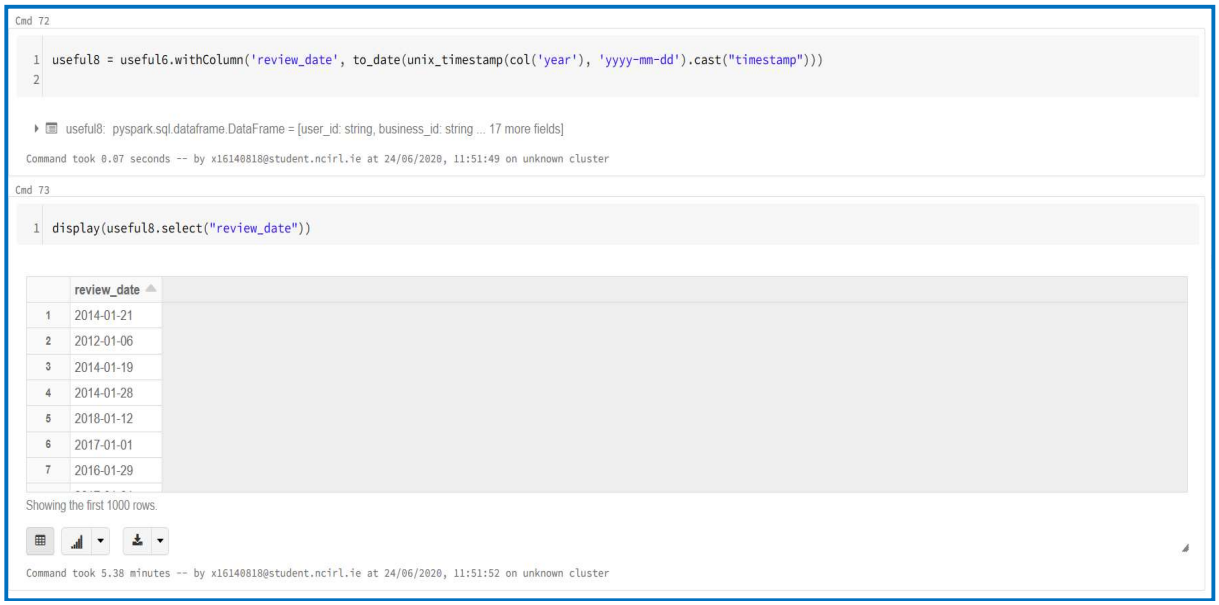


Figure 131: Convert Field

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

Figure 132: Sentence Count and Remove Field

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

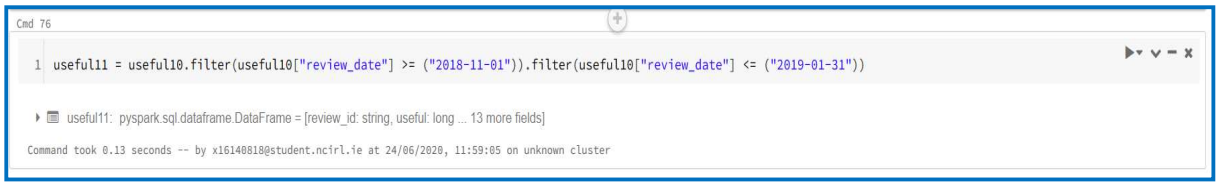


Figure 133: Filter Date

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

Figure 134: User Defined Function

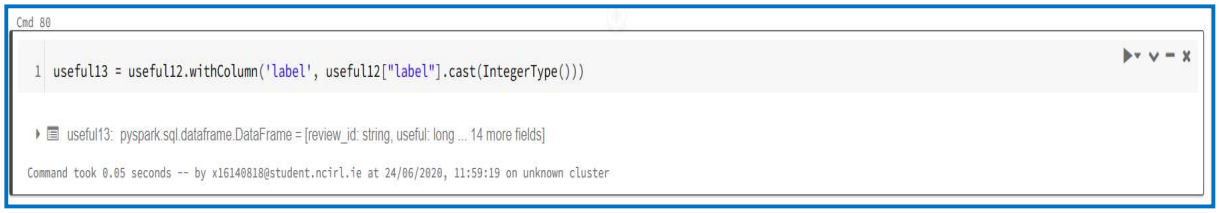


Figure 135: Cast to Integer

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

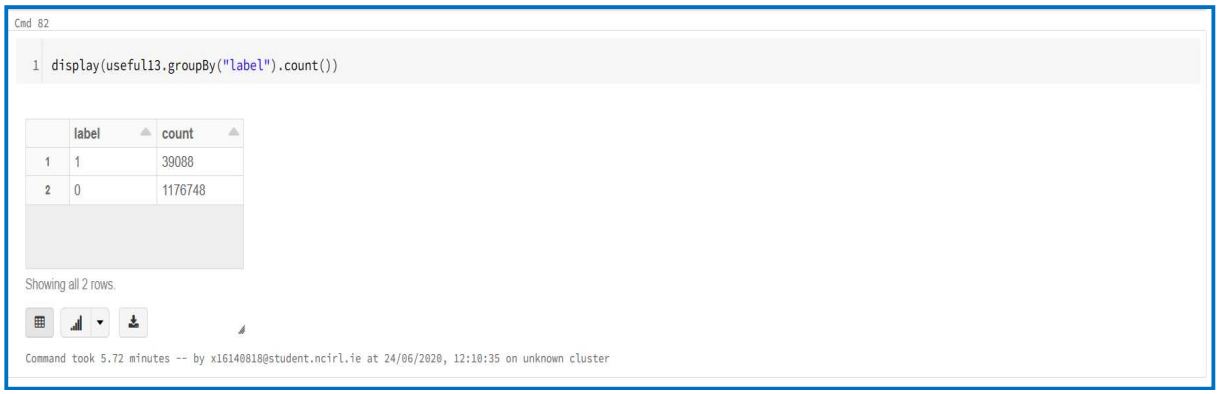


Figure 136: Star Count

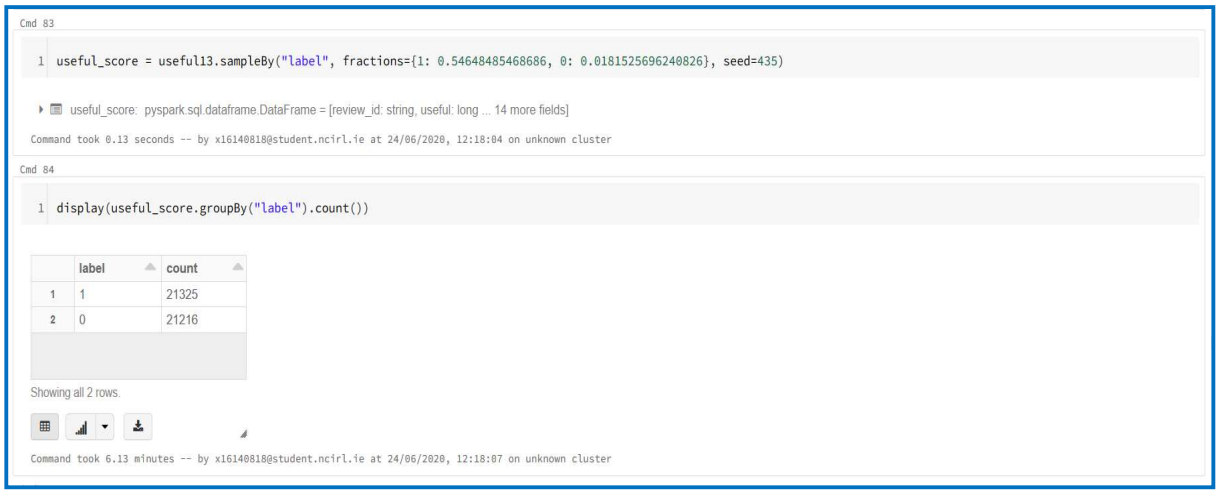


Figure 137: Starfield Sampling Attempt 1

Figure 138: Starfield Sampling Attempt 2

22. Drop unnecessary fields

Figure 139: Remove Field

23. Select independent variables and convert to Pandas DataFrame

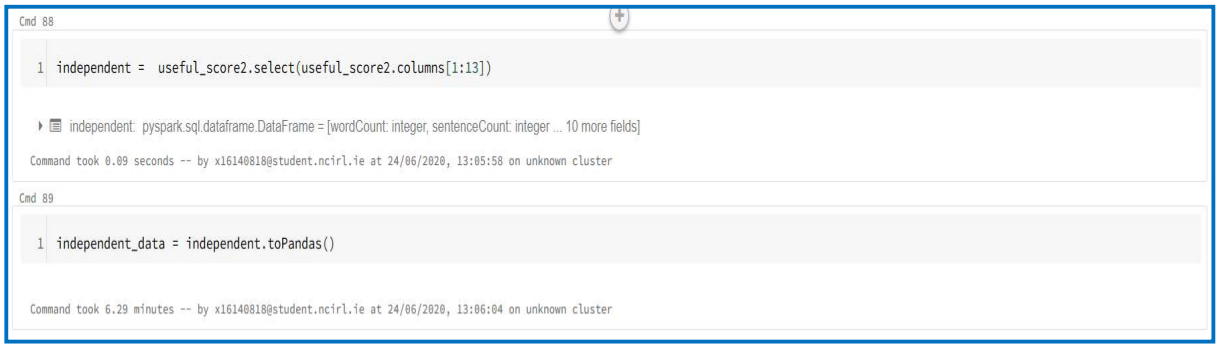


Figure 140: Independent Variables

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

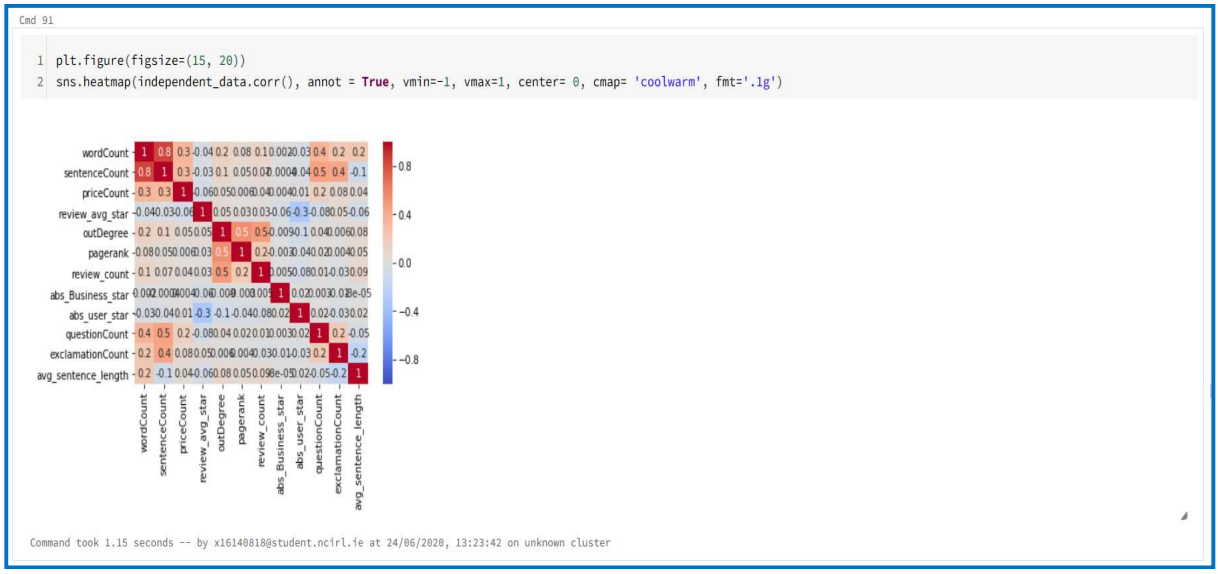


Figure 141: Correlation Heat Map

25. Check correlation between dependent variable and independent variables

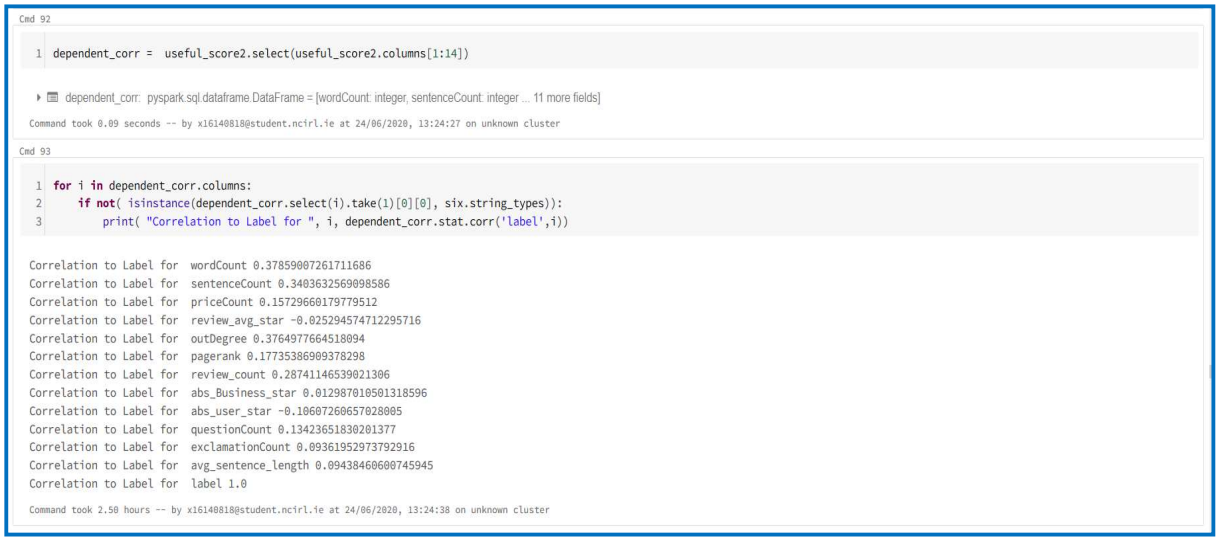


Figure 142: Correlation Dependent Variable

26. Check correlation if Sentence Count is dropped

Figure 143: Remove Sentence Count

