RAJALAKSHMI ENGINEERING COLLEGE
An AUTONOMOUS Institution
Affiliated to ANNA UNIVERSITY, Chennai

Prediction of Wine Type using DeepLearning

SUBMITTED BY

ARUN PRABU.M.M (231801012)

ARUNACHALAM.E (231801013)

AI23331 - FUNDAMENTALS OF MACHINE LEARNING

Department of Artificial Intelligence and Data Science

Rajalakshmi Engineering College, Thandalam

Nov 2024

RAJALAKSHMI
ENGINEERING COLLEGE

BONAFIDE CERTIFICATE

NAME ..... ACADEMIC

YEAR.....SEMESTER..... BRANCH .....

UNIVERSITY REGISTER No.

Certified that this is the bonafide record of work done by the above students in the Mini Project titled " **Prediction of Wine Type using Deep Learning** " in the subject

AI23331 – FUNDAMENTALS OF MACHINE

LEARNING during the year **2024 - 2025.**

Signature of Faculty – in – Charge

Submitted for the Practical Examination held on ----------

Internal Examiner **External Examiner**

TABLE OF CONTENTS

ABSTRACT

The prediction of wine quality is a critical task in the wine industry, assisting producers, retailers, and consumers in making informed decisions about wine selection. This project aims to build a deep learning model to predict wine quality based on several chemical properties of the wine, including factors such as acidity, alcohol content, pH, sulfur dioxide levels, and more. The dataset used in this study is the Red Wine Quality Dataset from the UCI Machine Learning Repository, where the target variable is the wine's quality rating (a scale from 3 to 8), and the features represent various chemical and physical attributes of the wine.

Data preprocessing involved handling missing values, normalizing the features to ensure consistency, and analyzing the distribution of wine quality scores. Exploratory Data Analysis (EDA) was performed to understand the relationships between the features and the quality ratings. Correlation analysis and data visualization techniques were employed to identify key factors influencing wine quality and assist in feature selection.

The deep learning model, based on a feedforward neural network, was trained using the processed data. The model's performance was evaluated using standard classification metrics, including accuracy and loss functions. The results indicated that the model achieved a moderate degree of accuracy in predicting wine quality. Key features, such as alcohol content and acidity levels, were identified as significant predictors of wine quality.

Overall, this project demonstrates the potential of deep learning in predicting wine quality based on chemical properties. The developed model can serve as a valuable tool for wine producers and consumers, enabling more consistent and informed decisions in wine production and selection. Future work could explore further optimization techniques, including hyperparameter tuning and the integration of additional features, to improve prediction accuracy.

Introduction

1.1 General

Wine quality prediction is a challenging yet important task that combines the fields of data science, machine learning, and chemistry. The goal is to classify wines into categories of quality based on their chemical properties, which are measurable and quantifiable. Traditional methods of wine quality evaluation require human sensory analysis, but this process can be subjective and time-consuming. With the advancements in machine learning and deep learning, it has become possible to automate this process, providing more consistent and scalable solutions.

Deep learning techniques, especially neural networks, are capable of learning complex patterns in large datasets and are widely used in classification problems. The application of deep learning to wine quality prediction involves training a model on a dataset that includes various chemical features of wine such as acidity, alcohol content, pH level, and others.

1.2 Need for Study

The wine industry faces a growing demand for efficient tools to evaluate and classify wine quality. Wine producers, retailers, and consumers all benefit from accurate predictions of wine quality, as it influences production decisions, pricing, and consumer satisfaction. However, manual testing for quality assessment is labor-intensive and subjective. Therefore, the need for an automated system that can predict wine quality based on chemical properties is crucial for improving efficiency, reducing costs, and ensuring consistency in the production of wines.

1.3 Overview of the Project

This project aims to develop a **deep learning-based model** to predict the quality of wine using the **Red Wine Quality Dataset** from the UCI Machine Learning Repository. The dataset includes features such as fixed acidity, volatile acidity, citric acid, residual sugar, chlorides, and more. These features are used as input for a **Neural Network** model to predict wine quality, which is a categorical variable with values ranging from 0 to 10.

