|  |  |
| --- | --- |
| Description: 18897 CIC A4 Portrait WordTemp_cropped.jpg | ASSIGNMENT COVER SHEET |
|  | |
|  | |
| |  |  | | --- | --- | | SUBJECT NUMBER & NAME | Advanced Data Science for Innovation | | NAME OF STUDENT  (PRINT CLEARLY - SURNAME, FIRST NAME) | Wenying Wu | | STUDENT ID NUMBER | 14007025 | | STUDENT EMAIL | Wenying.Wu-1@student.uts.edu.au | | STUDENT CONTACT NUMBER |  | | DUE DATE | 15/03/2022 23:59 pm | | ASSESSMENT ITEM NUMBER/TITLE | Assignment 2 | | * I confirm that the work submitted conforms with the university’s guidelines on academic integrity.   *Refer to the UTS policy on ‘Advice to Students on Good Academic Practice’*: <http://www.gsu.uts.edu.au/policies/academicpractice.html>   * I am aware of the penalties for plagiarism. This assignment is my own work and I have not handed in this assignment (either part or completely) for assessment in another subject. * If this assignment is submitted after the due date I understand that it will incur a penalty for lateness unless I have previously had an extension of time approved and have attached the written confirmation of this extension.   Please provide details of extensions granted here if applicable \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_  \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ | | | |

# Introduction

This report presents various aspects of the project to deploy a Neural Network Model into production, including neural networks architecture, the performance achieved, API structure, instructions for running predictions. The Neural Network Model is trained using the dataset (<https://drive.google.com/file/d/1vYyJL_IB6KjKCxuk9kg4vIMPGTtoX8Ek/view?usp=sharing>) to accurately predict a type of beer based on some users’ rating criteria such as appearance, aroma, palate, or taste.

Link to project GitHub repository: <https://github.com/WenyingWu-1/ADSI_AST2>

Link to API GitHub repository: <https://github.com/WenyingWu-1/adsi_ast2_api>

Link to API URL: <https://adsi-ast2-14007025.herokuapp.com/docs/>

# Data Preparation

Detailed data preparation steps are shown *in the “Data Preparation.ipynb”* notebook, and here is just a summary. The original dataset contains 1,586,614 rows and 13 columns, namely *brewery\_id, brewery\_name, review\_time, review\_overall, review\_aroma, review\_appearance, review\_profilename, beer\_style, review\_palate, review\_taste, beer\_name, beer\_abv, beer\_beerid.*

There are 104 *beer\_style* (the target variable) in this dataset, and the distribution is highly imbalanced; the top 30 are shown in Figure 1, full figure in the notebook. The most occurred *beer\_type* is “American IPA” with 117,586 (7.411%), and “Happoshu” is the *beer\_type* with the least occurrence 241 (0.015%).

#### Chart Description automatically generated with low confidence

#### Figure 1. the number of beer of *beer\_*style (top 30)

There are 348 null values in *review\_profilename* and 67,785 null values in *beer \_abv*. Considering the dataset contains more than 15 million data, even if combining these 2 numbers, there are only 4.27% rows with a null value. Therefore, these rows are decided to be dropped.

In the assignment brief, the expected inputs are *brewery\_name*, *review\_aroma*, *review\_appearance*, *review\_palate*, *review\_taste,* and *beer\_abv.* I decided not to add any feature engineering in this model as the inputs are all numeric reviews except for *brewery\_name,* cannot find any hidden feature in this dataset. Therefore, the last step to clean the dataset is to drop unused columns: *brewery\_id, review\_time, review\_overall, review\_profilename, beer\_name, beer\_beerid*. This action produced a cleaned dataset with 1,518,478 rows and 7 columns.

To make the neural network model perform better, I encoded the *brewery\_name* and *beer\_style* columns using label encoder from sklearn and then standardized all columns except for *beer\_style* using RobustScaler from sklearn. The choice of RobustScaler came from trial and error among StandardScaler, MinMaxScaler, and RobustScaler.

The last data preparation step is to separate *beer\_style* then separate the cleaned dataset to train, validate, and test sets.

# The Neural Network Model

After a few rounds of trial and error on different architectures of neural network model (experiment log shown in “*Neural Network Model.ipynb”* and detailed trails are recorded in ‘Draft’ folder inside ‘notebook’ folder), the final neural network model is a 7-layer (1 input layer, 1 output layer, and 5 hidden layers) feed-forward neural network with dropout and batch-norm. The number of neurons in each layer is shown in below list:

1. In 6, out 2048
2. In 2048, out 1512
3. In 1512, out 1024
4. In 1024, out 512
5. In 512, out 256
6. In 256, out 128
7. In 128, out 104

The 6 in the first layer is the number of input and 104 in the last layer is the number of classifications (*beer\_type*).

This model achieves a decent accuracy score of 39.8% on the train set, 47.7% on the validation set and 47.5% on the test set. I have trained more complex models and achieved higher accuracy on these datasets. Still, I decided to use this model as the final model because of the size limits on Heroku (the deployment platform). Please see *“Fast API & Heroku Deployment.ipynb”* for detail.

