Assessing the quality of white wine by analysing the physiochemical components using Multilayer Perceptron's and Support Vector Machines

Mussa Yousef
INM427 Neural Network
City, University of London
mussa.yousef.1@city.ac.uk

Introduction

Our dataset was collected from the UCI machine learning repository. Titled "Wine Quality Dataset" [1] which provided 11 physicochemical of wine testing results as features that were used to predict our target variable 'Quality'. Our project was a classification experiment where the quality of white wine was rated [1] as good and [0] as bad. We performed our analysis using python's open-source software - Jupyter notebook [2], with the assistance of modules ***[Table1].*** We introduce grid search approaches to iterate through our models and find both models can perform reasonably well. The data itself presents some noise however understanding this through our pre-processing was key to configuring when building our models which controlled this well. Our motivation was to introduce Neural networks to a field where an expert is required to give a quality rating. Hence the outcome of this experiment is of need to vineyard experts who would be able to diminish their dependence on experts classing the quality of their wines.

Pandas	Preprocessing	Extract information from tabluar form
Seaborn	Preprocessing	Process elegant visuals
Matplotlib	Preprocessing	Process visuals and inferntial statistical
Imblearn	Preprocessing	SMOTE - used for oversampling
Scipy	Preprocessing	Pdist & Cdist used for Principal component analysis
Keras	Preprocessing	ONLY Used to convert our predicted varible to categorical
Sklearn	Support Vector Machines	Building and hyperparameter anaysis of SVM's
Torch	Multi-Layer Perceptron	Deep learning networks used to bulid Neural network MLP model
Time	All	Used to calculate processing time throughout our stages

Table1: Modules used for this expeiment

Hypothesis statements

We have three hypothesis statements, our initial and fundamental hypothesis and two synchronized statements which are more specific. **H0** – MLP should generalize better than SVM **H1** – Oversampling our minority class should increase performance for both MLP and SVM **H2** – Considering space and time complexity MLP should be slower to train compared to our SVM model.

Data Wrangling and Exploratory analysis

The Wine Quality dataset contains 11 attributes which are all continuous, attributes are of the physicochemical test results of different white wines with one output variable which was the quality rating. The quality rating of the dataset was of a spectrum between 3-9(figure 1). To be able to categorise [3] and predict the quality for our experiment, we binned the quality variable where 78.3% labelled with **0** representing bad quality and 21.67% labelled with **1** representing good quality. Therefore, the entire dataset included 4898 samples with an imbalance of 1:4 for good quality rating. Hence to deal with the imbalanced nature of the dataset, we used SMOTE [4] to oversample our training features and will compare the performance of both oversampled features and unchanged features on both models.

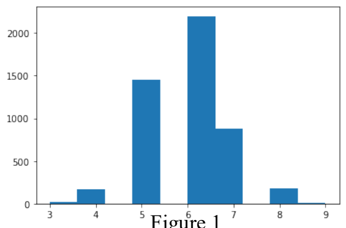


Figure 1

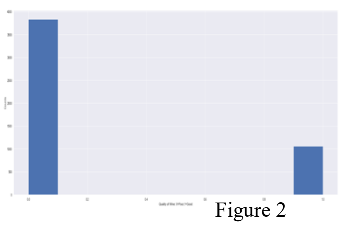
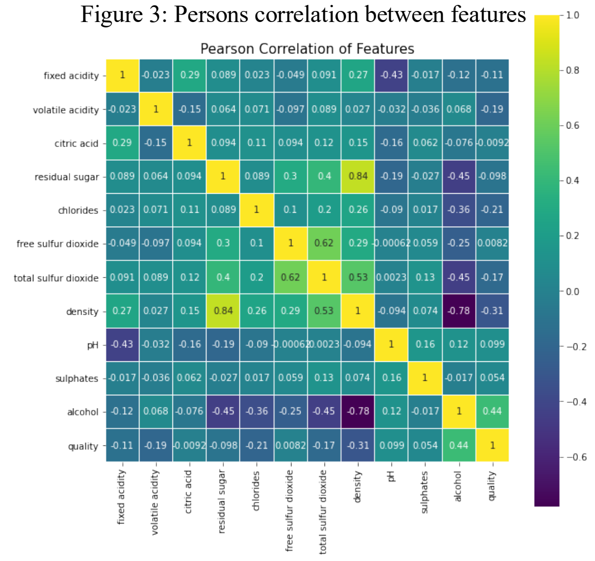
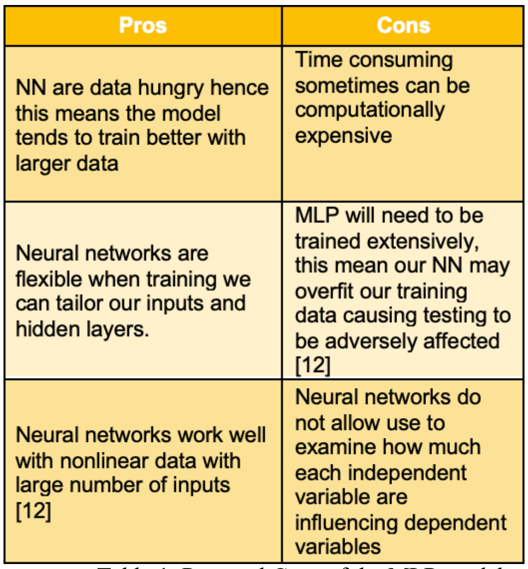


