Wine Quality Classification

Classifying wine quality based on its properties

Worked By:Elisa Gjuraj

Dataset used:
https://www.kaggle.com/datasets/rajyellow46/wine-quality

11 features:
- fixed acidity, volatile acidity, citric acid, residual sugar, chlorides, free sulfur dioxide, total sulfur dioxide, density, pH, sulphates, alcohol

1 target: Wine quality (takes values 3-9)

4898 instances of white wine
1599 instances of red wine

The dataset:

	A	B	C	D	E	F	G	H	I	J	K	L	M
1	type	fixed acidity	volatile acidity	citric acid	residual sugar	chlorides	free sulfur dioxid	total sulfur dioxic	density	pH	sulphates	alcohol	quality
2	white	7	0.27	0.36	20.7	0.045	45	170	1.001	3	0.45	8.8	6
3	white	6.3	0.3	0.34	1.6	0.049	14	132	0.994	3.3	0.49	9.5	6
4	white	8.1	0.28	0.4	6.9	0.05	30	97	0.9951	3.26	0.44	10.1	6
5	white	7.2	0.23	0.32	8.5	0.058	47	186	0.9956	3.19	0.4	9.9	6
6	white	7.2	0.23	0.32	8.5	0.058	47	186	0.9956	3.19	0.4	9.9	6
7	white	8.1	0.28	0.4	6.9	0.05	30	97	0.9951	3.26	0.44	10.1	6
8	white	6.2	0.32	0.16	7	0.045	30	136	0.9949	3.18	0.47	9.6	6
9	white	7	0.27	0.36	20.7	0.045	45	170	1.001	3	0.45	8.8	6
10	white	6.3	0.3	0.34	1.6	0.049	14	132	0.994	3.3	0.49	9.5	6
11	white	8.1	0.22	0.43	1.5	0.044	28	129	0.9938	3.22	0.45	11	6
12	white	8.1	0.27	0.41	1.45	0.033	11	63	0.9908	2.99	0.56	12	5
13	white	8.6	0.23	0.4	4.2	0.035	17	109	0.9947	3.14	0.53	9.7	5
14	white	7.9	0.18	0.37	1.2	0.04	16	75	0.992	3.18	0.63	10.8	5
15	white	6.6	0.16	0.4	1.5	0.044	48	143	0.9912	3.54	0.52	12.4	7
16	white	8.3	0.42	0.62	19.25	0.04	41	172	1.0002	2.98	0.67	9.7	5
17	white	6.6	0.17	0.38	1.5	0.032	28	112	0.9914	3.25	0.55	11.4	7
18	white	6.3	0.48	0.04	1.1	0.046	30	99	0.9928	3.24	0.36	9.6	6
19	white		0.66	0.48	1.2	0.029	29	75	0.9892	3.33	0.39	12.8	8
20	white	7.4	0.34	0.42	1.1	0.033	17	171	0.9917	3.12	0.53	11.3	6
21	white	6.5	0.31	0.14	7.5	0.044	34	133	0.9955	3.22	0.5	9.5	5
22	white	6.2	0.66	0.48	1.2	0.029	29	75	0.9892	3.33	0.39	12.8	8
23	white	6.4	0.31	0.38	2.9	0.038	19	102	0.9912	3.17	0.35	11	7
24	white	6.8	0.26	0.42	1.7	0.049	41	122	0.993	3.47	0.48	10.5	8
25	white	7.6	0.67	0.14	1.5	0.074	25	168	0.9937	3.05	0.51	9.3	5
26	white	6.6	0.27	0.41	1.3	0.052	16	142	0.9951	3.42	0.47	10	6

The dataset:

	A	B	C	D	E	F	G	H	I	J	K	L	M
4897	white	6.5		0.19	1.2	0.041	30	111	0.99254	2.99	0.46	9.4	6
4898	white	5.5	0.29	0.3	1.1	0.022	20	110	0.98869	3.34	0.38	12.8	7
4899	white	6	0.21	0.38	0.8	0.02	22	98	0.98941	3.26	0.32	11.8	6
4900	red	7.4	0.7	0	1.9	0.076	11	34	0.9978	3.51	0.56	9.4	5
4901	red	7.8	0.88	0	2.6	0.098	25	67	0.9968	3.2	0.68	9.8	5
4902	red	7.8	0.76	0.04	2.3	0.092	15	54	0.997	3.26	0.65	9.8	5
4903	red	11.2	0.28	0.56	1.9	0.075	17	60	0.998	3.16	0.58	9.8	6
4904	red	7.4	0.7	0	1.9	0.076	11	34	0.9978	3.51	0.56	9.4	5
4905	red	7.4	0.66	0	1.8	0.075	13	40	0.9978	3.51	0.56	9.4	5
4906	red	7.9	0.6	0.06	1.6	0.069	15	59	0.9964	3.3	0.46	9.4	5
4907	red	7.3	0.65	0	1.2	0.065	15	21	0.9946	3.39	0.47	10	7
4908	red	7.8	0.58	0.02	2	0.073	9	18	0.9968	3.36	0.57	9.5	7
4909	red	7.5	0.5	0.36	6.1	0.071	17	102	0.9978	3.35	0.8	10.5	5
4910	red	6.7	0.58	0.08	1.8	0.097	15	65	0.9959	3.28	0.54	9.2	5
4911	red	7.5	0.5	0.36	6.1	0.071	17	102	0.9978	3.35	0.8	10.5	5
4912	red	5.6	0.615	0	1.6	0.089	16	59	0.9943	3.58	0.52	9.9	5
4913	red	7.8	0.61	0.29	1.6	0.114	9	29	0.9974	3.26	1.56	9.1	5
4914	red	8.9	0.62	0.18	3.8	0.176	52	145	0.9986	3.16	0.88	9.2	5
4915	red	8.9	0.62	0.19	3.9	0.17	51	148	0.9986	3.17	0.93	9.2	5
4916	red	8.5	0.28	0.56	1.8	0.092	35	103	0.9969	3.3	0.75	10.5	7
4917	red	8.1	0.56	0.28	1.7	0.368	16	56	0.9968	3.11	1.28	9.3	5
4918	red	7.4	0.59	0.08	4.4	0.086	6	29	0.9974	3.38	0.5	9	4

The dataset:

