Freie Universität Bozen
Libera Università di Bolzano
Università Liedia de Bulsan

Capstone Project Report

Pinot Blanc Wine Quality Analysis and Estimation

Bianchi Edoardo

Introduction

This Capstone Project Report delves into the world of wine quality analysis, employing a comprehensive approach to explore, preprocess, analyze, and model data related to Pinot Blanc wine. In particular, the project is focused on the analysis and estimation of the Pinot Blanc wine quality. Quality values are expressed in decimal numbers which range from 1 (very low) to 10 (very high).

In this report, we detail our efforts in data preprocessing, exploratory data analysis, feature selection, regression modeling, and even the generation of synthetic data to enhance the modeling process.

The initial stage of our work involved data preprocessing and cleaning. Collaborating closely with domain experts and data providers, we addressed potential inconsistencies, formatting errors, and missing values in the provided CSV files.

Subsequently, we conducted exploratory data analysis to gain insights into the attributes influencing Pinot Blanc wine quality. Through features engineering, including the essential encoding of categorical variables, we transformed the dataset into a format suitable for machine learning. Employing various correlation techniques, we identified the most influential features and their relationships, which was crucial for the feature selection phase.

The report then describes the extensive process of feature selection, guided by statistical tests. Correlation matrices, one-vs-all correlations, ANOVA F-Test, and Mutual Information Score were utilized to select the most impactful features, ensuring that our models would be built upon relevant and non-correlated attributes.

For the regression modeling phase, we used two different strategies: the Random Forest Regressor and a custom neural network called "simpleWineNet". These initial regression models, run on the original data set, served as our baseline for subsequent evaluations. This initial baseline analysis formed the basis for identifying potential areas of improvement. The small size of the original dataset, consisting of a limited number of observations, encouraged the integration of synthetic data.

Recognizing the potential of generative models, we introduced WineGAN, a generative adversarial network designed to synthesize new data samples. WineGAN's generator and discriminator components were carefully defined, and the network was trained on the available data, resulting in the generation of 10,000 synthetic samples.

The report continues with the evaluation of the synthetic data's quality and the integration of this synthetic data into the modeling pipeline. We assessed the performance of our established regression models on the augmented dataset containing both original and synthetic data, providing a comprehensive analysis of their effectiveness in improving predictive outcomes. In particular, the integration of synthetic data increased the performance of the Random Forest Regressor by 33.25% and that of the neural network by 76%.

Through these multifaceted steps and methodologies, this report explores the process of wine quality analysis, from data preprocessing to model integration, revealing insights, methodologies, and outcomes that contribute to a comprehensive understanding of Pinot Blanc wine quality assessment.

1. Data Preprocessing

The first step we conducted was data cleaning and preprocessing. Specifically, we examined the content of the CSV files related to the sensory analysis of Pinot Blanc wine to identify and address potential inconsistencies or data formatting errors. Additionally, we handled the presence of missing values. These operations were made possible through fruitful collaboration with domain experts and the data provider, who assisted us in gaining a comprehensive understanding of the provided data.

As a result of these operations, we obtained a clean and preprocessed dataset comprising qualitative and technical information from various Pinot Blanc samples. The attributes present in this dataset are as follows:

```
['Vineyard', 'Fermentation', 'Vinification', 'storage', 'clarity',
'colour-tonality', 'colour-intensity', 'overall-intensity',
'olfactory-citrus-fruit', 'olfactory-pome-tree-fruit',
'olfactory-yellow-tree-fruit', 'olfactory-tropical-fruit',
'olfactory-fresh-vegetative', 'olfactory-nutty', 'olfactory-floral',
'olfactory-spicy', 'olfactory-cleanness', 'acidity-sourness',
'saltiness-sapidity', 'bitterness', 'sweetness', 'astringency-tannicity',
'warmness', 'gustatory-citrus-fruit', 'gustatory-pome-tree-fruit',
'gustatory-yellow-tree-fruit', 'gustatory-spicy', 'gustatory-cleanness',
'overall-quality']
```

The attributes `'Vineyard', 'Fermentation', 'Vinification', 'storage'` are categorical and will be encoded as explained in section 2.1.1. The target variable indicating the wine quality is `'overall-quality'`.

2. Exploratory Data Analysis and Profiling

In this section we present the feature engineering and selection methods that led us to select the subset of features most significant in determining wine quality.

2.1. Features Engineering

The process of extracting features from data and transforming them into a more suitable format for machine learning is named Feature Engineering. Encoding categorical variables is a classical step in this phase. It is also important to note that scaling or normalization operations are not necessary in this case since all attributes have a common range of values

2.1.1. Encoding Categorical Variables

Encoding categorical variables is important because algorithms are not always able to interpret or do calculations on non-numerical values. We encoded the categorical variables using a One Hot Encoding solution. One-hot encoding creates a new binary column for each value of the original variable. The new binary value is like a flag and indicates the presence or absence of one of the categories.

