MINISTRY OF SCIENCE AND TECHNOLOGY

POSTS AND TELECOMMUNICATIONS INSTITUTE OF TECHNOLOGY
HO CHI MINH CITY CAMPUS

----

ptit.edu.vn

FINAL PROJECT REPORT
MACHINE LEARNING

Project: Indentifying key features in determining wine quality

Instructor: PhD. Nguyen Hong Son
Group Members:
Kiều Tuấn Trung Anh – N21DCCN004
Phạm Trần Tất Thành – N21DCCN176

Class: E21CQCNTT01-N
School year: 2021-2026
Major: Artificial Intelligence

Ho Chi Minh City, June 2025

Contents

WORK :

No.	Task	Person in Charge
1	Data cleaning	Kiều Tuấn Trung Anh
2	Data exploration	Kiều Tuấn Trung Anh
3	Feature engineering	Kiều Tuấn Trung Anh
4	Modeling	Phạm Trần Tất Thành
5	Evaluation	Phạm Trần Tất Thành

CHAPTER 1: INTRODUCTION

- Once viewed as a luxury good, nowadays wine is increasingly enjoyed by a wider range of consumers. Portugal is a top ten wine exporting country with 3.17% of the market share in 2005 . Exports of its vinho verde wine (from the northwest region) have increased by 36% from 1997 to 2007. To support its growth, the wine industry is investing in new technologies for both wine making and selling processes. Wine certification and quality assessment are key elements within this context. Certification prevents the illegal adulteration of wines (to safeguard human health) and assures quality for the wine market. Quality evaluation is often part of the certification process and can be used to improve wine making (by identifying the most influential factors) and to stratify wines such as premium brands (useful for setting prices).
- Advances in information technologies have made it possible to collect, store and process massive, often highly complex datasets. All this data hold valuable information such as trends and patterns, which can be used to improve decision making and optimize chances of success.
- We present a case study for modeling wine quality based on analytical data that are easily available at the wine certification step. Building such model is valuable not only for certification entities but also wine producers and even consumers. It can be used to support the oenologist wine evaluations, potentially improving the quality and speed of their decisions. Moreover, measuring the impact of the physicochemical tests in the final wine quality is useful for improving the production process. Furthermore, it can help in target marketing by applying similar techniques to model the consumers preferences of niche and/or profitable markets.

CHAPTER 2: DATASET OVERVIEW

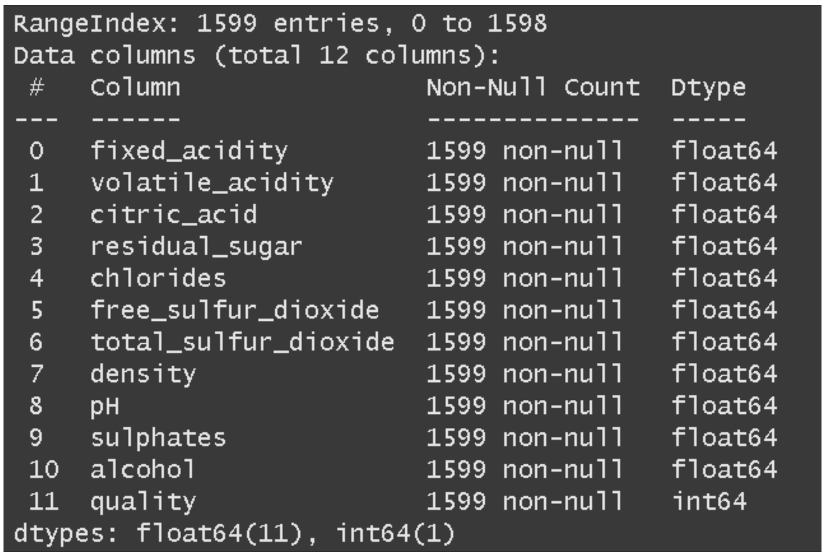
- The datasets are related to red and white variants of the Portuguese "Vinho Verde" wine.
- Due to privacy and logistic issues, only physicochemical (inputs) and sensory (the output) variables are available (e.g. there is no data about grape types, wine brand, wine selling price, etc.).
- These datasets can be viewed as classification or regression tasks. The classes are ordered and not balanced (e.g. there are much more normal wines than excellent or poor ones).
- Feature description:
    - **Fixed acidity**: most acids involved with wine or fixed or nonvolatile (do not evaporate readily)
    - **Volatile acidity**: the amount of acetic acid in wine, which at too high of levels can lead to an unpleasant, vinegar taste
    - **Citric acid**: found in small quantities, citric acid can add 'freshness' and flavor to wines
    - **Residual sugar**: the amount of sugar remaining after fermentation stops, it's rare to find wines with less than 1 gram/liter and wines with greater than 45 grams/liter are considered sweet
    - **Chlorides**: the amount of salt in the wine
    - **Free sulfur dioxide**: the free form of SO2 exists in equilibrium between molecular SO2 (as a dissolved gas) and bisulfite ion, it prevents microbial growth (killing bacterial) and the oxidation of wine (which make wine taste stale-'flat'), it is the portion that is actively protect the wine. Too much -> make wine smell like chemical, too little -> the wine is vunerable to compromise
    - **Total sulfur dioxide**: amount of free forms of S02 (still active) and bound forms of S02 (no longer active - already reacted with other stuff-sugar, acid,...); in low concentrations, SO2 is mostly undetectable in wine, but at free SO2 concentrations over 50 ppm, SO2 becomes evident in the nose and taste of wine
    - **Density**: the density of water is close to that of water depending on the percent alcohol and sugar content
    - **pH**: describes how acidic or basic a wine is on a scale from 0 (very acidic) to 14 (very basic); most wines are between 3-4 on the pH scale
    - **Sulphates**: a wine additive which can contribute to sulfur dioxide gas (S02) levels, wich acts as an antimicrobial and antioxidant
    - **Alcohol**: the percent alcohol content of the wine.
    - **Quality**: wine quality (score between 0 and 10); although there're only between 3 and 8 in the provided dataset.

CHAPTER 3: METHODOLOGY

1. Cleaning data:

1.1. Check data consistency:

- Here are the result that we obtained:

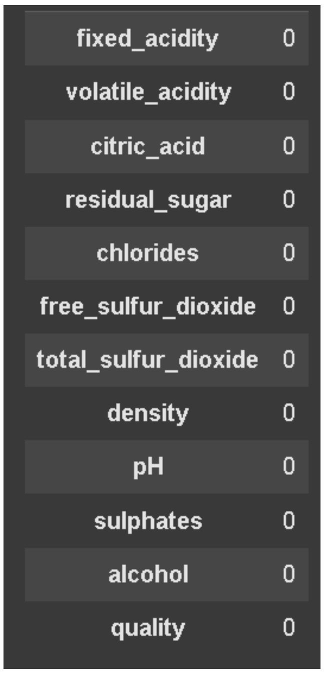


```
RangeIndex: 1599 entries, 0 to 1598
Data columns (total 12 columns):
 #   Column                Non-Null Count  Dtype  
---  ------                --------------  -----  
 0   fixed_acidity         1599 non-null   float64
 1   volatile_acidity      1599 non-null   float64
 2   citric_acid           1599 non-null   float64
 3   residual_sugar        1599 non-null   float64
 4   chlorides             1599 non-null   float64
 5   free_sulfur_dioxide   1599 non-null   float64
 6   total_sulfur_dioxide  1599 non-null   float64
 7   density               1599 non-null   float64
 8   pH                    1599 non-null   float64
 9   sulphates             1599 non-null   float64
 10  alcohol               1599 non-null   float64
 11  quality               1599 non-null   int64  
dtypes: float64(11), int64(1)
```

- All feature and target has the same numeric type.
- But note we only have 1599 rows to work with, which is too few.

1.2. Handle missing value:

- Here are the result we obtained after running a check:

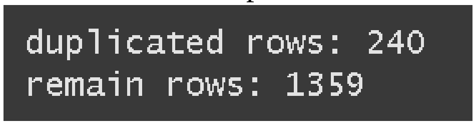


```
     fixed_acidity  0
  volatile_acidity  0
       citric_acid  0
    residual_sugar  0
         chlorides  0
free_sulfur_dioxide  0
total_sulfur_dioxide 0
           density  0
                pH  0
         sulphates  0
           alcohol  0
           quality  0
```

- And luckily, we have no missing value here.

1.3. Remove duplicated:

- We checked for duplicated and here was the result before and after we removed the dups:



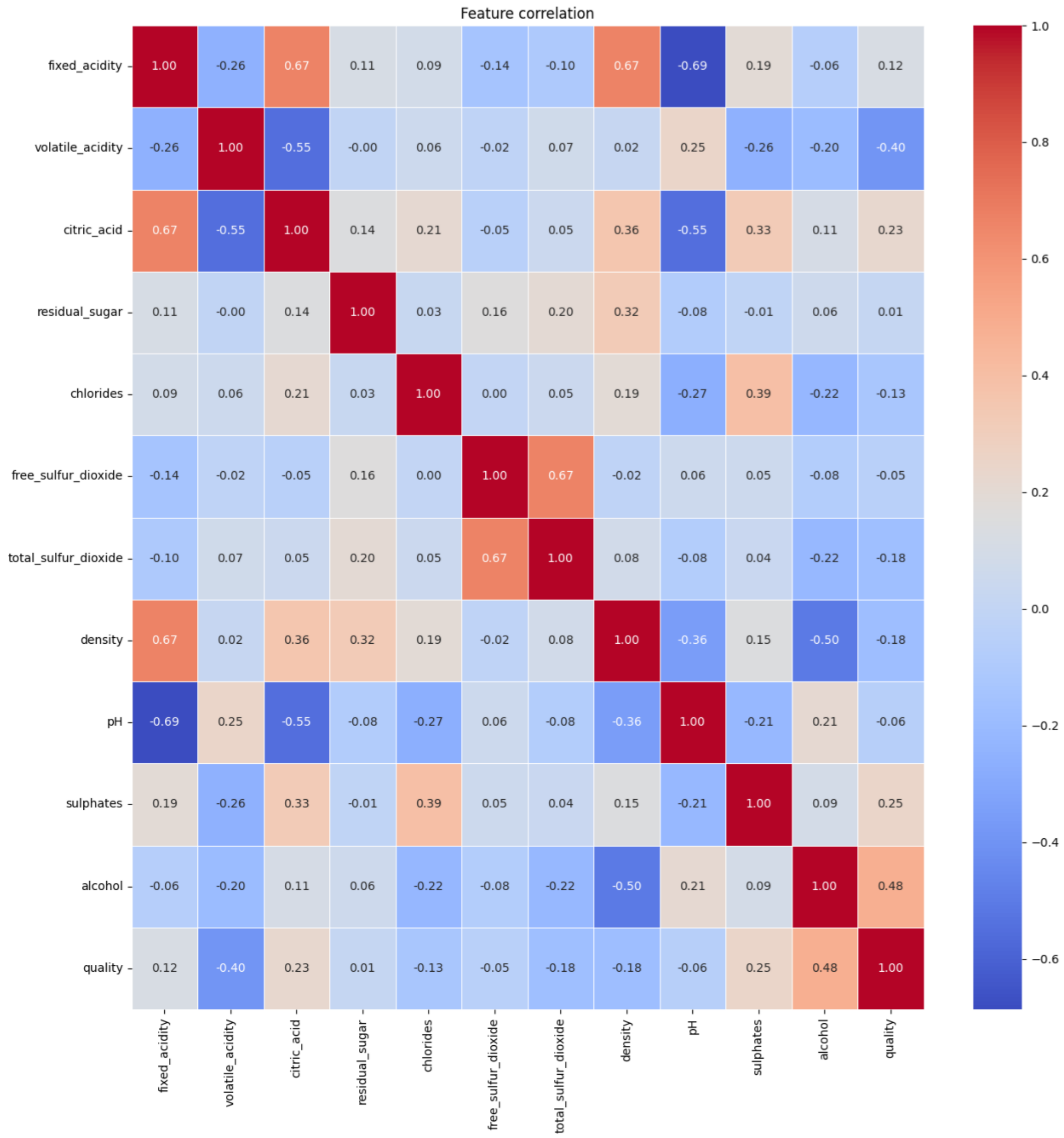
```
duplicated rows: 240
remain rows: 1359
```

- We got 85% of the original datasets left.

2. Data exploring:

2.1. Correlation matrix:

- We will check peason correlation heat map, strongly correlated features can be redundancy affecing model performance.