	A	B	C	D	E	F	G	H	I	J	K	L	M
6478	red	6.8	0.67	0.15	1.8	0.118	13	20	0.9954	3.42	0.67	11.3	6
6479	red	6.2	0.56	0.09	1.7	0.053	24	32	0.99402	3.54	0.6	11.3	5
6480	red	7.4	0.35	0.33	2.4	0.068	9	26	0.9947	3.36	0.6	11.9	6
6481	red	6.2	0.56	0.09	1.7	0.053	24	32	0.99402	3.54	0.6	11.3	5
6482	red	6.1	0.715	0.1	2.6	0.053	13	27	0.99362	3.57	0.5	11.9	5
6483	red	6.2	0.46	0.29	2.1	0.074	32	98	0.99578	3.33	0.62	9.8	5
6484	red	6.7	0.32	0.44	2.4	0.061	24	34	0.99484	3.29	0.8	11.6	7
6485	red	7.2	0.39	0.44	2.6	0.066	22	48	0.99494	3.3	0.84	11.5	6
6486	red	7.5	0.31	0.41	2.4	0.065	34	60	0.99492	3.34	0.85	11.4	6
6487	red	5.8	0.61	0.11	1.8	0.066	18	28	0.99483	3.55	0.66	10.9	6
6488	red	7.2		0.33	2.5	0.068	34	102	0.99414	3.27	0.78	12.8	6
6489	red	6.6	0.725	0.2	7.8	0.073	29	79	0.9977	3.29	0.54	9.2	5
6490	red	6.3	0.55	0.15	1.8	0.077	26	35	0.99314	3.32	0.82	11.6	6
6491	red	5.4	0.74	0.09	1.7	0.089	16	26	0.99402	3.67	0.56	11.6	6
6492	red	6.3	0.51	0.13	2.3	0.076	29	40	0.99574	3.42	0.75	11	6
6493	red	6.8	0.62	0.08	1.9	0.068	28	38	0.99651	3.42	0.82	9.5	6
6494	red	6.2	0.6	0.08	2	0.09	32	44	0.9949	3.45	0.58	10.5	5
6495	red	5.9	0.55	0.1	2.2	0.062	39	51	0.99512	3.52		11.2	6
6496	red	6.3	0.51	0.13	2.3	0.076	29	40	0.99574	3.42	0.75	11	6
6497	red	5.9	0.645	0.12	2	0.075	32	44	0.99547	3.57	0.71	10.2	5
6498	red	6	0.31	0.47	3.6	0.067	18	42	0.99549	3.39	0.66	11	6
6499													

Results from existing research

Table 2. The accuracy with the best tuning parameters in each machine learning model.

Method	Accuracy using default parameters	Choosing best parameters	Accuracy using choosing parameters
Naive Bayes	82.29%		
Decision Tree	84.16%		
KNN	88.54%	n_neighbors=43	91.04%
Random Forest	91.04%	max_depth=20, n_estimators=50	91.25%
SVM	91.25%	C = 3, gamma = 1	92.5%

Reference:
https://www.researchgate.net/publication/372794959_Comparison_of_the_red_wine_quality_prediction_accuracy_using_5_Machine_Learning_Model

Results from existing research

“Kumar, (2020) paper is similar in that they used similar performance measurements and similar machine learning algorithms such as support vector machine and naïve Bayes. The difference is that they trained the model on unbalanced classes and they used all features for the prediction of the model. In terms of performance analysis, they achieved the best of 67.25% accuracy from the support vector machine on the red wine dataset, Er and Atasoy, (2016) has been achieved the best accuracy result from the random forest on 69.90% in the red wine and 71.23% white wine datasets and use the principal components analysis technique for feature selection. Gupta, (2018) has been proposed that all features are not necessary for the prediction instead of selecting only necessary features to predict the wine quality. For that, they used linear regression for determining the dependencies of the target variable. Whereas our model achieved 69.06% accuracy in the red wine dataset and 67.83% accuracy in the white wine dataset from the support vector machine. Then after training, the model on the balanced data and selecting the best hyperparameters the performance of the model is improved and achieved 83.52% accuracy in the red wine and 86.86% accuracy in the white wine. In addition, our model achieved the best 85.16% accuracy in the red wine and 88.28% accuracy in the white wine from the artificial neural network model by applying the Pearson coefficient correlation matrices for the feature selection.”

Reference: https://www.diva-portal.org/smash/get/diva2:1574730/FULLTEXT01.pdf

Data Preprocessing

Dataset has a few empty values

14	white	8.3	0.42	0.62	19.25	0.040	41.0	172.0	1.0002	2.98	0.67	9.7	5
15	white	6.6	0.17	0.38	1.50	0.032	28.0	112.0	0.9914	3.25	0.55	11.4	7
16	white	6.3	0.48	0.04	1.10	0.046	30.0	99.0	0.9928	3.24	0.36	9.6	6
17	white	NaN	0.66	0.48	1.20	0.029	29.0	75.0	0.9892	3.33	0.39	12.8	8

```
wine_df.isna().any()
```

```
type                    False
fixed acidity            True
volatile acidity         True
citric acid              True
residual sugar           True
chlorides                True
free sulfur dioxide     False
total sulfur dioxide    False
density                 False
pH                       True
sulphates                True
alcohol                 False
quality                 False
dtype: bool
```

Our solution:
Impute with mean of column

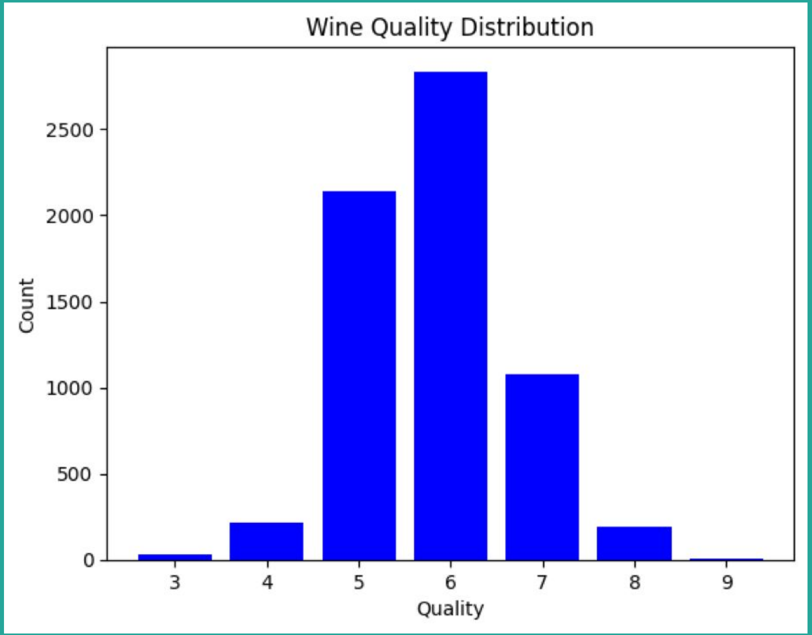
Other solutions:
Drop the rows

Distribution of the features

	fixed acidity	volatile acidity	citric acid	residual sugar	chlorides	free sulfur dioxide	total sulfur dioxide	density	pH	sulphates	alcohol	quality
count	6487.000000	6489.000000	6494.000000	6495.000000	6495.000000	6497.000000	6497.000000	6497.000000	6488.000000	6493.000000	6497.000000	6497.000000
mean	7.216579	0.339691	0.318722	5.444326	0.056042	30.525319	115.744574	0.994697	3.218395	0.531215	10.491801	5.818378
std	1.296750	0.164649	0.145265	4.758125	0.035036	17.749400	56.521855	0.002999	0.160748	0.148814	1.192712	0.873255
min	3.800000	0.080000	0.000000	0.600000	0.009000	1.000000	6.000000	0.987110	2.720000	0.220000	8.000000	3.000000
25%	6.400000	0.230000	0.250000	1.800000	0.038000	17.000000	77.000000	0.992340	3.110000	0.430000	9.500000	5.000000
50%	7.000000	0.290000	0.310000	3.000000	0.047000	29.000000	118.000000	0.994890	3.210000	0.510000	10.300000	6.000000
75%	7.700000	0.400000	0.390000	8.100000	0.065000	41.000000	156.000000	0.996990	3.320000	0.600000	11.300000	6.000000
max	15.900000	1.580000	1.660000	65.800000	0.611000	289.000000	440.000000	1.038980	4.010000	2.000000	14.900000	9.000000

