The School of Mathematics

THE UNIVERSITY *of* EDINBURGH

Anomaly Detection with Bayesian Neural Networks

by

Michael Petrouleas, s2123629

Dissertation Presented for the Degree of

MSc in Statistics with Data Science

July 2021

Supervised by

Dr. Michael Allerhand, Dr. David Elliot and Mr. Kostas Tampourakis

Acknowledgments

I would like to thank our supervisors Dr. Michael Allerhand, Mr. Kostas Tampourakis and Dr. David Elliot, for providing us with helpful instructions throughout the meetings and Piazza forum, as well as Mr. Allistair Hammilton for taking the time to join our meeting sessions and providing weekly valuable information on how to handle each case. I would thus also like to give thanks to Dr. Daniel Paulin, for providing us with the much useful textbooks on Piazza, way before the beginning of the project. Working on neural networks was something I never thought I would be able to do last year, as the whole notion with optimizing functions and training models seemed intimidating for a code illiterate person. These textbooks were decisive in helping me demystify that case.

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: Michael Petrouleas
Matriculation Number: s2123629
Title of work: Anomaly Detection with Bayesian Neural Networls

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 speci_ed 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 [illegible]

Date: 2/7/2021

Executive Summary

Frequentist Neural Networks (NNs) are widely used state-of-the-art models that are employed in a variety of applications and manage to tackle prediction (regression and classification) tasks with high performance. One major setback these models have is linked to their interpretability. In a real-world setting, where the input data is arbitrary and contains outliers and out-of-distribution (OOD) points, predictive outputs provided from these models are often difficult to be evaluated based on their accuracy and, often, these values are taken for granted. For this reason, different models that consider model uncertainty need to be employed. In this report we will define some of the existing types of uncertainty that can influence these predictions, establish the frequentist NN limitations and investigate the Bayesian Neural Networks, models able to quantify confidence placed on the predictions. Lastly, we will try to assess how sensitive the predictions of these models are in uncertain areas of the feature space, by providing visualization plots and confidence/predictive intervals. Although the analysis consists mainly of a classification task, a short review of results examined in a regression setting are provided. The report concludes with areas of improvement and potential limitations that occurred throughout the process.

Word Count (without Table elements & Figure Text): 4836

List of Contents

List of Figures

List of Tables

1. Introduction

1.1 Background and Motivation

Man has always tried to conquer uncertainty and create predictive models that capture it adequately, providing as precise forecasts as possible. Artificial Neural Networks (ANNs) are such recently developed state-of-the-art models. They use high dimensional feature vectors as inputs and, through complex computations that are difficult to be tracked (Kemp et. al 1997), create outputs/predictions that manage to combat classification and regression tasks with high accuracy, often surpassing this of a human.

However, such produced outputs are difficult to be evaluated based on their correctness, which is why they are often called "black-box" models (Kemp et. al 1997). By using a base frequentist neural network, we place our confidence on how good these predictions are, simply by looking at certain metrics evaluated on how well the model performed on an, unknown for it, test dataset. Yet, in real world examples (ex. Autonomous vehicles that use data collection and make decision on real time (Gal, 2016), where the input data is more arbitrary and anomalous, as it can include rare unforeseen observations that the original training dataset would not be able to contain (Karmakar & Pan, 2018), this can lead to misclassifications that can lead to devastating outcomes (Harris, IEE Spectrum 2019). It would be thus ideal to employ a method of adding a measure of uncertainty to these predictions plus calibrate the model to provide them according to its confidence.

Useful for such a scenario is identifying what type of uncertainty our problem is containing (Kiureghian, 2009). On one hand, there is the epistemic uncertainty. This type of uncertainty contains the variance, added to possible predictions, that is generated by a small training sample (i.e. not enough information is available for the model to come to conclusions) plus all hidden factors which, if included in the analysis, the said model could improve its confidence of the predicted output. It thus is an uncertainty captured adequately by estimating the parameters/weights of our model as precisely as possible (Kendall & Gal, 2017), which occurs by adding more data. On the other hand, there is the aleatoric uncertainty, which is inherent noise on the observations (Kendall & Gal, 2017) and, despite the analyst's efforts, cannot be further reduced. It is usually modelled by placing a distribution over the output and examining its variance.

It needs to be made clear at this point that the notion of these two uncertainties changes by the situation one is facing (Kiureghian, 2009). For some problems, aleatoric uncertainty might be perceived as epistemic and vice versa. Also, these two types of uncertainties can be proven hard to be separated.

This report aims to explore this phenomenon by establishing, through simple examples, where frequentist ANNs are lacking, when it comes to measuring predictive uncertainty to outcomes. We will provide alternative models that can remedy this case, namely, Bayesian Neural Networks (BNNs), that can capture and quantify this uncertainty on the predictions (Lingxue et. al,2017). The results will be mainly investigated on a classification scenario while the same phenomenon will be shortly evaluated on a regression example.

1.2 Data

In this analysis we will use the "wine" dataset (Cortez et. al, 2009). It consists of 4989 Portuguese white "Vinho Verde" wine samples of three types (categorized by the grape used in their production), each of which is assigned with a discrete number ranging from 0 to 10 that describes the wine quality in an increasing order, as evaluated by wine experts. The remaining 11 features in the dataset describe the physiochemical characteristics per sample of wine. It is thus easy to define a multiclass classification task, where the features per wine observation are used in the training process with the wine type being the target outcome. A regression task, which will serve as a prompt example of the notion we are investigating, can also be defined, by identifying the wine quality as the target output of the model.

2. Exploratory Data Analysis

Plotting the histograms per feature and type can give a clear view of the variable distributions to the reader. Useful descriptive statistic metrics are also given below.

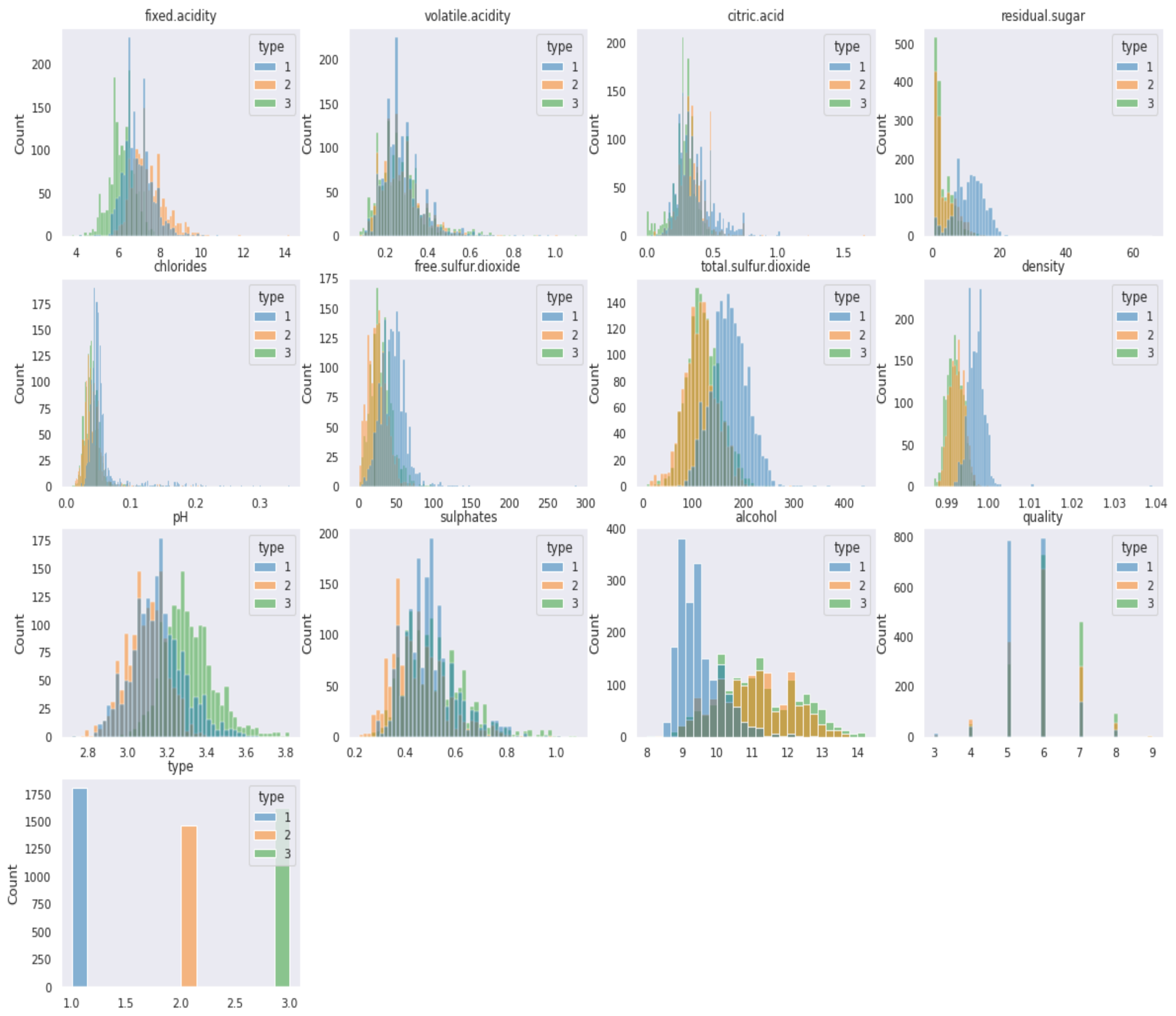


Fig 1. Histogram plots and descriptive statistics per variable (by type)

The dataset contains no missing values. As it is also evident, the number of type 1 wine observations is greater than the other two. However, there seem to be features whose distributions per type are distinguishable from each other. Specifically looking at the histograms and descriptive statistics (Fig.1) of the variables "Residual Sugar", "Total Sulfur Dioxide", "Alcohol" and "Density", observations originating from type 1 wine seem to have a different peak, mean and spread (judging by the relevant quantiles and standard deviation) values than the other two types, whose histogram plots seem to highly overlap. The "pH" variable seems to also be helpful in defining the type of wine, with type 3 wines having overall larger values of pH than the other two.