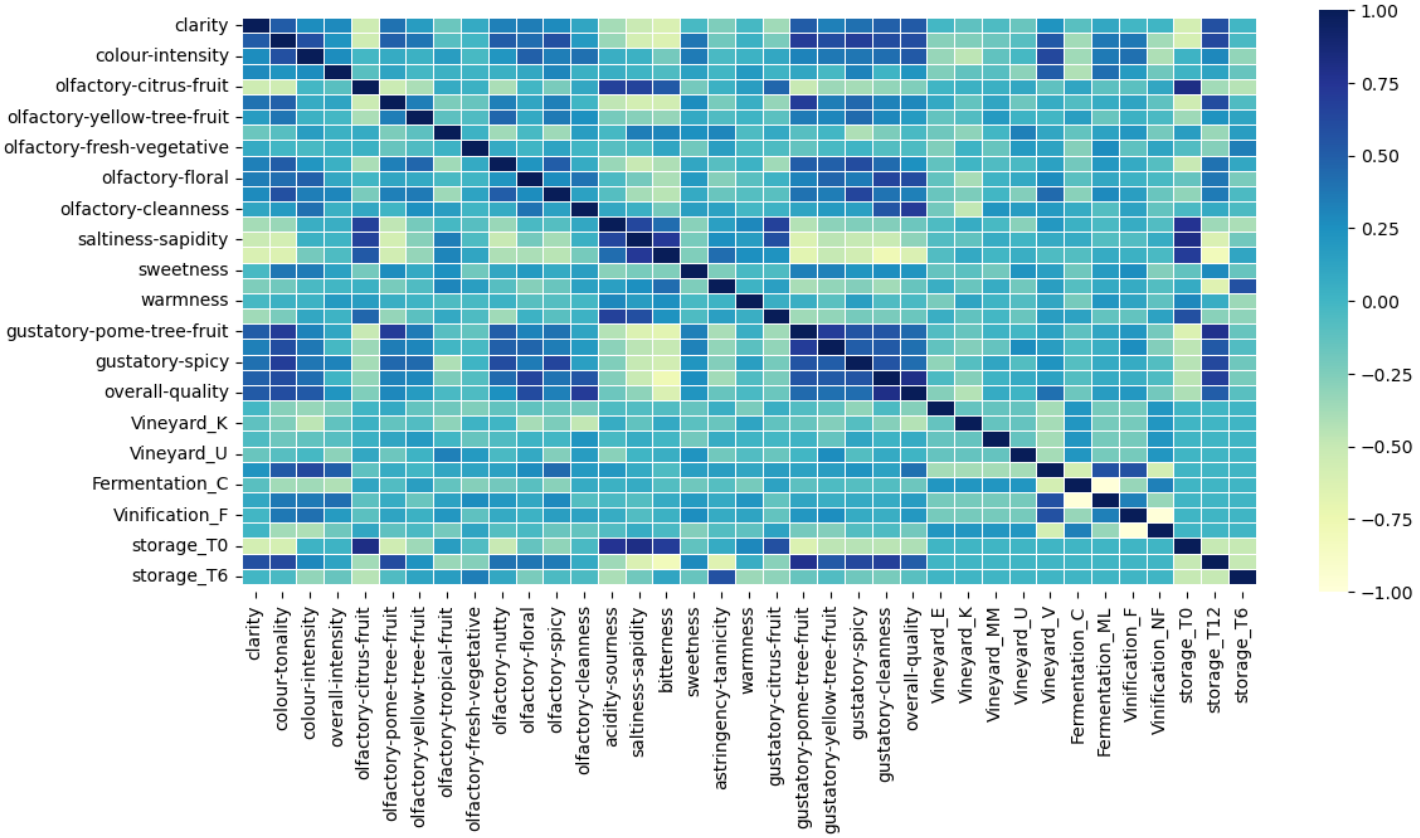
One Hot Encoding works well when the number of categories is not big. A disadvantage of this approach is the significant increase in the dimensionality of the dataset. Generally one of these new columns is dropped before fitting the algorithm, to avoid multicollinearity (Multicollinearity occurs where there is a dependency between the independent features).

2.2. Features Importance and Selection

In this section we compare and examine correlations between attributes in order to understand what are the most important features in determining the overall wine quality. In particular correlation matrices, statistical tests and mutual information are the methods used. We need only the features which are highly dependent on the response variable. It is important to remember that the categorical variables were coded using a One Hot approach. However, the calculation of correlations remains unchanged.