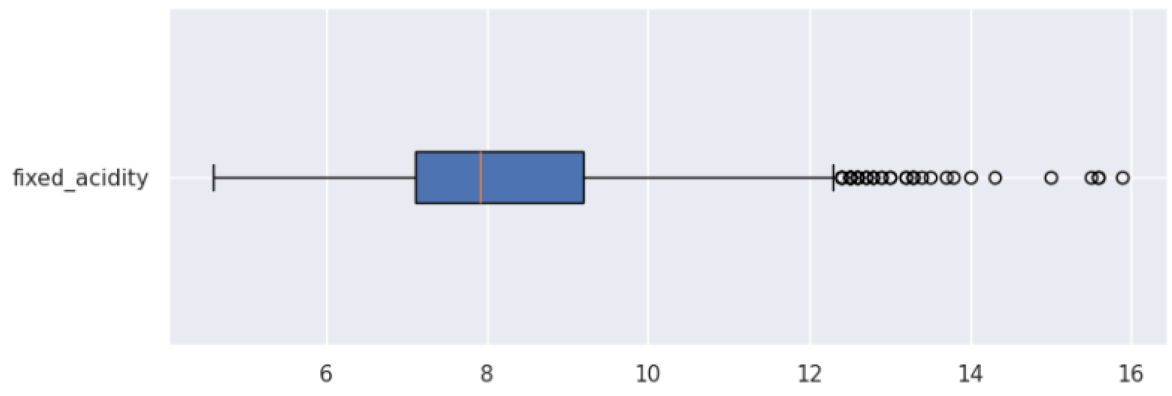


	fixed_acidity	volatile_acidity	citric_acid	residual_sugar	chlorides	free_sulfur_dioxide	total_sulfur_dioxide	density	pH	sulphates	alcohol	quality
fixed_acidity	1.00	-0.26	0.67	0.11	0.09	-0.14	-0.10	0.67	-0.69	0.19	-0.06	0.12
volatile_acidity	-0.26	1.00	-0.55	-0.00	0.06	-0.02	0.07	0.02	0.25	-0.26	-0.20	-0.40
citric_acid	0.67	-0.55	1.00	0.14	0.21	-0.05	0.05	0.36	-0.55	0.33	0.11	0.23
residual_sugar	0.11	-0.00	0.14	1.00	0.03	0.16	0.20	0.32	-0.08	-0.01	0.06	0.01
chlorides	0.09	0.06	0.21	0.03	1.00	0.00	0.05	0.19	-0.27	0.39	-0.22	-0.13
free_sulfur_dioxide	-0.14	-0.02	-0.05	0.16	0.00	1.00	0.67	-0.02	0.06	0.05	-0.08	-0.05
total_sulfur_dioxide	-0.10	0.07	0.05	0.20	0.05	0.67	1.00	0.08	-0.08	0.04	-0.22	-0.18
density	0.67	0.02	0.36	0.32	0.19	-0.02	0.08	1.00	-0.36	0.15	-0.50	-0.18
pH	-0.69	0.25	-0.55	-0.08	-0.27	0.06	-0.08	-0.36	1.00	-0.21	0.21	-0.06
sulphates	0.19	-0.26	0.33	-0.01	0.39	0.05	0.04	0.15	-0.21	1.00	0.09	0.25
alcohol	-0.06	-0.20	0.11	0.06	-0.22	-0.08	-0.22	-0.50	0.21	0.09	1.00	0.48
quality	0.12	-0.40	0.23	0.01	-0.13	-0.05	-0.18	-0.18	-0.06	0.25	0.48	1.00

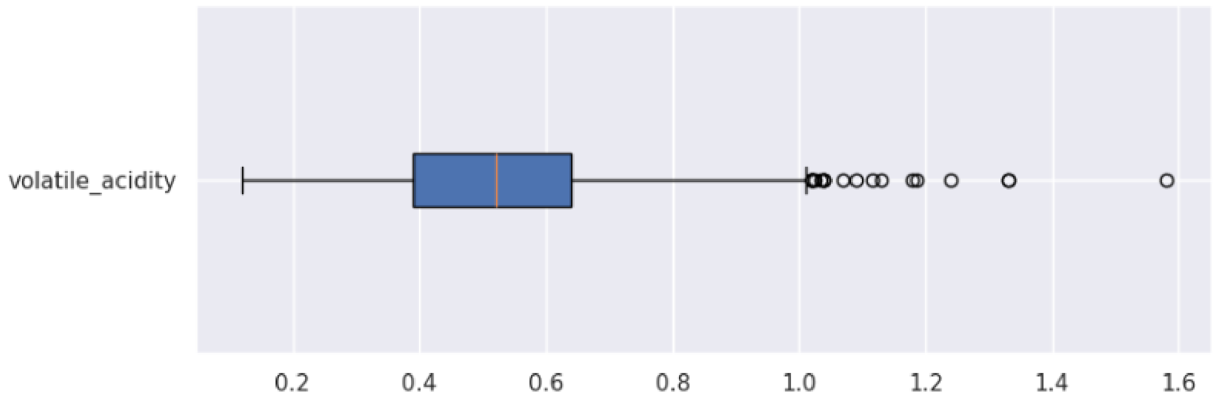
- Based on this, we can see that *(pH and fixed acidity)*, *(density and fixed acidity)*, *(citric acid and fixed acidity)* have a pretty strong correlated but not enough to be considered redundancy.

2.2. Features distribution:

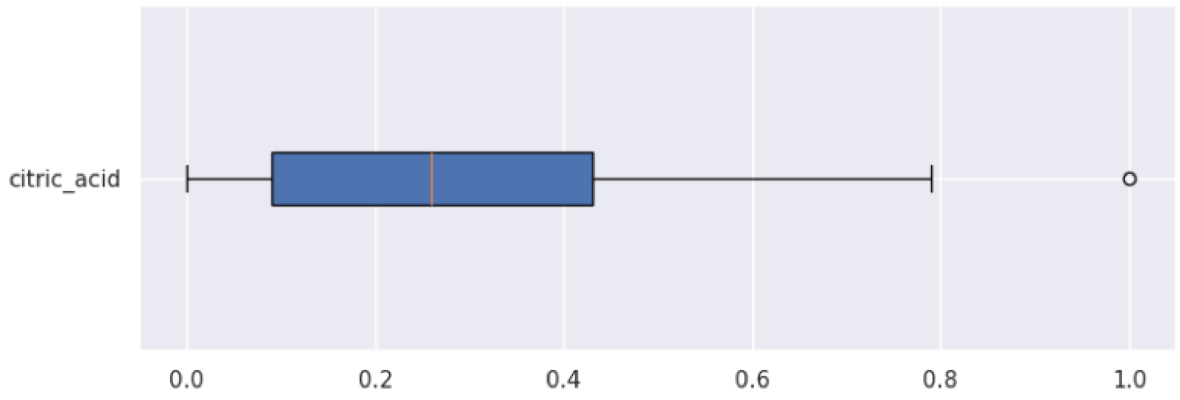
- Next we will find distribution of each features and check for any outliers.



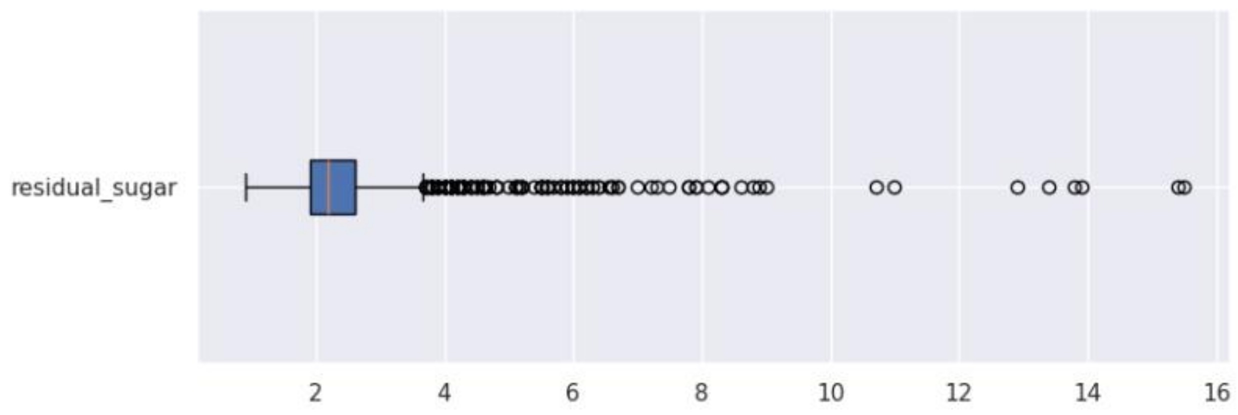
```
quatile_min: 3.95, q25: 7.1, q50: 7.9, q75: 9.2, quatile_max: 12.349999999999998
outliers: 41
```



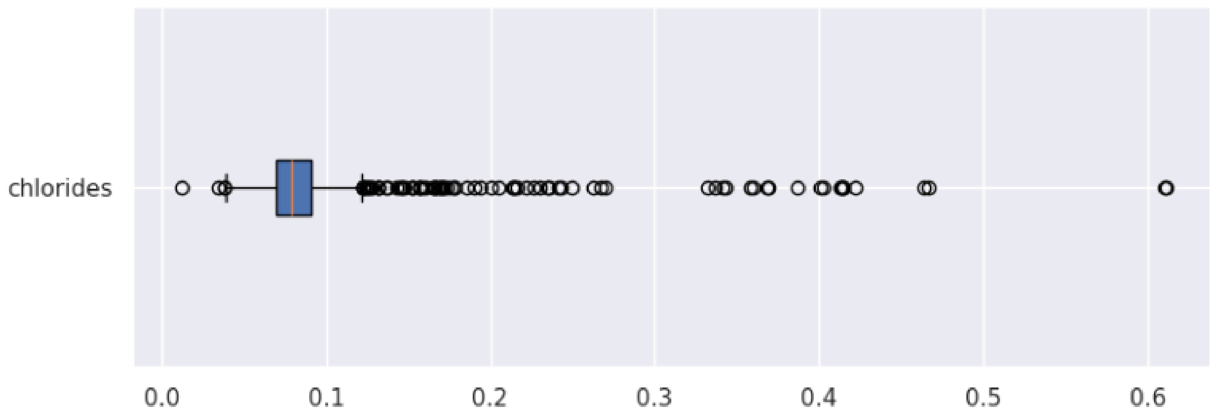
```
quatile_min: 0.015000000000000013, q25: 0.39, q50: 0.52, q75: 0.64, quatile_max: 1.0150000000000001
outliers: 19
```



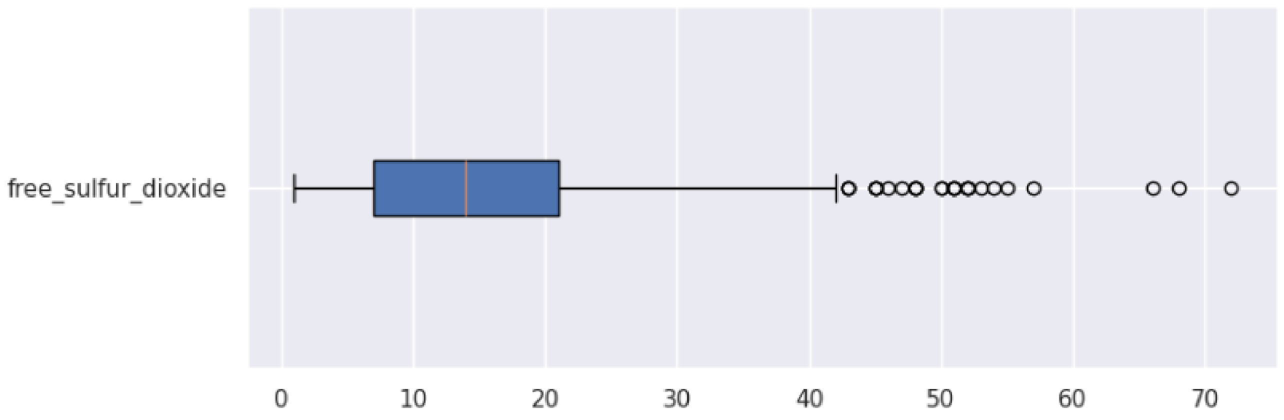
```
quatile_min: -0.42000000000000004, q25: 0.09, q50: 0.26, q75: 0.43, quatile_max: 0.94
outliers: 1
```



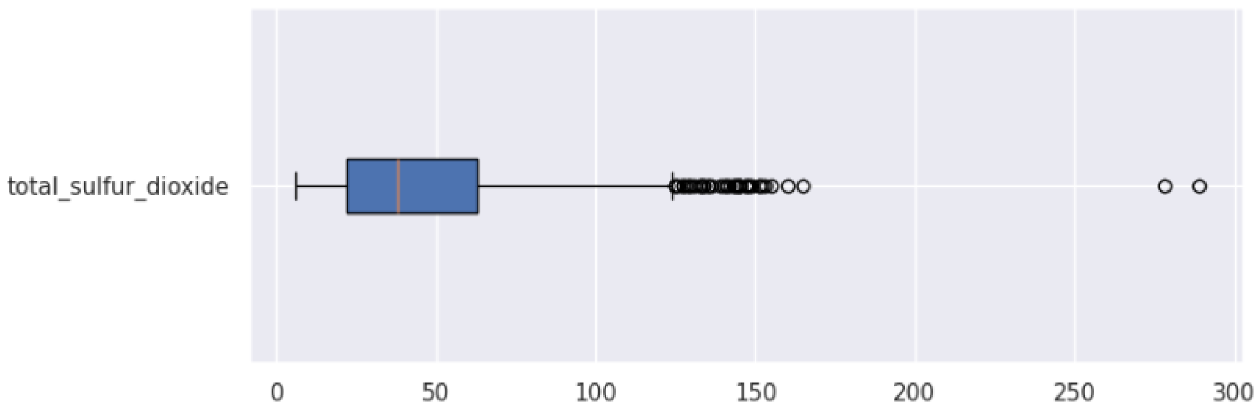
```
quatile_min: 0.8499999999999996, q25: 1.9, q50: 2.2, q75: 2.6, quatile_max: 3.6500000000000004
outliers: 126
```



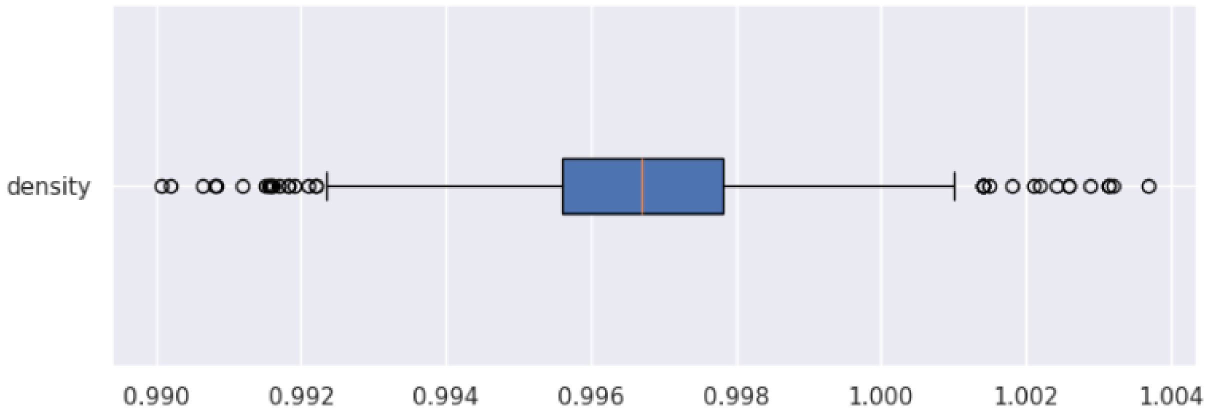
```
quatile_min: 0.03850000000000002, q25: 0.07, q50: 0.079, q75: 0.091, quatile_max: 0.12249999999999998
outliers: 87
```