# API Structure

FastAPI is used for this project, considering it is easy and fast to use. The API includes the below endpoints:

1. '/'(GET): "Displaying a brief description of the project objectives, list of endpoints, expected input parameters and output format of the model, link to the GitHub repo related to this project"
2. '/health/' (GET): "Returning status code 200 with a string with a welcome message"
3. '/brewer/'(GET): "Returning all available brewers"
4. '/brewer/valid'(POST): "Check for single *brewery\_name* validity"
5. '/beer/type/'(POST): "Returning prediction for a single input only"
6. '/beers/type/'(POST): "Returning predictions for multiple inputs"
7. '/model/architecture/' (GET): "Displaying the architecture of your Neural Networks (listing of all layers with their types)"
8. '/docs/'(GET): "Go to the docs page and use the app!"

More detailed steps and explanations can be found in the Fast API & Heroku *“Deployment.ipynb”* notebook, this report will only summarize the core logic of this API. The main functions in this API to support predicting *beer\_type* based on users’ input are format\_features() and multi\_format\_features(). format\_features() are used for a single input, it first checks the validity of *brewery\_name*, returns False is *brewery\_name* is invalid, and the API will return a string telling the user the *brewery\_name* is not valid. If the validity check is passed, this function will transform inputs into Pandas dataframe, then encode brewery\_name and scale other numeric inputs using the loaded encoder and robust scaler. The function lastly returns a list of formatted model-ready inputs to the API. format\_features() use the same logic; the only difference is that it takes all inputs as a string and separates them into lists. It also performs another check on the length of each input field, if the lengths of each input field are not the same, API will return a string report to the user.

In addition, other than those endpoints in the Assignment brief, 2 more endpoints are added for ease of use: '/brewer/'(GET) and '/brewer/valid'(POST). '/brewer/'(GET) returns a dictionary to the user indicating all valid inputs for the *brewery\_name* field. While '/brewer/valid'(POST) expects a single string input, the corresponding string will be returned to the user regarding whether the input is checked as valid *brewery\_name*.

# Instruction for running predictions

After the API is finished, the model can be deployed to Heroku (steps shown in *“Fast API & Heroku Deployment.ipynb”* notebook). This report only shows the steps to run predictions on the deployed app.

1. Go to this link: <https://adsi-ast2-14007025.herokuapp.com/docs>
2. Click on the dropdown arrow of /brewer -> click on ‘try it out’-> click on ‘execute’-> scroll down to see all available brewer name.

Or you can go to the dropdown arrow of /brewer/valid -> click on ‘try it out’-> input the brewery name you want to use to predict -> click on ‘execute’-> scroll down to see whether your input is a valid brewer in this model.

1. Click on the dropdown arrow of /brewer -> click on ‘try it out’-> click on ‘execute’-> scroll down to see all available brewer name.
2. Click on the dropdown arrow of /beer/type or /beers/type -> click on ‘try it out’-> input your preferred brewery name and beer review -> click on ‘execute’-> scroll down to see the prediction. Figures 2 and 3 show an example input and output for /beers/type

Graphical user interface, application

Description automatically generatedFigure2 Example inputs

Background pattern, rectangle

Description automatically generated

Figure3 Example outputs

More detailed steps are shown in *“Fast API & Heroku Deployment.ipynb”.*

## Exceptions handled:

* '/brewer/valid'(POST):
  + 'Brewer not valid, please select another brewer...'
* '/beer/type/'(POST):
  + If brewery\_name is invalid: "Please check your brewery\_name input, it is probably wrong. Refer '/brewer/'(GET) for available brewer names or check via '/brewer/valid'(POST)..."
* '/beers/type/'(POST):
  + If brewery\_name is invalid: "Please check your brewery\_name input, it is probably wrong. Refer '/brewer/'(GET) for available brewer names or check via '/brewer/valid'(POST)..."
  + If fields' lengths are not identical or separation is incorrect: "Please ensure all blanks are inputted with the same number of inputs. And make sure a comma and a blank space separate each input: 'input\_1, input\_2'"

# Reflection and Conclusion

Though the model only achieved an overall accuracy of 47.5% on the test set, it is an acceptable result considering 104 classifications. It will be interesting to build some follow-up comparison models using other models like XGBoost and RandomForest model to compare the outcome. Furthermore, though I did not find a way to deploy my best model on Heroku, it is interesting to see that the final model with a 27MB state\_dict has similar predicting power compared to the best one with a 180MB state\_dict (47.5% vs 49.5% accuracy).

In addition, I found that problem-solving skill is essential. For example, I struggled to deploy the FastAPI on Heroku successfully, and I spent a long time searching on Google and YouTube to find out what step I did wrong. It was running smoothly on my local machine but kept showing Application errors after deployment on Heroku. Finally, I read the Heroku log carefully and found that the problem is originating from reading the saved neural network model. Then I went to the PyTorch official website to read how to save the model correctly and find out that saving the model’s state\_dict is a much more portable method compared to saving the entire model. The full document can be found on:

<https://pytorch.org/tutorials/beginner/saving_loading_models.html>.

In conclusion, this project is a valuable experience that let me practice the whole life cycle of a data science project, including neural network design, API design, and model deployment. It also strengthened my python coding skill and deepened my modeling knowledge.