Introduction:

The primary aim of this report is to delve into the white wine dataset, exploring its features and potential issues for machine learning applications. I will investigate the efficacy of preprocessing techniques in improving model performance and evaluate different machine learning algorithms for predictive tasks.

This report will provide a comprehensive overview of my experimental setup, including the dataset description, programming languages, libraries utilized, and the machine learning algorithms considered. Furthermore, I will present the results obtained from my analysis, along with visualizations and insights derived from the data exploration process.

Through this exercise, I aim to demonstrate the effectiveness of automated machine learning (AutoML) techniques in streamlining the model-building process and selecting the most suitable algorithm for a given task. The models used in this experiment will be Random Forest, Support vector machine, Logistic regression, Decision tree and Gradient Boosting. By documenting my approach and findings, I hope to contribute valuable insights into the application of machine learning in predictive modeling tasks, particularly in the context of wine quality assessment.

Dataset Description: The dataset consists of the data of the white variant of the Portuguese "Vinho Verde" wine. It comprises of 12 different parameters named as fixed acidity, volatile acidity, citric acid, residual sugar, chlorides, free sulfur dioxide, total sulfur dioxide, density, pH, sulphates, alcohol and quality. We checked for the shape of the data to find out the number of rows and columns present in the data by using shape() function. The dataset consists of 4898 rows and 12 columns. We checked for the null values using isnull().sum() function. The dataset didn't have any null values. In this exercise, I will compare the performance of different machine learning algorithms on this dataset and will eventually report the best performing algorithm. The target variable or feature will be a binary class of good or bad wine quality where we would be classifying the quality value greater than equal to 7 to be of good quality and rest would be classified as bad quality.

Experimental setup: The programming language used for this experiment will be python. We begin with importing a few libraries named as pandas, numpy, matplotlib.pyplot and seaborn. Pandas library helps in data manipulation and data analysis, numpy helps in performing operations on arrays, matplotlib.pyplot and seaborn are used for plotting. We begin the experiment by uploading our dataset using the pd.read_csv function. As the data was uploaded, we found out that all the columns and values were present in a merged form in a single column. Hence to separate the data in different columns we used a separator, which was semi-colon (";") in our case, along with the pd.read_csv function. After this we used head() function to view the values of the first few rows of the uploaded data. Once we have confirmed that this is the right dataset we begin with our initial investigation of the data. For this firstly we used shape() function to know the dimensionality of the data. By dimensionality we mean the number of rows and columns present in the dataset. Following this we used the info() function to get the information of all data types present in the dataset. After this we used the isnull function to find out the missing values in the

dataset. Once we have figured out that the dataset doesn't have any missing values now, we will find out the correlation between different features present in our dataset. To find the correlation we have used corr() function and we have chosen 'pearson' method for that. The correlation has been plotted using the sns.heatmap() function. After this the parameters which were highly correlated (i.e. having a correlation value greater than 0.5) were removed. We have used drop() function to remove the parameters that were highly correlated. We then plotted the non-correlated parameters using sns.heatmap() function. Now we have used describe() function to find out the general statistical characteristics of our dataset. After this we have now divided our data into two parts. First is the input data which consists of all the input features and the second is target data which consists of the target feature. For the target feature we have performed label binarization i.e. we have classified the label which is 'quality' in our case into a binary class where quality value greater than equal to 7 has been labeled as 'Good' and lower values have been labeled as 'Bad'. For label binarization we have used apply(lambda y_value:'Good' if y_value>=7 else 'Bad') function. We are now using MinMax scaler to normalize the data. For this we have first imported the MinMax scaler from sklearn.preprocessing library. After this we used fit_transform() function to fit the transformation on our input data. After the preprocessing steps now, we will apply different machine learning algorithms on our preprocessed dataset. The algorithms used in this experiment are listed below:

