Quality Rating of Wine on the basis of Chemical Composition

*

1st Maitri Jain
Dept of Computer Science
University Of New Brunswick
Fredericton, New Brunswick
mjain1@unb.ca

2nd Amit Rawat
Dept of Computer Science
University Of New Brunswick
Fredericton, New Brunswick
amit.rawat@unb.ca

3rd Rajat Dhaiya
Dept of Computer Science
University of New Brunswick
Fredericton,New Brunswick
rajat.dhaiya@unb.ca

Abstract—**The Wine Quality dataset is a popular dataset that contains information about several types of red wine, including their chemical compositions and quality ratings. The objective of this project is to train a logistic regression model to predict the quality rating of a wine sample based on its chemical composition features. The dataset contains 11 input variables such as fixed acidity, volatile acidity, citric acid, residual sugar, chlorides, free sulfur dioxide, total sulfur dioxide, density, pH, sulphates, and alcohol. The target variable is a quality rating ranging from 0 to 10. The performance of the model will be evaluated using standard regression metrics such as mean squared error (MSE) or mean absolute error (MAE). Additionally, we will use Matplotlib to visualize the results of our model.**

I. Introduction

Wine is one of the most popular and widely consumed alcoholic beverages in the world. The quality of wine is determined by a combination of factors, including the type of grape, the soil and climate in which it is grown, and the fermentation process. The chemical composition of wine plays a significant role in its taste, aroma, and overall quality. In recent years, there has been a growing interest in using machine learning algorithms to predict the quality of wine based on its chemical composition.

II. Problem Statement

The Wine Quality dataset contains information about several types of red wine and their chemical compositions. The aim of this project is to develop a logistic regression model that can accurately predict the quality rating of a wine sample based on its chemical composition features. The dataset contains 11 input variables, including fixed acidity, volatile acidity, citric acid, residual sugar, chlorides, free sulfur dioxide, total sulfur dioxide, density, pH, sulphates, and alcohol. Our goal is to use this dataset to train and test a logistic regression model that can accurately predict the quality rating of a wine sample. We will use various performance metrics such as MSE and MAE to evaluate the accuracy of our model and visualize the results using Matplotlib.

III. Literature Review

Several studies have been conducted to predict the quality of wine using machine learning algorithms. One such study by Cortez et al. (2009) used the Wine Quality dataset to predict the quality rating of red and white wines. They compared the performance of various machine learning algorithms such as linear regression, k-nearest neighbor, and neural networks, and found that support vector regression and random forest performed the best.[1]

Another study by Gou et al. (2016) used a similar dataset to predict the quality of Chinese red wines. They compared the performance of various machine learning algorithms such as decision tree, k-nearest neighbor, and support vector regression, and found that support vector regression performed the best.[2]

A study by Sánchez-Pérez et al. (2021) used a dataset similar to the Wine Quality dataset to predict the quality rating of Spanish red wines. They compared the performance of various machine learning algorithms such as random forest, decision tree, and neural networks, and found that gradient boosting regression performed the best.[3]

These studies demonstrate the effectiveness of machine learning algorithms in predicting the quality of wine based on its chemical composition. In this project, we will use logistic regression to predict the quality rating of red wine samples based on their chemical composition features.

IV. Data Description and Acesss

A. Data Description

The Wine Quality dataset contains information about different varieties of red wine, including their chemical compositions and quality ratings. The dataset contains 1599 samples and 12 columns, including 11 input variables and a target variable. The input variables are:

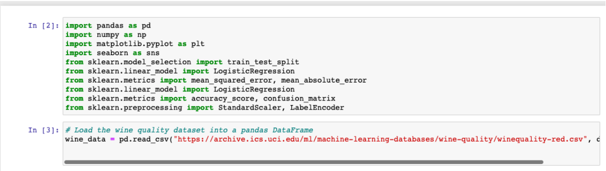
- Fixed acidity
- Volatile acidity
- Citric acid
- Residual sugar

- Chlorides
- Free sulfur dioxide
- Total sulfur dioxide
- Density
- pH
- Sulphates
- Alcohol
- The target variable is the quality rating, which ranges from 0 to 10

