COSC 5557
Pipeline Optimization

Almountassir Bellah Aljazwe

May 2024

1 Introduction

The culmination of what was learned during the class is put to test in automating the optimization and selection of the whole machine learning pipeline(s). In this task, we will use two different datasets to automate the optimization and selection of machine learning pipelines that will hopefully produce the most performant predictive results.

2 Description of Datasets

2.1 Shapes

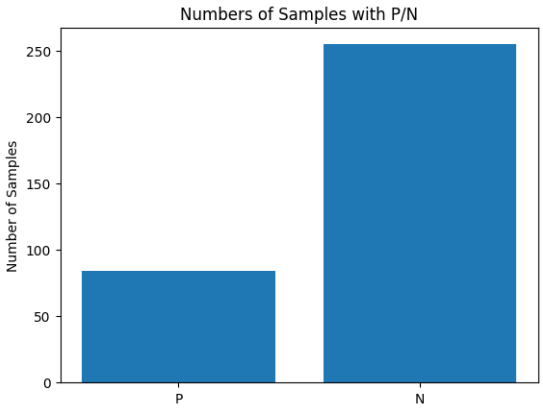
With regards to the shapes of the dataset, such as the number of samples and features, both of the datasets differ from one another. The "Primary Tumor" dataset contains 18 features, including the target feature, and 339 samples. On the other hand, the "Red Wine Quality" dataset contains 12 features, including the target feature, and 1599 samples.

2.2 Target Features

Two of the datasets have categorical target features. They differ, however, in the amount of categories a target value can be. To explain further, the "Primary Tumor" dataset contains only binary values for its target feature; these binary values are 'P' and 'N'. From the context of the dataset, these binary letters most likely represent the positive (P) presence of a primary tumor, or the negative (N) presence of a primary tumor in a single sample. On the other hand, the "Red Wine Quality" dataset contains 6 different categorical values for its target feature; these values are discrete integers,

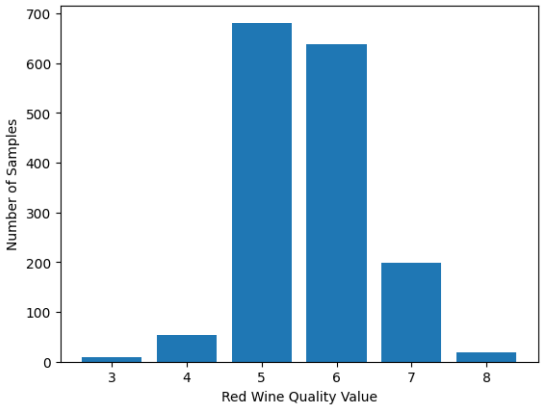
starting at '3' and ending at '8'. From the context of the dataset, these discrete integers most likely represent the number rating, with '3' being the lowest and '8' being the highest. There could be more values if more samples were to be added, such as '1' or '9', as the target feature values seem to be a number scale from one to ten.

Another important aspect, in exploring the raw dataset, is to look at the distribution of the samples with regards to the target values. The following figure displays the distribution of the "Primary Tumor" dataset :



P	N
84/339	255/339
$\approx 25\%$	$\approx 75\%$

Moving on to the "Red Wine Quality" dataset, the following figure displays the distribution of the "Red Wine Quality" dataset :



3	4	5	6	7	8
10/1559	53/1559	681/1559	638/1559	199/1559	18/1559
$\approx 0.6\%$	$\approx 3\%$	$\approx 44\%$	$\approx 41\%$	$\approx 13\%$	$\approx 1\%$

What we see from our results may offer some valuable information for us when we attempt to train our machine learning model on these datasets. In particular, the "Red Wine Quality" dataset is the most interesting; we can observe that the data distribution is cluttered around the middle quality values. This, in reality, is reasonable as it may be rare to encounter wine with extremely low quality or extremely high quality; it is reasonable to expect wine with average quality. With regards to our model, the imbalanced nature of this dataset may present obstacles in getting the best model performance; this is due to the low number of samples for quality values other than '5' and '6'. The "Primary Tumor" dataset, offers more samples for each categorical value, relative to the "Red Wine Quality" dataset. It could, however, be considered more imbalanced compared to anther dataset as there is an approximate 1:4 ratio of 'P' to 'N' target values.

