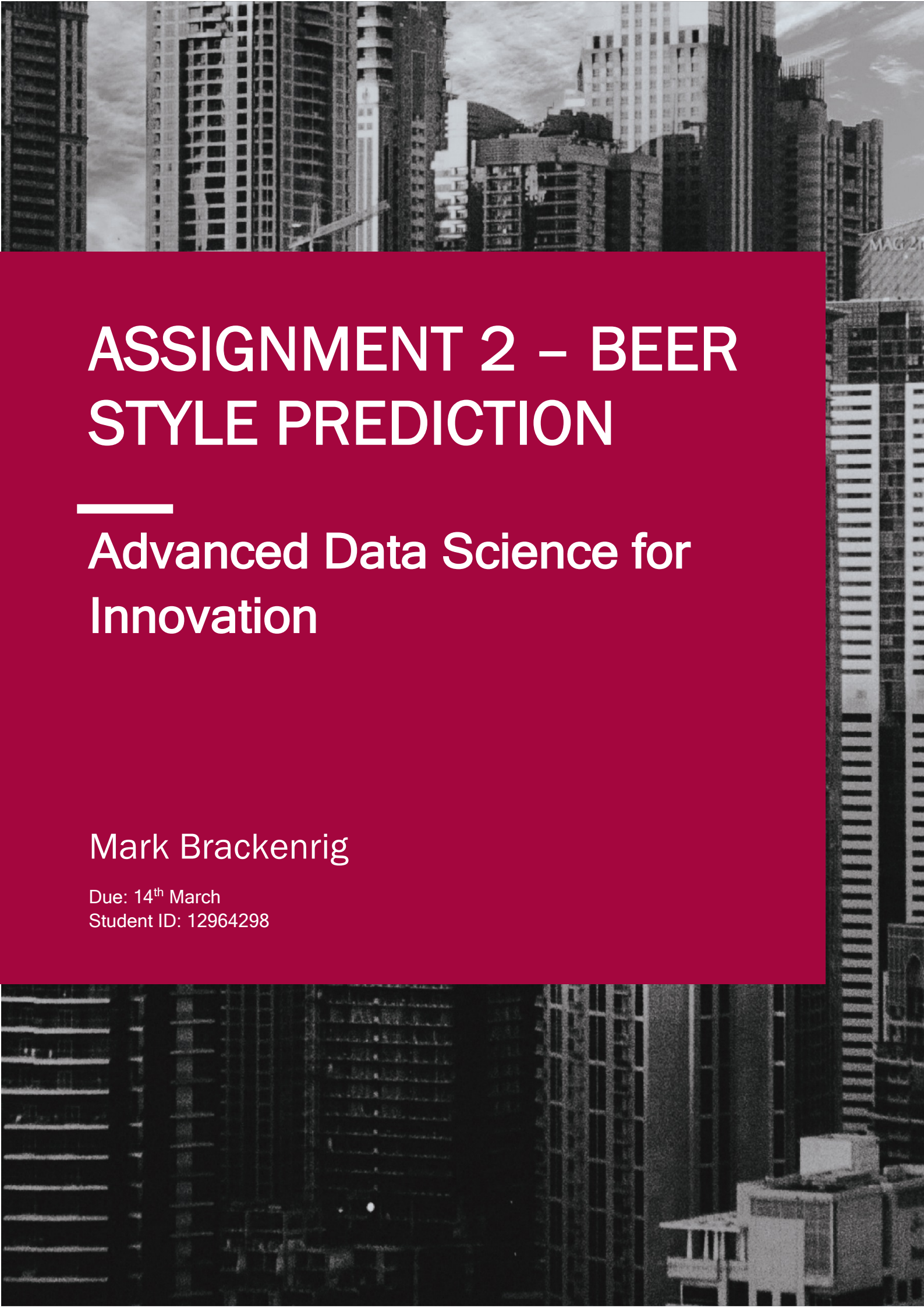

ASSIGNMENT 2 – BEER STYLE PREDICTION

Advanced Data Science for Innovation

Mark Brackenrig

Due: 14th March

Student ID: 12964298

Table of Contents

Business Understanding

The purpose of this project is to accurately predict the beer style of a particular beer based on a reviewers' rating of the beer. Fields include criteria such as appearance, aroma, palate or taste, as well as the name of the brewery.

The business has shown interest in making the model available for production. While no specific use case was included in the brief, this could be used to automatically fill in the 'beer style' field of a review form, to assist the reviewer in generating their review. The business has also insisted that the model is a neural network.

Data Understanding

The data was sourced from BeerAdvocates, a large beer review website. The data contained 1,586,614 records with 13 columns. The target variable was "beer_style" – which contained 104 unique styles of beer.