Another important observation is that the data is quite unbalanced when it comes to quality. While there seems to be an adequate sample of 5-7 rated wines, observations for 4 and 8 quality rated cases appear to be scarce while those lower than 3 or higher than 9 are absent. Therefore, a model trained for a regression task will possibly struggle to generalize or predict accurately such unseen cases.

The importance of each feature in helping towards the success of each task (regression/classification), can be evaluated more formally by using the results (Scores) of the "Extremely Randomized Trees" algorithm (random seed used = 4), a decision trees variation (Geurts et. al, 2005). (Fig 2.)

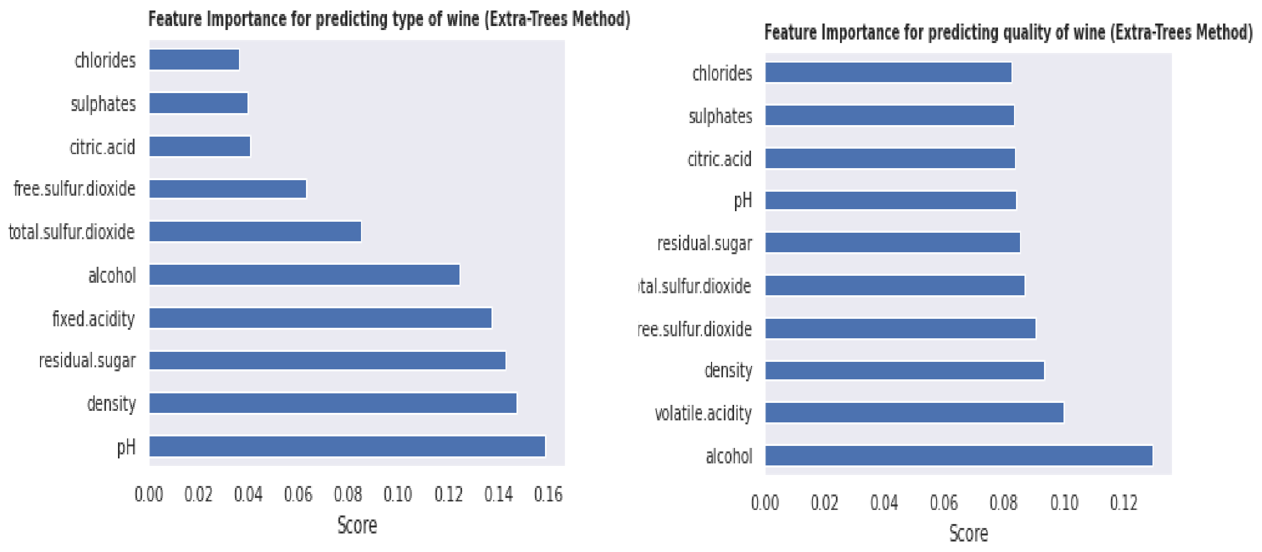


Fig 2. Feature importance scores for a wine type classification (left) and a wine quality prediction (right), ranked from lower to highest.

As the plots illustrate, many of the variables described above, along with fixed acidity, have the utmost importance for a classification task, whereas, alcohol seems to be the most defining factor in predicting the quality of each wine observation, with the other features attaining quite similar scores.

Another useful tool to explore how distinguishable the observations per type are, is by performing "principal component analysis" (PCA). By scaling the data, decomposing the correlation matrix (Fig.4) to its corresponding eigenvectors, or principal components (PCs), and transforming the already existing variables into new ones, the utmost variability within the data can be captured into a p-dimensional subspace of the original k-dimensional feature space ($p < k = 11$ features) (Joliffe, 2016). Overall, PCA is a method of dimensionality reduction, useful to make data representable in lower dimensions while simultaneously keeping most of the information contained in them. Usually, it is advised to keep PCs based on a criterion set by the researcher (percentage cut-off of captured variability, marginal variance explainability contribution of each added PC etc.). However, purely for visualization purposes, a 2-component PCA, which captures just 43,6% of the variability in the data, is demonstrated and its corresponding 2D plot is given below (Fig. 3). As the 2D plot also implies, the three types of wine can be mostly classified successfully, given the features of the dataset. The loading plot, which is depicted by red vectors assigned to features, is also included and indicates how much each variable contributes to each component.

2 - PCA Plot

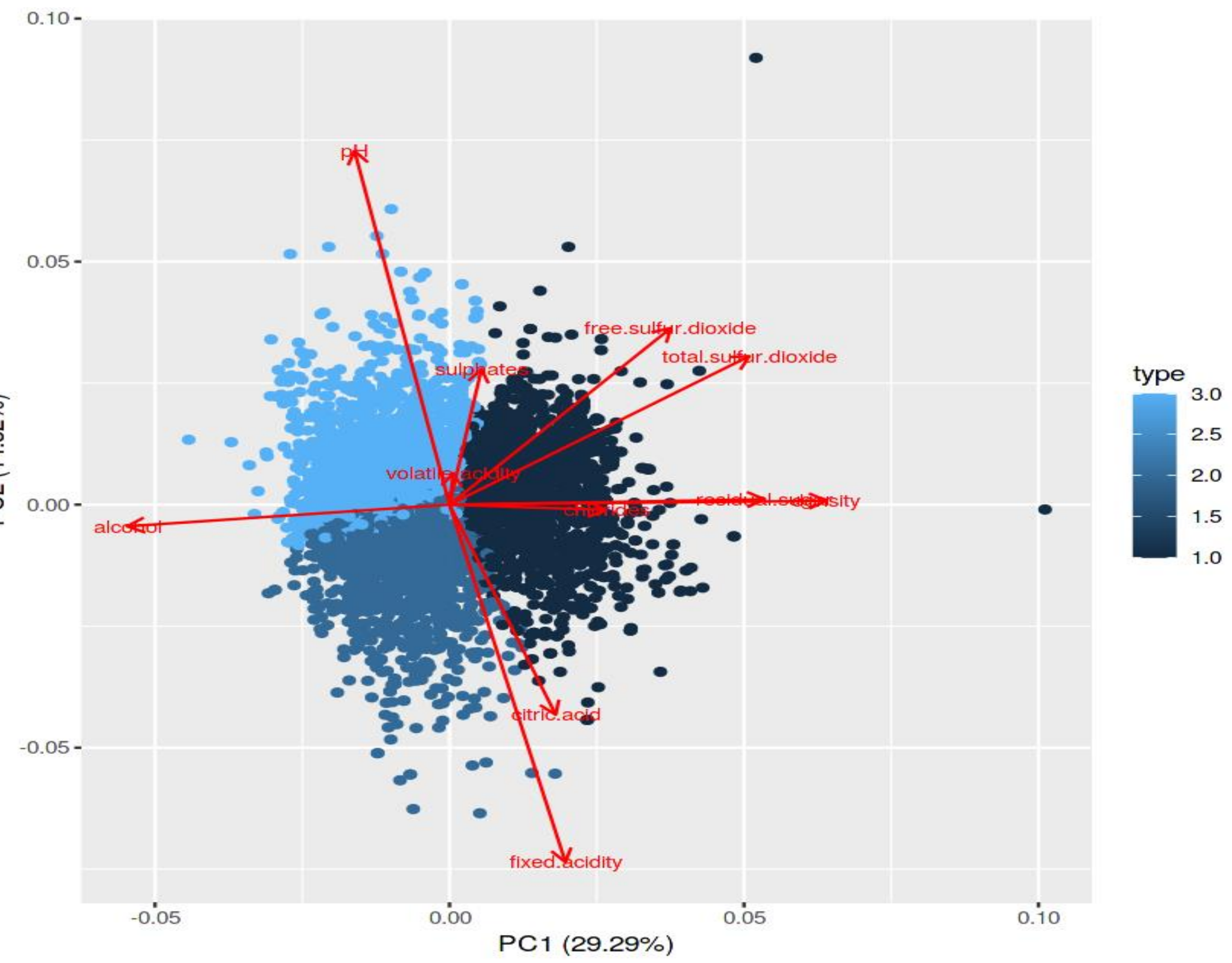


Fig 3. 2-PCA biplot (PC scores and loading plot). The percentage of explained variability equals the sum of the one explained by the two PCs (43,6%).

Additional information that can be extracted by the PCA plot involves the type of correlation between variables. In particular, feature vectors, represented on the plot, that form small angles are positively correlated (ex. residual sugar - density) whereas those that form 180^o angles are negatively correlated with each other (alcohol - residual sugar / density). Lastly, loadings that form 90^o angles, have relatively low correlation with each other.