Normalization needed

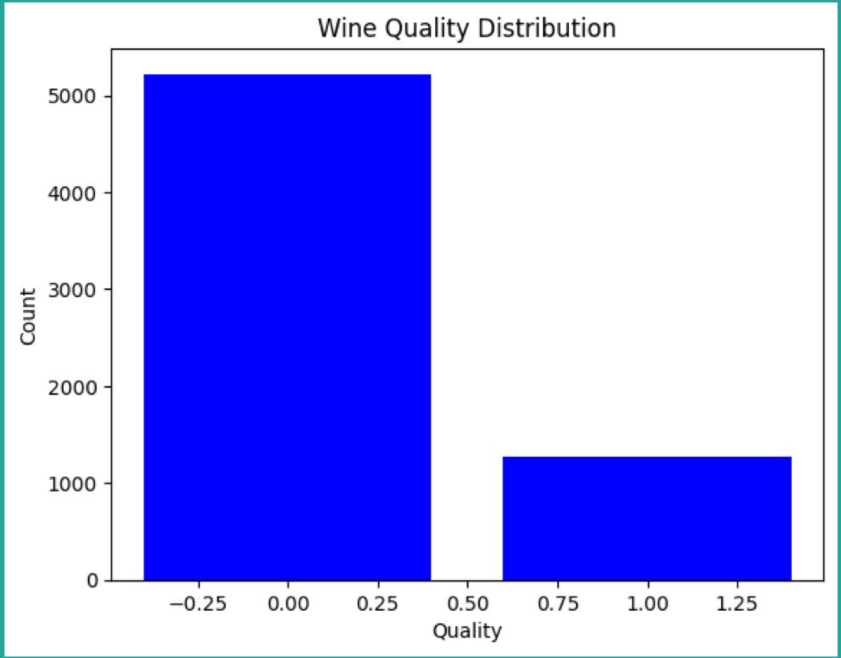
Dataset is very imbalanced



Highest quality has only 5 rows

This will pose a big challenge

Dataset is very imbalanced



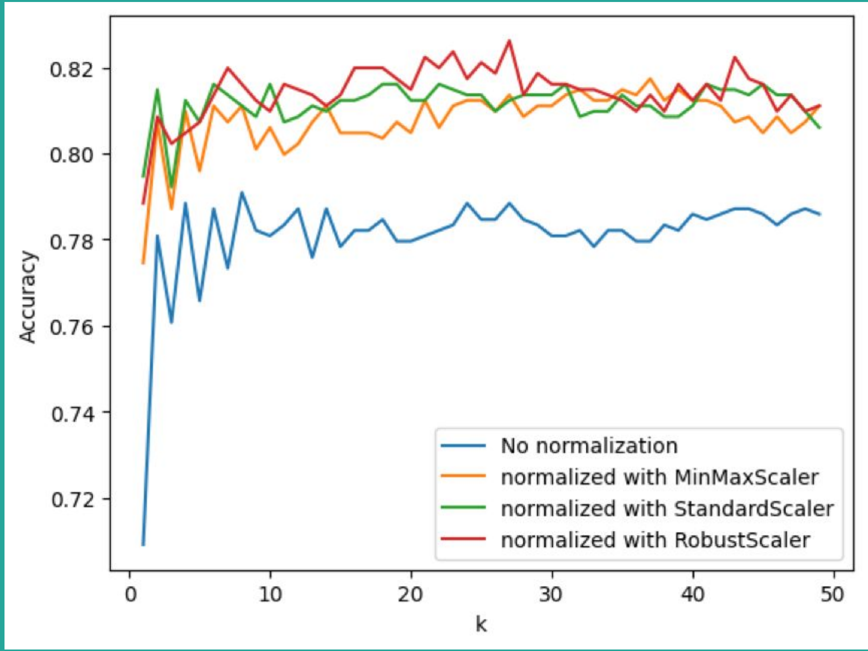
We will binarize the quality to make our job easier:

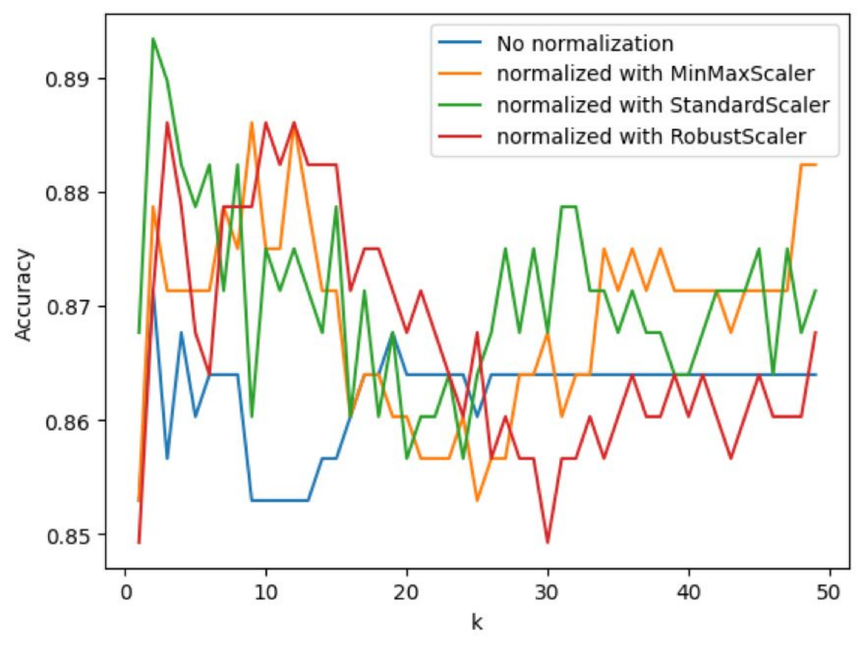
>= 7: good (1)

< 7: bad (0)

The Models

kNN





Best k for white wine: 27, accuracy: 83%

Best k for red wine: 2, accuracy: 89%

kNN

But precision and recall for the minority class are low:

