Predicting Student’s Weekend Alcohol Consumption

Gustavo De Souza Mendes†
Computer Science
University of Cape Town
AMCGUS001@myuct.ac.za

Dino Bossi
Computer Science
University of Cape Town
BSSDIN001@myuct.ac.za

ABSTRACT

This paper outlines the methodology of designing, implementing, training and evaluating a neural network. The model aims to predict alcohol consumption in students and has potential benefits for avoiding and managing alcohol abuse. In addition, the ethical implications of conducting such research are reviewed and considered.

1 Problem Formulation

For this project we chose a dataset obtained from Kaggle[1]. The dataset contains the results of a survey of secondary school students. The survey captured information about their social, personal, and academic lives, such as their gender, family size, and study time. The dataset also details the student’s alcohol consumption, and we chose this as the focus of this project.

We aimed to use a neural network to predict a student’s weekend alcohol consumption, using the remaining 32 features of the dataset as inputs for the model. The output of the prediction is a score from 1 to 5, indicating the level of weekend alcohol consumption. Thus, this problem can be defined as a multi-class classification problem.

Based on several underlying reasons, we believe that this is a moderately difficult problem to solve. First and foremost, the dataset we are using is not very large and only contains 649 data entries. This threatens the generalizability and overall accuracy of the model, as it can only be trained on a small amount of data. Secondly, the data is complex as it has a high number of input features. Finally, the project team is not knowledgeable in the domain of adolescent psychology and lifestyle, and the relationship between the two. As such, we cannot be certain about whether the features we have available in the dataset are adequate indicators of a student’s weekend alcohol consumption. The combination of these three factors compound the difficulty of the problem, however we believe that it presents a challenging, but interesting application of neural networks and artificial intelligence.

The insight gained from this project has several potential uses, both in terms of prediction of alcohol consumption and determining the impact of social and academic aspects on alcohol consumption. If the model could adequately predict the potential for alcohol abuse (i.e. high weekend alcohol consumption predictions), it could prove useful in determining which students require preventative intervention to reduce their risk of future alcohol abuse. The project’s outcome could also provide greater insight into the effect of a student’s ‘home-life’ on their academic performance, allowing support staff to preemptively identify which students require additional support. Finally, the project could further the understanding of important features in alcohol and other drug abuse predictions.

As artificial intelligence is increasingly integrated into our technological systems, we must strive to continually evaluate its ethical implications. We will discuss several ethical considerations relevant to this project.

The first consideration is about the generalizability of the model we create and remaining mindful of its limitations. As we are utilising a dataset that is relatively small and contextually specific (geographically, culturally, societally, etc.), it is possible that our model reflects these specificities. In addition, the data is obtained from surveys which are inherently subjective. This subjectivity further impacts the validity of the data, and the model created from it. We should not assert that the model is generalisable to any context and remain cautious of its predictions, as their implications could directly impact human lives.

As this project involves the creation of a predictive model, we need to remain mindful of the potential for exploitation. Students are a particularly vulnerable population: they are young and impressionable, are not legal adults and are often not financially independent and sustainable making them susceptible to corporate marketing. This model could, for example, be used to determine which students would respond ‘positively’ to alcohol advertisements and guide the advertising campaigns of such companies. This would have a negative effect on the targeted students, and ultimately for the community and society in which they live.

2 Baseline

To ensure that the best baseline was chosen for this project, both Dino [7] and Gustavo [9] created a baseline model separately. Both of the models [7, 9] already contained complex functionality (such as dropout, batch normalisation, etc.), which was removed for our baselines. The baseline model that was chosen was a Feed-Forward Neural Network, with minimal hyperparameter tuning

[1] https://www.kaggle.com/uciml/student-alcohol-consumption

and a simplified architecture. Through both team members creating a baseline model for the project, we gained a better understanding of the implementation tools (PyTorch[2], MatPlotLib, etc.) as well as experience in neural network implementations using PyTorch. Importantly, data pre-processing was implemented as an integral part of the baseline and the project. The code for our baseline model can be found in Baseline.py.

Architecture and Tuning:

Activation Function	ReLU
Loss Function	Cross Entropy Loss
Optimiser	Adam
Training Epochs	300
Data Split	Train:72% Validation:8% Test: 20%
Batch Size	16
Learning Rate	0.0007
Number of Hidden Layers	1
Hidden Layers: Inputs and Outputs	1: (512) => (256)
Batch Normalisation	Input Layer: 512, Hidden Layer 1: 256
Early Stopping Patience	-
Drop Out Rate	-
Last Epoch Information	Epoch 300: | Train Loss: 0.04254 | Val Loss: 2.83884 | Train Acc: 100.000| Val Acc: 30.769
Test Accuracy	36%

The baseline performance is reported in Appendix A, with an accuracy of 36% and very high validation loss, and the training accuracy converging to 100% very quickly.

3 Model Design

To evaluate the multi-class classification problem of student's weekend alcohol consumption, we implemented a feed-forward neural network. The reason why we chose this architecture was due to the high dimensionality of our dataset, having 32 features, and these features most likely have a complex relationship regarding the output. The feed-forward network is optimal for solving problems with these characteristics and is recommended for multi-class classification problems [1]

3.1 Data Preparation

The first step of implementing a neural network to make predictions is to perform preprocessing of the data being used. Our data has 33 columns in total, the column we chose to predict, our output column, is the Weekend alcohol consumption (Walc).

[2] https://pytorch.org/docs

The steps taken to ensure the data was prepared for use in the neural network were as follows: formatting the data to be valid as input, encoding the output values so they are supported by the PyTorch Library, splitting the data into train, validation, and test sets, normalising the input, and ensuring the output classes are represented equally.

The Student Alcohol Consumption dataset can be divided into two different types of data, categorical and numerical. The numerical data did not need processing and could be left as integer values, there were 15 columns with numerical data excluding 'Walc'. These were: 'age', 'Medu', 'Fedu', 'traveltime', 'studytime', 'failures', 'fmrel', 'freetime', 'goout', 'Dalc', 'Walc', 'health', 'absences', 'G1', 'G2', 'G3'. The remaining 17 categorical data were: 'school', 'sex', 'address', 'famsize', 'Pstatus', 'Mjob', 'Fjob', 'reason', 'guardian', 'schoolsup', 'famsup', 'paid', 'activities', 'nursery', 'higher', 'internet', 'romantic'. To ensure these could be used as features for our neural network, they were first turned into category types, which assigns each unique input an integer index. Then the data for categorical columns in our DataFrame was replaced with their integer index values. This made all the input data numerical and enabled it to be fed into the machine learning algorithm.

The second step of data preprocessing was to encode the output data to be in the format supported by the PyTorch Library. The loss functions implemented in our algorithm required the labels to be between [0…N], however, our targeted output was between 1 and 5. The 'Walc' column's data was mapped as follows, 1 to 0, 2 to 1, and so on. This resulted in the output lying between 0 and 4, and the loss functions could operate on these values without error.

Then the next steps of data preprocessing were to split the data and normalise the input values so they were in the range of 0 and 1. The sklearn library was used to perform both of these tasks. The data was first split into 80% training and validation data and 20% test data, randomising which the input rows in each set, then the training and validation data was split into 90% training and 10% validation. Then the function MinMaxScaler was used on data set splits, scaling it to being between the values of 0 and 1. This was done because normalising data can help make a neural network train faster, and reduce the chance of it getting stuck in local optima, as well as being more optimal for use with weight decay and Bayesian estimation [8].

Lastly, the class distribution of the output was measured to ensure the classes are equally represented in the different sets, and then corresponding weights were assigned to each output class. This was done due to the unbalanced output data, which has the highest frequency in the lowest class and decreases as the classes increase. The weights would help ensure the classes had better representation when training the model.

4 Model Validation

A set methodology was followed as we optimised our model. We used our baseline model as a starting point, tuning some of the hyper parameters to improve the accuracy. In the beginning of our validation, we were more concerned with increasing accuracy rather than dealing with the model overfitting the data. Once we had found an improved baseline model, we started to increase the number of hidden layers and their sizes. As we did this, we tuned the relevant hyper parameters to specifically optimise the model with the increased layer number and sizes. Finally, we experimented with and implemented several regularisation techniques to improve the generalizability of the model.

We utilised accuracy as the main performance metric of our model. The difference between the training and validation loss on the final training epoch was used as the indicator of whether the model was over or underfitting alongside the graphs of Train-Val Accuracy per Epoch, and Train-Val Loss per Epoch.

The complete documentation of our experimental results can be found in the 'Appendix' file.

4.1 Hidden Layers

We started our experiment by trying to determine the most performative architecture, in terms of the number of hidden layers and their neuron sizes. Online material varies when trying to provide a guideline on the number of hidden layers and their sizes, and ultimately all emphasise the need for experimentation.

When experimenting with the number of hidden layers, we removed regularisation from the model. This allowed us to determine the performance impact of the number of layers, without having to tune many hyperparameters. This also reduced the complexity of the experiment as we were only changing a handful of parameters. We added a layer, and then spent a reasonable amount of time manually tuning the learning rate, batch size, number of epochs (training time) and hidden layer neuron size. Throughout our experimentation we found that optimal the number of neurons in each hidden was when the layer size was decreased by a factor of 2 between layers. For example, Appendix E has 2 hidden layers. Hidden layer 1 has 512 inputs and 256 outputs, and hidden layer 2 has 256 inputs and 128 outputs.

Our baseline model (Appendix A) only had a single layer, and when trained only achieved 36% accuracy and overfit the training data immensely. We started out with a hidden layer size (hSize) starting with 512 neurons. When adjusting the hSize with a single layer, we found that a higher value led to decreased accuracy, but a more generalisable model. This can be seen in Appendix C and D, as well as Appendix AO and AP.

Throughout the project we experimented with models with up to 6 layers. When experimenting with 2 hidden layers, Appendix E & F performed moderately well (approx. 42%, with high overfitting). Comparatively, Appendix H had 2 hidden layers but a higher neuron size and did not perform as well as E or F (approx. 38%, with high overfitting).

Our 3 hidden layer models performed moderately well. We found that a lower hSize (512 < hSize < 2048) performed much better than a higher one (Appendix K compared to Appendix L). Our optimal 3 hidden layer model began as Appendix BC and was tuned up until Appendix BF, which achieved an accuracy of 51% with moderately low overfitting.

Experimentation with 4 to 6 hidden layer models was not as successful as the smaller models. Models with 4 hidden layers (Appendix N and O) were only able to achieve an optimal accuracy of 41%, while models with 5 and 6 hidden layers (Appendix P, Q, R and S) were able to achieve an optimal accuracy of 42%. All these models overfit the training data and were not especially generalisable. In addition, the larger model architecture meant that they required much longer training times – leading us to choose a lower number of hidden layers for our working models (those which we would continue to tune and optimse).

Notably, using the activation functions LeakyReLU and Hardswish produced interesting results (Appendix AD, AE, BB and Appendix AF, AG, BA respectively). However, using these activation functions also resulted in much higher training times. We found that models using these activation functions performed best on larger and more complex models.

We ultimately decided on a model with 1 hidden layer, as it produced the best results in terms of accuracy and generalizability.

4.2 Optimisation

As mentioned above, the model was tuned and optimised through experimentation as hidden layers were added on. For each hidden layer, we would configure it with different hyper parameters to see improvements and loss in quality. Many experiments have very similar configurations, where there is a minor change to one or two parameters. That is because those models showed promise and we tried to explore it more deeply. The parameters tweaked were the following: Epochs, learning rate, batch size, data split, batch normalisation, data splits, optimiser, and activation function. The parameters were configured trying to optimise accuracy, generalisation, and precision.

The baseline model (Appendix A) had a lackluster performance, with an accuracy of 36% and very high overfitting to the training data. We first tried to improve this baseline by changing the Epoch and batch size hyper parameter, and quickly found that the model seemed to operate best between 200 and 300 epochs, and between 16-32 batch size. Since an earlier experiment such as Appendix D, an epoch of 200 with batch size 16 showed slightly improved performance from the baseline, of 38% accuracy, 200 epochs and 16 batch size was used as a baseline for the further experimentation, changing it randomly during trial and error. For further experiments the epoch number was changed by looking at the graph for where the validation loss starts increasing consecutively, such as Appendix W, where the graph shows that the best epoch is probably in the range of 150 and 250, but 300 was used for that experiment, so it was adjusted for the next few experiments. A batch size of 32 also showed more promise in later experiments, that were better optimised and had more hidden layers, such as Appendix AJ.

The learning rate was optimised as the model became deeper, as a more hidden layers required different learning rates. Overall a learning rate of 0.0007 to 0.007 worked best, however 0.007 was the best fit for this model as shown in Appendix R and S. They are the same model the only difference is the learning rate, and Appendix S (0.007) outperforms Appendix R (0.0007) by 4% accuracy and has a smaller validation loss at the end of training. Appendix I and H are other two examples of 0.007 performing best. Smaller learning rates also gave good accuracy, like in Appendix F with 42%. However, it causes the model to overfit more as shown by the difference in training loss versus validation loss (0.0897 vs 1.9038), so we decided that 0.007 was the best, especially with later models such as Appendix W through AL.

One parameter that we tried to optimise unsuccessfully was the data split between training, validation, and testing sets. Our initial split for the baseline proved to be the best (72%/8%/20%), as evidence, when our final model was tested on three different splits, the baseline one performed the best. Appendix AK was split into 64% train, 16% validation, and 20% test, and had an accuracy of 44% with validation loss increasing rapidly. Appendix AL was split 81%/9%/10%, and had an accuracy of 40%, and validation loss also increase early. Appendix AB had the baseline split and performed at 52% accuracy with a improving smooth gradient for validation loss for about 250 epochs.

The factor which seemed to have the biggest influence on the model's performance was the optimiser. Changing from Adam to SGD improved the model's accuracy, precision, and helped to cull overfitting. On the baseline model, applying SGD (Appendix B), increased accuracy by 2%, but had a negative impact on training and validation accuracy and loss. However, with a more optimised model with parameters that fit the data better such as Appendix U, SGD reduced overfitting and performed better in accuracy. Since SGD has a weight decay parameter, the model worked best with a learning rate that was slightly bigger than the weight decay. We maintained the 0.007 as it was still performing well. To highlight how SGD fits our model better, we can contrast the results of Appendix AB and AM. AB has an accuracy of 52%, a well spread precision, and it well generalised according to the graphs. On the other hand, AM performs 10% worse, at 42%, with a precision of 0 for output class 4, and it is very overfit, the training accuracy approaches 100% early and validation loss increase is almost immediate.

A factor we took as being a baseline in neural networks in general, degraded the performance of the model. This is batch normalisation. When applying batch normalisation to Appendix AM, it results in an accuracy of 42%, with overfitting, and validation loss increases starting early. When batch normalisation is removed, Appendix AB the final model, performs 10% better in terms of accuracy. The model generalises better, and the precision is better for all output classes. Removing batch normalisation made a bigger difference than first expected.

Lastly, we tested three different activation functions with our final model to see if we could get better performance out of it. The three were ReLU, LeakyReLU, and Hardswish. The one that performed best was ReLU, however, the other two had similar results. Appendix AD and AE used LeakyReLU. The first one, resulted in 45% accuracy, but had a bigger validation loss than AB (ReLU). AE performed better than AD as it was given more time to train, it stopped at epoch 421, 100 over AD. The accuracy for this model was 48%, and the validation loss was 1.17 at the last epoch. Appendix AF and AG performed using Hardswish. Hardswish performed almost the same as ReLU, AF had a one percent less accuracy than AB at 50%, but trained for much longer, more than double the epochs (296 vs 612). When given more leeway to train, AG did not show an improvement, it performed at 48% accuracy. This led to us using ReLU for our final model, Appendix AB.

4.3 Regularisation

After optimising the hyper parameters and parameters of the model and finding the best accuracy we could, the model was still overfitting heavily. We applied regularisation techniques to cull the rate of increase of training accuracy, and to improve the validation loss. The techniques which worked best for this model were: weight decay, drop out, and early stopping.

The reason why SGD works so well with our model is its weight decay parameter. After adding SGD with a weight decay of 0.01 (Appendix W), the accuracy of the model went to 45%, which we had seen before, but the overfitting decreased by a lot. The training loss and validation loss ended at similar number, 0.939 and 1.125 respectively. The training accuracy also went up gradually instead of jumping to 100% from early iterations. The experiments for different weight decays showed that it performs best when it is just smaller than the learning rate. In Appendix X, we can see the accuracy drop when it is too low (0.0001), it drops to 43%, which is 2% lower than Appendix W. Since our best learning rate was 0.007, when the weight decay was 0.001, it performed best. Appendix Y shows the improvement from Appendix X, to 48% accuracy. As well as a good fit for validation and training loss.

In addition to weight decay, we implemented a drop out to regularise the model. If we compare Appendix W, to Appendix AA, the train and validation loss graphs show that the drop out flattens out the validation loss towards the end of training and the loss improves for longer. Appendix AA had an accuracy of 48% with a dropout rate of 0.3, which is 3% better than W with no drop out. Then tuning the dropout rate, in Appendix Z we see a 47% accuracy with a slightly better validation loss than AA. However, we found 0.1 to be the optimal dropout rate as is shown in Appendix Y and AB. This gave accuracies of 48% and 52%, with good generalisation.

The last regularisation technique implemented was early stopping, which stops the training after x consecutive iterations have had a validation loss. The x is the patience of the early stopping and was tuned by us. If the patience is too low then the model stops before it can start training if it has early validation losses, and if it is too high then it is not effective as it will not stop the training. We experimented with a few different numbers and found the range between 20 and 30 to work best with the final model. When using Other activation functions, having a higher patience helped as they need to train for longer, but Appendix AB still proved to be the best, with patience 20, resulting in an accuracy of 52%.

4.4 Bayesian Optimisation

Initially, we were manually tuning the hyperparameters for several reasons: we could avoid extra running times required to use the Bayesian optimiser to optimise the hyperparameters, we were able to quickly adjust hyperparameters to experiment with different configurations, and finally, we were able to improve our intuition and understanding of the relationships between the several hyperparameters.

Once we had sufficiently optimised the model manually, we experimented with Bayesian Optimization as a method of automatically calculating the optimal hyperparameter configuration. We chose to use the Bayesian approach to hyperparameter tuning over Grid Search and Random Search due to its increased performance and lower running time in general [6, 2].

Performing Bayesian Optimisation allowed us to further tune our hyperparameters and in some cases using the optimised hyperparameters to improve the model's performance. Comparing Appendix AQ to Appendix AR, AS and AT is one example of this. Appendix AQ is a manually tuned, 3 hidden layer architecture. When trained, it performed marginally better than the baseline (approx. 39% accuracy, and greatly overfit). Interestingly, Appendix AR, AS and AT's models each had the same architecture but varying sets of hyperparameters were determined when using the Bayesian Optimizer – a common theme throughout the experiments. Their performance results are shown below.

Optimised Model	Accuracy	Training Loss	Validation Loss
Appendix BD	44%	0.92283	1.61058
Appendix BE	42%	1.01109	1.36337
Appendix AT	48%	0.12290	1.51080

While the above 3-hidden-layer models may have been optimised with the Bayesian Optimizer, Appendix AR, AS and AT's models share the same architecture, but through manual tuning were able to perform better. Appendix BC, the best performing of the three, achieved 51% accuracy with minimal overfitting.