These verdicts can be also evaluated by a heatmap of the Pearson-correlation matrix of the dataset (Fig. 4). As seen by the graph, alcohol has indeed a negative correlation of 78% and 45% with density and residual sugar respectively, meaning that as one variable tends to increase the other follows an opposite linear trend. Although the magnitudes depicted are not alarming, high correlation between features implies a strong linear trend, which can be problematic in simpler models that incorporate assumptions about the independency of the variables. As more complex models that take into account potential unseen relationships between all variables will be used, plus emphasis is placed in anomaly detection, this phenomenon is of no high importance for this report. In the same sense, it is useful to inspect the "quality" row, where "alcohol" and "density", the variables highest ranked in the feature importance plot (Fig. 2), achieve relatively medium correlation and thus are quite informative for our predictions.

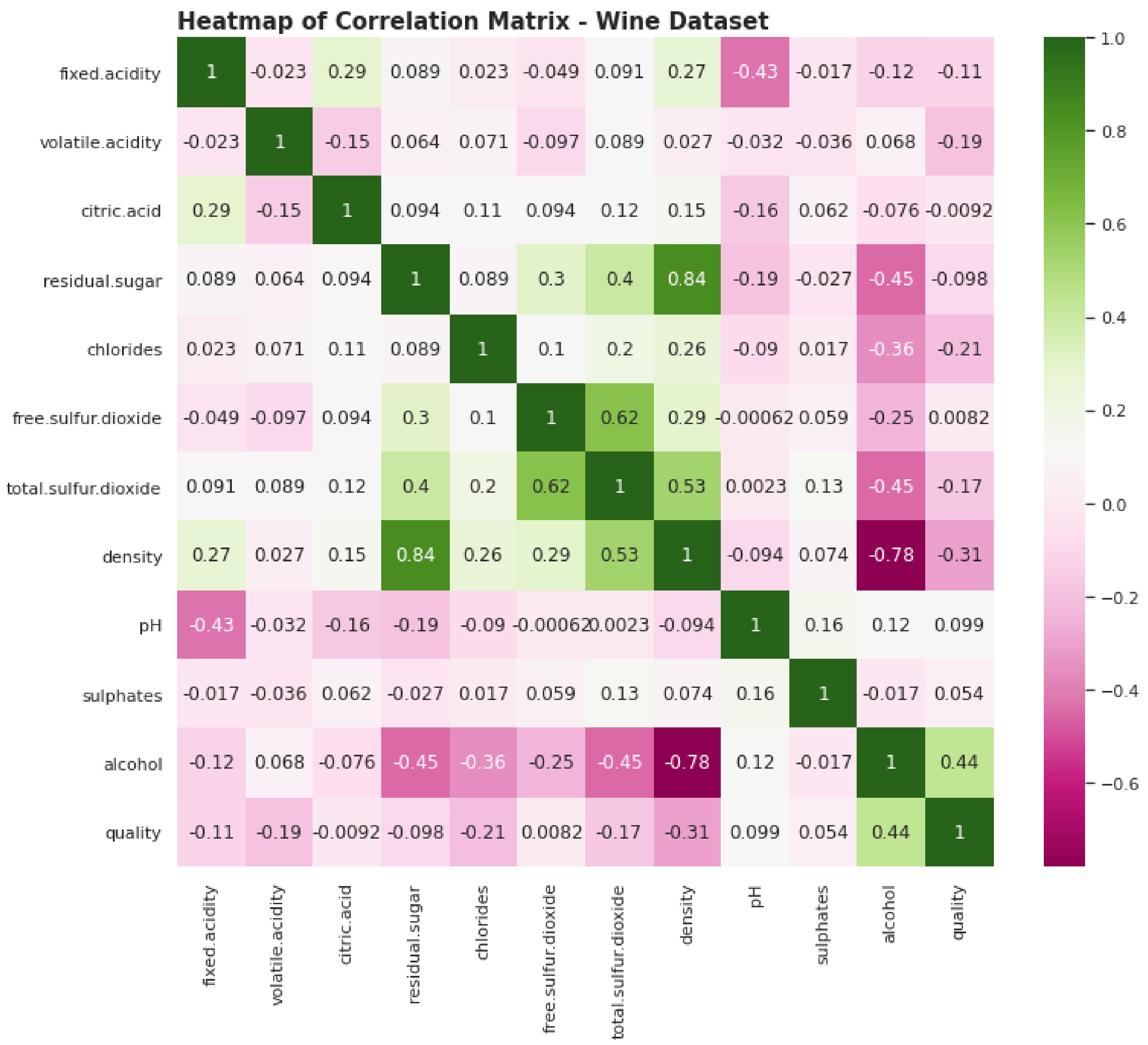


Fig. 4 Heatmap of correlation matrix of dataset

Overall, the representation of the data in two dimensions reveals that noticeable differences exist between the three types of wine, as observations per type are clearly organized into clusters. Still, the boundaries between classes are far from being distinct. By looking at the plot, there are areas between each of the formed classes' borders that overlap with each other. It is expected that cases that belong to these arbitrary areas will prove hard to be classified by the model, producing predictive inaccuracies that are difficult to be distinguished by the user.

3. Methods

3.1 Overview

In this section a regression and a classification task for the dataset will be defined and the appropriate models, target outputs, loss functions and notions will be given. To optimize the models, the "Adam" algorithm, a variation of stochastic gradient descent (Kingma & Ba. 2014), is used with learning rate equal to 0.05. To avoid randomness in the analysis induced by the initialization of weights and the optimization process, a seed equal to 0 was set for all the models. Lastly, as the size of the feature vector X is equal to 11, a layer with 11 units is used as an input layer among all neural network models, whereas all other layers, besides the one defining the output, use the "relu" activation function. To refer to the output's dependence on the weights W, which connect the in-between layers with each other, and the input vector X, the notation $f^W(\mathbf{X}) = \hat{y}$ will be used. The true target observations, y, are thus defined as $y = \hat{y} + E_{ep} + E_{al} + \varepsilon$, where E_{ep} and E_{al} are measures of noise that quantify epistemic and aleatoric uncertainty in the model, respectively. The value ε is reserved for noise added from other types of uncertainty.

To avoid overfitting, the overall dataset will be randomly split into two train and test samples of sizes 3918 (80% of dataset) and 980 (20%) respectively. For reproducibility reasons and without compromising the results a seed = 42 was set into the splitting function. A validation set, equal to 10% of the training set, is also reserved for hyperparameter tuning, which also helps with visualizing and comparing the process of loss minimization between the sets. The training section will thus serve the process of estimating the model parameters (weights for the frequentist or their posteriors for the Bayesian counterpart) while simultaneously minimizing the loss function. The model's performance will be evaluated on the unseen test data. However, as our goal is to lower epistemic uncertainty as much as possible by providing a large enough training set, we will use the whole dataset for the probabilistic-BNN (3.4).

3.2 Frequentist Neural Network (NN)

It would be useful at this point for the reader to gain a notion of how a NN operates and why, in order to adequately model uncertainty, alternative models need to be explored.

In this stage, a NN with one hidden layer that consists of five units and an additional layer defining the shape of the output, $\hat{y}$, is produced. For the classification task, the three-unit output layer's activation function will be the "SoftMax" function, which assigns percentages that reflect the model's confidence of an observation belonging to each of the three classes. For the regression task, a one-unit linear output layer will be used. Moreover, loss functions, for the NN to improve its confidence over each epoch, are defined per task and consist of the Mean Square Error (regression) and the Categorical Cross-Entropy (classification).

From the process described above, the baseline model will reach a set of optimal for it weights, **w*,** among all layers, and, since these weights are definite, predictions that use a common input **x*** will always generate the same output $f^{\mathbf{w}*}(\mathbf{x}^*) = \hat{y}$. Although in a controlled setting we can evaluate the fitted values against the true observed ones y, in a predictive environment, where only the input $\mathbf{x}_{\mathbf{new}}$ is available, this will not be the case. The testimony, whether this predicted $f^{\mathbf{w}*}(\mathbf{x}_{\mathbf{new}})$ value is correct or not, or how uncertain the model is about this prediction is at this point not able to be evaluated.

We thus need a measure to capture how confident the model is about these assigned weights. In quantitative terms, this refers to the standard deviation of their estimated values or, otherwise, the epistemic / reducible-given- enough-information, uncertainty. As simple scalars have a variance and standard error of 0, defining an adjustable distribution over the weights will be an appropriate method to capture it.

3.3 Bayesian Neural Network (BNN)

In order to model epistemic uncertainty, we will use a BNN. Thus, priors defined as Normal (mean = 0, scale = 1) will be set on the weight vectors. The corresponding weight posteriors (Normal distributions with trainable parameters) will be approximated with variational inference, by minimizing the Kullback-Leibler divergence function. For this model, layers that implement the "Flipout" method (Wen et. al 2018) were used.

Due to the stochastic nature of the BNN, the priors of the weights will gradually update and approximate their true posterior, over many training epochs. What varies from the frequentist NN is that now, each time we want to draw a prediction from the model, we have to also obtain a sample from each weight's posterior distribution to calculate it. Thus, the output, albeit a point estimate of the value we want to predict, will be stochastic in nature. This means that multiple draws, even if the same input **x*** is inserted to the model more than once**,** will always produce a different output. Also, the larger the training set is, the more accurate will be the representations of the weight posterior distributions. This in turn lowers the predictive standard deviation related to epistemic uncertainty.