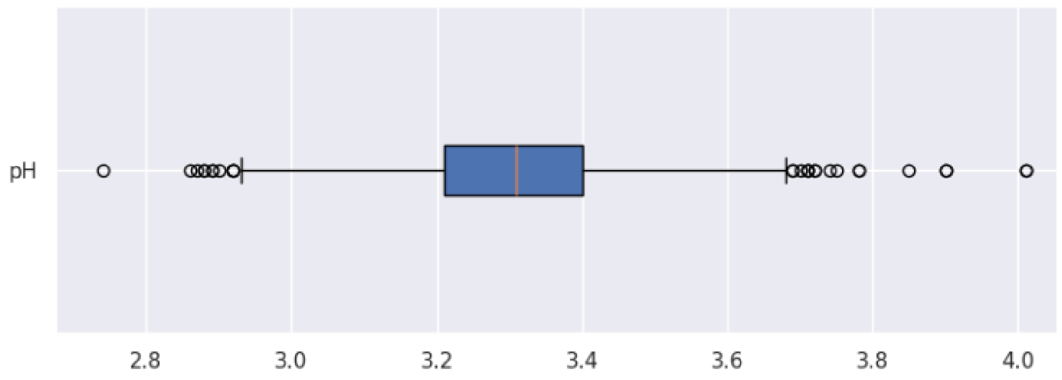
```
quatile_min: -14.0, q25: 7.0, q50: 14.0, q75: 21.0, quatile_max: 42.0
outliers: 26
```



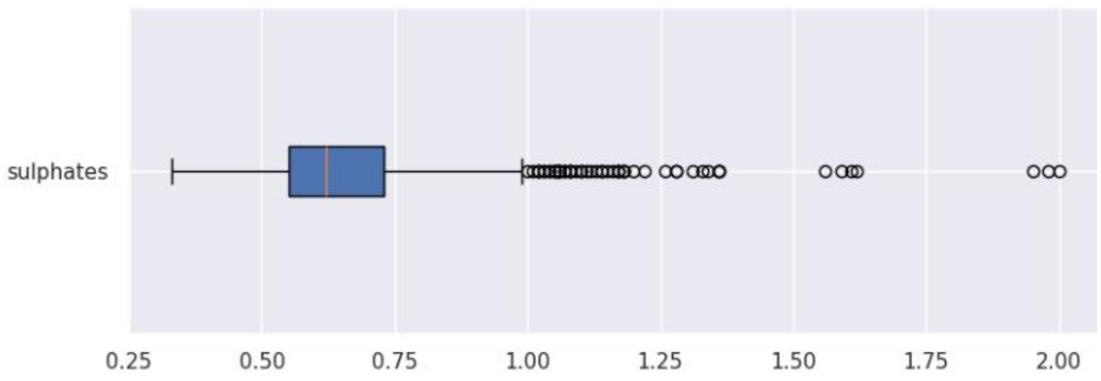
```
quatile_min: -39.5, q25: 22.0, q50: 38.0, q75: 63.0, quatile_max: 124.5
outliers: 45
```



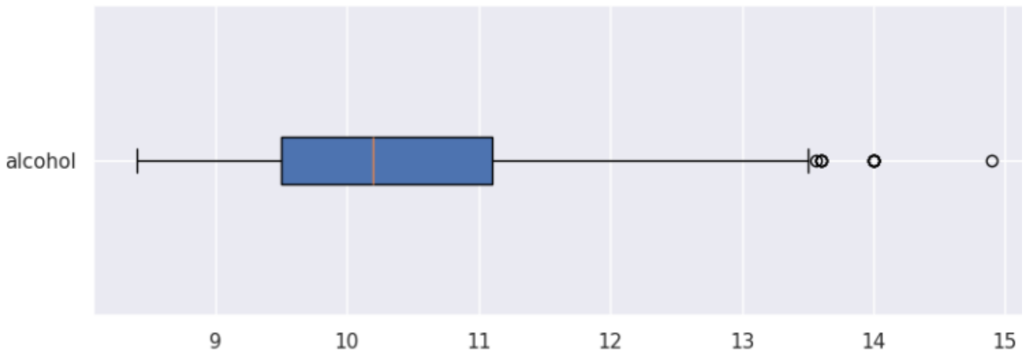
```
quatile_min: 0.99227, q25: 0.9956, q50: 0.9967, q75: 0.99782, quatile_max: 1.00115
outliers: 35
```



```
quatile_min: 2.925, q25: 3.21, q50: 3.31, q75: 3.4, quatile_max: 3.6849999999999996
outliers: 28
```

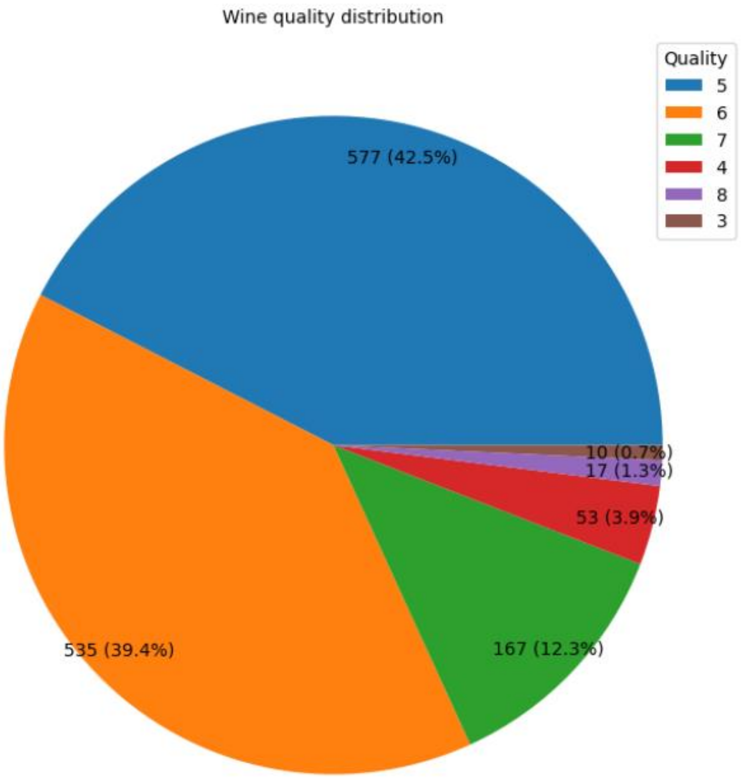


```
quatile_min: 0.28000000000000014, q25: 0.55, q50: 0.62, q75: 0.73, quatile_max: 0.9999999999999999
outliers: 55
```

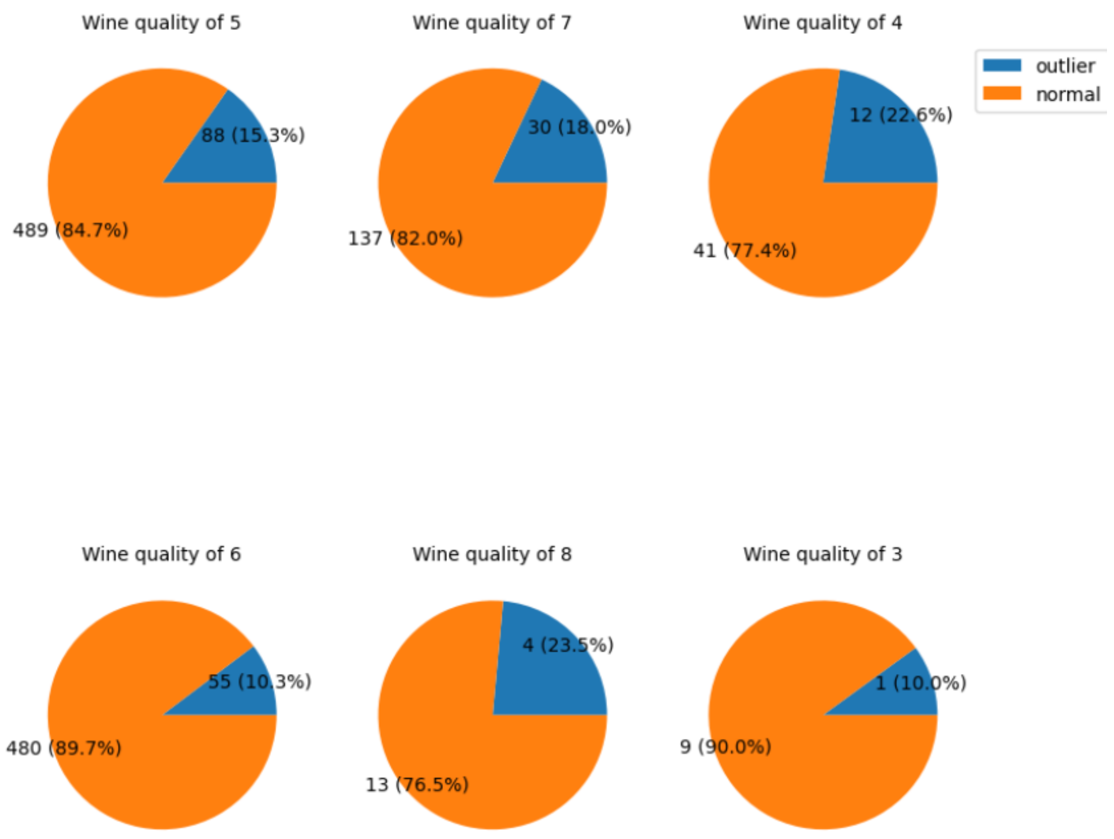


```
quatile_min: 7.1000000000000005, q25: 9.5, q50: 10.2, q75: 11.1, quatile_max: 13.5
outliers: 12
```

- All most all of them have outliers and most of them do not follow normal distribution (except **density** feature).
- Next, we check on the distribution of the target and the outliers laid within them.



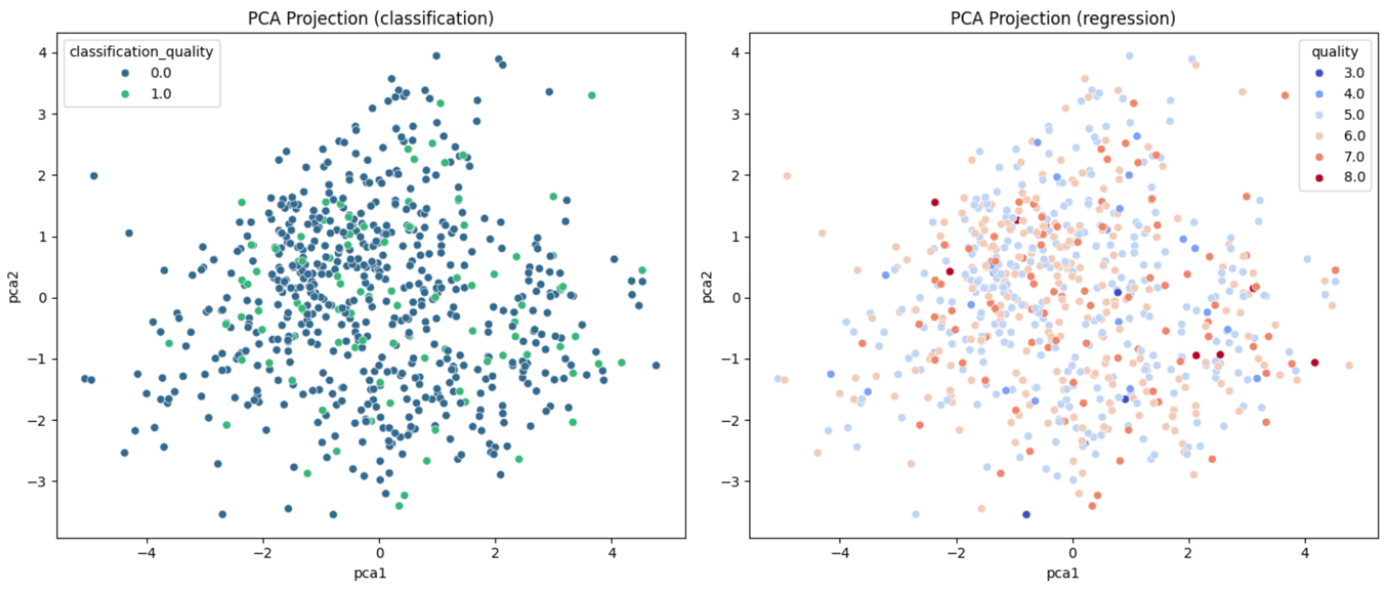
- The classes are all imbalance with quality wine rank 3, 8, 4, 7 are the lowest.



- Within each class, the outliers are never exceeded 25% of the class samples.

2.3. Detect clustering:

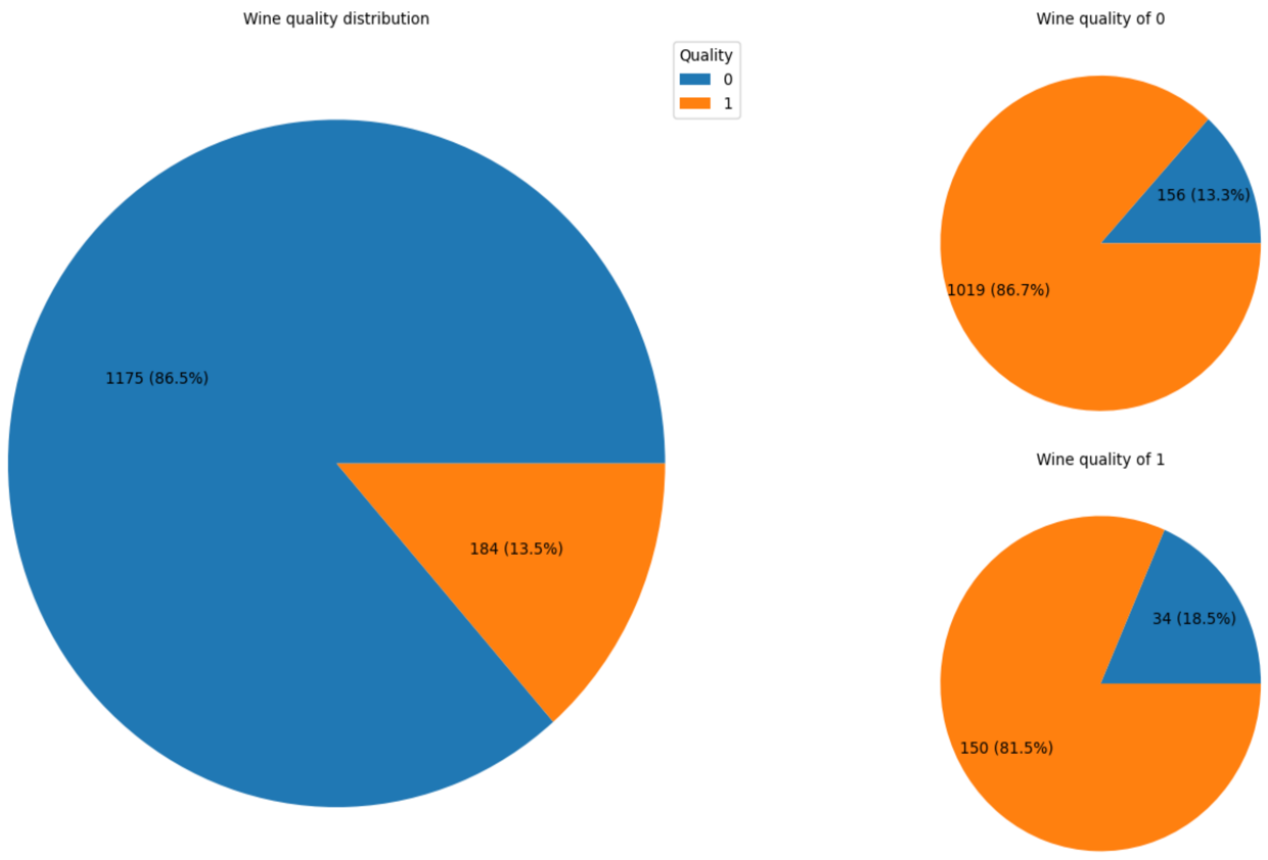
- Here we utilize pca to detect any cluster of each classes, both for regression and classification.