Unfortunately, the Bayesian Optimizer frequently provided hyperparameter tunings which performed worse than their manually tuned counterparts. Appendix AV and AW are models with 2 hidden layers, with hyperparameters tuned using the Bayesian Optimizer. Appendix AV performed best, but was still approximately 9% less accurate than the manually tuned Appendix AB. Similarly, with a 3 hidden layer model architecture Appendix AX, AZ and BC (tuned using the Bayesian Optimizer) performed worse in terms of accuracy and generalizability than Appendix AB.

When we initially setup the optimiser, we used the accuracy metric as the means of comparing different hyperparameter tunings during optimisation. We found that certain tunings may have optimised the accuracy but performed poorly when considering their precision for the different output classes. The importance of precision as a metric for generalizability is further discussed in section 6.3.

5 Evaluation

After methodologically trying to improve our baseline model, and after much experimenting we chose the final model to predict student's weekend alcohol consumption. The configuration of the model is as follows:

Activation Function	ReLU
Loss Function	Cross Entropy Loss
Optimiser	SGD, weight decay = 0.01
Training Epochs	10000
Data Split	Train:72% Validation:8% Test: 20%
Batch Size	32
Learning Rate	0.007
Number of Hidden Layers	1
Hidden Layers: Inputs and Outputs	1: (512) => (256)
Batch Normalisation	-
Early Stopping Patience	20
Drop Out Rate	0.1
Last Epoch Information	Epoch 296: | Train Loss: 1.04244 | Val Loss: 1.12140 | Train Acc: 80.000| Val Acc: 48.077
Test Accuracy	52%

The reason we chose this model as our final model is because of its superior accuracy, precision, and generalisation over other configurations. The output class distribution over these train, validation, and test sets is standardised so all classes are represented fairly in each split. The following split was used: 72% Training, 8% Validation, and 20% Test. This was the best split as can be seen by comparing it to the results of Appendix AK and Appendix AL, which had the same configuration but different data splits. The output class distribution between sets can be seen in Appendix Data Image B.

This final version of the model, with the above specifications, performed as follows:

	precision	recall	f1-score	support
0	0.59	0.86	0.70	50
1	0.53	0.27	0.36	30
2	0.30	0.33	0.31	24
3	0.38	0.18	0.24	17
4	0.86	0.67	0.75	9
accuracy			0.52	130
macro avg	0.53	0.46	0.47	130
weighted avg	0.51	0.52	0.49	130

The values of the model fluctuate each time it is run, so there is a margin of error for all the values including accuracy. From all the trial runs the margin seems to be between +- 3% and 5%. 52% is the highest performing accuracy from all model specifications, and the precision has performed reasonably well comparatively to the other experiment data from the appendix. With regularisation techniques such as weight decay, drop out rates, and early stopping we were also able to minimise overfitting of the data, which we mostly tracked via printing information after each Epoch, in the format shown in the table as 'Last Epoch Information', and visually in the form of the graphs below:

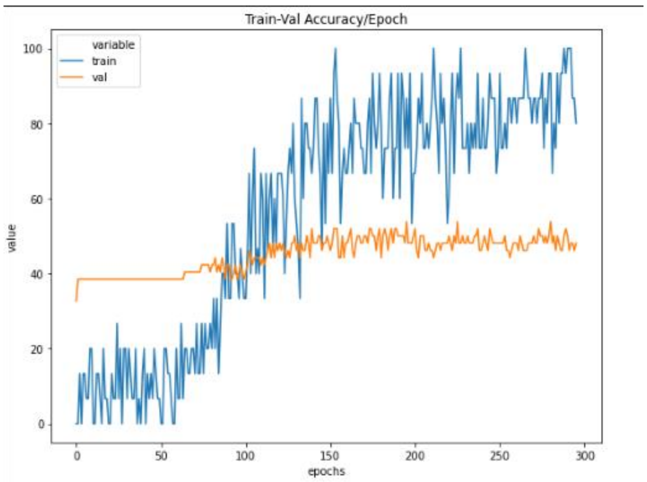


Figure 1: Train-Val Accuracy per Epoch

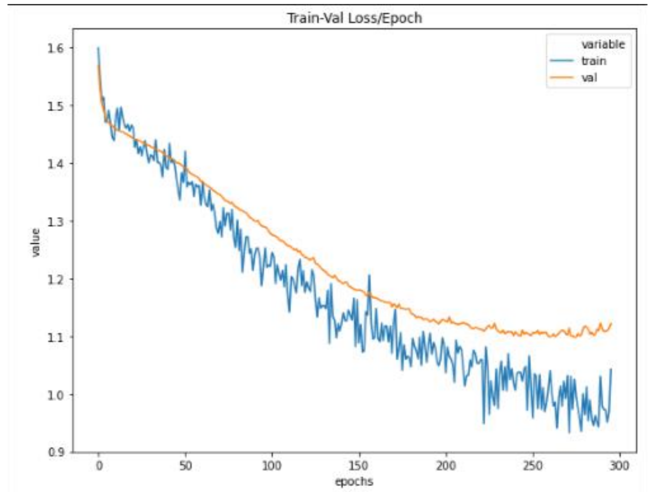


Figure 2: Train-Val Loss per Epoch

In Figure 2 the training loss and validation loss decrease at a similar rate, and when the validation loss starts increasing consecutively, the execution is stopped and tested. In Figure 1 there is an upwards trend in accuracy for both lines, even though it is not very apparent in the validation accuracy. The training set has a noise and some variance as shown by the alternating local peaks and valleys.

It is worth mentioning that there were a few other models that performed similarly, one of them was Appendix AH, which is the same but with an extra hidden layer. Appendix AD up until Appendix AG also performed well, they used different activation functions from the final model, LeakyReLU and Hardswish. Overall, this model (Appendix AB) was the best performing model from all the experiments we performed.

6 Analysis

Appendix AB, our final model showed a significant performance increase from the baseline model, Appendix A. The accuracy improved by 18%, however this number could be slightly bigger or smaller due to a margin of error, since the models get different results every time they are run. This is a good improvement however it is not ideal, as it only results in an accuracy of 52%. The final model is well generalised, the validation and training loss decrease at a similar rate as shown in Figure 2, which gives less volatile results when predicting output classes. The precision of the final model has also increased significantly from the baseline. The performance of the final model, its limitations and what can be done to improve it will be discussed further in this section.

6.1 Baseline

The baseline model was adapted from Akshaj Verma's feed-forward neural network which used a data set about wine quality, with 11 features and 6 output classes [9]. This dataset is less complex than ours, but it has similar characteristics, so it fit our model well. After removing the regularisation techniques from this model, we arrived at our baseline, which performed poorly. The baseline performed at 36% accuracy and was strongly overfitting the data. By evaluating the train-val accuracy per epoch graph, it is evident the training accuracy converges to 100% in less than 50 epochs, and the validation accuracy never has a steady increase. The loss per epoch graph confirms the overfitting, the validation loss forms the shape of a logarithmic function, increasing rapidly, while the training loss converges to 0.

The baseline seems to predict output class 0 well, with a 55% precision, but lacks in performance in class 1,2, and 3. This is most likely because of class 0 is the most frequent, so the model can learn it more generally than the rest. Output class 4 also had a good precision, of 50%. This is the least frequent class in the dataset, which could be the reason for performing well. There is a very small data sample for that class, so if there are a few features which have a high influence on the outcome, the model will overfit those features and have a higher probability of recalling the outcome from those features. The baseline gave us good insight into what kind of improvements we needed to look for, and guided us in the right direction to the final model.

6.2 Performance

Through extensive experimentation and trial and error we were able to configure our final model (Appendix AB) to perform adequately with the resources available to us in the context of the problem of Student's Weekend Alcohol Consumption. This section will discuss why the configurations of the final model led to its results and why some configurations were changed or left out of the model. Configurations such as the number of hidden layers, activation functions, and regularisation techniques had great influence on the final performance.

We noticed that as the number of hidden layers increased, the accuracy of the model would generally increase as well once it had been tuned. However, as the model was tuned the precision of the less frequent classes started to decrease. Thus, while the accuracy was increasing, we were starting to predict the most common class more often, and became less accurate with the less common classes. This is apparent with Appendix BF, AJ, and U, as they have 3, 3 and 6 hidden layers respectively. They performed well in terms of accuracy and generalisation but had 0% precision for output class 4, which is the least frequent class. This most likely occurs due to having a high number of hidden layers results in the additional layers trying to learn complex representations of features [4]. If the model tries to learn these complex representations with the little amount of data available in our dataset, it will probably fail. Since output class 4 is under represented in the data, this complex representation of extra layers can not make predictions for that output class correctly as it can not make a representation fitting that output with little data.

The best performance jump in accuracy in the experiments came from switching optimiser from Adam to SGD. In Keskar's research paper he discusses the difference between the two extensively [5]. Adam works better with untuned hyperparameters and performs better early, however SGD is better at generalising and performs best over time. Due to our experiments often having high overfitting, this is probably the reason SGD performed best, alongside its weight decay parameter, which further reduced overfitting of the data and made predictions more accurate. We also performed many experiments on the model, which allowed us to more finely tune the parameters, which makes the Adam' advantage of being more lenient with parameters choices less optimal. SGD with a learning rate of 0.07 and a weight decay of 0.001 worked well as the weight steps were still big enough for the model to learn but the weight decay adjusted it to ensure the data did not overfit, alongside a small drop out rate. having a smaller learning rate did not work well as the training set was not big enough. It is noted that an approach that could have improved performance is having an adaptive optimiser which switches from Adam to SGD at an optimal point [5].

It was interesting to see that removing batch normalisation also improved the performance of the model, which was unexpected. We assume this is because SGD undoes the normalisation done by batch normalisation if it will minimise the loss function [3]. We also normalise the inputs to be in the range of 0 and 1 before feeding it to the model, which could have a similar impact to batch normalisation.

An interesting comparison can be made between the performance of activation functions in the model. The final function we chose was ReLu, the reason for this was it had the best performance in terms of accuracy and precision, and it also performed best with a smaller model, leading to smaller training times and a less complex model. The alternatives were LeakyReLU and Hardswish. LeakyReLU worked better with slightly bigger models, and needed to train for longer to achieve optimal results. The results were just less optimal than ReLU, so the time and complexity trade-off were not worth it, though this might be different with more data available. Hardswish seemed to need a much deeper and complex model to perform well, and trained for a very long time. When implemented with early stopping, the patience had to be high for Hardswish, otherwise it stopped very early on or before improvements continued. LeakyReLU performed as a worse version of ReLU in this model, and Hardswish had volatile behavior and needed a more complex model, which is not optimal with little data, therefore ReLU was the best option for our final model; it had a good balance of short training time and good accuracy.

By having no regularisation when tuning the model we could find the best configurations that would fit the features of the experiment, and we would know which hyper parameter was making the difference. Once the tuning was optimal, we added weight decay to SGD, a drop out rate, and early stopping to the model. The weight decay allowed the model to continue learning for longer before starting to overfit, and it also gave more representation to the less frequent output classes, as it adjusts the weight of the classes accordingly, leading to much better precision in output classes 1 through 4. A drop out rate of 0.1 also helped the model become more generalised, by adding more noise between inputs sporadically, leading to the model having to adapt without overfitting. Lastly, by analysing the graphs we could approximate a better performance at a certain epoch range, so we implemented early stopping. This added more consistency to the model, and improved overall performance as it stops training before the results start deteriorating.

6.3 Evaluation Metrics

When evaluating the model's performance, we aimed to achieve a high level of prediction accuracy while keeping the model generalisable (ensuring a low difference between training loss and validation loss). We primarily used the difference between the training loss and the validation loss as an indication of the model's generalizability. However, we also noticed another metric that could be used to gain insight into its generalizability – precision.

Precision provides insight into how many false positives were predicted for a particular output class. Thus, it indicates which output classes the model predicts most accurately. If a model is highly generalisable, we would expect to see similar precision scores for each of the output classes, while a less generalisable model would produce precision scores with high variance and a lower average precision. In some cases, a more generalisable model could result in lower accuracy, as higher frequency output classes (more common output classes) with high precision would have a comparatively high number of predictions.

In general, we noticed that models with less hidden layers performed better in terms of prediction precision (Appendix Y, AB), while those with many hidden layers performed worse (Appendix AS, AV).

The difference in precision may have been due to the different class frequency present in the training dataset. For example, data items with output class 4 were much less common than class 0, and subsequently had lower precision scores. Balancing the frequency of different output classes in the training data may lead to an increase in the precision.

6.4 Data Impact

The type and amount of data contained within the dataset may have impacted the overall performance of our model. The dataset only contains 649 items, which places a hard limit on the amount of data that could be used to train the model. This may have been the reason that we were unable to achieve a higher accuracy. However, this hypothesis can only be tested with additional data.

In addition to considering the amount of data available to train the model, we must also concern ourselves with its quality. The data was gathered via a survey, and as such each answer is subjective and has the potential to be impacted by personal biases. This presents a challenge, as it may reduce the overall correlation between the values in a single data entry, and ultimately the dataset. This may be the reason that the model was unable to achieve a high degree of accuracy – the subjective data is so varied that a strong correlation between the feature values does not exist.

This dataset was created from survey results from two secondary schools in the same geographic location. Thus, the cultural, geographical and societal aspects of the learner's life may have impacted the survey results. This may have compromised the overall generalizability of the model, as the correlations may be due to these specific student's shared context.

6.5 Potential Improvements

While we believe that the model performed relatively well given the constraints mentioned, there are avenues to explore that could improve the overall performance of the model.

Improving the dataset may have a positive impact on the model's performance. A larger dataset may improve the accuracy and generalizability of the model, due to the increased amount of training data. Improving the data's integrity and quality – not using surveys to collect data, collecting data from several contexts (different countries, regions, socio-economic contexts, etc.) – could also improve the performance of the model. Indeed, more generalised data from a wider group of subjects is highly likely to increase the generalizability of the model.

The dataset that we used has a high number of features (33). Having a high number of features (and a low number of data records) increases its complexity, making it more difficult for a model to make accurate predictions. Through experimentation, one could determine which features are most important for predicting weekend alcohol consumption and drop them from the dataset. This could simplify the dataset and improve the model's performance.

Knowledge of the specific problem domain is extremely important in solving problems with artificial intelligence. The team members are understandably not extremely knowledgeable about teenage drinking habits. An improvement in our understanding may have potential benefits to the approach and methodology when solving this problem. It may have also informed us about whether such a problem is even solvable using a neural network and the data we had available. Alternatively, consultations with a domain expert could prove useful for further work with this model and problem space.

While we did explore the potential of using Bayesian Optimisation for guiding hyperparameter tuning, there are numerous other methods that could be explored. Through experimentation with different tuning methods, such as grid and random search, the model's overall performance could be improved.

REFERENCES

[1] Ganesh Arulampalam, Abdesselam Bouzerdoum. 2003. A generalised feedforward neural network architecture for classification and regression. Neural Networks, Volume 16, Issues 5–6, Pages 561-568, ISSN 0893-6080, https://doi.org/10.1016/S0893-6080(03)00116-3.

[2] James S Bergstra, Rémi Bardenet, Yoshua Bengio, Balázs Kégl. 2011. Algorithms for Hyper-Parameter Optimization. Advances in Neural Information Processing Systems 24 (NIPS 2011), pp. 2546-2554.

[3] - F D. 2017. Batch normalisation in Neural Networks. Retrieved November 12, 2020 from https://towardsdatascience.com/batch-normalization-in-neural-networks-1ac91516821c.

[4] - Jeff Heaton. 2017. Heaton Research, The Number of Hidden Layers. Retrieved November 11, 2020 from https://www.heatonresearch.com/2017/06/01/hidden-layers.html.

[5] - Nitish Shirish Keskar, and Richard Socher. 2017. Keskar, N.S. and Socher, R., 2017. Improving generalisation performance by switching from adam to sgd. arXiv preprint arXiv:1712.07628.

[6] Will Koehrsen. 2018. Model Tuning Results: Random vs Bayesian Opt. Retrieved November 8, 2020 from https://www.kaggle.com/willkoehrsen/model-tuning-results-random-vs-bayesian-opt.

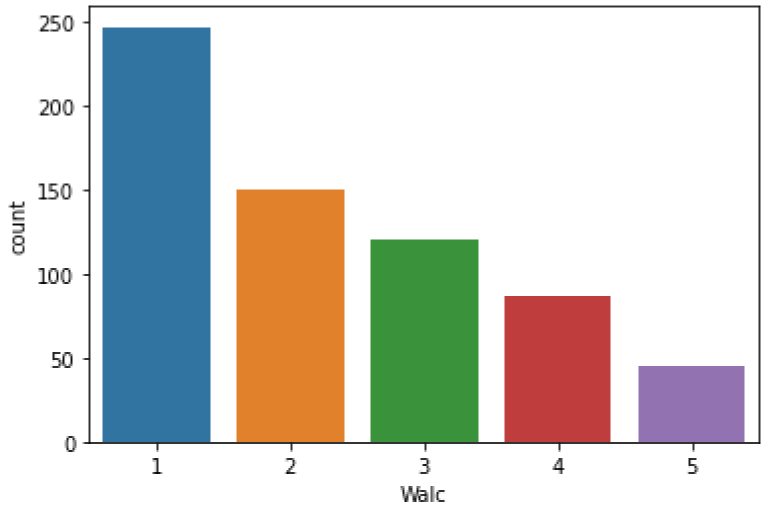
[7] Usman Malik. 2019. Introduction to PyTorch for Classification. Retrieved November 6, 2020 from https://stackabuse.com/introduction-to-pytorch-for-classification/.

[8] Warren S. Sarle. 2002. comp.ai.neural-nets FAQ, Part 2 of 7: Learning. Retrieved November 9, 2020 from http://www.faqs.org/faqs/ai-faq/neural-nets/part2/.

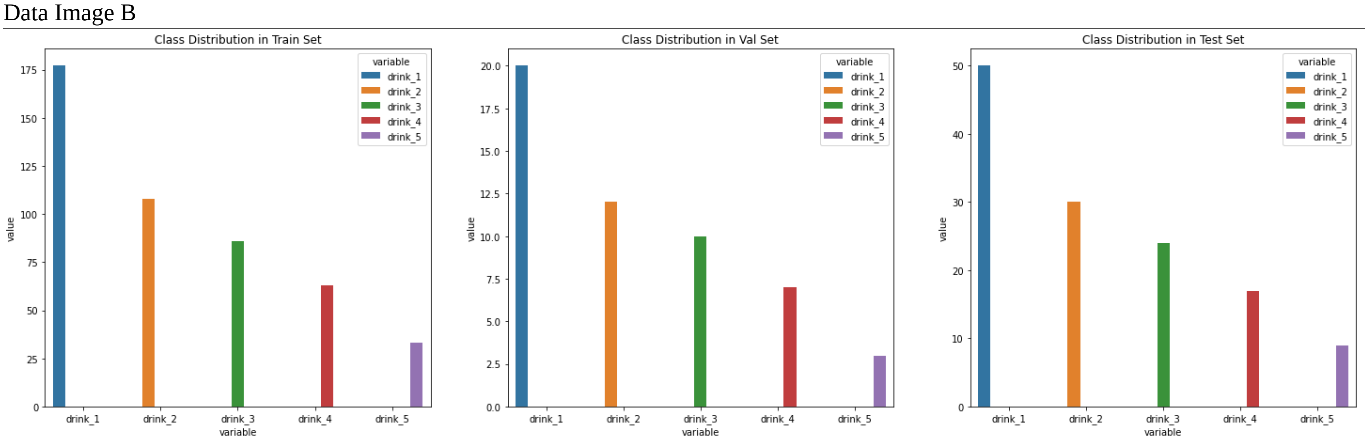
[9] Akshaj Verma. 2020. PyTorch [Tabular] —Multiclass Classification. Retrieved November 6, 2020 from https://towardsdatascience.com/pytorch-tabular-multiclass-classification-9f8211a123ab.