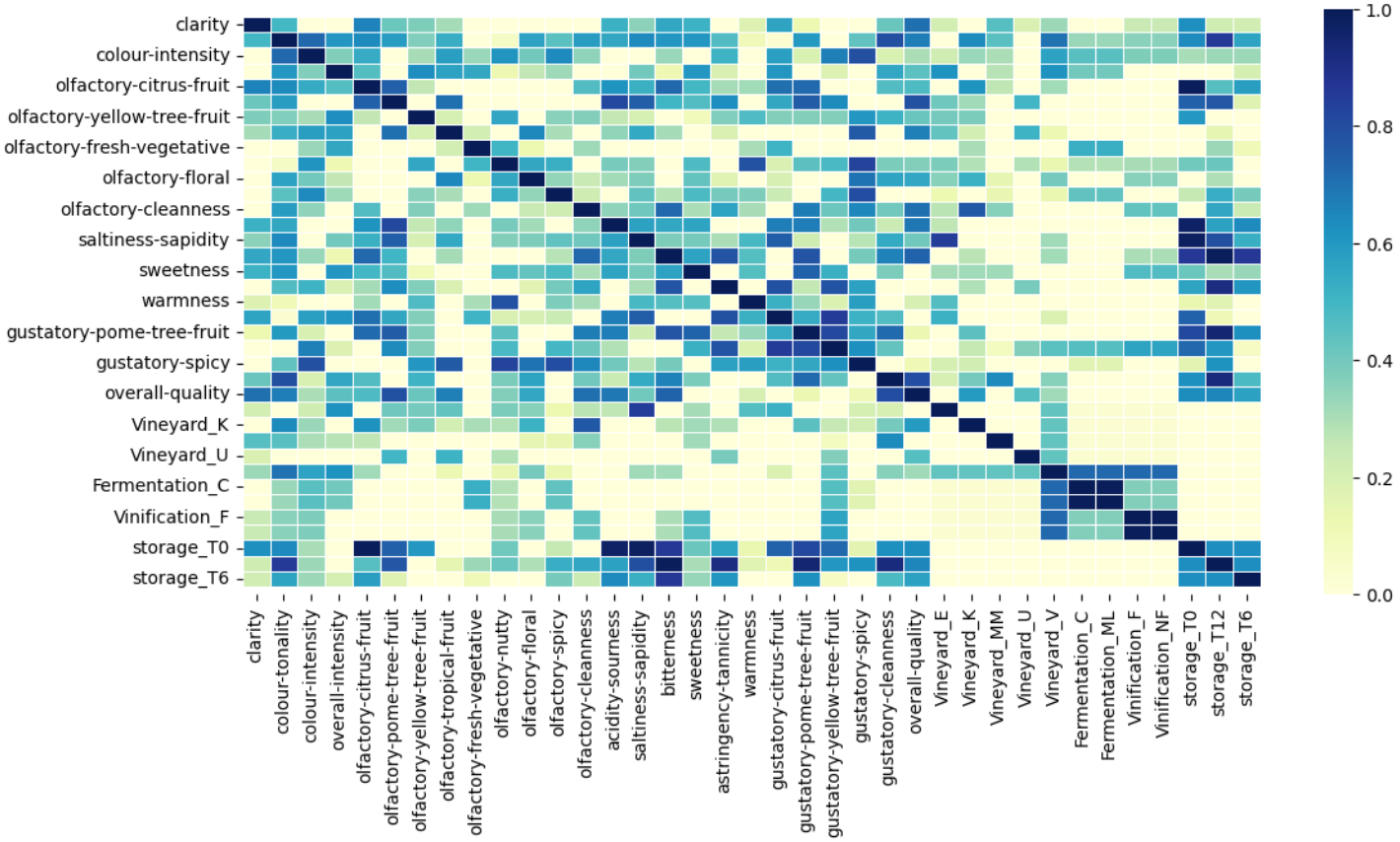
2.2.1. Correlation Matrix - Spearman Coefficient

The following heatmap is a correlation matrix computed with the Spearman coefficient and plotted with seaborn. The Spearman coefficient assesses how well the relationship between two variables can be described using a monotonic function. The vertical labels in the correlation matrix are cutted due to rendering in the notebook. The order of the labels is the same as the horizontal axis, in which all the labels are displayed.



2.2.2. Correlation Matrix - Phi-K Coefficient

The following heatmap is, like the above one, a correlation matrix but computed with the Phi-K coefficient and plotted with seaborn. According to the original paper, Phi-K is *a new correlation coefficient between categorical, ordinal and interval variables with Pearson characteristics.* The Phi-K coefficient is also capable of detecting nonlinear correlations. The vertical labels in the correlation matrix are cutted due to rendering in the notebook. The order of the labels is the same as the horizontal axis, in which all the labels are displayed.



2.2.3. Spearman and Phi-K Top Correlations

Since heatmaps with multiple variables could be hard to read, the following table reports the top correlations among variables computed with Spearman and Phi-K. The values given in the table are the same as those in the respective correlation matrices.

Spearman Top-Correlations (>0.7)

storage_T0	saltiness-sapidity	0.811417
olfactory-citrus-fruit	storage_T0	0.811329
overall-quality	gustatory-cleanness	0.801718
storage_T12	gustatory-pome-tree-fruit	0.780942
acidity-sourness	storage_T0	0.756636
saltiness-sapidity	bitterness	0.722103
olfactory-cleanness	overall-quality	0.709337
colour-tonality	gustatory-pome-tree-fruit	0.706583
gustatory-pome-tree-fruit	olfactory-pome-tree-fruit	0.701306
bitterness	storage_T0	0.700199
	gustatory-cleanness	-0.776403
	storage_T12	-0.812103
Fermentation_C	Fermentation_ML	-1.000000

Phi-K Top-Correlations (>0.8)

Fermentation_ML	Fermentation_C	0.996026
storage_T12	bitterness	0.995508
olfactory-citrus-fruit	storage_T0	0.991751
saltiness-sapidity	storage_T0	0.980092
storage_T0	acidity-sourness	0.971195
storage_T12	gustatory-pome-tree-fruit	0.946256
	gustatory-cleanness	0.924095
astringency-tannicity	storage_T12	0.911482
storage_T0	bitterness	0.863060
storage_T12	colour-tonality	0.857754
gustatory-yellow-tree-fruit	gustatory-citrus-fruit	0.847936
Vineyard_E	saltiness-sapidity	0.846242
gustatory-spicy	olfactory-nutty	0.828092
gustatory-yellow-tree-fruit	gustatory-pome-tree-fruit	0.815480
acidity-sourness	olfactory-pome-tree-fruit	0.814107
gustatory-pome-tree-fruit	storage_T0	0.808613
overall-quality	gustatory-cleanness	0.800145

2.2.4. One-vs-All: How the Target Variable Relates to All the Others

In this section, we propose a Spearman and Phi-K one-vs-all correlation analysis. This analysis is aimed at understanding the extent to which the predictor variables in the dataset are correlated with the target variable `'overall-quality'`. This operation involves correlating each variable with the response variable. For reference, we note that the correlation of the response variable with itself is equal to 1.