a) **Random forest:** We considered our problem as a Binary-class classification and began with importing the required modules like Random Forest classifier, Cross validation score and accuracy score from the sklearn library. Now we have fitted the random forest model on the input and target variables using the fit() function. We have now performed K-fold cross-validation where K=5 for our dataset. The general working of K-fold cross validation can be explained as a single parameter called k that refers to the number of groups that a given data sample is to be split into. When a specific value for k is chosen, it may be used in place of k in reference to the model, such as K=5 becoming 5-fold cross-validation. If K=5 the dataset will be divided into 5 equal parts and the process mentioned next will run 5 times, each time with a different holdout set. Firstly, take the group as a holdout or test data set. Secondly, take the remaining groups as a training data set. Thirdly, fit a model on the training set and evaluate it on test set and finally retain the evaluation score and discard the model. At the end of the process summarize the skill of the model using the sample of model evaluation scores which is "accuracy score" in our case. As can be inferred from the process of cross validation we get 5 accuracy score values for each fold or each iteration and finally by taking a mean of these accuracy scores we get a mean accuracy score. We have used get_params() function to get the default hyperparameters for our Random Forest model and the same has been mentioned in the results section.
b) **Support Vector Machine (SVM)**: It's also called support vector classifier. We began with importing the required modules like Support vector classifier, Cross validation score and accuracy score from the sklearn library. Now we have fitted the support vector classifier model on the input and target variables using the fit() function. We have now performed K-fold cross-validation where K=5 for our dataset. The general working of k-fold cross

validation is same as above. Here also we get mean accuracy score and default hyperparameter using the same procedure as used for random forest model and same has been reported in the results section.

c) **Logistic Regression**: We began by importing the required modules like Logistic Regression, Cross validation score and accuracy score from the sklearn library. Now we have fitted the logistic regression model on the input and target variables using the fit() function. We have now performed K-fold cross-validation where K=5 for our dataset. The general working of k-fold cross validation is same as described in random forest model section. Here also we get mean accuracy score and default hyperparameter using the same procedure as used for random forest model and same has been reported in the results section.

d) **Decision Tree**: We began by importing the required modules like Decision Tree Classifier, Cross validation score and accuracy score from the sklearn library. Now we have fitted the decision tree classifier model on the input and target variables using the fit() function. We have now performed K-fold cross-validation where K=5 for our dataset. The general working of k-fold cross validation is same as described in random forest model section. Here also we get mean accuracy score and default hyperparameter using the same procedure as used for random forest model and same has been reported in the results section.

e) **Gradient Boosting**: We began by importing the required modules like Gradient Boosting Classifier, Cross validation score and accuracy score from the sklearn library. Now we have fitted the gradient boosting classifier model on the input and target variables using the fit() function. We have now performed K-fold cross-validation where K=5 for our dataset. The general working of k-fold cross validation is same as described in random forest model section. Here also we get mean accuracy score and default hyperparameter using the same procedure as used for random forest model and same has been reported in the results section.

Results: The different results obtained in our experiment are mentioned in this section. The very first result came from using the shape() function which shows us that the data consists of 4898 rows and 12 columns. The info() function shows that all the 12 columns have 4898 data points, and we have 11 float data types and 1 integer data type columns. The isnull() function shows that there were no missing values in our dataset. Now we have found out the correlation between different input features and the same has been plotted in figure 1. As can be seen from figure 1, density and residual sugar are having high correlation, hence we are removing density from our study. Similarly, total sulfur dioxide and free sulfur dioxide have a high correlation, so total sulfur dioxide is removed from the study. The updated parameters with non-correlative parameters are again plotted and can be seen in figure 2.