Class	Precision	Recall	F1-Score
0 (Bad)	0.84	0.96	0.90
1 (Good)	0.68	0.30	0.42

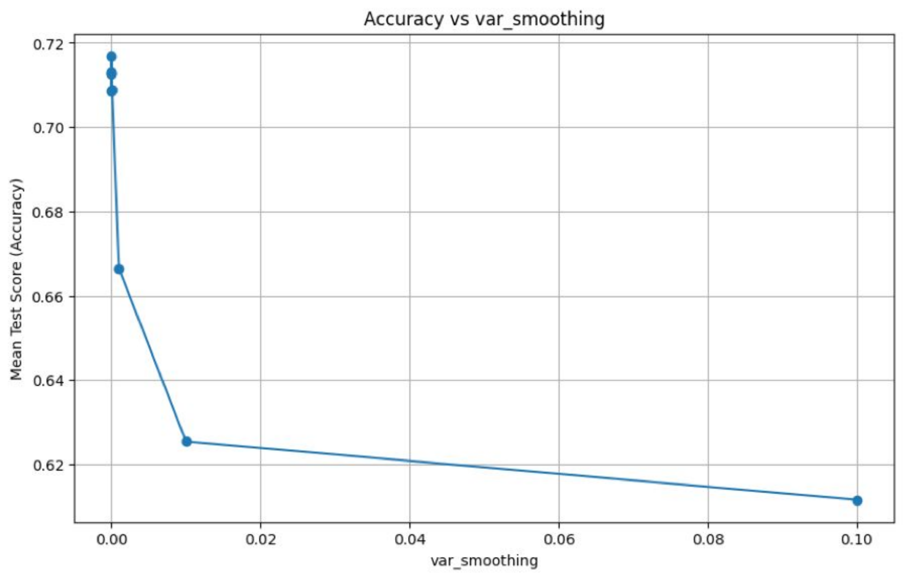
White wine

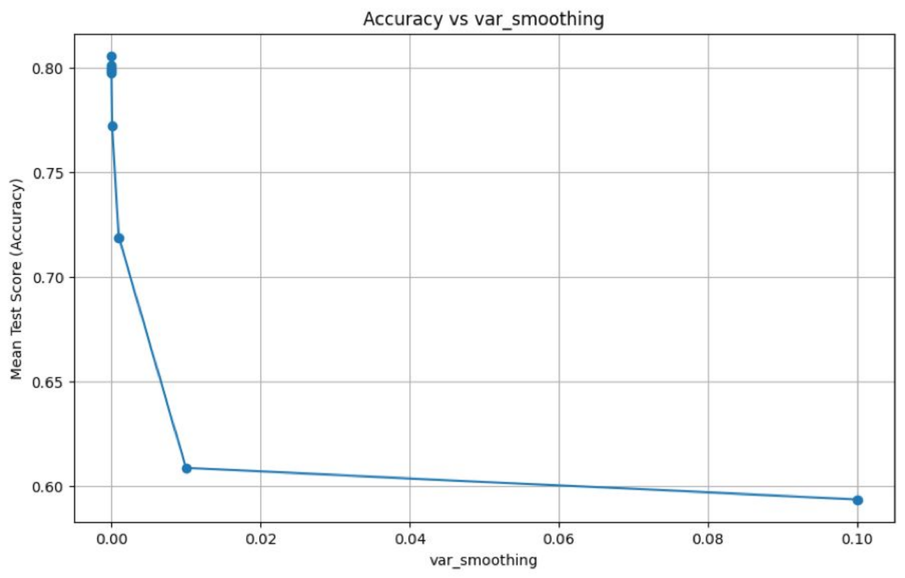
Class	Precision	Recall	F1-Score
0 (Bad)	0.90	0.99	0.94
1 (Good)	0.83	0.27	0.41

Red wine

We can increase recall by oversampling the training data using SMOTE, but at the cost of precision and overall accuracy. New recall: 87% (white wine), 54% (red wine)

Naïve Bayes





Plot of accuracy vs var_smoothing hyper-parameter on validation set for white wine (left) and red wine (right)

Best var_smoothing: 0.001, best accuracy: 80% (white wine), 89% (red wine)

Naïve Bayes

The results from raw data:

Class	Precision	Recall	F1-Score
0 (Bad)	0.83	0.94	0.88
1 (Good)	0.57	0.30	0.40

White wine, accuracy: 80%

Class	Precision	Recall	F1-Score
0 (Bad)	0.87	0.98	0.92
1 (Good)	0.33	0.08	0.12

Red wine, accuracy: 85%

Despite the high accuracy, very low precision and recall, especially on red wine

Naïve Bayes

Red wine results slightly improved after normalization, accuracy 89%, minority class precision 68%

After balancing, we can increase recall to 77% and 74%

Red wine:

Raw data:

Class	Precision	Recall	F1-Score
0 (Bad)	0.87	0.98	0.92
1 (Good)	0.33	0.08	0.12

Accuracy: 85%

Normalized data:

Class	Precision	Recall	F1-Score
0 (Bad)	0.91	0.97	0.94
1 (Good)	0.68	0.42	0.51

Accuracy: 89%

Multi Layer Perceptron Neural Network

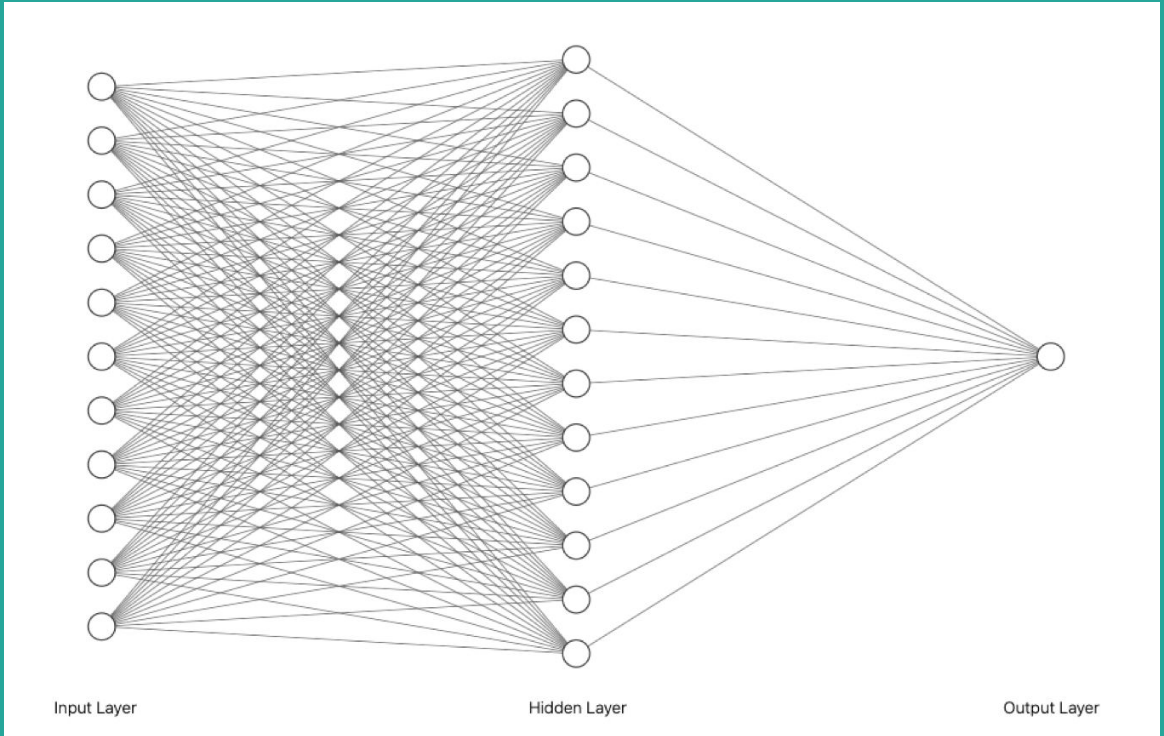
11 input nodes (one per feature)