2.3 Initial Issues

A clear issue when first viewing one of the raw datasets is the missing values. This issue is not present in the "Red Wine Quality" dataset, but it is present in the "Primary Tumor" dataset. For the "Primary Tumor" dataset, the following table describes the missing values, per column :

Sex	Histologic-Type	Degree-of-Diffe	Skin	Axillar
1	67	155	1	1

We can see that the "Histologic-Type" and "Degree-of-Diffe" columns contain the highest percentage of missing values : $67/339 \approx 20\%$ and $155/339 \approx 46\%$, respectively.

In terms of missing values, per row, the dataset contains 207 rows with missing values; that is $207/339 \approx 61\%$. However, most of the rows have only a single value missing. In addition, the maximum amount of missing values, per row, is only two.

2.4 Correlation

Exploring the correlation of individual features, with the target feature, is a potentially crucial piece of information; it may provide us clues as to the important features, when deciding the target feature value.

Beginning with the "Primary Tumor" Dataset, we can observe the correlation between each feature and the target feature. From what is observed, the features that offer the highest "Spearman Correlation" values are the following:

Histologic-Type	Degree-of-Diffe	Brain	Mediastinum
0.42	-0.46	0.22	0.45

These relatively high values can be supported through the plots below. For each relationship, a bar plot is used to count the number of 'P' and 'N' for each feature value. This can be seen in the following :

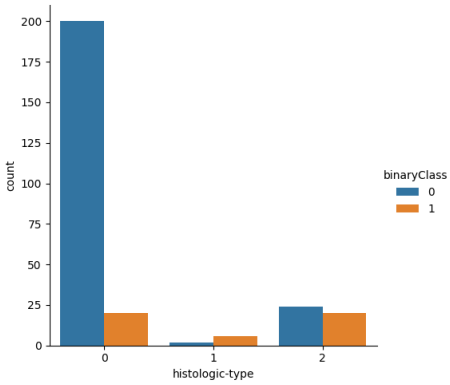


Figure 1: Histologic-Type Correlation

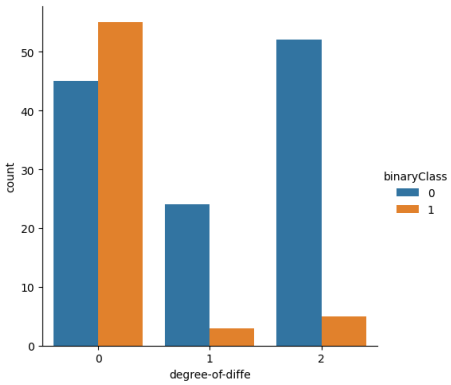


Figure 2: Degree-of-Diffe Correlation

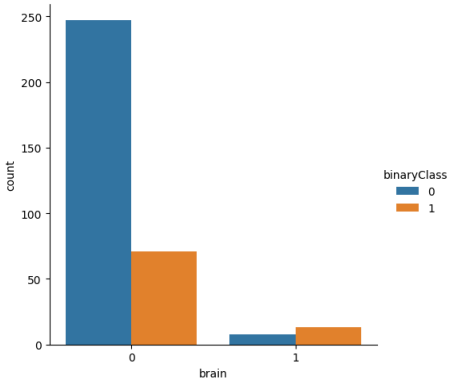


Figure 3: Brain Correlation

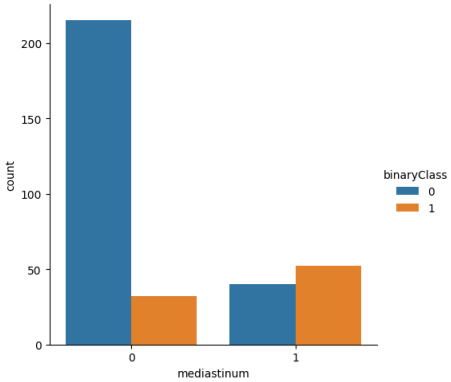


Figure 4: Mediastinum Correlation

Lastly, with the "Red Wine Quality" Dataset, we can also observe the correlation between each feature and the target feature. From what is observed, the features that offer the highest "Pearson Correlation" values are the following :

