**CSCE 5210 – Fundamentals of Artificial Intelligence**

**Prediction of**

**Quality of Wine**

**(Group III)**

Team Members:

Vasanth Kandagatla -11503339

Pranavi Balakulla - 11508387

Sai Chandu Yalamarthi - 11526256

Padma Priya Dirisala - 11508872

Sai Chaithanya Bollam - 11520516

**Abstract/Project Proposal:**

Wine is so popular among different generations of people. It is very shocking to know that the worldwide distribution of wine is almost 31 million tonnes, which is a humongous magnitude. The wine industry shows an exponential growth as social drinking is on rise. Nowadays, the industry is using product quality certifications to understand the quality of the wine. It will always be a tedious task to have human intervention for certifying the quality and is an expensive process. It is now more important to have a model, preferably an AI model, to evaluate and understand the quality.

**Features/Attributes:**

One of the important factors in certifying and quality assessing is physiochemical tests, which are lab based and considered parameters like **fixed acidity, volatile acidity, citric acid, residual sugar, chlorides, free sulfur dioxide, total sulfur dioxide, density, pH, sulphates,** and **alcohol**. In the dataset, we have considering two types of wine, namely **Red** and **White** wine, slightly inclined to the data related more to red wine.

On assessing and accounting the above attributes, the output variable (i.e., based on sensory data) will be **quality.**

**Source** of the dataset(s):

* [**https://www.kaggle.com/datasets/rajyellow46/wine-quality**](https://www.kaggle.com/datasets/rajyellow46/wine-quality)
* [**https://archive.ics.uci.edu/ml/datasets/wine+quality**](https://archive.ics.uci.edu/ml/datasets/wine+quality)

**Requirement:**

An extensive dataset which can be used for training and the testing the model so that the outcome can be a fruitful result.

In terms of software and packages:

* Jupyter Notebook
* Python 3.6 or above
* Pandas for creating dataframes and data analysis
* Numpy for n-dimensional array
* Seaborn for visualization
* Sklearn for ML libraries
* Matplotlib for visualization
* Keras for creating deep models
* Classifiers used are **Logistic Regression, Decision Tree, Naïve Bayes, Random Forest Classifier, Multi-Layer Perceptron, Stochastic Gradient Descent Classifier** and **Support Vector Machine.**

**Evaluation Criteria:**

The quality of the wine is determined by the two criterions:

* To experiment with different classification methods to see which yields the highest accuracy
* To determine which features are the most indicative of a good quality wine

**Project Design**

Diagram

Description automatically generated

Source: Created in PPT by group III

**Data Specification:**

In this study, we use the UCL Machine Learning Repository's publicly available wine quality dataset, which offers a vast variety of datasets that have been widely used by the machine learning community. The link to the data set is <https://archive.ics.uci.edu/ml/datasets/wine+quality>

Because red wine is more popular than white wine among the two categories of wine quality datasets (red wine and white wine), we chose **red wine** data for our investigation. There are **1599** instances of red wine in the dataset. The red wine dataset includes **11** input variables (based on physicochemical tests): *fixed acidity, volatile acidity, citric acid, residual sugar, chlorides, free sulfur dioxide, total sulfur dioxide, density, pH, sulfates, alcohol, and quality*, as well as one output variable based on sensory data. Sensory data is categorized into 11 quality levels ranging from 0 to 10. (0-very bad to 10-very good)

**Algorithms in this study: (In pipeline)**

In this we are using five different techniques

1. Logistic Regression
2. Decision Tree
3. Naive Bayes (NB)
4. Random Forest Classifier
5. Support Vector Machine (SVM)
6. Stochastic Gradient Descent Classifier
7. Multi-Layer Perceptron

**Logistic Regression:**

A logistic regression model predicts a dependent data variable by analyzing the relationship between one or more existing independent variables. For example, a logistic regression could be used to predict whether a political candidate will win or lose an election or whether a high school student will be admitted or not to a particular college. These binary outcomes allow straightforward decisions between two alternatives.