12 hidden layer nodes

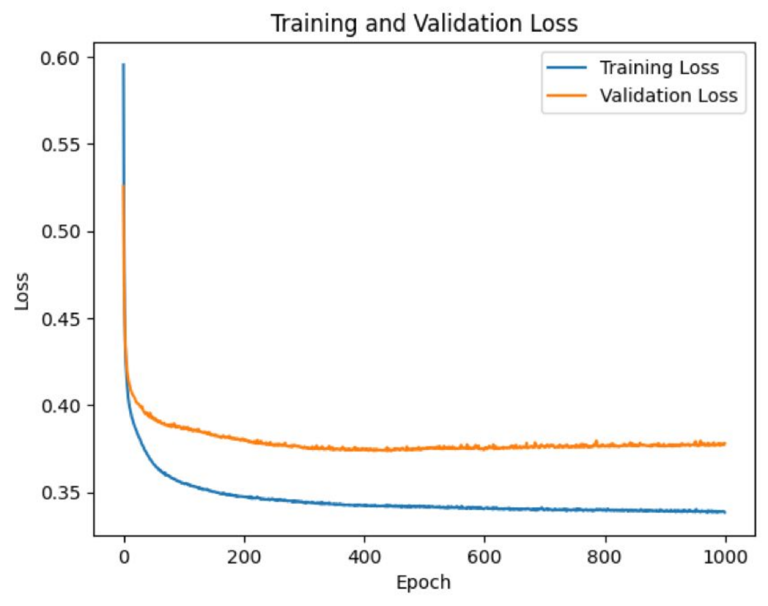
1 output (probability of being in class 1)

Sigmoid used for output

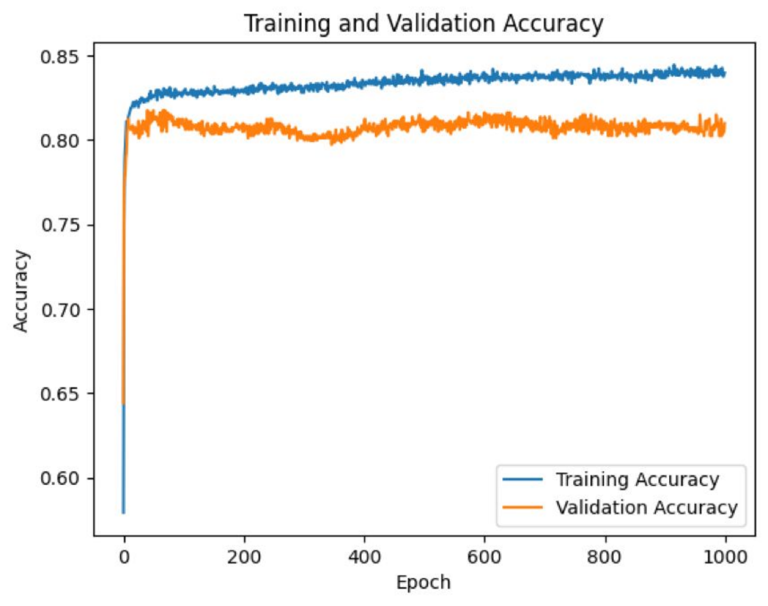
Different functions used for input->hidden



Multi Layer Perceptron Neural Network



Loss against iterations



Accuracy against iterations

Multi Layer Perceptron Neural Network

Relu transfer function:

Class	Precision	Recall	F1-Score
0 (Bad)	0.85	0.93	0.89
1 (Good)	0.62	0.39	0.48

Accuracy: 82%

Hyperbolic tangent transfer function:

Class	Precision	Recall	F1-Score
0 (Bad)	0.85	0.92	0.88
1 (Good)	0.57	0.39	0.46

Accuracy: 80%

Sigmoid transfer function:

Class	Precision	Recall	F1-Score
0 (Bad)	0.84	0.94	0.89
1 (Good)	0.62	0.35	0.45

Accuracy: 81%

Relu wins by a small margin, but accuracy can go up and down due to randomness

Multi Layer Perceptron Neural Network

Class	Precision	Recall	F1-Score
0 (Bad)	0.92	0.98	0.95
1 (Good)	0.76	0.44	0.56

Accuracy: 91%

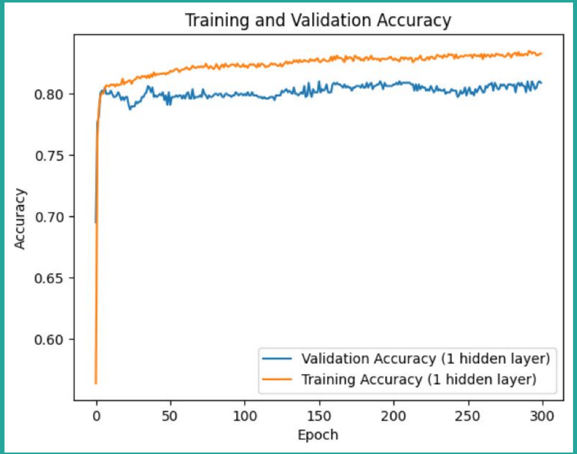
Red wine (Relu)

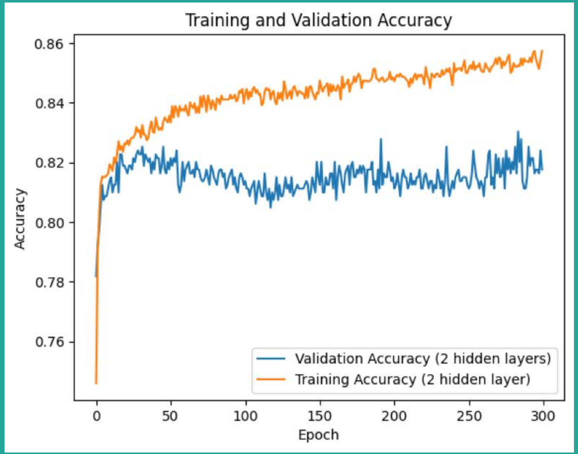
After oversampling we get 76% recall on the white wine and 77% on the red wine for the minority class