Volatile Acidity	Citric Acid	Sulphates	Alcohol
-0.39	0.23	0.25	0.48

These relatively high values can be supported through the plots below. For each relationship, a box plot is used to summarize the data corresponding to each target feature value. This can be seen in the following :

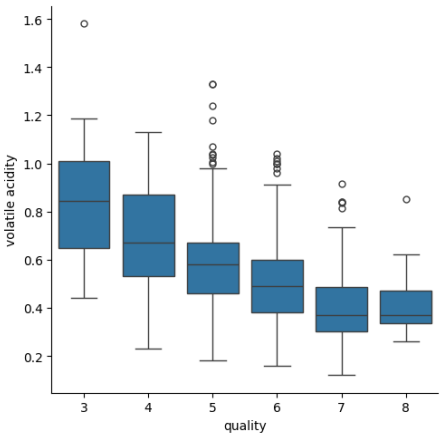


Figure 5: Volatile Acidity Correlation

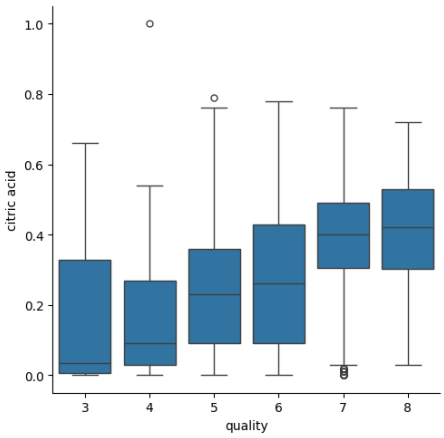


Figure 6: Citric Acid Correlation

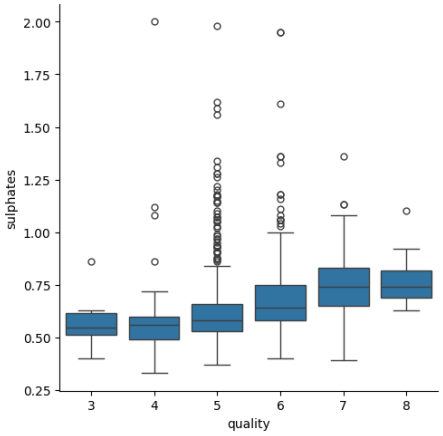


Figure 7: Sulphates Correlation

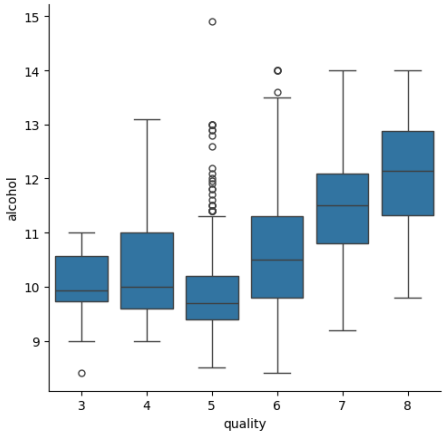


Figure 8: Alcohol Correlation

3 Experimental Set-up

3.1 Creation of Pipelines

As there are a wide variety of options for each pipeline step, I found out that I had to manually create and initialize the different combination of pipeline step options. This need arises for the later phase of hyper-parameter optimization; the Sklearn hyper-parameter optimization classes are not able to map the different hyper-parameters to their respective pipeline step option; as a result, errors are thrown.

Since manually creating the different pipeline combinations is time-consuming, I created a function that allowed me to generate all possible pipelines from different pipeline steps. In short, this function creates a power set, of different pipelines, when given the pipeline steps and their options. For example, consider a pipeline that contains a scaling step of 2 different scaling methods, and a final model prediction step with 2 different types of models. The resulting number of pipelines will be $(2 \times 2) = 4$.