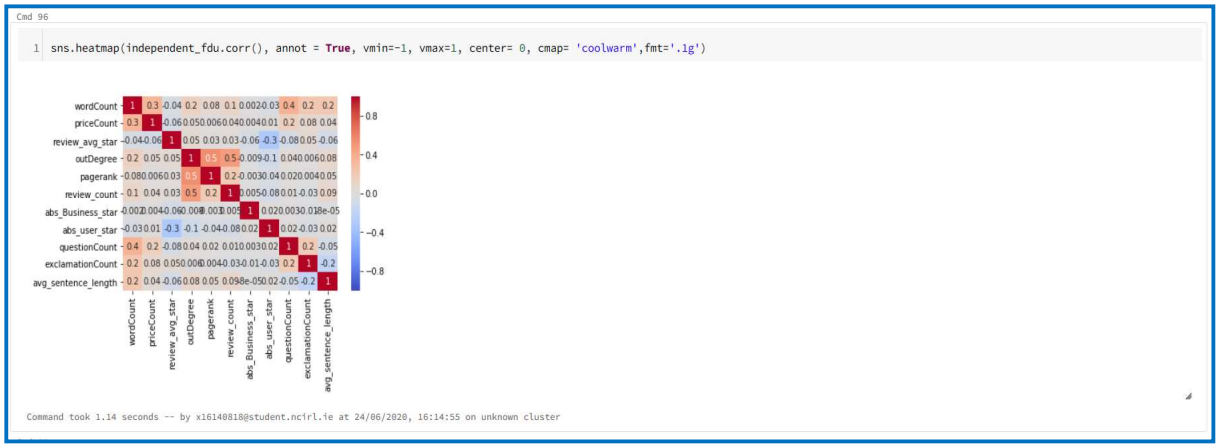


Figure 144: Correlation Heat Map

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

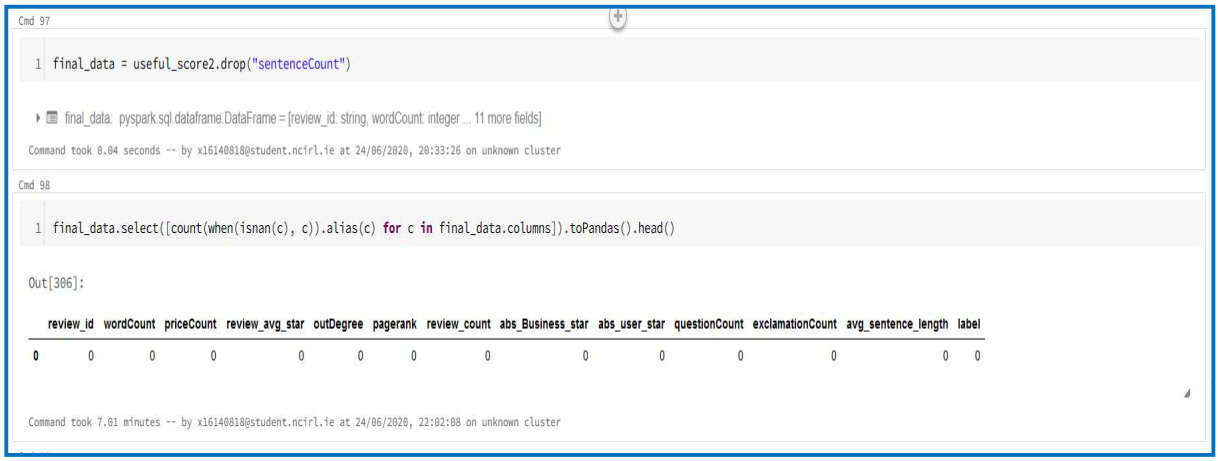


Figure 145: Sentence Count and Null Values

28. Combine all independent variables to one column

Figure 146: Combine to Single Column

29. Apply VectorAssembler and StandardScaler to the independent variables

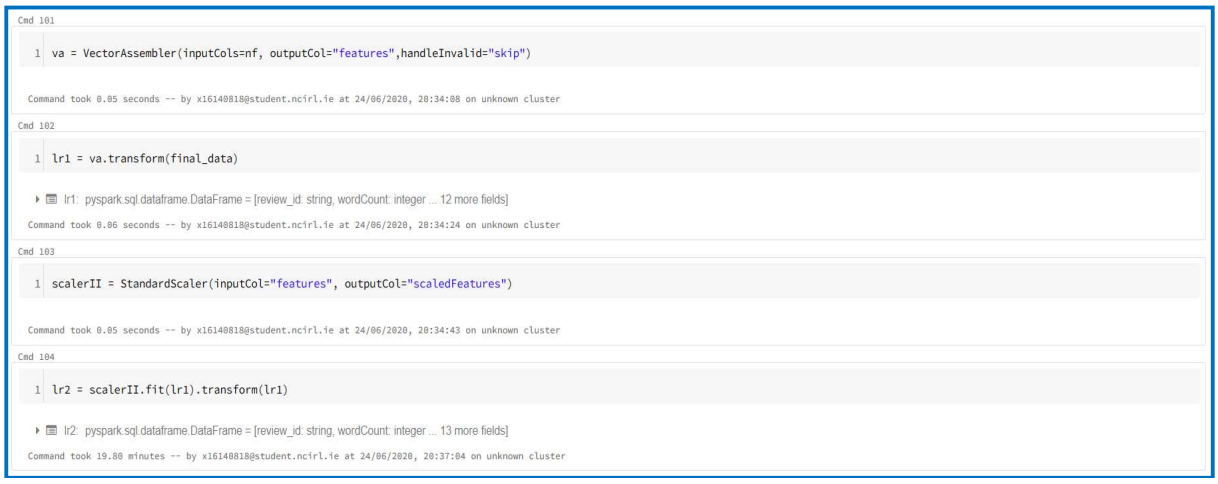


Figure 147: VectorAssembler and StandardScaler

30. Split DataFrame into train and test

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

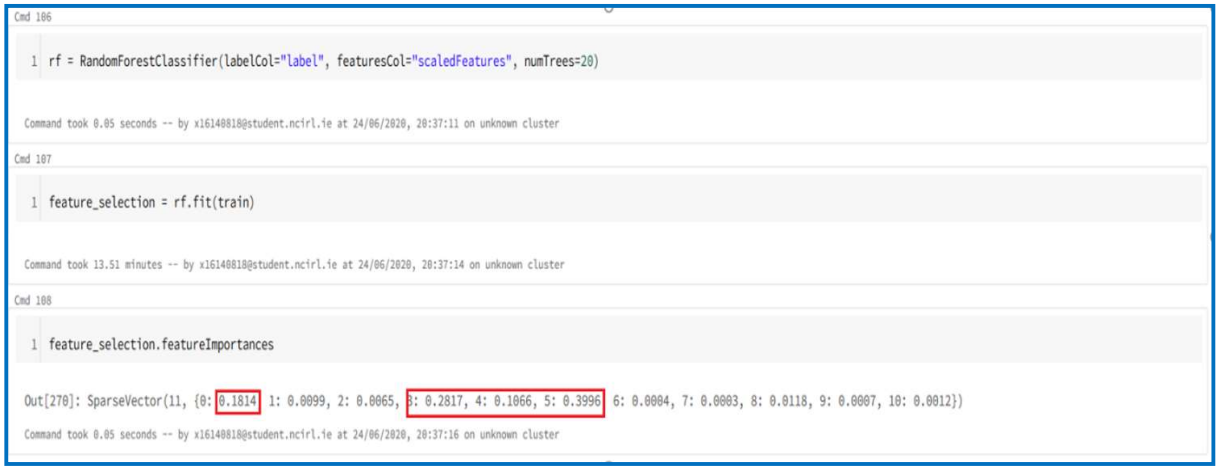


Figure 149: Random Forrest Feature Importance

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

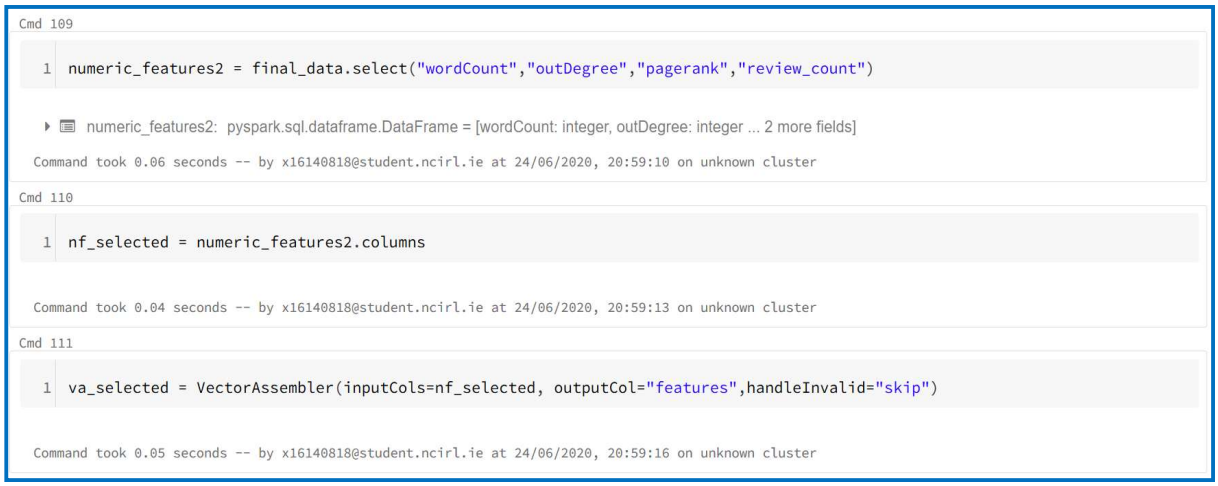


Figure 150: VectorAssembler

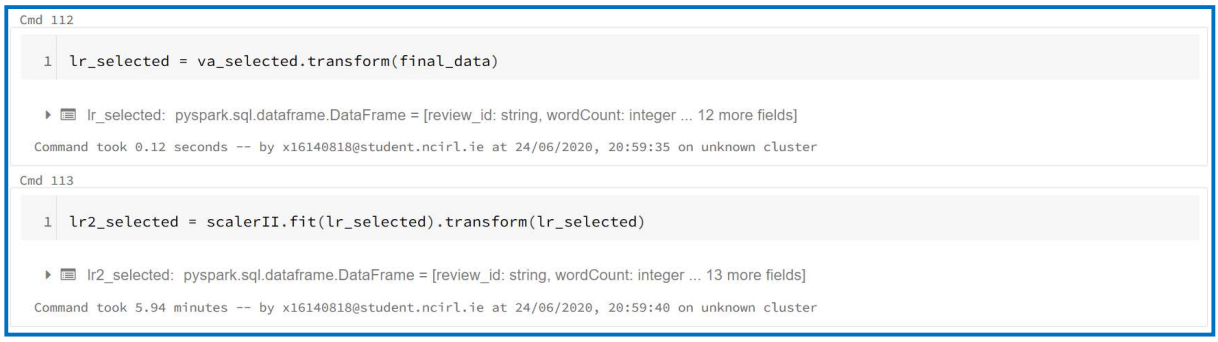


Figure 151: StandardScaler

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

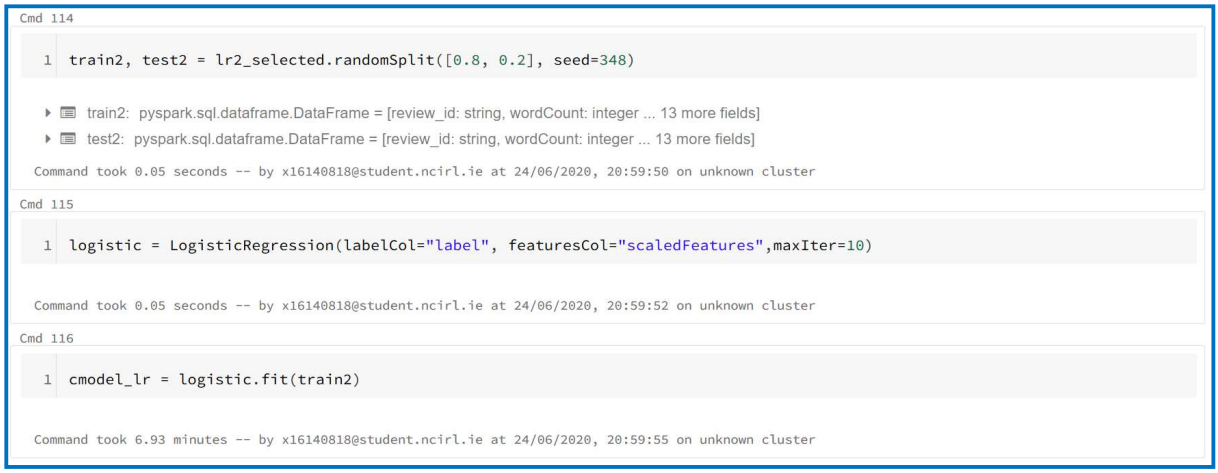


Figure 152: Logistic Regression (1 of 2)

Figure 153: Logistic Regression (2 of 2)

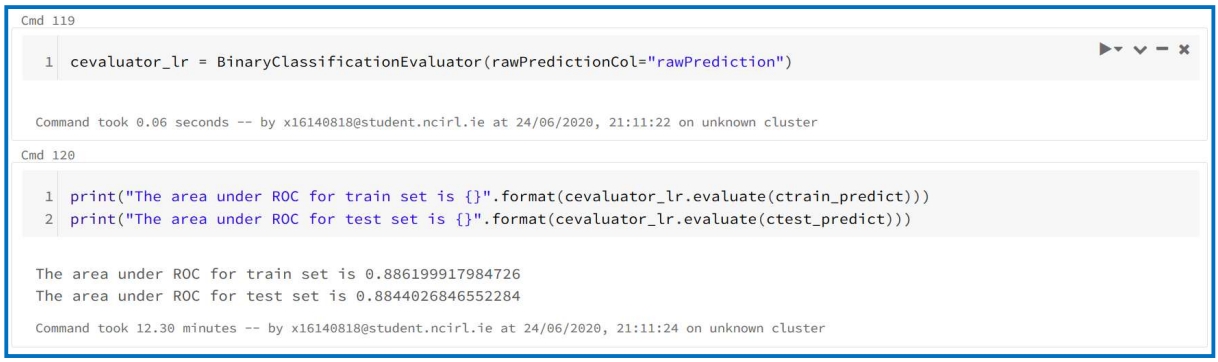


Figure 154: Logistic Regression Results (1 of 5)

34. Calculate Accuracy for LR model

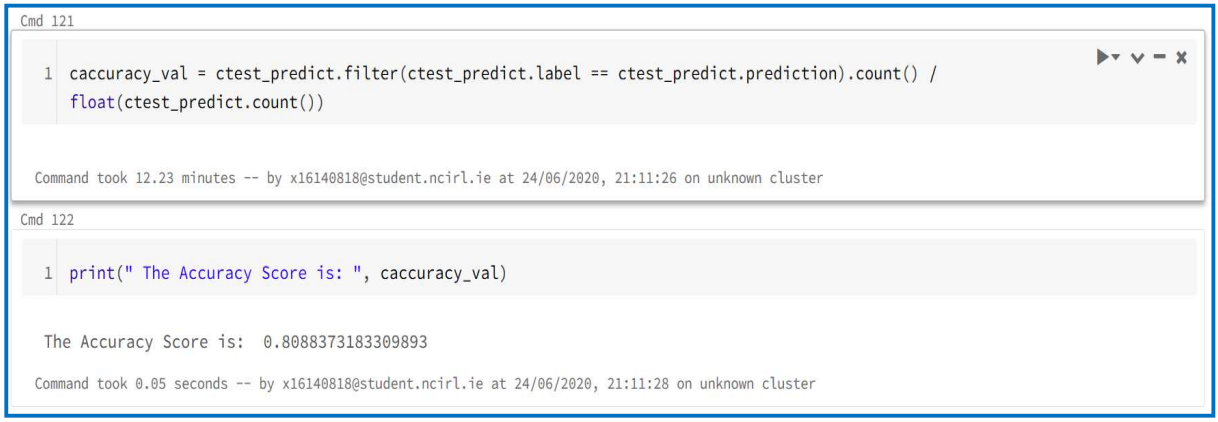


Figure 155: Logistic Regression Results (2 of 5)

