The School of Mathematics

THE UNIVERSITY *of* EDINBURGH

Anomaly Detection with Bayesian Neural Networks

by

Theodoros Ladas

Dissertation Presented for the Degree of
MSc in Statistics with Data Science

July 2021

Supervised by
Dr Bruce Worton and Dr Daniel Paulin

Executive Summary

In this project a way of capturing anomalies on a dataset using Bayesian Neural networsk is presented. After a robust validation and explanation of the model, two metrics are presented that could make it easy in ranking and identifying anomalies by measuring how uncertain the predictions of the algorithm are.

Acknowledgements

To Sotiria, who was by my side.

University of Edinburgh – Own Work Declaration

This sheet must be filled in, signed and dated - your work will not be marked unless this is done.

Name: ....
Matriculation Number: ....
Title of work: ....

I confirm that all this work is my own except where indicated, and that I have:

- Clearly referenced/listed all sources as appropriate
- Referenced and put in inverted commas all quoted text (from books, web, etc)
- Given the sources of all pictures, data etc. that are not my own
- Not made any use of the report(s) or essay(s) of any other student(s) either past or present
- Not sought or used the help of any external professional academic agencies for the work
- Acknowledged in appropriate places any help that I have received from others (e.g. fellow students, technicians, statisticians, external sources)
- Complied with any other plagiarism criteria specified in the Course handbook

I understand that any false claim for this work will be penalised in accordance with the University regulations (`https://teaching.maths.ed.ac.uk/main/msc-students/msc-programmes/statistics/data-science/assessment/academic-misconduct`).

Signature ....

Date ....

Contents

List of Tables

List of Figures

1 Introduction

1.1 Motivation

Anomaly detection is a beneficial technique leveraged among others by the military, for cybersecurity reasons, and others [3], [4], [2]. Specifically in the banking sector, it is used to block fraudulent transactions automatically. This dissertation project explains how an anomaly detection system could be implemented using Bayesian neural networks. This application uses machine learning to train a model on a set of data to predict an outcome of interest. Afterwards, the predictions are drawn multiple times (bootstrapped) to estimate confidence about the prediction. The project aims to automatically flag transactions that could be fraudulent, thus allowing a human to spend time in a deep investigation of those transactions, rather than manually going through each transaction one by one.

This report is divided into the *Introduction*, *Exploratory Data Analysis*, *Methods*, and *Results* sections and a brief summary of each one is presented here. The rest of the *Introduction* section will explain the specific dataset used for the presentation of the problem, as well as all the software requirements to be able to reproduce the results. Next, the *Exploratory Data Analysis* section presents various aspects of the data, such as the distributions of the features, their correlations as well as a quick way to discover potential anomalies visually. On the *Methods* section, the specific architecture of all the neural networks models are presented, as well as the way the best model was selected (model validation). Finally, three related but different metrics are presented in the *Results* section of what constitutes an anomaly, yet the specific threshold for those metrics is arbitrarily set. In a real-life application, this is a business decision the stakeholders should set according to their criteria.

1.2 Data Sources

The dataset used in this project was given by the University of Edinburgh in the context of the dissertation project, and it is the wine dataset. It is an especially clean and very well-known real-life dataset for prototype creation. The basic assumption is that if the algorithm manages to capture anomalies on this dataset, it is in principal possible to productionise a variant of the architecture for an application of interest. More specifically, it is a 4898 (rows) $\times$ 12 (columns) matrix, with no missing values and no duplicated rows. Each row represents one wine, and the columns are the various features of each wine, such as its degrees of *alcohol*, the *acidity* of the wine, the *pH* level, which measures how acidic or basic the solution etc. The problem this dissertation is going to try to solve is a classification problem, of the *type* of the wine, according to all other features. The variable *type* is either *1, 2, or 3* and the classes are almost perfectly balanced, thus making easier the preprocessing stage of the problem and focusing more on the architecture and the interpretation of the algorithm as well as its results. Thus, the only preprocessing steps needed are the centring and scaling of the data matrix to ensure that variables can potentially have the same impact regardless of the scale they are measured in, and one-hot encoding the *quality* feature, since it is also not a numeric variable, but a factor one. Lastly, the dataset is split into train (80%), validation (25%) and test (25%) sets to ensure that no algorithm overfits the given dataset.

1.3 Software

A combination of the programming languages R and Python are used to produce this report. The use of both languages is in no way binding. R is used for the exploration of the dataset, as well as for the dimensionality reduction plots, while Python is used in combination with tensorflow,tensorflow-probability, and keras to build, validate, and test the neural networks. These packages can also be used from the R ecosystem, yet the setup process is more complicated and thus avoided. The reason R is selected for the EDA is due to the ggplot2 package, which is a very powerful and easy package for creating complicated plots.

2 Exploratory Data Analysis

In this section, a basic overview of the dataset and its properties is going to be presented. Firstly, various statistics regarding the dataset are shown in the form of graphs in univariate, bivariate and multivariate analysis. Afterwards, the dataset is reduced in dimensions with two techniques (a linear and a non-linear) in order to be able to plot it with a two-dimensional graph. This is also a quick way to identify potential anomalies as well visually. The reason for the extended EDA is to understand what preprocessing steps could be needed in order for future models to work correctly. Also, the knowledge gained from this exploration of the dataset will help in the explanation of the model in the *Model Validation* section.

2.1 Univariate Analysis

The dataset, consist of twelve feature variables, out of which only one (*quality*) is categorical, while all the others are numeric. The target variable (*type*) is also a target variable with three levels type=1, type=2, type=3. There are no missing values. In Figure 1, the target variable type is represented