3.2 The Pipelines and their Steps

Since there are two different datasets, there will also be two different sets of pipelines for each respective dataset; this difference is due to the different characteristics of the datasets. As a note, some steps cannot be turned off such as an imputation or categorical encoding step; the other steps do contain the option to turn off.

3.2.1 Primary Tumor - Pipeline Steps

1. Categorical Encoder
    - Ordinal Encoder
    - One Hot Encoder
2. Imputer
    - Simple Imputer
3. Scaler
    - Robust Scaler
    - Quantile Transformer
    - Normalizer

- None

4. Predictive Model
    - SVC
    - Random Forest

Thus, there will be $(2 \times 1 \times 4 \times 2) = 16$ different pipelines for the "Primary Tumor" dataset.

3.2.2 Red Wine Quality - Pipeline Steps

1. Scaler
    - Robust Scaler
    - Quantile Transformer
    - Normalizer
    - None
2. Feature Selector
    - Select Percentile
    - None
3. Predictive Model
    - SVC
    - Random Forest

Thus, there will be $(4 \times 2 \times 2) = 16$ different pipelines for the "Red Wine Quality" dataset.

3.3 Pipeline Steps - Hyper-parameters

A key part of this process is hyper-parameter optimization. Not only do we have hyper-parameters for the machine learning models, but we also have important hyper-parameters for our pre-processing steps. The following tables describe the hyper-parameter spaces for each of the options in each pipeline step :

3.3.1 Categorical Encoders

Table 1: Ordinal Encoder

None

Table 2: One Hot Encoder

Hyper-Parameter	Space Type	Range
Handle Unknown	Categorical	ignore or infrequent_if_exist

3.3.2 Imputers

Table 3: Simple Imputer

Hyper-Parameter	Space Type	Range
Strategy	Categorical	Mean, Median, or Most Frequent

3.3.3 Scalers

Table 4: Robust Scaler

None

Table 5: Quantile Transformer

Hyper-Parameter	Space Type	Range
n_quantiles	Integer	[1, 1000]
Output Distribution	Categorical	Uniform or Normal

Table 6: Normalizer

Hyper-Parameter	Space Type	Range
Norm	Categorical	l1, l2, or max

3.3.4 Feature Selectors

Table 7: SelectPercentile() using "f_classif"

Hyper-Parameter	Space Type	Range
Percentile	Integer	[10, 90]

3.3.5 Models

Table 8: SVM Classifier

Hyper-Parameter	Space Type	Range
C	Float	[0.1, 1000]
Kernel	Categorical	Linear, RBF, or Sigmoid
Degree	Integer	[1, 10000]
Gamma	Categorical	Scale or Auto
Coef0	Float	[0.0, 1.0]
Tol	Float	[0.001, 1.0]

Table 9: Random Forest

Hyper-Parameter	Space Type	Range
N Estimators	Integer	[50, 1000]
Criterion	Categorical	Gini, Entropy or Log-Loss
Max Depth	Integer	[1, 100]
Min Samples Split	Integer	[2, 100]
Min Samples Leaf	Integer	[1, 100]
Min Weight Fraction Leaf	Float	[0.0, 0.5]
Max Features	Categorical	Sqrt, Log2, or None

3.4 Pipeline Evaluation

In order to fairly evaluate the optimized pipelines, nested resampling is done. The process follows the explanations by Dr. Bernd Bischl in the video titled : "I2ML - Evaluation - Resampling I" and Dr. Sebastian Raschka in the video titled : "11.5 Nested CV for Algorithm Selection (L11 Model Eval. Part 4)".