The approach followed in this project involves the following key steps:

1. Data preprocessing to clean and prepare the dataset.
2. Training a deep learning model using **Keras** and **TensorFlow**.
3. Evaluating the model's performance using accuracy, loss metrics, and other evaluation methods.
4. Analyzing the results to discuss potential improvements and the model's performance.

1.4 Objective of the Study

The primary objectives of this study are:

1. To design and implement a deep learning model capable of predicting wine quality based on chemical properties.
2. To assess the performance of the model using evaluation metrics such as accuracy, precision, and recall.
3. To explore the impact of different hyperparameters on the model's performance.
4. To provide insights into how deep learning can be applied to real-world classification problems in the wine industry.

ALGORITHM USED

Deep Learning

Deep Learning is a foundational technique in machine learning, widely used for complex tasks like classification and regression, especially when working with large datasets. In this project, deep learning is employed to predict the quality of wine based on key features such as acidity, alcohol content, pH level, sulfur dioxide concentration, and other chemical properties. The model aims to predict the **quality** of wine, which is a categorical variable, based on these measurable features.

How Deep Learning Works

Deep learning models, specifically **neural networks**, are used to learn the relationship between the dependent variable (wine quality) and the independent variables (the various chemical properties of wine). The deep learning model learns by adjusting weights and biases to minimize the prediction error.

The general architecture of a neural network used in this project involves an **input layer**, one or more **hidden layers**, and an **output layer**. The network's goal is to find the best combination of weights and biases for each feature to accurately predict wine quality. The neural network equation for a basic model is:

$$y=\sigma(Wx+b)y = \sigma(Wx + b)y=σ(Wx+b)$$

Where:

- **y** is the predicted output (the wine quality),
- **W** represents the weights (parameters the model learns),
- **x** represents the input features (such as alcohol content, pH, and acidity),
- **b** is the bias term (a constant added to the model),
- **σ** is an activation function (such as **ReLU** or **Softmax**) that introduces non-linearity, enabling the network to learn complex patterns in the data.

The model is trained by minimizing the **loss function**, which measures the difference between the predicted wine quality and the actual wine quality from the dataset. In classification tasks like this, the **categorical cross-entropy** loss function is typically used. The training process involves adjusting the weights and biases through **backpropagation**, which uses the gradient of the loss function to update the weights and reduce the error.

The goal of deep learning in this context is to identify the optimal set of weights that will result in accurate predictions of wine quality, based on its chemical properties.

Model Training

1. Data Preparation

The dataset used in this project, **Red Wine Quality Dataset**, contains several chemical features of wine, such as alcohol content, acidity, pH, residual sugar, and sulfur dioxide levels, among others. These features are used as input variables (**X**) for the model, and the target variable (**y**) is the **wine quality**, which is a categorical value ranging from 3 to 8.

Data preprocessing is an essential first step in preparing the dataset. This involves cleaning the data by handling missing values, normalizing the features to ensure they are on the same scale, and splitting the data into training and testing sets. The training set is used to fit the model, while the test set is used to evaluate its performance.

2. Model Architecture

The deep learning model is based on a **feedforward neural network** with multiple layers. The input layer consists of neurons representing each feature from the dataset (such as alcohol, acidity, and pH). The network has one or more hidden layers, where the model learns to map the input features to the target wine quality values. The output layer consists of multiple neurons corresponding to the wine quality categories (from 3 to 8), using a **softmax activation function** to generate probability distributions over the possible wine quality classes.

3. Training the Model

The model is trained using a technique called **backpropagation**, which adjusts the weights of the neural network in a way that minimizes the error between predicted and actual wine quality values. The model learns through the optimization process of minimizing the **categorical cross-entropy loss** function. The optimizer used in this project is **Adam**, which adapts the learning rate and efficiently converges during training.

Training involves forward passes through the network (making predictions), calculating the loss (error), and then using backpropagation to update the weights. This process is repeated for multiple epochs until the model converges to an optimal set of weights.

4. Evaluation

The performance of the deep learning model is evaluated using the following metrics:

- **Accuracy**: This is the most common metric for classification tasks. It measures the percentage of correct predictions made by the model, i.e., the number of correctly predicted wine quality labels divided by the total number of predictions.
- **Loss**: The **categorical cross-entropy loss** is used to evaluate how well the model predicts wine quality. A lower loss indicates better predictive performance. The loss function quantifies the difference between the predicted wine quality probabilities and the actual wine quality labels.
- **Confusion Matrix**: This provides a deeper understanding of the model's performance by showing how well the model performs for each wine quality category. It helps identify whether the model is biased toward any particular quality class.