- But here, all classes are intertwine, so that mean there are no relationship in term of clustering of each classes.

CHAPTER 4: FEATURE ENGINEERING

- Because of imbalanced classes. The strategy will be combined classes into 2 bins:
    - bin (0 = Bad) contains class (3, 4, 5, 6)
    - bin (1 = good) contains class (7, 8)
- Note that, we had to keep the original target, for the reason we will explain below.
- Then we check for target distribution again and their outliers.

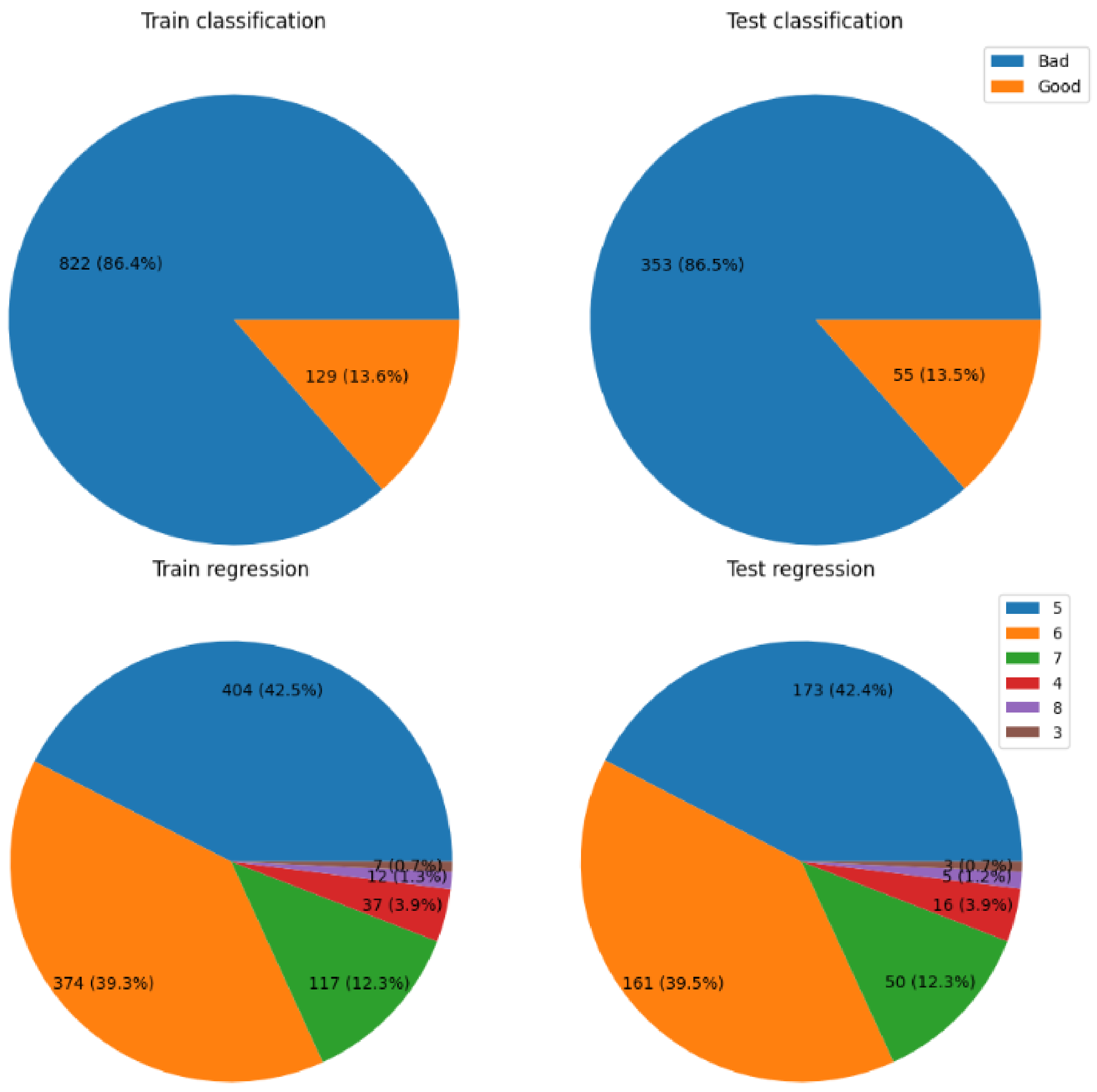


- We can see that the classes were now a bit less imbalance bescause of the good quality being the least.
- And outliers within classes were still not exceed 25% of sample classes.

CHAPTER 5:   PICK IMPORTANT FEATURES

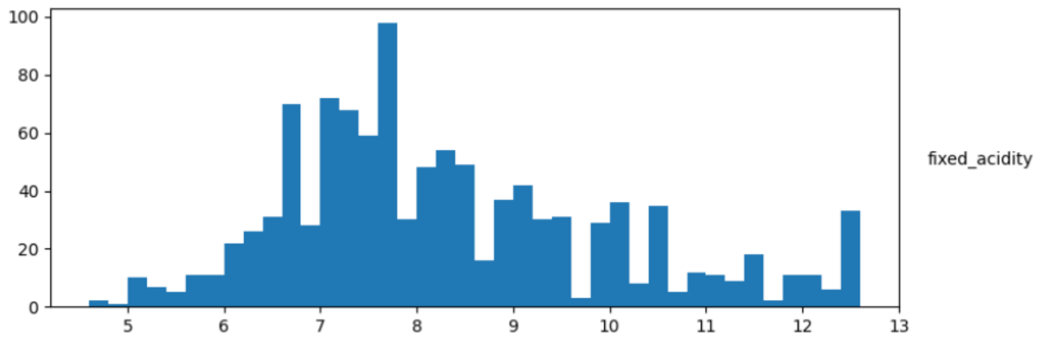
1. Spliting data:

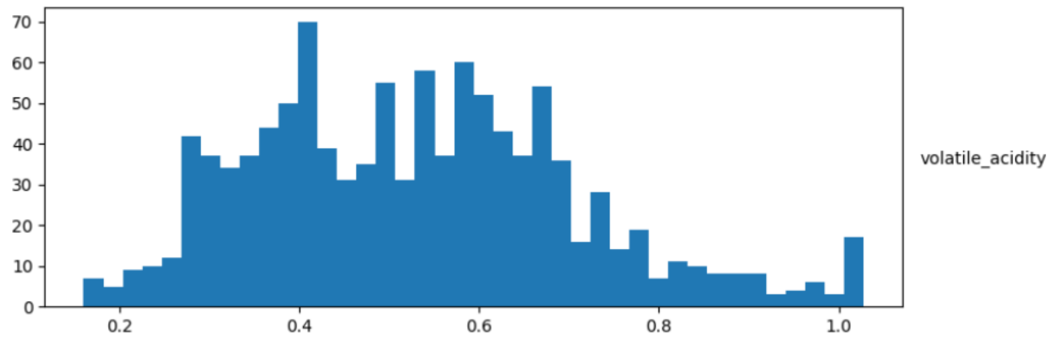
- We then proceeded to split the data into train and test with the ratio of 80:30.
- The catch here was that we required to apply both classification and regresstion model for predicting. So we have to split data in 2 way.
- One split for classification, other for regression.

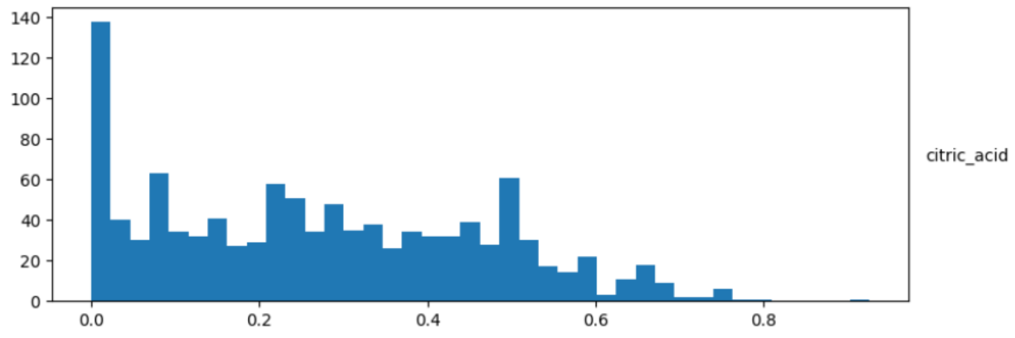


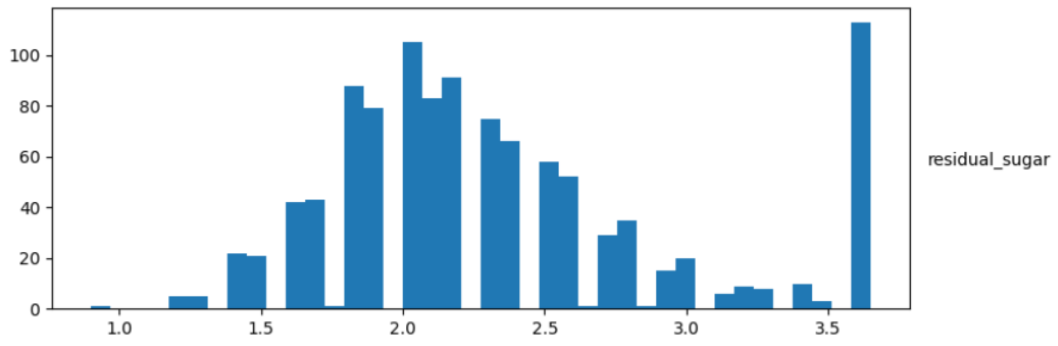
2. Handle outliers:

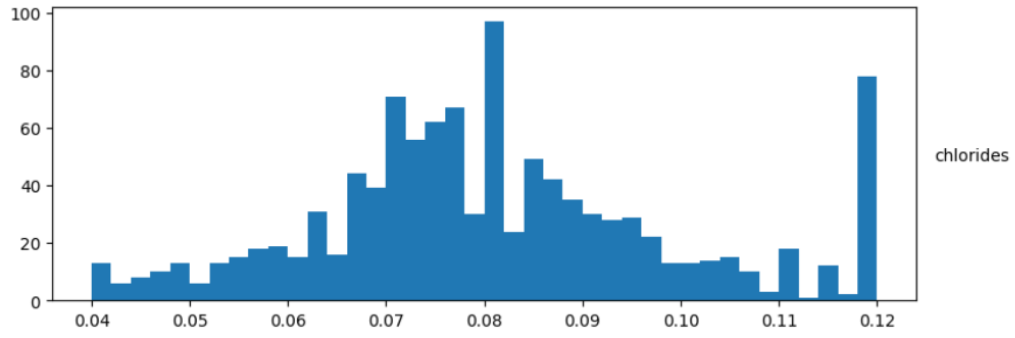
- To handle outliers, we utilize iqr method to capped them to a threshold that within iqr range.
- These are features distribution after getting capped.

