DSTS Assignment 1

Alan Gaugler

U885853

30/09/2023

Section A – Link to the Tableau Dashboard

My Tableau dashboard is on the following link:

https://public.tableau.com/app/profile/alan.gaugler/viz/DSTS_Ast1/Suburbs?publish=yes

Section B – Regression and Classification Tables

The results of the tables are included below. For more details of the creation of these tables please refer to the attached Jupyter Notebook for this assignment.

It must be noted that the data was data was standardized before modelling, so the numbers in the tables below are for standardized data. Standardization is important when the various features have different ranges. The modelling process will be biased more by the features with the largest variation. Standardization will give equal weight to all the features.

Table 1 - Regression Model Summary Table

Model	Mean Square Error	Mean Average Error	R Squared
Linear Regression 1	0.07471	0.19251	0.62083
SGD Regressor 2	0.07534	0.19471	0.61764
SGD Regressor 3	0.07478	0.19305	0.62045

Looking at the three metrics for the three models, the results are very similar. The initial linear regression model looks to be slightly better than the Stochastic Gradient Descent models although the margin is extremely slight. Model SGDR3 is slightly better than SGDR2 which is expected as its best hyperparameters were determined using the randomized search. It was the second-best performing model on this dataset. The linear regression model had the smallest MSE and MAE values and the highest R^2 value, winning on all three metrics.

It would have to be concluded that all models are very similar but Linear Regression performs the best for this dataset.

Table 2 - Classification Model Summary Table

Model	Accuracy	Precision	Recall	F1-Score
Logistic Regression 1	0.85178	0.85001	0.85178	0.85013
Support Vector 2	0.85456	0.86294	0.85456	0.85661
Random Forest 3	0.85943	0.85835	0.85943	0.85871
Multi-Layer Perceptron 4	0.85317	0.85140	0.85317	0.85136

Looking at the table, all four models performed very similarly, however the random forest classifier has the best overall accuracy of 85.94% followed by the support vector machine at 85.46%. The SVM has the highest precision, but the RFC won on all the other metrics.

All the metrics were calculated by using weighted average which takes into account the class imbalance. Looking below, class 1 (Low Rating) is 65% of all the dataset and class 2 (High Rating) is 34%. This is not a huge imbalance, but it is still better to use weighted average.

In this dataset, which is reasonably balanced, and precision and recall are not critical, overall accuracy would be the most important measure.

To conclude, all models perform very similar, but the random forest classifier is the best.

Section C – GitHub Commands and Procedure

The task is to deploy the code and relevant documents onto a GitLab repository. I created the GitLab account with a 30-day free trial approximately a month ago. This free trial will expire before this assignment is marked. I have spoken with Dr Ibrahim Radwan regarding this, and he has informed us that it is OK to upload the assignment on a GitHub repository instead.