Logistic models can also transform raw data streams to create features for other types of AI and machine learning techniques. In fact, logistic regression is one of the commonly used algorithms in machine learning for binary classification problems, which are problems with two class values, including predictions such as "this or that," "yes or no," and "A or B."

statisticians and citizen data scientists must keep a few assumptions in mind when using logistic regression. For starters, the variables must be independent of one another. So, for example, zip code and gender could be used in a model, but zip code and state would not work.

**Decision Tree**

A decision tree is a diagram that depicts the various outcomes of a set of related options. It enables a person or organization to compare several options based on their prices, probabilities, and advantages. They can be used to spark informal debate or to create an algorithm that mathematically predicts the optimal option. A decision tree usually begins with a single node and branches out into different outcomes. Each of those results leads to new nodes, each of which leads to new possibilities. It takes on a tree-like shape as a result of this.

**Naive Bayes (NB)**

The Naive Bayes method is a supervised learning technique for addressing classification issues that is based on Bayes theorem. It's most employed in text categorization with a large training set. The Naive Bayes Classifier is a basic and effective classification method that aids in the development of fast machine learning models capable of making accurate predictions. It's a probabilistic classifier, which means it makes predictions based on the object's likelihood.

Bayes theorem: Bayes' theorem, often known as Bayes' Rule or Bayes' law, is a mathematical formula for calculating the probability of a hypothesis given previous information. It is conditional probability that determines. The formula for Bayes' theorem is given as:

Where, P(A|B) is Posterior probability: Probability of hypotheses A on the observed event B.

P(B|A) is Likelihood probability: Probability of the evidence given that the probability of a hypotheses is true.

P(A) is Prior probability: Probability of hypotheses before observing the evidence.

P(B) is Marginal Probability: Probability of Evidence.

**Random Forest Classifier**

Random Forest is a well-known machine learning algorithm that uses the supervised learning method. In machine learning, it can be utilized for both classification and regression issues. It is based on ensemble learning, which is a method of integrating several classifiers to solve a complex problem and increase the model's performance. "Random Forest is a classifier that contains a number of decision trees on various subsets of a given dataset and takes the average to enhance the predicted accuracy of that dataset," according to the name. Instead, than relying on a single decision tree, the random forest collects the forecasts from each tree and predicts the final output based on the majority votes of predictions.

The bigger the number of trees in the forest, the more accurate it is and the problem of overfitting is avoided. The random forest is formed in two phases: the first is to combine N decision trees to build the random forest, and the second is to make predictions for each tree created in the first phase.

The steps of the working procedure are as follows:

Step 1: Pick K data points at random from the training set.

Step 2: Create decision trees for the data points you've chosen (Subsets).

Step 3: Decide on the number N for the decision trees you wish to create.

Step 4: Repetition of Steps 1 and 2.

Step 5: Find the forecasts of each decision tree for new data points, and allocate the new data points to the category with the most votes.

**Support Vector Machine (SVM):**

SVMs differ from other machine learning algorithms in the manner they are implemented. It has classification, regression, and outlier detection capabilities. A separative hyperplane is used to officially create a Support Vector Machine, which is a discriminative classifier. It is a representation of examples as points in space that are mapped with a gap as large as possible between the points of different categories. SVM also has the ability to do non-linear classification. Let's look at the Support Vector Machine in action. The basic goal of a support vector machine is to sort the data as efficiently as feasible. The margin is the distance between the nearest spots once segregation is completed. The method entails choosing a hyperplane with the greatest feasible margin between the support vectors in the data sets.

**Stochastic Gradient Descent Classifier**

Stochastic gradient descent (often abbreviated SGD) is an iterative method for optimizing an objective function with suitable smoothness properties (e.g. differentiable or subdifferentiable). It can be regarded as a stochastic approximation of gradient descent optimization, since it replaces the actual gradient (calculated from the entire data set) by an estimate thereof (calculated from a randomly selected subset of the data). Especially in high-dimensional optimization problems this reduces the very high computational burden, achieving faster iterations in trade for a lower convergence rate.