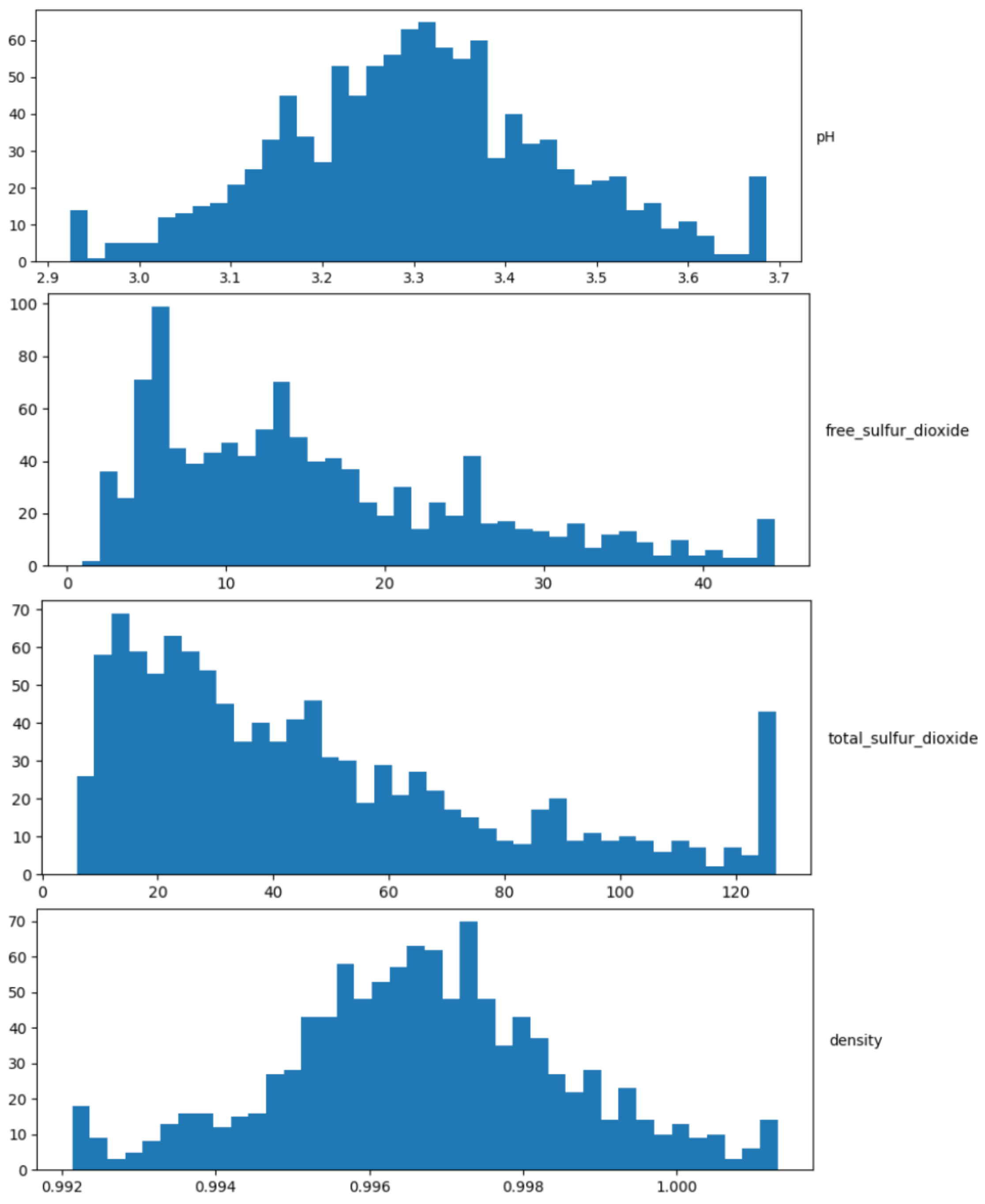

fixed_acidity

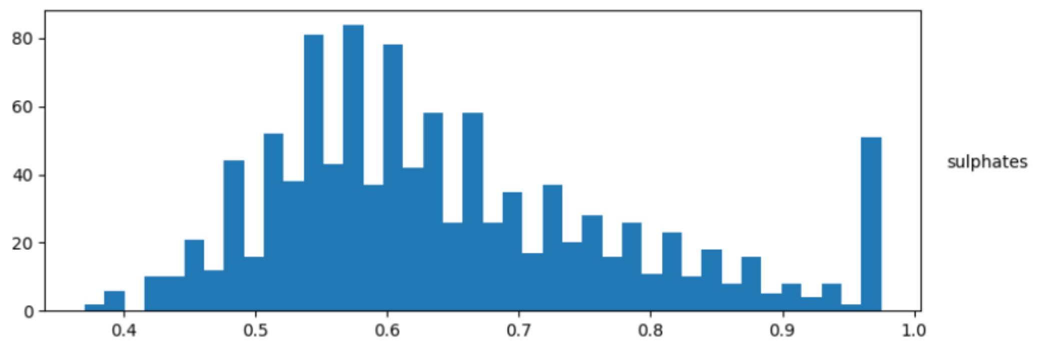

volatile_acidity

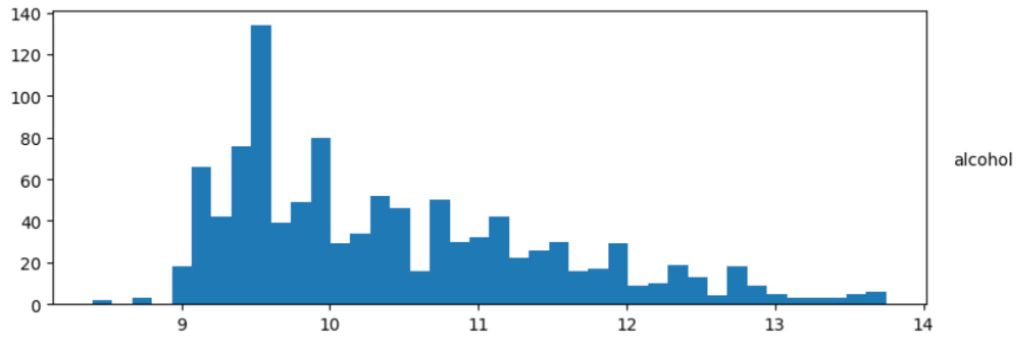

citric_acid


residual_sugar


chlorides

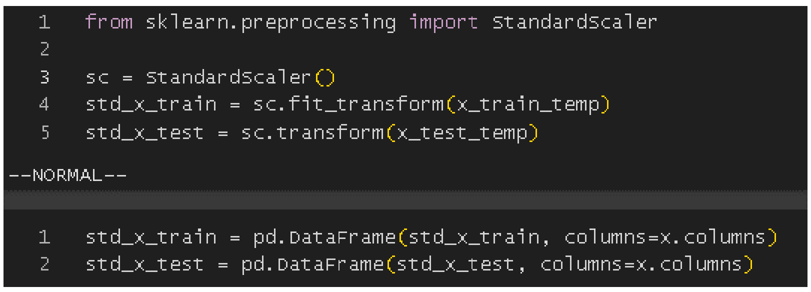

pH
free_sulfur_dioxide
total_sulfur_dioxide
density





3. Feature scaling:

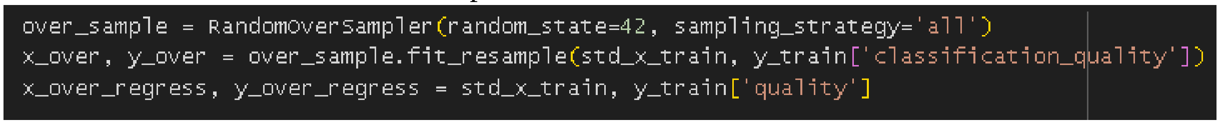
- Some models we used required the features to be scaled. Therefore, we applied StandardScaler to those feature



```
from sklearn.preprocessing import StandardScaler

sc = StandardScaler()
std_x_train = sc.fit_transform(x_train_temp)
std_x_test = sc.transform(x_test_temp)
--NORMAL--

std_x_train = pd.DataFrame(std_x_train, columns=x.columns)
std_x_test = pd.DataFrame(std_x_test, columns=x.columns)
```

4. Resampling:

- We then pursude to apply oversampling method in order to balance out the train set so that we can use rfecv to find the most important set of features that each model used.

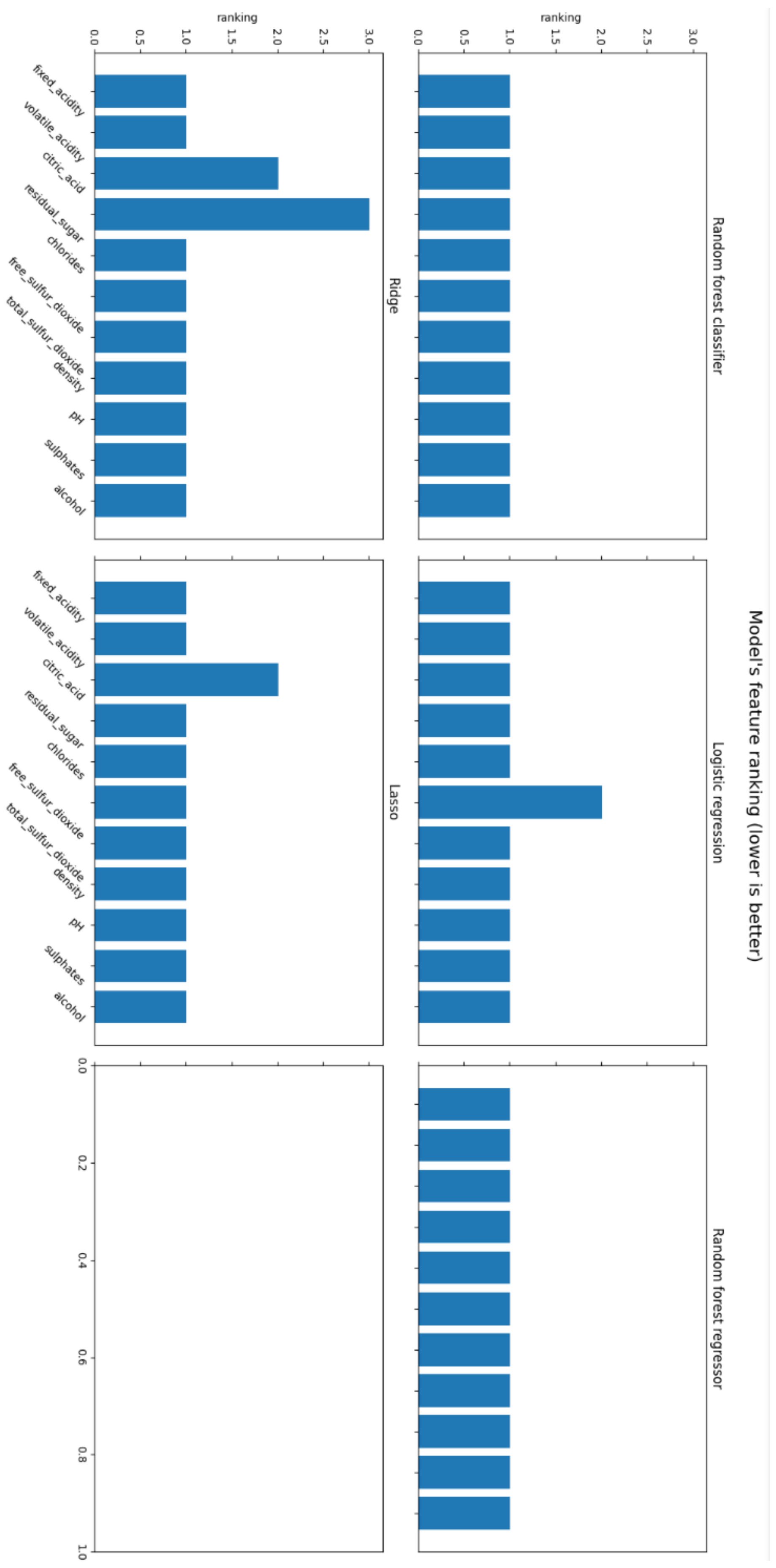


```
over_sample = RandomOverSampler(random_state=42, sampling_strategy='all')
x_over, y_over = over_sample.fit_resample(std_x_train, y_train['classification_quality'])
x_over_regress, y_over_regress = std_x_train, y_train['quality']
```

- Note that we don't apply oversampling to the original target, so it can be use for regression.

5. Recursive feature elimination cross-validation:

- We then apply rfecv to find a set of optimum features for each model we going to use.
- Here are the result:



- The lower the ranking, the more important the feature be with 1 being the most important.
- In modeling section, we going to utilize these features for modeling.

CHAPTER 6: EVALUATION & RESULT

1. Modeling:

- In this project, we explored both regression and classification approaches to predict wine quality based on physicochemical attributes using the Red Wine Quality dataset.

1.1. Regression Tasks:

- To ensure the models were trained effectively for each task, we split the dataset separately for regression and classification.
- For regression, the target variable was the original quality score (an integer from 3 to 8). which is **y_train [‘quality’].**

1.1.1 Linear Regression:

1.1.1.1 Baseline Linear Regression:

- A basic Linear Regression model was trained on the standardized training data. This model served as a baseline for comparison with regularized and ensemble methods. The training process was straightforward, using all available features without regularization.

1.1.1.2 Optimize Linear Regression using Lasso Regression and Gridsearch CV (with Feature Selection):

- To enhance model performance Lasso Regression was trained using a feature-selected subset of the dataset.
- Lasso Regression was applied with hyperparameter tuning using GridSearchCV.

1.1.1.3 Optimize Linear Regression using Ridge Regression and Gridsearch CV (with Feature Selection):

- Like Lasso, Ridge Regression was trained on a feature-selected dataset with hyperparameter tuning via GridSearchCV.
- The model was retrained using the best alpha value found during cross-validation.

1.1.2 Random Forest Regressor:

- To ensure the models were trained effectively for each task, we split the dataset separately for regression and classification:
- For classification, the quality scores were converted into categorical classes (e.g., bad (3,4,5,6),good(7,8) and class imbalance was handled using oversampling techniques.

1.1.2.1 Baseline Random Forest Regressor:

- A Random Forest Regressor was trained on the standardized training dataset. This ensemble method uses multiple decision trees and averages their predictions.

1.1.2.2 Optimize Random Forest Regressor pipeline with GridsearchCV:

- After the initial training, the model was further optimized using GridSearchCV to find the best combination of n_estimators, max_depth, min_samples_split, and max_features.

1.2. Classification Tasks:

- To predict wine quality as a classification problem (i.e., high or low quality), we trained and evaluated two classification models: Random Forest Classifier and Logistic Regression.

Each model was tested in both baseline and optimized versions to assess performance improvements through hyperparameter tuning, feature selection, and oversampling.

1.2.1 Random Forest Classifier:

1.2.1.1 Baseline Model Random Forest Classifier:

- We first trained a baseline RandomForestClassifier using the default parameters and standardized input features without any feature selection or resampling.

1.2.1.2 Optimized Random Forest Classifier using Gridsearch with Oversampling Data:

- Hyperparameter tuning: Conducted a GridSearchCV using 5-fold cross-validation on the training set, optimizing for f1-score.

1.2.2 Logistic Regression:

1.2.2.1 Baseline Model Logistic Regression:

- A logistic regression model was trained using default hyperparameters (max_iter=1000) and standardized features.

1.2.2.2 Optimize Logistic Regression pipeline with GridsearchCV using feature selection and oversampling data:

- We improved the model through:
  - Feature selection
  - Oversampling
  - GridSearchCV: We explored different regularization strategies (L1, L2, ElasticNet), solvers (liblinear, lbfgs, saga), regularization strength (C), and class balancing.

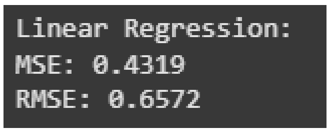
2. Evaluation and Results:

2.1. Regression Tasks:

2.1.1 Linear Regression:

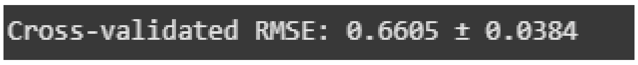
2.1.1.1 RMSE and Cross-validated RMSE for Baseline Linear Regression:

- For the Linear Regression model, we trained the model using standardized features and evaluated its performance on the test set. The results are as follows:



```
Linear Regression:
MSE: 0.4319
RMSE: 0.6572
```

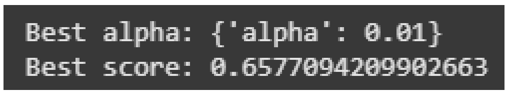
- To ensure the model's performance is consistent and not dependent on the specific train-test split, we performed 5-fold cross-validation. The cross-validated RMSE was:



```
Cross-validated RMSE: 0.6605 ± 0.0384
```

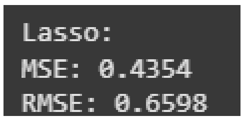
2.1.1.2 RMSE and Cross-validated RMSE for Optimize Linear Regression using Lasso Regression and Gridsearch CV (with Feature Selection).