Only 5 fields were kept for modelling purposes – review_aroma, review_appearance, review_palate, review_taste and brewery_name. All fields except for brewery name were numeric, ranging from 1 to 5. There were 5742 unique breweries in the dataset.

Data Preparation

Due to the large volume of data, only 50% of the data was used for modelling. The target variable was encoded using the OrdinalEncoder method from sklearn. While the data is not ordinal, this function still index encodes the target variable. All numeric data was normalized using the StandardScaler function.

For the "brewery_name" field, only 2500 breweries were considered for encoding into the model. All other breweries were considered as 'other'. This was because of a data processing error when generating a one-hot encoded matrix for the brewery names. Generating a non-sparse matrix that size was too large for the hardware available.

However, this also had the added benefit of flexibility when strings were provided to the API. In the case where the argument passed to the API is unknown (i.e., a new brewery), the model will still be able to produce a prediction.

If time permitted, I would have refactored my code to include the 'fitting' of the one-hot encoder in the data generator to avoid the size issues and include all data in my model training. This would have involved building a custom data generator, rather than the out of the box pytorch data loader.

Modelling

Based on brief requirements, the model selected was a neural network. Some experiments were completed using different architectures, however most resulted in poor results. The best performing models were typically shallow networks.

Initially, I only extracted the top 1000 brewery names for modelling. Intuitively this made sense as this represented 93% of the brewery names in the dataset. In one of my experiments, I decided to increase the number of extracted brewery names to 2500, which led to an improvement in accuracy (21.0% vs 22.3%). This could be caused by smaller breweries having fewer types of beer in the dataset.

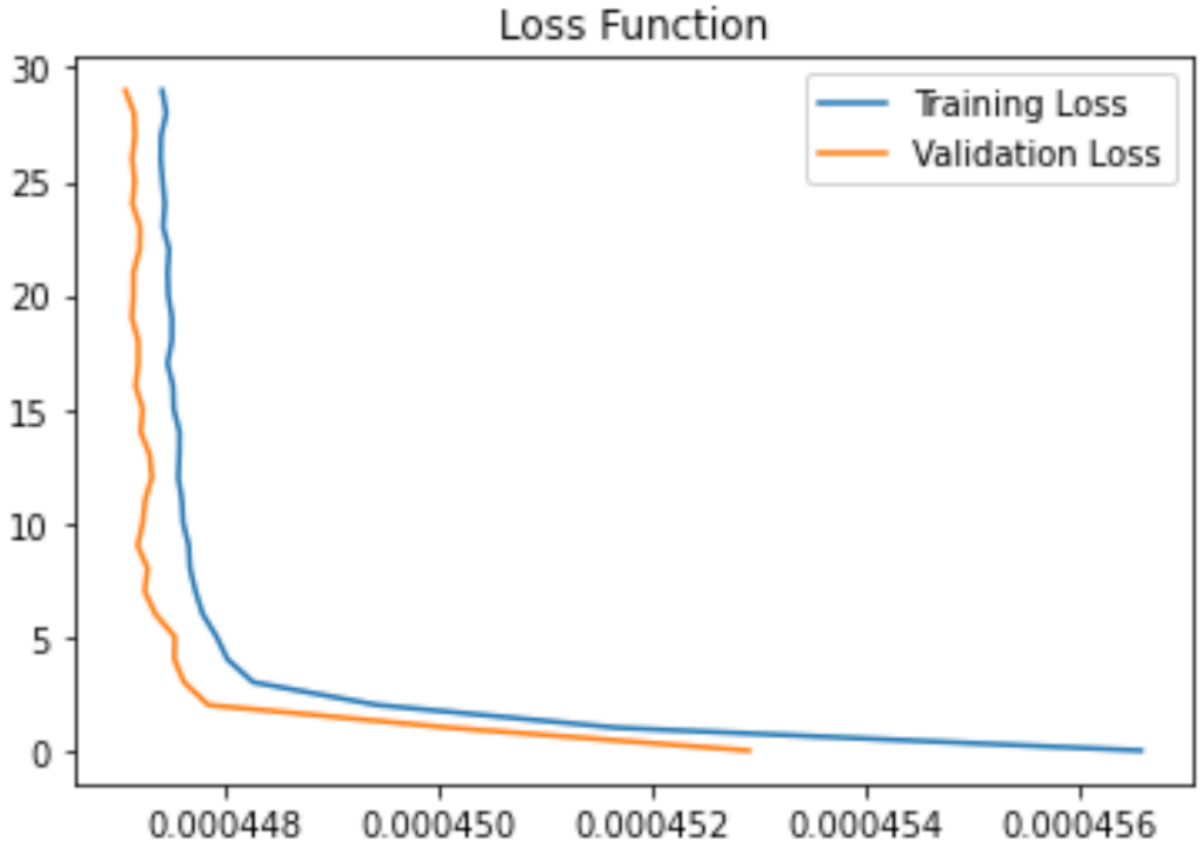

Figure 1: Loss function of second experiment

In my final model, I followed the same process as my previous experiment, however I trained the model for another 30 epochs, which further improved the accuracy of the model (23.7%).