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

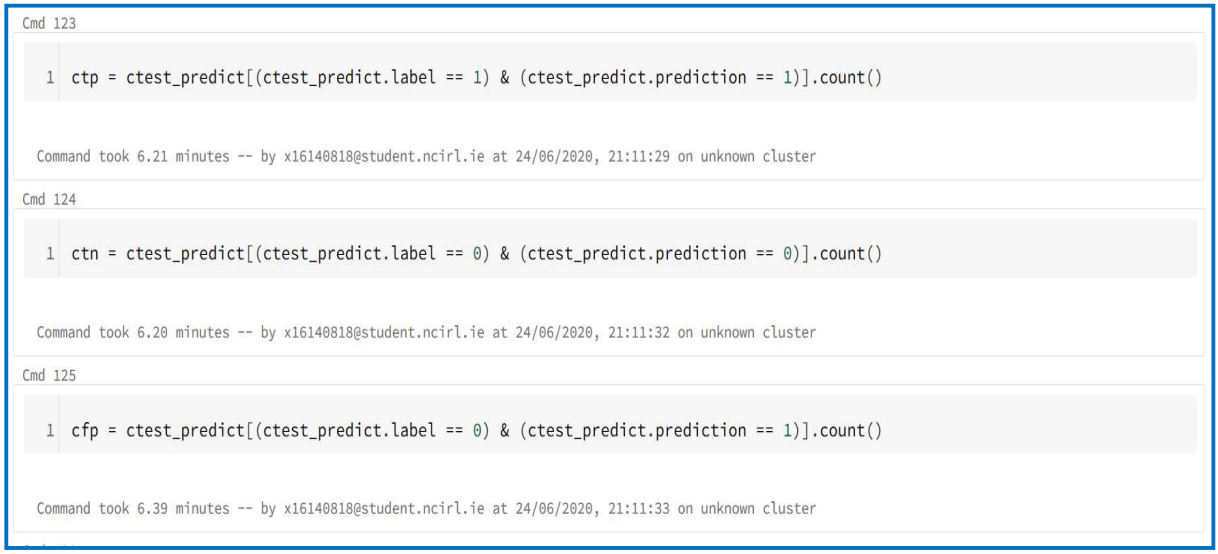


Figure 156: Logistic Regression Results (3 of 5)

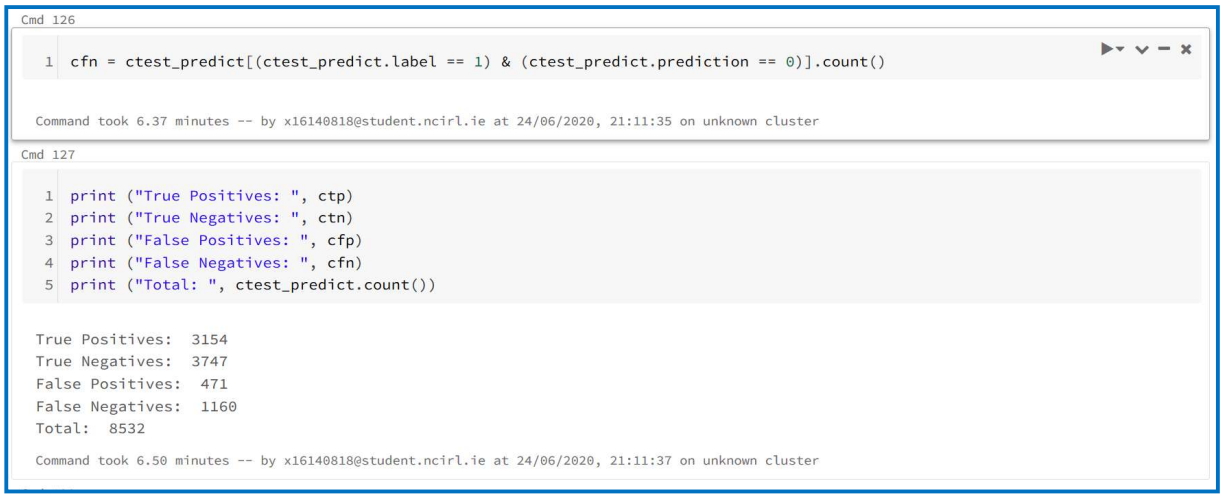


Figure 157: Logistic Regression Results (3 of 5)

36. Calculate Recall, Precision and F1 score

Figure 158: Logistic Regression Results (4 of 5)

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.

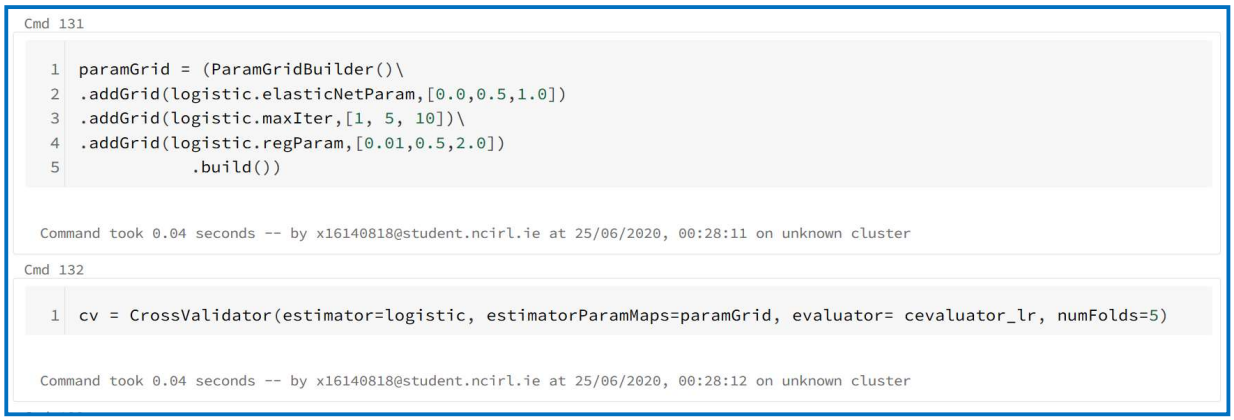


Figure 160: Cross Validation (1 of 2)

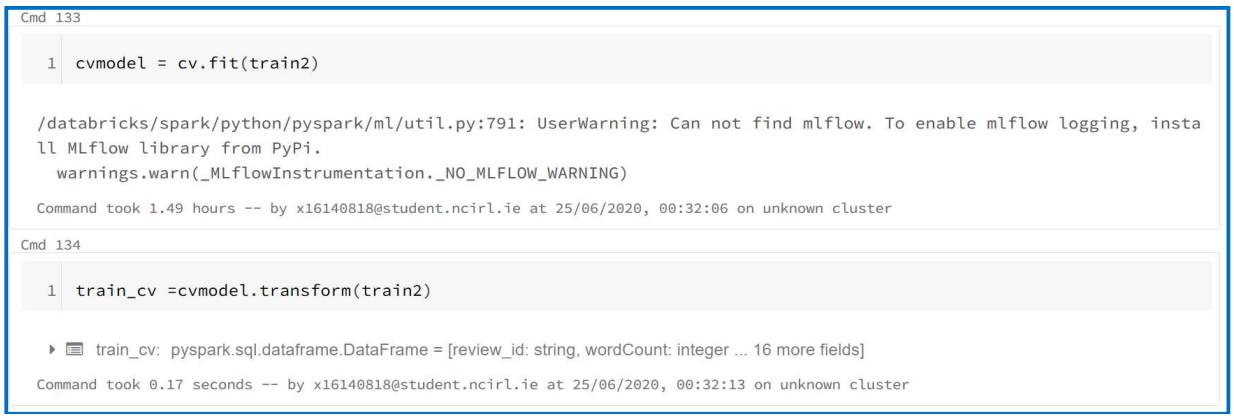


Figure 161: Cross Validation (2 of 2)

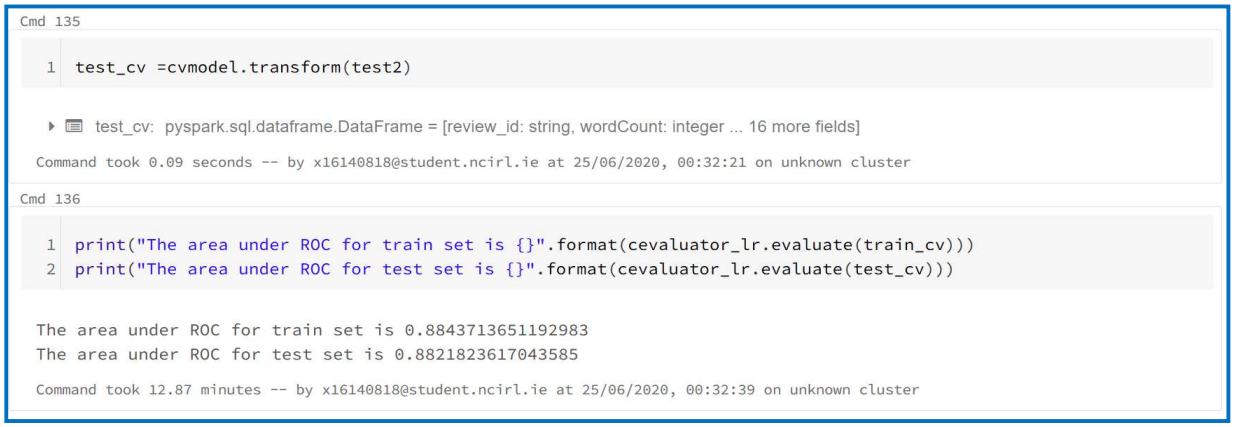


Figure 162: Cross Validation Results

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

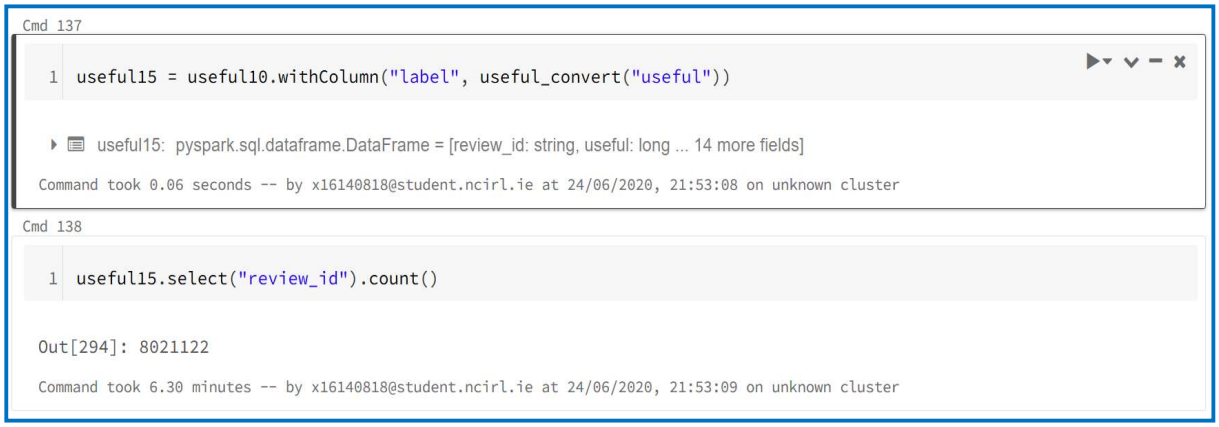


Figure 163: Logistic Regression Full Review DataFrame (1 of 4)

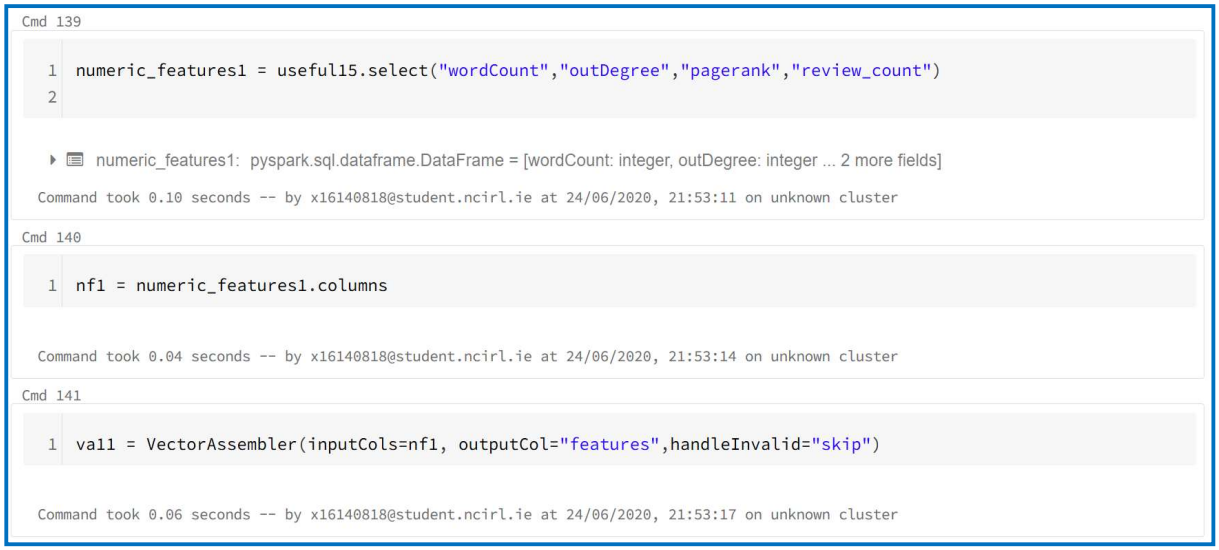


Figure 164: Logistic Regression Full Review DataFrame (2 of 4)

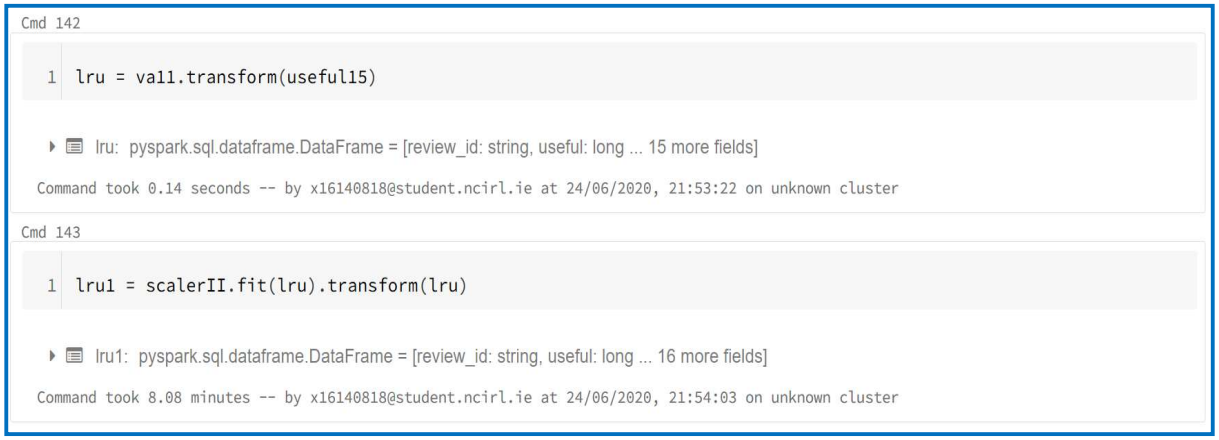


Figure 165: Logistic Regression Full Review DataFrame (3 of 4)