**Multi-Layer Perceptron**

A multilayer perceptron (MLP) is a fully connected class of feedforward artificial neural network (ANN). The term MLP is used ambiguously, sometimes loosely to mean any feedforward ANN, sometimes strictly to refer to networks composed of multiple layers of perceptron’s (with threshold activation); see § Terminology. Multilayer perceptron’s are sometimes colloquially referred to as "vanilla" neural networks, especially when they have a single hidden layer.

**Method and Approach:**

Firstly, I have imported all the relevant and needful packages/libraries in the program

Graphical user interface, text, application

Description automatically generated

In the next step, the data is inserted into the code by using pandas library and just to make clear, the columns are displayed in the data incorporated.

Graphical user interface, application

Description automatically generated

There are a total of 1599 rows and 12 columns, including the output variable “**quality**”.

Now let us explore the variables and check the correlation between the variables

we are curious about the relationships between the variables we are working with. In a single glance, this allows us to gain a far better knowledge of the relationships between our variables. We can see right away that there are some characteristics that are highly linked to quality. These factors are likely to be the most important features in our machine learning model as well.

Table

Description automatically generated

Let's do some plotting to know how the data columns are distributed in the dataset. Bivariate Analysis/ Data Viz., In other words, lets see graph between each outlier to its impact on quality.

Chart

Description automatically generated with medium confidence

Graphical user interface, application

Description automatically generated

Graphical user interface

Description automatically generated with medium confidence

Chart, bar chart

Description automatically generated

Chart

Description automatically generated with medium confidence

Chart

Description automatically generated

Let us now see the holistic view of all the variables in the dataset using a heatmap

Chart

Description automatically generated

From the above correlation plot for the given dataset for wine quality prediction, we can easily see which items are related strongly with each other and which items are related weekly with each other.

**The strongly correlated items are:**

1.fixed acidity and citric acid. 2.free sulfur dioxide and total sulfur dioxide. 3.fixed acidity and density. 4. alcohol and quality.

**The weak correlated items are:**

1.citric acid and volatile acidity. 2.fixed acidity and ph. 3. density and alcohol

A picture containing calendar

Description automatically generated

Categorizing the values into good or bad and using label encoder to process the data in two output values only.

Graphical user interface, text, application, email

Description automatically generated

Graphical user interface, text, application, email

Description automatically generated

Let’s draw a counter plot to see the number of good and bad values in quality.

A picture containing graphical user interface

Description automatically generated

Using model selection, dividing the dataset in training and testing models.

Graphical user interface, text, application, email

Description automatically generated

Using Logistic Regression, modelling the code.

In modelling the code, we are going to train the code using the training data to inherit the behavior of the data trend. Similarly, we are going to test the code to know its efficiency. We will extract a classification report and confusion matrix and, the accuracies of the testing and training data.

Graphical user interface

Description automatically generated with medium confidence

The above result quantifies the training accuracy as 74.72%, testing accuracy as 72.25% and briefly shows the classification report and confusion matrix using the Logistic Regression.

Using Support Vector Machine, modelling the code.

In modelling the code, we are going to train the code using the training data to inherit the behavior of the data trend. Similarly, we are going to test the code to know its efficiency. We will extract a classification report and confusion matrix and, the accuracies of the testing and training data.

**Text

Description automatically generated**

The above result quantifies the training accuracy as 80.72%, testing accuracy as 75% and briefly shows the classification report and confusion matrix using the SVM.

Using Decision Tree, modelling the code.

In modelling the code, we are going to train the code using the training data to inherit the behavior of the data trend. Similarly, we are going to test the code to know its efficiency. We will extract a classification report and confusion matrix and, the accuracies of the testing and training data.

Text

Description automatically generated

The above result quantifies the training accuracy as 100%, testing accuracy as 72.81% and briefly shows the classification report and confusion matrix using the Decision Tree.

Using Random Forest, modelling the code.

In modelling the code, we are going to train the code using the training data to inherit the behavior of the data trend. Similarly, we are going to test the code to know its efficiency. We will extract a classification report and confusion matrix and, the accuracies of the testing and training data.