Figure 2

Figure 3: Persons correlation between features



Examining the Pearson's correlation [5] (r) measure which measures the linear association between two variables. Scrutinising the labels column, we find some fairly strong correlations with quality. Furthermore, the variance between we features are consistent however the Free Sulphur and total Sulphur dioxide features share extreme outlier variance. Although this wouldn't affect our Neural network as it's a universal approximator, we will utilise Principal Component Analysis (PCA) [6] to test on our SVM to examine how visually our decision matrix would decide on boundaries using a linear Kernel.



Pros	Cons
NN are data hungry hence this means the model tends to train better with larger data	Time consuming sometimes can be computationally expensive
Neural networks are flexible when training we can tailor our inputs and hidden layers.	MLP will need to be trained extensively, this mean our NN may overfit our training data causing testing to be adversely affected [12]
Neural networks work well with nonlinear data with large number of inputs [12]	Neural networks do not allow use to examine how much each independent variable are influencing dependent variables

Table 1: Pros and Cons of the MLP model

Description of Models

Neural networks (NN) are derived from biological neural networks, built like the human brain with interconnected nodes which distributes work and decisions parallel to each other [7]. Therefore, it resembles the brain in two aspects;1) Knowledge is acquired by the network through a learning process 2) Interneuron connection strengths known as synaptic weights are used to store knowledge [8]. The motivation here to develop NN is to examine how one of the most sophisticated algorithms used in deep learning could work on tabular data. Recent studies from Abutbul, Ami et al [9] suggests a novel approach is needed when applying neural networks to tabular data. Therefore, keeping this in mind we will go against this report and apply our learning from of an MLP to our data and examine the performance. The MLP consists of three fundamental layers. The input layer is essentially where our data is initiated into our model, hidden layer where the calculations are made to produce a classification predication in the output layer is where our model makes a classification prediction.

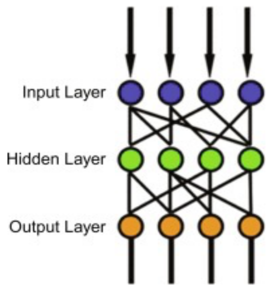
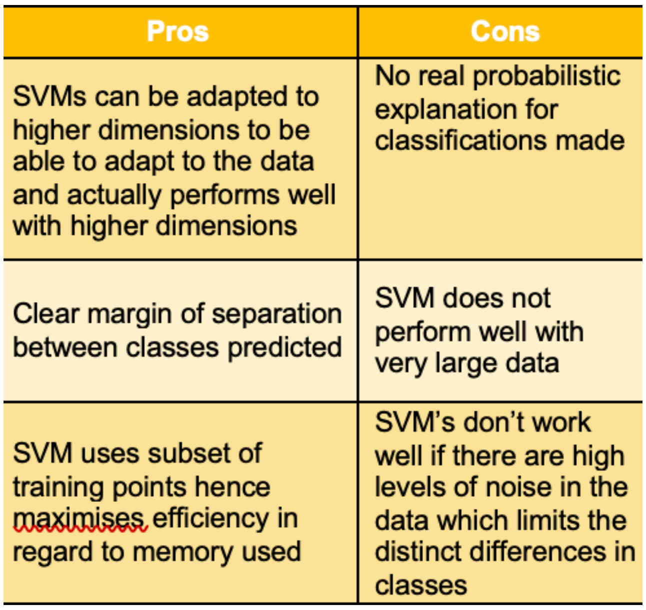