Due to this stochastic nature of the predictive value, the output's distribution per observation can be accessed by performing several (1000 in our case) iterations of the prediction process. Sampling from the trained weights posterior distributions per iteration, on the test data, ensures that we can get different SoftMax probabilities on each observation per iteration. We expect that correctly classified observations will tend to have higher assigned SoftMax probabilities to the respective class (Hendrycks, 2018) whereas, ambiguous areas of anomaly will produce overlapping distributions.

In other words, we run a Monte Carlo Experiment on an already trained model and observe its results and performance, evaluated on a test dataset. Through the standard deviation of the predictions, one can thus measure the confidence of the model in each area of the feature space. It Is also possible to construct approximate 95% predictive intervals in the form of $[\bar{y} - 1.96 * sd_y, \bar{y}\ + 1.96 * sd_y]$, where $\bar{y}$ is an estimate of the mean prediction and sd_y its corresponding standard deviation.

Although a simplistic method, adding restrictions on the extracted output distributions can identify observations that the BNN model struggles to predict accurately. For the classification task, a threshold of any median, per SoftMax assigned probability distribution on every class, being higher or equal than 0.5 is set. For the regression task we simply evaluate the "quality" predictions based on the magnitude of their standard deviation. As we will note in the results, ambiguous areas on the feature space have proven to produce the highest predictive uncertainty.

3.4 Probabilistic Bayesian Neural Network (p-BNN)

Lastly, in order to include aleatoric uncertainty (E_{al}) in the model, a probabilistic BNN model that produces a distribution per data point instead of a probability vector (classification) or single output (regression) will be set. In that case, the mean and standard deviation can in this case be accessed immediately without a simulation experiment and confidence intervals van be assigned to each prediction.

As we have a distribution on the output, and thus it would not be logical to use a usual loss function, the negative log likelihood will serve this purpose and its value over the outputs is the one that should be minimized in this setting.

4. Results

4.1 Classification task

4.1.1 Performance of Models

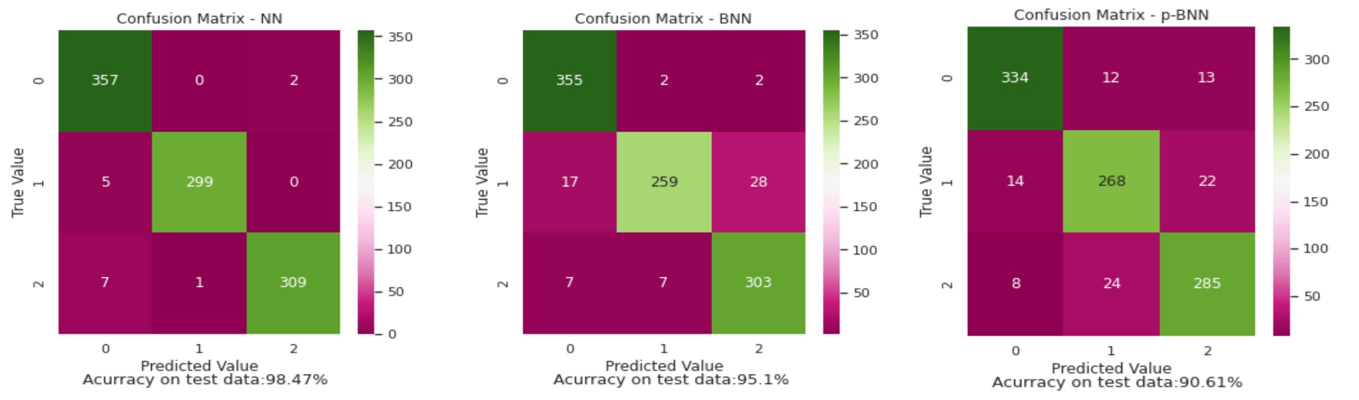


Fig. 5 Representation of Confusion Matrices on the test dataset per model.

In the figure presented above, we can evaluate each model's predictions based on the true observations. It is notable how well the baseline NN performs on the test data, producing a 98,47% accuracy. It also appears that, due to the added complexity and uncertainty, the BNN and p-BNN models have produced lower accuracy metrics, even though in the latter case the whole dataset was used in the training phase. By looking at the corresponding rows and columns of the confusion matrices, the models mostly struggle to make the distinction between type 1 and 2 wines. Although the NN is the preferred model, simply judging by the accuracy, we will establish that there are areas where a clear distinction between classes is hard to be made.

4.1.2 Capturing Epistemic Uncertainty with a BNN

As described previously, after conducting a simulation experiment with 1000 repetitions, we can access the predictive distributions per input. By setting a threshold-lower bound (>=0.5) on the medians of SoftMax distributions, our predictions on the test set can now be categorized as valid (890 in total), those that meet this criterion, or invalid (90 in total).

Ambiguous areas, where invalid observations of the dataset were captured, can be thus visualized on a revised version of the PCA plot (Fig. 6). As it is demonstrated, these area falls along the lines of the boundaries between the three wine types.

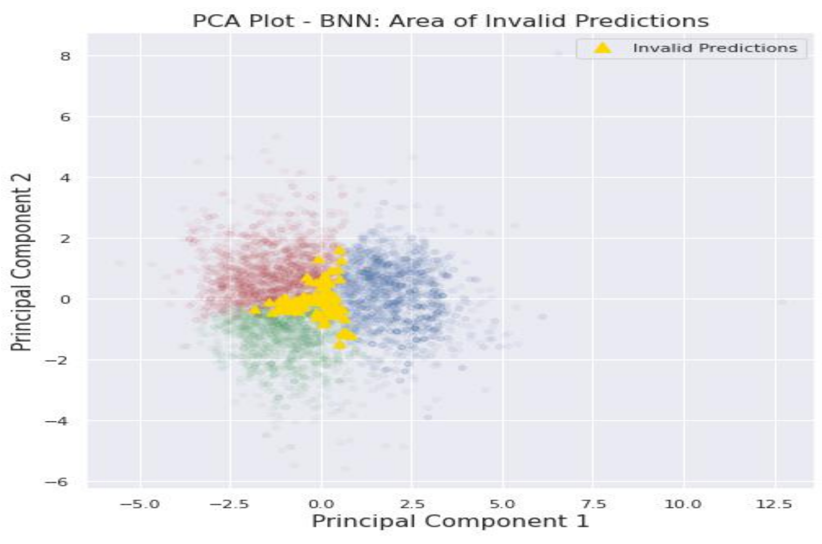


Fig. 6 Representation of invalid test cases (median of SoftMax probability distributions per class <0.5) on a 2-dimensional PCA plot for wines 1 ,2 and 3, as shown in Fig.3. Class 0 is shown in Blue, Class 1 in Green and Class 2 in Red.

It is notable to investigate how these distributions per class are formed, both on invalid and valid predictions. Some of these cases, extracted from the test dataset are presented in Figures 7 and 8.