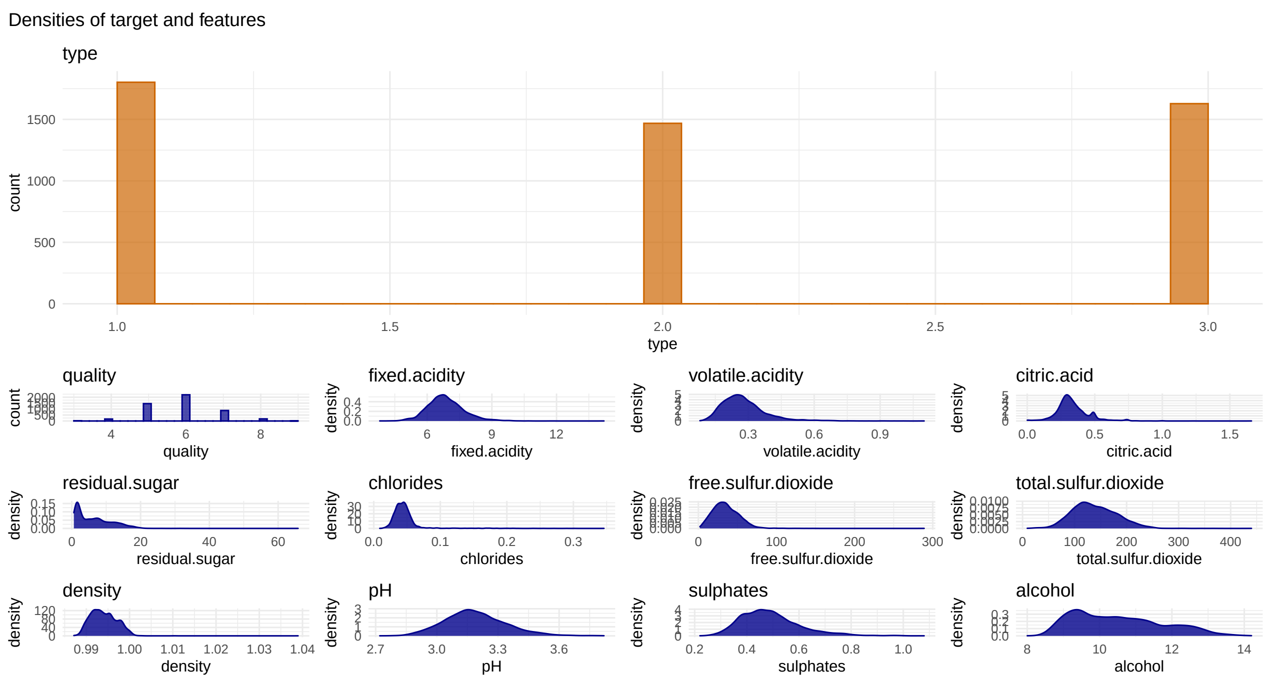


Figure 1: Univariate EDA of Dataset

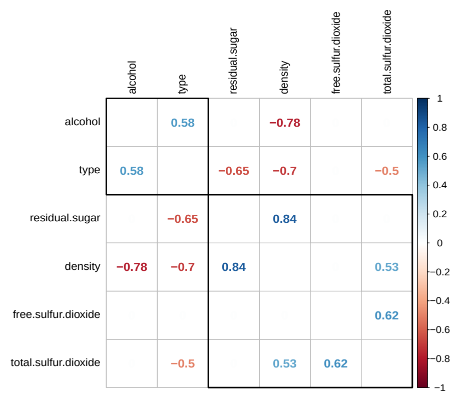
by the orange barplot. The number of rows in each class is balanced; therefore, no preprocessing to upsample (downsample) the underrepresented (overrepresented) classes is needed. On the variable *quality*, the majority of cases are
$(quality \geq 5) \vee (quality \leq 7)$, which would make the very poor and very good wines difficult to predict. Finally, evidently, the features, are measured on vastly different scales. For example *pH* is ranging from 2.7 to 3.6, while *free.residual.dioxide* is ranging from 0 to 300. This means that centring and scaling the data matrix is crucial for any algorithm to work properly.

2.2 Bivariate Analysis

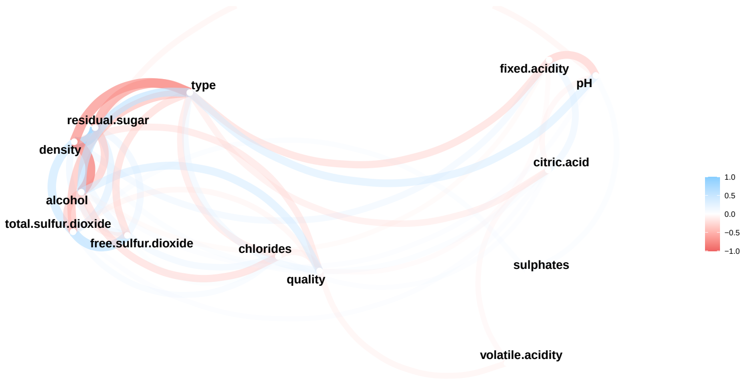
Extending the univariate analysis of the previous subsection, the bivariate analysis reveals how the features correlate to each other. Figure 2 (a) and Figure 2 (b) are both producing the same information but presented differently. On Figure 2 (a) the exact values of the linear relationships are shown, while on Figure 2 (b) clear clusters of high and low linear correlation are formed. Out of these clusters that are present in Figure 2 (b), some are expected, but others are not. For measuring the correlation, the *Pearson's correlation coefficient* is used, which measures the covariance of two random variables, X and Y, and normalizes it with the product of their standard deviations, or:

$$\rho(x, y) = \frac{\text{cov}(x, y)}{\text{std}(x)\text{std}(y)} \tag{2.1}$$

For example, Figure 2 (a) shows a strong positive relationship between *free.sulfur.dioxide* and *to-*



(a) Correlation matrix



(b) Correlation network

Figure 2: Bivariate EDA of Dataset

tal.sulfur.dioxide, which is expected, as *total.sulfur.dioxide* is the *free.sulfur.dioxide* plus other sulfur dioxides from other ingredients. Intrestingly, the same is not true regarding *volatile.acidity*, *fixed.acidity* and *citric.acid*. The feature with the most strong correlations however, is *density*, as it strongly correlates, both positively or negatively with *alcohol*, *type*, *residual.sugar* and *total.sulfur.dioxide*. In general, we identify two clusters. One with strong correlations containing *type*, *residual.sugar*, *density*, *alcohol*, *total.sulfure.dioxide* and *free.sulfure.dioxide*, and a second one with low correlations containing *fixed.acidity*, *pH*, *citric.acid*, *sulphates* and *volatile.acidity*. The model is going to leverage these relationships in order to predict the outcome.

2.3 Dimensionality Reduction

In exploring the dataset, two different techniques were used to reduce the dimensions and visualise them with plots. The first method is the Principal Component Analysis (PCA), which is a singular value decomposition (SVD) of the centred data matrix. The SVD of an $m \times n$ matrix A is the factorization of the matrix into $USV^\top$ [5]. Secondly, a more advanced non-linear dimensionality reduction method, called t-distributed Stochastic Neighbor Embedding (t-SNE), was tried to cross evaluate the results of PCA. This method works by finding a way to project the high dimensional data into a lower dimension while preserving the clustering of the higher dimension.

2.3.1 PCA

Using the singular value decomposition to factorise the data matrix X, we produce three matrixes, U, S and V, each carrying information about the dataset in some aspect. First of all, we can create the following graph of the percentage of variance explained by each principal component [5]. In Figure 3,

we can see the cumulative variance explained by each additional principal component. This is a critical graph as we can reduce the dimension of the data matrix by selecting a threshold (for example, 99%

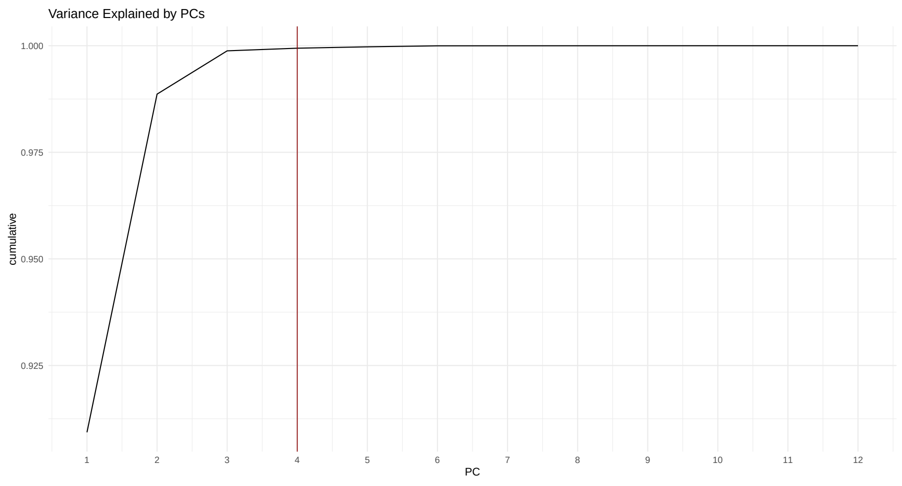


Figure 3: Variance Explained by each Principal Component

of the variance). In this example, it is clear that with only the first four principal components, that threshold is surpassed. The principal components are a linear combination of all the dataset columns, and each one is perpendicular to the previous one. Therefore, we can continue the exploration by visualising these four principal components' weights (loadings) and trying to understand whether these components make sense according to our knowledge of the topic. In Figure 2.3.1 the first