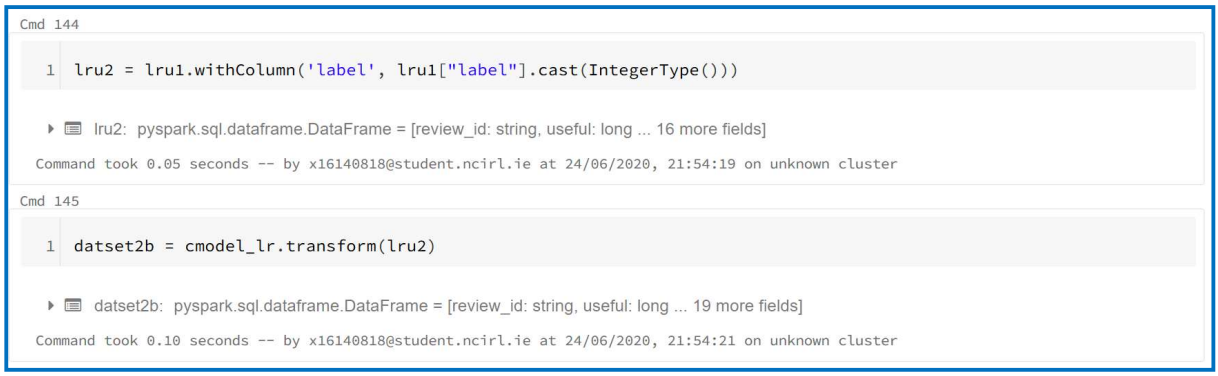


Figure 166: Logistic Regression Full Review DataFrame (4 of 4)

39. The Dataframe wads 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

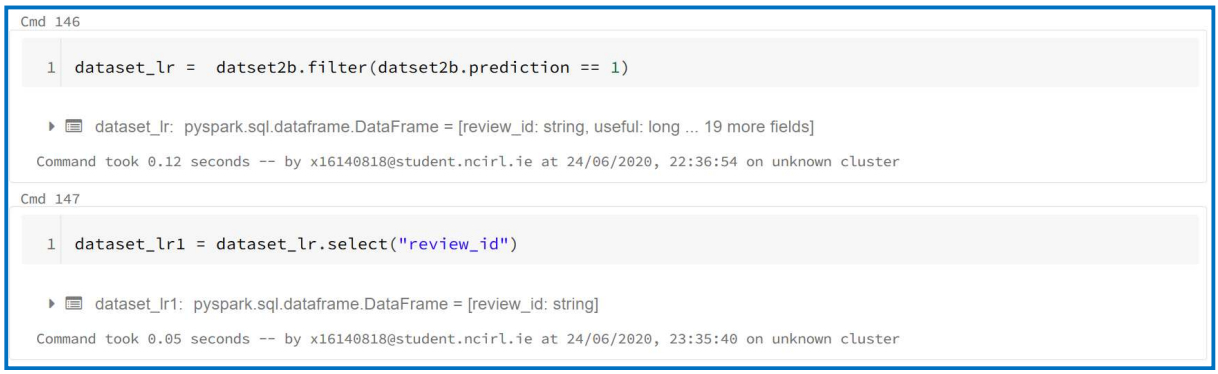


Figure 167: Filter for Useful Reviews

Figure 168: Merge DataFrames

40. Distribution of Star Rating

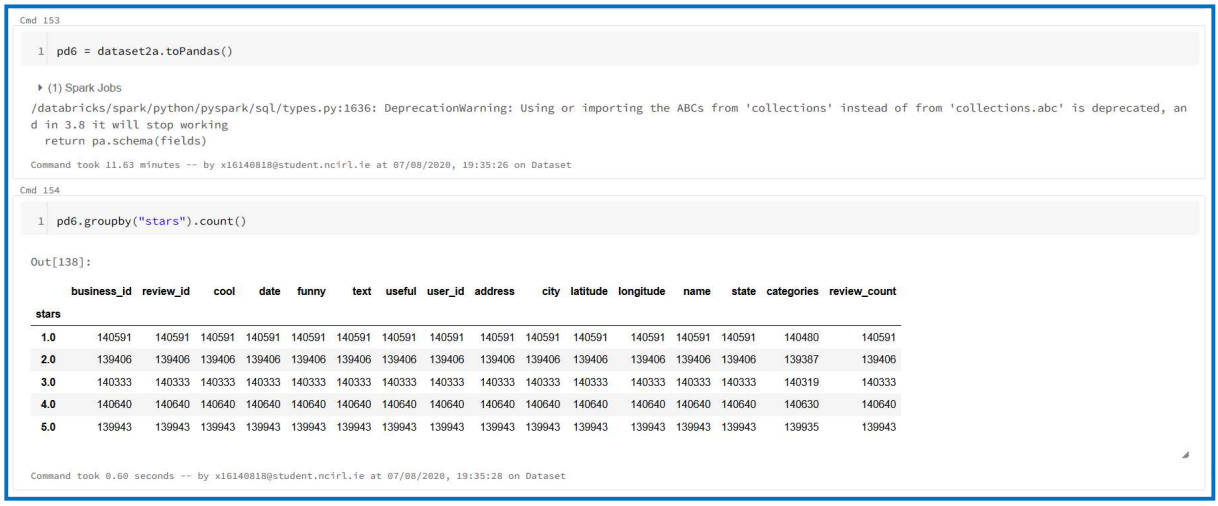
Figure 169: Star Distribution

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 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]:
```

stars	business_id	review_id	cool	date	funny	text	useful	user_id	address	city	latitude	longitude	name	state	categories	review_count
1.0	140591	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	139406	139387	139406
3.0	140333	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	140640	140630	140640
5.0	139943	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
```

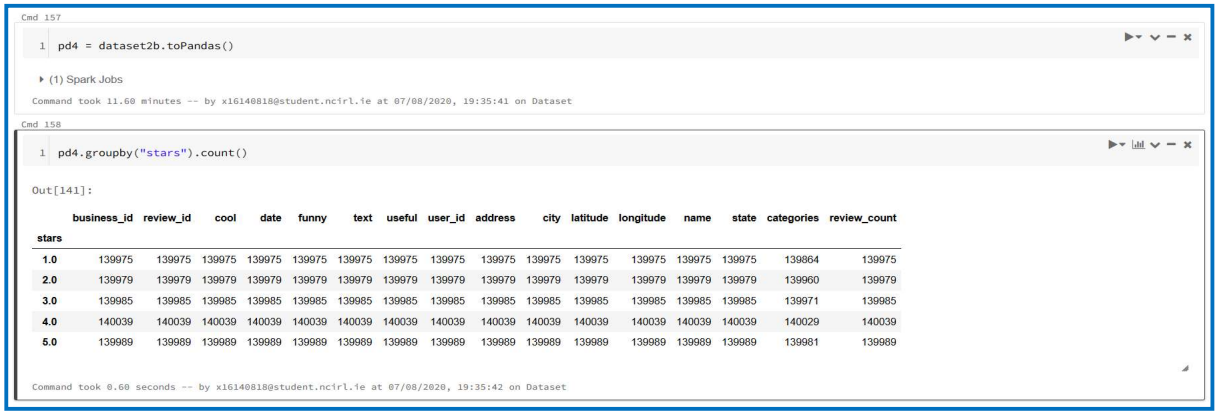
Figure 172: Pandas Groupby

```
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	140465
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)

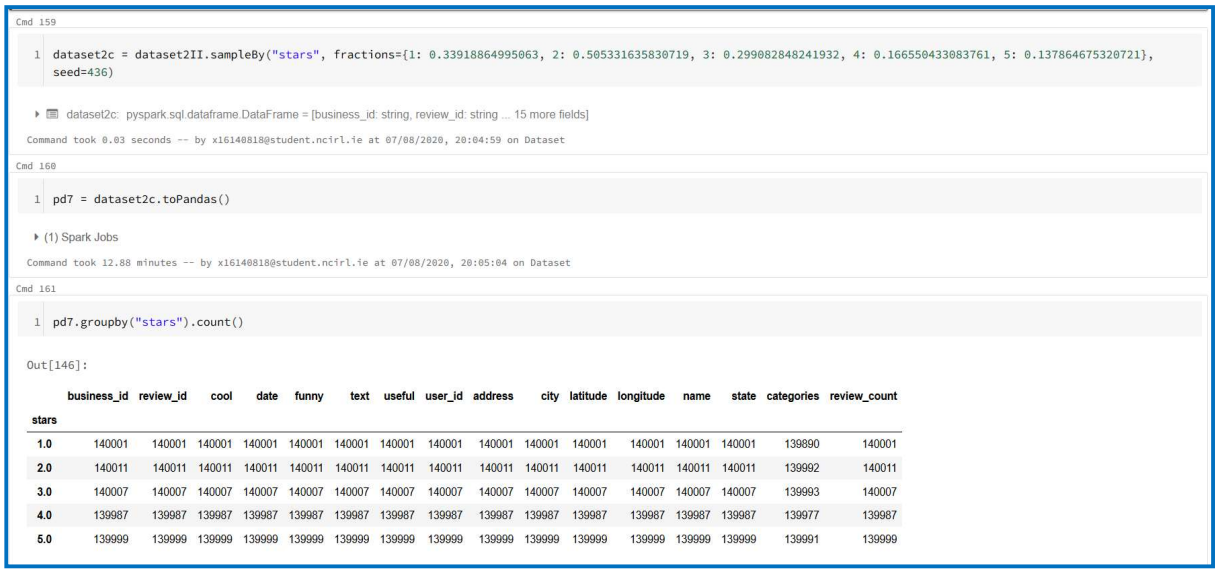


```
Cmd 157
1 pd4 = dataset2b.toPandas()

▸ (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]:
```

stars	business_id	review_id	cool	date	funny	text	useful	user_id	address	city	latitude	longitude	name	state	categories	review_count
1.0	139975	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	139979	139960	139979
3.0	139985	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	140039	140029	140039
5.0	139989	139989	139989	139989	139989	139989	139989	139989	139989	139989	139989	139989	139989	139989	139981	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 0.03 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]:
```

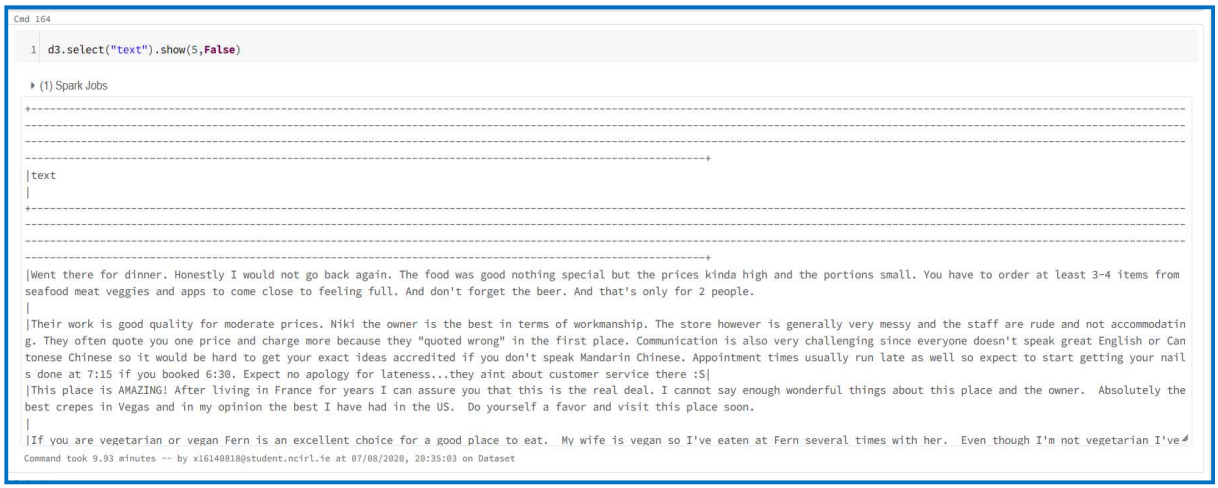
stars	business_id	review_id	cool	date	funny	text	useful	user_id	address	city	latitude	longitude	name	state	categories	review_count
1.0	140001	140001	140001	140001	140001	140001	140001	140001	140001	140001	140001	140001	140001	140001	139890	140001
2.0	140011	140011	140011	140011	140011	140011	140011	140011	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