- We used GridSearchCV to tune the alpha parameter for Lasso regression. The best result was achieved with:



```
Best alpha: {'alpha': 0.01}
Best score: 0.6577094209902663
```

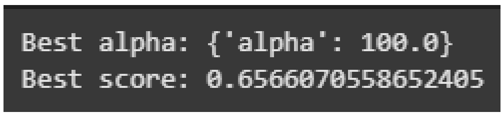
- On the test set, the model performance was:



```
Lasso:
MSE: 0.4354
RMSE: 0.6598
```

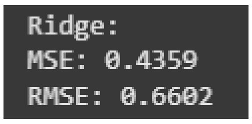
2.1.1.3 RMSE and Cross-validated RMSE for Optimize Linear Regression using Ridge Regression and Gridsearch CV (with Feature Selection).

- Similarly, for Ridge regression, the optimal alpha was found using cross-validation:



```
Best alpha: {'alpha': 100.0}
Best score: 0.6566070558652405
```

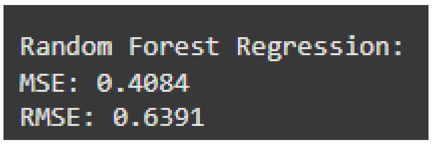
- Model evaluation on the test set gave:



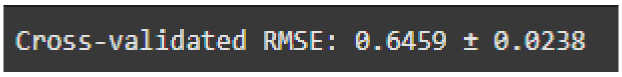
```
Ridge:
MSE: 0.4359
RMSE: 0.6602
```

2.1.2 Random Forest Regressor:

2.1.2.1 RMSE and Cross-validated RMSE for Baseline Random Forest Regressor:

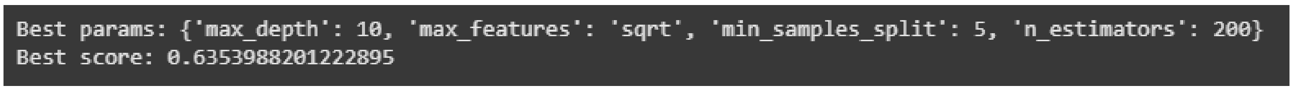


```
Random Forest Regression:
MSE: 0.4084
RMSE: 0.6391
```

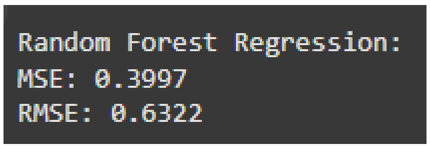


```
Cross-validated RMSE: 0.6459 ± 0.0238
```

2.1.2.2 RMSE and Cross-validated RMSE Optimize Random Forest Regressor pipeline with GridsearchCV:



```
Best params: {'max_depth': 10, 'max_features': 'sqrt', 'min_samples_split': 5, 'n_estimators': 200}
Best score: 0.6353988201222895
```

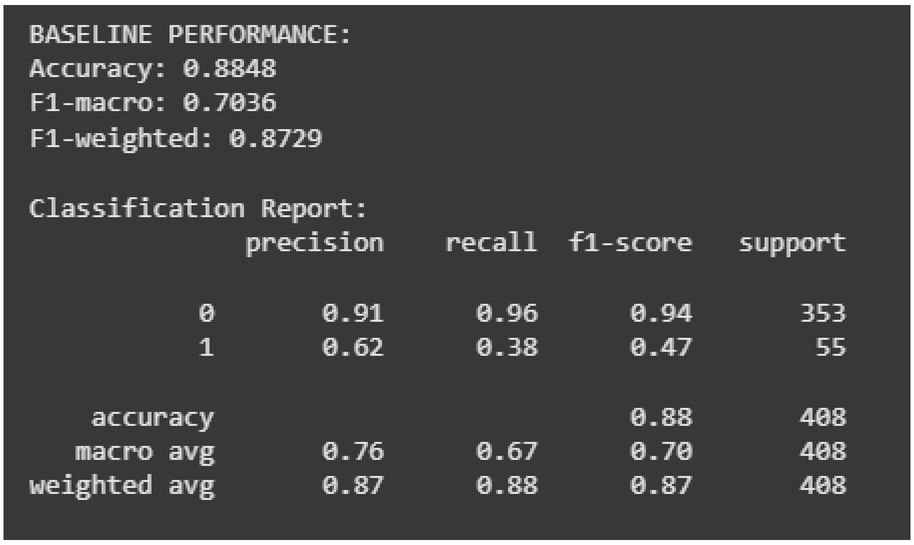


```
Random Forest Regression:
MSE: 0.3997
RMSE: 0.6322
```

2.2. Classification Tasks:

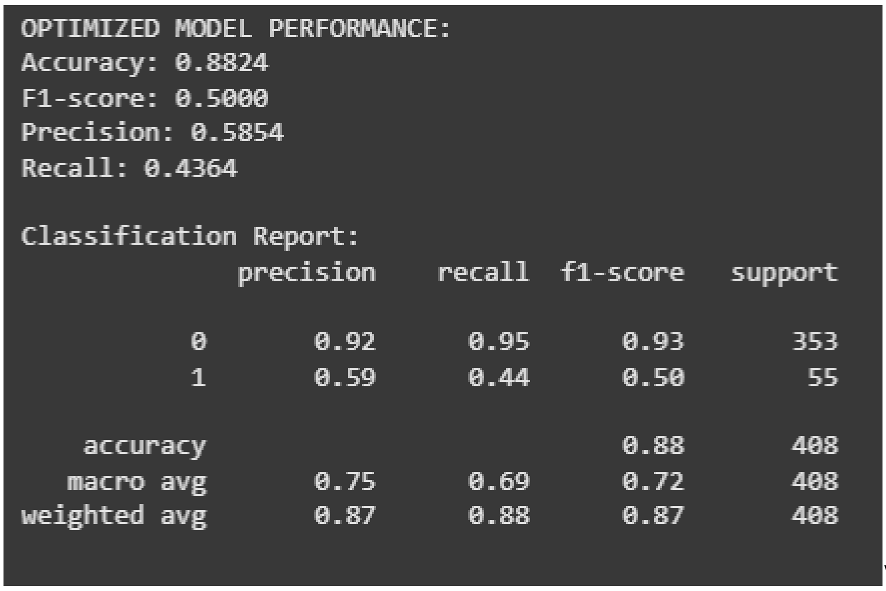
2.2.1 Random Forest Classifier:

2.2.1.1 F1 and accuracy for Baseline Model Random Forest Classifier:



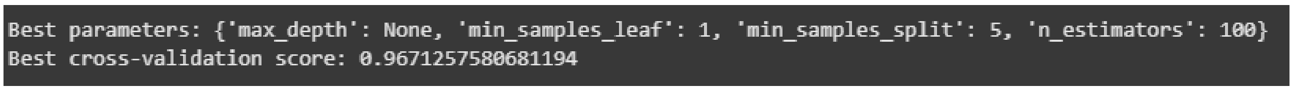
```
BASELINE PERFORMANCE:
Accuracy: 0.8848
F1-macro: 0.7036
F1-weighted: 0.8729

Classification Report:
              precision    recall  f1-score   support

           0       0.91      0.96      0.94       353
           1       0.62      0.38      0.47        55

    accuracy                           0.88       408
   macro avg       0.76      0.67      0.70       408
weighted avg       0.87      0.88      0.87       408
```

2.2.1.2 Cross-validation score F1,F1 and accuracy for Optimized Random Forest Classifier using Gridsearch with Oversampling Data:



```
OPTIMIZED MODEL PERFORMANCE:
Accuracy: 0.8824
F1-score: 0.5000
Precision: 0.5854
Recall: 0.4364

Classification Report:
              precision    recall  f1-score   support

           0       0.92      0.95      0.93       353
           1       0.59      0.44      0.50        55

    accuracy                           0.88       408
   macro avg       0.75      0.69      0.72       408
weighted avg       0.87      0.88      0.87       408
```

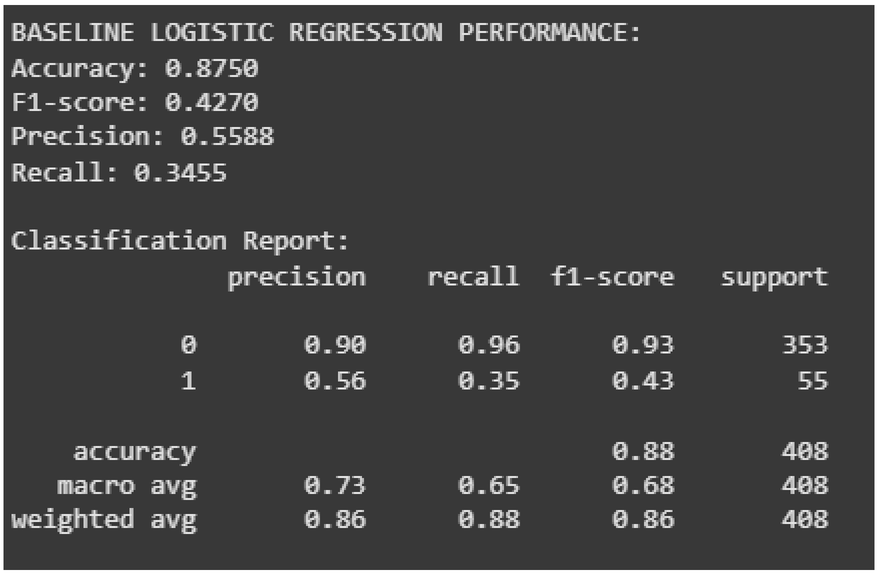
\



```
Best parameters: {'max_depth': None, 'min_samples_leaf': 1, 'min_samples_split': 5, 'n_estimators': 100}
Best cross-validation score: 0.9671257580681194
```

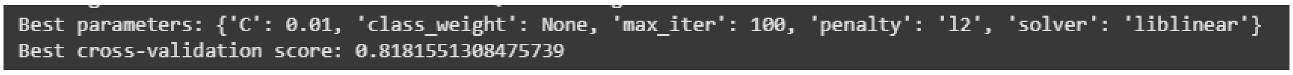
2.2.2 Logistic Regression:

2.2.2.1 F1 and accuracy for Baseline Model Logistic Regression:

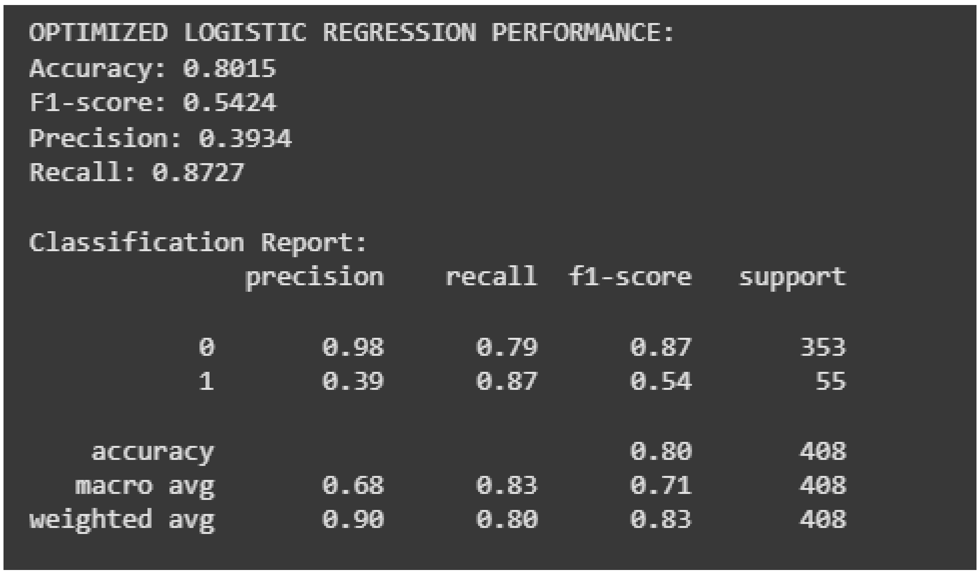


```
BASELINE LOGISTIC REGRESSION PERFORMANCE:
Accuracy: 0.8750
F1-score: 0.4270
Precision: 0.5588
Recall: 0.3455

Classification Report:
              precision    recall  f1-score   support

           0       0.90      0.96      0.93       353
           1       0.56      0.35      0.43        55

    accuracy                           0.88       408
   macro avg       0.73      0.65      0.68       408
weighted avg       0.86      0.88      0.86       408
```

2.2.2.2 Cross-validation score F1,F1 and accuracy Optimize Logistic Regression pipeline with GridsearchCV using feature selection and oversampling data:



```
Best parameters: {'C': 0.01, 'class_weight': None, 'max_iter': 100, 'penalty': 'l2', 'solver': 'liblinear'}
Best cross-validation score: 0.8181551308475739
```



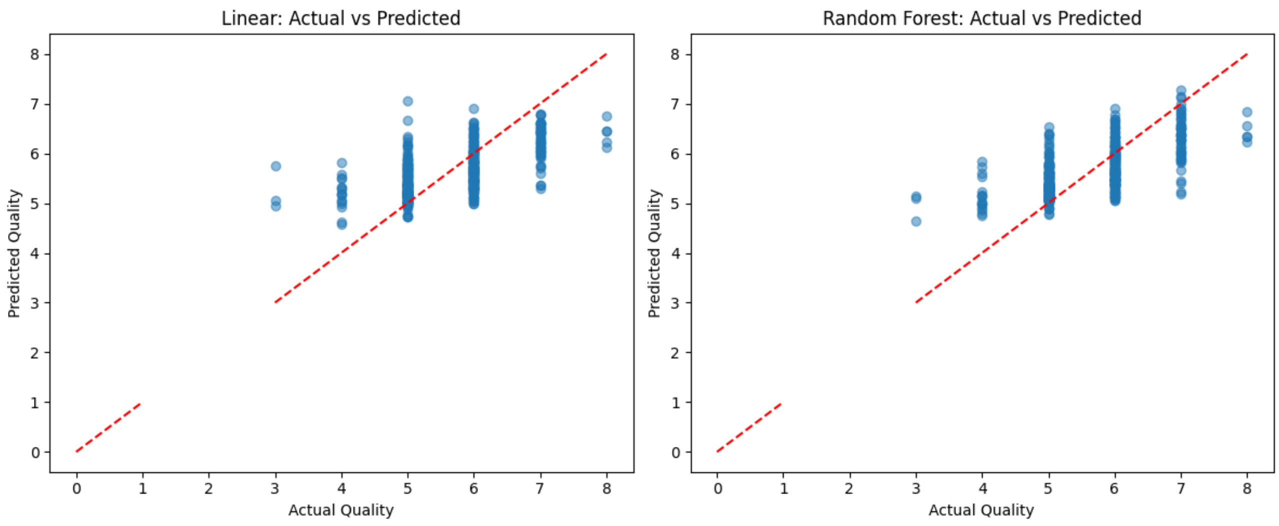
```
OPTIMIZED LOGISTIC REGRESSION PERFORMANCE:
Accuracy: 0.8015
F1-score: 0.5424
Precision: 0.3934
Recall: 0.8727

Classification Report:
              precision    recall  f1-score   support

           0       0.98      0.79      0.87       353
           1       0.39      0.87      0.54        55

    accuracy                           0.80       408
   macro avg       0.68      0.83      0.71       408
weighted avg       0.90      0.80      0.83       408
```

CHAPTER 7: DISCUSSION

- The distribution is **skewed toward scores 5 and 6**, meaning most wines are of medium quality. This unbalanced distribution affects prediction difficulty and model evaluation.
- High (8) and low (3, 4) quality levels are rare, posing challenges for predicting these cases.
- The total sample size (1,019) is moderate, but the imbalance may impact model performance, particularly for linear models like Linear Regression.