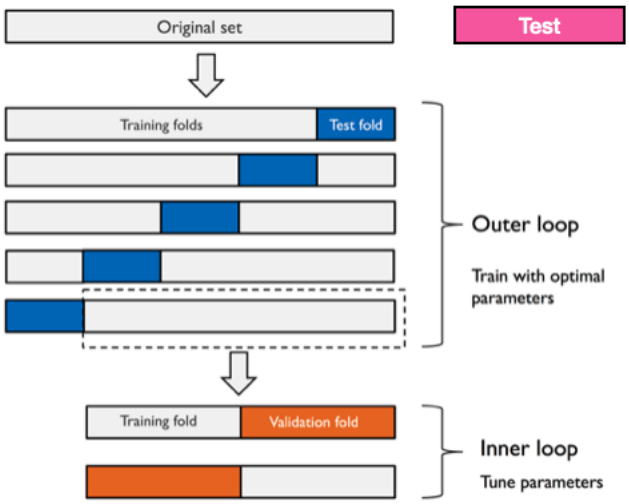


Figure 9: taken from Dr. Sebastian Raschka

In evaluating the optimized pipelines, the whole data set is used, contrary to the diagram above which leaves out a testing data set. The data set is fed into the nested resampling loops; it consists of an outer 5-fold cross validation loop and an inner 2-fold cross validation loop. For each outer loop, the data set is split up into a 4:1 ratio of training and testing samples. For each inner loop, the training set is split into an "inner-loop" training set and "inner-loop" validation set; the inner loop is used for hyper-parameter optimization.

For each of the five outer loop training sets, a hyper-parameter optimization algorithm is run to find an optimal hyper-parameter configuration. The optimization algorithm used is "Bayesian Optimization".

In the case of the "Bayesian Optimization" algorithm, 30 iterations of the algorithm are run; each iteration results in a hyper-parameter configuration that is chosen based off past evaluations (average cross validation weighted-F1 scores from the inner loop folds); once all 30 iterations are run, a single hyper-parameter configuration is chosen from the inner loops.

When a hyper-parameter configuration is returned from the inner loops, the pipeline is configured to the hyper-parameter configuration; then, the pipeline is evaluated on an outer loop testing set; the evaluation result is kept as a single unbiased performance estimate for the single outer loop fold. This process repeats for each of the outer loop folds. At the end, there will be five unbiased performance estimates for each of the five outer loop folds.

3.5 Pipeline Hyper-parameter Selection

Independent of the "Optimized Pipeline Evaluation" process is the hyper-parameter selection process; this process is exclusively focused on the selection of hyper-parameters and is not focused on providing unbiased performance estimates for the optimized pipelines. To select the optimal hyper-parameters for each pipeline, "Bayesian Optimization" is run on the entire data set and a single hyper-parameter configuration is chosen.

In the case of the "Bayesian Optimization" algorithm, 30 iterations of the algorithm are run; each iteration results in a hyper-parameter configuration that is chosen based off past evaluations (average cross validation weighted-F1 scores from the entire data set); once all 30 iterations are finished, a single hyper-parameter configuration is chosen.

3.6 Technologies Used

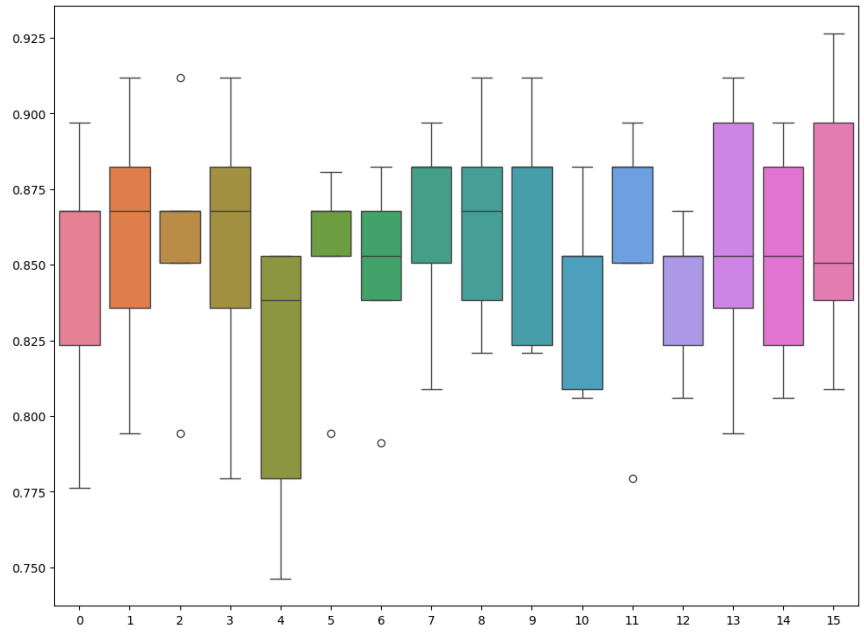
The source code for all the models and the text pre-processing steps was taken from the Sklearn Python library. The source code for the "Bayesian Optimization" algorithm was taken from the "baytune" Python library.