APPENDIX

Data Image A



Data Image B



Appendix A

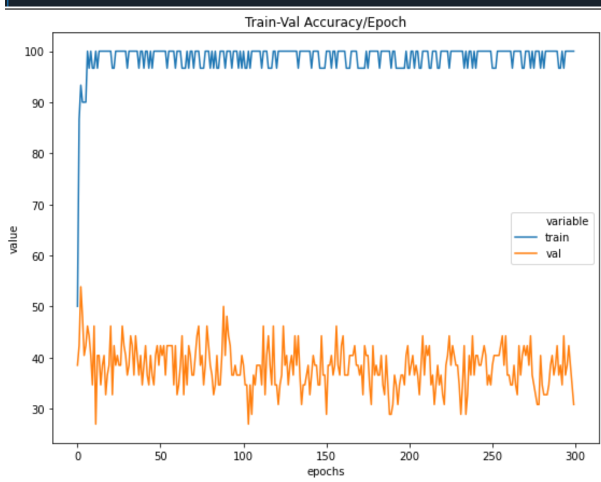
Activation Function	ReLU
Loss Function	Cross Entropy Loss
Optimizer	Adam
Training Epochs	300
Batch Size	16
Learning Rate	0.0007
Number of Hidden Layers	1
Hidden Layers: Inputs and Outputs	1: (512) => (256)
Batch Normalization	Input Layer: 512, Hidden Layer 1: 256
Early Stopping Patience	-
Drop Out Rate	-
Last Epoch Information	Epoch 300: | Train Loss: 0.04254 | Val Loss: 2.83884 | Train Acc: 100.000| Val Acc: 30.769
Test Accuracy	36%

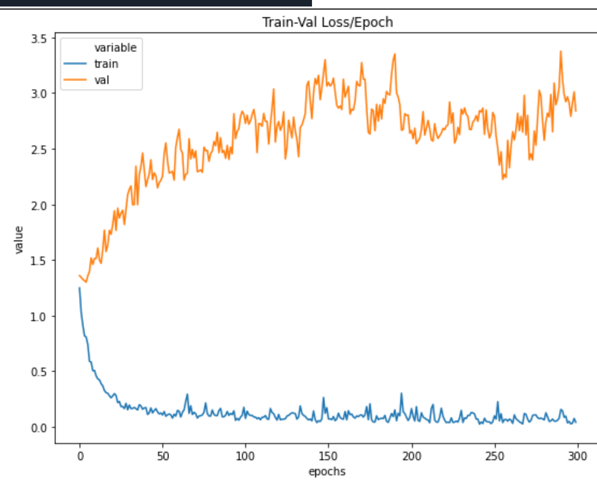
Output table screen shot:

```
              precision    recall  f1-score   support

           0       0.55      0.60      0.57        50
           1       0.19      0.20      0.19        30
           2       0.10      0.08      0.09        24
           3       0.36      0.29      0.32        17
           4       0.50      0.44      0.47         9

    accuracy                           0.36       130
   macro avg       0.34      0.32      0.33       130
weighted avg       0.35      0.36      0.36       130
```





Appendix B

Activation Function	ReLU
Loss Function	Cross Entropy Loss
Optimizer	SGD
Training Epochs	200
Batch Size	16
Learning Rate	0.0007
Number of Hidden Layers	1
Hidden Layers: Inputs and Outputs	1: (512) => (256)
Batch Normalization	Input Layer: 512, Hidden Layer 1: 256
Early Stopping Patience	-
Drop Out Rate	-
Last Epoch Information	Epoch 200: | Train Loss: 1.42068 | Val Loss: 1.43115 | Train Acc: 6.667| Val Acc: 38.462
Test Accuracy	38%

Output table screen shot:

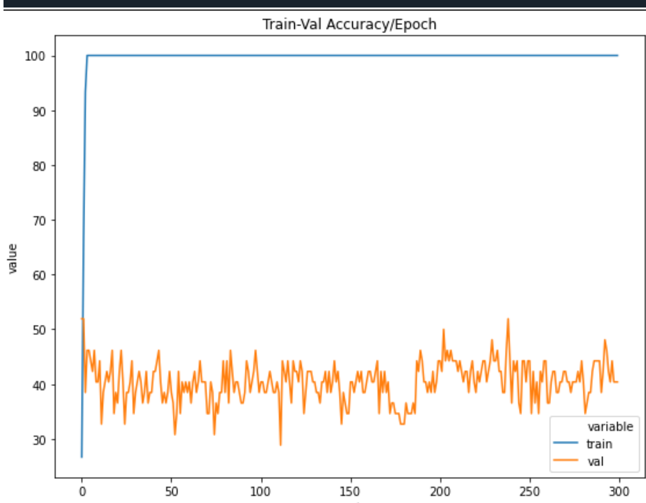
	precision	recall	f1-score	support
0	0.38	1.00	0.56	50
1	0.00	0.00	0.00	30
2	0.00	0.00	0.00	24
3	0.00	0.00	0.00	17
4	0.00	0.00	0.00	9
accuracy			0.38	130
macro avg	0.08	0.20	0.11	130
weighted avg	0.15	0.38	0.21	130

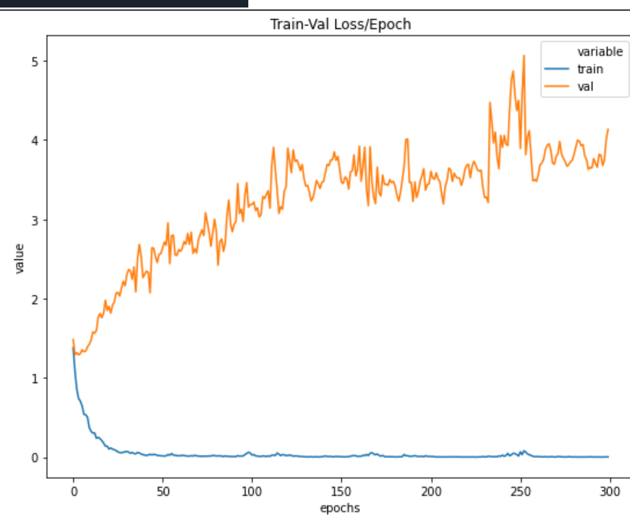
Appendix C

Activation Function	ReLU
Loss Function	Cross Entropy Loss
Optimizer	Adam
Training Epochs	300
Batch Size	32
Learning Rate	0.0007
Number of Hidden Layers	1
Hidden Layers: Inputs and Outputs	1: (512) => (256)
Batch Normalization	Input Layer: 512, Hidden Layer 1: 256
Early Stopping Patience	-
Drop Out Rate	-
Last Epoch Information	Epoch 300: | Train Loss: 0.00193 | Val Loss: 4.13492 | Train Acc: 100.000| Val Acc: 40.385
Test Accuracy	40%

Output table screen shot:

	precision	recall	f1-score	support
0	0.50	0.70	0.58	50
1	0.30	0.23	0.26	30
2	0.11	0.08	0.10	24
3	0.38	0.29	0.33	17
4	0.50	0.33	0.40	9
accuracy			0.40	130
macro avg	0.36	0.33	0.34	130
weighted avg	0.37	0.40	0.37	130



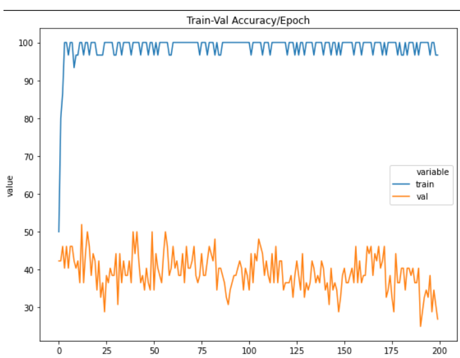


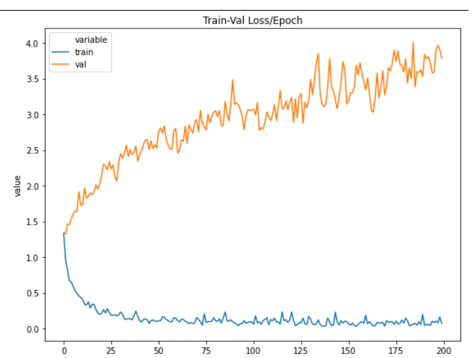
Appendix D

Activation Function	ReLU
Loss Function	Cross Entropy Loss
Optimizer	Adam
Training Epochs	200
Batch Size	16
Learning Rate	0.0007
Number of Hidden Layers	1
Hidden Layers: Inputs and Outputs	1: (1024) => (512)
Batch Normalization	Input Layer: 1024, Hidden Layer 1: 512
Early Stopping Patience	-
Drop Out Rate	-
Last Epoch Information	Epoch 200: | Train Loss: 0.07149 | Val Loss: 3.78730 | Train Acc: 96.667| Val Acc: 26.923
Test Accuracy	38%

Output table screen shot:

	precision	recall	f1-score	support
0	0.51	0.54	0.52	50
1	0.19	0.17	0.18	30
2	0.18	0.21	0.19	24
3	0.43	0.35	0.39	17
4	0.75	0.67	0.71	9
accuracy			0.38	130
macro avg	0.41	0.39	0.40	130
weighted avg	0.38	0.38	0.38	130





Appendix E

Activation Function	ReLU
Loss Function	Cross Entropy Loss
Optimizer	Adam
Training Epochs	200
Batch Size	16
Learning Rate	0.0007
Number of Hidden Layers	2
Hidden Layers: Inputs and Outputs	1: (512) => (256) 2: (256) => (128)
Batch Normalization	Input Layer: 512, Hidden Layer 1: 256, Hidden Layer 2: 128
Early Stopping Patience	-
Drop Out Rate	-
Last Epoch Information	Epoch 200: | Train Loss: 0.08759 | Val Loss: 3.19291 | Train Acc: 96.667| Val Acc: 30.769
Test Accuracy	42%

Output table screen shot:

	precision	recall	f1-score	support
0	0.53	0.72	0.61	50
1	0.32	0.30	0.31	30
2	0.15	0.08	0.11	24
3	0.29	0.24	0.26	17
4	0.43	0.33	0.38	9
accuracy			0.42	130
macro avg	0.34	0.33	0.33	130
weighted avg	0.37	0.42	0.39	130

Appendix F

Activation Function	ReLU
Loss Function	Cross Entropy Loss
Optimizer	Adam
Training Epochs	200
Batch Size	16
Learning Rate	0.00007
Number of Hidden Layers	2
Hidden Layers: Inputs and Outputs	1: (512) => (256) 2: (256) => (128)
Batch Normalization	Input Layer: 512, Hidden Layer 1: 256, Hidden Layer 2: 128
Early Stopping Patience	-
Drop Out Rate	-
Last Epoch Information	Epoch 200: | Train Loss: 0.08966 | Val Loss: 1.90381 | Train

Predicting Student's Weekend Alcohol Consumption

	Acc: 100.000| Val Acc: 34.615
Test Accuracy	42%

Output table screen shot:

	precision	recall	f1-score	support
0	0.57	0.64	0.60	50
1	0.27	0.23	0.25	30
2	0.21	0.21	0.21	24
3	0.33	0.29	0.31	17
4	0.56	0.56	0.56	9
accuracy			0.42	130
macro avg	0.39	0.39	0.39	130
weighted avg	0.40	0.42	0.41	130

Appendix G

Activation Function	ReLU
Loss Function	Cross Entropy Loss
Optimizer	Adam
Training Epochs	200
Batch Size	32
Learning Rate	0.0007
Number of Hidden Layers	2
Hidden Layers: Inputs and Outputs	1: (512) => (256) 2: (256) => (128)
Batch Normalization	Input Layer: 512, Hidden Layer 1: 256, Hidden Layer 2: 128
Early Stopping Patience	-
Drop Out Rate	-
Last Epoch Information	Epoch 200: | Train Loss: 0.00513 | Val Loss: 3.05927 | Train Acc: 100.000| Val Acc: 40.385
Test Accuracy	39%

Output table screen shot:

	precision	recall	f1-score	support
0	0.55	0.66	0.60	50
1	0.35	0.27	0.30	30
2	0.04	0.04	0.04	24
3	0.25	0.18	0.21	17
4	0.50	0.67	0.57	9
accuracy			0.39	130
macro avg	0.34	0.36	0.34	130
weighted avg	0.37	0.39	0.37	130

Appendix H

Activation Function	ReLU
Loss Function	Cross Entropy Loss
Optimizer	Adam
Training Epochs	200
Batch Size	16
Learning Rate	0.0007
Number of Hidden Layers	2
Hidden Layers: Inputs and Outputs	1: (1024) => (512) 2: (512) => (256)
Batch Normalization	Input Layer: 1024, Hidden Layer 1: 512, Hidden Layer 2: 256
Early Stopping Patience	-
Drop Out Rate	-
Last Epoch Information	Epoch 200: | Train Loss: 0.08633 | Val Loss: 3.25121 | Train Acc: 100.000| Val Acc: 34.615
Test Accuracy	38%

Output table screen shot:

	precision	recall	f1-score	support
0	0.56	0.54	0.55	50
1	0.28	0.30	0.29	30
2	0.13	0.12	0.13	24
3	0.35	0.35	0.35	17
4	0.50	0.56	0.53	9
accuracy			0.38	130
macro avg	0.37	0.37	0.37	130
weighted avg	0.39	0.38	0.39	130

Appendix I

Activation Function	ReLU
Loss Function	Cross Entropy Loss
Optimizer	Adam
Training Epochs	200
Batch Size	16
Learning Rate	0.007
Number of Hidden Layers	2
Hidden Layers: Inputs and Outputs	1: (1024) => (512) 2: (512) => (256)
Batch Normalization	Input Layer: 1024, Hidden Layer 1: 512, Hidden Layer 2: 256
Early Stopping Patience	-
Drop Out Rate	-
Last Epoch Information	Epoch 200: | Train Loss: 0.19520 | Val Loss: 3.64504 | Train Acc: 96.667| Val Acc: 42.308
Test Accuracy	42%

Output table screen shot:

	precision	recall	f1-score	support
0	0.57	0.68	0.62	50
1	0.27	0.20	0.23	30
2	0.17	0.17	0.17	24
3	0.36	0.29	0.32	17
4	0.45	0.56	0.50	9
accuracy			0.42	130
macro avg	0.36	0.38	0.37	130
weighted avg	0.39	0.42	0.40	130

Appendix J

<table>
<tr><td>Activation Function</td><td>ReLU</td></tr>
<tr><td>Loss Function</td><td>Cross Entropy Loss</td></tr>
<tr><td>Optimizer</td><td>Adam</td></tr>
<tr><td>Training Epochs</td><td>200</td></tr>
<tr><td>Batch Size</td><td>16</td></tr>
<tr><td>Learning Rate</td><td>0.0007</td></tr>
<tr><td>Number of Hidden Layers</td><td>2</td></tr>
<tr><td>Hidden Layers: Inputs and Outputs</td><td>1: (256) => (128) 2: (128) => (64)</td></tr>
<tr><td>Batch Normalization</td><td>Input Layer: 256, Hidden Layer 1: 128, Hidden Layer 2: 64</td></tr>
<tr><td>Early Stopping Patience</td><td>-</td></tr>
<tr><td>Drop Out Rate</td><td>-</td></tr>
<tr><td>Last Epoch Information</td><td>Epoch 200: | Train Loss: 0.14128 | Val Loss: 2.63619 | Train Acc: 100.000| Val Acc: 34.615</td></tr>
<tr><td>Test Accuracy</td><td>39%</td></tr>
</table>

Output table screen shot:

	precision	recall	f1-score	support
0	0.55	0.64	0.59	50
1	0.22	0.17	0.19	30
2	0.24	0.25	0.24	24
3	0.29	0.24	0.26	17
4	0.40	0.44	0.42	9
accuracy			0.39	130
macro avg	0.34	0.35	0.34	130
weighted avg	0.37	0.39	0.38	130

Appendix K

Activation Function	ReLU
Loss Function	Cross Entropy Loss
Optimizer	Adam
Training Epochs	200
Batch Size	16
Learning Rate	0.0007
Number of Hidden Layers	3
Hidden Layers: Inputs and Outputs	1: (512) => (256) 2: (256) => (128) 3: (128) => (64)
Batch Normalization	Input Layer: 512, Hidden Layer 1: 256, Hidden Layer 2: 128, Hidden Layer 3: 64
Early Stopping Patience	-
Drop Out Rate	-
Last Epoch Information	Epoch 200: | Train Loss: 0.11458 | Val Loss: 2.97218 | Train Acc: 100.000| Val Acc: 46.154
Test Accuracy	42%

Output table screen shot:

	precision	recall	f1-score	support
0	0.57	0.66	0.61	50
1	0.30	0.27	0.28	30
2	0.17	0.12	0.14	24
3	0.29	0.35	0.32	17
4	0.67	0.44	0.53	9
accuracy			0.42	130
macro avg	0.40	0.37	0.38	130
weighted avg	0.40	0.42	0.40	130

Appendix L

Activation Function	ReLU
Loss Function	Cross Entropy Loss
Optimizer	Adam
Training Epochs	200
Batch Size	16
Learning Rate	0.0007
Number of Hidden Layers	3
Hidden Layers: Inputs and Outputs	1: (2048) => (1024) 2: (1024) => (512) 3: (512) => (256)
Batch Normalization	Input Layer: 2048, Hidden Layer 1: 1024, Hidden Layer 2: 512, Hidden Layer 3: 256
Early Stopping Patience	-
Drop Out Rate	-
Last Epoch Information	Epoch 200: | Train Loss: 0.04682 | Val Loss: 3.26323 | Train Acc: 100.000| Val Acc: 44.231
Test Accuracy	35%

Output table screen shot:

	precision	recall	f1-score	support
0	0.53	0.60	0.56	50
1	0.14	0.10	0.12	30
2	0.20	0.21	0.20	24
3	0.25	0.24	0.24	17
4	0.40	0.44	0.42	9
accuracy			0.35	130
macro avg	0.30	0.32	0.31	130
weighted avg	0.33	0.35	0.34	130

Appendix M

Activation Function	ReLU
Loss Function	Cross Entropy Loss
Optimizer	Adam
Training Epochs	200
Batch Size	16
Learning Rate	0.0007
Number of Hidden Layers	4
Hidden Layers: Inputs and Outputs	1: (512) => (256) 2: (256) => (128) 3: (128) => (64) 4: (64) => (32)
Batch Normalization	Input Layer: 2048, Hidden Layer 1: 1024, Hidden Layer 2: 512, Hidden Layer 3: 256, Hidden Layer 4: 256
Early Stopping Patience	-
Drop Out Rate	-
Last Epoch Information	Epoch 200: | Train Loss: 0.18398 | Val Loss: 2.32281 | Train Acc: 96.667| Val Acc: 40.385
Test Accuracy	40%