1. Regression Tasks:

1.1. Model Performance



- Linear Regression:
  - **Data Distribution**: The scatter points are clustered around the middle quality range (4 to 6), with fewer points at the extremes (3 and 7-8). This reflects the imbalanced distribution of the dataset, where quality levels 5 and 6 dominate.
  - **Prediction Accuracy**:
    - Many points deviate significantly from the diagonal line, especially for qualities 5 and 6, where the error bars (vertical lines) indicate a wide spread in predicted values.
    - The model tends to underpredict or overpredict, particularly for quality levels around 5-6, suggesting limited ability to capture the variability in the data.
  - **Error Bars**: The presence of error bars (likely representing confidence intervals or standard deviation) shows higher uncertainty in predictions, especially around the central quality values.
  - **Overall Fit**: The scatter suggests that Lasso struggles to fit the data well, aligning with its RMSE of 0.6598, which indicates moderate performance.
- Random Forest Regression:
  - **Data Distribution**: Similar to Lasso, the points are concentrated between 4 and 6, with sparse representation at the extremes.
  - **Prediction Accuracy**:
    - The points are closer to the diagonal line compared to Lasso, indicating better alignment between actual and predicted values.
    - The error bars are shorter, suggesting lower uncertainty and more consistent predictions.

        - The model performs better at capturing the central quality range (5-6) and shows improved predictions for higher qualities (7-8).
    - **Overall Fit**: The tighter clustering around the diagonal line reflects Random Forest's superior performance, consistent with its optimized RMSE of 0.6322, which is lower than Lasso's RMSE.
- Overall:
    - Random Forest is the best-performing model (RMSE = 0.6322 after optimization), outperforming Linear Regression (RMSE = 0.6572), Lasso (RMSE = 0.6598), and Ridge (RMSE = 0.6602). This aligns with the dataset's non-linear relationships and imbalanced distribution.
    - Linear Regression and its variants (Lasso, Ridge) have similar performance (RMSE ~0.6572-0.6602), suggesting the data is not entirely linear, and regularization offers little benefit.
    - The imbalanced data (~84% samples at quality 5-6) likely hinders accurate predictions for rare quality levels (3, 4, 8).

1.2. Optimization

- Random Forest optimization via GridSearchCV reduced RMSE by ~0.0069, underscoring the importance of hyperparameter tuning for non-linear models.
- Lasso and Ridge optimizations did not improve performance, likely due to prior feature selection (e.g., citric_acid removal) or the remaining features having comparable importance.

1.3. Practical Applications

- **Quality Prediction**: Random Forest can be applied in the wine industry to assess quality based on physicochemical properties, optimizing production processes.
- **Key Features**: alcohol, sulphates, and volatile_acidity are critical factors, suggesting producers focus on controlling these properties.

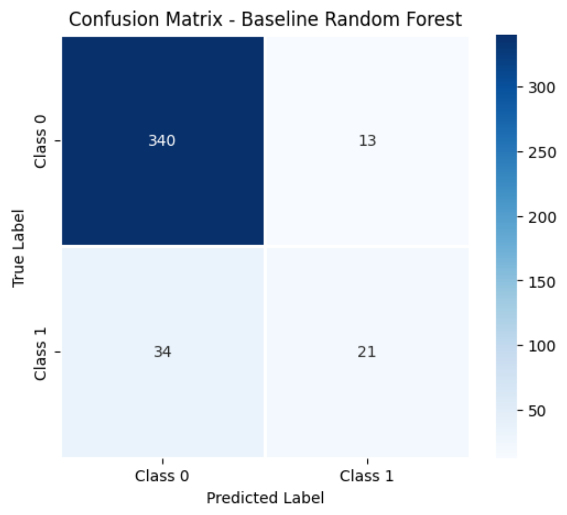
1.4. Limitations

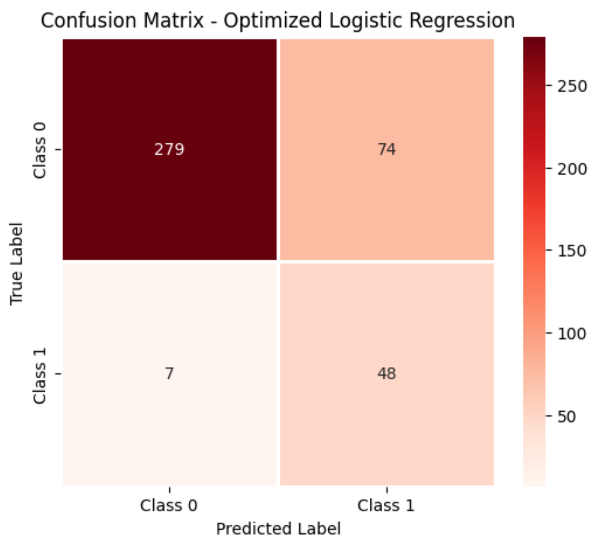
- Imbalanced data makes predicting rare quality levels challenging.
- Linear Regression struggles with non-linear relationships, while Random Forest, though superior, is less interpretable.

2. Classification Tasks:

2.1. Analysis of Confusion Matrices

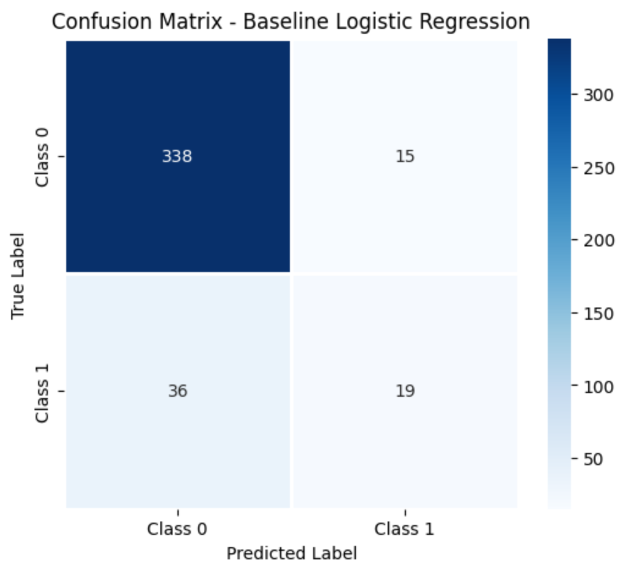
2.1.1 Baseline Random Forest vs. Optimized Random Forest

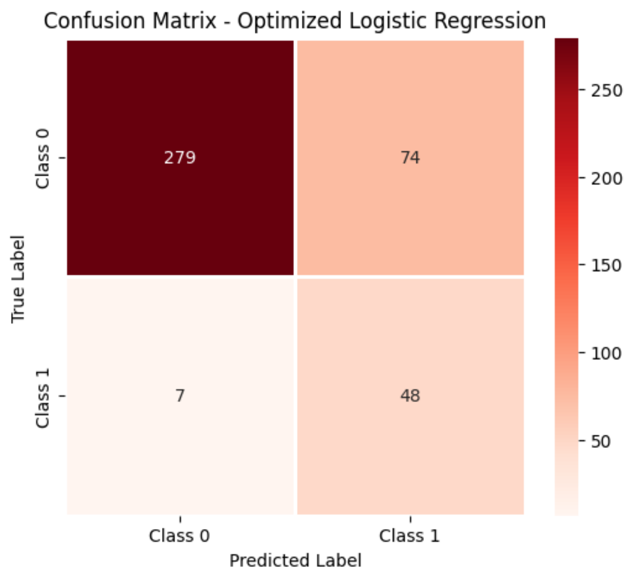




- **Baseline Random Forest:**
  - True Class 0 (822 samples): 340 correctly predicted, 13 misclassified as Class 1.
  - True Class 1 (129 samples): 34 misclassified as Class 0, 21 correctly predicted.
  - Observation: The model performs well in predicting Class 0 (high coverage), but its performance on Class 1 is poor (only 21/55 correct), reflecting the imbalanced dataset (Class 0 is 86.4%).
- **Optimized Random Forest:**
  - True Class 0: 336 correctly predicted, 17 misclassified as Class 1.
  - True Class 1: 31 misclassified as Class 0, 24 correctly predicted.
  - Observation: After optimization with GridSearchCV and oversampling, the number of correct predictions for Class 1 increased from 21 to 24, though still limited. The number of misclassifications for Class 0 increased slightly (13 to 17), indicating a better balance but continued preference for Class 0.

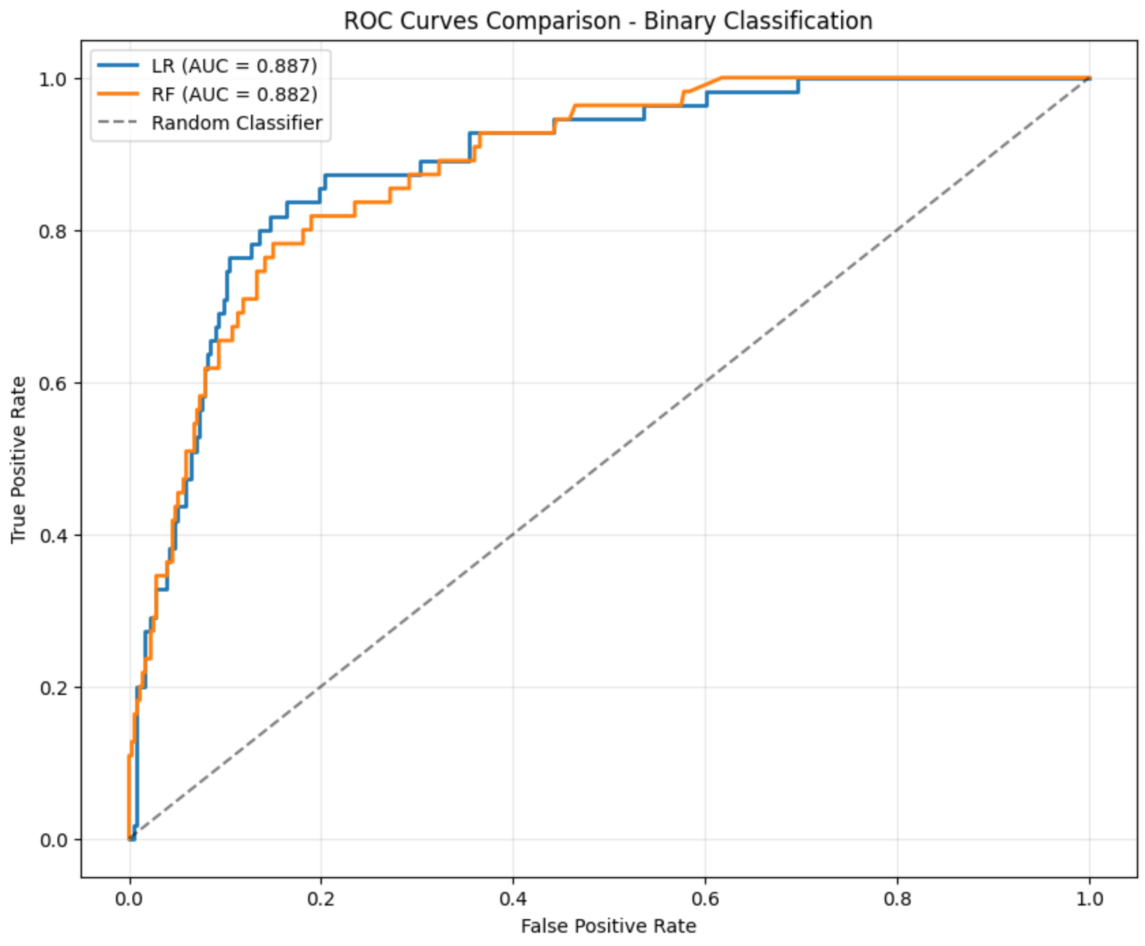
2.1.2 Baseline Logistic Regression vs. Optimized Logistic Regression





- **Baseline Logistic Regression:**
  - True Class 0: 338 correctly predicted, 15 misclassified as Class 1.
  - True Class 1: 36 misclassified as Class 0, 19 correctly predicted.
  - Observation: Similar to Random Forest, the model favors Class 0, with poor performance on Class 1 (19/55).
- **Optimized Logistic Regression:**
  - True Class 0: 279 correctly predicted, 74 misclassified as Class 1.
  - True Class 1: 7 misclassified as Class 0, 48 correctly predicted.
  - Observation: After optimization with GridSearchCV and oversampling, Recall for Class 1 improved significantly (from 19 to 48), but Precision decreased (more misclassifications to Class 1, 74 vs. 15), indicating a shift toward favoring Class 1 predictions.