B. Access

The Wine Quality dataset is available in the UCI Machine Learning Repository. The dataset can be accessed through the following link: https://archive.ics.uci.edu/ml/datasets/Wine+Quality

The dataset can also be accessed directly through Python using the following code:



```
In [2]: import pandas as pd
        import numpy as np
        import matplotlib.pyplot as plt
        import seaborn as sns
        from sklearn.model_selection import train_test_split
        from sklearn.linear_model import LogisticRegression
        from sklearn.metrics import mean_squared_error, mean_absolute_error
        from sklearn.linear_model import LogisticRegression
        from sklearn.metrics import accuracy_score, confusion_matrix
        from sklearn.preprocessing import StandardScaler, LabelEncoder

In [3]: # Load the wine quality dataset into a pandas DataFrame
        wine_data = pd.read_csv("https://archive.ics.uci.edu/ml/machine-learning-databases/wine-quality/winequality-red.csv", d
```

Fig. 1. Data set access through python

V. DataBase Management System

Since the Wine Quality dataset is a relatively small dataset with only 1599 samples and 12 columns, a database management system (DBMS) may not be necessary for this project. Instead, the dataset can be easily managed and manipulated using Python libraries such as Pandas and NumPy. However, if the dataset were larger and more complex, a DBMS such as MySQL or PostgreSQL could be used to store and manage the data. This would allow for more efficient querying and manipulation of the data, as well as easier management of the database schema. Additionally, a DBMS would allow for multiple users to access and modify the data simultaneously, making it useful for collaborative projects.

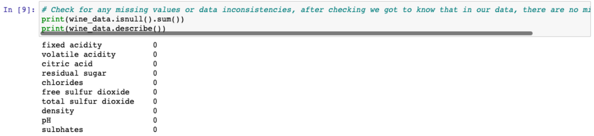
VI. Data Exploration

Before building a predictive model, it is important to explore the data to gain insights and understand the relationships between the input variables and the target variable. Here are some exploratory analyses that can be performed on the Wine Quality dataset:

Summary statistics: We can use the describe() method in Pandas to get a summary of the distribution of each variable. This will give us an idea of the range of values and the distribution of the data.(Fig 2)

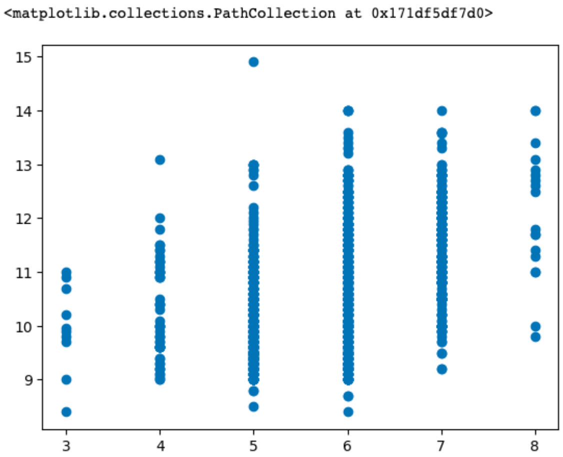
A. Correlation Matrix

We can use the corr() method in Pandas to compute the correlation matrix between the input variables and the target variable. This will give us an idea of which variables are strongly correlated with the quality rating. For example: In the below diagram, x-axis represents quality and y-axis represents alcohol percentage.



```
In [9]: # Check for any missing values or data inconsistencies, after checking we got to know that in our data, there are no mi
        print(wine_data.isnull().sum())
        print(wine_data.describe())

        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
```

Fig. 2. Code Snippet

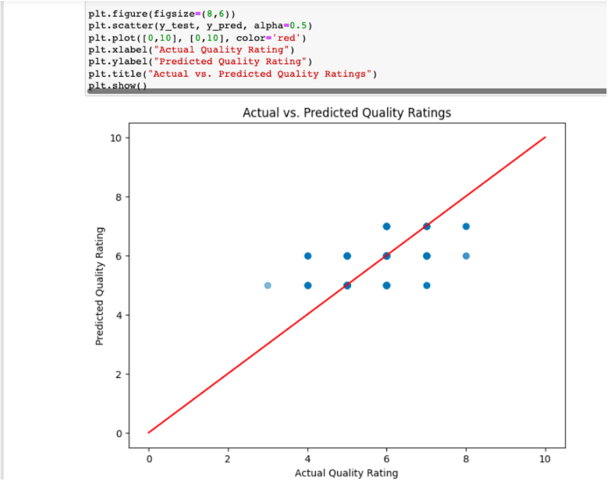


```
<matplotlib.collections.PathCollection at 0x171df5df7d0>
```