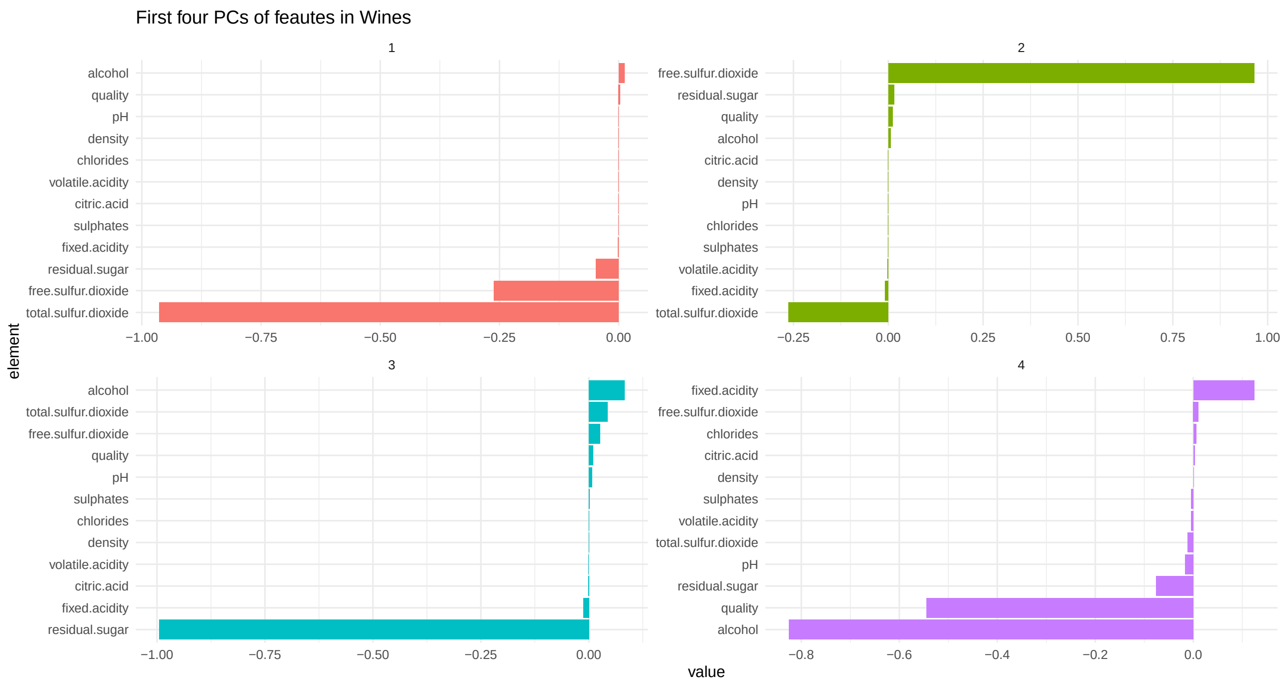


Figure 4: Loading of the top four Principal Component

four principal components are visualised along with the loadings of each column. By on the two most extreme positive and negative values of the loadings, we can interpret the components into new kinds of variables.

- PC1: Alcohol vs Sulfur Dioxide (free and total)
- PC2: Free Sulfur Dioxide vs Total Sulfur Dioxide
- PC3: Alcohol vs Sugar

- PC4: Alcohol vs Acidity

It is clear that this dimensionality reduction technique produces results that reflect the real world. Sulfur Dioxide is a vital component in winemaking as it regulates bacteria growth among other essential tasks, yet it also gives unpleasant odours and tastes to the wine. PCA immediately captures that reality by assigning the two most significant negative loadings on sulfur dioxide concentration (both free and total) to the first principal component. In addition to that, the second most important component in order to classify the wines is the origin of that Sulfur Dioxide. The total Sulfur Dioxide is the Free Sulfur Dioxide plus Sulfur Dioxide bound to other ingredients such as sugars etc. After investigating how much SO2 a wine has and where it comes from (free vs total), the next most important factors have to do with the specific taste of the wine, mainly how sweet and how acidic the taste the wine has. We can now visualise the first two principal components on a plot while also colouring the datapoint according to their `type`, the target variable. It is immediately clear by Figure 5 that `type` classes two

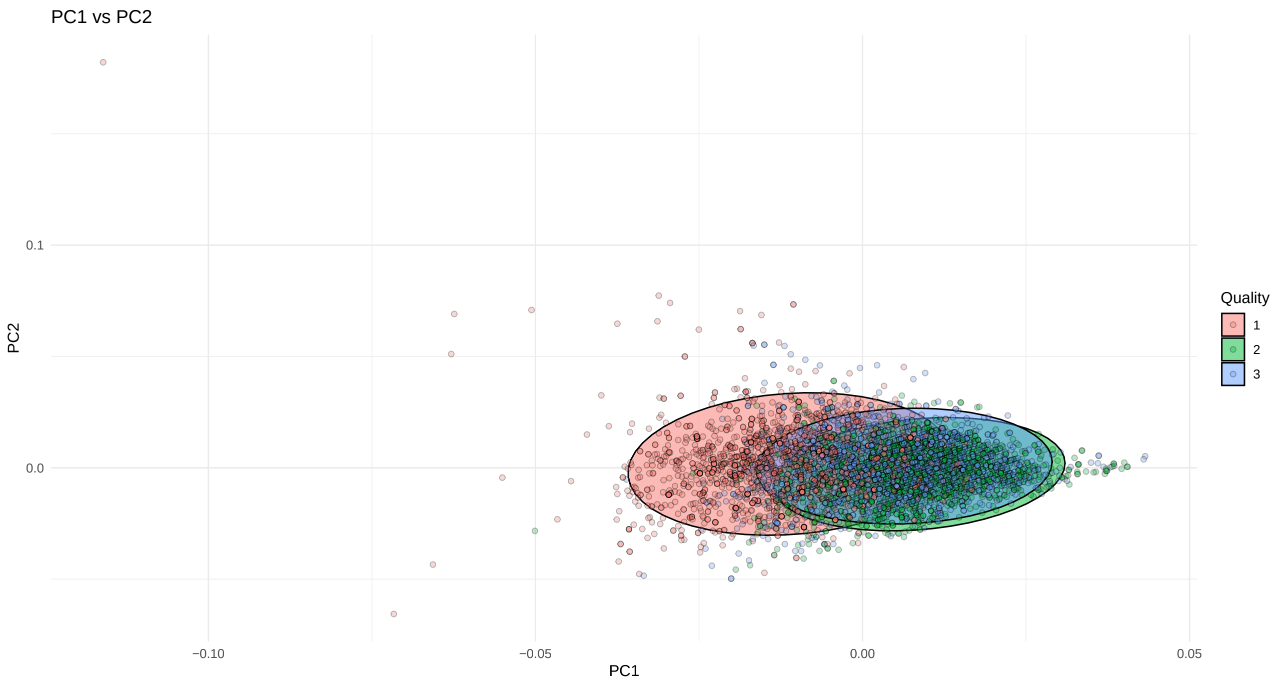


Figure 5: PCA biplot

and three, have an extensive overlap in this graph. That is useful information in the model evaluation, as we expect the model to "find it difficult" to distinguish a wine of type two from a wine of type three while simultaneously having better accuracy in predicting class one. However, there a second major takeaway from *Figure 3*, and that is the existence of potential anomalies. While the majority of the points are centred around $(0, 0)$, some points are geometrically far away from the others of the same class.