One-vs-All Spearman Coefficient

overall-quality	1.000000
gustatory-cleanness	0.801718
olfactory-cleanness	0.709337
olfactory-floral	0.609984
colour-tonality	0.595630
colour-intensity	0.524250
clarity	0.516183
storage_T12	0.507796
gustatory-pome-tree-fruit	0.439911
gustatory-spicy	0.424228
Vineyard_V	0.411007
gustatory-yellow-tree-fruit	0.408999
olfactory-spicy	0.332726
olfactory-pome-tree-fruit	0.290311
olfactory-nutty	0.243322
overall-intensity	0.226319
sweetness	0.223150
Vinification_F	0.196442
olfactory-yellow-tree-fruit	0.186795
Fermentation_ML	0.172104
Vineyard_MM	0.072836
Vineyard_U	0.009105
olfactory-tropical-fruit	-0.035844
warmness	-0.036896
olfactory-fresh-vegetative	-0.052060
storage_T6	-0.097407
astringency-tannicity	-0.154357
gustatory-citrus-fruit	-0.165289
Fermentation_C	-0.172104
acidity-sourness	-0.179206
Vinification_NF	-0.196442
olfactory-citrus-fruit	-0.251578
Vineyard_E	-0.270860
saltiness-sapidity	-0.301964
storage_T0	-0.410389
Vineyard_K	-0.432465
bitterness	-0.617945

One-vs-All Phi-K Coefficient

overall-quality	1.000000
gustatory-cleanness	0.800145
olfactory-pome-tree-fruit	0.787824
bitterness	0.736843
olfactory-cleanness	0.705451
clarity	0.704287
acidity-sourness	0.686571
colour-tonality	0.668800
olfactory-tropical-fruit	0.664996
storage_T12	0.650301
storage_T0	0.633984
Vineyard_K	0.593584
storage_T6	0.572635
olfactory-floral	0.554121
olfactory-citrus-fruit	0.474088
Vineyard_U	0.460171
overall-intensity	0.442058
olfactory-yellow-tree-fruit	0.421528
saltiness-sapidity	0.406168
olfactory-nutty	0.368631
Vineyard_V	0.316867
colour-intensity	0.299315
warmness	0.198007
gustatory-pome-tree-fruit	0.130582
gustatory-spicy	0.114992
Vinification_NF	0.000000
Vinification_F	0.000000
Fermentation_ML	0.000000
Fermentation_C	0.000000
olfactory-spicy	0.000000
Vineyard_MM	0.000000
olfactory-fresh-vegetative	0.000000
gustatory-yellow-tree-fruit	0.000000
gustatory-citrus-fruit	0.000000
astringency-tannicity	0.000000
sweetness	0.000000
Vineyard_E	0.000000

2.2.5. ANOVA F-Test and Mutual Information Score for Features Selection

In this section, we propose two additional tests for feature selection. The first one is the ANOVA F-Test, which relies on the F-statistic, while the second one is based on the dependency between variables.

The F-value in an ANOVA is calculated as: *variation between sample means / variation within the samples*. When the F-value increases, the p-value decreases. The F-test allows us to understand how much each variable is significant for the target variable. In order to do this test, we used the SelectKBest() function with f_regression as a scoring function. We calculated the f-test and the p-values at a significance level of 99% (alpha = 0.01). In this example, the SelectKBest function is used to sort features by importance, consequently K is equal to 'all' so that all features are considered.

Another method used to check the importance of the features is the Mutual Information Score. Mutual information (MI) between two variables is a measure of dependency between the variables. It is equal to zero only if two variables are independent and higher values mean higher dependency. In this example we calculate the score between the predictors and the target variable.

The following table reports the results of the ANOVA F-Test and the Mutual Information Score.

ANOVA F-Test (99% significance)

Feature	ANOVA F-stats
gustatory-cleanness	91.843795
clarity	77.803691
colour-tonality	38.424345
bitterness	34.405098
olfactory-cleanness	29.970891
olfactory-floral	26.729551
storage_T0	15.180948
gustatory-pome-tree-fruit	15.166074
storage_T12	14.073269
Vineyard_V	12.073276
Vineyard_K	11.170641
colour-intensity	11.165068
olfactory-citrus-fruit	10.623875
gustatory-yellow-tree-fruit	9.058171
saltiness-sapidity	8.943461
olfactory-pome-tree-fruit	8.349366
gustatory-spicy	8.025524

Mutual Information Score

Feature	Importance
gustatory-cleanness	0.642652
clarity	0.473517
colour-tonality	0.372589
bitterness	0.328218
olfactory-floral	0.290735
olfactory-cleanness	0.261545
colour-intensity	0.219692
warmness	0.190662
storage_T12	0.183973
gustatory-pome-tree-fruit	0.171802
olfactory-citrus-fruit	0.171781
olfactory-nutty	0.162391

2.2.6. Selecting the Most Important Features

The results of the methods presented in the previous section are consistent with each other and confirm what is observed from the confusion matrices.

We proceeded to select the best features according to Spearman and Phi-K (column "a" in the table below) and according to ANOVA and Mutual Information Score (column "b" in the table below). These features are not yet the final ones and represent an intermediate step. We note that the results of the different methods are concordant.