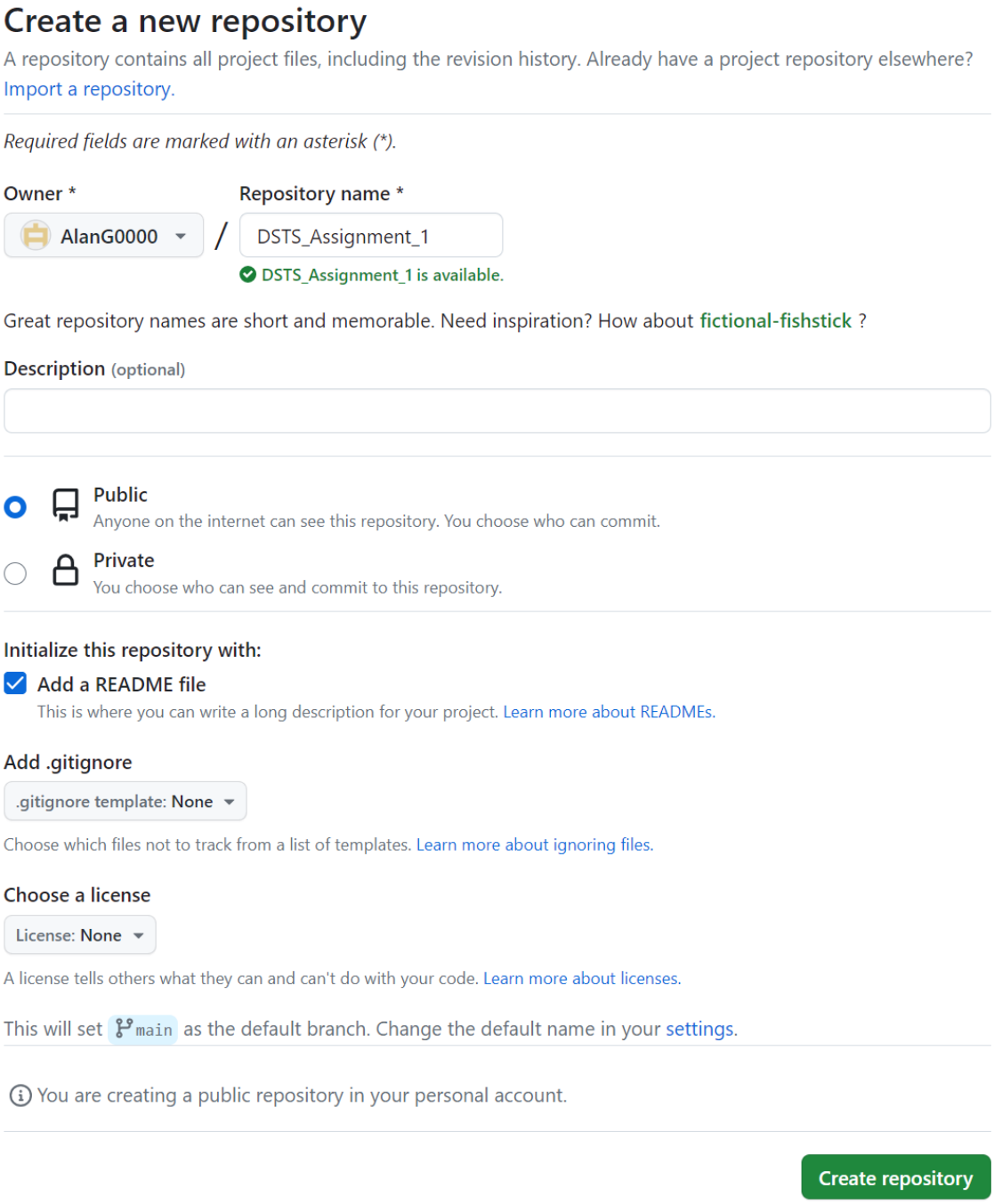
I will document the entire procedure involved in doing this, including all the required Git commands.

Part 1 – Create a GitHub Repository

I already have a GitHub account, so I did not have to create a new one.

I have signed in and created a new repository called “DSTS_Assignment_1”

It is created with a readme file, which contains a brief description of the deliverables and the assignment. This repository is made Public for simplicity.



Part 2 – Connect the Local Files with the Repository

A command prompt was opened in my Windows 11 PC by typing ‘cmd’ in the search slot.

I changed to the directory containing my assignment files,

```
Microsoft Windows [Version 10.0.22621.2361]
(c) Microsoft Corporation. All rights reserved.

C:\Users\alang>cd c:\DSTS\Ast 1

c:\DSTS\Ast 1>
```

Check that the files are in the directory:

```
c:\DSTS\Ast 1>dir
 Volume in drive C is OS
 Volume Serial Number is 3435-6083

 Directory of c:\DSTS\Ast 1

28/09/2023  03:06 PM    <DIR>          .
28/09/2023  03:06 PM    <DIR>          ..
28/09/2023  02:59 PM            17,185 DSTS Assignment 1 xx.docx
28/09/2023  02:43 PM         4,273,610 DSTS_Ast1_V4.ipynb
               2 File(s)      4,290,795 bytes
               2 Dir(s)  507,901,865,984 bytes free

c:\DSTS\Ast 1>
```

Initialize the Git repository in the project folder.

```
c:\DSTS\Ast 1>git init
Initialized empty Git repository in C:/DSTS/Ast 1/.git/

c:\DSTS\Ast 1>
```