2.3.2 t-SNE

Principal Component Analysis through singular value decomposition produced significant findings and deepened our understanding of the dataset. However, it was clear by Figure 5 that it could not create a clear enough separation between the classes. This could be because of the nature of the dataset, or it means that PCA is too simple to capture the complexity of the dataset due to its linear nature. Therefore, a second, non-linear [11] technique was tried in order to produce the corresponding figure as Figure 6. This technique calculates the distances from each point to each neighbour and imposes a t-distribution on those values. It is an iterative algorithm that many times produces very good clustering. However, in Figure 6, it is clear that *type* classes two and three are still clustered together, while class one is separated. The main takeaway from this analysis is that the dataset, although clean and straightforward to understand, is more complicated than it looks. This suggests that a simple model such as logistic regression might not be the model of choice. Instead, neural networks are a good candidate for such tasks due to their flexible nature.

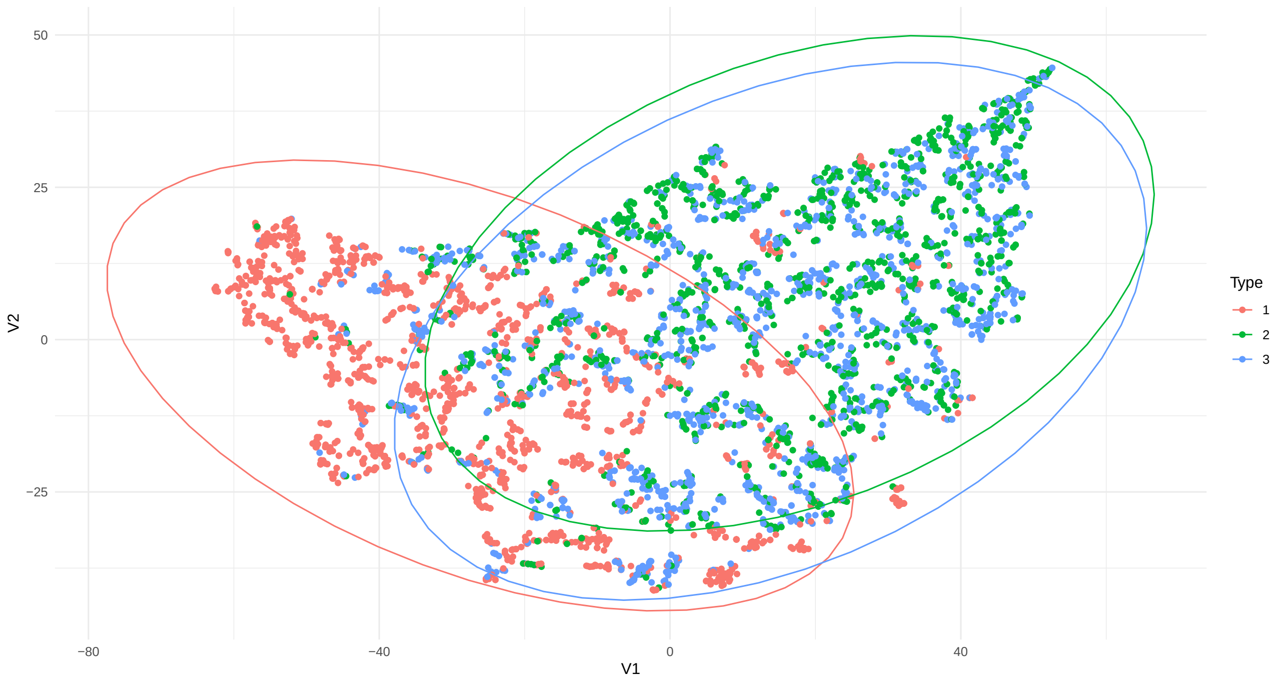


Figure 6: t-SNE biplot

2.4 Multivariate Analysis

With the information revealed from PCA, we can now plot the highest and the lowest loadings of the four principal components for each level of the target variable. The reason for plotting this graph is to investigate whether some level of the *type* variable behaves differently than the others when plotted on the same axis. In all four subplots of Figure 7 we identify the same exact behaviour. The coefficient

of the regression is negative on the *total.sulfure.dioxide* against *alcohol* axis for all levels, positive on the *total.sulfure.dioxide* against *free.sulfure.dioxide* axis, and close to zero for the other two PC scores. This finding, while not very exciting, is important, because it confirms once again that the relationship of our target variable *type* and the features identified at the dimensionality reduction stage (*total.sulfure.dioxide*, *free.sulfure.dioxide*, *alcohol*, *residual.sugar*, and *fixed.acidity*) are important for establishing a good model.

3 Models

In this section, the methods used to create the model will be discussed. This includes the baseline model that is deterministic in nature [7], as well as the various Bayesian approaches. Firstly, the reason to choose neural networks as the model of choice is due to the complexity of the research question, in combination with the complexity of the dataset as demonstrated from the exploratory data analysis. Secondly, the Bayesian approach is essential to make the neural network prediction *stochastic* [10]. In order to train the model and produce both high accuracy on the training and validation sets, as well