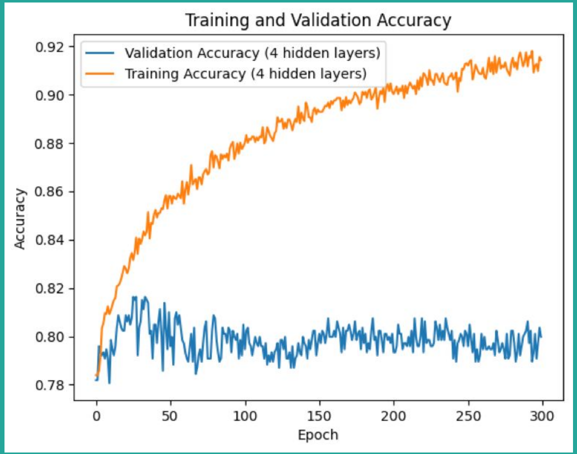
Deep Learning

Deep Learning is performed when the neural network has more than one hidden layer

We tried to gradually add hidden layers (each 12 nodes)







Network quickly overfits and validation accuracy does not get better

Deep Learning

Best results with:

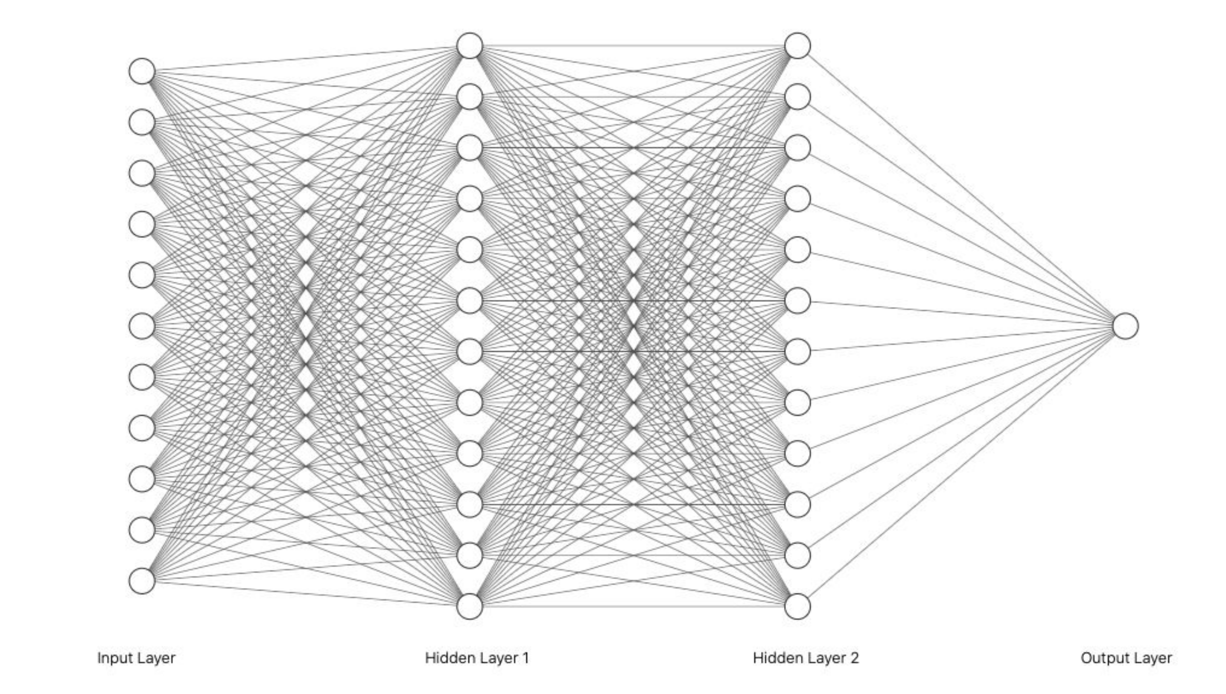
11 input nodes (one per feature)

12 hidden layer nodes (2 hidden layers)

1 output (probability of being in class 1)

Sigmoid used for output

Relu used for hidden layers



Deep Learning

White wine:

Class	Precision	Recall	F1-Score
0 (Bad)	0.85	0.95	0.89
1 (Good)	0.66	0.39	0.48

Accuracy: 82%

Red wine:

Class	Precision	Recall	F1-Score
0 (Bad)	0.91	0.99	0.95
1 (Good)	0.80	0.37	0.51

Accuracy: 90%

Best results table

Logistic Regression

The results from raw data:

Class	Precision	Recall	F1-Score
0 (Bad)	0.83	0.95	0.89
1 (Good)	0.62	0.29	0.39

White wine, accuracy: 81%

Class	Precision	Recall	F1-Score
0 (Bad)	0.90	0.98	0.94
1 (Good)	0.74	0.33	0.45

Red wine, accuracy: 89%

Logistic Regression

We tried different normalization techniques. It did not significantly improve overall accuracy but affected class-specific performance metrics differently, for example:

White wine:

Normalized data (with StandardScaler):

Class	Precision	Recall	F1-Score
0 (Bad)	0.81	0.99	0.89
1 (Good)	0.74	0.15	0.25

Accuracy: 81%

Red wine:

Normalized data (with RobustScaler):

Class	Precision	Recall	F1-Score
0 (Bad)	0.91	0.97	0.94
1 (Good)	0.68	0.35	0.46

Accuracy: 89%

Balancing increased recall to 74% for the white wine and 81% for the red wine

Support Vector Machines

The model with default parameters and without oversampling for white wine:

White wine:

Class	Precision	Recall	F1-Score
0 (Bad)	0.78	1.00	0.88
1 (Good)	0.00	0.00	0.00

Accuracy: 78%

The model with default parameters and without oversampling for red wine:

Red wine:

Class	Precision	Recall	F1-Score
0 (Bad)	0.87	1.00	0.93
1 (Good)	0.00	0.00	0.00

Accuracy: 87%

Support Vector Machines

We will standardize the data and perform a grid search over different kernel types to find optimal parameters.
The best results were obtained using the default rbf kernel => the data is not linearly separable.

Results:

White wine:

Class	Precision	Recall	F1-Score
0 (Bad)	0.95	0.73	0.83
1 (Good)	0.47	0.86	0.61

Accuracy: 76%

Balanced data:

Class	Precision	Recall	F1-Score
0 (Bad)	0.95	0.76	0.84
1 (Good)	0.49	0.85	0.62

In the white wine, balancing has barely affected the results.

Support Vector Machines

In the red wine, balancing made recall for the minority class higher at the expense of precision:

Red wine:

Class	Precision	Recall	F1-Score
0 (Bad)	0.91	0.98	0.94
1 (Good)	0.76	0.37	0.50

Accuracy: 90%

Balanced data:

Class	Precision	Recall	F1-Score
0 (Bad)	0.96	0.89	0.93
1 (Good)	0.52	0.77	0.62

Accuracy: 88%

Decision Trees

Decision trees are not affected by scaling or normalization like other models so we will not be applying it here. We will perform a grid search to find the best values for the hyperparameters.