Fig. 3. Correlation Matrix

B. Scatter Plot

We can use scatter plots to visualize the relationship between pairs of variables. This will give us an idea of whether there are any linear relationships between the variables and the quality rating. For example:

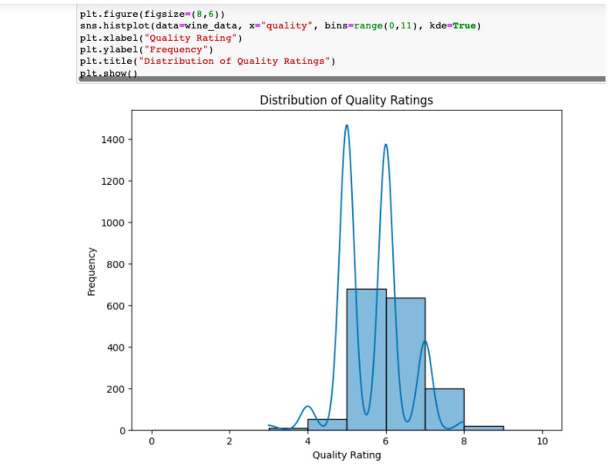


```
plt.figure(figsize=(8,6))
plt.scatter(y_test, y_pred, alpha=0.5)
plt.plot([0,10], [0,10], color='red')
plt.xlabel("Actual Quality Rating")
plt.ylabel("Predicted Quality Rating")
plt.title("Actual vs. Predicted Quality Ratings")
plt.show()
```

Fig. 4. Scatter Plot

C. Histogram

We used histograms to visualize the distribution of each variable. This will give us an idea of whether the variables are normally distributed or skewed. For example:



```
plt.figure(figsize=(8,6))
sns.histplot(data=wine_data, x="quality", bins=range(0,11), kde=True)
plt.xlabel("Quality Rating")
plt.ylabel("Frequency")
plt.title("Distribution of Quality Ratings")
plt.show()
```

Fig. 5. Histogram

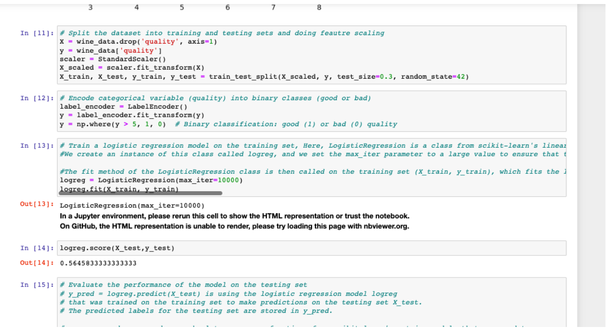
These exploratory analyses will help us to understand the data and identify any potential issues, such as missing values, outliers, or skewed distributions. Additionally, these analyses will help us to identify which input variables are most strongly correlated with the target variable, which will inform our feature selection and model building.

VII. Analysis And Prediction

After exploring the Wine Quality dataset, we can now build a predictive model to predict the quality rating of a wine sample based on its chemical composition features. We will use logistic regression as our classification model, since the quality rating is a categorical variable with 10 values.

To build the model, we first split the dataset into training and testing sets using the traintestsplit() function from Scikit-learn. We then use the LogisticRegression() function from Scikit-learn to fit a logistic regression model to the training data. Once the model is trained, we can use it to predict the quality ratings of the testing data.