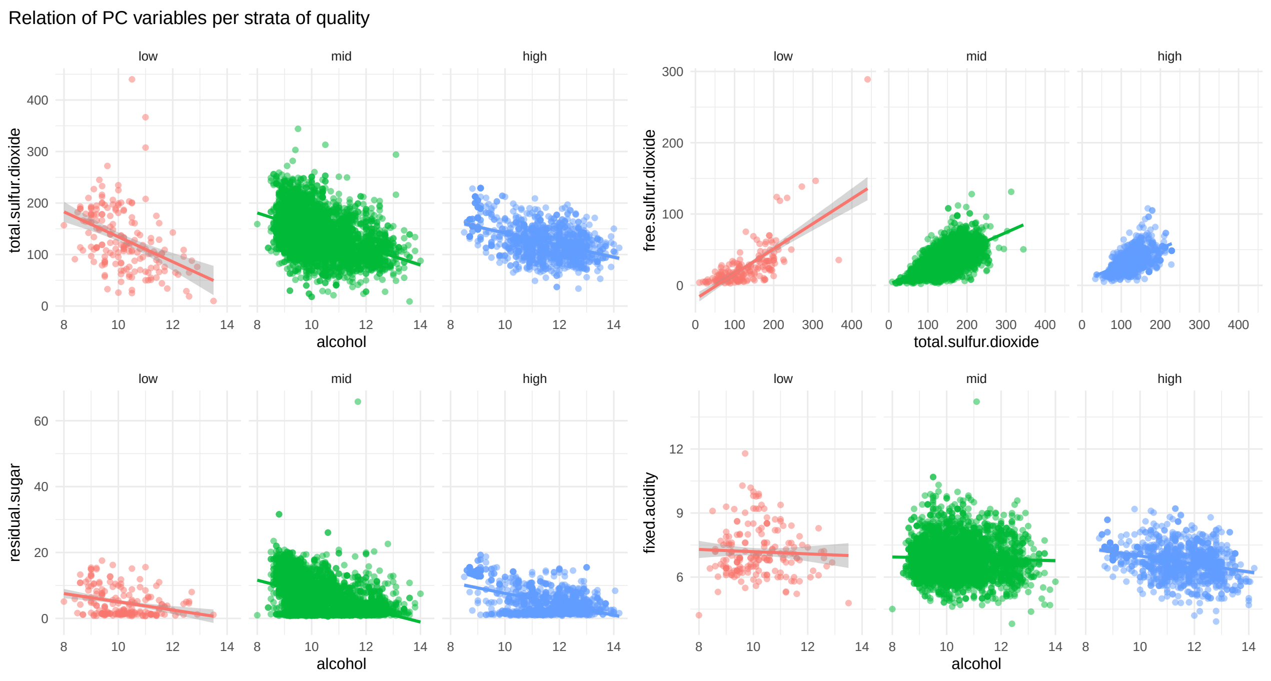


Figure 7: Multivariate EDA of Dataset

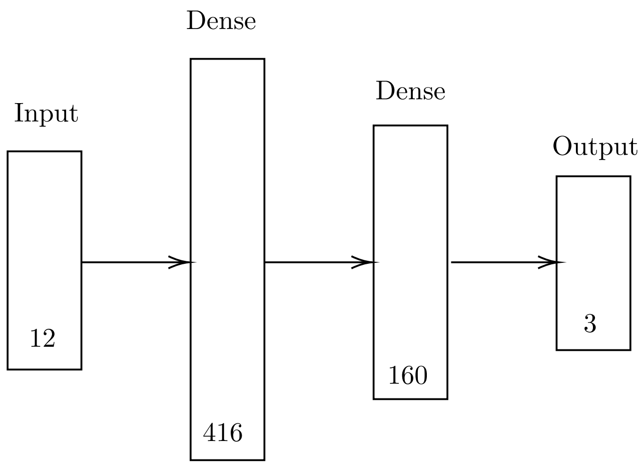
as generalize well on the test set, a series of actions had to be taken. Firstly, a set of hyperparameters had been set for all models. Those hyperparameters are:

- The Learning Rate - ranging from $1e-1$ to $1e-8$
- The number of layers between the Input and the Output - ranging from 2 to 8
- The activation function of these layers - where the choices where:
  - Rectified Linear Unit (reLU): $f(x) = \max(0, x)$
  - Exponential Linear Unit (ELU): $f(x) = \max(e^x - 1, x)$
  - Sigmoid: $f(x) = \frac{1}{1+e^{-x}}$
- The number of units in each function - rangin from 32 to 512, with a step of 128

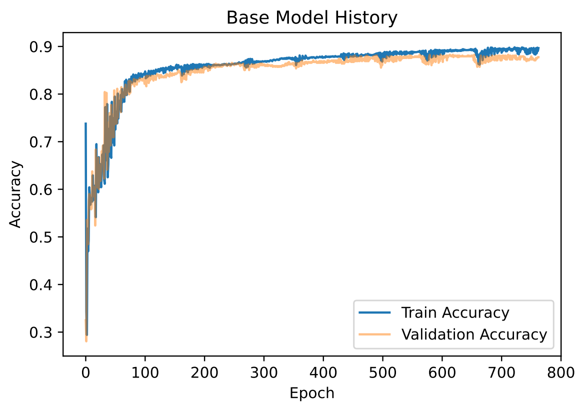
The optimization process, was a RandomSearch algorithm, implemented with the KerasTuner package. This results in slightly different results in every run, however by switching to a GridSearch algorithm instead, the same outcome can be ensured each time. However, this process takes more time since it is searching the whole hyperparameter space, while the RandomSearch takes less time to find the best model in each run. The metric used to judge which model is best was chosen to be *accuracy* due to its simplicity and suitability at the research problem. Lastly, the loss function that the models are minimising in order to produce the best weights was a custom made function that calculated the sum of the categorical cross-entropy between the true and the predicted labels. The true labels have a known (for the training set) probability distribution (PMF), and the predicted labels also have their probability distribution (PMF). When the sum of the categorical cross entropies is minimum, it shows that the predicted PMF and the true PMF are as close as possible. Mathematically this is expressed as:

$$\text{Loss} = \sum_{i=1}^{3} y_{true}^{(i)} \log \frac{1}{y_{pred}^{(i)}} = -\sum_{i=1}^{3} y_{true}^{(i)} \log y_{pred}^{(i)} \tag{3.1}$$

Lastly, the models are trained for 2000 training epochs. After a certain number of epochs, it started to overfit. Early stopping was implemented in the model with the patience of 200 epochs to avoid this pitfall. That means that if the validation accuracy drops for 200 epochs, the model assumes that



(a) Base Model Neural Network Architecture



(b) Base model History of epochs

Figure 8: Baseline Architecture and Epoch History

it overfitted and stops trying to minimise the loss function further. This is why the x-axis varies in every Figure (8, 9, 11) where the history of accuracy per epoch is plotted. After the successful training of the base neural network, the predictions are deterministic. That means that each time a specific input is passed through the NN, the same output is produced. While this can be useful in particular applications, the goal of this project is to find a metric for the *uncertainty* of the predictions. This is impossible to do with a deterministic process as samples cannot be drawn from the posterior distribution to estimate it. Therefore, after establishing a baseline to prove that the problem can be solved with good enough accuracy, the network can become Bayesian by changing some of its layers to Bayesian layers. Afterwards, we can sample from the posterior distribution of the predictions by predicting n times the same input, in our case, the test set. Then by storing each predicted probability (one for each class) in each draw $(1 \dots n)$, we can compute average predictions, variances, confidence intervals, and more metrics discussed in section 4.

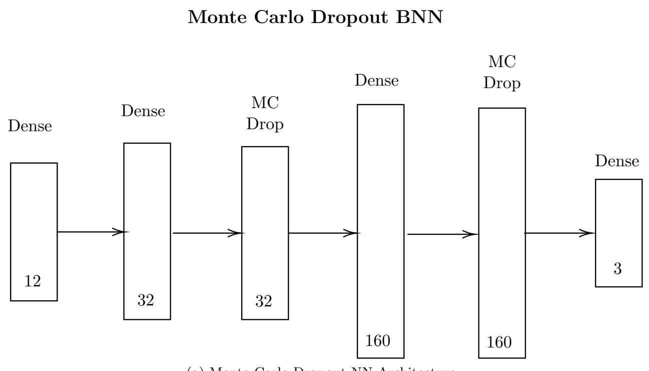
3.1 Base Neural Network

Before tackling the Bayesian Neural Network, a deterministic baseline should be first implemented. The model is built using the keras package for Python. The architecture that produced the best results is one with two hidden layers of 416 and 160 neurons, respectively, densely connected. This produces a total number of parameters to be estimated at $75,107$. The neural network is architecture depicted in Figure 8 (a). Figure 8 (b) is a typical history of the training accuracy increasing over each epoch.

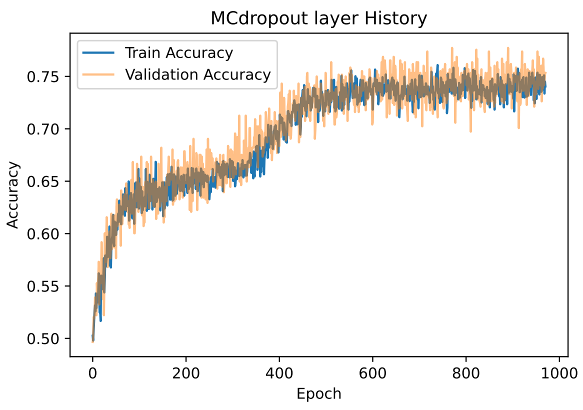
Typicaly it stops before iteration 1000 and the best accuracy of the model is close to 90%. This is a very good accuracy, since as we have seen at the exploratory data analysis, *type 2* and *type 3* are very difficult to distinguish. With the baseline model explained and trained, we gained an understanding of the extent to which the problem can be solved with neural networks. Arguably, an accuracy score of 90% does not mean anything in a vacuum. For example, 90% accuracy in predicting a particular disease in a patient might be too low. In contrast, the same accuracy on an application less sensitive to critical decision making might be excellent. However, this could be a good final score for the model in the specific task at question (classifying types of wines).