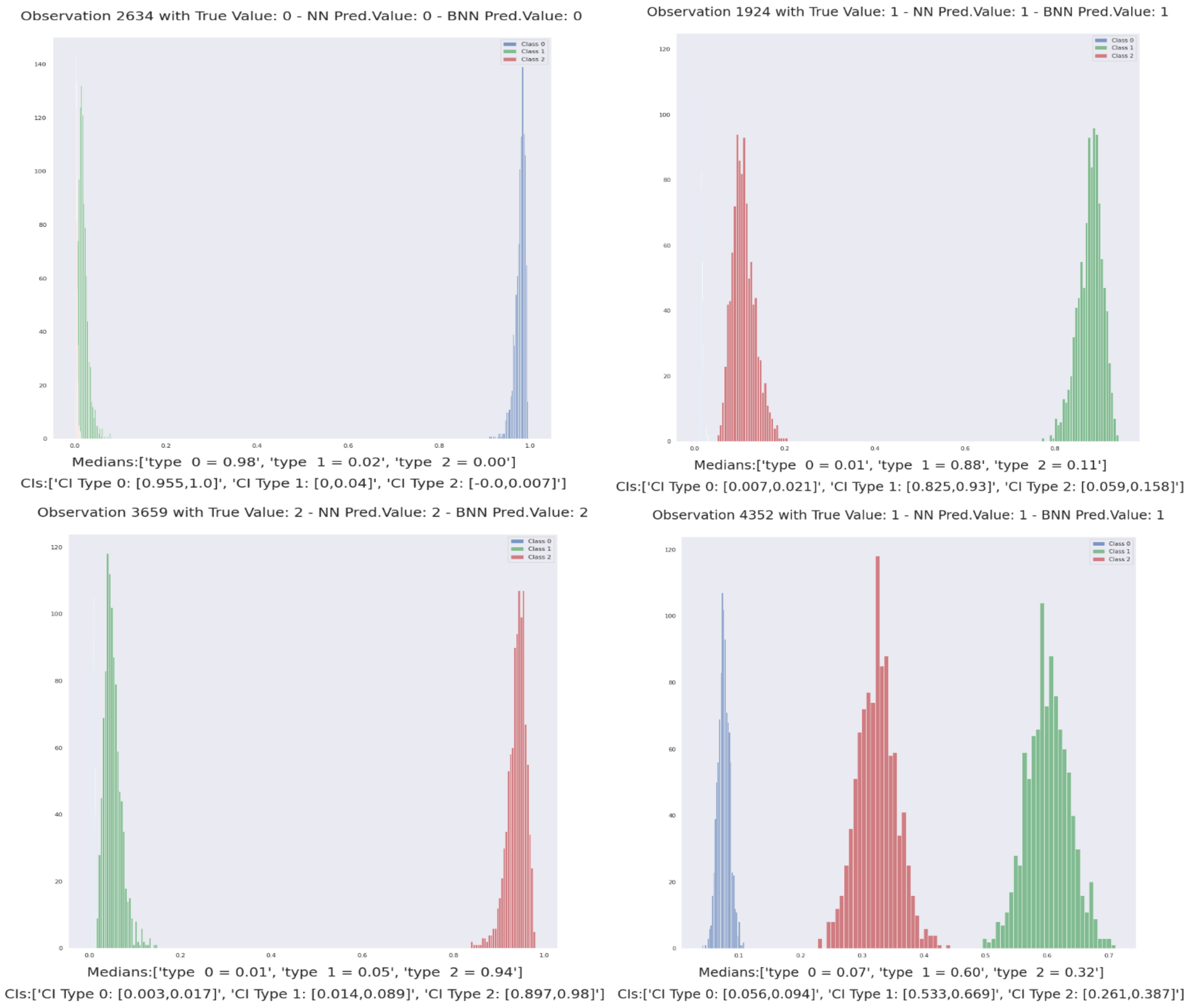


Fig.7 Valid classifications (at least one median of predictive probability SoftMax distribution >= 0.5) - Assigned SoftMax probabilities per class, over 1000 iterations of the Monte Carlo simulation experiment. Clear classifications are portrayed in the top-left for class 0 (1), top-right for class 1 (2) and bottom-left for class 2 (3) histograms. A borderline case (highest median being 0.60) is shown in the bottom-right (4). The observation index, on the whole unseparated dataset, and its corresponding True Value, Frequentist NN Predicted Value and initial BNN Predicted Value are included on the title. Confidence intervals for the probabilities per class and distribution medians are also provided.

It is clear that cases (1), (2) and (3) could be easily classified by the model. It is also noteworthy that in the borderline case presented above (4), all the distributions are clearly separable from one another and the model managed to accurately predict the correct true value. This is of course not surprising as, despite the captured uncertainty between classes, in the SoftMax multiclass classification case, the class whose probability distribution is positioned at the furthest right area of the horizontal axis (largest assigned probabilities throughout the MC experiment) will always define the predicted case from the model.

To investigate how uncertainty is visualized in the rest of the observations, we will now examine some of the invalid predictions on the test dataset. In the same manner, the histograms of four observations, some of which were misclassified by the frequentist NN and initial prediction of the BNN, are presented below.

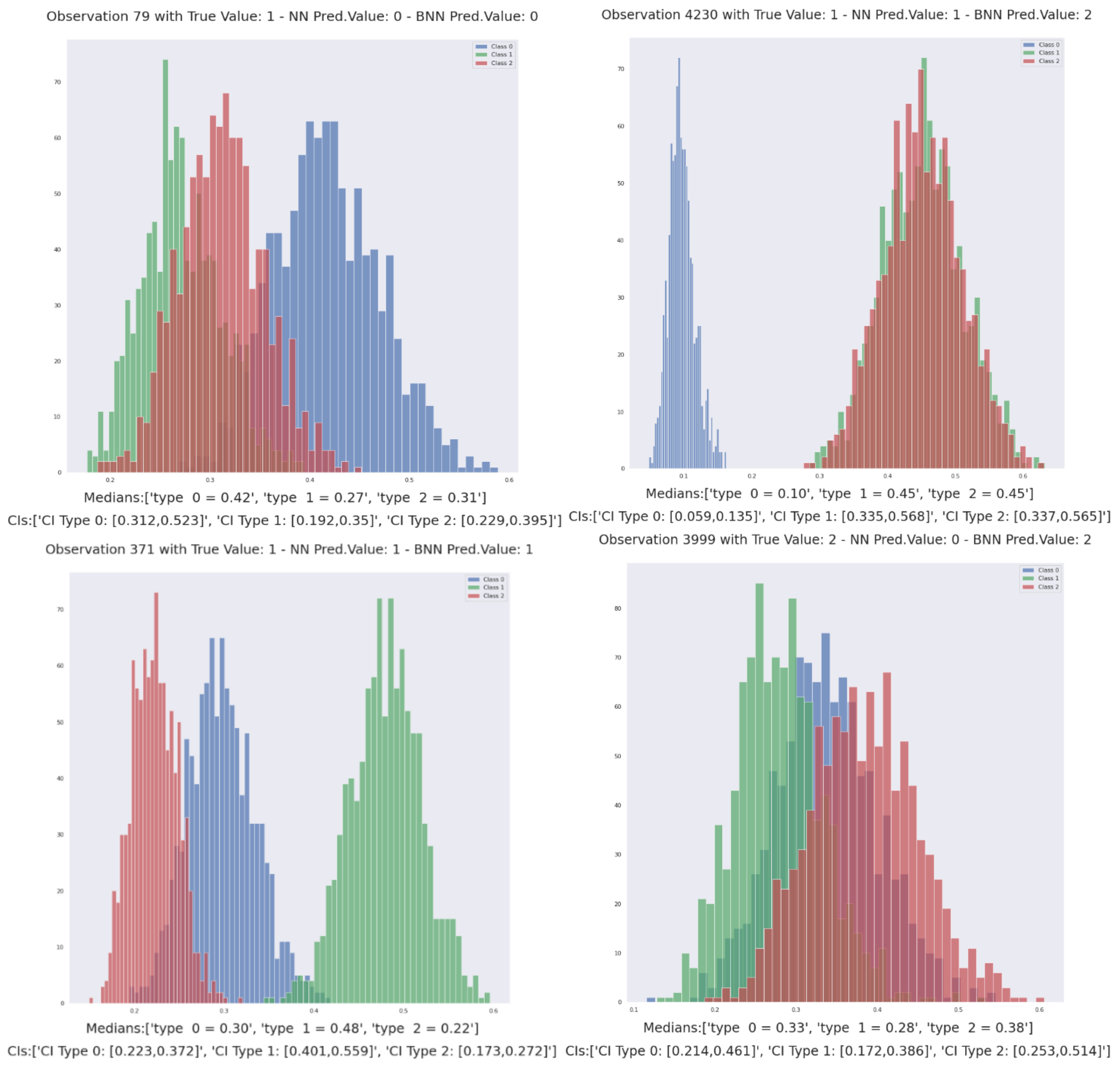


Fig.8 Invalid classifications (all the medians of predictive probability SoftMax distribution < 0.5). Presented cases include: top left –incorrect classifications by both models (5), top right – almost identical class 1 and 2 distributions (6), borderline case with correct predictions but highest median lower than threshold (7), highly ambiguous case with all medians approaching 0.33 (8).

SoftMax predictive distributions of this type of anomalous, in the sense that they are hard to be classified, observations appear to overlap with each other. Standard Deviations and, thus, Confidence Intervals per class are wider than valid cases, something that also signifies the model's reduced confidence. In the BNN setting it is thus a matter of pure chance (defined by posterior draws within the model) on whether the model will correctly classify each invalid observation. Cases (5) and (8) are clear example of inputs that produces equally uncertain classes, as medians over 1000 iterations are close to 0.33.

To get a clear notion of where these valid and observations lie on the feature space, it would be useful to reexamine the PCA plot and pinpoint their location (Fig.9).