Figure 4: Example of a simple MLP [10]

SVM is a supervised learning algorithm which provides a nonlinear general architecture that can be applied to our classification problem. More simplistic compared to our MLP the SVM find the best hyperplane which would separate our features to provide comparison on our classified prediction [11]. One hurdle however when using SVM is the dimensionality of the given data, SVM's perform exceptionally well with 2-dimensional feature spaces. However, using SVM to solve a real-world problem, we are typically faced with several dimensions which influence our predictions. SVM can be adapted utilizing Lagrange Multiplier therefore provide Kernels which we will explore to adapt our model to a high dimensional space.



Pros	Cons
SVMs can be adapted to higher dimensions to be able to adapt to the data and actually performs well with higher dimensions	No real probabilistic explanation for classifications made
Clear margin of separation between classes predicted	SVM does not perform well with very large data
SVM uses subset of training points hence maximises efficiency in regard to memory used	SVM's don't work well if there are high levels of noise in the data which limits the distinct differences in classes

Table 2: Pros and Cons of the SVM model

Methodology

MLP	SVM
Activation function	Kernels
Loss function	PCA - Dimensionality reduction
Dropout	C Penalty
Hidden layer dimension	Gamma
Learning rate	
Optimisers	
Batch-size	

Table 4: Hyperparameters scrutinised in this experiment to to build the best model possible

We initiated our methodology by holding out 20% of our data for testing our SVM and MLP models. We used the remaining 80% of our original data in training our data, this allowed systematic comparison to be made. The process in choosing our final models was utilising grid search to tune the best hyperparameters [Table 4] which will provide the best generalised model. We used k-fold cross validation when tuning our SVM model, splitting our training data into 5 while varying our hyperparameters. However, as previously highlighted the MLP model is a data hungry algorithm, therefore utilising cross validation would hinder our models training. Therefore, we focused on tuning our batch size, to maximise resource utilisation and generalisability. Inspired by the initial perceptron model from Minsky-papert and later further developed by McCullogh and Pitts the fundamental element of a neural network is the perceptron itself. Here we navigate an ensemble, using backpropagation to update each weight through each iteration. We deployed an Artificial Neural network, initiating a backpropagation, calculating and updating each weight backwards using derivatives based on the error to calculate gradient descent.

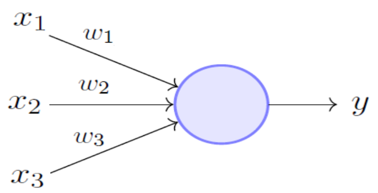


Figure 5:Perceptiron Model diagram (Minsky-papert

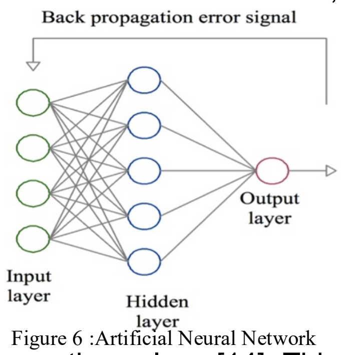


Figure 6 :Artificial Neural Network

Activation function - Essentially Activation functions are the key ingredient which makes a neural network non-linear. Deciding whether a perceptron should fire or not, makes an activation function an important element in the backpropagation. Hence in this experiment we looked at how each activation affected our model. ***Rectified linear unit**: -* Switched on for positive values and zero for negative values [14]. This allows converge to happen faster hence no plateau or saturation when our inputs into the nodes are very large. ***Tanh (Hyperbolic tangent)**: -* From -1 to 1 as intervals crossing 0, the hyperbolic function acts somewhat like the sigmoid function. In this experiment we tried using the Tanh function with a dropout however as we will see with the results this affected our loss function too much. ***ELU (Leaky ReLU)**: -* A variation of the ReLU. This Activation provides a small positive slope reaching 0 at 0 then proceeds to y=x for positive numbers. This enables backpropagation and also calculates negative values. ***Softmax**: -* Calculated probabilities range from 0 to 1.[14] This is the only activation function we kept consistent in this experiments grid search. This is essentially the last activation function after derivatives which calculates the probability distribution.