3.2 Monte Carlo Dropout

Adding some Monte Carlo Dropout layers in the middle of each Dense layer is the first approach in turning the base model into a Bayesian one. This method is traditionaly used by the literature for data



(a) Monte Carlo Dropout NN Architecture



(b) Monte Carlo Dropout History of epochs

Figure 9: Monte Carlo Dropout Architecture and Epoch History

generation purposes in fields where data collection is costly or / and difficult [6]. Simple dropout layers work during the training of the neural network by randomly ignoring or ”dropping out” a percentage of neurons along with the backward and forward connections. This is being handled by random draws of a Bernoulli distribution with the ”success” probability defined by the user. That process still generates a *deterministic* prediction since at test time, which neurons are online is already decided. Monte Carlo dropout is a wrapper of the simple dropout function that preserves the dropout at testing time. That way, the same prediction passed through the network twice will produce different predictions according to which neurons and connections are online. Finally, these predictions are interpreted as random draws from the posterior distribution.

It is evident in Figure 9 (b) that there is a lot more variance in the prediction from epoch to epoch due to the random nature of the dropout layers. An important finding in this approach is that the validation and the training accuracy are almost in all epochs at the same level. Contrary to the base model, where the validation accuracy increases at a slower rate than the training accuracy, this does not happen when the Monte Carlo Dropout layers are added to the model. It can even maintain for a short duration a higher validation accuracy than training accuracy, as seen from epochs 200 to 400 in Figure 9 (b) due to random chance.

3.3 Other Bayesian Approaches

In addition to the Monte Carlo Dropout model, various other approaches have been tried. Their results on training were not as good as the MC dropout model; hence they are presented in this section, and their specific architecture is in Figure 10. A more robust way of validating the models is presented in section 4. For both the Variational and the Flipout models, a $N(x; 0, 1)$ distribution was choosen, since we have no prior information to choose otherwise. While the Fliptout model behaves similarly to the MC dropout the Variational model, is behaving completely at random. This could suggest that something is wrong with the prior set. However, the Flipout model reaches a lower maximum accuracy than the MC dropout model; therefore, while it does work as a solution, it is likely not the model of choice.

3.4 Model Validation

In order to choose the best model among the Bayesian approaches, the test set is used to calculate the accuracy per class. In Figure 12 the base model (non-Bayesian) confusion matrix is presented. We can clearly see that *type 2* and *type 3* are mixed, as we expected from the EDA. In Figure 13, the

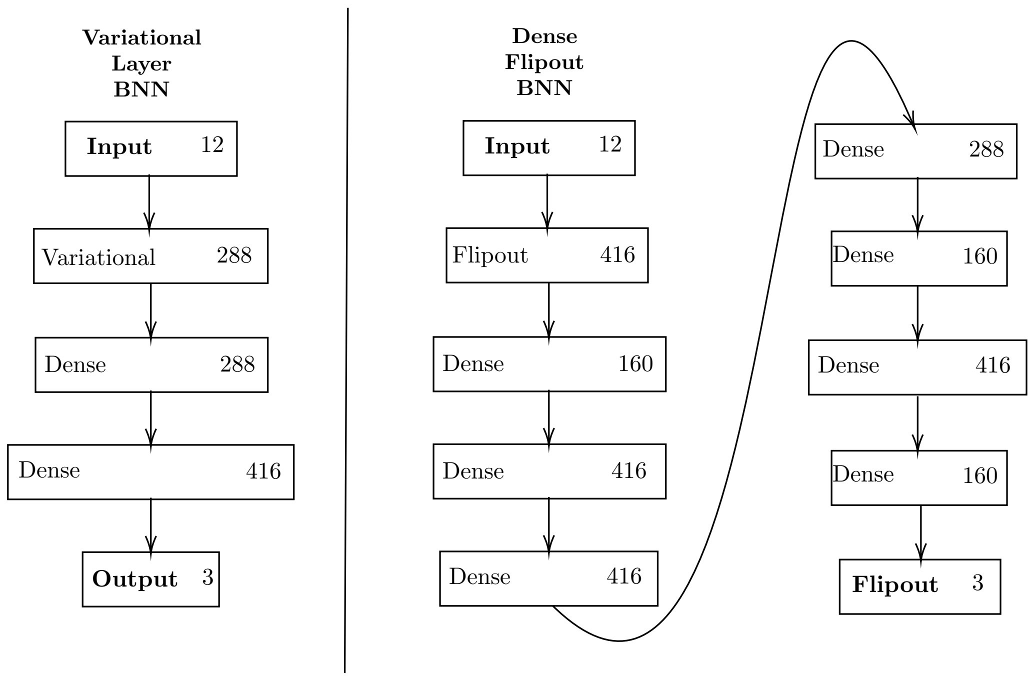
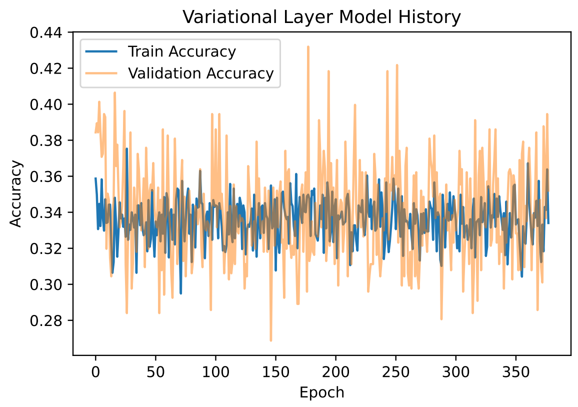
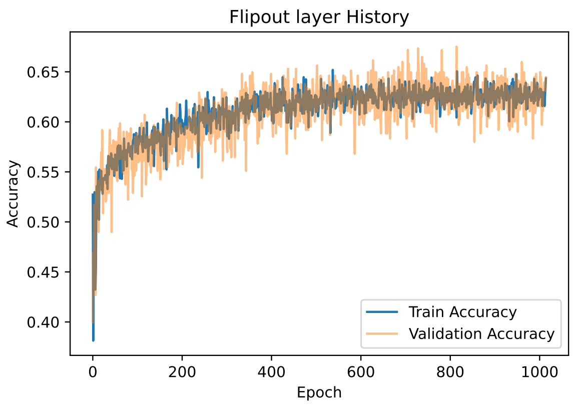


Figure 10: Variational and Flipout model Architectures



(a) Variational layers History of epochs



(b) Flipout layers History of epochs

Figure 11: History of other Bayesian Models (Flipout and Dense Variational)

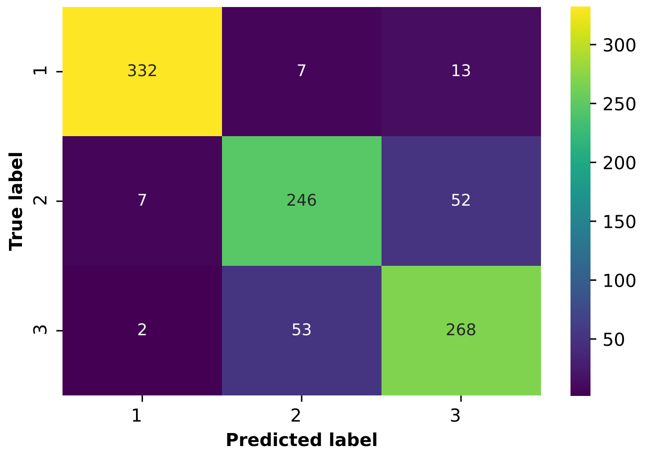
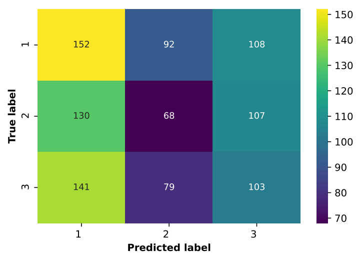
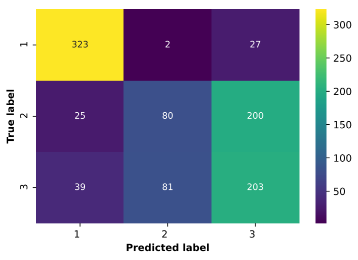