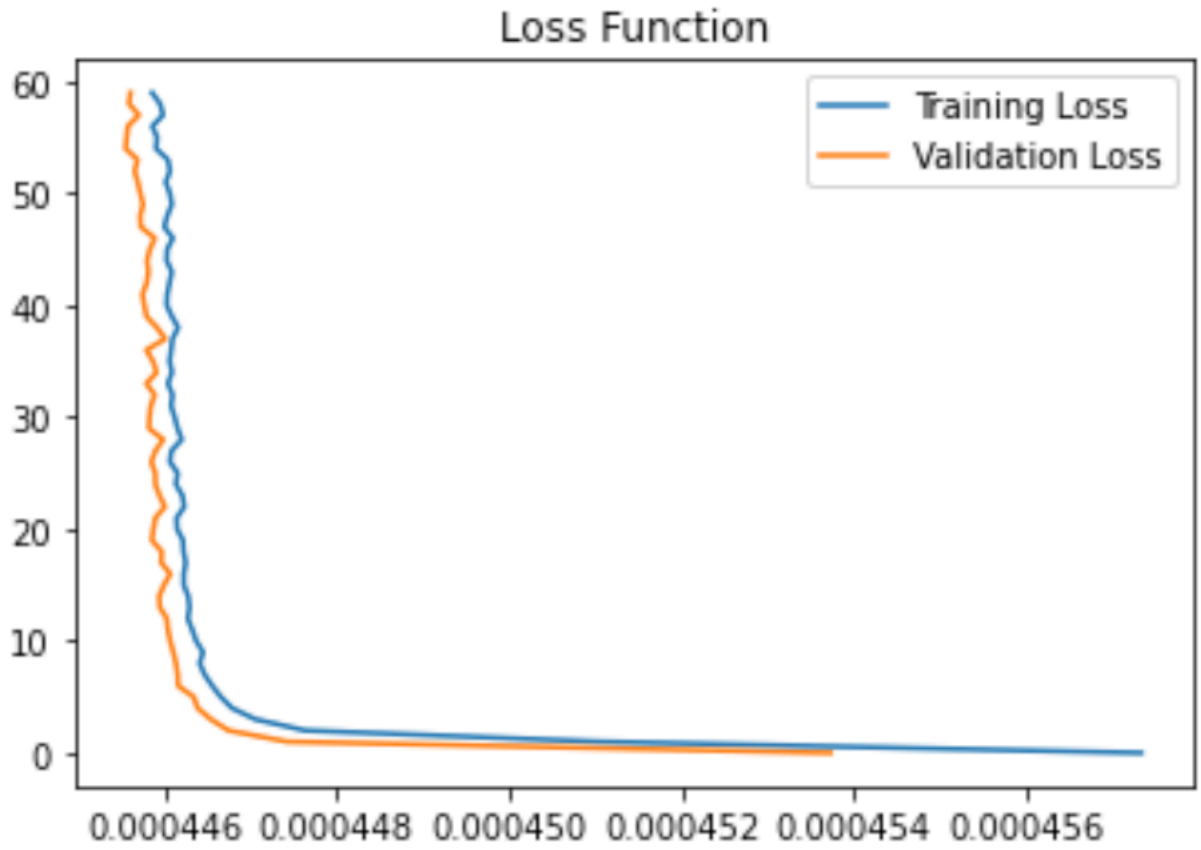

Figure 2: Loss function of third experiment

Due to time pressures, minimal hyperparameter tuning was completed. Throughout the modelling process, manual hyperparameter tuning was completed.

Architecture

Due to the limited data available, a shallow architecture was selected. This was to ensure that vanishing gradients would not be a problem. Also, I believe there is limited benefit in adding additional 'features' (via adding additional layers) with such as limited dataset.

The architecture consists of only 1 hidden layer and the output layer. Dropout regularization is applied to the hidden layer. The relu activation function was applied to the hidden layer, and a softmax activation was applied to the output layer. The size of the hidden layer was based on the number of classes. Through manual experimentation I found that increasing this number did not appear to have a major impact on accuracy of the model.

```
class PytorchMultiClass(nn.Module):
    def __init__(self, num_features,class_num):
        super(PytorchMultiClass, self).__init__()

        self.layer_1 = nn.Linear(num_features, 104)
        self.layer_out = nn.Linear(104,class_num)
        self.softmax = nn.Softmax(dim=1)

    def forward(self, x):
        x = F.dropout(F.relu(self.layer_1(x)), training=self.training)
        x = self.layer_out(x)
        return self.softmax(x)


model = PytorchMultiClass(num_features = X_train.shape[1],class_num= 104)
```

Figure 3: Architecture of Neural Network

The relu activation function was chosen due to its popularity in hidden layers. The softmax activation function was chosen to allow for multiclass prediction. Other activation functions could have been attempted, however due to limited time, I decided to focus on deploying the model, rather than optimising the model.