(a) Selected from Spearman and Phi-K Correlations	**(b) Selected from ANOVA and Mutual Info. Score**
“gustatory-cleanness”, “olfactory-pome-tree-fruit”, ”bitterness”, ”olfactory-cleanness”, ”clarity”, ”acidity-sourness”, ”colour-tonality”, ”colour-intensity”, ”olfactory-tropical-fruit”, ”storage_T12”, “storageT0”, ”Vineyard_K”, ”storage_T6”, “olfactory-floral”, “olfactory-citrus-fruit”, “Vineyard_U”, “overall-intensity”, “olfactory-yellow-tree-fruit”, “saltiness-sapidity”, “Vineyard_V”, ”gustatory-pome-tree-fruit”	“gustatory-cleanness”, ”clarity”,”colour-tonality”, ”bitterness”, ”olfactory-floral”, ”olfactory-cleanness”, ”storage_T0”, ”colour-intensity”, ”gustatory-pome-tree-fruit”, ”storage_T12”, ”Vineyard_V”, ”Vineyard_K”, ”olfactory-citrus-fruit”, ”saltiness-sapidity”, “olfactory-pome-tree-fruit”, “overall-intensity”

After comparing the features in the table above, we made the final selection by taking the intersection of the features from the previous table. The final features are those that most influence the quality of the wine and are poorly correlated with each other (avoid multicollinearity, threshold 0.7).

Final Selected Features (after comparing Confusion Matrices, One-vs-All, ANOVA, and Mutual Info Scores)
“gustatory-cleanness”, ”clarity”,”colour-tonality”, ”bitterness”, ”olfactory-floral”, ”olfactory-cleanness”, ”colour-intensity”, ”gustatory-pome-tree-fruit”, ”Vineyard_V”, ”Vineyard_K”, ”olfactory-citrus-fruit”, “olfactory-pome-tree-fruit”, “overall-intensity”

As the final step of this exploratory analysis, we exported and saved the dataset containing only the selected features. This trimmed dataset will be utilized in subsequent operations, such as model training, testing, and any further data analysis. The dataset obtained from the analyses in this chapter has 13 predictors and one target variable.

3. Regression Models

In this section we propose two different regression methods on the original data. The goal is to estimate the value of wine quality, expressed by `'overall-quality'`, using the features selected.

The first method we employed consists of a Random Forest Regressor. This choice is motivated by the robustness of this algorithm which uses a number of regression trees trained on different portions of the dataset. This ensemble technique allows combining different trees, weak regressors, and obtaining a powerful model.

The second method we adopted consists of a simple regression neural network.

In the case of the random Forest regressor, we used a dataset split of 80% training and 20% testing. For neural networks we used a split of the dataset equal to 60% training, 20% validation, and 20% testing.

3.1. Baseline with Random Forest Regressor

Random Forest Regressor is a powerful ensemble machine learning algorithm used for regression tasks. It belongs to the family of decision tree-based methods and is known for its robustness and high accuracy. The random forest algorithm builds multiple decision trees during the training process and combines their predictions to make more reliable and accurate predictions. In order to find a good model, we performed a grid search to test different hyperparameters combinations.

For evaluation, we considered the following evaluation metrics:
- **R squared score on Train Set**: The R2 coefficient of determination on the training set, indicating how well the model fits the training data.
- **R squared score on Test Set**: The R2 coefficient of determination on the test set, indicating the model's performance on unseen data.
- **MAE**: Mean Absolute Error, a metric measuring the average absolute difference between predicted and actual values.
- **MSE**: Mean Squared Error, a metric measuring the average squared difference between predicted and actual values.
- **MSLE**: Mean Squared Log Error, a metric measuring the mean squared logarithmic difference between predicted and actual values.

Random Forest Regressor Performance:

Metric	Score
R squared score on Train Set	0.963
R squared score on Test Set	0.833
Mean Absolute Error on Test Set	0.264
Mean Squared Error on Test Set	0.115
Mean Squared Log Error on Test Set	0.002

Overall, the results indicate that the Random Forest Regressor is performing well on the given dataset. It demonstrates good generalization to new data and provides accurate predictions, as evidenced by the relatively low values of MAE, MSE, and MSLE.

3.2. Baseline with “simpleWineNet”

The SimpleWineNet is a PyTorch Lightning module designed for wine quality estimation. It has dropout layers to prevent overfitting, and it uses common regression metrics to evaluate its performance. The combination of ReLU activations, dropout, and the Adam optimizer with a learning rate scheduler helps in training the model effectively.

3.2.1. Architecture

The neural network architecture consists of five fully connected (linear) layers with ReLU activation functions, followed by dropout layers to reduce overfitting. The final output layer has a single neuron and uses the ReLU activation function as well. The architecture can be summarized as follows:

```
Input (Input_dim) ->   16 ->
                       ReLU ->
                       Dropout (0.5) ->
```

24 ->
ReLU ->
Dropout (0.5) ->

24 ->
ReLU ->
Dropout (0.5) ->

16 ->
ReLU ->
Dropout (0.5) -> Output (1).

3.2.2. Loss Function

The Mean Squared Error (MSE) loss function is used to compute the difference between predicted and actual target values during training.

3.2.3. Evaluation

The network uses several metrics to monitor the model's performance during training, validation, and testing. These metrics include R squared, Mean Squared Error (MSE), Mean Absolute Error (MAE), and Mean Squared Log Error (MSLE).