Figure 12: Base Model Confusion matrix

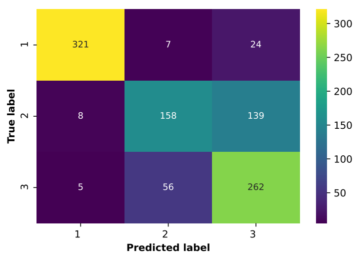
same confusion matrix for all the Bayesian models is presented. The Variational model is, as seen from the History graph as well, is almost completely random. The Flipout model, is having an exceptional performance in predicting *type 1*, yet it confuses *type 2* with *type 3*. Lastly, as predicted from the History plot, the MC dropout model has the best overall performance. Therefore, it is abundantly



(a) Variational Confusion matrix



(b) Flipout Confusion matrix



(c) MC dropout Confusion matrix

Figure 13: Confusion matrices for all the Bayesian Models

clear that the model of choice for the specific problem at hand is the MC dropout model, as it has the highest accuracy across classes.

3.5 Model Explanation

In this section, a more profound validation of the model is presented. Accuracy is a suitable enough metric for many tasks; however, when the project at hand needs to be both accurate and interpretable to some extent, further examination of what the model "thinks" when predicting is required. To

conduct this analysis, a package called SHAP is used. SHAP stands for *SHapley Additive exPlanation* and is a concept borrowed from Economics. A Shapley value represents the average marginal contribution of a feature value to the predicted outcome in the Machine Learning field. The next plots are produced on the test set to understand how the algorithm "decides" in a held-out, never seen before dataset. In Figure 14 we see the following. The top four, most influential variables to the output are *residual.sugar*, *fixed.acidity*, *alcohol*, and *total.sulfur.dioxide*. This seems to be in line with the finding of the much simpler (yet much faster) PCA in the EDA, although the order of the variables is not exaclty the same. More specifically, the mean SHAP value for *residula.sugar* for class zero, is about 0.17, while the same for class two is around 0.13 $(0.3 - 0.17)$ and even lower for class 1. That

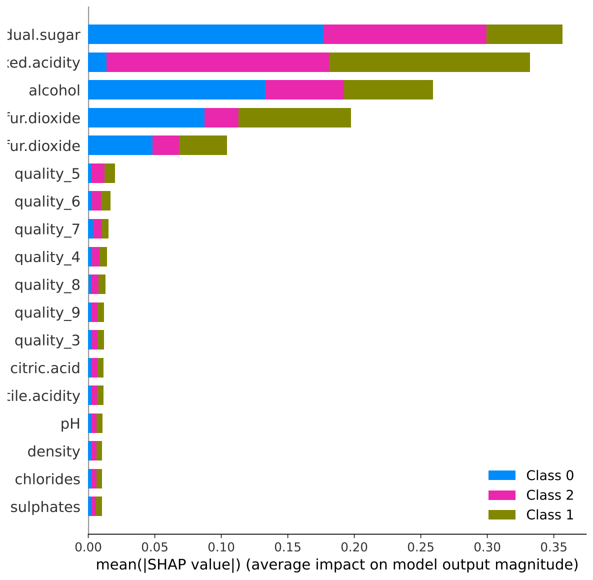
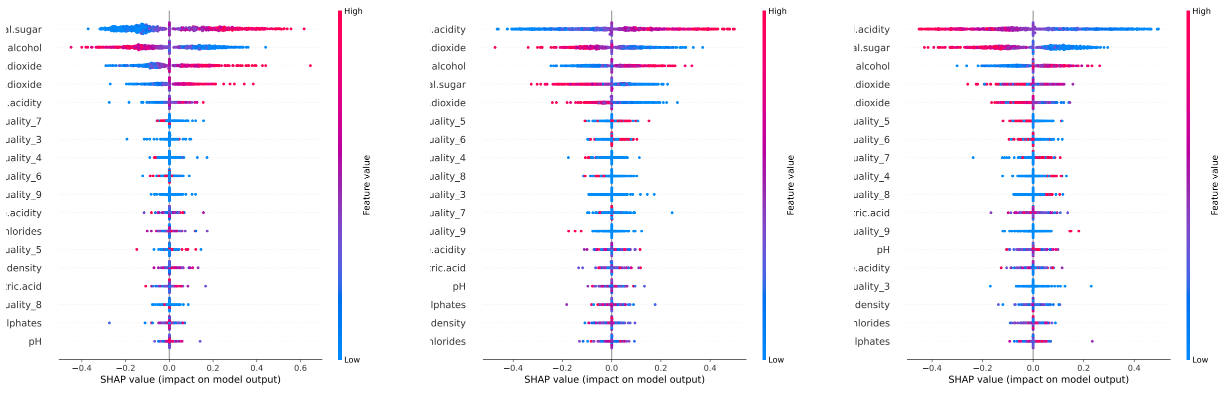


Figure 14: Shapley Values on a feature by feature level

means that *residual.sugar* is more important when predicting *type 0*, then *type 2* and lastly *type 1*. The same logic can be extrapolated for *fixed.acidity*, *alcohol*, etc. Taking a look into one stage deeper



(a) Class 1 Explanation (b) Class 2 Explanation (c) Class 3 Explanation

Figure 15: Shap values in a value by value level

into the decision making process of the algorithm, Figure 15 is produced.In Figure 15 (a) we see the most influential features in descending order. Firstly, in all three graphs, the SHAP values of all the features are centred around zero. So that indicated that the distribution of the test dataset. Secondly, as explained in Figure 14, *residual.sugar* is the first most important feature in predicting class zero. However, this graph also provides further granularity. In particular, when the *residual.sugar* values are low, this feature has a low impact in classifying the datapoint as class zero, and oppositely, when the value *residual.sugar* then the feature has a very strong influence on the "decision" of the algorithm to classify the datapoint as class zero. The same logic can be extended to the other two classes and the other variables. While this model explanation did not add anything new to the analysis so far, it is, regardless, a critical step. Understanding why the model predicts this specific way and not in some other is vital for the workflow, confirming or rejecting our prior beliefs. In this particular case,

ID	pred 1	pred 2	pred 3	True Label	CI Range
542	0.992509	0.000377	0.007114	3	0.042003
357	0.772490	0.000741	0.226768	1	0.041914
804	0.915421	0.000747	0.083832	3	0.039591
112	0.308868	0.010652	0.680480	3	0.039252
275	0.382482	0.012079	0.605438	1	0.039077
831	0.999840	0.000150	0.000011	2	0.000368
717	0.999962	0.000033	0.000005	2	0.000360
721	0.999984	0.000007	0.000009	3	0.000321
866	0.999868	0.000069	0.000036	2	0.000274
878	0.999894	0.000057	0.000075	2	0.000265

Table 1: Top five smallest and largest CI ragnes for `type 2`

the prior knowledge gained in EDA seems to be confirmed.