Text

Description automatically generated

The above result quantifies the training accuracy as 100%, testing accuracy as 82.18% and briefly shows the classification report and confusion matrix using the Random Forest.

Using Naive Bayes, modelling the code.

In modelling the code, we are going to train the code using the training data to inherit the behavior of the data trend. Similarly, we are going to test the code to know its efficiency. We will extract a classification report and confusion matrix and, the accuracies of the testing and training data.

Graphical user interface, text

Description automatically generated with medium confidence

The above result quantifies the training accuracy as 73.10%, testing accuracy as 73.125% and briefly shows the classification report and confusion matrix using the Naive Bayes.

Using Multi-Layer Perceptron, modelling the code.

In modelling the code, we are going to train the code using the training data to inherit the behavior of the data trend. Similarly, we are going to test the code to know its efficiency. We will extract a classification report and confusion matrix and, the accuracies of the testing and training data.

Text

Description automatically generated

The above result quantifies the training accuracy as 93.75%, testing accuracy as 75.31% and briefly shows the classification report and confusion matrix using the Multi-Layer Perceptron.

Using Stochastic Gradient Descent Classifier, modelling the code.

In modelling the code, we are going to train the code using the training data to inherit the behavior of the data trend. Similarly, we are going to test the code to know its efficiency. We will extract a classification report and confusion matrix and, the accuracies of the testing and training data.

Text

Description automatically generated with low confidence

The above result quantifies the training accuracy as 72.86%, testing accuracy as 74.68% and briefly shows the classification report and confusion matrix using the Stochastic Gradient Descent Classifier.

**Milestones Acheived:**

1. Data prepared and analyzed
2. The dataset was read
3. The target variable was updated after the change
4. The properties of the data were observed
5. Bivariate analysis of two variables is seen
6. Strongly and weakly correlated entities are identified from the data.
7. Train and test sets are created
8. Here we used 7 machine learning algorithms to create the model they are Logistic Regression, Decision Tree, naive Bayes (NB), Random Forest Classifier, and support vector machine (SVM), Stochastic Gradient Descent Classifier, Multi-Layer Perceptron.
9. We calculated Accuracy and performed confusion matrix for all the models

**Conclusion:**

Using a variety of machine learning techniques, predict the quality of red wine. When done the old approach, the feature selects on algorithm provided a clear understanding about the importance of the attributes for quality prediction, which was time consuming and expensive. We also compared the accuracy of each technique employed in quality prediction and found that these classifiers worked admirably. We discovered that feature sets based on ***Random Forest*** performed better than others. We also discovered that the ***Random Forest classifier*** outperformed all other classifiers on the red wine data set.

We can use different performance measurements and machine learning techniques in the future to compare results more effectively. This study will aid the wine industry in predicting the quality of various types of wines based on certain characteristics, as well as assisting them in producing high-quality products in the future.

**Code**:

**Wine prediction project by Group 3**

import pandas as pd

import numpy as np

import sklearn

import seaborn as sns

import matplotlib.pyplot as plt

from sklearn.model\_selection import train\_test\_split

from sklearn.preprocessing import StandardScaler

from sklearn.linear\_model import LogisticRegression

from sklearn.metrics import confusion\_matrix, accuracy\_score

import pandas as pd

import numpy as np

import sklearn

import seaborn as sns

import matplotlib.pyplot as plt

from sklearn.model\_selection import train\_test\_split

from sklearn.preprocessing import StandardScaler

from sklearn.linear\_model import LogisticRegression

from sklearn.metrics import confusion\_matrix, accuracy\_score

**Loading the data set into the project.**

winedata = pd.read\_csv("winequality-red.csv")

winedata.head()

winedata.columns

**Checking the correlation for each of the fields**

winedata.corr

**Let's do some plotting to know how the data columns are distributed in the dataset**

**Bivariate analysis/Graphs**

f = plt.figure(figsize = (10,6))

**Quality vs fixed acidity**

sns.barplot(x = 'quality', y = 'fixed acidity', ci=None, data = winedata)