Evaluation

The evaluation metric used was accuracy. The Null model achieved an accuracy of 7.4% on the training set, whereas my final model achieved an accuracy of 23.9% on the training set, and 23.7% on the test set. While this is an improvement on the null model, I find it difficult that this accuracy could produce a meaningful.

One of the major downsides to this model is that it is only based on the brewery and the rating. Details about the beer, for example, the beers alcohol content were omitted from the model. When inspecting the confusion matrix, it appears that the model often confuses particular styles with their imperial (higher alcohol content) counter parts. For example, for the class “American IPA” had a recall of 0.61 (i.e., 61% of true class were predicted correctly) – however, “American Double / Imperial IPA” accounted for another 16.7% of predictions.

Similarly, “American Malt Liquor” is often confused with “American Adjunct Lager”, which is typically described as “[beers] which are made with ingredients and processes resembling those for American-style lagers”[1]. In addition, Belgian strong ales, “Dubbel”, “Tripel” and “Quadrupel” can all be used to describe trappiest style beers that vary in alcohol content.

Other model issues occur due to the structure of the data. The data contains ratings rather than descriptors. If you were to interpret, say “review_aroma” as the “strength” or “pleasantness” of the aroma, you can have two vastly different styles predicted to be similar. For example, ‘Bière de Champagne / Bière Brut’ is most commonly predicted as “American IPA”, while both having very distinct aromas and tastes, they are vastly different.

Finally, the granularity of the categories is contentious. There are many small, but highly related or near identical categories. For example, “Rauchbier” and “smoked beer” are arguably identical[2], and “Light Lager” and “American Adjunct Lager” could both be used to describe the same beer – 73% of ‘Light Lager’ observations in the test set were predicted to be American Adjunct Lagers.

Future Model improvements

Three main improvements would be made to the model. Firstly, inclusion of data that already exists in the dataset, in particular “beer_abv” should be included to differentiate styles based on alcohol content. I believe this would greatly improve the model as explained above.

Secondly, I would combine like styles in the model. Depending on the business application, differentiating between various beer styles may not be required. For example, “American IPA”, “English IPA”, and “Belgian IPA” could be described just as “IPA”.

[1] (Various, 2021)
[2] The literal translation of ‘Rauchbier” is “Smoked beer”

Finally, training the model on the entire dataset, using all brewery names in the encoder. This may lead to the model overfitting on the brewery name. However, some niche breweries specialise in specific styles; in these cases, the model would become more accurate.

For these reasons, it is my recommendation that the model would not be deployed to production. However, the business has insisted this model be deployed to production. While this model could give some utility to an end user, I believe these three areas should be resolved prior to the model being used for critical functionality.

Deployment

The model was deployed to a Heroku server on their free tier. The model was deployed using the 'FastAPI' web framework for python. The model was deployed using docker, to ensure safer and consistent deployments. Despite my objections to deploying the model to production above, a major advantage of deploying the model in its current state is the ability to iterate and deploy quickly. This would allow for the inclusion of new features (such as images of the beer), without needing to build out an entire deployment pipeline from scratch. The API was deployed at https://afternoon-ocean-26363.herokuapp.com/ and the repository can be found at: https://github.com/mark-brackenrig/assignment_2. The following endpoints and methods are available in the API (see next page).

Endpoint	Request Body	Description
'/' GET	NA	Returns the README of the project which includes all relevant data to use the API.
'/health/' GET	NA	Returns 'Hello World! App running.' And 200 response.
'beer/type/' POST	{"review_aroma": float, "review_appearance": float, "review_palate": float, "review_taste": float, "brewery_names": string}	Returns the predicted class name as a JSON object. For example: `{"class_name": "American Adjunct Lager"}`
'beers/type' POST	{ "review_aroma": list (float), "review_appearance": list (float), "review_palate": list (float), "review_taste": list (float), "brewery_names": list (string) }	Returns the predicted class names as a JSON object. For example: `{"class_name": ["American IPA", "American Adjunct Lager"]}`
'/model/architecture' POST		Returns the model architecture. Expected output: `{"layer_1":"Linear(in_features=2504, out_features=104, bias=True)", "activation":"relu", "regularisation":"dropout", "layer_out":"Linear(in_features=104, out_features=104, bias=True)", "softmax":"Softmax(dim=1)"}`
'/docs' GET	NA	API documentation as provided by FastAPI.

Bibliography

Various. (2021, March 14). *Malt Liquor*. Retrieved from Wikipedia: https://en.wikipedia.org/wiki/Malt_liquor