2.2. Analysis of ROC Curves and AUC



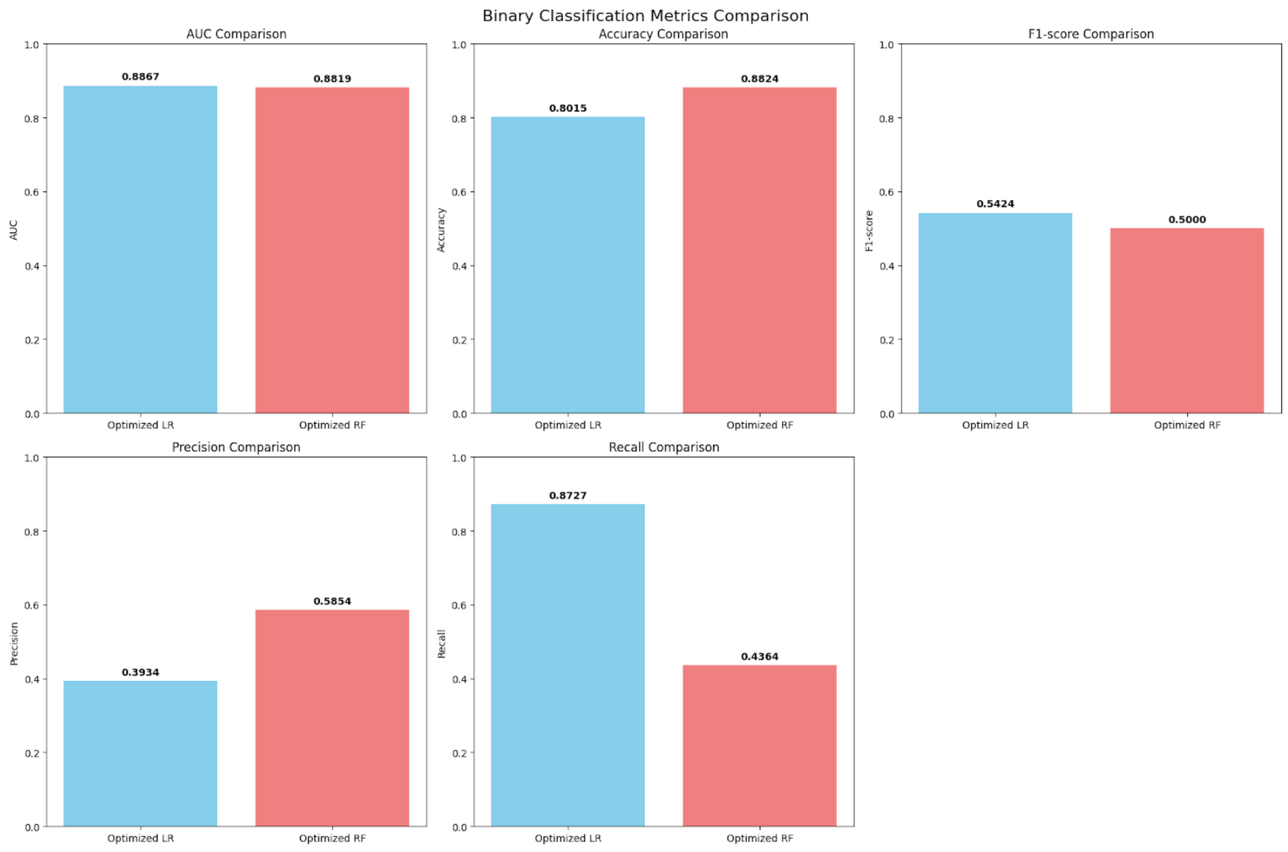
- **ROC Curves**:
    - **Logistic Regression (AUC = 0.8867)**: The curve is above the Random Classifier line, indicating good discriminative ability.
    - **Random Forest (AUC = 0.8819)**: The curve is close to Logistic Regression but slightly lower.
    - **Observation**: Both models outperform the Random Classifier (AUC = 0.5), with Logistic Regression having a slight edge (0.8867 vs. 0.8819).
- **Implication**: The high AUC values suggest strong classification performance, though the small difference (0.0048) is negligible.

2.3. Comparison of Classification Metrics

- Optimized Logistic Regression
    - **Accuracy**: 0.8015 (down 8.4% from 0.8750).
    - **F1-score**: 0.5424 (up 27.03%).
    - **Precision**: 0.3934 (down 29.59%).
    - **Recall**: 0.8727 (up 152.63%).
    - **Observation**: The model prioritizes Recall, improving Class 1 detection but reducing Precision due to increased false positives for Class 1.
- Optimized Random Forest
    - **Accuracy**: 0.8824 (down 0.28% from 0.8848).
    - **F1-score**: 0.5000 (up 5.95%).
    - **Precision**: 0.5854 (down 5.23%).
    - **Recall**: 0.4364 (up 14.29%).

- **Observation**: The model maintains high Accuracy, with modest improvements in F1-score and Recall for Class 1, less pronounced than Logistic Regression.

- Overall Comparison



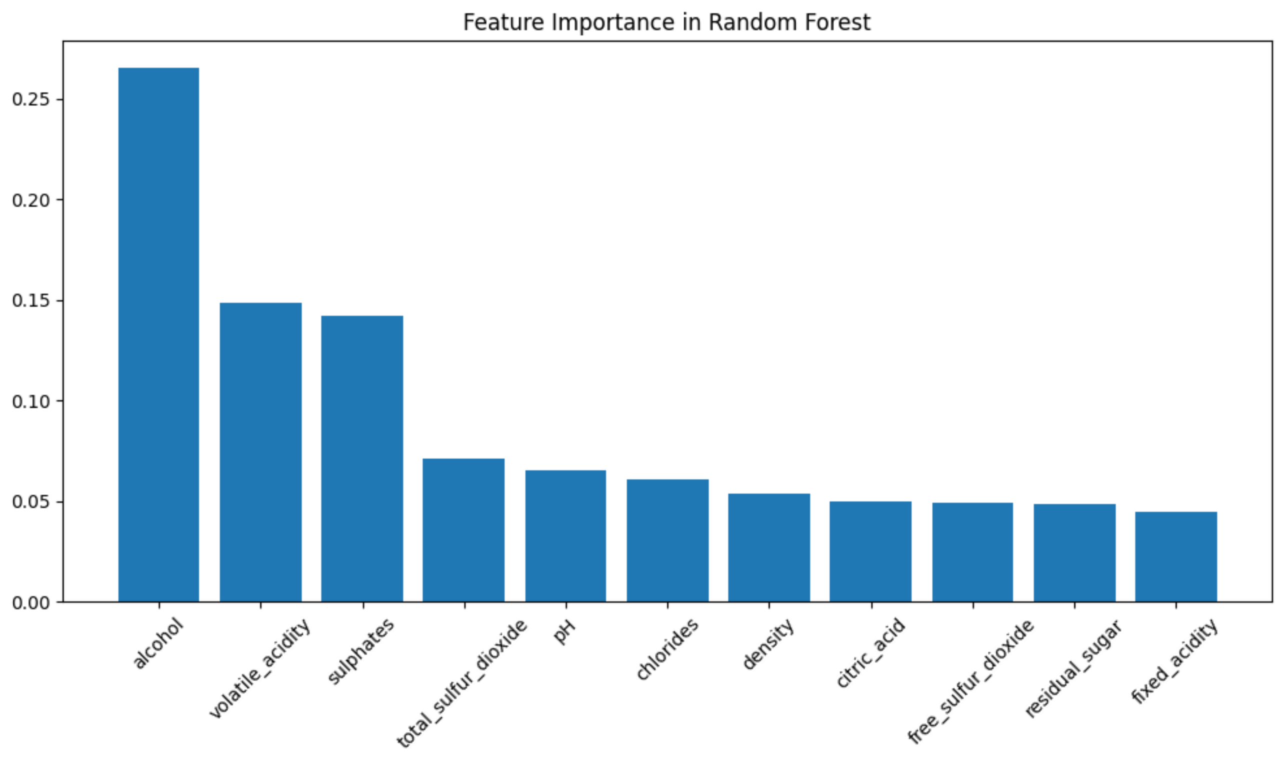
- **Accuracy**: Random Forest (0.8824) outperforms Logistic Regression (0.8015), with a difference of 0.0809.
- **F1-score**: Logistic Regression (0.5424) exceeds Random Forest (0.5000), with a difference of 0.0424.
- **Precision**: Random Forest (0.5854) is higher than Logistic Regression (0.3934), with a difference of 0.1919.
- **Recall**: Logistic Regression (0.8727) surpasses Random Forest (0.4364), with a difference of 0.4364.
- **Conclusion**: Logistic Regression is better suited for high Recall (detecting Class 1), while Random Forest excels in Accuracy and Precision.

CHAPTER 8: CONCLUSIONS

1. Regression Tasks:

- The **Wine Quality Prediction** project successfully built and compared regression models to predict wine quality:
  - Confirm that Random Forest Regression is more effective than Lasso Regression in predicting wine quality, aligning with the quantitative RMSE results. Random Forest's ability to handle non-linear patterns and its optimization via GridSearchCV contribute to its superior performance. However, the imbalanced data distribution remains a limiting factor, suggesting that future improvements could involve addressing this imbalance or exploring advanced models like Gradient Boosting.

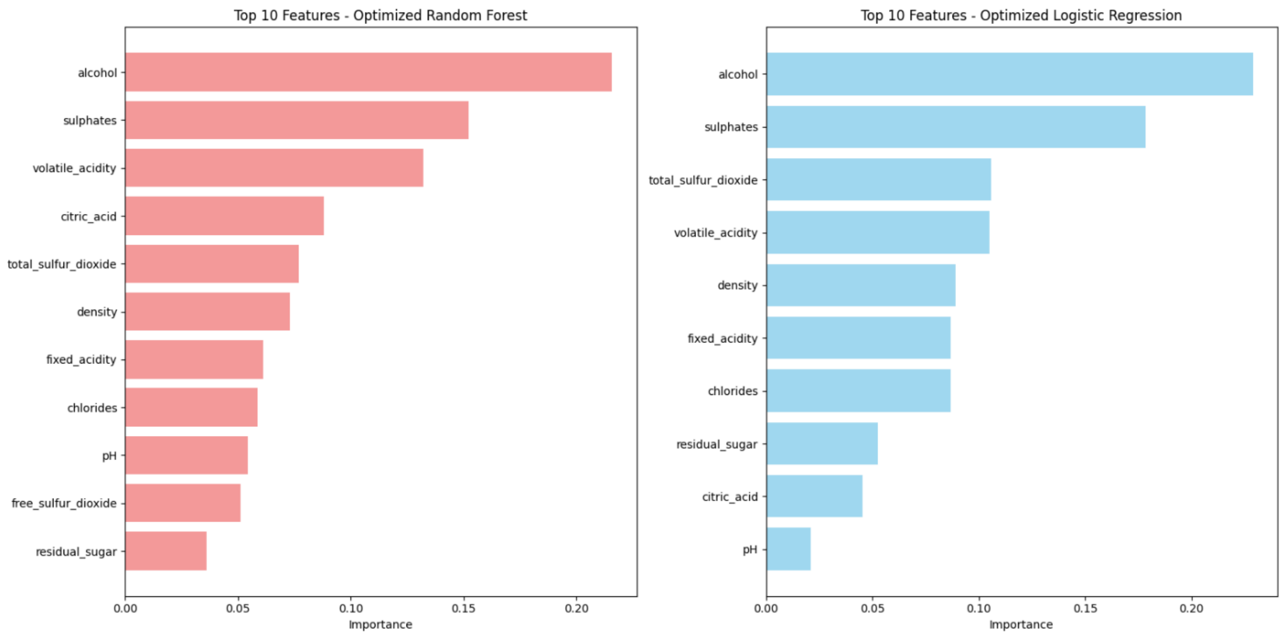
- **Random Forest Regression** achieved the best performance (RMSE = 0.6322 after optimization), outperforming Linear Regression (RMSE = 0.6572), Lasso (RMSE = 0.6598), and Ridge (RMSE = 0.6602).
- **GridSearchCV** optimization significantly improved Random Forest, while Lasso and Ridge offered no notable benefits.



- **Key features:** like alcohol, sulphates, and volatile_acidity strongly influence wine quality, providing valuable insights for the wine industry.

2. Classification Tasks:

- **Performance**: Random Forest maintains high Accuracy (0.8824) and better balances classes, while Logistic Regression improves Recall (0.8727), suitable for detecting Class 1 (high quality).
- **Impact of Optimization**:
  - Random Forest: Increases Recall (0.3818 to 0.4364) and F1-score (0.4719 to 0.5000), but improvements are modest due to data imbalance.
  - Logistic Regression: Significantly boosts Recall (0.3455 to 0.8727) and F1-score (0.4270 to 0.5424), but reduces Precision and Accuracy due to oversampling favoring Class 1.
- **Data Imbalance**: Class 0 (86.4%) dominates, making Class 1 (13.6%) prediction challenging. Oversampling helps Logistic Regression improve Recall but introduces reverse imbalance.
- **Optimized Logistic Regression** is preferable when prioritizing Recall (detecting high-quality wines), with a superior F1-score (0.5424) and Recall (0.8727).
- **Optimized Random Forest** performs better in Accuracy (0.8824) and Precision (0.5854), suitable for general prediction.



- **Key feature:** The feature importance charts for Optimized Random Forest and Optimized Logistic Regression highlight alcohol, sulphates, and volatile_acidity as the most critical factors influencing wine quality, with slight variations in the ranking of other features. Random Forest emphasizes citric_acid and total_sulfur_dioxide, while Logistic Regression prioritizes total_sulfur_dioxide and density

CHAPTER 9: FUTURE WORK

- Combine oversampling with class weight adjustments or explore models like XGBoost to enhance overall performance, especially for the rare Class 1.
- Apply techniques to address data imbalance (e.g., SMOTE) to enhance predictions for rare quality levels**.**
- Experiment with advanced models like Gradient Boosting (XGBoost, LightGBM) to further reduce RMSE.

CHAPTER 10: REFERENCES

- Scikit-learn Documentation: Linear Regression, Random Forest, GridSearchCV.
- Cortez, P., Cerdeira, A., Almeida, F., Matos, T., & Reis, J. (2009). Modeling wine preferences by data mining from physicochemical properties. *Decision Support Systems*, 47(4), 547-553.