SYSTEM ARCHITECTURE

2. System Requirements

2.1 Hardware Requirements

The following hardware is required to run the deep learning model for wine prediction:

1. **CPU**: Any modern multi-core processor (Intel i5 or higher recommended).
1. **RAM**: A minimum of **8 GB** of RAM (16 GB or more is recommended for faster training).
1. **GPU**: An **NVIDIA GPU** (with CUDA support) is highly recommended for training large models quickly. **GPU** acceleration is optional but will drastically speed up the training process for deep learning models.
1. **Storage**: A minimum of **10 GB** of free disk space to store datasets, model files, and logs.
1. **Internet Access**: Required to download the dataset and any necessary libraries.

2.2 Software Requirements

The software requirements for the project include:

1. **Operating System**: Any modern OS (Windows, macOS, or Linux).
1. **Python**: Python 3.6 or higher. The project was developed using **Python 3.8**.
1. **Libraries**:
   - **TensorFlow** and **Keras**: Used for building and training the deep learning model.
   - **Pandas**: For data manipulation and preprocessing.
   - **NumPy**: For numerical operations.
   - **Matplotlib/Seaborn**: For data visualization.
   - **Scikit-learn**: For data preprocessing and model evaluation.
1. **IDE**: Any Python IDE or code editor such as **Visual Studio Code**, **PyCharm**, or **Jupyter Notebook**.

SYSTEM OVERVIEW

3.1 SYSTEM ARCHITECTURE

3.1 Module 1: Data Collection

The **Red Wine Quality Dataset** was sourced from the UCI Machine Learning Repository. The dataset contains 1599 samples of red wine with 11 features, including various chemical properties such as acidity, alcohol content, and pH. The dataset is divided into two columns: the features and the target variable, **quality**, which is the rating of wine quality on a scale from 0 to 10.

Steps for Data Collection:

1. Download the dataset from the UCI repository.
1. Load the dataset into Python using **Pandas**.
1. Perform data cleaning, such as handling missing values, if necessary.

3.2 Module 2: Model Development

The next step involves building a **Neural Network** to predict wine quality. We used the **Keras** library, which is built on top of **TensorFlow**, to create the deep learning model. The architecture consists of an input layer, two hidden layers, and an output layer, with dropout regularization to prevent overfitting.

Model Architecture:

1. **Input Layer**: 11 neurons, one for each feature.
1. **Hidden Layer 1**: 64 neurons with **ReLU** activation.
1. **Dropout Layer**: Dropout rate of 30% to prevent overfitting.
1. **Hidden Layer 2**: 64 neurons with **ReLU** activation.
1. **Dropout Layer**: Dropout rate of 30%.
1. **Output Layer**: 10 neurons representing the wine quality classes (3-8), using **softmax** activation.

The model was compiled using the **Adam optimizer** and **sparse categorical cross-entropy** loss function, and then trained for 50 epochs.