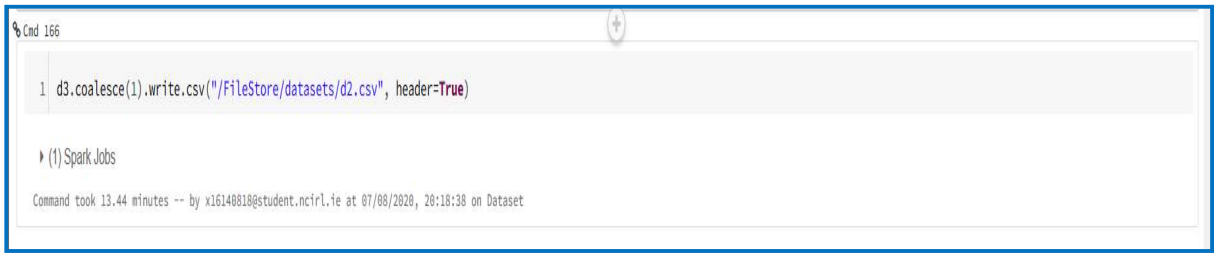


Figure 178: Write to Databricks File Store

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

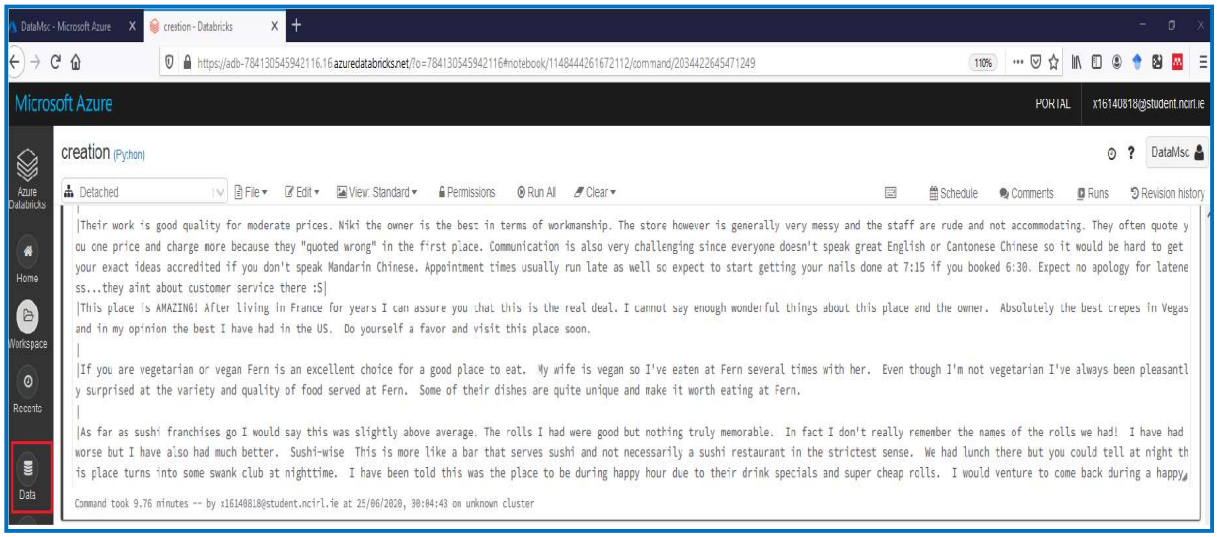


Figure 179: Connect to Data in DBFS

46. Add Data

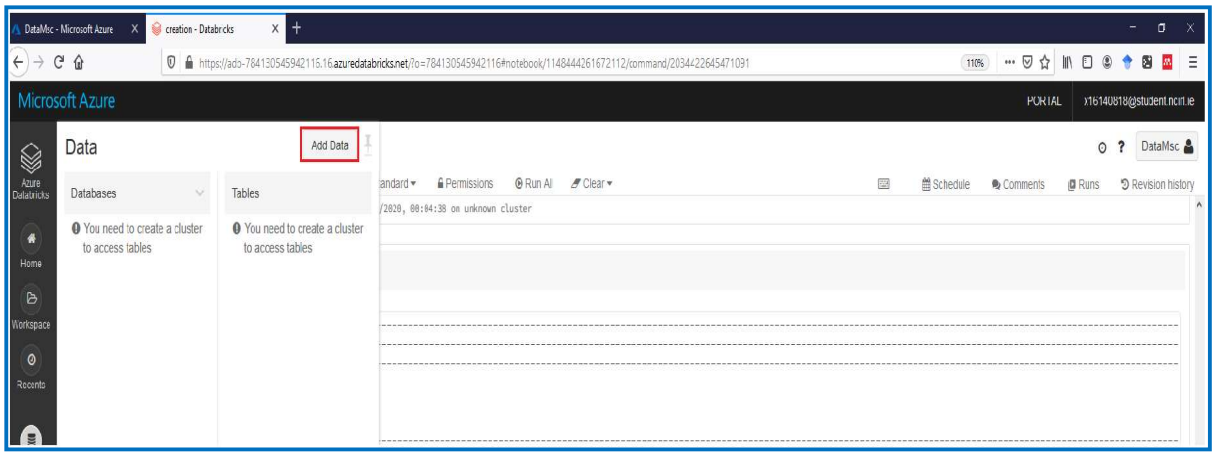


Figure 180: Click Add Data

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.

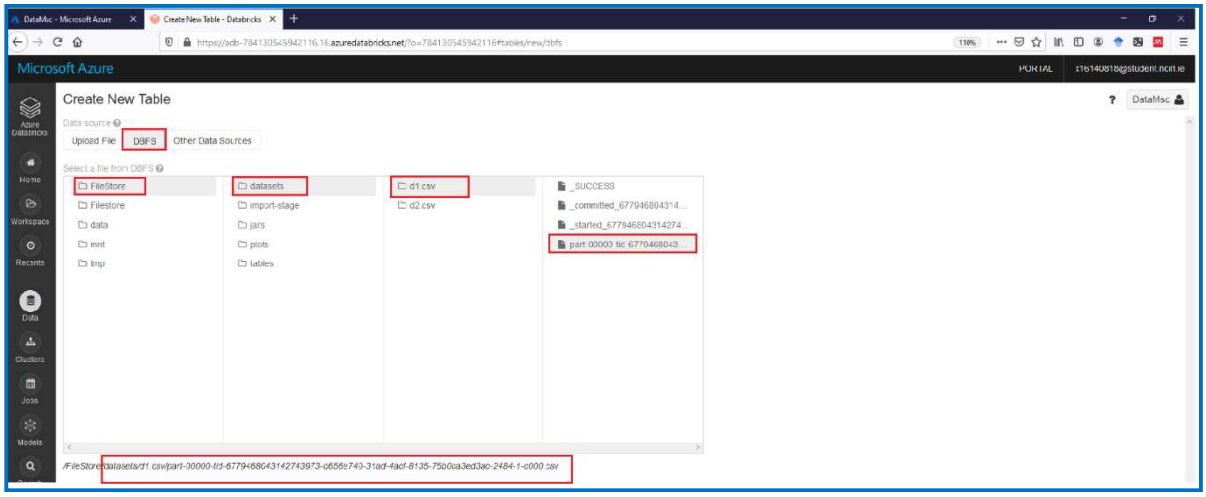


Figure 181: Dataset 1 File Path

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 Radom 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 website[12]. 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-6624391422310732 86-bad7a3a4-8a45-4896-8737-cba2757ef27f-145331-1-c000.csv

[12] https://drive.google.com/drive/u/0/my-drive

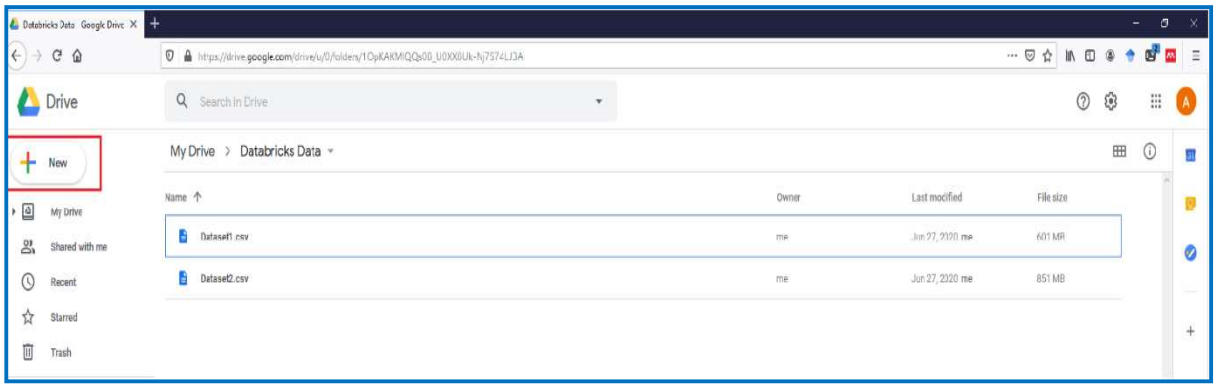


Figure 182: Upload Data from Local

3. Go the Colab website[13] and click File, New Notebook

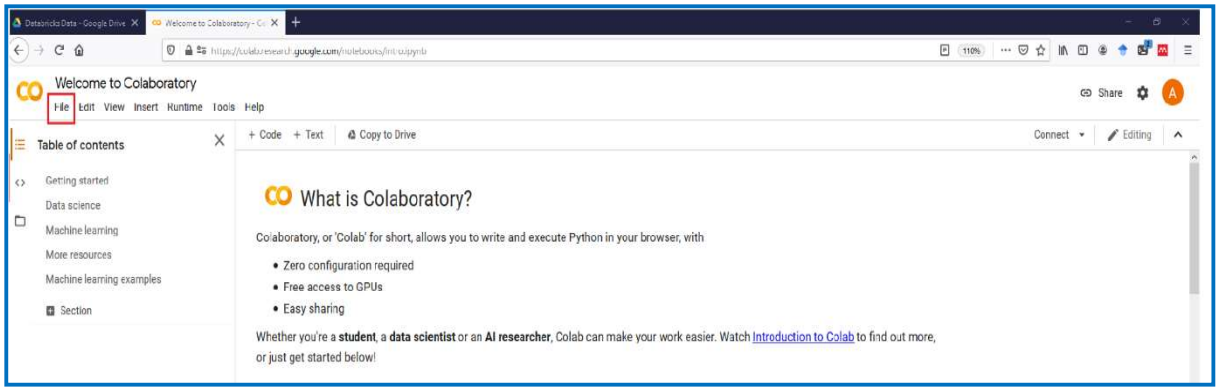


Figure 183: Launch Notebook

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

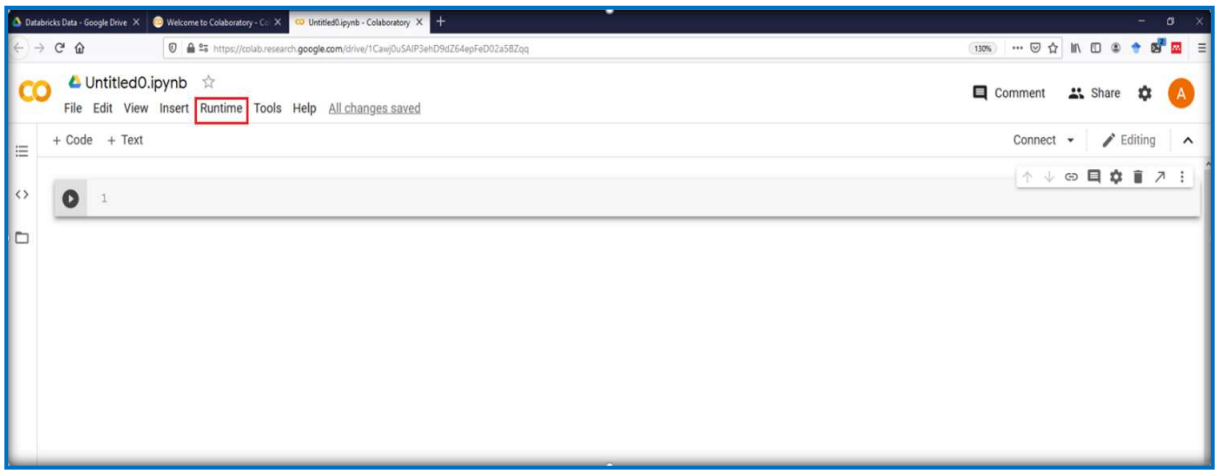


Figure 184: TPU Set-up (1 of 2)

[13] https://colab.research.google.com/notebooks/intro.ipynb

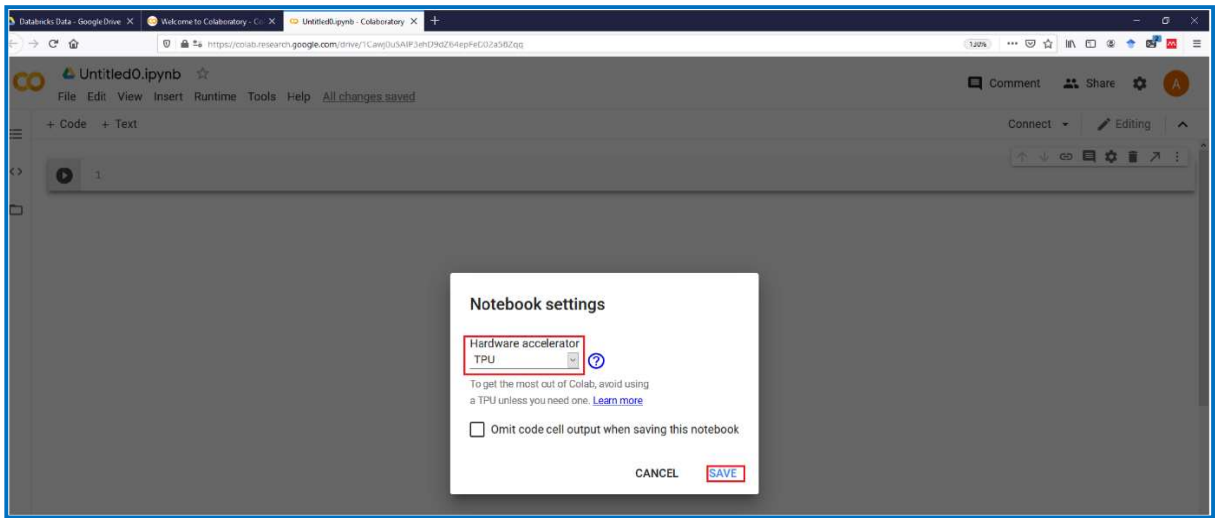


Figure 185: TPU Set-up (2 of 2)

5. 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

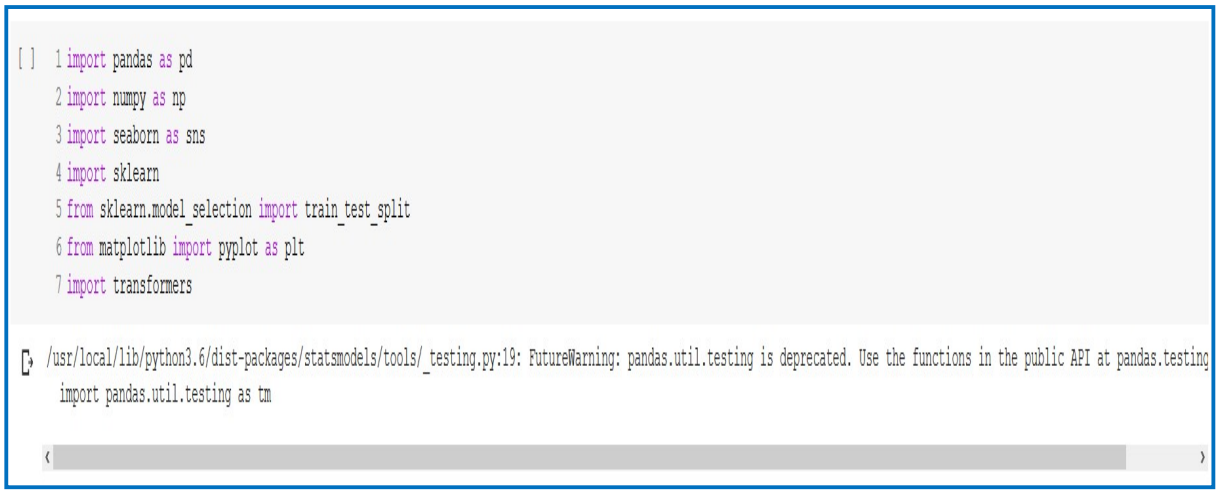


Figure 187: Import Libraries

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

Figure 188: Mount Google Drive

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

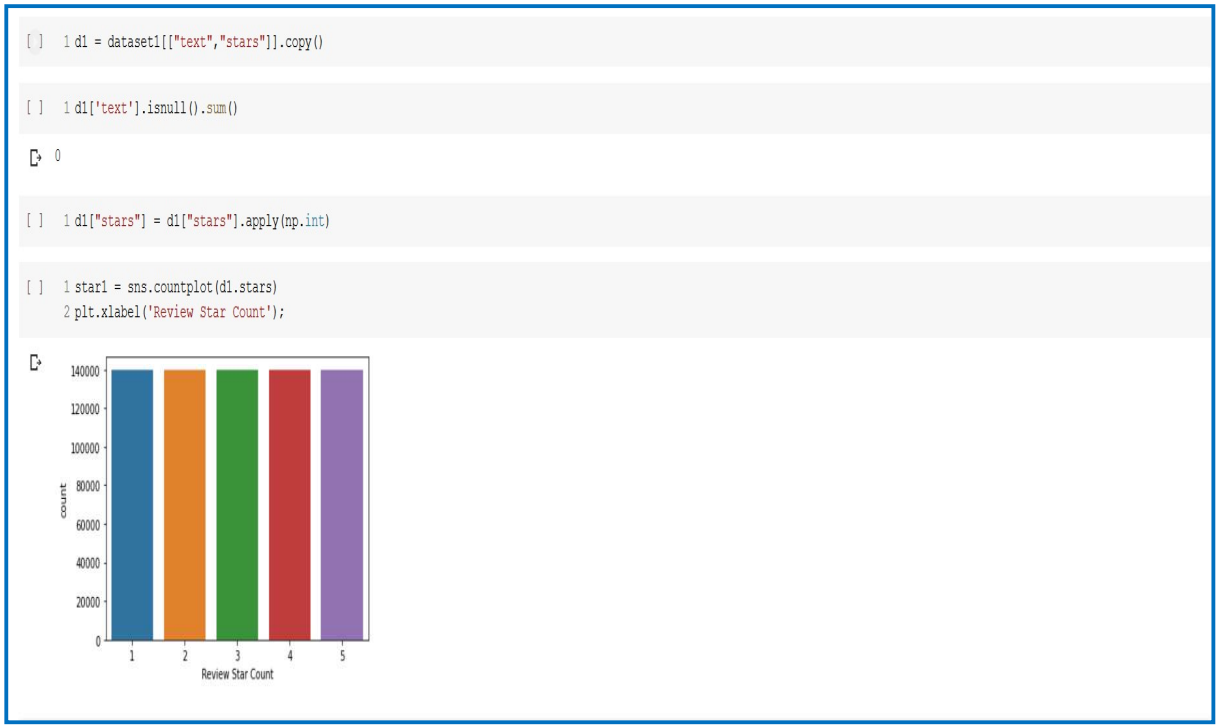


Figure 189: Star Distribution

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

```
1 d1["stars"] = (d1["stars"] - 1).copy()
```