Loss function *Cross Entropy: -* Measures the predicted probability distribution in comparison to true distribution. We use this Loss function throughout our experiment keeping it unchanged.

Dropout - Essentially drops out neurons from network, negating their use which causes the information to voluntary travel through another passage. This changes through each epoch, hence limiting the potential of overfitting. We apply this to our grid search between 0.1-0.9 dropout to examine our model’s performance, we find our accuracy completely stagnate which we will discuss in our results.

Hidden layer dimension - We grid search through a dimension of hidden layers. Essentially the dimension of the hidden layer can play an integral role in optimising our model. As we are using backpropagation calculation of each node’s gradient will affectively minimise loss and allow our activation function to maximise affect.

Learning rate - One of the most important hyperparameters, the learning rate essentially tunes the set taken towards the optimal gradient. As we tune this hyperparameter we aim to gradually lower our loss as our epochs increase

Optimizers -*Stochastic gradient decent (SDG):* For each epoch trained the gradient is calculated and frequently updated on each node hence minimising the computational loss [14]. *Our experience using this optimisation showed that results sometimes shot high even after a local minimum was being reached.*

Adaptive moment estimation (Adam): - Uses an exponential decay of average calculated gradients, therefore narrowing into a local optimal final gradient. Changing our optimiser to Adam in this experiment showed improvements in our validation accuracy and lowered our loss. We incorporated the AdamW which initiates a weight decay. We tune our weight decay which we find furthers our performance.

Support vector machines aims to find the optimal hyperplane which maximises the margin between two labels. [11] Target variables which can be separated via linear plane provides an easy solution. However here as we have several dimensions to consider, we utilise the kernel to transform the dimension. In this experiment we utilise cross validation grid search to find the best parameters. Further analysis of dimension analysis using feature reduction to be able to visualise our SVM on a linear plane.

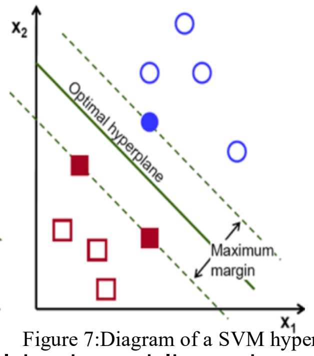


Figure 7:Diagram of a SVM hyperplane

Kernels - The Kernel function returns the inner product between two points in a suitable feature, hence minimise computation cost regardless of how many dimensions the data has [11]. In this experiment we grid searched between polynomial, gaussian radial basis and linear kernels. Assessing the ability to find optimal margins

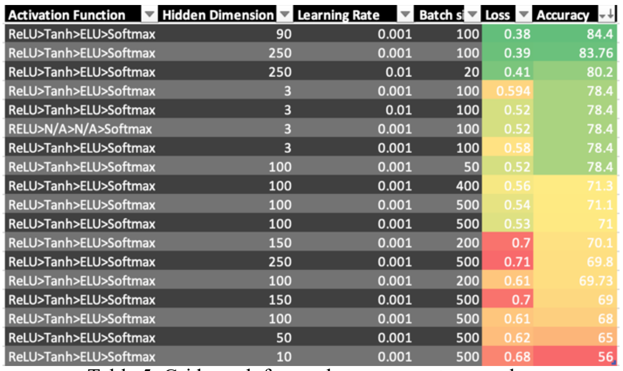
Principal component analysis (PCA) - Although not a hyperparameter we utilise principal component analysis to compact our features dimensions from 11 to 2. This allowed us to visually examine our SVM performance on a linear kernel. We also find our hyperparameters grid search provide the same optimal parameters as using all our features without PCA.

C Penalty - Our 'C' Parameter determines our penalty for the misclassification of our training. This navigates the margin on our hyperplane. For this experiment we use logspace parameters to grid search i.e., 0.001,0.01,0.1,1,10

Gamma - Gamma sets the level of curvature in our margin. Therefore, the Gamma is utilised when the RBF kernel is set, when utilising the Linear and Polynomial kernels we only set our 'C' hyperparameter [11]. For this experiment we set our hyperparameter grid search using logspace i.e., 0.001,0.01,0.1,1,10

Results, Findings & Evaluation

Table 5 shows our MLP grid search for our best models. We iterated over sever different parameters focusing on minimising our loss function given this would enhance generalisability when testing. We found that generally the Learning rate was sensitive and when increased above 0.001 the loss would shoot. However, coupled with a high hidden number of neuron (>200) and a lower batch size (<30) we found our accuracy increased. This suggests our algorithm was attempting to reach our minima. Parallel to this we changed our optimizer to add regularisation with Adam, attempting to see how weight decay can influence our model. We found this lowered our loss further and visually brought a gradual rise in our accuracy.