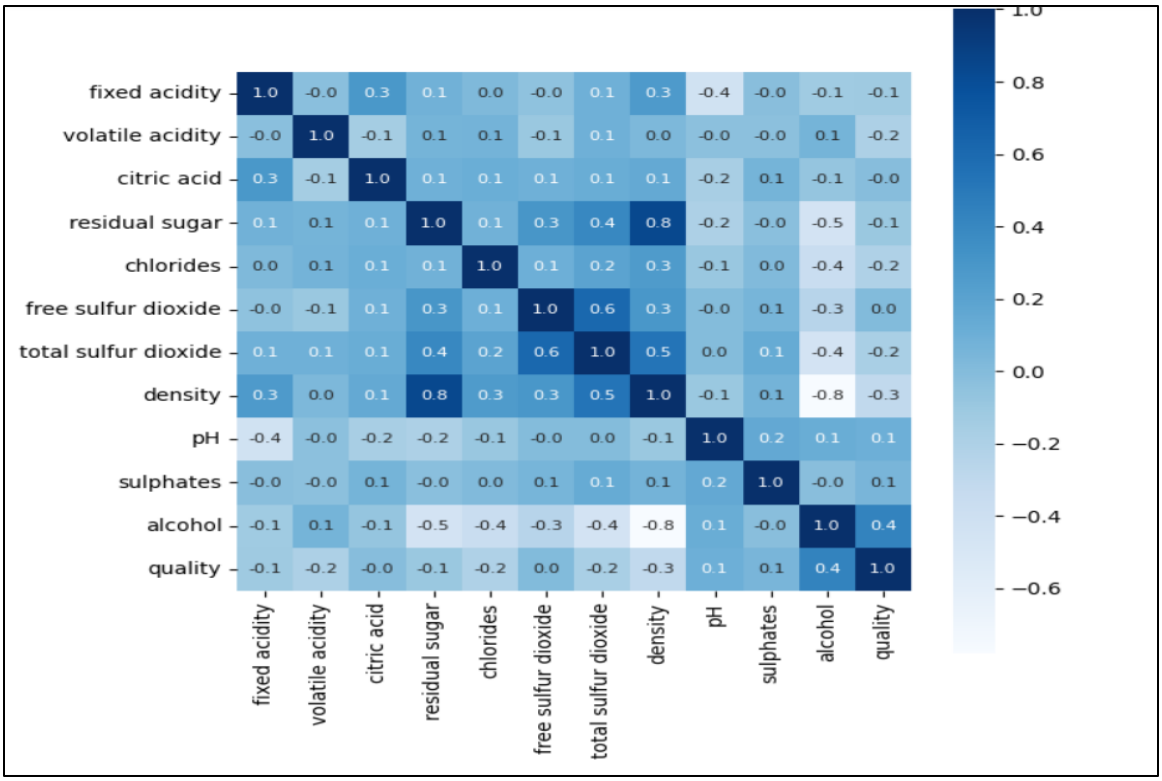

Figure 1. Correlation plot on initial data

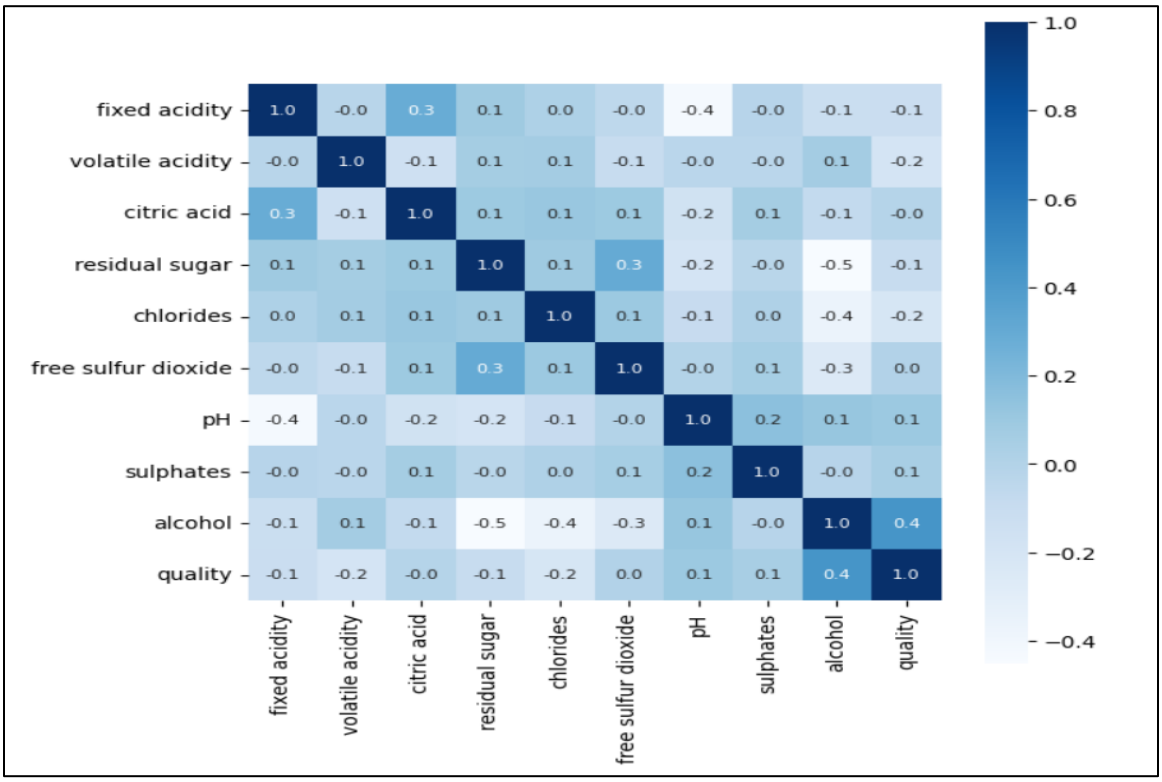

Figure 2. Correlation plot after removing correlated features in the data

The describe() function now gives us the statistical characteristics of the data like counts, mean, standard deviation etc. Now we split the data into input and target features. The input features are fixed acidity, volatile acidity, citric acid, residual sugar, chlorides, free sulfur dioxide, pH, sulphates, and alcohol, while the target feature is wine quality. After label binarization the target variable was modified and now labels the data as 'Good' or 'Bad'. After this we normalized the data using MinMax scaler so now all the input features are normalized and have values ranging between 0 and 1. After the normalization of input data we used different algorithms to make our prediction and the mean accuracy scores obtained from different models have been mentioned in the table below.