Output table screen shot:

	precision	recall	f1-score	support
0	0.60	0.62	0.61	50
1	0.26	0.17	0.20	30
2	0.21	0.25	0.23	24
3	0.27	0.35	0.31	17
4	0.44	0.44	0.44	9
accuracy			0.40	130
macro avg	0.36	0.37	0.36	130
weighted avg	0.40	0.40	0.39	130

Appendix N

Activation Function	ReLU
Loss Function	Cross Entropy Loss
Optimizer	Adam
Training Epochs	200
Batch Size	16
Learning Rate	0.0007
Number of Hidden Layers	4
Hidden Layers: Inputs and Outputs	1: (1024) => (512) 2: (512) => (256) 3: (256) => (128) 4: (128) => (64)
Batch Normalization	Input Layer: 1024, Hidden Layer 1: 512, Hidden Layer 2: 256, Hidden Layer 3: 128, Hidden Layer 4: 64
Early Stopping Patience	-
Drop Out Rate	-
Last Epoch Information	Epoch 200: | Train Loss: 0.11862 | Val Loss: 2.52515 | Train Acc: 100.000| Val Acc: 38.462
Test Accuracy	41%

Output table screen shot:

	precision	recall	f1-score	support
0	0.61	0.66	0.63	50
1	0.31	0.30	0.31	30
2	0.16	0.12	0.14	24
3	0.24	0.29	0.26	17
4	0.43	0.33	0.38	9
accuracy			0.41	130
macro avg	0.35	0.34	0.34	130
weighted avg	0.40	0.41	0.40	130

Appendix O

Activation Function	ReLU
Loss Function	Cross Entropy Loss
Optimizer	Adam
Training Epochs	200
Batch Size	16
Learning Rate	0.0007
Number of Hidden Layers	5
Hidden Layers: Inputs and Outputs	1: (512) => (256) 2: (256) => (128) 3: (128) => (64) 4: (64) => (32) 5: (32) => (16)
Batch Normalization	Input Layer: 512, Hidden Layer 1: 256, Hidden Layer 2: 128, Hidden Layer 3: 64, Hidden Layer 4: 32, Hidden Layer 5: 16
Early Stopping Patience	-
Drop Out Rate	-
Last Epoch Information	Epoch 200: | Train Loss: 0.21391 | Val Loss: 2.13607 | Train Acc: 100.000| Val Acc: 34.615
Test Accuracy	36%

Output table screen shot:

	precision	recall	f1-score	support
0	0.57	0.56	0.57	50
1	0.19	0.17	0.18	30
2	0.18	0.21	0.19	24
3	0.22	0.24	0.23	17
4	0.62	0.56	0.59	9
accuracy			0.36	130
macro avg	0.36	0.35	0.35	130
weighted avg	0.37	0.36	0.36	130

Appendix P

Activation Function	ReLU
Loss Function	Cross Entropy Loss
Optimizer	Adam
Training Epochs	200
Batch Size	16
Learning Rate	0.0007
Number of Hidden Layers	5
Hidden Layers: Inputs and Outputs	1: (1024) => (512) 2: (512) => (256) 3: (256) => (128) 4: (128) => (64) 5: (64) => (32)
Batch Normalization	Input Layer: 1024, Hidden Layer 1: 512, Hidden Layer 2: 256, Hidden Layer 3: 128, Hidden Layer 4: 64, Hidden Layer 5: 32
Early Stopping Patience	-
Drop Out Rate	-
Last Epoch Information	Epoch 200: | Train Loss: 0.13348 | Val Loss: 2.54188 | Train Acc: 100.000| Val Acc: 36.538
Test Accuracy	39%

Output table screen shot:

	precision	recall	f1-score	support
0	0.53	0.64	0.58	50
1	0.33	0.30	0.32	30
2	0.13	0.12	0.13	24
3	0.31	0.29	0.30	17
4	0.50	0.22	0.31	9
accuracy			0.39	130
macro avg	0.36	0.32	0.33	130
weighted avg	0.38	0.39	0.38	130

Appendix Q

Activation Function	ReLU
Loss Function	Cross Entropy Loss
Optimizer	Adam
Training Epochs	200
Batch Size	16
Learning Rate	0.0007
Number of Hidden Layers	6
Hidden Layers: Inputs and Outputs	1: (512) => (256) 2: (256) => (128) 3: (128) => (64) 4: (64) => (32) 5: (32) => (16) 6: (16) => (8)
Batch Normalization	Input Layer: 512, Hidden Layer 1: 256, Hidden Layer 2: 128, Hidden Layer 3: 64, Hidden Layer 4: 32, Hidden Layer 5: 16, Hidden Layer 5: 8
Early Stopping Patience	-
Drop Out Rate	-
Last Epoch Information	Epoch 200: | Train Loss: 0.37764 | Val Loss: 2.12405 | Train Acc: 96.667| Val Acc: 32.692
Test Accuracy	34%

Output table screen shot:

	precision	recall	f1-score	support
0	0.52	0.50	0.51	50
1	0.19	0.17	0.18	30
2	0.16	0.21	0.18	24
3	0.29	0.29	0.29	17
4	0.50	0.44	0.47	9
accuracy			0.34	130
macro avg	0.33	0.32	0.33	130
weighted avg	0.35	0.34	0.34	130

Appendix R

Activation Function	ReLU
Loss Function	Cross Entropy Loss
Optimizer	Adam
Training Epochs	200
Batch Size	16
Learning Rate	0.0007
Number of Hidden Layers	6
Hidden Layers: Inputs and Outputs	1: (1024) => (512) 2: (512) => (256) 3: (256) => (128) 4: (128) => (64) 5: (64) => (32) 6: (32) => (16)
Batch Normalization	Input Layer: 1024, Hidden Layer 1: 512, Hidden Layer 2: 256, Hidden Layer 3: 128, Hidden Layer 4: 64, Hidden Layer 5: 32, Hidden Layer 5: 16
Early Stopping Patience	-
Drop Out Rate	-
Last Epoch Information	Epoch 200: | Train Loss: 0.30472 | Val Loss: 2.37141 | Train Acc: 96.667| Val Acc: 38.462
Test Accuracy	38%

Output table screen shot:

	precision	recall	f1-score	support
0	0.55	0.58	0.56	50
1	0.22	0.20	0.21	30
2	0.22	0.21	0.21	24
3	0.21	0.24	0.22	17
4	0.62	0.56	0.59	9
accuracy			0.38	130
macro avg	0.36	0.36	0.36	130
weighted avg	0.37	0.38	0.37	130

Appendix S

Activation Function	ReLU
Loss Function	Cross Entropy Loss
Optimizer	Adam
Training Epochs	200
Batch Size	16
Learning Rate	0.007
Number of Hidden Layers	6
Hidden Layers: Inputs and Outputs	1: (1024) => (512) 2: (512) => (256) 3: (256) => (128) 4: (128) => (64) 5: (64) => (32) 6: (32) => (16)
Batch Normalization	Input Layer: 1024, Hidden Layer 1: 512, Hidden Layer 2: 256, Hidden Layer 3: 128, Hidden Layer 4: 64, Hidden Layer 5: 32, Hidden Layer 5: 16
Early Stopping Patience	-
Drop Out Rate	-
Last Epoch Information	Epoch 200: | Train Loss: 0.33242 | Val Loss: 2.11070 | Train Acc: 96.667| Val Acc: 44.231
Test Accuracy	42%

Output table screen shot:

	precision	recall	f1-score	support
0	0.52	0.72	0.61	50
1	0.24	0.17	0.20	30
2	0.12	0.08	0.10	24
3	0.39	0.41	0.40	17
4	0.67	0.44	0.53	9
accuracy			0.42	130
macro avg	0.39	0.37	0.37	130
weighted avg	0.38	0.42	0.39	130

Appendix T

Activation Function	ReLU
Loss Function	Cross Entropy Loss
Optimizer	Adam
Training Epochs	200
Batch Size	16
Learning Rate	0.007
Number of Hidden Layers	6
Hidden Layers: Inputs and Outputs	1: (1024) => (512) 2: (512) => (256) 3: (256) => (128) 4: (128) => (64) 5: (64) => (32) 6: (32) => (16)
Batch Normalization	Input Layer: 1024, Hidden Layer 1: 512, Hidden Layer 2: 256, Hidden Layer 3: 128, Hidden Layer 4: 64, Hidden Layer 5: 32, Hidden Layer 5: 16
Early Stopping Patience	-
Drop Out Rate	0.2
Last Epoch Information	Epoch 200: | Train Loss: 0.57295 | Val Loss: 1.78169 | Train Acc: 96.667| Val Acc: 44.231
Test Accuracy	42%

Output table screen shot:

	precision	recall	f1-score	support
0	0.53	0.64	0.58	50
1	0.29	0.30	0.30	30
2	0.17	0.08	0.11	24
3	0.41	0.53	0.46	17
4	0.60	0.33	0.43	9
accuracy			0.42	130
macro avg	0.40	0.38	0.38	130
weighted avg	0.40	0.42	0.40	130

Appendix U

Activation Function	ReLU
Loss Function	Cross Entropy Loss
Optimizer	SGD, weight decay = 0.01
Training Epochs	200
Batch Size	16
Learning Rate	0.007
Number of Hidden Layers	6
Hidden Layers: Inputs and Outputs	1: (1024) => (512) 2: (512) => (256) 3: (256) => (128) 4: (128) => (64) 5: (64) => (32) 6: (32) => (16)
Batch Normalization	Input Layer: 1024, Hidden Layer 1: 512, Hidden Layer 2: 256, Hidden Layer 3: 128, Hidden Layer 4: 64, Hidden Layer 5: 32, Hidden Layer 5: 16
Early Stopping Patience	-
Drop Out Rate	0.2
Last Epoch Information	Epoch 200: | Train Loss: 1.15199 | Val Loss: 1.32413 | Train Acc: 53.333| Val Acc: 44.231
Test Accuracy	46%

Output table screen shot:

	precision	recall	f1-score	support
0	0.62	0.66	0.64	50
1	0.39	0.40	0.39	30
2	0.25	0.29	0.27	24
3	0.44	0.47	0.46	17
4	0.00	0.00	0.00	9
accuracy			0.46	130
macro avg	0.34	0.36	0.35	130
weighted avg	0.43	0.46	0.45	130

Appendix V

Activation Function	ReLU
Loss Function	Cross Entropy Loss
Optimizer	SGD, weight decay = 0.01
Training Epochs	150
Batch Size	16
Learning Rate	0.0007
Number of Hidden Layers	3
Hidden Layers: Inputs and Outputs	1: (512) => (256) 2: (256) => (128) 3: (128) => (64)
Batch Normalization	Input Layer: 512, Hidden Layer 1: 256, Hidden Layer 2: 128, Hidden Layer 3: 64
Early Stopping Patience	-
Drop Out Rate	0.2
Last Epoch Information	Epoch 150: | Train Loss: 0.93559 | Val Loss: 1.34033 | Train Acc: 83.333| Val Acc: 46.154
Test Accuracy	42%

Output table screen shot:

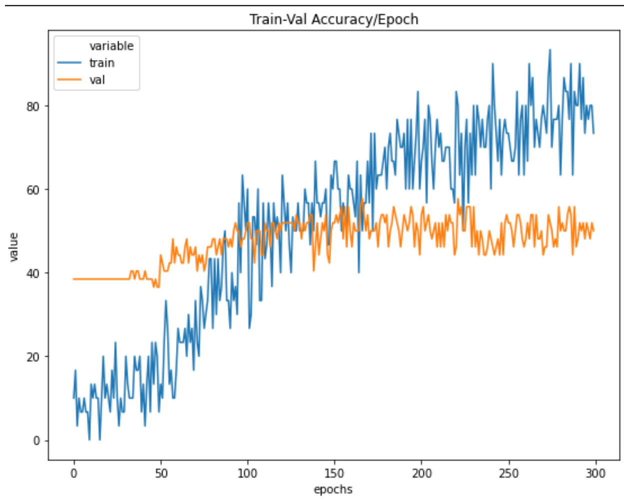
	precision	recall	f1-score	support
0	0.57	0.66	0.61	50
1	0.30	0.27	0.28	30
2	0.17	0.12	0.14	24
3	0.29	0.35	0.32	17
4	0.67	0.44	0.53	9
accuracy			0.42	130
macro avg	0.40	0.37	0.38	130
weighted avg	0.40	0.42	0.40	130

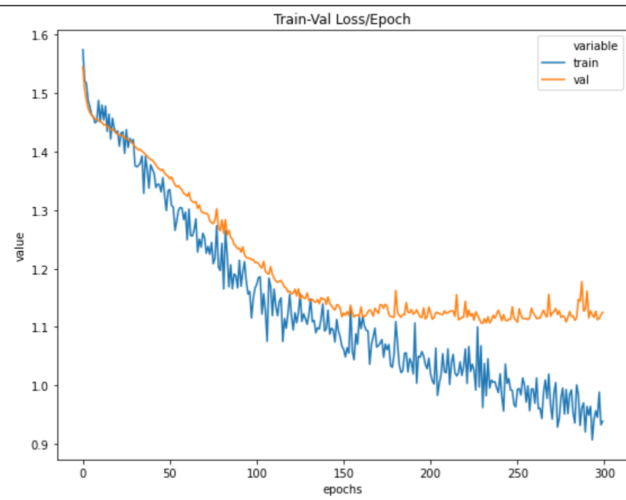
Appendix W

Activation Function	ReLU
Loss Function	Cross Entropy Loss
Optimizer	SGD, weight decay = 0.01
Training Epochs	300
Batch Size	16
Learning Rate	0.007
Number of Hidden Layers	1
Hidden Layers: Inputs and Outputs	1: (512) => (256)
Batch Normalization	Input Layer: 512, Hidden Layer 1: 256
Early Stopping Patience	-
Drop Out Rate	-
Last Epoch Information	Epoch 300: | Train Loss: 0.93910 | Val Loss: 1.12516 | Train Acc: 73.333| Val Acc: 50.000
Test Accuracy	45%

Output table screen shot:

	precision	recall	f1-score	support
0	0.60	0.72	0.65	50
1	0.25	0.20	0.22	30
2	0.27	0.38	0.32	24
3	0.50	0.12	0.19	17
4	0.67	0.67	0.67	9
accuracy			0.45	130
macro avg	0.46	0.42	0.41	130
weighted avg	0.45	0.45	0.43	130





Appendix X

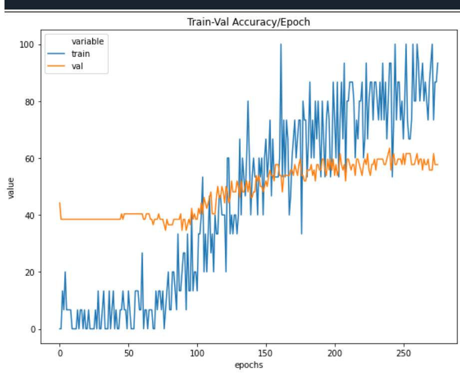
Activation Function	ReLU
Loss Function	Cross Entropy Loss
Optimizer	SGD, weight decay = 0.0001
Training Epochs	10000

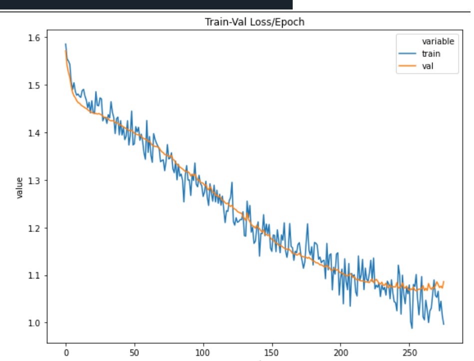
Predicting Student’s Weekend Alcohol Consumption

Batch Size	32
Learning Rate	0.007
Number of Hidden Layers	1
Hidden Layers: Inputs and Outputs	1: (512) => (256)
Batch Normalization	-
Early Stopping Patience	20
Drop Out Rate	0.1
Last Epoch Information	Epoch 276: | Train Loss: 0.99649 | Val Loss: 1.08610 | Train Acc: 93.333| Val Acc: 57.692
Test Accuracy	43%

Output table screen shot:

	precision	recall	f1-score	support
0	0.55	0.74	0.63	50
1	0.24	0.20	0.22	30
2	0.16	0.12	0.14	24
3	0.38	0.29	0.33	17
4	0.83	0.56	0.67	9
accuracy			0.43	130
macro avg	0.43	0.38	0.40	130
weighted avg	0.40	0.43	0.41	130





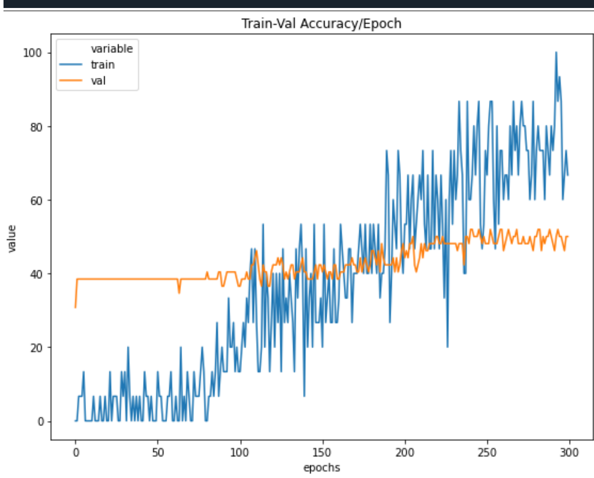
Appendix Y

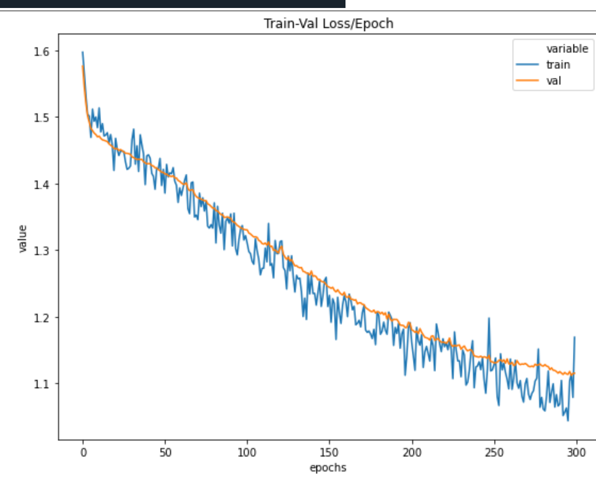
Activation Function	ReLU
Loss Function	Cross Entropy Loss
Optimizer	SGD, weight decay = 0.001
Training Epochs	300
Batch Size	32
Learning Rate	0.007
Number of Hidden Layers	1
Hidden Layers: Inputs and Outputs	1: (512) => (256)
Batch Normalization	-
Early Stopping Patience	-
Drop Out Rate	0.1
Last Epoch Information	Epoch 300: | Train Loss: 1.16879 | Val Loss: 1.11471 | Train Acc: 66.667| Val Acc: 50.000
Test Accuracy	48%