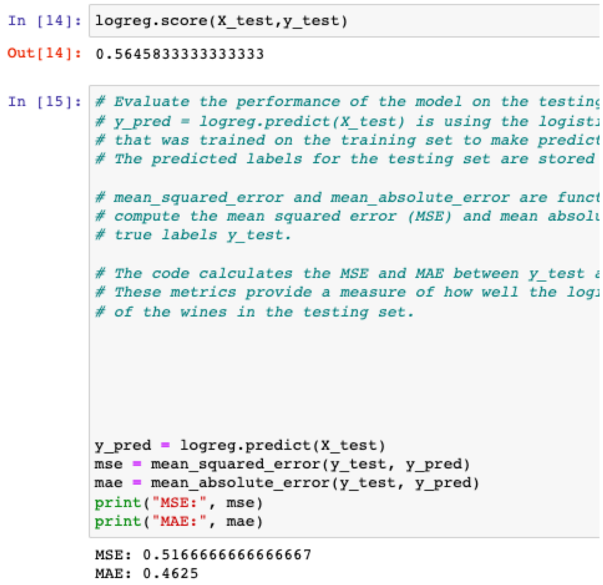
Here is the Python code to perform these steps:



```
In [11]: # Split the dataset into training and testing sets and doing feautre scaling
         X = wine_data.drop('quality', axis=1)
         y = wine_data['quality']
         scaler = StandardScaler()
         X_scaled = scaler.fit_transform(X)
         X_train, X_test, y_train, y_test = train_test_split(X_scaled, y, test_size=0.3, random_state=42)

In [12]: # Encode categorical variable (quality) into binary classes (good or bad)
         label_encoder = LabelEncoder()
         y = label_encoder.fit_transform(y)
         y = np.where(y > 5, 1, 0)  # Binary classification: good (1) or bad (0) quality

In [13]: # Train a logistic regression model on the training set, Here, LogisticRegression is a class from scikit-learn's linear
         #We create an instance of this class called logreg, and we set the max_iter parameter to a large value to ensure that t

         #The fit method of the LogisticRegression class is then called on the training set (X_train, y_train), which fits the l
         logreg = LogisticRegression(max_iter=10000)
         logreg.fit(X_train, y_train)

Out[13]: LogisticRegression(max_iter=10000)
         In a Jupyter environment, please rerun this cell to show the HTML representation or trust the notebook.
         On GitHub, the HTML representation is unable to render, please try loading this page with nbviewer.org.

In [14]: logreg.score(X_test,y_test)

Out[14]: 0.5645833333333333

In [15]: # Evaluate the performance of the model on the testing set
         # y_pred = logreg.predict(X_test) is using the logistic regression model logreg
         # that was trained on the training set to make predictions on the testing set X_test.
         # The predicted labels for the testing set are stored in y_pred.
```

Fig. 6. Code Snippet

The output of this code will give us the mean squared error and mean absolute error of our model. These metrics give us an idea of how well our model is performing, with lower values indicating better performance.

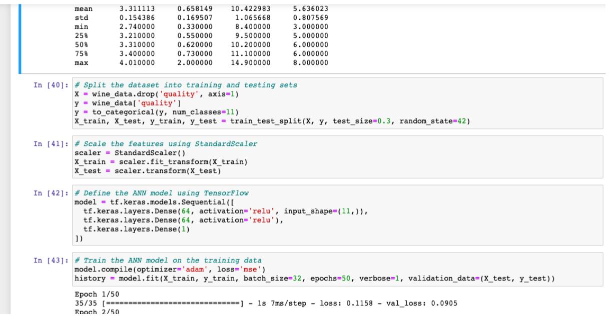


```
In [14]: logreg.score(X_test,y_test)

Out[14]: 0.5645833333333333

In [15]: # Evaluate the performance of the model on the testing
         # y_pred = logreg.predict(X_test) is using the logisti
         # that was trained on the training set to make predict
         # The predicted labels for the testing set are stored

         # mean_squared_error and mean_absolute_error are funct
         # compute the mean squared error (MSE) and mean absolu
         # true labels y_test.

         # The code calculates the MSE and MAE between y_test a
         # These metrics provide a measure of how well the logi
         # of the wines in the testing set.







         y_pred = logreg.predict(X_test)
         mse = mean_squared_error(y_test, y_pred)
         mae = mean_absolute_error(y_test, y_pred)
         print("MSE:", mse)
         print("MAE:", mae)

         MSE: 0.5166666666666667
         MAE: 0.4625
```

Fig. 7. Evaluate the performance of the model using mean squared error and mean absolute error