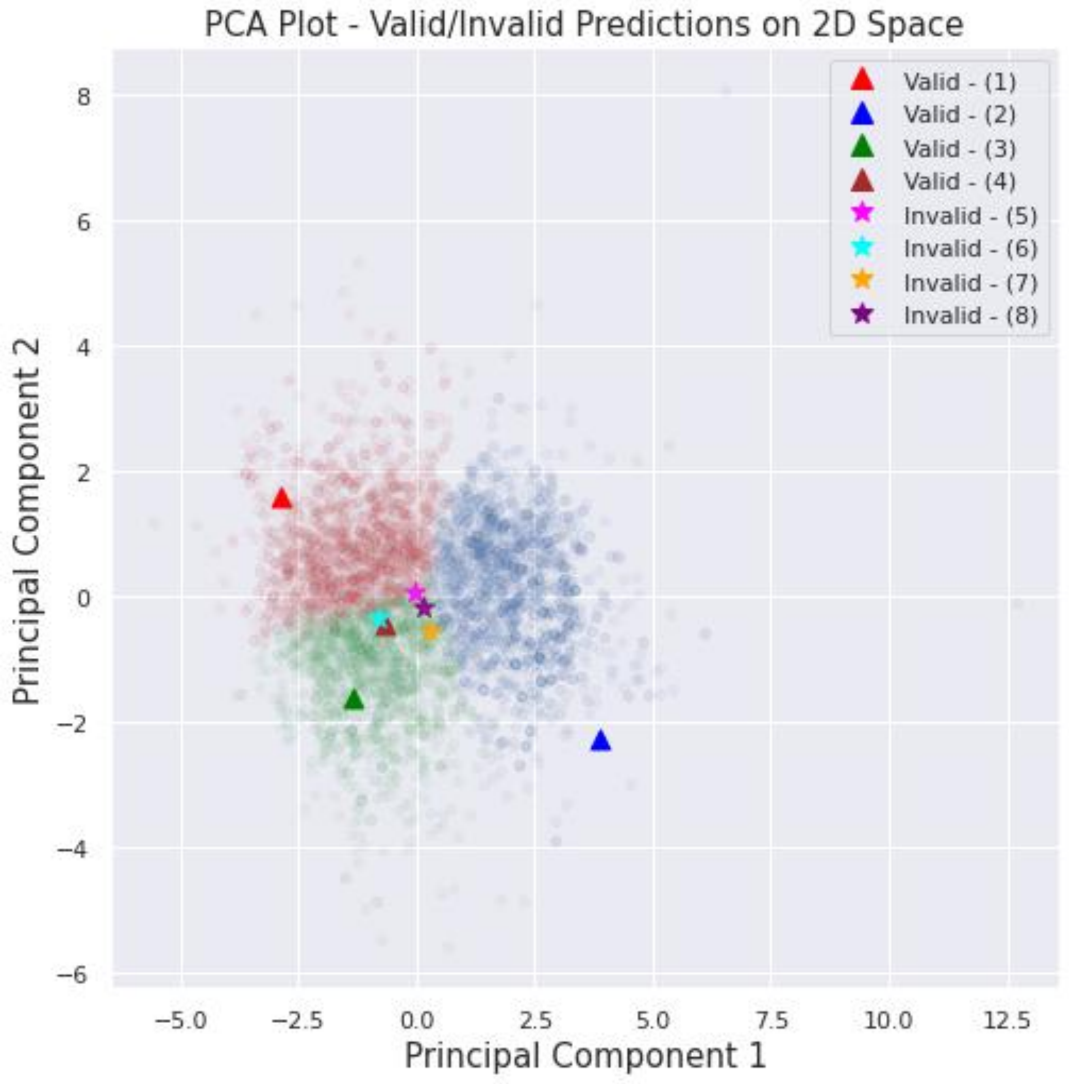


Fig. 9

Depicted as Triangles: Valid Predictions on the 2D feature space.

The corresponding histograms per number below can be viewed in Fig.7

(1) Class 0 in Blue
(2) Class 1 in Green
(3) Class 2 in Red
(4) Borderline case in brown

Depicted as Stars: Invalid Predictions on the 2D feature space.

The corresponding histograms per number below can be viewed in Fig.8

(5) Incorrect from both models - fuchsia
(6) Almost identical class 1 and 2 distributions - cyan
(7) Borderline case - orange
(8) Ambiguous case with all SoftMax medians close to 0.33 - purple

As it is evident from the graph (Fig.9) and previous diagnostics (Fig.6), the closer the observation lies on the borders between classes, the more difficult it is for it to get classified by the model. Cases of high uncertainty (medians of distributions relatively close to 0.33) like (5 - fuchsia) and (8 - purple) also appear to lie in the center (0,0) of the 2-PCA plot.

We should of course not forget that defining a prior on the weights captures the epistemic uncertainty within the model. Should more data be available, the 2D borders between classes would possibly become more distinguished, the approximated weight posteriors would approach their true ones adequately and thus aid with providing more accurate predictions. due to minimizing this type of uncertainty, the predictive standard error, would thus decrease and predictive intervals would get narrower, improving the accuracy of the model. As such, invalid cases that are defined by ambiguous SoftMax distributions could possibly turn to valid ones.

4.1.3 Capturing Aleatoric Uncertainty with a p-BNN

In order to also model aleatoric uncertainty, we can place a distribution on the output. Therefore, now the model produces a distribution instead of a single SoftMax vector. This an extremely useful scenario for real-time applications as there is no need to perform an MC experiment and the means and standard deviations of these predictive distributions can be accessed immediately (Kendall & Gal, 2017).

Below are given the predictive intervals per valid observation, as examined in Fig. 7 and their corresponding p-BNN ones (seed set as 0 on training and prediction phases). For the p-BNN, intervals were bounded on 0 and 1 for their lower and upper end respectively.

Table 1 - Valid Observations and their Predictive Interval Bounds per class and model

Observation	**True Value**	**Class**	**BNN Lower**	**BNN Upper**	**p-BNN Lower**	**p-BNN Upper**
(1) 2634	0	Class 0	**0.955**	**1.000**	**0.979**	**1.000**
		Class 1	0.000	0.040	0.000	0.018
		Class 2	0.000	0.007	0.000	0.010
(2) 1924	1	Class 0	0.007	0.021	0.000	0.073
		Class 1	**0.825**	**0.930**	**0.817**	**1.000**
		Class 2	0.059	0.158	0.000	0.167
(3) 3659	2	Class 0	0.003	0.017	0.000	0.017
		Class 1	0.014	0.089	0.000	0.086
		Class 2	**0.897**	**0.980**	**0.912**	**1.000**
(4) 4352	1	Class 0	0.056	0.094	0.000	0.281
		Class 1	**0.533**	**0.669**	**0.221**	**1.000**
		Class 2	0.261	0.387	**0.000**	**0.71**

*Bold rows indicate prediction of model.

What is worth mentioning about the results is that, although the p-BNN was trained on the whole dataset, the predictive intervals on the test data do not appear to be extremely narrow. On the contrary, it appears that the p-BNN model is more sensitive to quantify the uncertainty between classes. Overall, the bounds between models in the true valid cases (1 -3) do not appear to differ much. Notable however is the valid ambiguous case (4), where the upper and lower bounds of the p-BNN have an extreme difference, especially when compared to the BNN. In order to assess this intuitive claim, we can, in a similar manner, examine the invalid classifications and their corresponding intervals.

Table 2 - Invalid Observations and their Predictive Interval Bounds per class and model

Observation	True Value	Class	BNN Lower	BNN Upper CI	p-BNN Lower	p-BNN Upper
(5) 79	1	Class 0	**0.312**	**0.523**	0.000	1.000
		Class 1	0.192	0.350	0.000	1.000
		Class 2	0.229	0.395	0.000	1.000
(6) 4230	1	Class 0	0.059	0.135	0.000	0.249
		Class 1	**0.335**	**0.568**	0.000	1.000
		Class 2	0.337	0.565	0.000	1.000
(7) 371	1	Class 0	0.223	0.372	0.000	1.000
		Class 1	**0.401**	**0.559**	0.000	1.000
		Class 2	0.173	0.272	0.000	0.566
(8) 3999	2	Class 0	0.214	0.461	0.000	1.000
		Class 1	0.172	0.386	0.000	0.674
		Class 2	**0.253**	**0.514**	0.000	1.000

As shown in Table 2, the p-BNN seems to produce overly wide predictive intervals for invalid ambiguous cases. Especially in data points where the visualized MC SoftMax probabilities overlap (Fig. 8 – (5), (6), (8)), the predictive probability interval becomes [0,1]. It is interesting to note that in case (6), class 0 got assigned a smaller upper bound, narrowing the interval, possibly due to the fact that the distribution of this class is clearly separable from the other two.

Possibly, these results could occur due to the small sample size of the wine dataset. Should more training data were enough to further reduce epistemic uncertainty, the above predictive intervals on uncertain for the model areas would expect to get narrower. This claim however cannot be evaluated in the current setting.

4.2 Regression Task (Short Summary)

In the same manner, it would be interesting to investigate the relationship of the wine quality when it comes to its features. Again, a MC (500 iterations) experiment on predictions based on a BNN, modified for a regression task, was implemented and predictive samples were taken. Simply by assessing the standard deviation of these samples, we can identify areas where these predictions have been given high or low uncertainty.