EPOCHS = 300

Output table screen shot:

	precision	recall	f1-score	support
0	0.54	0.88	0.67	50
1	0.35	0.20	0.26	30
2	0.20	0.08	0.12	24
3	0.44	0.41	0.42	17
4	0.80	0.44	0.57	9
accuracy			0.48	130
macro avg	0.47	0.40	0.41	130
weighted avg	0.44	0.48	0.43	130





Appendix Z

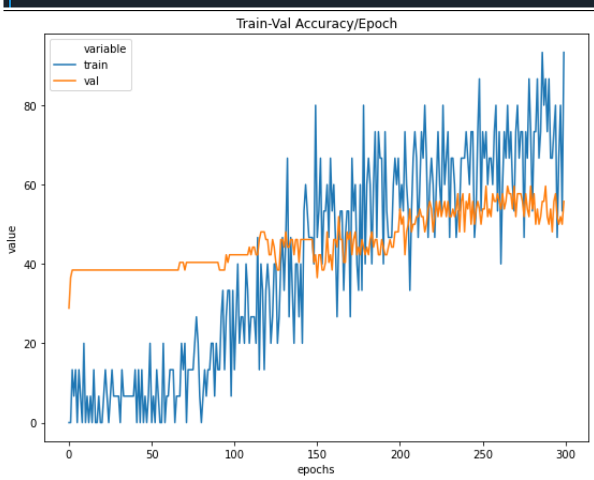
Activation Function	ReLU
Loss Function	Cross Entropy Loss

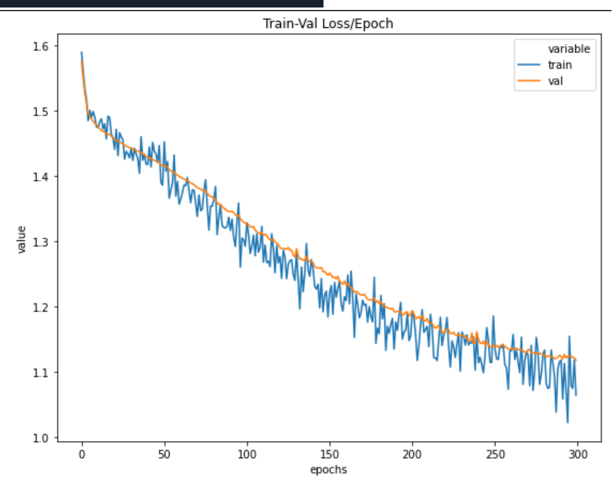
Predicting Student's Weekend Alcohol Consumption

Optimizer	SGD, weight decay = 0.001
Training Epochs	300
Batch Size	32
Learning Rate	0.007
Number of Hidden Layers	1
Hidden Layers: Inputs and Outputs	1: (512) => (256)
Batch Normalization	-
Early Stopping Patience	-
Drop Out Rate	0.2
Last Epoch Information	Epoch 300: | Train Loss: 1.06376 | Val Loss: 1.11708 | Train Acc: 93.333| Val Acc: 55.769
Test Accuracy	47%

Output table screen shot:

	precision	recall	f1-score	support
0	0.54	0.86	0.67	50
1	0.19	0.10	0.13	30
2	0.23	0.12	0.16	24
3	0.44	0.41	0.42	17
4	0.83	0.56	0.67	9
accuracy			0.47	130
macro avg	0.45	0.41	0.41	130
weighted avg	0.41	0.47	0.42	130





Appendix AA

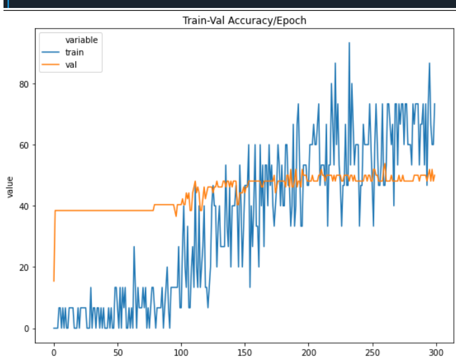
Activation Function	ReLU
Loss Function	Cross Entropy Loss
Optimizer	SGD, weight decay = 0.001
Training Epochs	300
Batch Size	32
Learning Rate	0.007
Number of Hidden Layers	1
Hidden Layers: Inputs and Outputs	1: (512) => (256)
Batch Normalization	-
Early Stopping Patience	-
Drop Out Rate	0.3
Last Epoch Information	Epoch 300: | Train Loss: 1.13414 | Val Loss: 1.13454 | Train Acc: 73.333| Val Acc: 50.000
Test Accuracy	48%

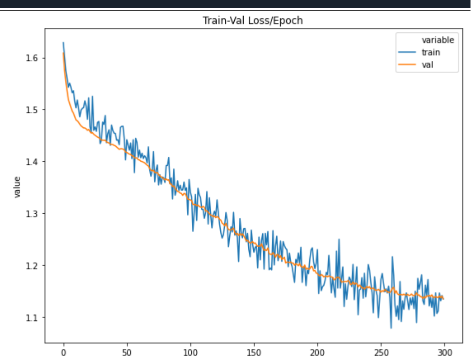
Output table screen shot:

```
              precision    recall  f1-score   support

           0       0.56      0.88      0.68        50
           1       0.14      0.07      0.09        30
           2       0.23      0.12      0.16        24
           3       0.41      0.41      0.41        17
           4       0.86      0.67      0.75         9

    accuracy                           0.48       130
   macro avg       0.44      0.43      0.42       130
weighted avg       0.40      0.48      0.42       130
```



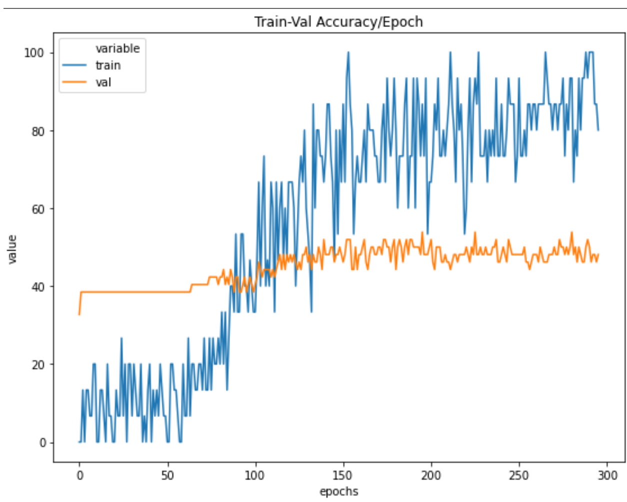


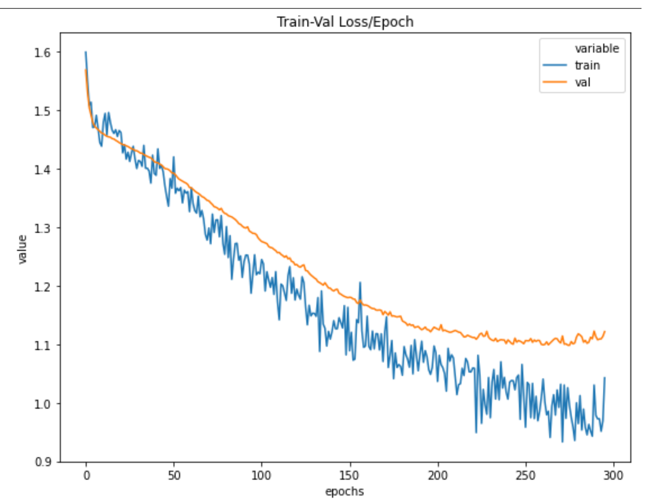
Appendix AB

Activation Function	ReLU
Loss Function	Cross Entropy Loss
Optimizer	SGD, weight decay = 0.001
Training Epochs	10000
Batch Size	32
Learning Rate	0.007
Number of Hidden Layers	1
Hidden Layers: Inputs and Outputs	1: (512) => (256)
Batch Normalization	-
Early Stopping Patience	20
Drop Out Rate	0.1
Last Epoch Information	Epoch 296: | Train Loss: 1.04244 | Val Loss: 1.12140 | Train Acc: 80.000| Val Acc: 48.077
Test Accuracy	52%

Output table screen shot:

	precision	recall	f1-score	support
0	0.59	0.86	0.70	50
1	0.53	0.27	0.36	30
2	0.30	0.33	0.31	24
3	0.38	0.18	0.24	17
4	0.86	0.67	0.75	9
accuracy			0.52	130
macro avg	0.53	0.46	0.47	130
weighted avg	0.51	0.52	0.49	130





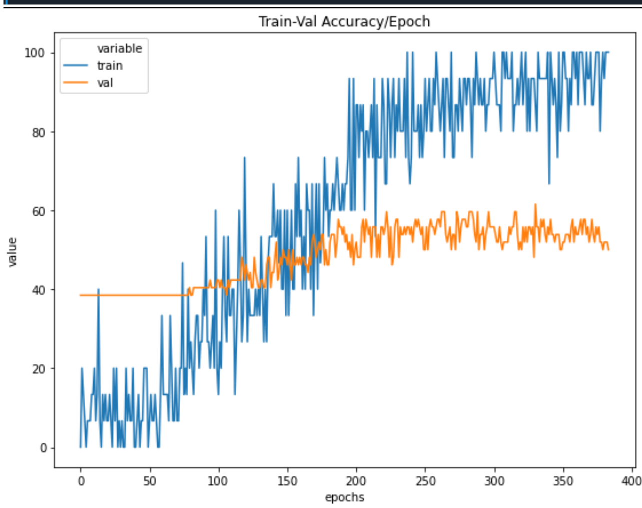
Appendix AC

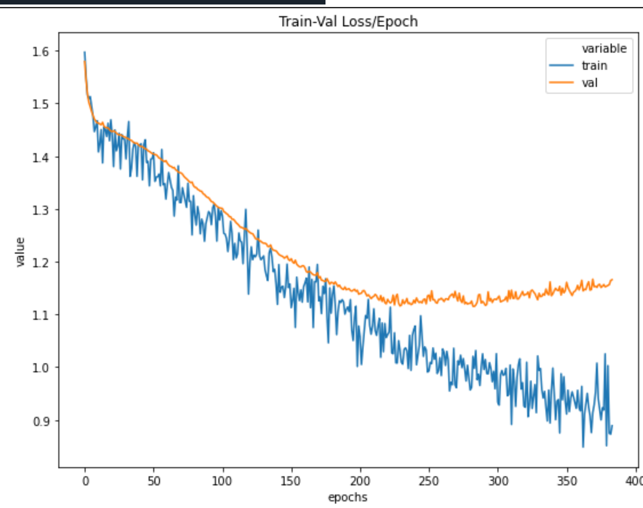
Activation Function	ReLU
Loss Function	Cross Entropy Loss

Optimizer	SGD, weight decay = 0.001
Training Epochs	10000
Batch Size	32
Learning Rate	0.007
Number of Hidden Layers	1
Hidden Layers: Inputs and Outputs	1: (512) => (256)
Batch Normalization	-
Early Stopping Patience	100
Drop Out Rate	0.1
Last Epoch Information	Epoch 384: | Train Loss: 0.88907 | Val Loss: 1.16580 | Train Acc: 100.000| Val Acc: 50.000
Test Accuracy	47%

Output table screen shot:

	precision	recall	f1-score	support
0	0.57	0.72	0.64	50
1	0.35	0.27	0.30	30
2	0.26	0.25	0.26	24
3	0.40	0.35	0.38	17
4	0.83	0.56	0.67	9
accuracy			0.47	130
macro avg	0.48	0.43	0.45	130
weighted avg	0.46	0.47	0.46	130





Appendix AD

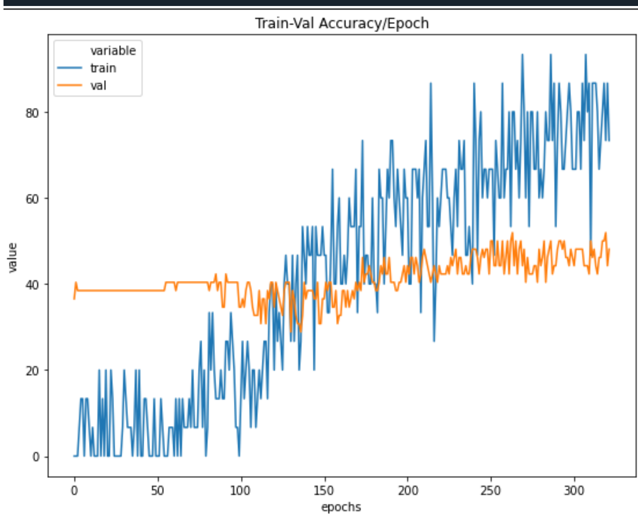
Activation Function	LeakyReLU
Loss Function	Cross Entropy Loss
Optimizer	SGD, weight decay = 0.001
Training Epochs	10000
Batch Size	32

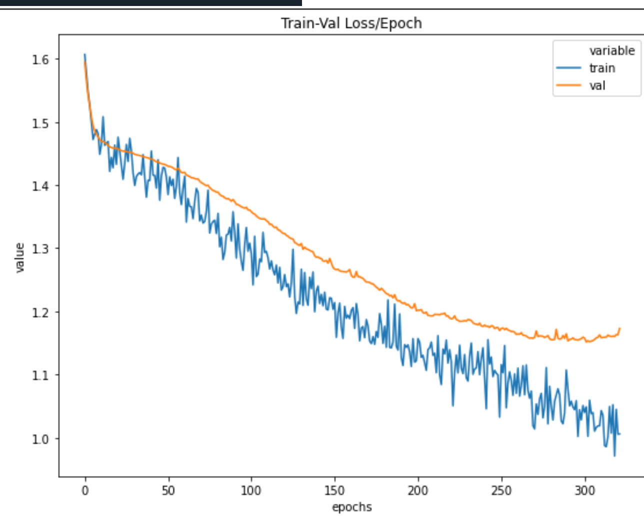
Predicting Student's Weekend Alcohol Consumption

Learning Rate	0.007
Number of Hidden Layers	1
Hidden Layers: Inputs and Outputs	1: (512) => (256)
Batch Normalization	-
Early Stopping Patience	20
Drop Out Rate	0.1
Last Epoch Information	Epoch 322: | Train Loss: 1.00613 | Val Loss: 1.17306 | Train Acc: 73.333| Val Acc: 48.077
Test Accuracy	45%

Output table screen shot:

	precision	recall	f1-score	support
0	0.57	0.84	0.68	50
1	0.18	0.13	0.15	30
2	0.23	0.12	0.16	24
3	0.31	0.29	0.30	17
4	0.80	0.44	0.57	9
accuracy			0.45	130
macro avg	0.42	0.37	0.37	130
weighted avg	0.40	0.45	0.41	130



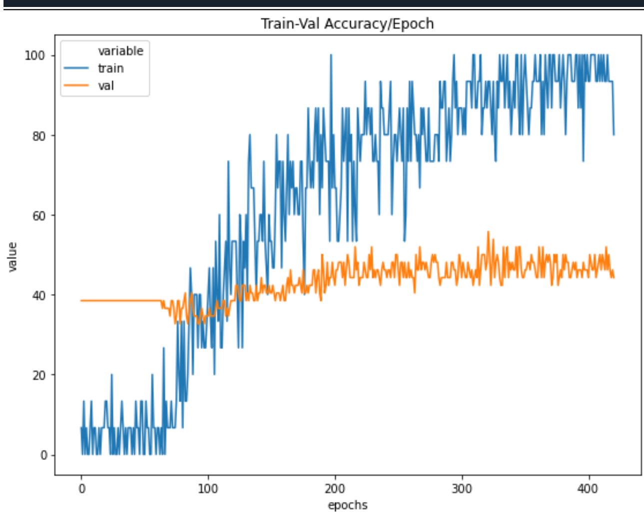


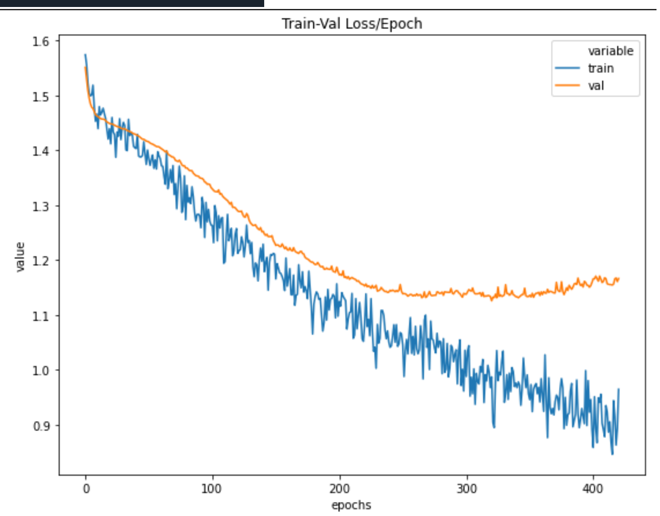
Appendix AE

Activation Function	LeakyReLU
Loss Function	Cross Entropy Loss
Optimizer	SGD, weight decay = 0.001
Training Epochs	10000
Batch Size	32
Learning Rate	0.007
Number of Hidden Layers	1
Hidden Layers: Inputs and Outputs	1: (512) => (256)
Batch Normalization	-
Early Stopping Patience	100
Drop Out Rate	0.1
Last Epoch Information	Epoch 421: | Train Loss: 0.96482 | Val Loss: 1.16665 | Train Acc: 80.000| Val Acc: 44.231
Test Accuracy	48%

Output table screen shot:

	precision	recall	f1-score	support
0	0.57	0.78	0.66	50
1	0.37	0.23	0.29	30
2	0.25	0.29	0.27	24
3	0.57	0.24	0.33	17
4	0.75	0.67	0.71	9
accuracy			0.48	130
macro avg	0.50	0.44	0.45	130
weighted avg	0.48	0.48	0.46	130





Appendix AF

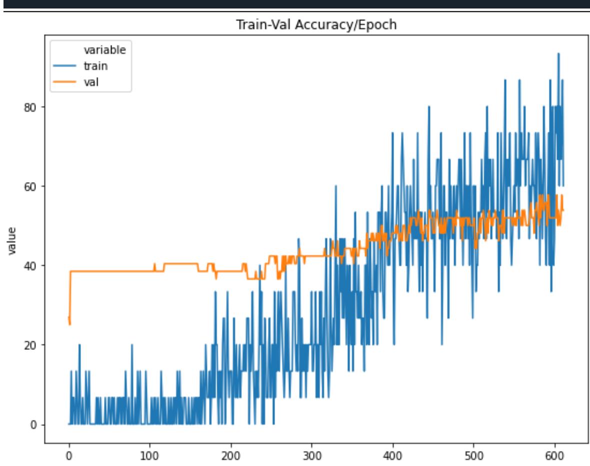
Activation Function	Hardswish
Loss Function	Cross Entropy Loss
Optimizer	SGD, weight decay = 0.001

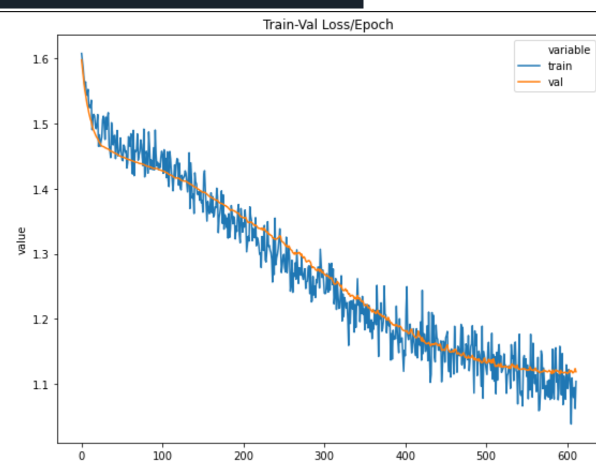
Predicting Student's Weekend Alcohol Consumption