Activation Function	Hidden Dimension	Learning Rate	Batch si	Loss	Accuracy
ReLU>Tanh>ELU>Softmax	90	0.001	100	0.38	84.4
ReLU>Tanh>ELU>Softmax	250	0.001	100	0.39	83.76
ReLU>Tanh>ELU>Softmax	250	0.01	20	0.41	80.2
ReLU>Tanh>ELU>Softmax	3	0.001	100	0.594	78.4
ReLU>Tanh>ELU>Softmax	3	0.01	100	0.52	78.4
RELU>N/A>N/A>Softmax	3	0.001	100	0.52	78.4
ReLU>Tanh>ELU>Softmax	3	0.001	100	0.58	78.4
ReLU>Tanh>ELU>Softmax	100	0.001	50	0.52	78.4
ReLU>Tanh>ELU>Softmax	100	0.001	400	0.56	71.3
ReLU>Tanh>ELU>Softmax	100	0.001	500	0.54	71.1
ReLU>Tanh>ELU>Softmax	100	0.001	500	0.53	71
ReLU>Tanh>ELU>Softmax	150	0.001	200	0.7	70.1
ReLU>Tanh>ELU>Softmax	250	0.001	500	0.71	69.8
ReLU>Tanh>ELU>Softmax	100	0.001	200	0.61	69.73
ReLU>Tanh>ELU>Softmax	150	0.001	500	0.7	69
ReLU>Tanh>ELU>Softmax	100	0.001	500	0.61	68
ReLU>Tanh>ELU>Softmax	50	0.001	500	0.62	65
ReLU>Tanh>ELU>Softmax	10	0.001	500	0.68	56