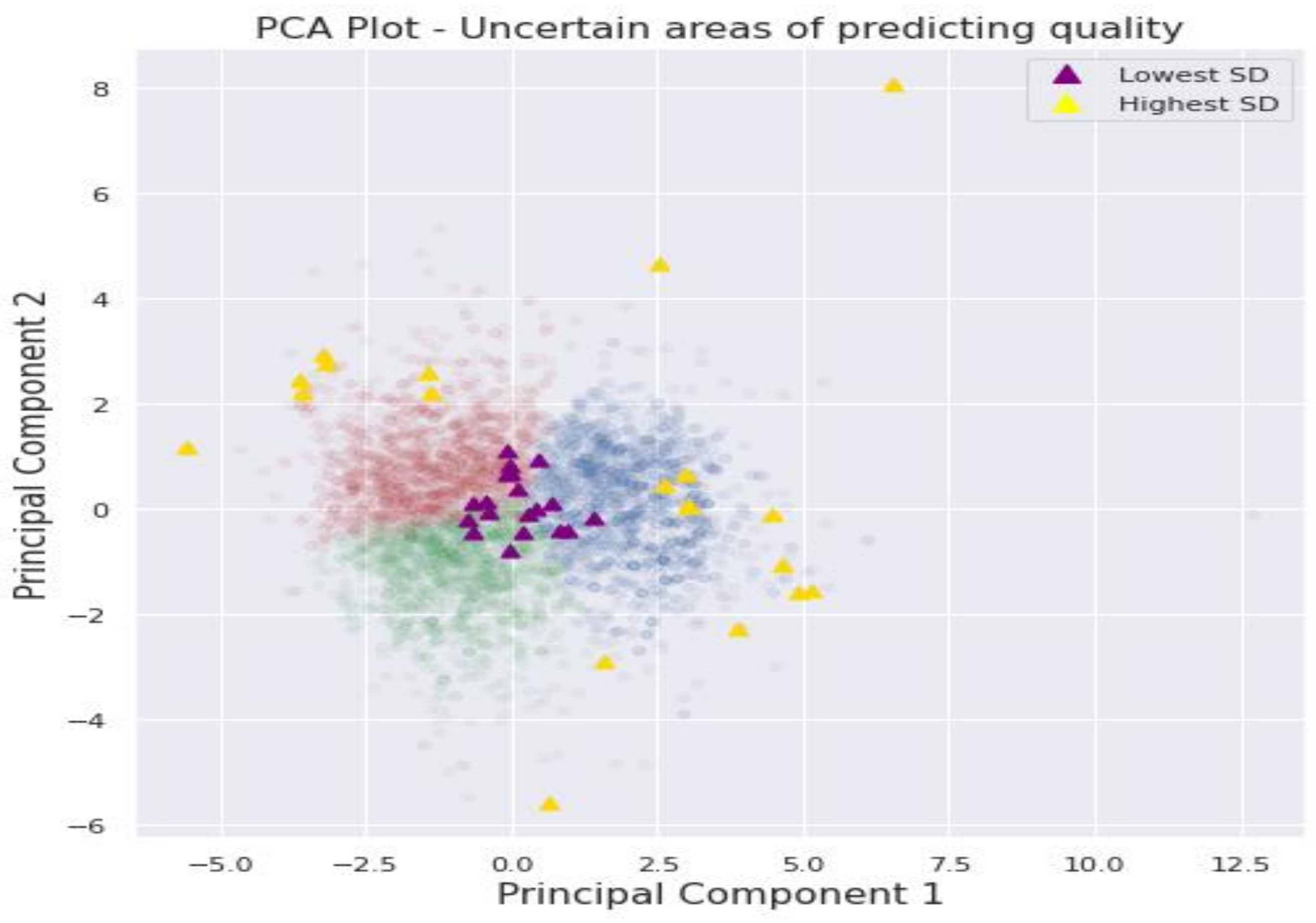


Fig. 10 Observations with the highest/lowest measure of uncertainty (20 represented per category).

It appears that, contrary to the classification task, the model is more confident to predict the qualities for observations clustered to the center of the PCA plot. Unfortunately, possibly to the unbalanced data when it comes to quality (Fig.1), the means of the said predictions were not solid for all models and the true value was not adequately approached. It would be thus meaningless in this report to compare predictive intervals between them. The results for the top 5 observations with the lowest/highest recorded standard deviation will be however given in the following section, so that the reader can have a notion of how the BNN and p-BNN operated in areas of uncertainty.

Table 3 – Predictive intervals for top 5 observations with lowest recorded standard deviation

Observations	**True Value**	**BNN Lower**	**BNN Upper:**	**p-BNN Lower**	**p-BNN Upper**
4009	6	3.110	3.564	0.000	4.499
4590	6	2.737	3.217	0.000	4.773
1162	7	2.981	3.469	0.000	5.252
3501	6	3.059	3.555	0.000	4.486
3494	6	3.063	3.562	0.000	4.644

Table 4 – Predictive intervals for top 5 observations with highest recorded standard deviation

Observations	**True Value**	**BNN Lower**	**BNN Upper**	**p-BNN Lower**	**p-BNN Upper**
4745	3	10.000	10.000	0.000	10.000
1526	6	10.000	10.000	0.000	10.000
3710	5	9.308	10.000	0.298	10.000
1835	5	6.366	8.498	0.000	10.000
683	5	5.918	7.990	0.000	10.000

Obviously, as said previously, it would not be wise to judge the models based on their prediction accuracy. What we will focus on is that predictive intervals for the BNN-Low-SD case (Table 3) have a solid lower and upper end. When aleatoric uncertainty is added to the model, these intervals get wider.

Especially in uncertain areas of the feature space (Table 4), the BNN model fails to approach the true value, despite the relatively narrow predictive intervals it provides. On the other hand, the p-BNN seems to behave similarly to the classification task and is not able to provide a constrained prediction at all as its lower and upper bounds exceed 0 and 10 (values were manually constrained for interpretability).

Perhaps feature engineering or reconstructing the output would remedy these cases. What needs to be said however is that, for the sake of evaluating the models further, we would need to further try and reduce epistemic uncertainty (relationship between features and quality would get more explainable) and reexamine anomalous areas of the feature space under both models.

5. Conclusions

Throughout this analysis, we have established the limitations of a frequentist NN and the need of a BNN to capture the model's confidence on its outputs. Although, by adding uncertainty to the weight parameters, we sacrificed a percentage of accuracy, our predictions were given confidence intervals and anomalous areas could be visualized. Even observations that were, initially, confidently predicted by a baseline NN (Fig.4), were later proven to lie in certain areas within the feature space, where much uncertainty regarding their type was generated.

It goes without saying that this analysis comes with limitations as well.

First of all, the simplistic method of defining a threshold on the SoftMax output distribution could be replaced with a more sophisticated method, as it serves as a baseline for classification problems (Hendrycks, 2017).

Secondly, the analysis has shown that out of distribution data points (OOD) or, otherwise, observations with extreme features that have no meaning within our dataset or are physically impossible (ex. alcohol percentage > 100%) could not be evaluated by the current models. Even though it has been shown that modelling aleatoric and, especially, epistemic uncertainty remedies this case (Kendall & Gal, 2017), sufficient data, to adequately minimize the latter, is still a requirement to identify OOD points. In a relatively small dataset like the one we analyzed, observations of this type got always classified as class 0 wines, regardless of their true value. More training data would thus be needed to recheck the p-BNN model's performance on areas of high uncertainty and testify this claim.

Thirdly, the analysis was prone to seeds for invalid classification cases, producing different results in ambiguous areas of the feature space. Also, as shown, regression outputs proved to be non-interpretable. Different model architectures could be explored to alleviate this effect.

Especially on occasions where the uncertainty is quite high (Fig.8 - 5,8), it would be useful to calibrate the model in order to stop the prediction process midway. Research has already been implemented on performing this technique (Karmakar & Pal, 2018). By managing that scenario, whichever decision relies upon the unconfident model's final estimation would not be affected and potential accidents due to misclassifications could be avoided.

Finally, we should not forget that although the methods of this analysis have tried to model these two types of uncertainty, the two remain correlated with each other. Recent research into this field indicates that incorporating latent variables into BNNs can separate end extract them adequately (Depeweg et. al, 2018). It would thus be interesting to investigate the improvements such a model would make in the above scenario.

6. References:

1) Cortez P., Cerdeira A., Almeida F., Matos T. and Reis J. (2009) Modeling wine preferences by data mining from physicochemical properties. In Decision Support Systems, Elsevier, 47(4):547-553
2) Depeweg S., Hernandez-Lobato M.J., Velez-Doshi F., Udluft S., (2018), "Decomposition of Uncertainty in Bayesian Deep Learning for Efficient and Risk-Sensitive Learning", URL: https://arxiv.org/abs/1710.07283
3) Gal. Y, (2016), "Uncertainty in Deep Learning", chapter 1.3, PhD Thesis, University of Cambridge,
4) Geurts, P., Ernst, D. & Wehenkel, L., (2006), "Extremely randomized trees". Mach Learn 63, 3–42 URL: https://doi.org/10.1007/s10994-006-6226-17
5) Harris M., (2019), "Article: NTSB Investigation Into Deadly Uber Self-Driving Car Crash Reveals Lax Attitude Toward Safety", IEE Spectrum https://spectrum.ieee.org/cars-that-think/transportation/self-driving/ntsb-investigation-into-deadly-uber-selfdriving-car-crash-reveals-lax-attitude-toward-safety
6) Hendrycks D., Gimpel K., (2017), A baseline for detecting misclassified and out-of-distribution examples in Neural Networks, ICLR 2017, URL: https://arxiv.org/abs/1610.02136
7) Jollife, Cadima, (2016), "Principal component analysis: a review and recent developments", Phil. Trans. R. Soc. A 374: 20150202, URL: https://doi.org/10.1098/rsta.2015.0202
8) Kamarkar B., Pal N., (2018) "How to make a neural network say "Don't Know"", Information Sciences, Volumes 430–431, March 2018, Pages 444-466, URL: https://doi.org/10.1016/j.ins.2017.11.061
9) Kemp R., Macaulay C., Palcic B., (1997), "Opening the black box: The relationship between neural networks and linear discriminant factors, URL: https://www.hindawi.com/journals/acp/1997/646081/
10) Kendall A., Gal Y., (2017), "What Uncertainties Do We Need in Bayesian Deep Learning for Computer Vision?", NIPS 2017, URL: https://arxiv.org/abs/1703.04977
11) Kingma D., Ba J., (2014), "Adam: A Method for Stochastic Optimization", ICLR 2015, URL: https://arxiv.org/abs/1412.6980
12) Kiureghian A,, Ditlevsen O., (2009), "Aleatory or Epistemic? Does it matter?", Structural Safety 31 (2009) 105–112, URL: https://doi.org/10.1016/j.strusafe.2008.06.020
13) Lingxue Z., Laptev N., (2017), "Deep and Confident Prediction for Time Series at Uber", URL: https://arxiv.org/abs/1709.01907
14) Wen Y, Vicol P., Ba J., Tran D., Grosse R., (2018), "Flipout: Efficient Pseudo-Independent Weight Perturbations on Mini-Batches", ICLR 2018, URL: arXiv:1803.04386