3.2.4. Optimization

The network uses the Adam optimizer with an initial learning rate of 0.01 for updating the model's parameters during training. It also applies a learning rate scheduler, specifically the CosineAnnealingLR scheduler, which reduces the learning rate based on the number of epochs to help the model converge effectively. An early stopper with patience 3 stops the training if the validation loss stops decreasing.

3.2.5. Training and Validation

The model on the original data is trained for 180 epochs, with a batch size of 12. During training and validation, the model logs and tracks the losses and metrics.

In contrast, the model on synthetic data is trained for 80 epochs with batch size 64.

3.2.6. Testing

The model is also tested on a separate test dataset, and the test loss and metrics are recorded as well.

Network structure and training parameters, such as batch size and learning rate, were chosen after conducting several experiments. The following table reports the SimpleWineNet performance on original and synthetic data.

Metric	Original Data	Original + Synthetic Data
R squared score on Train Set	-9.761	-0.017
R squared score on Test Set	-0.697	-0.019
Loss on Train Set	2.933	0.480
Mean Absolute Error on Test Set	0.608	0.542

Mean Squared Error on Test Set	0.651	0.520
Mean Squared Log Error on Test Set	0.013	0.010

3.3. Results and Discussion on Baseline Regressors

Comparing the results of both regression methods, the Random Forest Regressor performed better by obtaining lower error metrics than the neural network.

This result is partly due to the ensemble technique used: the different trees were able to "correct and compensate" each other, thus going to maximize the performance of the final model.

On the other hand, the low performance of the neural network may be due to the size of the dataset, which has fewer than 50 observations. Despite the small size of the network, after numerous tests with different parameters, we could not pass the Random Forest Regressor. However, the use of dropout and learning rate scheduling allowed us to avoid overfitting and achieve convergence.

The reasons for a negative R2 in the neural network are related to the small number of samples in the training set. In particular, the model correctly captures some aspects of the data, but still cannot capture the variance well. The metrics are low even using the network on the synthetic dataset, indicating how the network is structured for small datasets and not usable on larger amounts of data.

4. Synthetic Data Generation

Neural networks generally need a lot of data, this is to prevent the network from memorizing the training set and ending up overfitting. Given the limited amount of data about wine we propose WineGAN, a generative adversarial network able to produce synthetic data from the available ones.

4.1. WineGAN for Synthetic Data Generation

WineGAN consists of a generator and a discriminator. The generator generates synthetic samples from random noise, while the discriminator tries to distinguish between real and fake samples.

We first defined a data preparation function that scales the input data using quantile transformation and min-max scaling, then we defined the GAN architecture providing both the `Generator` and the `Discriminator` classes. transformations we use were taken from another implementation of GAN for tabular data, available online at https://github.com/NextBrain-ai/nbsynthetic.

The generator consists of fully connected layers with leaky ReLU activation, dropout, and batch normalization, while the discriminator has similar layers but without batch normalization. The GAN is trained using the adversarial loss, which is the binary cross-entropy loss between the discriminator's output and the target labels.

The GAN is implemented as a subclass of `pl.LightningModule`, which provides a training interface. It defines the forward method for generating synthetic samples and configures the Adam optimizers for the generator and discriminator.
The training loop is implemented in the `training_step` method. It takes a batch of real data and performs one step of training for either the generator or the discriminator, depending on the optimizer index. For the generator, it generates fake data, computes the discriminator's output, and calculates the generator loss. For the discriminator, it calculates the discriminator loss on both real and fake data.

The class also includes a method to generate synthetic samples using the trained generator: it takes the number of samples, the scaler object defined in the preprocessing step, and column names as input, and returns a DataFrame containing the generated samples.

We trained WineGAN for 500 epochs, with batch size 48, and constant learning rate 0.002. We used the Adam optimizer for both the generator and the discriminator with beta1 = 0.5. After the training, we used WineGAN to generate 10,000 synthetic samples of data.

4.2. Synthetic Data Evaluation

Assessing the quality of the generated data is essential to understand whether the generative model can reproduce real data. To evaluate the quality of the synthetic data, two main methods are used: the first involves the use of statistical tests (Kolmogorov-Smirnov, Wilcoxon), while the second involves the visualization of kernel Density Estimations of the generated attributes.