ML Algorithm	Mean accuracy Score
Random Forest	0.8093
Support Vector Machine	0.8048
Logistic Regression	0.7990
Decision Tree	0.7392
Gradient Boosting	0.7982

From the above table it can be inferred that random forest was the best model followed by support vector machine. Logistic regression and gradient boosting had similar performance while decision tree was the worst performing algorithm for this experiment. The default hyperparameters for each of the ML algoriths have also been found out and are mentioned in the appendix below.

References:

1) Cortez,Paulo, Cerdeira,A., Almeida,F., Matos,T., and Reis,J.. (2009). Wine Quality. UCI Machine Learning Repository. https://doi.org/10.24432/C56S3T.
2) "Detect and Remove the Outliers using Python", https://www.geeksforgeeks.org/detect-and-remove-the-outliers-using-python/
3) "How to use seaborn plotting", https://www.geeksforgeeks.org/python-seaborn-tutorial/
4) "Normalizing the data by use of scaler", https://www.geeksforgeeks.org/data-pre-processing-wit-sklearn-using-standard-and-minmax-scaler/
5) Z-score for normalization, https://medium.com/@TheDataGyan/day-8-data-transformation-skewness-normalization-and-much-more-4c144d370e55
6) Cross Validation Explained: Evaluating estimator performance. Improve your ML model using cross validation. https://towardsdatascience.com/cross-validation-explained-evaluating-estimator-performance-e51e5430ff85
7) Sklearn API Reference. https://scikit-learn.org/stable/modules/classes.html#module-sklearn.ensemble

Appendix

ML Alogorithm	Hyperparameters
Random Forest	'bootstrap': True, 'ccp_alpha': 0.0, 'class_weight': None, 'criterion': 'gini', 'max_depth': None, 'max_features': 'sqrt', 'max_leaf_nodes': None, 'max_samples': None, 'min_impurity_decrease': 0.0, 'min_samples_leaf': 1, 'min_samples_split': 2, 'min_weight_fraction_leaf': 0.0, 'n_estimators': 100, 'n_jobs': None, 'oob_score': False, 'random_state': None, 'verbose': 0, 'warm_start': False
Support Vector Machine	'C': 1.0, 'break_ties': False, 'cache_size': 200, 'class_weight': None, 'coef0': 0.0, 'decision_function_shape': 'ovr', 'degree': 3, 'gamma': 'scale', 'kernel': 'rbf', 'max_iter': -1, 'probability': False, 'random_state': None, 'shrinking': True, 'tol': 0.001, 'verbose': False
Logistic Regression	'C': 1.0, 'class_weight': None, 'dual': False, 'fit_intercept': True, 'intercept_scaling': 1, 'l1_ratio': None, 'max_iter': 100, 'multi_class': 'auto', 'n_jobs': None, 'penalty': 'l2', 'random_state': None, 'solver': 'lbfgs', 'tol': 0.0001, 'verbose': 0,

	'warm_start': False
Decision Tree	'ccp_alpha': 0.0, 'class_weight': None, 'criterion': 'gini', 'max_depth': None, 'max_features': None, 'max_leaf_nodes': None, 'min_impurity_decrease': 0.0, 'min_samples_leaf': 1, 'min_samples_split': 2, 'min_weight_fraction_leaf': 0.0, 'random_state': None, 'splitter': 'best'
Gradient Boosting	'ccp_alpha': 0.0, 'criterion': 'friedman_mse', 'init': None, 'learning_rate': 0.1, 'loss': 'log_loss', 'max_depth': 3, 'max_features': None, 'max_leaf_nodes': None, 'min_impurity_decrease': 0.0, 'min_samples_leaf': 1, 'min_samples_split': 2, 'min_weight_fraction_leaf': 0.0, 'n_estimators': 100, 'n_iter_no_change': None, 'random_state': None, 'subsample': 1.0, 'tol': 0.0001, 'validation_fraction': 0.1, 'verbose': 0, 'warm_start': False