APPENDIX A

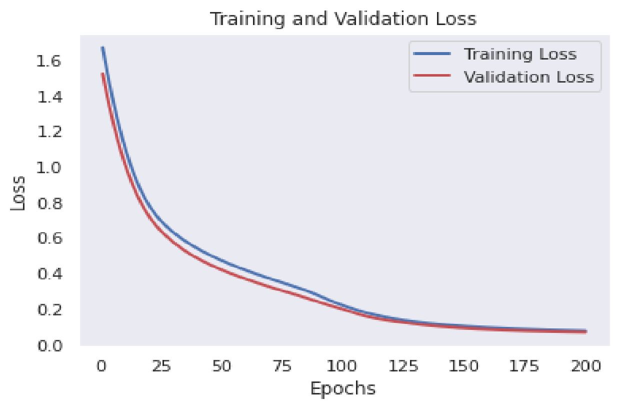
Info will be given about code parts as well as neural networks loss convergence/diagnostics, to ensure that the examined models operated adequately during the analysis. Plots are reproducible since a fixed random seed equal to 0 was used.

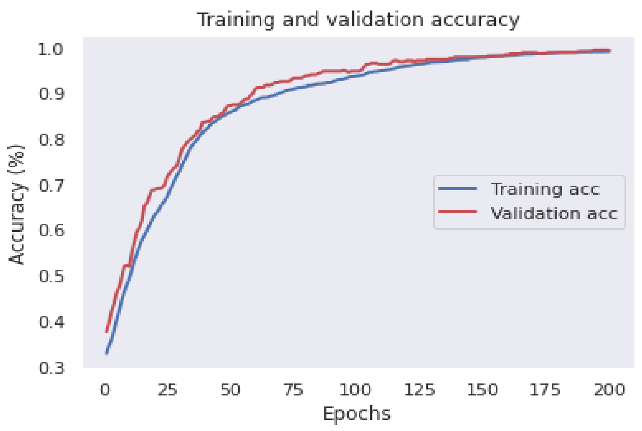
1. R Code for PCA plot in Fig.3.

Although the examined PCA plot is available in the Python code, automated PCA plot that included loadings was eventually used, with the aid of ggplot and ggfortify packages in R. The code for its creation is provided below, along with the picture of a correct output.

```
library(ggplot2)  # visualisation package
library(ggfortify) # visualise PCA with loadings

wine = read.csv("../input/samples/wine.csv")

# principal components – scale = TRUE indicates correlation matrix usage
wine.pca <- prcomp(wine[,c(1:(ncol(wine)-2))], center = TRUE,scale. = TRUE)

# plot PCA scatterplot
autoplot(wine.pca, data = wine, colour = "type", loadings = TRUE,
      loadings.label = TRUE, loadings.label.size = 3, title = "2-PCA Plot")
```

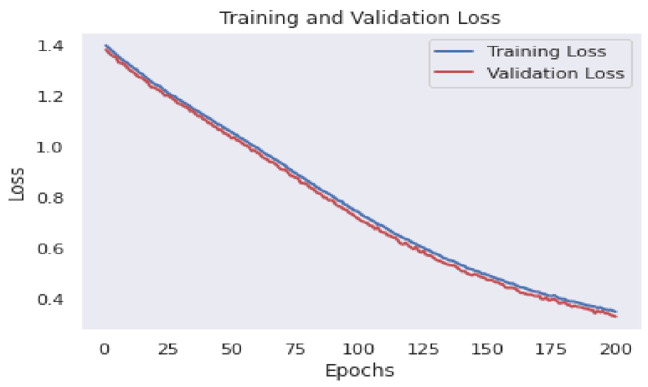
2. Convergence Diagnostics for NN (Classification)

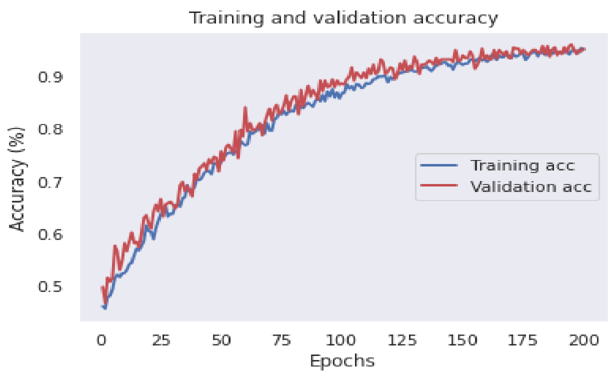




Minimization function: Categorical Cross-Entropy
Accuracy on training data: 0.992%
 Error on training data: 0.077
Accuracy on test data: 0.985%
 Error on test data: 0.081

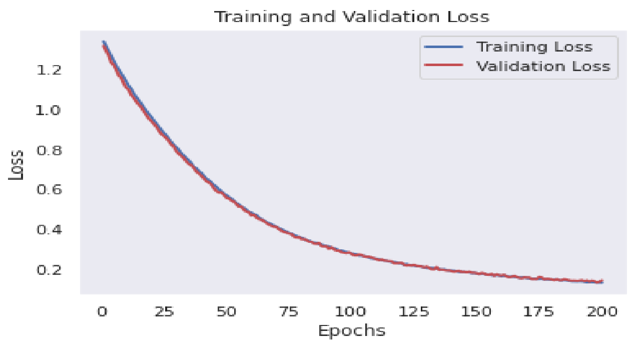
3. Convergence Diagnostics for BNN (Classification)

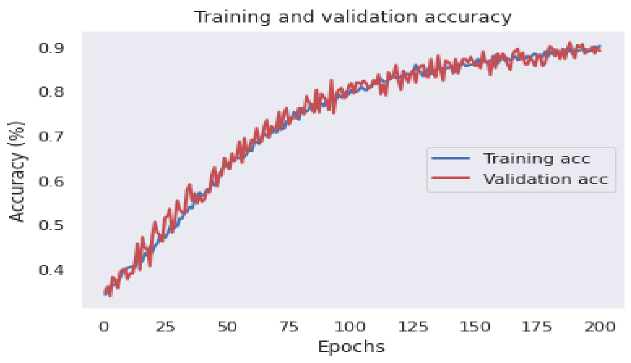




Minimization function: Categorical Cross-Entropy
Accuracy on training data: 0.955%
Error on training data: 0.348
Accuracy on test data: 0.951%
Error on test data: 0.346

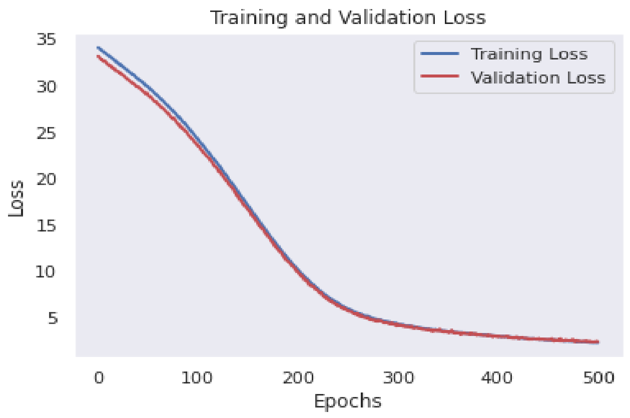
4. Convergence diagnostics for p-BNN (Classification)





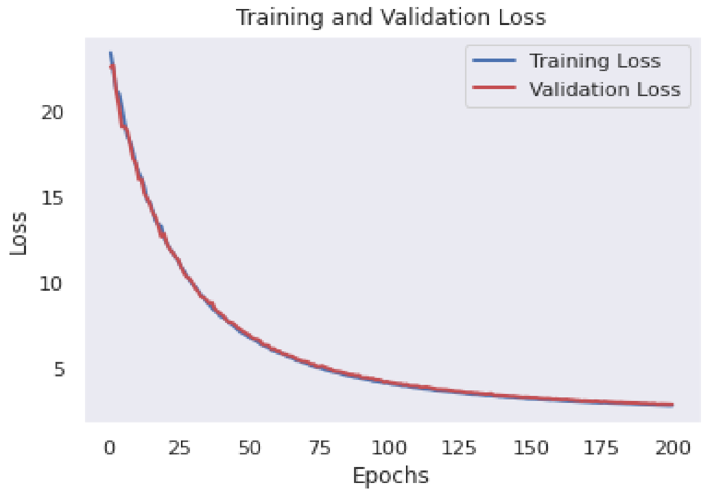
Minimization function: Negative Log Likelihood
Accuracy on training data: 0.955%
Error on training data: 0.348
Accuracy on test data: 0.906%
Error on test data: 0.135

5. Convergence Diagnostics for BNN (Regression)



Minimization function: Mean Square Error (MSE)
RMSE on training data: 1.542
Error on training data: 2.408
RMSE on test data: 1.62
Error on test data: 2.654

6. Convergence Diagnostics for p-BNN (Regression)


Training and Validation Loss
Training Loss
Validation Loss
Loss
Epochs