4 Results

The last important task is to produce a measure of uncertainty. This will help to automatically identify datapoints that are anomalous. The assumption is that if the Neural Network is uncertain as to how to classify a specific wine, then that wine needs human judgment to be classified correctly. In order to produce a measure of uncertainty, firstly we need to sample from the posterior distribution. This is being done by drawing $m = 1000$ predictions from the test set. In essense, the test set is being passed through the MC dropout neural network 1000 times, generating a different probability for each class in each pass due to the Bayesian nature of the algorithm. Then, the average per class is calculated for the 1000 predictions on each row of the dataset. These are the average probabilities for each class generated by the Bayesian neural network.

In the rest of section 4 three candidate measures will be presented.

4.1 Confidence Intervals

Since there are m samples per observation, the variance and hence the standard deviation of the prediction can be measured. This is a first approach in trying to capture uncertainty. If a prediction has a big standard deviation, that means that in each pass through the network, the predicted class turns out to be a different one everytime, essentially the algorithm doesn't know. what the predicted class is.

The Central Limit Theorem suggests that any aggregate function of data, such as their arithmetic mean ($\bar{x}$), for a large enough sample (with replacement), that aggregate function approaches a normal distribution. Hence, we can can create confidence intervals on the sampled prediction according to the following formula:

$$\mathrm{CI} = \bar{x} \pm z\frac{\sigma}{\sqrt{n}} \tag{4.1}$$

Where $\bar{x}$ is the arithmetic mean over the n predictions drawn and σ is the standard deviation of them, and z is the Z-value representing the confidence level (for examle, $z = 1.96$ produces a 95% confidence interval).

This approach has various caveats. First of all, the probabilities produces by the algorithm for each type, are not exactly probabilities. They are generated by passing the final output of the neural network through a softmax function. While this function does produce an output between zero and one, this on its own does not mean that these are probabilities. Hence, by calculating the standard deviation and producing the confidence intervals it is possible to find an upper bound for the probability that is bigger than one or a lower bound that is less than zero. Both of this cases do not make sense, and this is happening exactly because we interpret as probability something that is not. Nevertheless, a closer look at the output of this metric can be presented.

In Table 1 we see the biggest and smallest confidence interval ranges for the predictions of `type 2`. This was choosen as it was abandontly clear from the EDA and the confusion matrices that the algorith find particularly difficult to identify this type of wines. In the first five examples, the confidence interval is the largest possible, implying that the algorithm is uncertain about the type of those specific datapoints being `type 2`. In this example this turns out to be correct as no observation is of true label 2. In contrast, the second five examples, the algorithm predicts with high probability that the observation is of `type 1` and assigning almost zero probability of it being `type 2`. While the majority of the true labels are of `type 2`, the results are not interpretable, hinting that a different measure should be explored.

4.2 Entropy

A more robust measure of uncertainty, is the informational **entropy** or Shannon entropy [8]. A random variable X with a PMF p, has entropy $H(X)$ and it is a measure of it's uncertainty. Entropy is defined as:

$$H(X) = -\sum_{k=1}^{K} p(X = k) \log_2 p(X = k) \tag{4.2}$$

For this project, $k = 3$ are the three classes. Entropy is a function that is maximum when $p(X = k) = \frac{1}{K}$. Therefore, when the algorithm predicts $\frac{1}{3}$ for all three classes, then the entropy is maximized wich is exactly the objective. In contrast, a distribution where all the mass is in exatly one possible outcome, has the minimum amount of entropy possible since that specific distribution has no uncertainty related to it.

ID	pred 1	pred 2	pred 3	Entropy	ID	pred 1	pred 2	pred 3	Entropy
167	0.358059	0.267728	0.374213	1.570201	866	0.999341	0.000381	0.000278	0.008561
530	0.400761	0.326552	0.272687	1.567135	721	0.999180	0.000524	0.000296	0.010363
473	0.284649	0.424455	0.290896	1.558966	878	0.999162	0.000546	0.000293	0.010560
947	0.233179	0.385329	0.381492	1.550319	831	0.999153	0.000453	0.000394	0.010710
780	0.419048	0.232465	0.348487	1.545133	749	0.999103	0.000341	0.000556	0.011229

(a) Most uncertain (b) Least uncertain

Table 2: Top five most and least anomalous datapoints by Entropy

Indeed in *Table 2* we see this relationship. The top five most uncertain predictions, as classified by Entropy, are the ones that are almost random. On the contrary, the least uncertain predictions, are the ones that are predicted with highest probability. This all happen to be of `type 1`, because we know that the algorithm misclassifies `type 1` far fewer times than the other classes. This doesn't mean that the low entropy predictions are correct while the low entropy are incorrect. Entropy is just a way to summarize, the average probabilities produced for each class in a single value. By setting a threshold t on that number, then we can easily create a binary measure of uncertainty. The level of that threshold represents the tolerance the stakeholder has on anomalies and it is a business desition.

$$f(\text{entropy}) = \begin{cases} 1 & \text{entropy} \geq t \\ 0 & \text{entropy} < t \end{cases}$$

5 Discussion

In this dissertation project, a Bayesian Neural Network was trained on a dataset about wines, with the goal of capturing the uncertainty of the data. Beyond that, an extensinve exploratory data analysis, along side a robust explanation of the model where conducted, that worked as a validation that the specific model works as intended. The reason these parts where empasized, is that a machine learning algorithm that is very good in predicting an outcome due to deep pattern recognized on the training dataset, might be excelent in a vacuum, yet it could cause harm when productionized and used in a real life application. With the EDA conducted first, building a ground understanding about the dataset, and a deep dive in the model explanation with the Shap values, we can be more confident about the generalization of the model.

As for the Bayesian approach to the algorithm, far more complidated models could be built that lift the assumptions of normality and produce better results. A typical example of a caveat of the dissertation project it the Variational inference model.

Lastly, the uncertainty measure of Entropy seems to work well enough for the specific application. However, in the context of a regression problem, rather than the classification problem tackled, this measure would need readjustment in order to produce a measure of uncertainty.

References

[1] C. C. Aggarwal. An introduction to outlier analysis. In *Outlier analysis*, pages 1–34. Springer, 2017.

[2] T. Auld, A. W. Moore, and S. F. Gull. Bayesian neural networks for internet traffic classification. *IEEE Transactions on Neural Networks*, 18(1):223–239, 2007.

[3] T. Brotherton and T. Johnson. Anomaly detection for advanced military aircraft using neural networks. In *2001 IEEE Aerospace Conference Proceedings (Cat. No.01TH8542)*, volume 6, pages 3113–3123 vol.6, 2001.

[4] J. Goh, S. Adepu, M. Tan, and Z. S. Lee. Anomaly detection in cyber physical systems using recurrent neural networks. In *2017 IEEE 18th International Symposium on High Assurance Systems Engineering (HASE*, pages 140–145, 2017.

[5] M. Gutmann and A. Onken. Lecture notes in data mining and exploration, January 2021.

[6] K. Miok, D. Nguyen-Doan, D. Zaharie, and M. Robnik-Šikonja. Generating data using monte carlo dropout. In *2019 IEEE 15th International Conference on Intelligent Computer Communication and Processing (ICCP)*, pages 509–515. IEEE, 2019.

[7] I. Murray and A. Onken. Lecture notes in machine learning and pattern recognition, Autumn 2020.

[8] C. E. Shannon. A mathematical theory of communication. *The Bell System Technical Journal*, 27(3):379–423, 1948.

[9] L. S. Shapley. A value for n-person games. *Contributions to the Theory of Games*, 2(28):307–317, 1953.

[10] D. Tran, M. W. Dusenberry, M. van der Wilk, and D. Hafner. Bayesian layers: A module for neural network uncertainty. *arXiv preprint arXiv:1812.03973*, 2018.

[11] L. Van der Maaten and G. Hinton. Visualizing data using t-sne. *Journal of machine learning research*, 9(11), 2008.