4 Results

4.1 (Nested Resampling) Optimized Performance Estimates

As mentioned before, individual performance estimates (weighted-F1 scores) are stored from each of the five outer loops of the nested resampling evaluation process. For each of the two datasets, the five performance estimates, for each pipeline, are plotted.

4.1.1 Primary Tumor



To help distinguish the different box-plots, the following is a mapping of each of the numbers to their pipelines:

- Pipeline 0
    1. Ordinal Encoder
    2. Simple Imputer
    3. Robust Scaler
    4. SVM Classifier
- Pipeline 1
    1. Ordinal Encoder

2. Simple Imputer
3. Robust Scaler
4. Random Forest Classifier

- Pipeline 2
  1. Ordinal Encoder
  2. Simple Imputer
  3. Quantile Transformer
  4. SVM Classifier
- Pipeline 3
  1. Ordinal Encoder
  2. Simple Imputer
  3. Quantile Transformer
  4. Random Forest Classifier
- Pipeline 4
  1. Ordinal Encoder
  2. Simple Imputer
  3. Normalizer
  4. SVM Classifier
- Pipeline 5
  1. Ordinal Encoder
  2. Simple Imputer
  3. Normalizer
  4. Random Forest Classifier
- Pipeline 6
  1. Ordinal Encoder
  2. Simple Imputer
  3. SVM Classifier
- Pipeline 7
  1. Ordinal Encoder
  2. Simple Imputer
  3. Random Forest Classifier
- Pipeline 8

1. One Hot Encoder
2. Simple Imputer
3. Robust Scaler
4. SVM Classifier

- Pipeline 9
    1. One Hot Encoder
    2. Simple Imputer
    3. Robust Scaler
    4. Random Forest Classifier
- Pipeline 10
    1. One Hot Encoder
    2. Simple Imputer
    3. Quantile Transformer
    4. SVM Classifier
- Pipeline 11
    1. One Hot Encoder
    2. Simple Imputer
    3. Quantile Transformer
    4. Random Forest Classifier
- Pipeline 12
    1. One Hot Encoder
    2. Simple Imputer
    3. Normalizer
    4. SVM Classifier
- Pipeline 13
    1. One Hot Encoder
    2. Simple Imputer
    3. Normalizer
    4. Random Forest Classifier
- Pipeline 14
    1. One Hot Encoder
    2. Simple Imputer