Results:

White wine:

Class	Precision	Recall	F1-Score
0 (Bad)	0.90	0.89	0.90
1 (Good)	0.62	0.65	0.64

Accuracy: 84%

Red wine:

Class	Precision	Recall	F1-Score
0 (Bad)	0.93	0.98	0.96
1 (Good)	0.80	0.56	0.66

Accuracy: 92%

Confusion matrices for the white and red wines respectively:

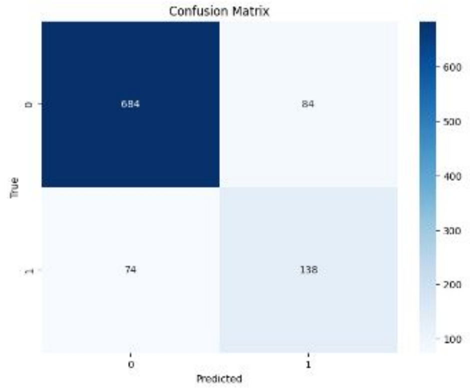


Figure 17

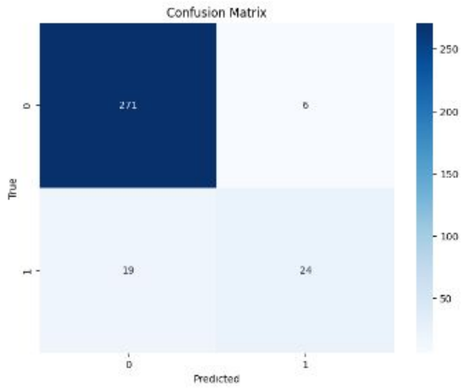


Figure 18

Decision Trees

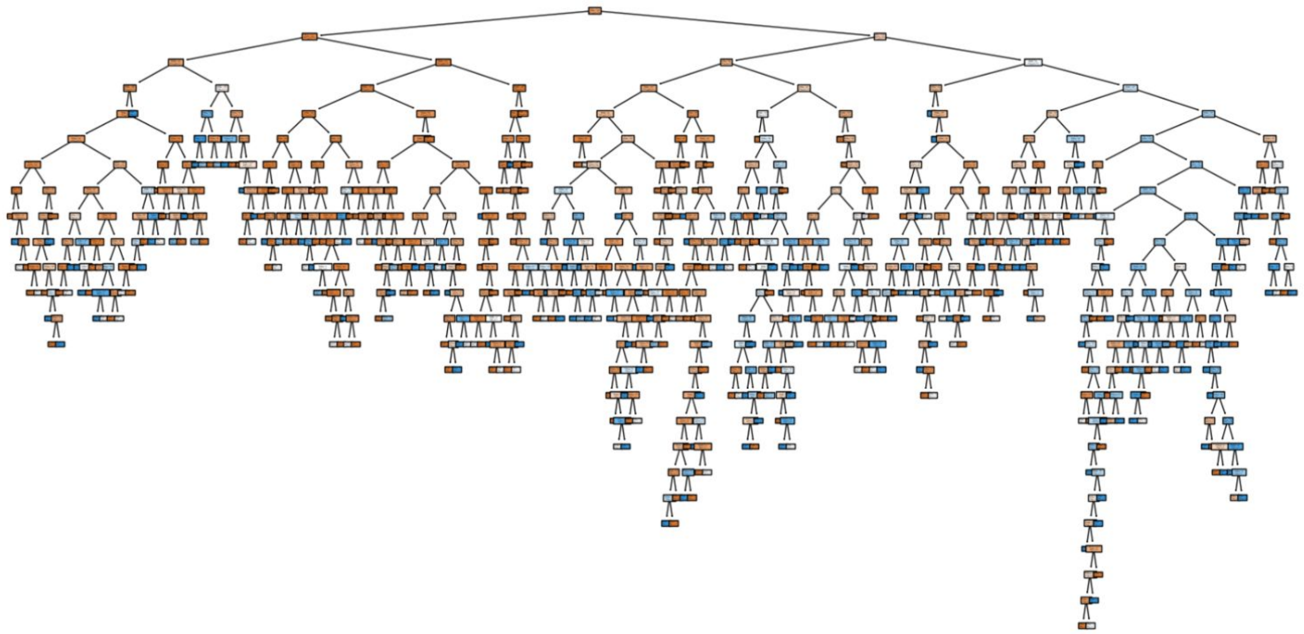


Figure 19

Resulting tree for the white wine

Decision Trees

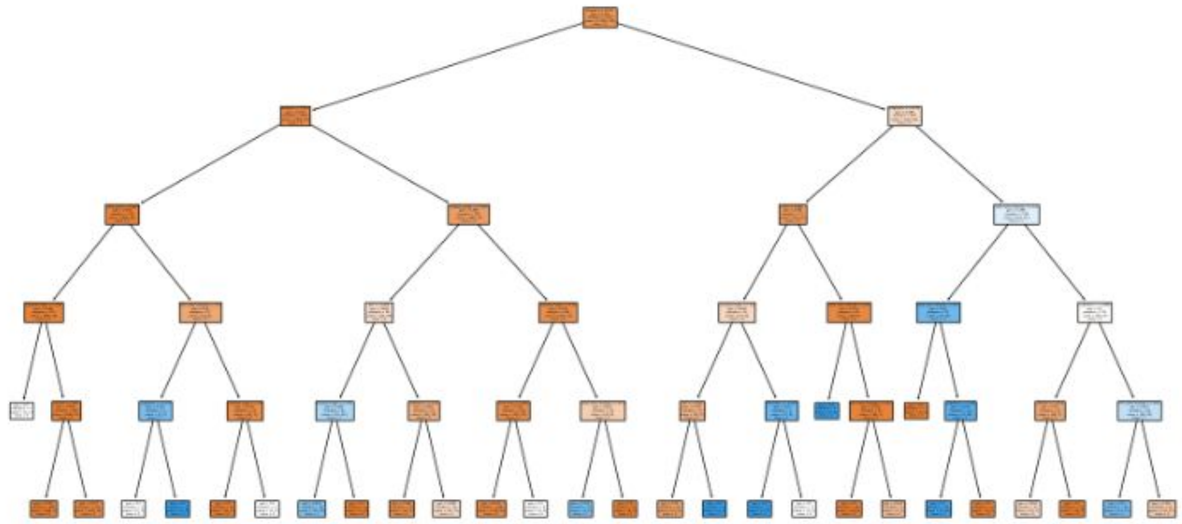


Figure 20

Resulting tree for the red wine

Random Forest

Like before we will perform a grid search to find the best hyperparameters.
Best results:

White wine:

Class	Precision	Recall	F1-Score
0 (Bad)	0.90	0.96	0.93
1 (Good)	0.83	0.63	0.71

Accuracy: 89%

Red wine:

Class	Precision	Recall	F1-Score
0 (Bad)	0.95	0.99	0.97
1 (Good)	0.93	0.63	0.75

Accuracy: 94%

Random Forest

After performing balancing with SMOTE:

White wine:

Class	**Precision**	**Recall**	**F1-Score**
0 (Bad)	0.95	0.89	0.92
1 (Good)	0.68	0.82	0.74

Accuracy: 88%

Red wine:

Class	**Precision**	**Recall**	**F1-Score**
0 (Bad)	0.96	0.94	0.95
1 (Good)	0.66	0.72	0.69

Accuracy: 91%

As before, oversampling increases recall at the expense of precision and overall accuracy.