**Quality vs Volatile acidity**

sns.barplot(x = 'quality', y = 'volatile acidity', ci=None, data = winedata)

**Quality vs Alcohol**

sns.barplot(x = 'quality', y = 'alcohol', ci=None, data = winedata)

**Quality vs Citric acid**

sns.barplot(x = 'quality', y = 'citric acid', ci=None, data = winedata)

**Quality vs Residual Sugar**

sns.barplot(x = 'quality', y = 'residual sugar', ci=None, data = winedata)

**Quality vs Chlorides**

sns.barplot(x = 'quality', y = 'chlorides', ci=None, data = winedata)

**Quality vs Free Sulfur Dioxide**

sns.barplot(x = 'quality', y = 'free sulfur dioxide', ci=None, data = winedata)

**Quality vs Sulphates**

sns.barplot(x = 'quality', y = 'sulphates',ci=None, data = winedata)

**Quality vs Total Sulfur Dioxide**

sns.barplot(x = 'quality', y = 'total sulfur dioxide',ci=None, data = winedata)

**Checking correlation between attributes using a heat map**

f, ax = plt.subplots(figsize=(8, 6))

corr = winedata.corr()

sns.heatmap(corr, cmap=sns.diverging\_palette(210, 10, as\_cmap=True),

square=True, ax=ax)

sns.pairplot(winedata)

**Understanding the data and data pre-processing**

winedata.info()

winedata.shape

winedata.describe()

winedata['quality'].value\_counts()

winedata.shape

winedata.describe()

winedata['quality'] = winedata['quality'].map({3 : 'bad', 4 :'bad', 5: 'bad',

6: 'good', 7: 'good', 8: 'good'})

# analyzing the different values present in the dependent variable(quality column)

winedata['quality'].value\_counts()

from sklearn.preprocessing import LabelEncoder

le = LabelEncoder()

winedata['quality'] = le.fit\_transform(winedata['quality'])

winedata['quality'].value\_counts

sns.countplot(winedata['quality'])

# dividing the dataset into dependent and independent variables

x = winedata.iloc[:,:11]

y = winedata.iloc[:,11]

# determining the shape of x and y.

print(x.shape)

print(y.shape)

# dividing the dataset in training and testing set

from sklearn.model\_selection import train\_test\_split

x\_train, x\_test, y\_train, y\_test= train\_test\_split(x,y,test\_size=0.20,random\_state=1)

# determining the shapes of training and testing sets

print(x\_train.shape)

print(y\_train.shape)

print(x\_test.shape)

print(y\_test.shape)

# standard scaling

from sklearn.preprocessing import StandardScaler

sc = StandardScaler()

x\_train = sc.fit\_transform(x\_train)

x\_test = sc.fit\_transform(x\_test)

**Logistic Regression**

from sklearn.linear\_model import LogisticRegression

from sklearn.metrics import classification\_report, confusion\_matrix

​

# creating the model

model = LogisticRegression()

​

# feeding the training set into the model

model.fit(x\_train, y\_train)

​

# predicting the results for the test set

y\_pred = model.predict(x\_test)

​

# calculating the training and testing accuracies

print("Training accuracy :", model.score(x\_train, y\_train))

print("Testing accuracy :", model.score(x\_test, y\_test))

​

# classification report

print(classification\_report(y\_test, y\_pred))

​

# confusion matrix

print(confusion\_matrix(y\_test, y\_pred))

**Support Vector Machine**

from sklearn.svm import SVC

​

# creating the model

model = SVC()

​

# feeding the training set into the model

model.fit(x\_train, y\_train)

​

# predicting the results for the test set

y\_pred = model.predict(x\_test)

​

# calculating the training and testing accuracies

print("Training accuracy :", model.score(x\_train, y\_train))

print("Testing accuracy :", model.score(x\_test, y\_test))

​

# classification report

print(classification\_report(y\_test, y\_pred))

​

# confusion matrix

print(confusion\_matrix(y\_test, y\_pred))

**Decision Tree**

from sklearn.tree import DecisionTreeClassifier

​

# creating the model

model = DecisionTreeClassifier()