+ Code + Text

```
[ ] 1 d1 = d1.rename(columns={'stars': 'labels'})
```

```
[ ] 1 d1_train, d1_test = train_test_split(d1, test_size=0.0714299, random_state=43)
```

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

	text	labels
0	Came here on a Friday night because our first ...	1
1	Great location and close to everything! We had...	3
2	All UPS stores aren't created equally they may...	1
3	I stopped here for a quick dinner with a frien...	3
4	This is a fun place for people of all ages. If...	3
...	...	...

Figure 190: Format Labels

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

```
[ ] 1 d1_train, d1_eval = train_test_split(d1_train, test_size=0.076925, random_state=44)
```

```
[ ] 1 star5 = sns.countplot(d1_train.labels)
    2 plt.xlabel('Review Star Count')
    3 plt.title('Star Distribution');
```

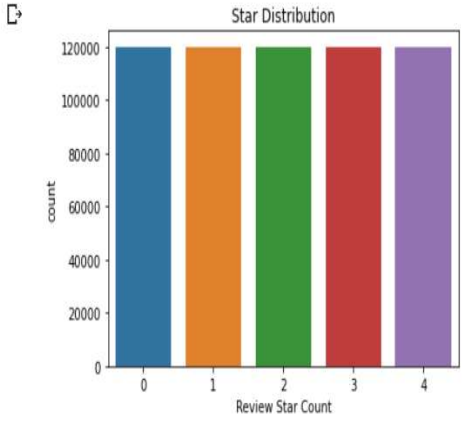


Figure 191: Training Star Distribution

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

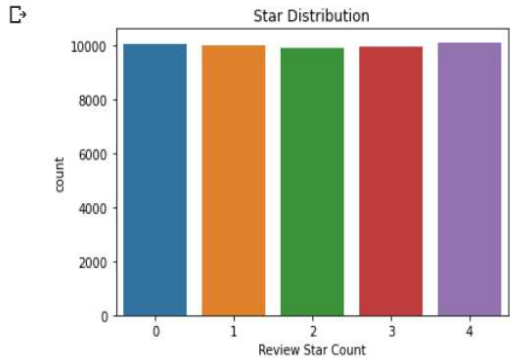


Figure 192: Evaluation Star Distribution

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

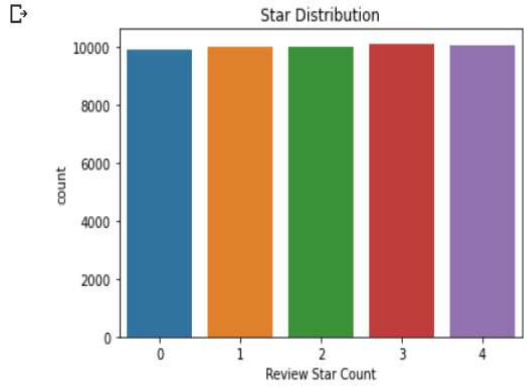


Figure 193: Test Star Distribution

```
[ ]  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');
```

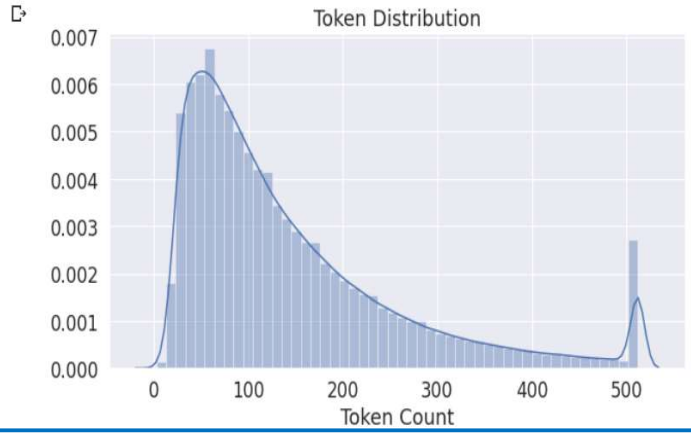


Figure 201: Token Distribution

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 website[14]. Click Create New Instance

[14] https://compute.genesiscloud.com/dashboard/instances

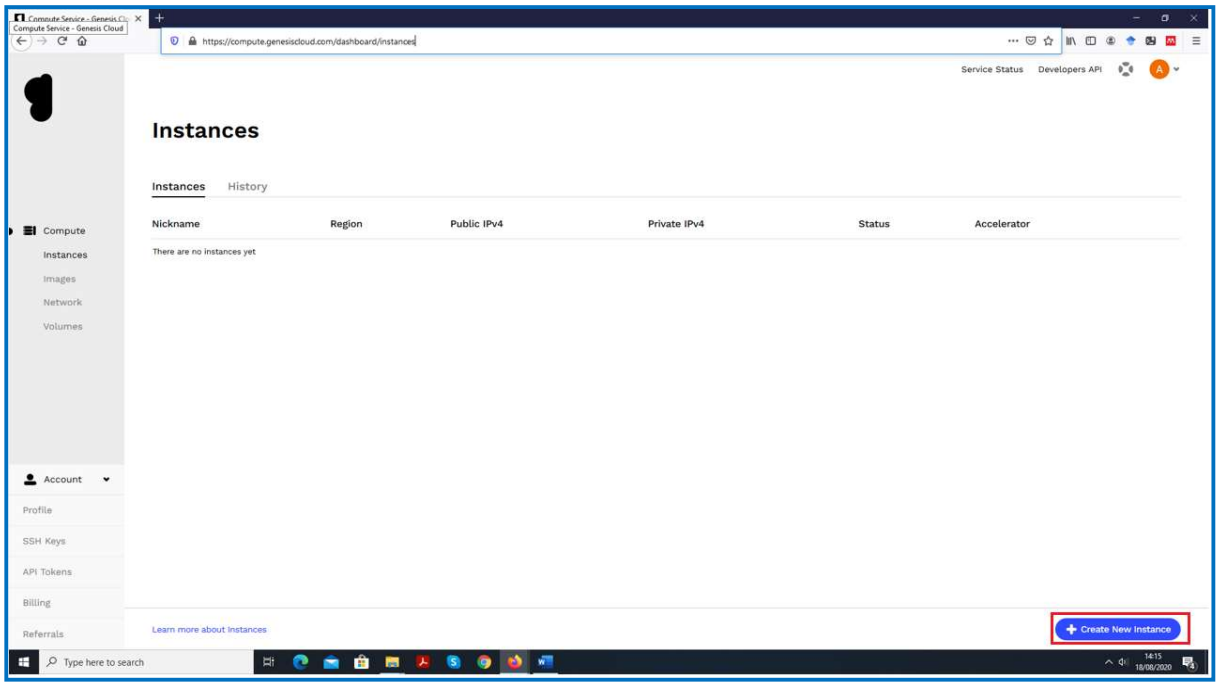


Figure 202: Instance Webpage

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.

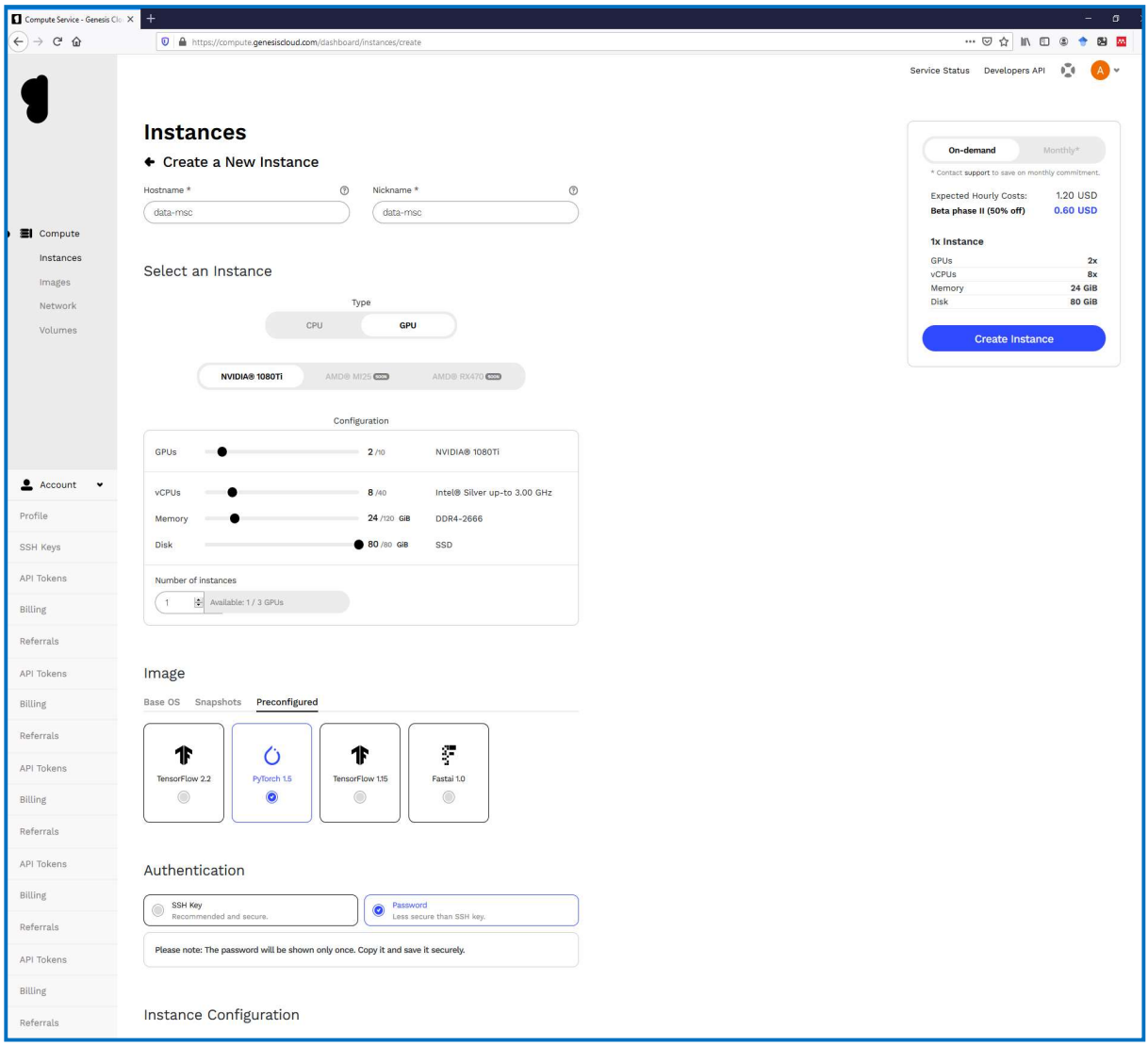


Figure 203: Create New Instance (1 of 2)

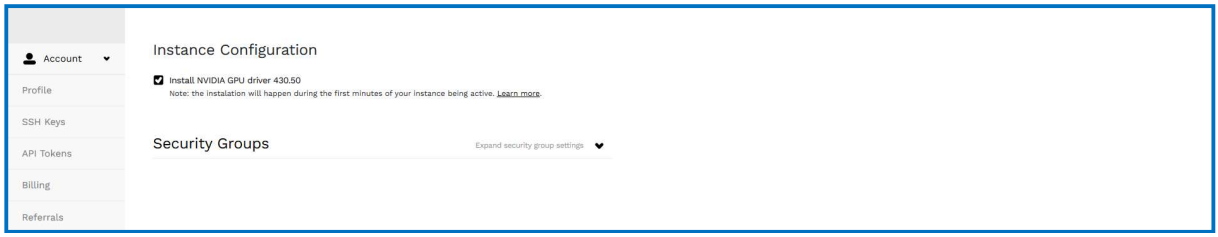


Figure 204: Create New Instance (2 of 2)

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.