In GitHub the main branch is called ‘main’, not ‘master’. I want to ensure the same happens locally, to avoid problems when pushing.

```
c:\DSTS\Ast 1>git config --global init.defaultBranch main

c:\DSTS\Ast 1>
```

Initialize again.git init

```
c:\DSTS\Ast 1>git init
Reinitialized existing Git repository in C:/DSTS/Ast 1/.git/
```

Connect the local repository to the GitHub repository that I created previously. The link for the repository I created is:

https://github.com/AlanG0000/DSTS_Assignment_1

```
c:\DSTS\Ast 1>git remote add origin https://github.com/AlanG0000/DSTS_Assignment_1.git

c:\DSTS\Ast 1>
```

Confirm that the connection has been established.

```
c:\DSTS\Ast 1>git remote -v
origin  https://github.com/AlanG0000/DSTS_Assignment_1.git (fetch)
origin  https://github.com/AlanG0000/DSTS_Assignment_1.git (push)

c:\DSTS\Ast 1>
```

Part 3 – Commit and Push the Files

Add all the files in the folder to the staging area. The '.' means all files in the command **git add .**

```
c:\DSTS\Ast 1>git add .
warning: in the working copy of 'DSTS_Ast1_V4.ipynb',

c:\DSTS\Ast 1>git status
On branch main

No commits yet

Changes to be committed:
  (use "git rm --cached <file>..." to unstage)
        new file:   DSTS Assignment 1 V1.docx
        new file:   DSTS_Ast1_V4.ipynb
```

Git status above confirms the files are now in the staging area.

The files will now be committed with a brief note. Verify the

```
c:\DSTS\Ast 1>git commit -m "Upload Assignment 1"
[main (root-commit) d5cf710] Upload Assignment 1
 2 files changed, 10964 insertions(+)
 create mode 100644 DSTS Assignment 1 V1.docx
 create mode 100644 DSTS_Ast1_V4.ipynb
```

Verify the commit.

```
c:\DSTS\Ast 1>git log
commit d5cf710d63cbe0b177fd1d51a353bf13e2eaebfb (HEAD -> main)
Author: alang0000 <alan.g100@yahoo.com>
Date:   Thu Sep 28 16:28:49 2023 +1000

    Upload Assignment 1
```

Push the committed project files to the GitHub repository.

This appears to have been performed successfully. Several lines end with "done"

I will also check on GitHub.

```
c:\DSTS\Ast 1>git push -u origin main
```

```
Enumerating objects: 4, done.
Counting objects: 100% (4/4), done.
Delta compression using up to 32 threads
Compressing objects: 100% (4/4), done.
Writing objects: 100% (4/4), 2.77 MiB | 467.00 KiB/s, done.
Total 4 (delta 0), reused 0 (delta 0), pack-reused 0
To https://github.com/AlanG0000/DSTS_Assignment_1.git
 + 40889fd...4d8b3f4 main -> main (forced update)
```