Table 5: Gridsearch for our hyperparameter search

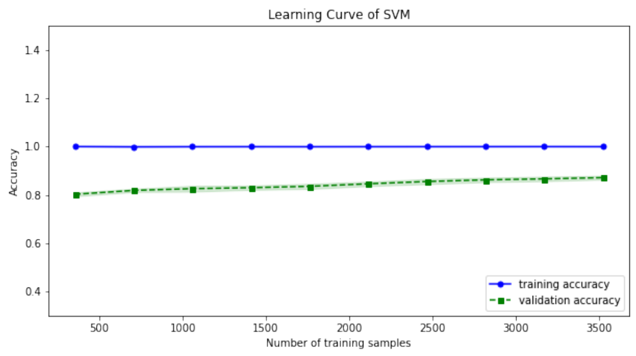


Figure 8 :SVM Learning Curve

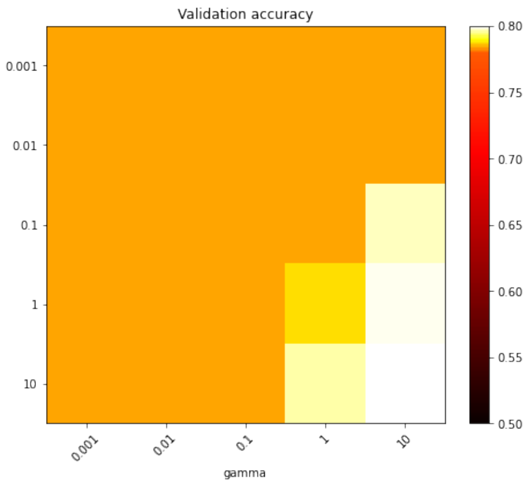


Figure 9: SVM Validation accuracy grid

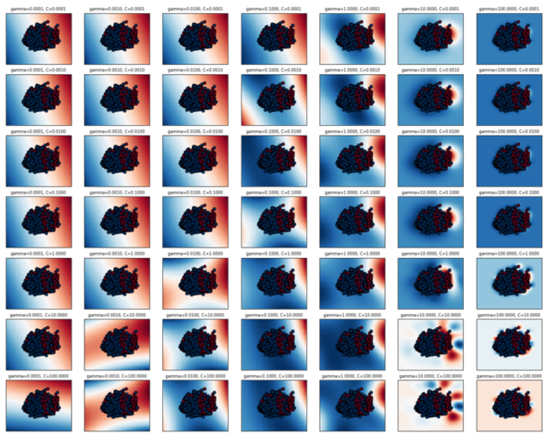
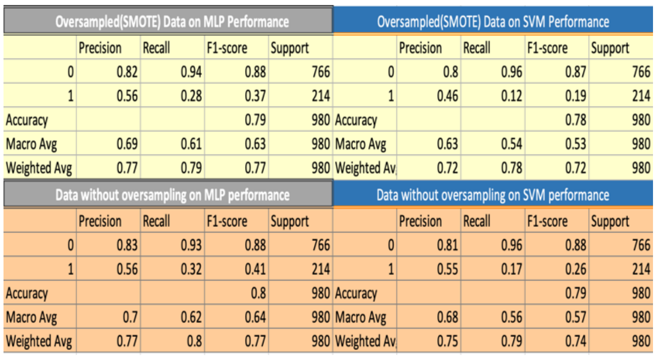


Figure 10: SVM Validation Grid search boundries using PCA

Our cross validation searching for the optimal kernel showed RBF to provide the best validation accuracy for our SVM model. Hence, we used RBF to examine our models training and validation performance (figure 8). We find our validation performance converges towards our training accuracy which suggest our model generalises well. From our SVM validation grid (figure 9) we find our optimal parameters is 10 and 10 for our C regularisation and Gamma respectively, resulting in 86% Validation Accuracy.

We compared our results using data with and without oversampling to examine the impact of using SMOTE on our data for both algorithms. Although our models provided reasonable results in the training process, further scrutiny and testing showed poor performance for our minority class. We find that for our MLP throughout the training process, accuracy continuously stagnated, which now during the testing that our F1 and recall measure showed that just around 30% of our minority class was being accurate predicted. Oversampling our data didn't improve our generalisability of our MLP, although our MLP slightly improved in AUC, our overall precision wasn't enhanced as we our Recall for our majority class being hindered when using our data which was not oversampled. Our SVM performance was also poor considering our minority class accumulates to 21% our recall for SVM only reached 17%, which means that from all the prediction made 36/214 were accurately classified as bad wine. SVM here was more robust with grid searching our parameters we still faced appalling minority class precision.



Oversampled(SMOTE) Data on MLP Performance					Oversampled(SMOTE) Data on SVM Performance				
	Precision	Recall	F1-score	Support		Precision	Recall	F1-score	Support
0	0.82	0.94	0.88	766	0	0.8	0.96	0.87	766
1	0.56	0.28	0.37	214	1	0.46	0.12	0.19	214
Accuracy			0.79	980	Accuracy			0.78	980
Macro Avg	0.69	0.61	0.63	980	Macro Avg	0.63	0.54	0.53	980
Weighted Avg	0.77	0.79	0.77	980	Weighted Av	0.72	0.78	0.72	980
Data without oversampling on MLP performance					**Data without oversampling on SVM performance**				
	Precision	Recall	F1-score	Support		Precision	Recall	F1-score	Support
0	0.83	0.93	0.88	766	0	0.81	0.96	0.88	766
1	0.56	0.32	0.41	214	1	0.55	0.17	0.26	214
Accuracy			0.8	980	Accuracy			0.79	980
Macro Avg	0.7	0.62	0.64	980	Macro Avg	0.68	0.56	0.57	980
Weighted Avg	0.77	0.8	0.77	980	Weighted Av	0.75	0.79	0.74	980

Figure 10: MLP results table for oversampled data and data with being oversampled