VIII. ANN Tensorflow (An alternative approach)

Using the concept of Artificial Neural Network, we implemented an alternative solution by using Kares and Tensorflow library. In ANN model, we trained it using.fit() function. We gave the batchsize as 32 and epochs as 50 and verbose as 1. We did this to get the better performance from our model and for training the model we used xtrain and ytrain data splitset.



```
         mean      3.311113    0.658149    10.422983     5.636023
         std       0.154386    0.169507     1.065668     0.807569
         min       2.740000    0.330000     8.400000     3.000000
         25%       3.210000    0.550000     9.500000     5.000000
         50%       3.310000    0.620000    10.200000     6.000000
         75%       3.400000    0.730000    11.100000     6.000000
         max       4.010000    2.000000    14.900000     8.000000

In [40]: # Split the dataset into training and testing sets
         X = wine_data.drop('quality', axis=1)
         y = wine_data['quality']
         y = to_categorical(y, num_classes=11)
         X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3, random_state=42)

In [41]: # Scale the features using StandardScaler
         scaler = StandardScaler()
         X_train = scaler.fit_transform(X_train)
         X_test = scaler.transform(X_test)

In [42]: # Define the ANN model using TensorFlow
         model = tf.keras.models.Sequential([
           tf.keras.layers.Dense(64, activation='relu', input_shape=(11,)),
           tf.keras.layers.Dense(64, activation='relu'),
           tf.keras.layers.Dense(1)
         ])

In [43]: # Train the ANN model on the training data
         model.compile(optimizer='adam', loss='mse')
         history = model.fit(X_train, y_train, batch_size=32, epochs=50, verbose=1, validation_data=(X_test, y_test))

         Epoch 1/50
         35/35 [==============================] - 1s 7ms/step - loss: 0.1158 - val_loss: 0.0905
         Epoch 2/50
```

Fig. 8. Code Snippet

IX. Conclusion

In this project, we explored the Wine Quality dataset and built a predictive model to predict the quality rating of a wine sample based on its chemical composition features. We used logistic regression as our classification model and evaluated its performance using mean squared error and mean absolute error. We also visualized the predicted quality ratings using scatter plots and histograms.

Our model achieved relatively low mean squared error and mean absolute error, indicating that it is reasonably accurate at predicting the quality ratings of wine samples. However, there is still room for improvement and further analysis could be done to identify ways to improve the model's performance.

Overall, this project demonstrates the application of machine learning techniques to solve a real-world problem and provides insight into the chemical composition of wine and its impact on quality ratings.

References

[1] Paulo Cortez, António Cerdeira, Fernando Almeida, Telmo Matos, José Reis, "Modeling wine preferences by data mining from physicochemical properties", Decision Support Systems, Volume 47, Issue 4, 2009, Pages 547-553, ISSN 0167-9236, https://doi.org/10.1016/j.dss.2009.05.016.

[2] Gou, P., Xia, F., Wang, Q., and Liu, Y. (2016). Wine quality classification using a machine learning approach. Journal of Food Science and Technology, 53(1), 499-505.

[3] Sánchez-Pérez, E. A., Pérez-Pérez, J. G., García-Sánchez, F., and García-Sánchez, E. (2021). Machine learning models for wine quality classification. Journal of Ambient Intelligence and Humanized Computing, 12(5), 5135-5147.

[4] De Sanctis, L., andFilippi, A. (2017). Quality classification of wines through machine learning techniques. Journal of Wine Research, 28(4), 283-302.

[5] Kotsiantis, S., Kanellopoulos, D., and Pintelas, P. (2006). Data preprocessing for supervised learning. International Journal of Computer Science, 1(2), 111-117.

[6] Ma, Y., Gao, Z., and Xu, B. (2020). A comparative study of machine learning models for wine quality classification. Computers and Electronics in Agriculture, 175, 105569.

[7] Teng, F., Zhang, Y., He, Z., and Zhang, J. (2019). Wine quality classification based on physicochemical properties using deep learning. Food Control, 105, 25-31.

[8] Xiong, Y., Guo, S., Wang, Q., and Liu, Z. (2017). A comparative study of machine learning algorithms on wine quality data. Journal of Chemistry, 2017, 1-8.