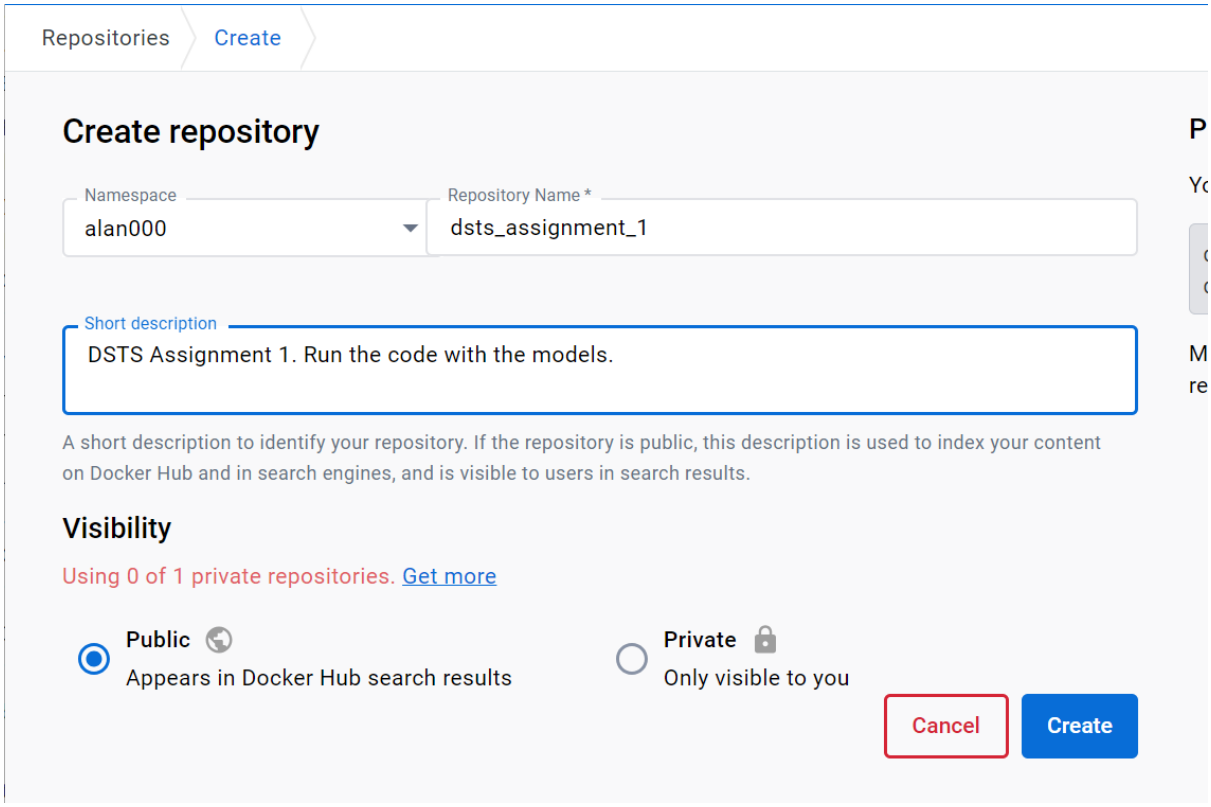
Check the commit history to verify.

```
c:\DSTS\Ast 1>git log
commit 4d8b3f4110510ce01c6bd52224760b8c88574a9c (HEAD -> main, origin/main)
Author: alang0000 <alan.g100@yahoo.com>
Date:   Thu Sep 28 16:37:04 2023 +1000
```

The files are now on my GitHub repository.

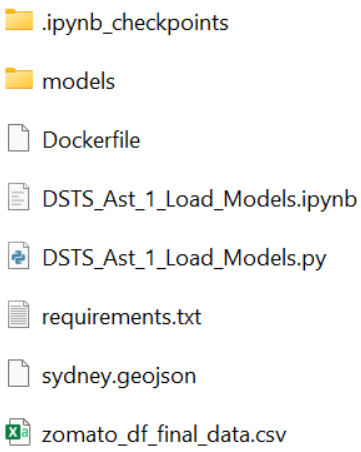
Section D – Docker Commands and Procedure

Step 1: Create a Docker Hub Repository



Step 2:

First all the project files were placed in the project's working directory. A sub directory called models was made where all the models created in the Jupyter Notebook would be exported. The folder consists of the following files:



The original csv and geojson files, the Jupyter notebook and other files which will be explained as follows:

Step 3:

- The Jupyter notebook was converted to a .py file so it can be executed as a python script.
- A requirements.txt file was made which lists all the python dependencies, or all the versions of the libraries that need to be installed when running the python script of the project. The contents of this file are as follows:

```
pandas==1.3.5
numpy==1.21.2
matplotlib==3.4.3
seaborn==0.11.2
scikit-learn==1.0.2
joblib==1.1.1
geopandas==0.9.0
shapely==1.8.4
```

- A docker file is then created, which is a brief script the contains the instructions required to automatically create a Docker image. This image will be used to create a Docker container.
  The FROM command has the required version of Python.
  LABEL is metadata.
  COPY copies the files from the local syste, into the mentioned directory. In this case the root directory of the project. The requirements.txt file is included here.
  WORKDIR sets the working directory within the container. In this case the root directory.
  RUN will install the required dependencies from the requirements.txt file.
  CMD states which commands need to be executed upon opening the container from the built image.