Training Epochs	10000
Batch Size	32
Learning Rate	0.007
Number of Hidden Layers	1
Hidden Layers: Inputs and Outputs	1: (512) => (256)
Batch Normalization	-
Early Stopping Patience	20
Drop Out Rate	0.1
Last Epoch Information	Epoch 612: | Train Loss: 1.10404 | Val Loss: 1.11871 | Train Acc: 60.000| Val Acc: 53.846
Test Accuracy	50%

Output table screen shot:

	precision	recall	f1-score	support
0	0.57	0.84	0.68	50
1	0.27	0.20	0.23	30
2	0.31	0.21	0.25	24
3	0.60	0.35	0.44	17
4	0.75	0.67	0.71	9
accuracy			0.50	130
macro avg	0.50	0.45	0.46	130
weighted avg	0.47	0.50	0.47	130





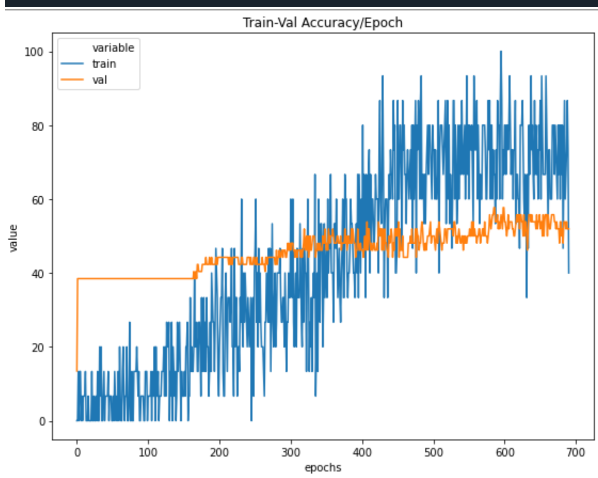
Appendix AG

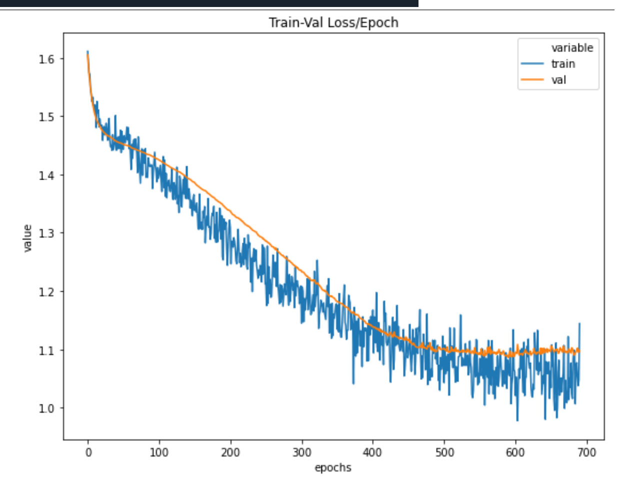
Activation Function	Hardswish
Loss Function	Cross Entropy Loss
Optimizer	SGD, weight decay = 0.001
Training Epochs	10000
Batch Size	32
Learning Rate	0.007

Number of Hidden Layers	1
Hidden Layers: Inputs and Outputs	1: (512) => (256)
Batch Normalization	-
Early Stopping Patience	100
Drop Out Rate	0.1
Last Epoch Information	Epoch 691: | Train Loss: 1.14419 | Val Loss: 1.09624 | Train Acc: 40.000| Val Acc: 51.923
Test Accuracy	48%

Output table screen shot:

	precision	recall	f1-score	support
0	0.58	0.82	0.68	50
1	0.31	0.17	0.22	30
2	0.27	0.29	0.28	24
3	0.40	0.24	0.30	17
4	0.86	0.67	0.75	9
accuracy			0.48	130
macro avg	0.48	0.44	0.44	130
weighted avg	0.46	0.48	0.45	130





Appendix AH

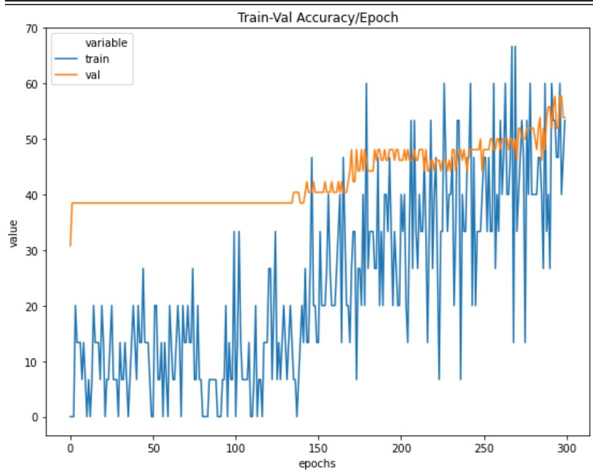
Activation Function	ReLU
Loss Function	Cross Entropy Loss
Optimizer	SGD, weight decay = 0.001
Training Epochs	300
Batch Size	32
Learning Rate	0.007
Number of Hidden Layers	2
Hidden Layers: Inputs and Outputs	1: (512) => (256) 2: (256) => (128)
Batch Normalization	-
Early Stopping Patience	-

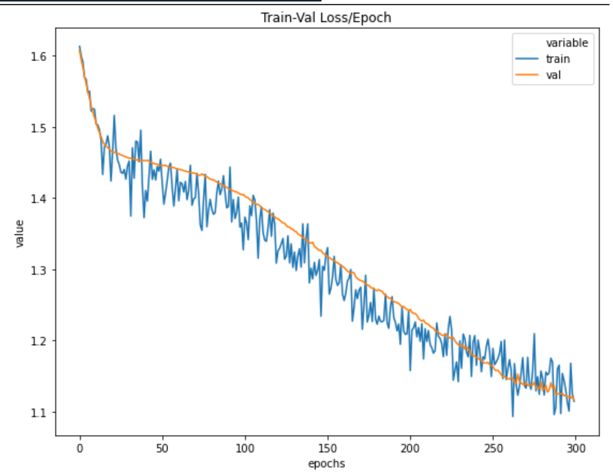
Predicting Student's Weekend Alcohol Consumption

Drop Out Rate	0.1
Last Epoch Information	Epoch 300: | Train Loss: 1.11420 | Val Loss: 1.11584 | Train Acc: 53.333| Val Acc: 53.846
Test Accuracy	49%

Output table screen shot:

	precision	recall	f1-score	support
0	0.56	0.90	0.69	50
1	0.25	0.03	0.06	30
2	0.28	0.33	0.30	24
3	0.50	0.29	0.37	17
4	0.83	0.56	0.67	9
accuracy			0.49	130
macro avg	0.48	0.42	0.42	130
weighted avg	0.45	0.49	0.43	130



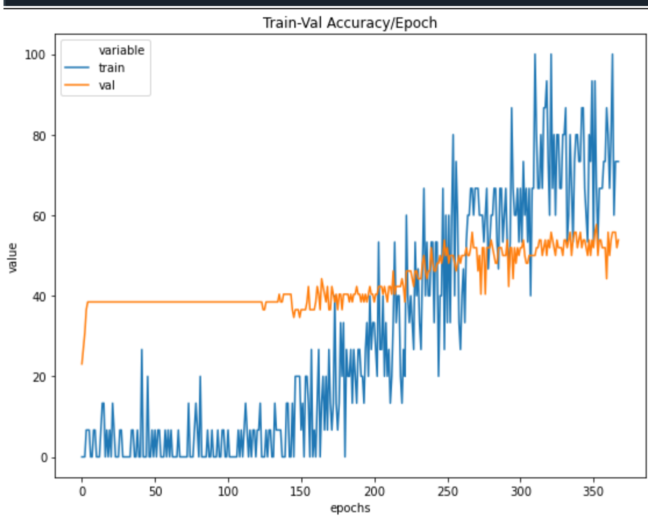


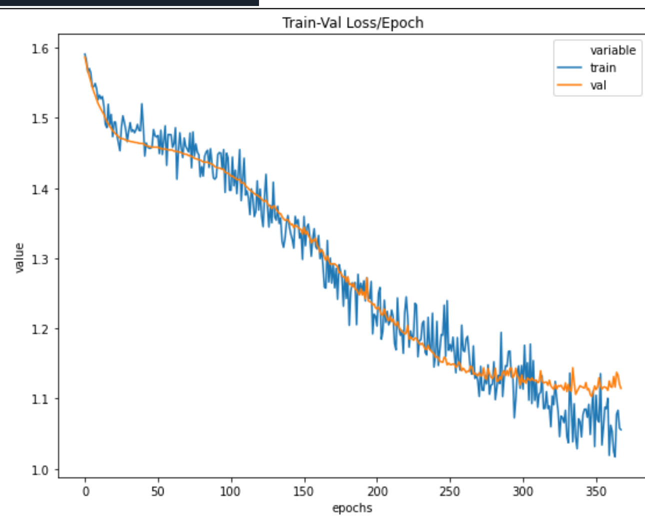
Appendix AI

Activation Function	ReLU
Loss Function	Cross Entropy Loss
Optimizer	SGD, weight decay = 0.001
Training Epochs	10000
Batch Size	32
Learning Rate	0.007
Number of Hidden Layers	2
Hidden Layers: Inputs and Outputs	1: (512) => (256) 2: (256) => (128)
Batch Normalization	-
Early Stopping Patience	20
Drop Out Rate	0.1
Last Epoch Information	Epoch 368: | Train Loss: 1.05560 | Val Loss: 1.11449 | Train Acc: 73.333| Val Acc: 53.846
Test Accuracy	48%

Output table screen shot:

	precision	recall	f1-score	support
0	0.59	0.78	0.67	50
1	0.36	0.33	0.34	30
2	0.22	0.17	0.19	24
3	0.46	0.35	0.40	17
4	0.80	0.44	0.57	9
accuracy			0.48	130
macro avg	0.49	0.42	0.44	130
weighted avg	0.47	0.48	0.47	130



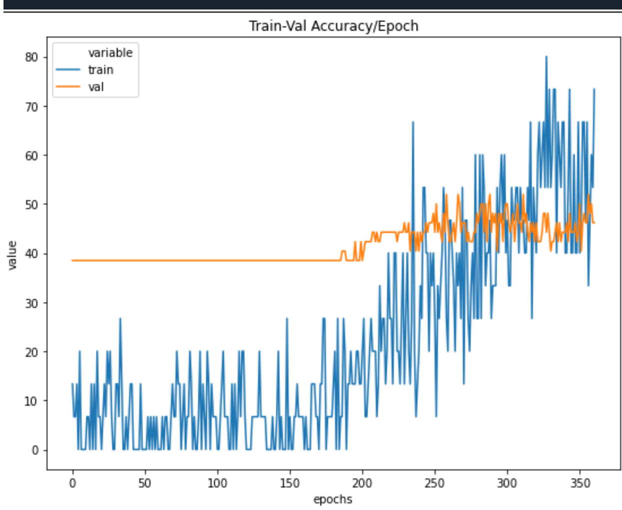


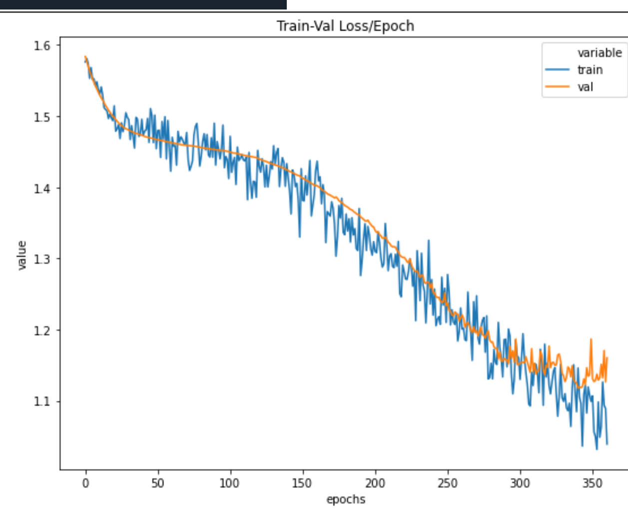
Appendix AJ

Activation Function	ReLU
Loss Function	Cross Entropy Loss
Optimizer	SGD, weight decay = 0.001
Training Epochs	10000
Batch Size	32
Learning Rate	0.007
Number of Hidden Layers	3
Hidden Layers: Inputs and Outputs	1: (512) => (256) 2: (256) => (128) 3: (128) => (64)
Batch Normalization	-
Early Stopping Patience	20
Drop Out Rate	0.1
Last Epoch Information	Epoch 361: | Train Loss: 1.03909 | Val Loss: 1.16008 | Train Acc: 73.333| Val Acc: 46.154
Test Accuracy	50%

Output table screen shot:

	precision	recall	f1-score	support
0	0.57	0.88	0.69	50
1	0.46	0.20	0.28	30
2	0.27	0.17	0.21	24
3	0.44	0.65	0.52	17
4	0.00	0.00	0.00	9
accuracy			0.50	130
macro avg	0.35	0.38	0.34	130
weighted avg	0.43	0.50	0.44	130





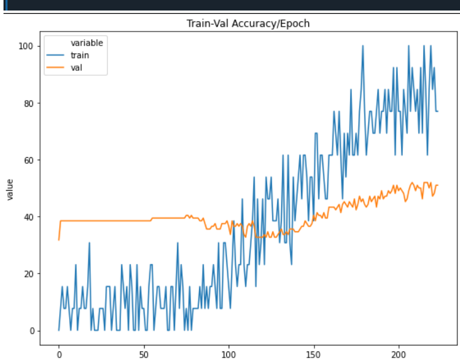
Appendix AK

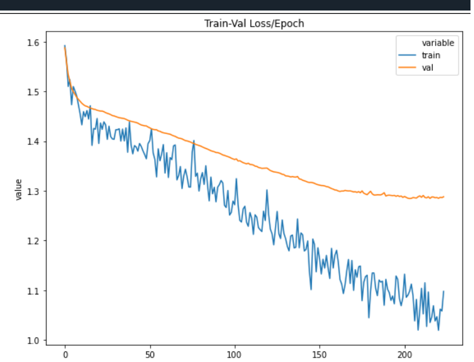
Data Split: Train – 64%, Validation – 16% ,Test: 20%

Activation Function	ReLU
Loss Function	Cross Entropy Loss
Optimizer	SGD, weight decay = 0.001
Training Epochs	10000
Batch Size	32
Learning Rate	0.007
Number of Hidden Layers	1
Hidden Layers: Inputs and Outputs	1: (512) => (256)
Batch Normalization	-
Early Stopping Patience	20
Drop Out Rate	0.1
Last Epoch Information	Epoch 224: | Train Loss: 1.09780 | Val Loss: 1.28799 | Train Acc: 76.923| Val Acc: 50.962
Test Accuracy	44%

Output table screen shot:

	precision	recall	f1-score	support
0	0.59	0.80	0.68	50
1	0.21	0.20	0.21	30
2	0.08	0.04	0.06	24
3	0.38	0.35	0.36	17
4	0.67	0.44	0.53	9
accuracy			0.44	130
macro avg	0.39	0.37	0.37	130
weighted avg	0.39	0.44	0.40	130





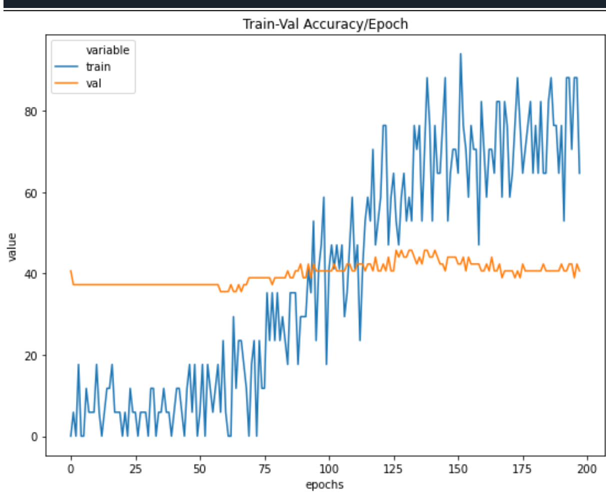
Appendix AL

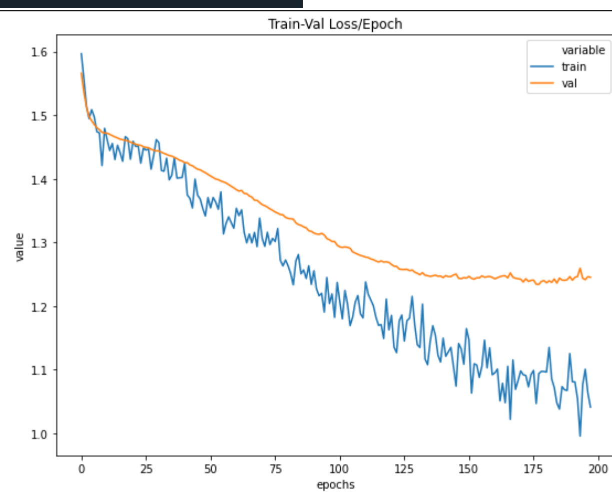
Data Split: Train – 81%, Validation – 9% ,Test: 10%

Activation Function	ReLU
Loss Function	Cross Entropy Loss
Optimizer	SGD, weight decay = 0.001
Training Epochs	10000
Batch Size	32
Learning Rate	0.007
Number of Hidden Layers	1
Hidden Layers: Inputs and Outputs	1: (512) => (256)
Batch Normalization	-
Early Stopping Patience	20
Drop Out Rate	0.1
Last Epoch Information	Epoch 198: | Train Loss: 1.04097 | Val Loss: 1.24517 | Train Acc: 64.706| Val Acc: 40.678
Test Accuracy	40%

Output table screen shot:

	precision	recall	f1-score	support
0	0.55	0.84	0.67	25
1	0.11	0.07	0.08	15
2	0.20	0.17	0.18	12
3	0.14	0.11	0.12	9
4	1.00	0.25	0.40	4
accuracy			0.40	65
macro avg	0.40	0.29	0.29	65
weighted avg	0.36	0.40	0.35	65