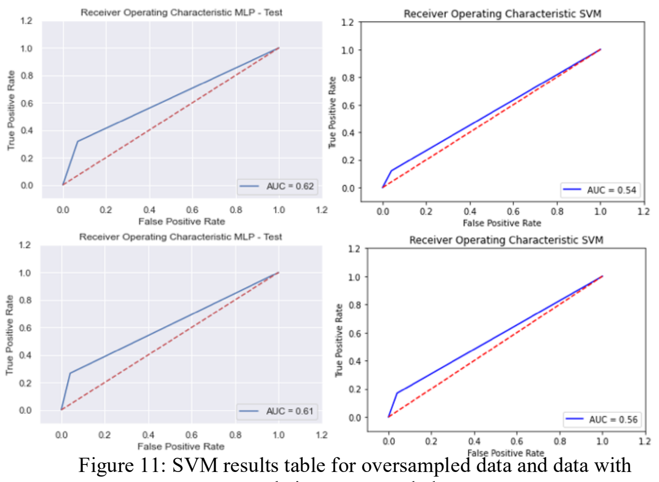


Figure 11: SVM results table for oversampled data and data with being oversampled

Conclusion and Further work

Although both performed poorly, from our testing results we accept H0 as the MLP model had provided not only more room in tuning our parameter but a better performance for our model. We reject H1, oversampling in this case had not improved model performance, in all model performance measures we saw an improvement when using data which was not oversampled. Lastly, we accept H2, MLP did take longer to train, provided we did consider more parameters and iterations when building our MLP, SVM had less flexibility when training although training theoretically still is slower to train.

There are several routes in which we could have conducted this experiment, given both models have been used for several sophisticated experiment, indicates potentially more scrutiny should have been given in the pre-processing state. In the future we would firstly divide the label into three targets allowing a normal distribution among our quality labels. Furthermore, our two features which shared high variance, we would omit both features from this experiment as from this experiment we find the high usage of regularisation of both MLP and SVM is as a result of high variance from our data.

[1] UCI ML Repository – "Center of Machine Learning and intelligent systems" https://archive.ics.uci.edu/ml/datasets/Wine

[2] Pythons open source software https://jupyter.org/

[3] Supervised discretization of continuous variables https://nextjournal.com/eda/discretize-cont-var

[4] Nitesh V. chawla, Kevin W. Bowyer, Lawrence O. Hall, W. Philip Kegelmeyer. "SMOTE: Synthetic minority over-sampling technique" https://dl.acm.org/doi/10.5555/1622407.1622416

[5] (2008) Pearson's Correlation Coefficient. In: Kirch W. (eds) Encyclopedia of Public Health. Springer, Dordrecht. https://doi.org/10.1007/978-1-4020-5614-7_2569

[6] Jolliffe I. (2011) Principal Component Analysis. In: Lovric M. (eds) International Encyclopedia of Statistical Science. Springer, Berlin, Heidelberg. https://doi.org/10.1007/978-3-642-04898-2_455

[7] Daniel G.G. (2013) Artificial Neural Network. In: Runehov A.L.C., Oviedo L. (eds) Encyclopedia of Sciences and Religions. Springer, Dordrecht. https://doi.org/10.1007/978-1-4020-8265-8_200980

[8] Ripley, B. (1996). Introduction and Examples. In *Pattern Recognition and Neural Networks* (pp. 1-16). Cambridge: Cambridge University Press. doi:10.1017/CBO9780511812651.002

[9] Abutbul, Ami et al. "DNF-Net: A Neural Architecture for Tabular Data." ***ArXiv*** abs/2006.06465 (2020): n. pag.

[10] S. Abirami, P.Chitra, in advances in computers, 2020 – Multilayer Perceptron – "Th Digital Twin Paradigm for Smarter Systems and Enviroments: The Industry Use Cases" https://www.sciencedirect.com/topics/computer-science/multilayer-perceptron

[11] Cristianini N., Ricci E. (2008) Support Vector Machines. In: Kao MY. (eds) Encyclopedia of Algorithms. Springer, Boston, MA. https://doi.org/10.1007/978-0-387-30162-4_415

[12] Akshay L Chandra –"Perceptron: The Artificial Neuron (An Essential Upgrade To the McCulloch-Pitts Neuron)- https://towardsdatascience.com/perceptron-the-artificial-neuron-4d8c70d5cc8d

[13] Daanqing Liu – "Rectified Linear Unit" - https://medium.com/@danqing/a-practical-guide-to-relu-b83ca804f1f7