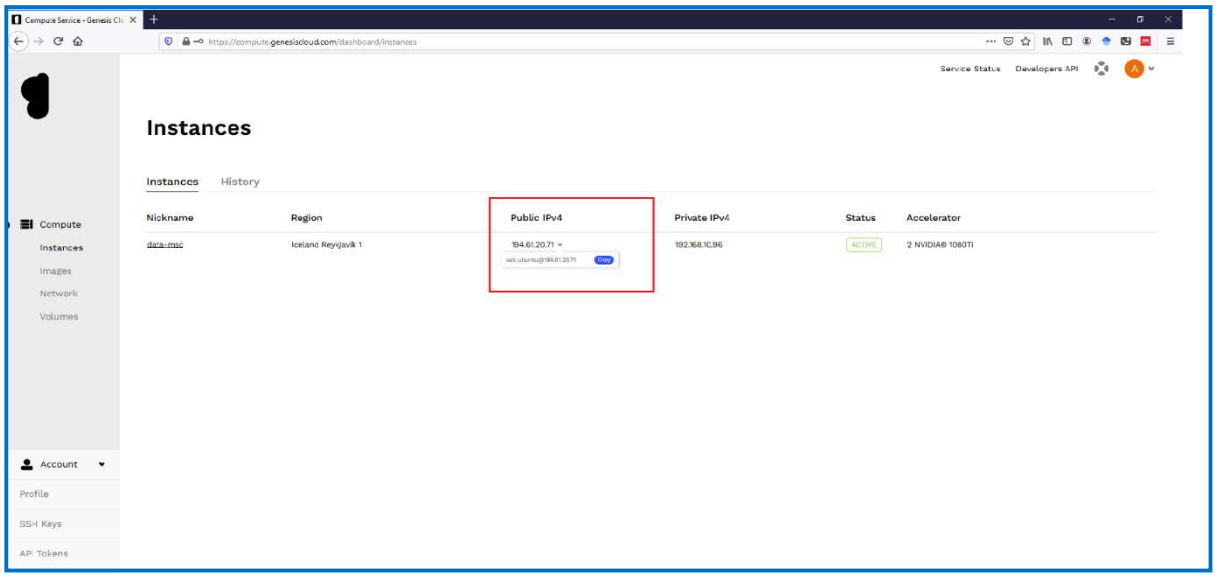


Figure 205: Ipv4

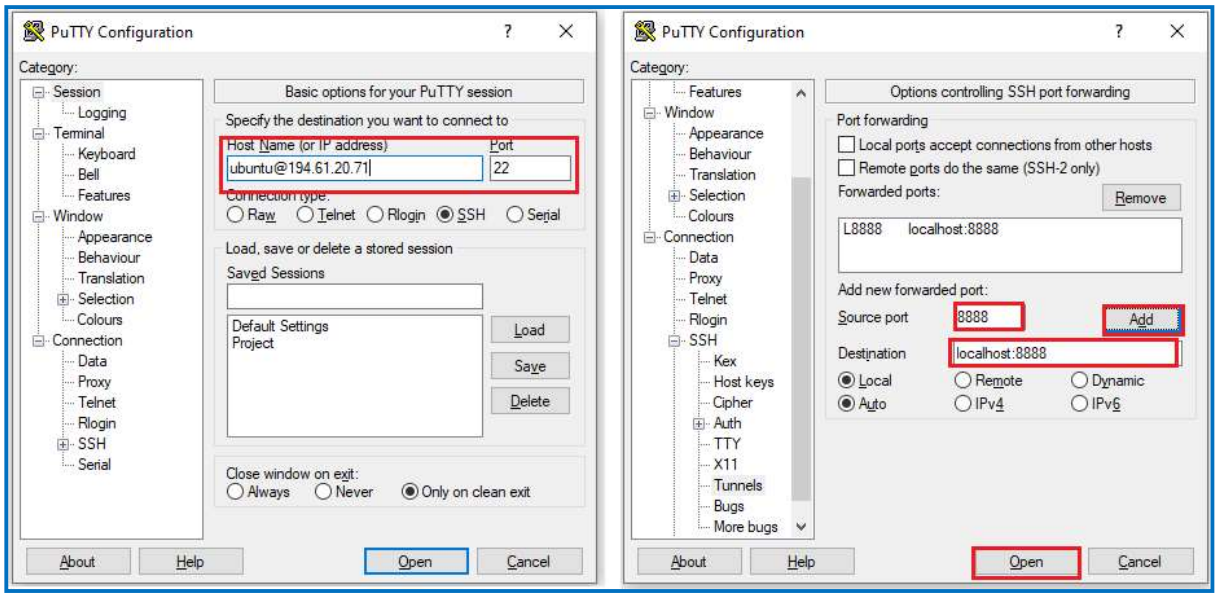


Figure 206: PuTTY Configuration

4. 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.

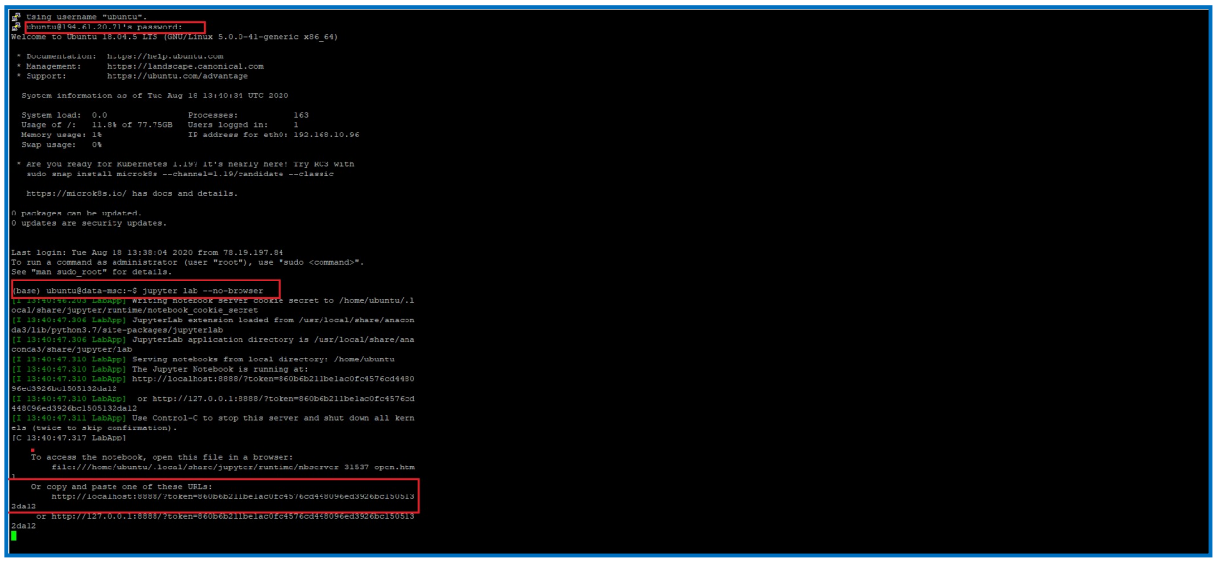


Figure 207: PuTTy Server

5. After JupyterLab is connected the data stored in Drive needs to be uploaded using Cliget. There are 6 files in total relating to train, evaluation and test data. Right click on a file. Click Download, when the window appears do not click save. Check the Cliget add-on in the top right-hand corner of the browser window. A number should appear indicating that the add-on has generated the curl command. Repeat for the remaining 5 files. Next go to JupyterLab and click terminal. Paste the curl commands in the terminal. The data will be uploaded from Drive to Jupyter.

6. After the data has been transferred the 6 models are run. For the configuration manual the code saved by W&B will be used as it good for reporting. The candidate has included both the W&B python files and the Notebook files run in Jupyter for the project. The code and results are displayed next