Appendix AM

Activation Function	ReLU
Loss Function	Cross Entropy Loss

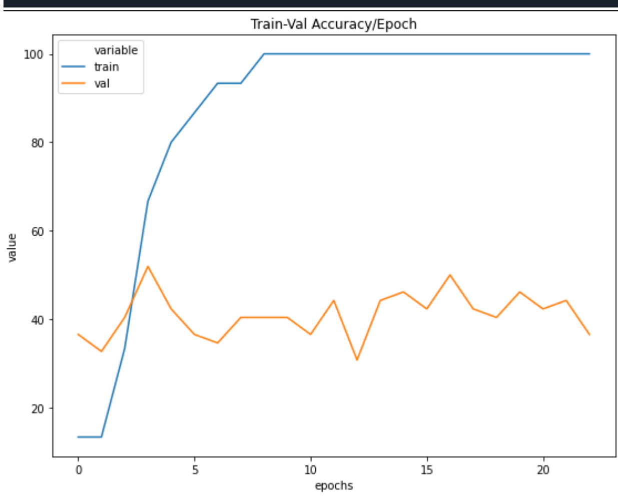
Optimizer	Adam
Training Epochs	10000
Batch Size	32
Learning Rate	0.007
Number of Hidden Layers	1
Hidden Layers: Inputs and Outputs	1: (512) => (256)
Batch Normalization	-
Early Stopping Patience	20
Drop Out Rate	0.1
Last Epoch Information	Epoch 023: | Train Loss: 0.31766 | Val Loss: 2.79718 | Train Acc: 100.000| Val Acc: 36.538
Test Accuracy	42%

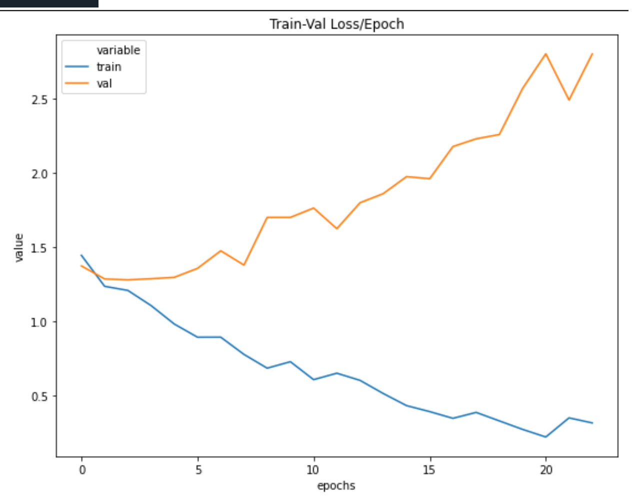
Output table screen shot:

```
              precision    recall  f1-score   support

           0       0.48      0.90      0.63        50
           1       0.60      0.10      0.17        30
           2       0.17      0.04      0.07        24
           3       0.24      0.35      0.29        17
           4       0.00      0.00      0.00         9

    accuracy                           0.42       130
   macro avg       0.30      0.28      0.23       130
weighted avg       0.38      0.42      0.33       130
```





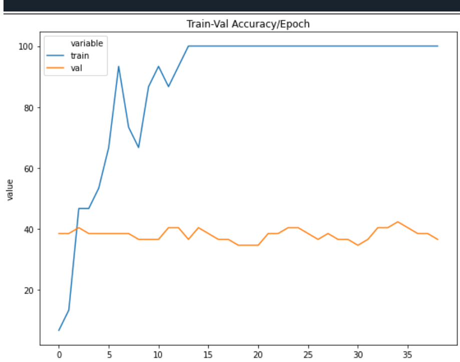
Appendix AN

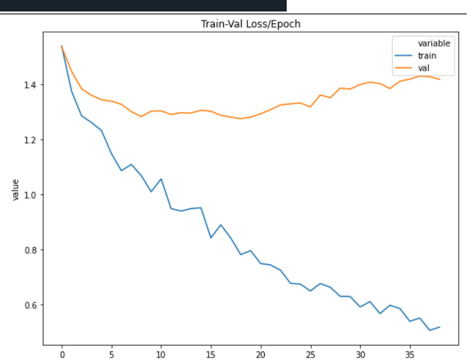
Data Split: Train – 81%, Validation – 9% ,Test: 10%

Activation Function	ReLU
Loss Function	Cross Entropy Loss
Optimizer	SGD, weight decay = 0.01
Training Epochs	10000
Batch Size	32
Learning Rate	0.007
Number of Hidden Layers	1
Hidden Layers: Inputs and Outputs	1: (512) => (256)
Batch Normalization	Input Layer: 512, Hidden Layer 1: 256
Early Stopping Patience	20
Drop Out Rate	0.1
Last Epoch Information	Epoch 039: | Train Loss: 0.51671 | Val Loss: 1.41744 | Train Acc: 100.000| Val Acc: 36.538
Test Accuracy	42%

Output table screen shot:

	precision	recall	f1-score	support
0	0.53	0.84	0.65	50
1	0.26	0.20	0.23	30
2	0.08	0.04	0.06	24
3	0.30	0.18	0.22	17
4	0.60	0.33	0.43	9
accuracy			0.42	130
macro avg	0.35	0.32	0.32	130
weighted avg	0.36	0.42	0.37	130





Appendix AO

Activation Function	ReLU
Loss Function	Cross Entropy Loss
Optimizer	Adam
Training Epochs	300
Batch Size	16
Learning Rate	0.0007
Number of Hidden Layers	1
Hidden Layers: Inputs and Outputs	1: (512) => (256)
Batch Normalization	Input Layer: 512, Hidden Layer 1: 256
Early Stopping Patience	-
Drop Out Rate	-
Last Epoch Information	Epoch 300: | Train Loss: 0.0228 | Val Loss: 3.5525 | Train Acc: 96.667| Val Acc: 38.462
Test Accuracy	38%

Output table screen shot:

	precision	recall	f1-score	support
0	0.49	0.60	0.54	50
1	0.33	0.30	0.32	30
2	0.12	0.08	0.10	24
3	0.33	0.35	0.34	17
4	0.43	0.33	0.38	9
accuracy			0.38	130
macro avg	0.34	0.33	0.33	130
weighted avg	0.36	0.38	0.37	130

Appendix AP

Activation Function	ReLU
Loss Function	Cross Entropy Loss
Optimizer	Adam
Training Epochs	58
Batch Size	16
Learning Rate	0.0038493557916637407
Number of Hidden Layers	1
Hidden Layers: Inputs and Outputs	1: (4589) => (2294)
Batch Normalization	Input Layer: 4589, Hidden Layer 1: 2294
Early Stopping Patience	-
Drop Out Rate	-

Last Epoch Information	Epoch 058: | Train Loss: 0.13564 | Val Loss: 3.74467 | Train Acc: 100.000| Val Acc: 38.462
Test Accuracy	37%

Output table screen shot:

	precision	recall	f1-score	support
0	0.51	0.54	0.52	50
1	0.26	0.20	0.23	30
2	0.22	0.21	0.21	24
3	0.29	0.41	0.34	17
4	0.43	0.33	0.38	9
accuracy			0.37	130
macro avg	0.34	0.34	0.34	130
weighted avg	0.36	0.37	0.36	130

Appendix AQ

Activation Function	ReLU
Loss Function	Cross Entropy Loss
Optimizer	Adam
Training Epochs	300
Batch Size	16
Learning Rate	0.2
Number of Hidden Layers	3
Hidden Layers: Inputs and Outputs	1: (256) => (128) 2: (128) => (64) 3: (64) => (32)
Batch Normalization	Input Layer: 256, Hidden Layer 1: 128, Hidden Layer 2: 64, Hidden Layer 3: 32
Early Stopping Patience	-
Drop Out Rate	-
Last Epoch Information	Epoch 300: | Train Loss: 0.20646 | Val Loss: 6.52117 | Train Acc: 100.000| Val Acc: 28.846
Test Accuracy	39%

Output table screen shot:

	precision	recall	f1-score	support
0	0.62	0.66	0.64	50
1	0.38	0.20	0.26	30
2	0.11	0.17	0.13	24
3	0.29	0.35	0.32	17
4	0.50	0.22	0.31	9
accuracy			0.39	130
macro avg	0.38	0.32	0.33	130
weighted avg	0.42	0.39	0.39	130

Appendix AR

Activation Function	ReLU
Loss Function	Cross Entropy Loss
Optimizer	Adam
Training Epochs	80
Batch Size	16
Learning Rate	0.19727524535520638
Number of Hidden Layers	2
Hidden Layers: Inputs and Outputs	1: (2962) => (1481) 2: (1481) => (740)
Batch Normalization	Input Layer: 2962, Hidden Layer 1: 1481, Hidden Layer 2: 740
Early Stopping Patience	-
Drop Out Rate	-
Last Epoch Information	Epoch 080: | Train Loss: 0.92283 | Val Loss: 1.61058 | Train Acc: 63.333| Val Acc: 50.000
Test Accuracy	44%

Output table screen shot:

	precision	recall	f1-score	support
0	0.58	0.64	0.61	50
1	0.30	0.47	0.36	30
2	0.14	0.04	0.06	24
3	0.40	0.24	0.30	17
4	0.55	0.67	0.60	9
accuracy			0.44	130
macro avg	0.39	0.41	0.39	130
weighted avg	0.41	0.44	0.41	130

Appendix AS

Activation Function	ReLU
Loss Function	Cross Entropy Loss
Optimizer	SGD
Training Epochs	76
Batch Size	105
Learning Rate	0.013920344480894973
Number of Hidden Layers	3
Hidden Layers: Inputs and Outputs	1: (6595) => (3298) 2: (3298) => (1649) 3: (1649) => (824)
Batch Normalization	Input Layer: 6595, Hidden Layer 1: 3298, Hidden Layer 2: 1649, Hidden Layer 3: 824
Early Stopping Patience	-
Weight Decay	0.8516583412890875
Drop Out Rate	0.2934481075639418
Last Epoch Information	Epoch 076: | Train Loss: 1.01109 | Val Loss: 1.36337 | Train Acc: 100.000| Val Acc: 44.231
Test Accuracy	42%

Output table screen shot:

```
              precision    recall  f1-score   support

           0       0.57      0.66      0.61        50
           1       0.29      0.27      0.28        30
           2       0.19      0.17      0.18        24
           3       0.31      0.29      0.30        17
           4       0.57      0.44      0.50         9

    accuracy                           0.42       130
   macro avg       0.39      0.37      0.37       130
weighted avg       0.40      0.42      0.41       130
```

Appendix AT

Activation Function	ReLU
Loss Function	Cross Entropy Loss
Optimizer	Adam
Training Epochs	260
Batch Size	125
Learning Rate	0.00015054303711951883
Number of Hidden Layers	3
Hidden Layers: Inputs and Outputs	1: (4749) => (3298) 2: (3298) => (1649) 3: (1649) => (824)
Batch Normalization	Input Layer: 4749, Hidden Layer 1: 3298, Hidden Layer 2: 1649, Hidden Layer 3: 824
Early Stopping Patience	-
Weight Decay	0.35890558069944384
Drop Out Rate	0.036093945652246474
Last Epoch Information	Epoch 260: | Train Loss: 0.12290 | Val Loss: 1.51080 | Train Acc: 100.000| Val Acc: 36.538
Test Accuracy	48%

Output table screen shot:

	precision	recall	f1-score	support
0	0.59	0.84	0.69	50
1	0.32	0.23	0.27	30
2	0.27	0.17	0.21	24
3	0.35	0.35	0.35	17
4	0.60	0.33	0.43	9
accuracy			0.48	130
macro avg	0.43	0.39	0.39	130
weighted avg	0.44	0.48	0.44	130

Appendix AU

Activation Function	ReLU
Loss Function	Cross Entropy Loss
Optimizer	SGD
Training Epochs	52
Batch Size	74
Learning Rate	0.00948236203858607
Number of Hidden Layers	2
Hidden Layers: Inputs and Outputs	1: (6068) => (3034) 2: (3034) => (1517)
Batch Normalization	Input Layer: 6068, Hidden Layer 1: 3034, Hidden Layer 2: 1517
Early Stopping Patience	-

Weight Decay	0.16317779064178467
Drop Out Rate	0.1314163376390934
Last Epoch Information	Epoch 052: | Train Loss: 0.38496 | Val Loss: 1.38251 | Train Acc: 100.000| Val Acc: 34.615
Test Accuracy	45%

Output table screen shot:

	precision	recall	f1-score	support
0	0.63	0.72	0.67	50
1	0.30	0.23	0.26	30
2	0.32	0.33	0.33	24
3	0.24	0.24	0.24	17
4	0.50	0.44	0.47	9
accuracy			0.45	130
macro avg	0.40	0.39	0.39	130
weighted avg	0.44	0.45	0.44	130

Appendix AV

Activation Function	ReLU
Loss Function	Cross Entropy Loss
Optimizer	SGD
Training Epochs	135
Batch Size	19
Learning Rate	0.007146503748942547
Number of Hidden Layers	1
Hidden Layers: Inputs and Outputs	1: (7877) => (3939)
Batch Normalization	Input Layer: 7877, Hidden Layer 1: 3939
Early Stopping Patience	-
Weight Decay	0.4278550745917246
Drop Out Rate	0.22873144020458944
Last Epoch Information	Epoch 135: | Train Loss: 0.69038 | Val Loss: 1.32140 | Train Acc: 100.000| Val Acc: 48.077
Test Accuracy	43%

Output table screen shot:

```
              precision    recall  f1-score   support

           0       0.56      0.70      0.62        50
           1       0.34      0.33      0.34        30
           2       0.12      0.08      0.10        24
           3       0.38      0.35      0.36        17
           4       0.50      0.33      0.40         9

    accuracy                           0.43       130
   macro avg       0.38      0.36      0.36       130
weighted avg       0.40      0.43      0.41       130
```

Appendix AW

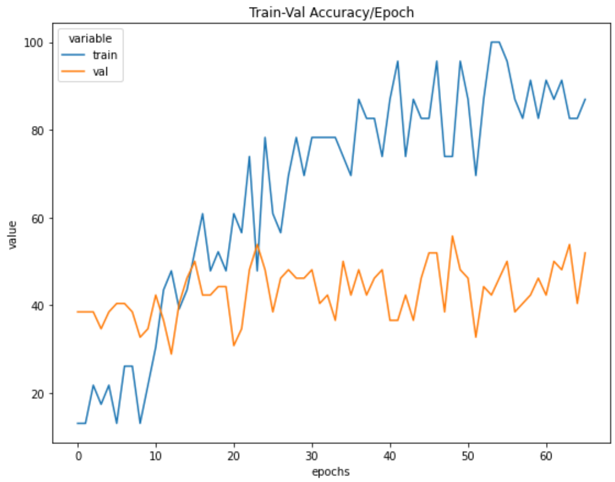
Activation Function	ReLU
Loss Function	Cross Entropy Loss
Optimizer	SGD
Training Epochs	8538
Batch Size	54
Learning Rate	0.00075395244746686675
Number of Hidden Layers	2
Hidden Layers: Inputs and Outputs	1: (8538) => (4269) 2: (4269) => (2134)
Batch Normalization	Input Layer: 8538, Hidden Layer 1: 4269, Hidden Layer 2: 2134
Early Stopping Patience	0.5340685918653861
Weight Decay	0.6080295007545474
Drop Out Rate	0.7786589029651306
Last Epoch Information	Epoch 052: | Train Loss: 0.38496 | Val Loss: 1.38251 | Train Acc: 100.000| Val Acc: 34.615
Test Accuracy	45%

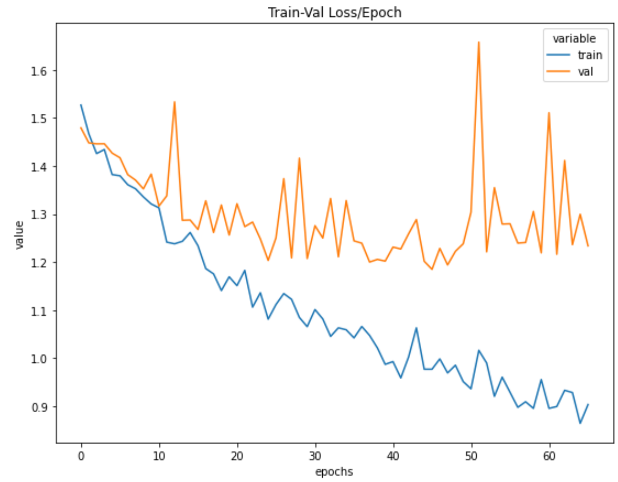
Appendix AX

Activation Function	ReLU
Loss Function	Cross Entropy Loss
Optimizer	SGD
Training Epochs	203
Batch Size	21
Learning Rate	0.019473904989422124
Number of Hidden Layers	1
Hidden Layers: Inputs and Outputs	1: (2624) => (1312)
Batch Normalization	Input Layer: 2624, Hidden Layer 1: 1312
Early Stopping Patience	20
Weight Decay	0.001
Drop Out Rate	0.46674072359143415
Last Epoch Information	Epoch 066: | Train Loss: 0.90298 | Val Loss: 1.23355 | Train Acc: 86.957| Val Acc: 51.923
Test Accuracy	49%

Output table screen shot:

	precision	recall	f1-score	support
0	0.56	0.96	0.71	50
1	0.30	0.10	0.15	30
2	0.21	0.17	0.19	24
3	0.38	0.18	0.24	17
4	0.75	0.67	0.71	9
accuracy			0.49	130
macro avg	0.44	0.41	0.40	130
weighted avg	0.43	0.49	0.42	130





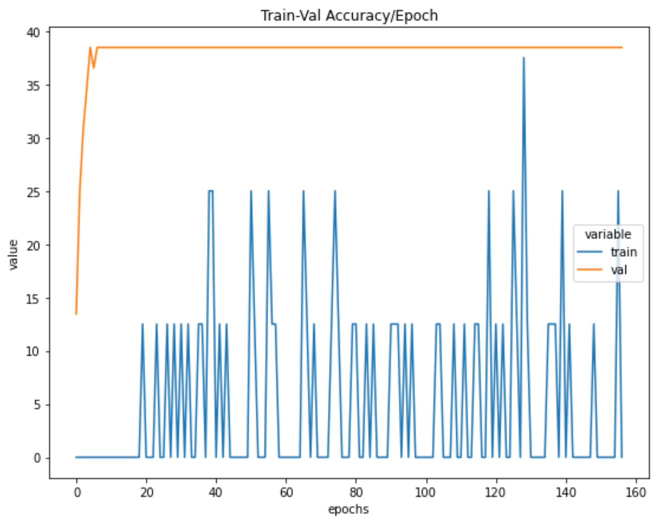
Appendix AZ

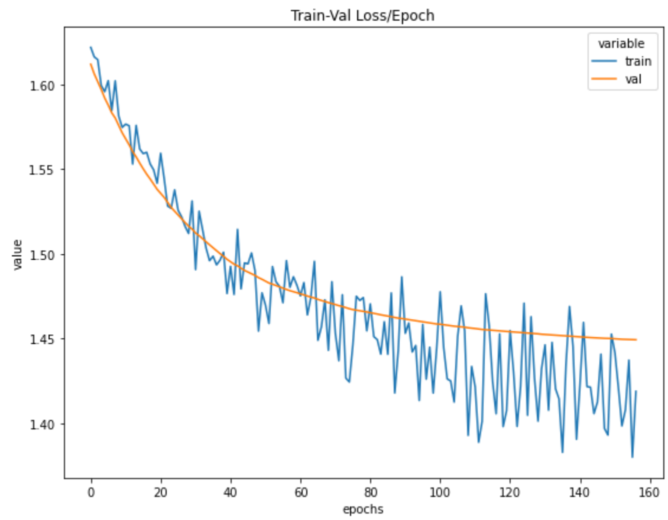
Activation Function	ReLU
Loss Function	Cross Entropy Loss
Optimizer	SGD
Training Epochs	157
Batch Size	63
Learning Rate	0.00022009709124994273
Number of Hidden Layers	1
Hidden Layers: Inputs and Outputs	1: (3798) => (1899)
Batch Normalization	Input Layer: 3798, Hidden Layer 1: 1899
Early Stopping Patience	0.8134192433402521
Weight Decay	0.6086040733570699
Drop Out Rate	0.7594671450530076
Last Epoch Information	Epoch 157: | Train Loss: 1.41896 | Val Loss: 1.44941 | Train Acc: 0.000| Val Acc: 38.46
Test Accuracy	38%