Glossary

Terminology		Definition
1.	Classification	Machine Learning- simplifies real like decision by building respective models; a type of method which computers use to predict is called 'classification' where the model is able to predict using class labels. [2]
2.	Algorithm	A mathematical sequence which implements instructions to the computer which solves machine learning problems and is an integral part in prediction.3]
3.	Physicochemical	Relating to the wine dataset used in this project the physicochemical is the combination of both physical and chemical elements within the structure of red wine. [5]
4.	Hyperparameters	A hyper-parameter is a parameters value which has been inspected and externally chosen to best optimise a model. A standard parameter us usually defined via training. [6]
5.	Correlation	Correlation is the statistic the investigates the relationship between two variables and how the move in relation to one another.[7]
6.	Recall	Recall is conserved to be the proportion of how many actual positives was identified correctly [8].
7.	F-score	In statistical analysis F-score is the accuracy of the test data being compared to the training data.[10]
8.	Computational cost	This is the complexity cost of an algorithm is the consideration of the number of resources required to run the algorithm i.e., Time and Memory used. [13]
9.	Accuracy	Accuracy is the number of correctly predicted data points from all predicted variables.[14]
10.	Optimisation	Optimisation is essentially the process of making sure the Machine learning model uses the most efficient number of resources while performing at its best given the amount of information being processed. [15]
11.	Misclassification	Misclassification is the calculation of error which refers to the number of predictions which has been wrongly predicted as another label.
12.	Grid search	Grid-search is the process of looking through data and configuring optimal hyperparameters via indicators. [20]
13.	Loop search	The process of iterating a programming structure that repeats a sequence of instruction until a specific condition is met. [22]
14.	Confusion matrix	Summarises the performance of a classification algorithm by calculating the outcome of Actual and Predicted variables. [23]
15.	Cross validation	Cross-validation is the technique used to train the data set. Cross-validation is largely used in settings where the data is used to validate itself by splitting into folds and training on parts of the data while testing on another and vice versa. This process allows us to estimate the accuracy of the performance.[24]
16.	Precision	Precision is the proportion of positive identifications which was actually correctly predicted.[9]
17.	PCA (Principal component analysis)	Principal Component Analysis (PCA) is a statistical procedure that converts correlated and uncorrelated variables. PCA is common tool to explore data analysis. [26]

Reference sourced to definition

[2] https://machinelearningmastery.com/types-of-classification-in-machine-learning/

[3] https://dictionary.cambridge.org/dictionary/english/algorithm

[4] https://towardsdatascience.com/simple-guide-for-ensemble-learning-methods-d87cc68705a2

[5] https://www.collinsdictionary.com/dictionary/english/physicochemical

[6] https://en.wikipedia.org/wiki/Hyperparameter_(machine_learning)

[7] https://www.investopedia.com/terms/c/correlation.asp

[8] https://en.wikipedia.org/wiki/Precision_and_recall

[9] https://developers.google.com/machine-learning/crash-course/classification/precision-and-recall

[10] https://en.wikipedia.org/wiki/F-score

[13] https://en.wikipedia.org/wiki/Computational_complexity

[14] https://deepai.org/machine-learning-glossary-and-terms/accuracy-error-rate

[15] https://towardsdatascience.com/demystifying-optimizations-for-machine-learning-c6c6405d3eea

[16] https://www.researchgate.net/post/What-is-misclassification-error-and-what-algorithm-is-best-for-this

[20] https://elutins.medium.com/grid-searching-in-machine-learning-quick-explanation-and-python-implementation-550552200596

[21] https://zyxo.wordpress.com/2011/07/04/how-to-use-the-settings-to-control-the-size-of-decision-trees/

[22] https://techterms.com/definition/loop

[23] https://machinelearningmastery.com/confusion-matrix-machine-learning/

[24] https://www.techopedia.com/definition/32064/cross-validation

[26] https://www.geeksforgeeks.org/ml-principal-component-analysispca/

Implementation details

1) Load first three entries:

1- All modules needed for the first part of this experiment are in the first entry

   - Further loading of SVM modules needed are underneath the title “SVM Model”

Load dataframe in entry [3] which the CSV file will be included in the zip file

To test MLP Model:

We have saved the path as Model8 hence load entry called “Loaded_Model…” This will load the trained model which will also be included in the zip file.

Prior to testing our model run the entry underneath title “Prepare Data for Training” this will convert data to array split data and scale ready for Testing

To test SVM Model:

We have saved our SVM path as “FinalSVM” hence you would be able to load this model if you scroll to “Load model for testing” under the SVM sector

You can now test our model under the sector “Lets now Test our SVM” this will load our chosen parameters hence prepared for testing.