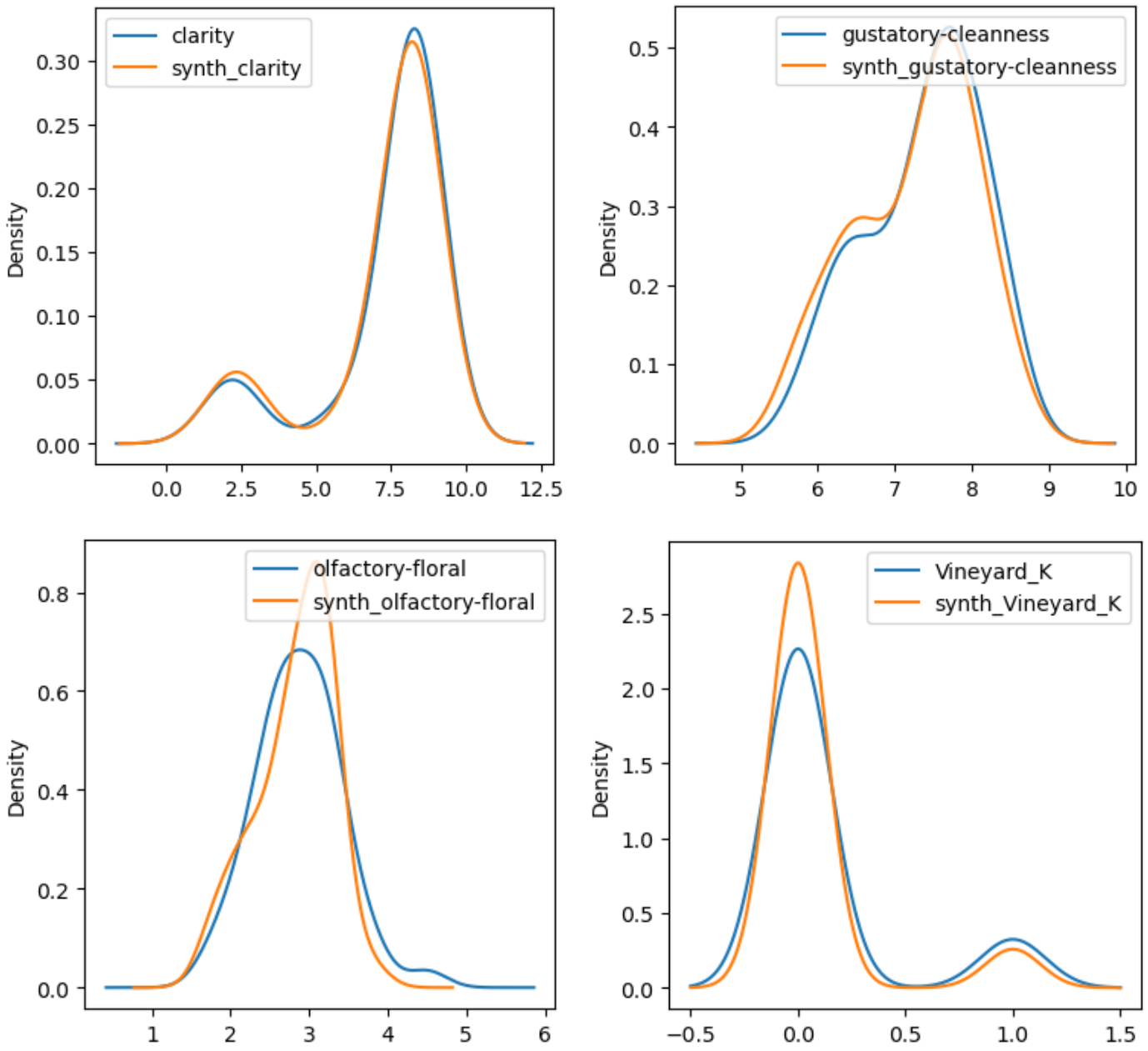
The first method for evaluating synthetic data involves the use of two statistical tests presented below:
1. Kolmogorov-Smirnov Test:
The Kolmogorov-Smirnov test is used to examine whether the two datasets share a common underlying distribution. The null hypothesis (H0) assumes that the two datasets have identical distributions, while the alternative hypothesis (H1) posits that they differ significantly. The test yields a p-value, which indicates the probability of observing the observed differences in distribution under the assumption that H0 is true. A small p-value (typically less than 0.05) rejects the null hypothesis, suggesting significant differences between the distributions.

2. Wilcoxon Signed-Rank Test:
The Wilcoxon signed-rank test is employed to compare paired samples from the datasets, taking into account their interdependence. The null hypothesis (H0) assumes that there is no significant difference between the two datasets' paired samples, whereas the alternative hypothesis (H1) posits that a significant difference exists. The test computes a signed-rank statistic and derives a p-value. A small p-value (typically less than 0.05) rejects the null hypothesis, implying notable differences between the paired samples.

By examining the p-value of the comparison between original and synthetic data for each attribute, we are able to assess the similarity of the synthetic data to the original data. We calculated the p-value for each synthetic and original attribute pair and observed that its value is always greater than 0.1. This indicates that statistically the synthetic data are similar to the original.

The second evaluation method consists in the **comparison of Kernel Density Estimation curves**. This method allows the visual evaluation of the KDE of the original and synthetic data. The closer the KDE curves are, the higher the quality of the synthetic data. The following images represent the KDE of some attributes.

In the following figures, the x-axis represents the range of values of the variable, and the y-axis represents the Kernel Density Estimation of the considered variable.



The results of both methods confirm that the synthetic data are similar to the original data. Original data and synthetic data will be merged together to create a new dataset consisting of 10,048 samples.

5. Regression on Synthetic Data

In this section we train and test two regression approaches using the synthetic data generated by WineGAN. Specifically, we propose:

1. A Random Forest Regressor trained on original + synthetic data.
2. A wider and deeper neural network suitable for working with the amount of data generated (original + synthetic).

We also compare baselines on original data from previous sections with the models trained on synthetic data. In the case of the random Forest regressor, we used a dataset split of 80% training and 20% testing. For neural networks we used a split of the dataset equal to 60% training, 20% validation, and 20% testing.

5.1. Random Forest Regressor on Synthetic Data

In this subsection, we compare the results of the random forest regressor trained on the original data with an equivalent random forest regressor trained on the new dataset containing both the original data and the synthetic data generated by WineGAN.

Random Forest Regressor Performance, Baseline vs Synthetic Data:

Metric	Original Data	Original Data + Synthetic Data
R squared score on Train Set	0.963	0.987
R squared score on Test Set	0.833	0.902
Mean Absolute Error on Test Set	0.264	0.151
Mean Squared Error on Test Set	0.115	0.047
Mean Squared Log Error on Test Set	0.002	0.001

Looking at the results we notice how the use of synthetic data increased the performance of the model: R2 on the test set increased and all error metrics dropped.