Output table screen shot:

0	0.38	1.00	0.56	50
1	0.00	0.00	0.00	30
2	0.00	0.00	0.00	24
3	0.00	0.00	0.00	17
4	0.00	0.00	0.00	9
accuracy			0.38	130
macro avg	0.08	0.20	0.11	130
weighted avg	0.15	0.38	0.21	130

Predicting Student's Weekend Alcohol Consumption



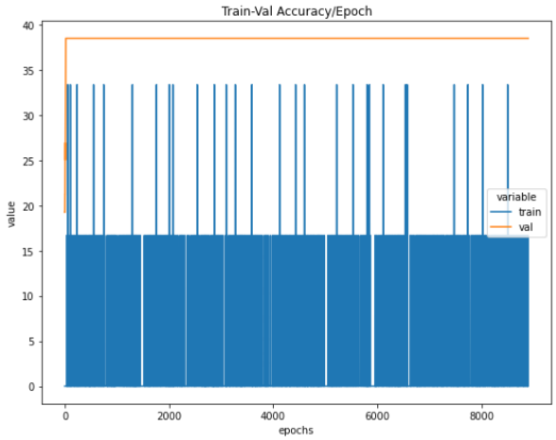


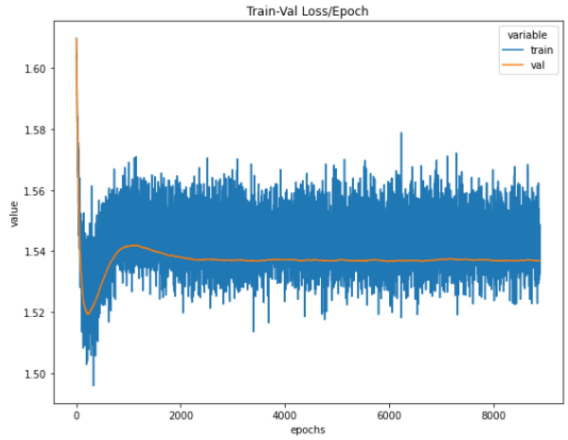
Appendix BA

Activation Function	Hardswish
Loss Function	Cross Entropy Loss
Optimizer	SGD
Training Epochs	8895
Batch Size	87
Learning Rate	0.0006327001948782599
Number of Hidden Layers	1
Hidden Layers: Inputs and Outputs	1: (2639) => (1319)
Batch Normalization	Input Layer: 2639, Hidden Layer 1: 1319
Early Stopping Patience	-
Weight Decay	0.4264997841858085
Drop Out Rate	0.42359007485630035
Last Epoch Information	Epoch 8895: | Train Loss: 1.54611 | Val Loss: 1.53685 | Train Acc: 0.000| Val Acc: 38.462
Test Accuracy	38%

Output table screen shot:

0	0.38	1.00	0.56	50
1	0.00	0.00	0.00	30
2	0.00	0.00	0.00	24
3	0.00	0.00	0.00	17
4	0.00	0.00	0.00	9
accuracy			0.38	130
macro avg	0.08	0.20	0.11	130
weighted avg	0.15	0.38	0.21	130





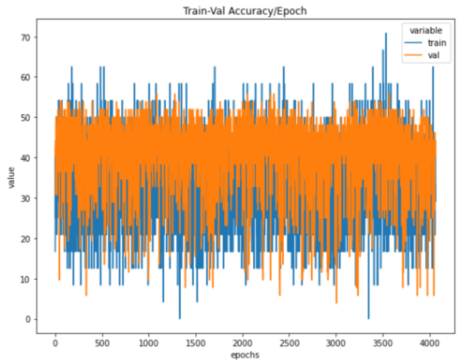
Appendix BB

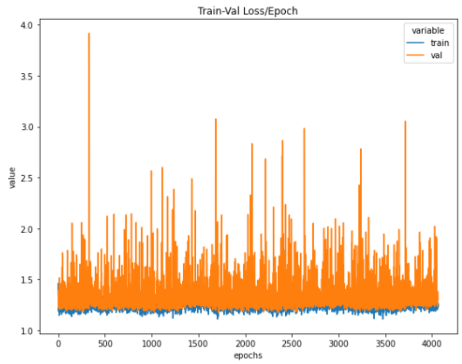
Activation Function	LeakyReLU
Loss Function	Cross Entropy Loss
Optimizer	SGD
Training Epochs	4066
Batch Size	20
Learning Rate	0.378637059648244
Number of Hidden Layers	1
Hidden Layers: Inputs and Outputs	1: (2784) => (1392)
Batch Normalization	-
Early Stopping Patience	-
Weight Decay	0.014236697245389222
Drop Out Rate	0.030501206666231157
Last Epoch Information	Epoch 4066: | Train Loss: 1.22724 | Val Loss: 1.30061 | Train Acc: 29.167| Val Acc: 44.231
Test Accuracy	43%

Output table screen shot:

0	0.42	1.00	0.60	50
1	0.20	0.03	0.06	30
2	0.00	0.00	0.00	24
3	0.00	0.00	0.00	17
4	0.71	0.56	0.63	9
accuracy			0.43	130
macro avg	0.27	0.32	0.26	130
weighted avg	0.26	0.43	0.29	130

Predicting Student's Weekend Alcohol Consumption



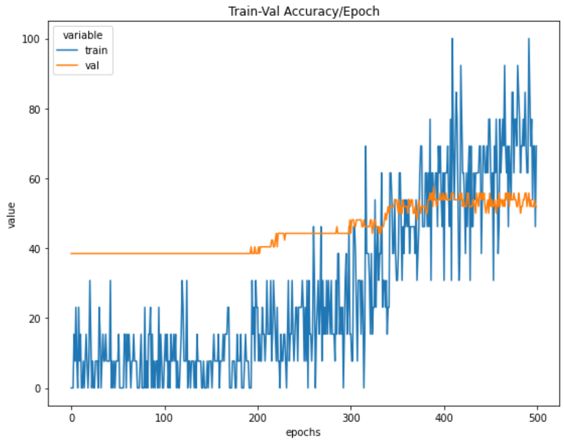


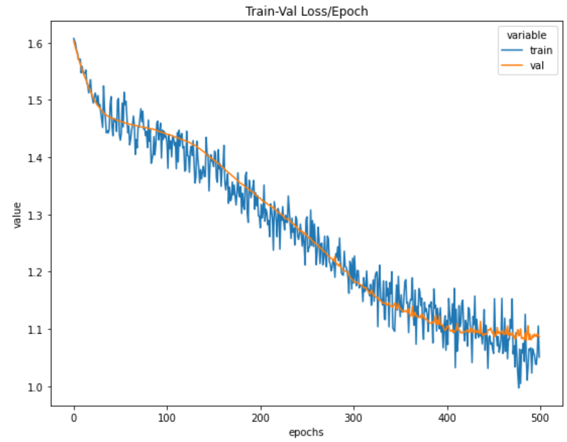
Appendix BC

Activation Function	ReLU
Loss Function	Cross Entropy Loss
Optimizer	SGD
Training Epochs	500
Batch Size	36
Learning Rate	0.0045
Number of Hidden Layers	3
Hidden Layers: Inputs and Outputs	1: (2048) => (1024) 2: (1024) => (512) 3: (512) => (256)
Batch Normalization	-
Early Stopping Patience	-
Weight Decay	0.01
Drop Out Rate	0.15
Last Epoch Information	-
Test Accuracy	51%

Output table screen shot:

	precision	recall	f1-score	support
0	0.59	0.82	0.68	50
1	0.29	0.13	0.18	30
2	0.33	0.25	0.29	24
3	0.45	0.59	0.51	17
4	0.83	0.56	0.67	9
accuracy			0.51	130
macro avg	0.50	0.47	0.47	130
weighted avg	0.47	0.51	0.47	130



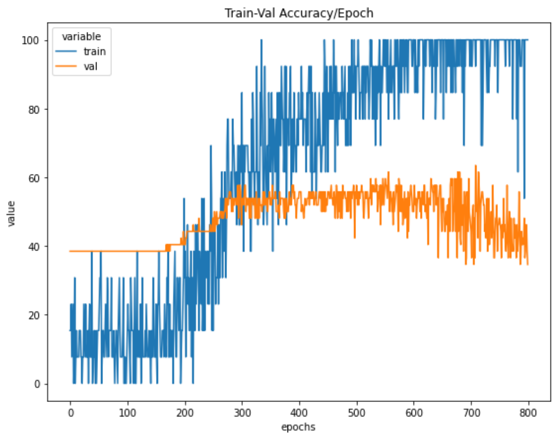


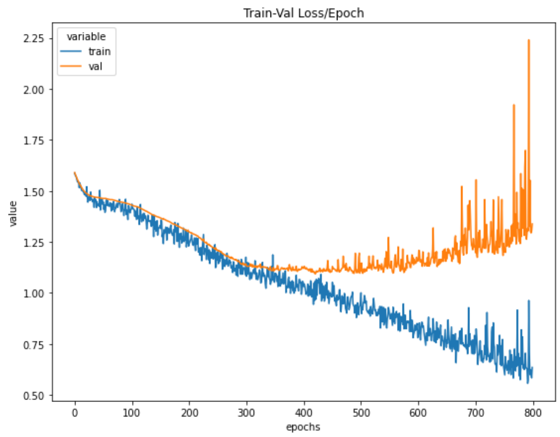
Appendix BD

Activation Function	ReLU
Loss Function	Cross Entropy Loss
Optimizer	SGD
Training Epochs	800
Batch Size	36
Learning Rate	0.0055
Number of Hidden Layers	3
Hidden Layers: Inputs and Outputs	1: (2048) => (1024) 2: (1024) => (512) 3: (512) => (256)
Batch Normalization	-
Early Stopping Patience	-
Weight Decay	0.01
Drop Out Rate	0.15
Last Epoch Information	Epoch 800: | Train Loss: 0.63405 | Val Loss: 1.33776 | Train Acc: 100.000| Val Acc: 34.615
Test Accuracy	44%

Output table screen shot:

	precision	recall	f1-score	support
0	0.66	0.54	0.59	50
1	0.29	0.40	0.33	30
2	0.20	0.17	0.18	24
3	0.50	0.53	0.51	17
4	0.56	0.56	0.56	9
accuracy			0.44	130
macro avg	0.44	0.44	0.44	130
weighted avg	0.46	0.44	0.44	130



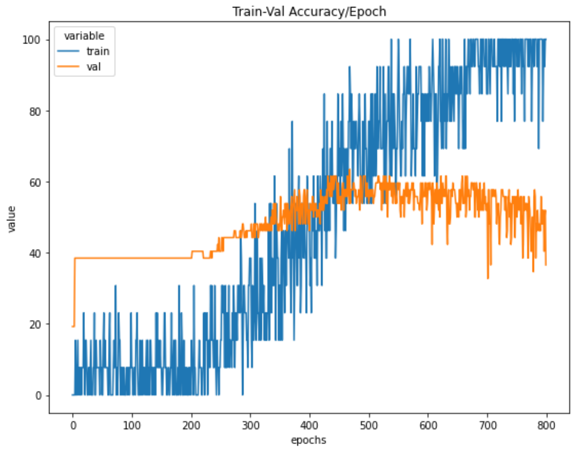


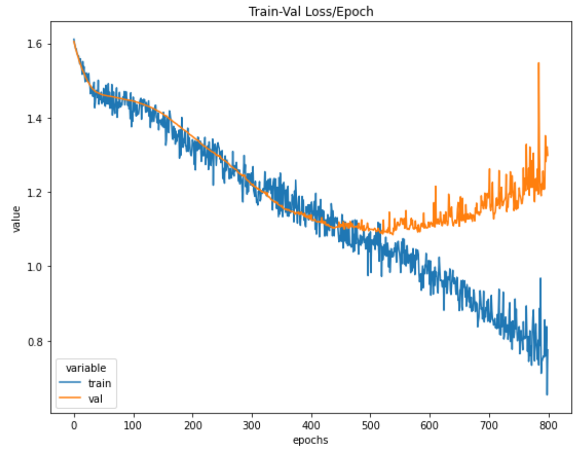
Appendix BE

Activation Function	ReLU
Loss Function	Cross Entropy Loss
Optimizer	SGD
Training Epochs	800
Batch Size	36
Learning Rate	0.005
Number of Hidden Layers	3
Hidden Layers: Inputs and Outputs	1: (2048) => (1024) 2: (1024) => (512) 3: (512) => (256)
Batch Normalization	-
Early Stopping Patience	-
Weight Decay	0.01
Drop Out Rate	0.15
Last Epoch Information	Epoch 800: | Train Loss: 0.77542 | Val Loss: 1.29845 | Train Acc: 100.000| Val Acc: 36.538
Test Accuracy	42%

Output table screen shot:

	precision	recall	f1-score	support
0	0.64	0.50	0.56	50
1	0.26	0.27	0.26	30
2	0.23	0.38	0.29	24
3	0.55	0.35	0.43	17
4	0.60	0.67	0.63	9
accuracy			0.42	130
macro avg	0.46	0.43	0.43	130
weighted avg	0.46	0.42	0.43	130





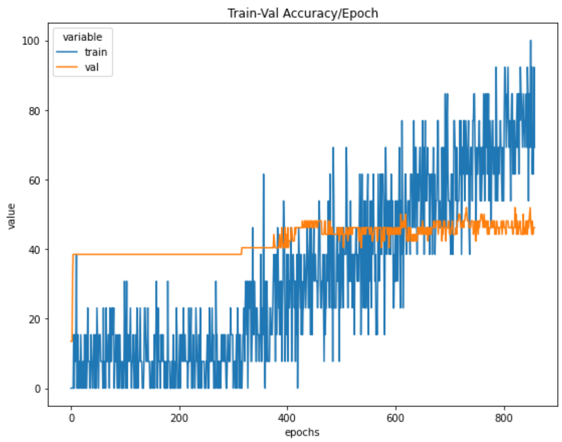
Appendix BF

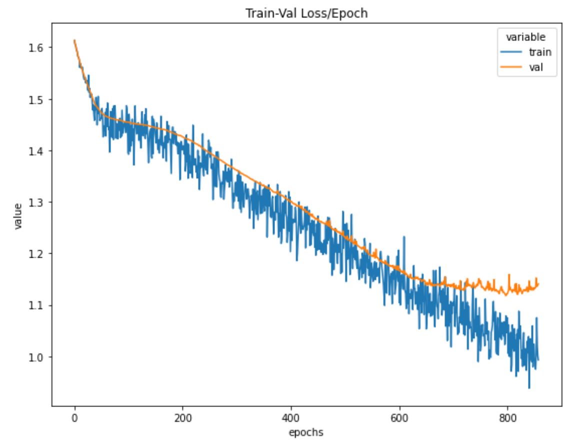
Activation Function	ReLU
Loss Function	Cross Entropy Loss
Optimizer	SGD
Training Epochs	900
Batch Size	36
Learning Rate	0.0032
Number of Hidden Layers	3
Hidden Layers: Inputs and Outputs	1: (2048) => (1024) 2: (1024) => (512) 3: (512) => (256)
Batch Normalization	-
Early Stopping Patience	-
Weight Decay	0.01
Drop Out Rate	0.15
Last Epoch Information	Epoch 858: | Train Loss: 0.99290 | Val Loss: 1.14043 | Train Acc: 69.231| Val Acc: 46.154
Test Accuracy	51%

Output table screen shot:

0	0.58	0.90	0.70	50
1	0.50	0.27	0.35	30
2	0.20	0.08	0.12	24
3	0.42	0.65	0.51	17
4	0.00	0.00	0.00	9
accuracy			0.51	130
macro avg	0.34	0.38	0.34	130
weighted avg	0.43	0.51	0.44	130

Predicting Student's Weekend Alcohol Consumption





Appendix BG

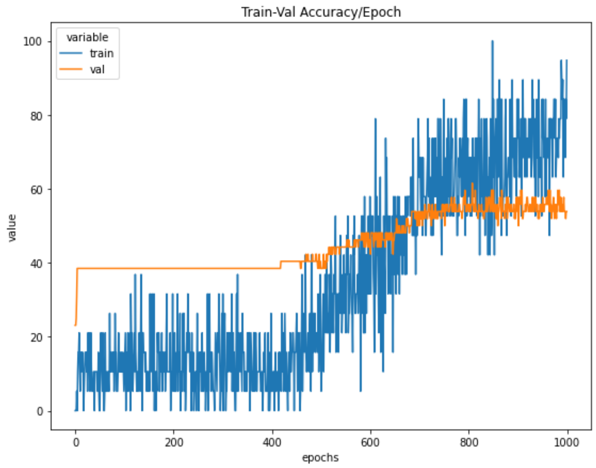
Activation Function	ReLU
Loss Function	Cross Entropy Loss
Optimizer	SGD
Training Epochs	1000
Batch Size	25
Learning Rate	0.0023
Number of Hidden Layers	2
Hidden Layers: Inputs and Outputs	1: (1024) => (512) 2: (512) => (256)
Batch Normalization	-
Early Stopping Patience	-
Weight Decay	0.01
Drop Out Rate	0.01
Last Epoch Information	Epoch 1000: | Train Loss: 0.93929 | Val Loss: 1.07883 | Train Acc: 94.737| Val Acc: 53.846
Test Accuracy	51%

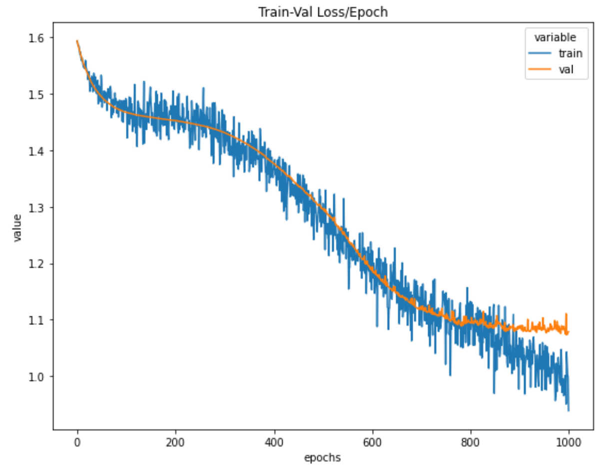
Output table screen shot:

```
              precision    recall  f1-score   support

           0       0.60      0.74      0.66        50
           1       0.33      0.27      0.30        30
           2       0.33      0.33      0.33        24
           3       0.62      0.47      0.53        17
           4       0.71      0.56      0.63         9

    accuracy                           0.51       130
   macro avg       0.52      0.47      0.49       130
weighted avg       0.50      0.51      0.50       130
```


Train-Val Accuracy/Epoch
variable
train
val
value
epochs
0
20
40
60
80
100
0
200
400
600
800
1000


Train-Val Loss/Epoch
variable
train
val
value
epochs
1.0
1.1
1.2
1.3
1.4
1.5
1.6
0
200
400
600
800
1000