Contents of the Dockerfile

```
FROM python:3.7.16-slim
LABEL maintainer="Alan Gaugler" \
      description="Docker image for Assignment 1."
COPY . /
WORKDIR /
RUN pip install --no-cache-dir -r requirements.txt
CMD ["python", "DSTS_Ast_1_Load_Models.py"]
```

Step 4: Build the Image

Navigate to the project directory and run the command:

docker build -t dsts_ast_1_image .

The command builds the docker image.

Test the image locally.

All print statements used in the original code will be displayed. I have added additional statements at the end so that the summaries of the regression and classification models can be seen.

```
Summary of the Regression Models' Accuracy
                Model      MSE      MAE      R^2
0  Linear Regression 1  0.07471  0.19251  0.62083
1      SGD Regressor 2  0.07534  0.19471  0.61764
2      SGD Regressor 3  0.07478  0.19305  0.62045


Summary of the Classification Models' Accuracy
     Model  Accuracy  Precision (weighted)  Recall (weighted)  \
0  LogReg1   0.85177               0.85001            0.85177
1     SVC2   0.85456               0.86294            0.85456
2     RFC3   0.85943               0.85835            0.85943
3     MLP4   0.85317               0.85140            0.85317

   F1-Score (weighted)
0              0.85013
1              0.85661
2              0.85871
3              0.85136
```

The image has been run successfully locally.

Step 5: Push the Image to the Repository

Log in, Tag the Image and then push it to the Docker repository.

```
c:\Users\alang\DSTS\Assignment_1>docker login
Authenticating with existing credentials...
Login Succeeded

c:\Users\alang\DSTS\Assignment_1>docker tag dsts_ast_1_image alan000/dsts_assignment_1:v1

c:\Users\alang\DSTS\Assignment_1>docker push alan000/dsts_assignment_1:v1
The push refers to repository [docker.io/alan000/dsts_assignment_1]
beaa405aa745: Pushed
2de111d46bd2: Pushed
69462bf1e6c3: Mounted from library/python
5fbeb7afacaf: Mounted from library/python
962ae0fa6889: Mounted from library/python
6669ab6e06c6: Mounted from library/python
8cbe4b54fa88: Mounted from library/python
v1: digest: sha256:d2759ab37f6116e56db197e10195ed0a00ede5e959338d98afded23741723aa1 size: 1795
```

The image is now successfully uploaded at the following link:

https://hub.docker.com/r/alan000/dsts_assignment_1/tags

The image can be pulled as follows:
docker pull alan000/dsts_assignment_1:v1

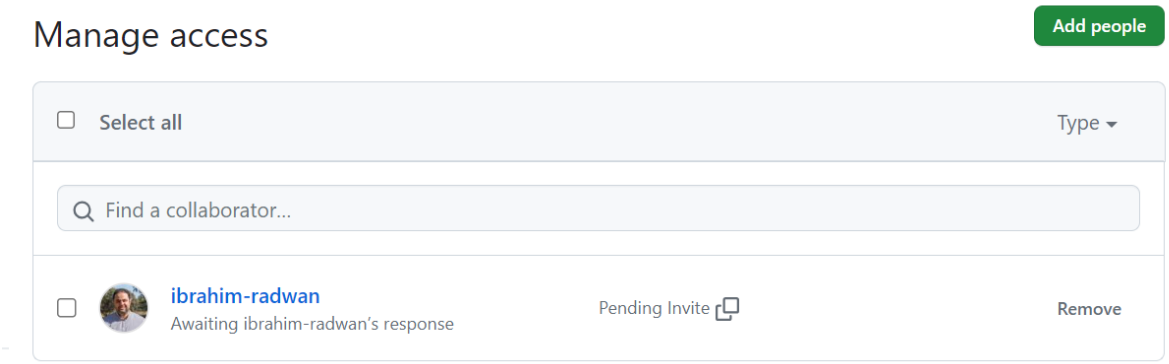
Section E – GitHub Link to the Source Code

The link to my source code is:

https://github.com/AlanG0000/DSTS_Assignment_1

Callaborator access has been granted to Ibrahim at his email address:

radwanebrahim@gmail.com



Section F – Docker Hub Link to the Docker Image

The link to the Docker image I have deployed on Docker Hub is as follows:

https://hub.docker.com/r/alan000/dsts_assignment_1/tags

v2 is the latest version.