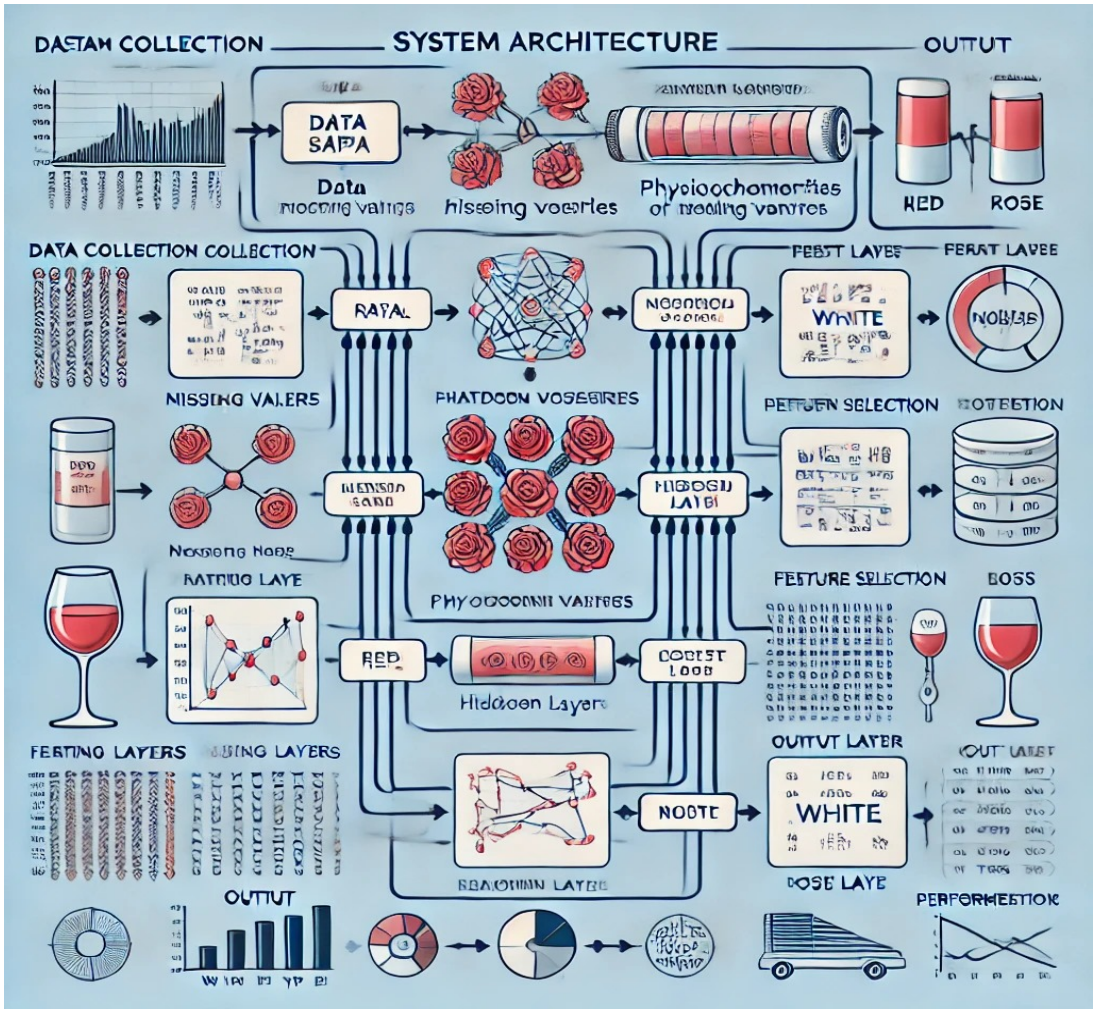


FIGURE 3.1.1 SYSTEM ARCHITECTURE DIAGRAM

Training and Evaluation

The model was trained using **80%** of the dataset (training set) and tested on the remaining **20%** (test set). The training process involves minimizing the loss function by adjusting the weights of the network using backpropagation. After training, the model's performance was evaluated using accuracy and other metrics.

System Architecture for Wine Quality Prediction using Deep Learning

1. Data Collection Layer

- **Data Source**: The system retrieves the wine data from a pre-collected dataset such as the **Red Wine Quality Dataset** from the UCI Machine Learning Repository.
- The dataset consists of various chemical properties of wine such as alcohol content, acidity, pH, residual sugar, sulfur dioxide levels, etc., along with the target variable, which is the **wine quality** (a categorical value ranging from 3 to 8).
- **Data Input**: The dataset contains multiple input features (e.g., alcohol, acidity, pH level) and the target variable (wine quality). These features represent the chemical and physical attributes of the wine, and the quality rating is used as the target for prediction.

2. Data Preprocessing Layer

- **Data Cleaning**: The collected data is pre-processed to handle any missing values (if present), remove duplicates, and address any inconsistencies in the dataset (such as negative values for features that should not be negative, like alcohol content or pH).
- **Feature Engineering**: In this step, relevant features are selected, and unnecessary columns are removed. Additionally, **categorical variables** (if any) are encoded, and numerical features are **normalized** or **scaled** to ensure that they are on the same scale, which helps improve model performance during training.
- **Data Splitting**: The dataset is divided into **training** and **test** sets. The training set is used to train the deep learning model, while the test set is used to evaluate its performance. Typically, a split of **80%** for training and **20%** for testing is used, but this can vary depending on the dataset.