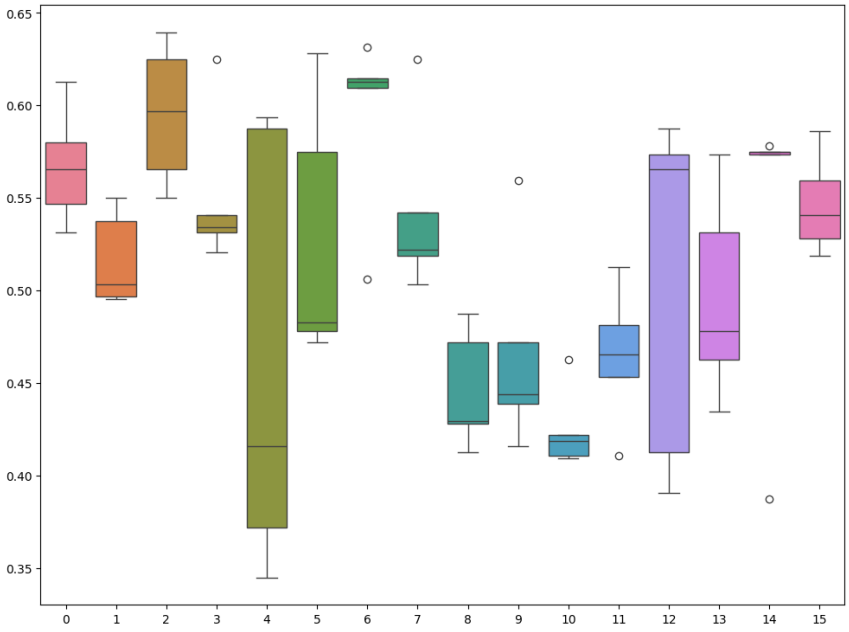
3. SVM Classifier

- Pipeline 15
  1. One Hot Encoder
  2. Simple Imputer
  3. Random Forest Classifier

The statistical measurements from the pipelines' performance estimates are provided in the following tables :

Pipeline	Avg. Weighted-F1 Score	Standard Deviation
0	$\approx 85\%$	≈ 4
1	$\approx 86\%$	≈ 4
2	$\approx 86\%$	≈ 4
3	$\approx 86\%$	≈ 5
4	$\approx 81\%$	≈ 4
5	$\approx 85\%$	≈ 3
6	$\approx 85\%$	≈ 3
7	$\approx 86\%$	≈ 3
8	$\approx 86\%$	≈ 3
9	$\approx 86\%$	≈ 4
10	$\approx 84\%$	≈ 3
11	$\approx 86\%$	≈ 4
12	$\approx 84\%$	≈ 2
13	$\approx 86\%$	≈ 4
14	$\approx 85\%$	≈ 3
15	$\approx 86\%$	≈ 4

4.1.2 Red Wine Quality



To help distinguish the different box-plots, the following is a mapping of each of the numbers to their pipelines:

- Pipeline 0
    1. Robust Scaler
    2. SelectPercentile
    3. SVM Classifier
- Pipeline 1
    1. Robust Scaler
    2. SelectPercentile
    3. Random Forest Classifier
- Pipeline 2
    1. Robust Scaler
    2. SVM Classifier
- Pipeline 3

1. Robust Scaler
2. Random Forest Classifier

- Pipeline 4
  1. Quantile Transformer
  2. SelectPercentile
  3. SVM Classifier
- Pipeline 5
  1. Quantile Transformer
  2. SelectPercentile
  3. Random Forest Classifier
- Pipeline 6
  1. Quantile Transformer
  2. SVM Classifier
- Pipeline 7
  1. Quantile Transformer
  2. Random Forest Classifier
- Pipeline 8
  1. Normalizer
  2. SelectPercentile
  3. SVM Classifier
- Pipeline 9
  1. Normalizer
  2. SelectPercentile
  3. Random Forest Classifier
- Pipeline 10
  1. Normalizer
  2. SVM Classifier
- Pipeline 11
  1. Normalizer
  2. Random Forest Classifier
- Pipeline 12

1. SelectPercentile
2. SVM Classifier

- Pipeline 13
  1. SelectPercentile
  2. Random Forest Classifier
- Pipeline 14
  1. SVM Classifier
- Pipeline 15
  1. Random Forest Classifier

The statistical measurements from the pipelines' performance estimates are provided in the following tables :

Pipeline	Avg. Weighted-F1 Score	Standard Deviation
0	$\approx 57\%$	≈ 3
1	$\approx 52\%$	≈ 2
2	$\approx 60\%$	≈ 3
3	$\approx 55\%$	≈ 4
4	$\approx 46\%$	≈ 11
5	$\approx 53\%$	≈ 6
6	$\approx 59\%$	≈ 4
7	$\approx 54\%$	≈ 4
8	$\approx 45\%$	≈ 3
9	$\approx 47\%$	≈ 5
10	$\approx 42\%$	≈ 2
11	$\approx 46\%$	≈ 3
12	$\approx 51\%$	≈ 9
13	$\approx 50\%$	≈ 5
14	$\approx 54\%$	≈ 8
15	$\approx 55\%$	≈ 2

4.2 Pipeline Hyper-parameter Selection

The following table compares the average 5-fold cross validation weighted-F1 scores for the non-optimized pipelines and the average 5-fold cross validation weighted-F1 scores for the (Bayesian) optimized pipelines.