6.1 Experiment 1 ALBERT-base D1

Since the 31st of July PyTorch has been updated to 1.6. For running the models the candidate used PyTorch 1.5 therefore if run after 31st a user must replace !pip install simpletransformers with !pip install simpletransformers==0.45.5. Only the models used as part of the Demo Video have been updated.

```
[ ]: !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

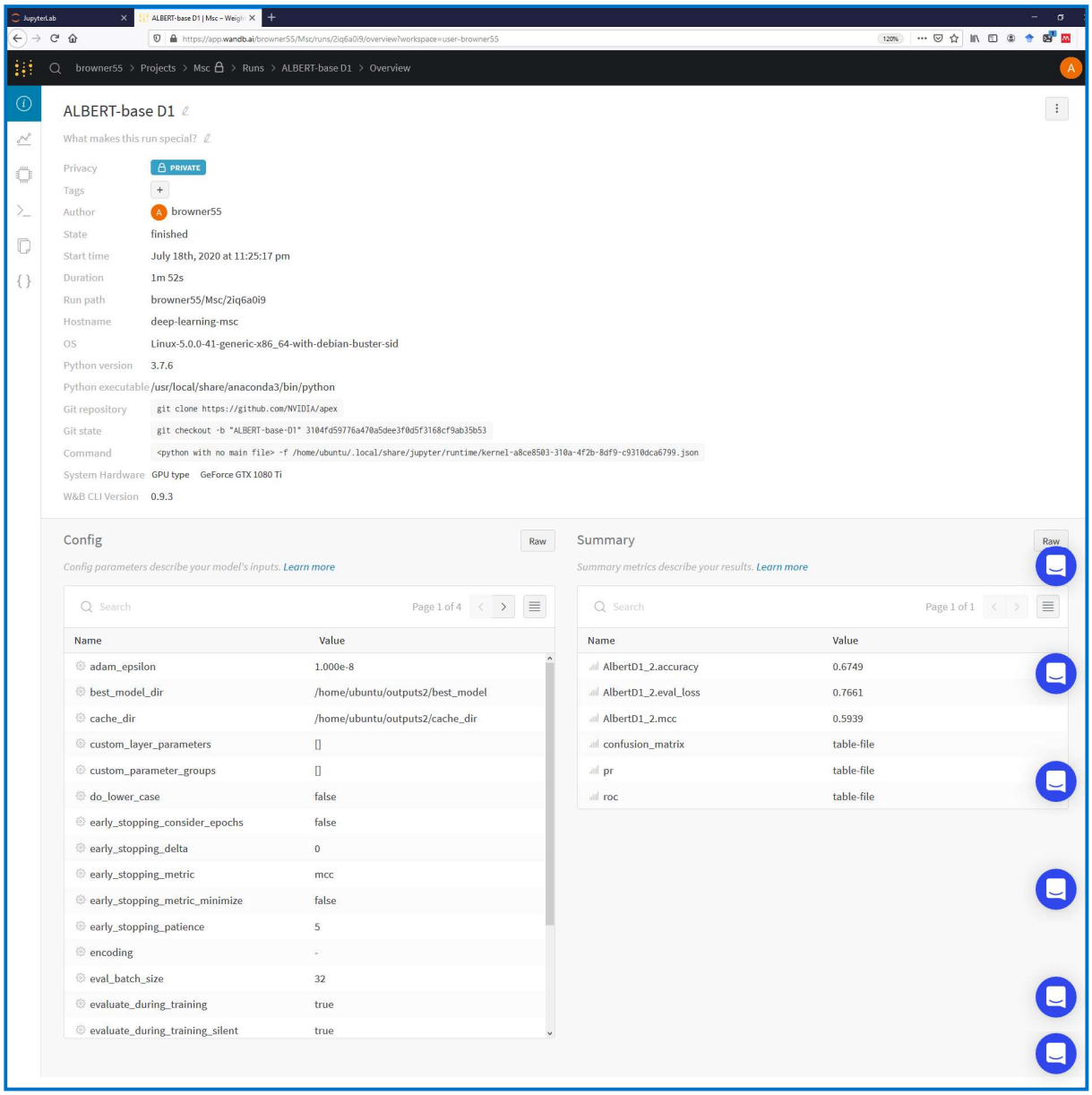


Figure 209: ALBERT-base D1 Results

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

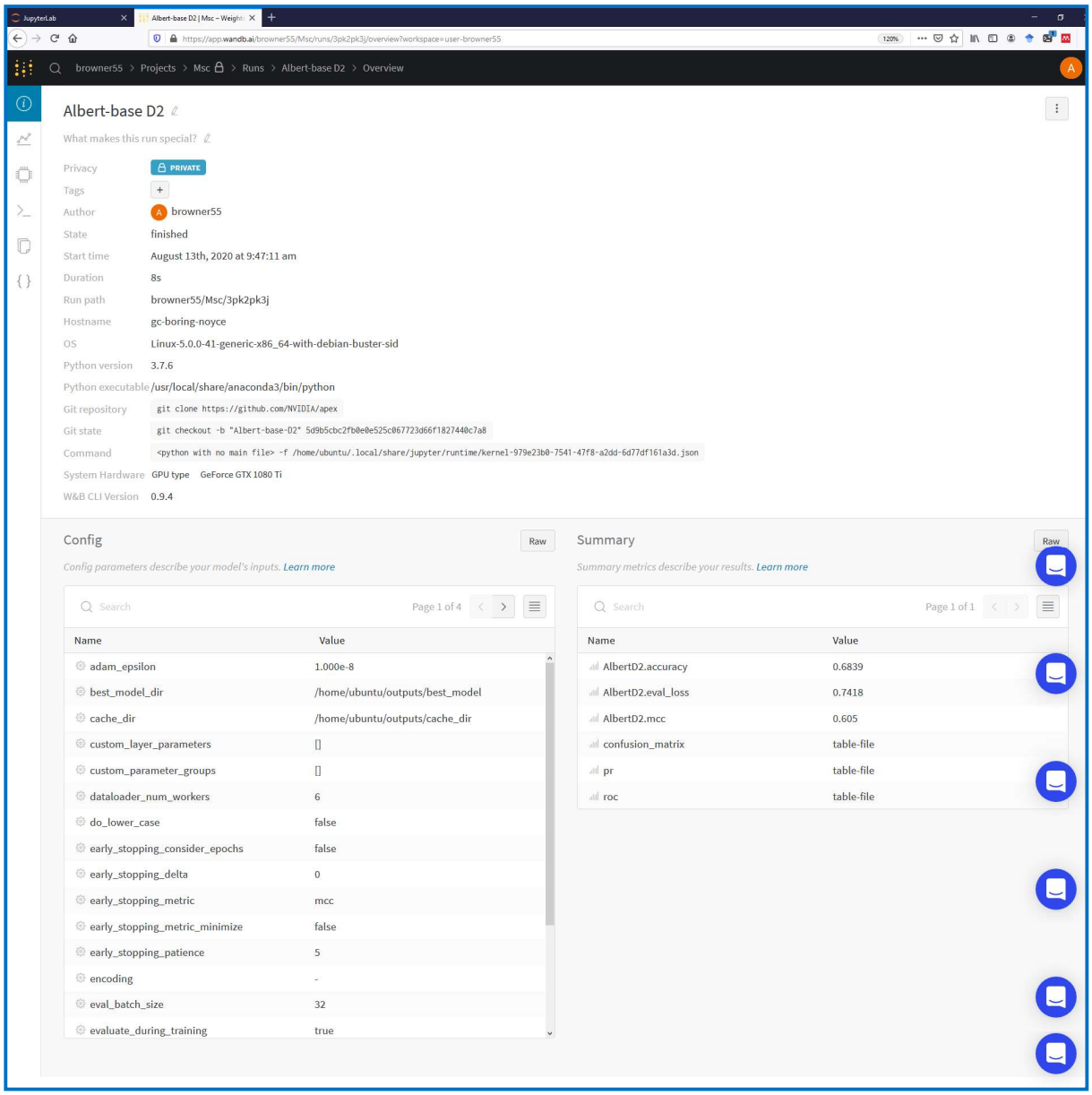


Figure 211: ALBERT-base D2 Results

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.overwrite_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

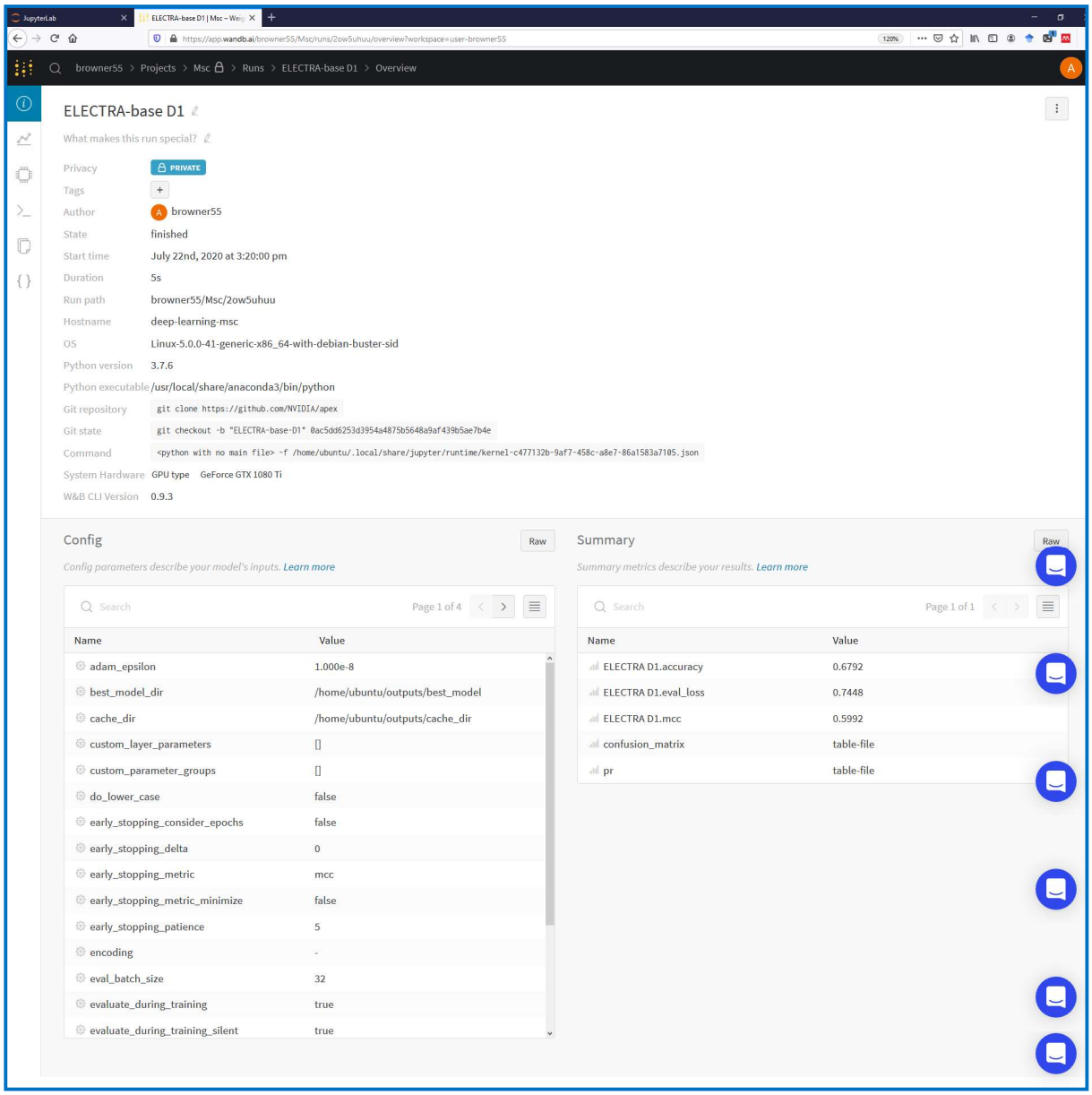


Figure 213: ELECTRA-base D1 Results

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.overwrite_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

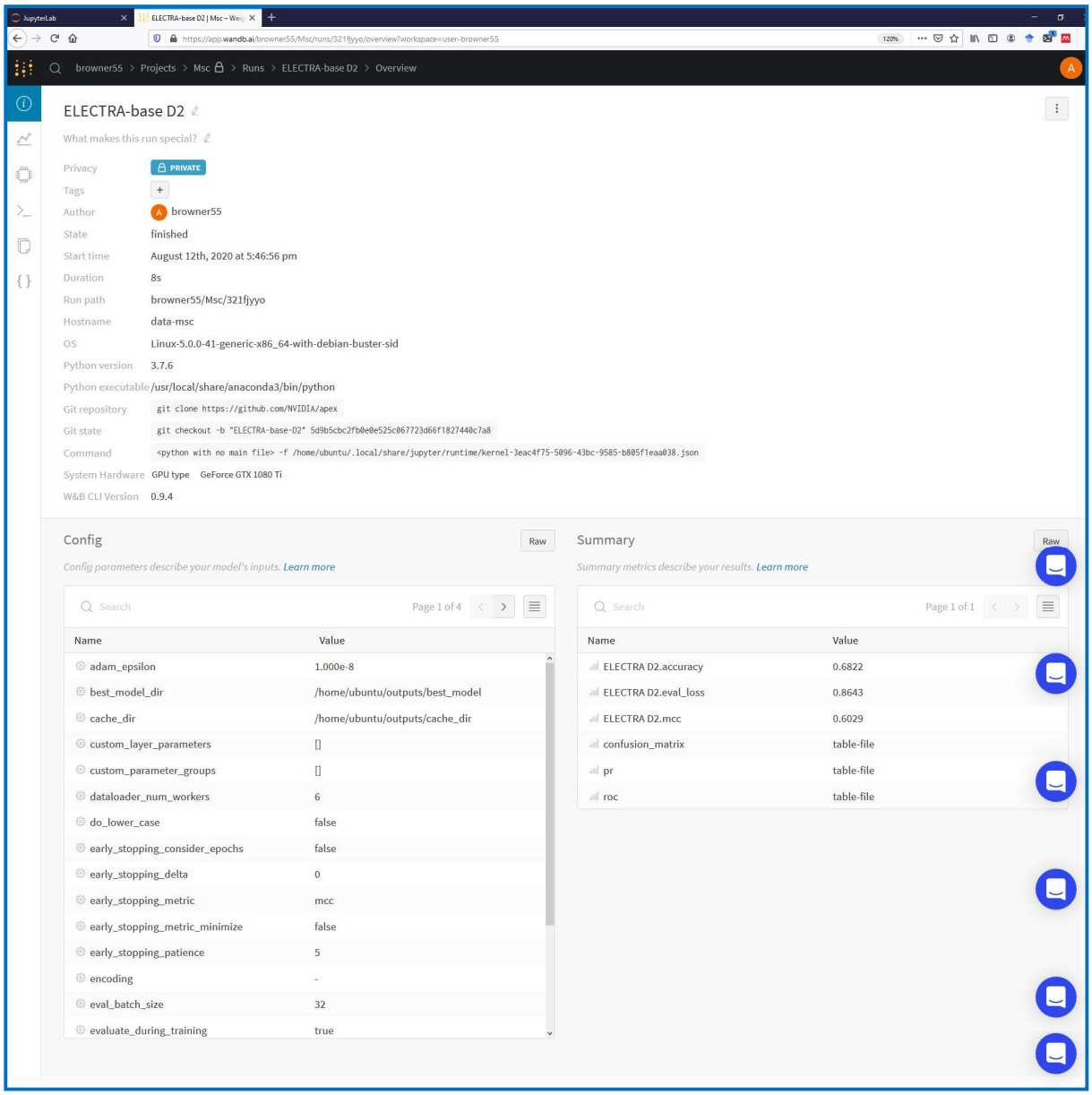


Figure 215: ELECTRA-base D2 Results

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.overwrite_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

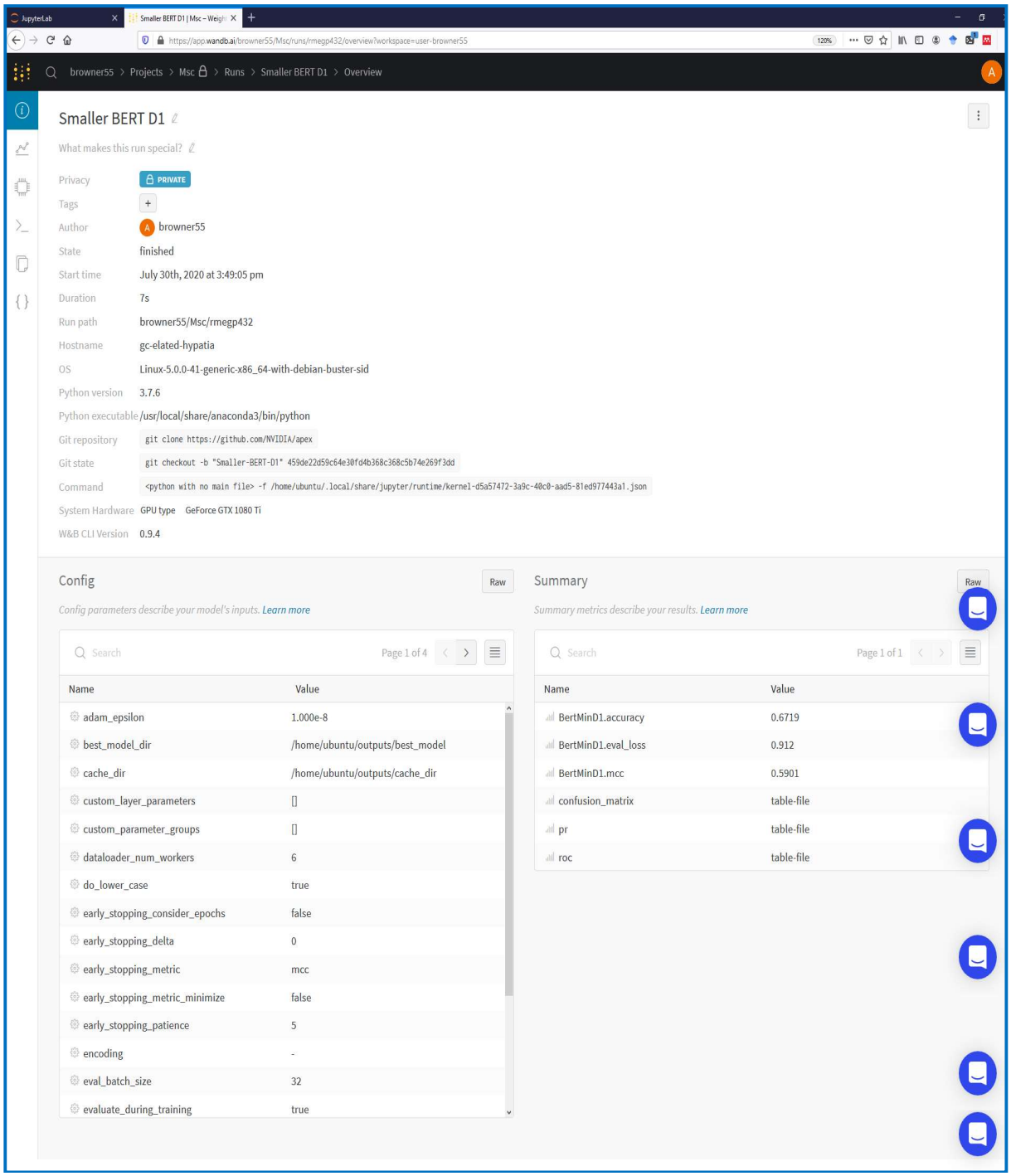


Figure 217: Smaller Bert D1 Results

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.overwrite_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

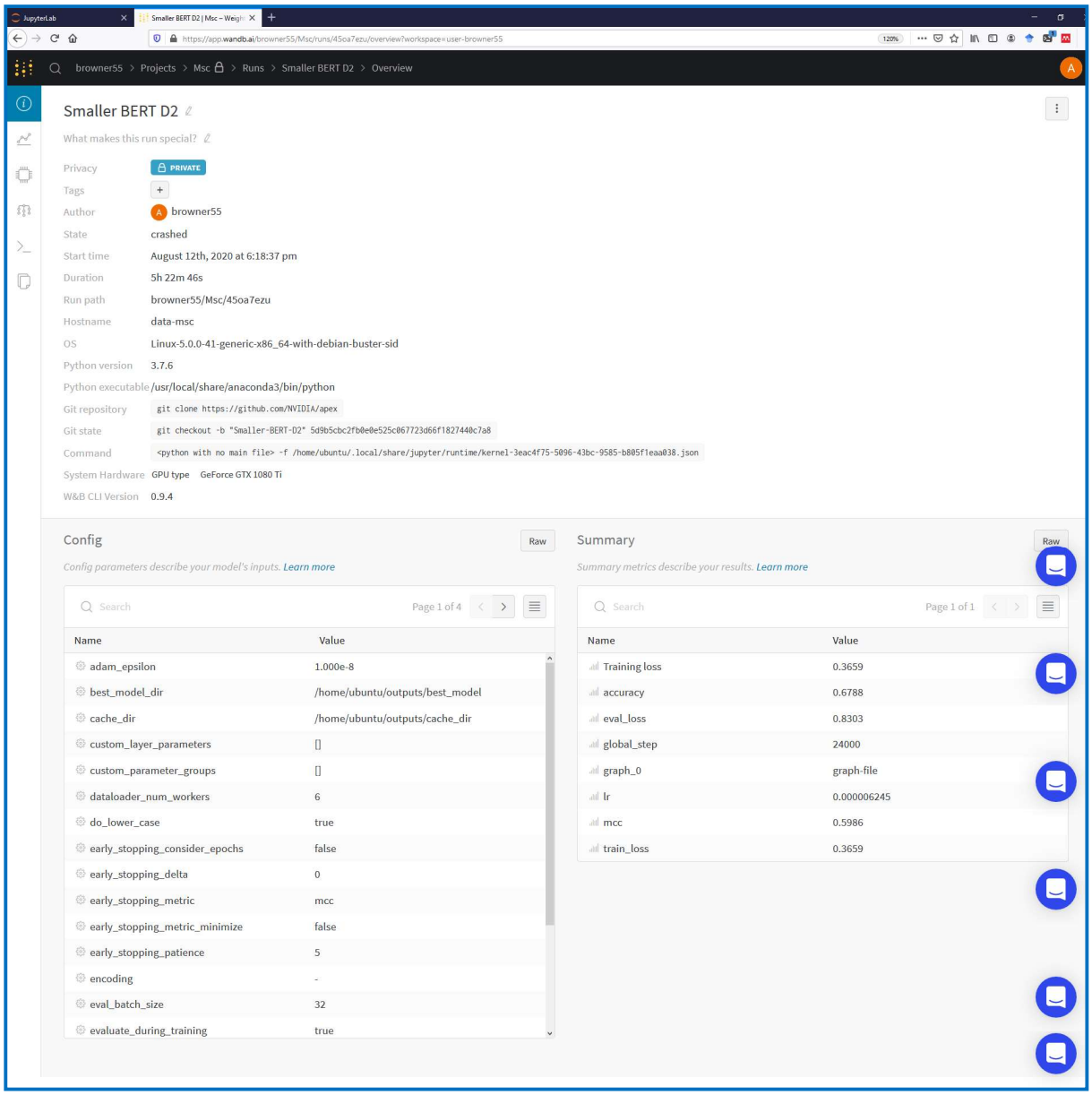


Figure 219: Smaller Bert D2 Results

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 here[15].

[15] https://app.wandb.ai/browner55/Msc?workspace=user-browner55