5.2. WineNet for Wine Data Regression

In this subsection, we expand SimpleWineNet to work more data. We then present WineNet, a deeper and wider network capable of making better use of the synthetic data generated with WineGAN.

The WineNet is a PyTorch Lightning module designed for wine quality estimation. It has dropout layers to prevent overfitting, and it uses common regression metrics to evaluate its performance. The combination of Leaky ReLU activations, dropout, and the Adam optimizer with a learning rate scheduler helps in training the model effectively.

5.2.1. Architecture

The neural network architecture consists of seven fully connected (linear) layers with Leaky ReLU activation functions (negative slope = 0.9), batch normalization, and dropout layers (probability 0.2) to reduce overfitting. The network has a total number of 37.5K trainable parameters. The architecture of the network is:

```
Input (self.input_dim) ->   24 ->
                            BatchNorm1d(24) ->
                            LeakyReLU (0.9) ->
                            Dropout (0.2) ->

                            64 ->
                            BatchNorm1d(64) ->
                            LeakyReLU (0.9) ->
                            Dropout (0.2) ->

                            128 ->
                            BatchNorm1d(128) ->
                            LeakyReLU (0.9) ->
                            Dropout (0.2) ->
```

128 ->
BatchNorm1d(128) ->
LeakyReLU (0.9) ->
Dropout (0.2) ->

64 ->
BatchNorm1d(64) ->
LeakyReLU (0.9) ->
Dropout (0.2) ->

24 ->
BatchNorm1d(24) ->
LeakyReLU (0.9) ->
Dropout (0.2) -> Output (1).

5.2.2. Loss Function

The Mean Squared Error (MSE) loss function is used to compute the difference between predicted and actual target values during training.

5.2.3. Evaluation

The network uses several metrics to monitor the model's performance during training, validation, and testing. These metrics include R squared, Mean Squared Error (MSE), Mean Absolute Error (MAE), and Mean Squared Log Error (MSLE).

5.2.4. Optimization

The network uses the Adam optimizer with an initial learning rate of 0.01 for updating the model's parameters during training. It also applies a learning rate scheduler, specifically the CosineAnnealingLR scheduler, which reduces the learning rate based on the number of epochs to help the model converge effectively. An early stopper stops training if the validation loss stops decreasing.

5.2.5. Training and Validation

The model is trained for 39 epochs, with a batch size of 64. During training and validation, the model logs and tracks the losses and metrics.

5.2.6. Testing

The model is also tested on a separate test dataset, and the test loss and metrics are recorded as well.

Network structure and training parameters, such as batch size and learning rate, were chosen after conducting several experiments.

WineNet and SimpleWineNet model Performance:

Metric	SimpleWineNet Original Data	SimpleWineNet Synthetic + Original Data	WineNet Original + Synthetic Data
R squared score on Train Set	-9.761	-0.017	0.678
R squared score on Test Set	-0.697	-0.019	0.885

Loss on Train Set	2.933	0.480	0.148
Mean Absolute Error on Test Set	0.608	0.542	0.182
Mean Squared Error on Test Set	0.651	0.520	0.057
Mean Squared Log Error on Test Set	0.013	0.010	0.001

Looking at The results we notice how the use of synthetic data increased the performance of the model.

Discussion and Conclusions

In this work, we analyzed dai related to Pinot Blanc wine quality, using statistical techniques, machine learning and Deep Learning. We proposed algorithms capable of estimating the final wine quality based on features that were selected through specific analysis and feature selection operations. We also proposed a GAN capable of generating synthetic data similar to the originals, allowing us to test the proposed algorithms on more data.

The table below shows the scores of all trained and tested models, both on original and synthetic data. We obtained the best results (in bold) with the Random Forest Regressor on synthetic data. Right after is WineNet on synthetic data (results underlined), which, however, has a lower R2 value on the train set than the Random Forest regressor on original data.

Comparing the various results, we can say that models trained also on synthetic data achieve better performance on average than models trained only on original data.

Metric	**Random Forest Original Data**	**Random Forest Synthetic + Original Data**	**SimpleWineNet Original Data**	**SimpleWineNet Synthetic + Original Data**	**WineNet Original + Synthetic Data**
R2 Train	<u>0.963</u>	**0.987**	-9.761	-0.017	0.678
R2 Test	0.833	**0.902**	-0.697	-0.019	<u>0.885</u>
MAE Test	0.264	**0.151**	0.608	0.542	<u>0.182</u>
MSE Test	0.115	**0.047**	0.651	0.520	<u>0.057</u>
MSLE Test	<u>0.002</u>	**0.001**	0.013	0.010	**0.001**

The Random Forest Regressor on synthetic data is the method that achieved the best performance, confirming that the ensemble technique results are often successful. Thanks to the grid search, it was also possible to systematically test different combinations of parameters and find the best one.

References

https://github.com/Diyago/GAN-for-tabular-data/tree/master
https://github.com/NextBrain-ai/nbsynthetic/tree/master
https://www.kaggle.com/code/ttunjic/gans-for-tabular-data
https://samanemami.medium.com/gan-on-tabular-data-example-with-code-96b1b09c08c4
https://scikit-learn.org/stable/modules/classes.html
https://pytorch.org/docs/stable/index.html
https://lightning.ai/docs/pytorch/latest/