NOTE : These evaluations can be found in the "cosc5557_pipeline-optimization-two_almountassir.ipynb" file.

4.2.1 Primary Tumor

Table 10: Non-optimized vs. Optimized Pipelines Comparison

Pipeline	Non-optimized Weighted-F1 Score	Optimized Weighted-F1 Score
0	$\approx 80\%$	$\approx 86\%$
1	$\approx 86\%$	$\approx 86\%$
2	$\approx 84\%$	$\approx 86\%$
3	$\approx 85\%$	$\approx 85\%$
4	$\approx 86\%$	$\approx 87\%$
5	$\approx 84\%$	$\approx 85\%$
6	$\approx 85\%$	$\approx 86\%$
7	$\approx 84\%$	$\approx 85\%$
8	$\approx 86\%$	$\approx 88\%$
9	$\approx 84\%$	$\approx 86\%$
10	$\approx 85\%$	$\approx 88\%$
11	$\approx 84\%$	$\approx 86\%$
12	$\approx 85\%$	$\approx 87\%$
13	$\approx 85\%$	$\approx 86\%$
14	$\approx 85\%$	$\approx 88\%$
15	$\approx 86\%$	$\approx 86\%$

To select the optimal hyper-parameter configuration, the pipeline with the highest optimized weighted-F1 score is chosen; here there is a 3-way tie of pipelines with a score of 88%; therefore, we will just choose the 8^{th} pipeline; then we will choose the hyper-parameter configuration obtained by Bayesian Optimization for the specific pipeline.

Table 11: Pipeline 8 Hyper-parameter Configuration

Step	Hyper-parameter	Configuration
One Hot Encoder	handle_unknown	ignore
Simple Imputer	strategy	Mean
SVM Classifier	C	2.85
SVM Classifier	Kernel	RBF
SVM Classifier	Degree	13
SVM Classifier	Gamma	Scale
SVM Classifier	Coef0	0.25
SVM Classifier	Tol	0.17

4.2.2 Red Wine Quality

Table 12: Non-optimized vs. Optimized Pipelines Comparison

Pipeline	Non-optimized Weighted-F1 Score	Optimized Weighted-F1 Score
0	$\approx 34\%$	$\approx 58\%$
1	$\approx 34\%$	$\approx 51\%$
2	$\approx 47\%$	$\approx 61\%$
3	$\approx 55\%$	$\approx 56\%$
4	$\approx 28\%$	$\approx 48\%$
5	$\approx 36\%$	$\approx 55\%$
6	$\approx 47\%$	$\approx 62\%$
7	$\approx 56\%$	$\approx 58\%$
8	$\approx 29\%$	$\approx 46\%$
9	$\approx 37\%$	$\approx 49\%$
10	$\approx 40\%$	$\approx 47\%$
11	$\approx 50\%$	$\approx 51\%$
12	$\approx 34\%$	$\approx 57\%$
13	$\approx 33\%$	$\approx 56\%$
14	$\approx 38\%$	$\approx 47\%$
15	$\approx 56\%$	$\approx 57\%$

To select the optimal hyper-parameter configuration, the pipeline with the highest optimized weighted-F1 score is chosen; therefore, the 6^{th} pipeline; then we will choose the hyper-parameter configuration obtained by Bayesian Optimization for the specific pipeline.

Table 13: Pipeline 6 Hyper-parameter Configuration

Step	Hyper-parameter	Configuration
Quantile Transformer	n_quantile	927
Quantile Transformer	output_distribution	Uniform
SVM Classifier	C	26.84
SVM Classifier	Kernel	RBF
SVM Classifier	Degree	49
SVM Classifier	Gamma	Scale
SVM Classifier	Coef0	0.08
SVM Classifier	Tol	0.72