​

# feeding the training set into the model

model.fit(x\_train, y\_train)

​

# predicting the results for the test set

y\_pred = model.predict(x\_test)

​

# calculating the training and testing accuracies

print("Training accuracy :", model.score(x\_train, y\_train))

print("Testing accuracy :", model.score(x\_test, y\_test))

​

# classification report

print(classification\_report(y\_test, y\_pred))

​

# confusion matrix

print(confusion\_matrix(y\_test, y\_pred))

**Random Forest**

from sklearn.ensemble import RandomForestClassifier

​

# creating the model

model = RandomForestClassifier()

​

# feeding the training set into the model

model.fit(x\_train, y\_train)

​

# predicting the results for the test set

y\_pred = model.predict(x\_test)

​

# calculating the training and testing accuracies

print("Training accuracy :", model.score(x\_train, y\_train))

print("Testing accuracy :", model.score(x\_test, y\_test))

​

# classification report

print(classification\_report(y\_test, y\_pred))

​

# confusion matrix

print(confusion\_matrix(y\_test, y\_pred))

**Naive Bayes**

from sklearn.naive\_bayes import GaussianNB

​

# creating the model

model = GaussianNB()

​

# feeding the training set into the model

model.fit(x\_train, y\_train)

​

# predicting the results for the test set

y\_pred = model.predict(x\_test)

​

# calculating the training and testing accuracies

print("Training accuracy :", model.score(x\_train, y\_train))

print("Testing accuracy :", model.score(x\_test, y\_test))

​

# classification report

print(classification\_report(y\_test, y\_pred))

​

# confusion matrix

print(confusion\_matrix(y\_test, y\_pred))

**Multi Layer Perceptron**

from sklearn.neural\_network import MLPClassifier

​

# creating the model

model = MLPClassifier(hidden\_layer\_sizes = (100, 100), max\_iter = 150)

​

# feeding the training data to the model

model.fit(x\_train, y\_train)

​

# calculating the accuracies

print("training accuracy :", model.score(x\_train, y\_train))

print("testing accuracy :", model.score(x\_test, y\_test))

​

# classification report

print(classification\_report(y\_test, y\_pred))

​

# confusion matrix

print(confusion\_matrix(y\_test, y\_pred))

**Stochastic Gradient Descent Classifier**

from sklearn.linear\_model import SGDClassifier

# creating the model

model = SGDClassifier(penalty=None)

# feeding the training data to the model

model.fit(x\_train, y\_train)

# calculating the accuracies

print("training accuracy :", model.score(x\_train, y\_train))

print("testing accuracy :", model.score(x\_test, y\_test))

# classification report

print(classification\_report(y\_test, y\_pred))

# confusion matrix

print(confusion\_matrix(y\_test, y\_pred))

from sklearn.linear\_model import SGDClassifier

​

# creating the model

model = SGDClassifier(penalty=None)

​

# feeding the training data to the model

model.fit(x\_train, y\_train)

​

# calculating the accuracies

print("training accuracy :", model.score(x\_train, y\_train))

print("testing accuracy :", model.score(x\_test, y\_test))

​

# classification report

print(classification\_report(y\_test, y\_pred))

​

**Reference:**

1. Logistic Regression

<https://www.techtarget.com/searchbusinessanalytics/definition/logistic-regression>

1. Stochastic Gradient Descent Classifier <https://en.wikipedia.org/wiki/Stochastic_gradient_descent>
2. <https://towardsdatascience.com/predicting-wine-quality-with-several-classification-techniques-179038ea6434>
3. WINE QUALITY PREDICTION MODEL USING MACHINE LEARNING TECHNIQUES by Rohan Dilip Kothawade
4. Quality Prediction of Red Wine based on Different Feature Sets Using Machine Learning Techniques by Nikita Sharma
5. Dahal, K., Dahal, J., Banjade, H., Gaire, S., 2021. Prediction of Wine Quality Using Machine Learning Algorithms. Open J. Stat. 11, 278–289. <https://doi.org/10.4236/ojs.2021.112015>