Conclusions

Conclusions

Summarized results tables shown in the right. In bold are the best values of the column

Summarized table for white wine:

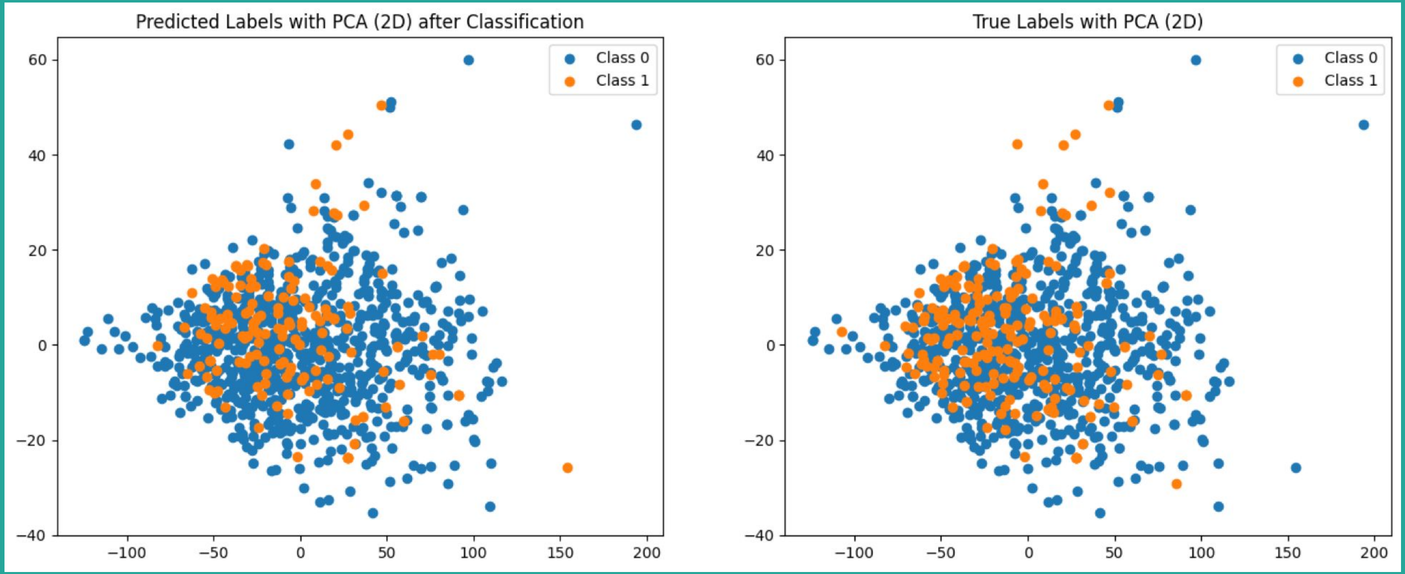
Model	Accuracy	Precision (0)	Precision (1)	Recall (0)	Recall (1)
kNN	83%	84%	68%	96%	30%
Naïve Bayes	80%	83%	57%	94%	30%
Logistic Regression	81%	81%	74%	**99%**	15%
Support Vector Machines	76%	**95%**	47%	73%	**86%**
ANN	82%	85%	62%	93%	39%
Deep Learning	82%	85%	66%	95%	39%
Decision Trees	84%	90%	62%	89%	65%
Random Forest	**89%**	90%	**83%**	96%	63%

Summarized table for red wine:

Model	Accuracy	Precision (0)	Precision (1)	Recall (0)	Recall (1)
kNN	89%	90%	83%	**99%**	27%
Naïve Bayes	89%	91%	68%	97%	42%
Logistic Regression	89%	91%	68%	97%	35%
Support Vector Machines	90%	91%	76%	98%	37%
ANN	91%	92%	76%	98%	44%
Deep Learning	90%	91%	80%	**99%**	37%
Decision Trees	92%	93%	80%	98%	56%
Random Forest	**94%**	**95%**	**93%**	**99%**	**63%**

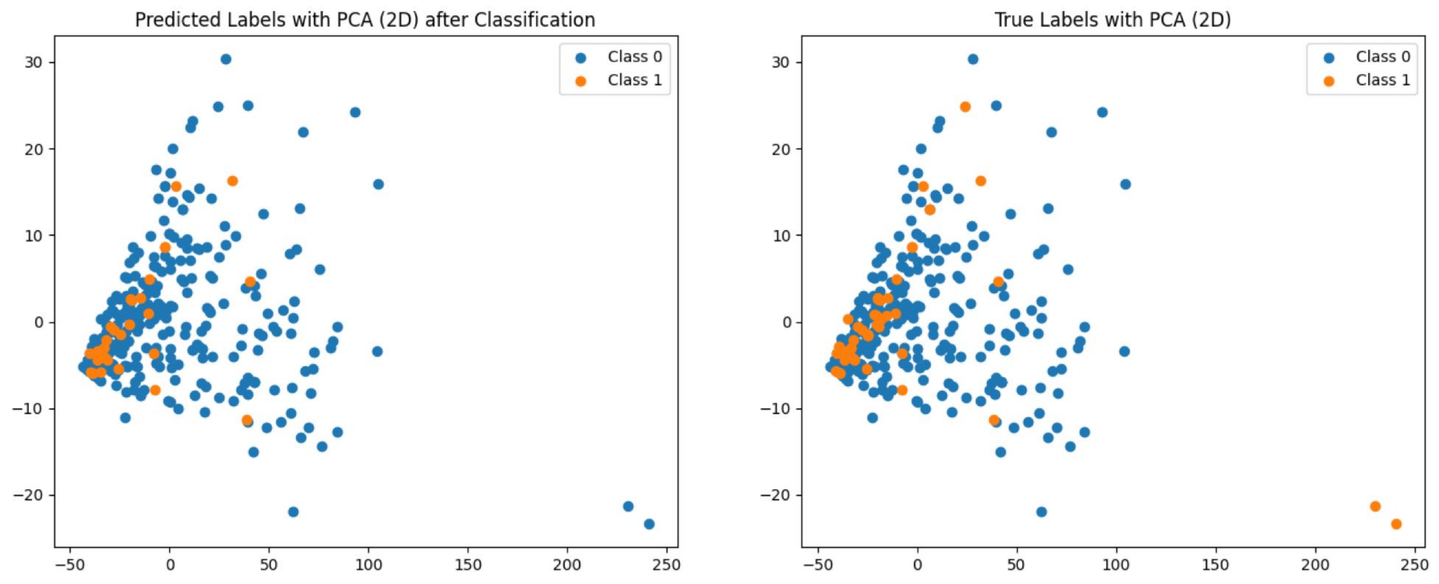
Conclusions

RandomForest performed the best out of the models and configurations that we tried. It gave us 89% accuracy for the white wine and 94% for the red wine.
We can use PCA to visualize the results in 2D, white wine:



Conclusions

Red wine:



Thank you for your attention!