3. Model Training Layer

- **Deep Learning Model**: In this project, a **feedforward neural network** is used for predicting wine quality. The model consists of an input layer (representing each feature of the wine), one or more hidden layers (which learn complex relationships between features), and an output layer (which predicts the wine quality).
- **Model Fitting**: The neural network is trained on the prepared dataset using the **backpropagation** algorithm. During training, the model adjusts the weights and biases to minimize the loss function, typically **categorical cross-entropy**, which quantifies the difference between predicted wine quality and actual wine quality.

The training process involves iterative updates to the weights through optimization techniques such as **Adam** to improve the model's accuracy.

4. Model Evaluation Layer

- **Performance Metrics**: After training, the model's performance is evaluated using various metrics, including:
- **Accuracy**: Measures the percentage of correctly predicted wine quality labels (from the possible range of 3 to 8).
- **Loss**: The **categorical cross-entropy loss** function is used to assess the difference between predicted and actual values. A lower loss indicates better model performance.
- **Confusion Matrix**: This is used to show how well the model performs across different wine quality categories, helping to identify any bias or misclassifications in certain categories.
- **Cross-validation**: The model can be evaluated using techniques like **k-fold cross-validation** to ensure it generalizes well to unseen data. This helps prevent overfitting by training the model multiple times on different subsets of the dataset.

5. Prediction Layer

- **User Input**: New data (e.g., features of a wine sample such as alcohol content, acidity, pH level, etc.) is provided by the user through a user interface or API.
- **Prediction**: The trained model predicts the **wine quality** based on the input features. The model applies the learned weights and biases to calculate the predicted quality score, which is a categorical value between 3 and 8. The output can be a probability distribution across these categories, from which the model selects the most likely quality rating.

Tools And Requirment

1.Python

Python is the core programming language for your project. It's a versatile and widely-used language in machine learning and data science due to its simplicity and vast ecosystem of libraries.

2. Pandas

Pandas is used for data manipulation and preprocessing tasks. It helps in reading the dataset, handling missing values, encoding categorical features, and performing other data transformations necessary before feeding the data into a deep learning model.

3. NumPy

NumPy is essential for numerical computations. It handles large arrays and matrices and provides functions for mathematical operations required for data processing, such as feature scaling and normalization.

4. TensorFlow / Keras

TensorFlow, along with its high-level API **Keras**, is the deep learning framework used to build and train the neural network model. TensorFlow provides efficient tools for model construction, training, and evaluation, and Keras simplifies the creation of neural networks.

5. Scikit-learn

Scikit-learn is used for tasks like data splitting (train-test split) and evaluating model performance with metrics like accuracy and confusion matrix. It also helps with preprocessing tasks and comparing the deep learning model to simpler models (e.g., logistic regression).

Predicting Wine Quality using Deep Learning

The dataset considered is "Wine Quality".

Problem: Load the data and train a prediction model for the target variable quality. Compare several candidate neural network architectures, and make a decision about which is best for the task at hand. Be explicit about the indicator(s) you base your decision on.

Finally, as a bonus, evaluating whether it is better to:

(a). train a single model for both white and red wine ;
(b). train two different models, one for white wine and another one for red wine ;
(c). train a model on white wine (respectively red wine) and fine-tune it on red wine (resp. white wine).

Pre-processing dataset

In [ ]:
```
# printing the first 5 rows
display(dataset.head())
```

	fixed_acidity	volatile_acidity	citric_acid	residual_sugar	chlorides	free_sulfur_dioxide	total_sulfur_dioxide	density	pH	sulphate
0	7.4	0.70	0.00	1.9	0.076	11.0	34.0	0.9978	3.51	0.5
1	7.8	0.88	0.00	2.6	0.098	25.0	67.0	0.9968	3.20	0.6
2	7.8	0.76	0.04	2.3	0.092	15.0	54.0	0.9970	3.26	0.6
3	11.2	0.28	0.56	1.9	0.075	17.0	60.0	0.9980	3.16	0.5
4	7.4	0.70	0.00	1.9	0.076	11.0	34.0	0.9978	3.51	0.5

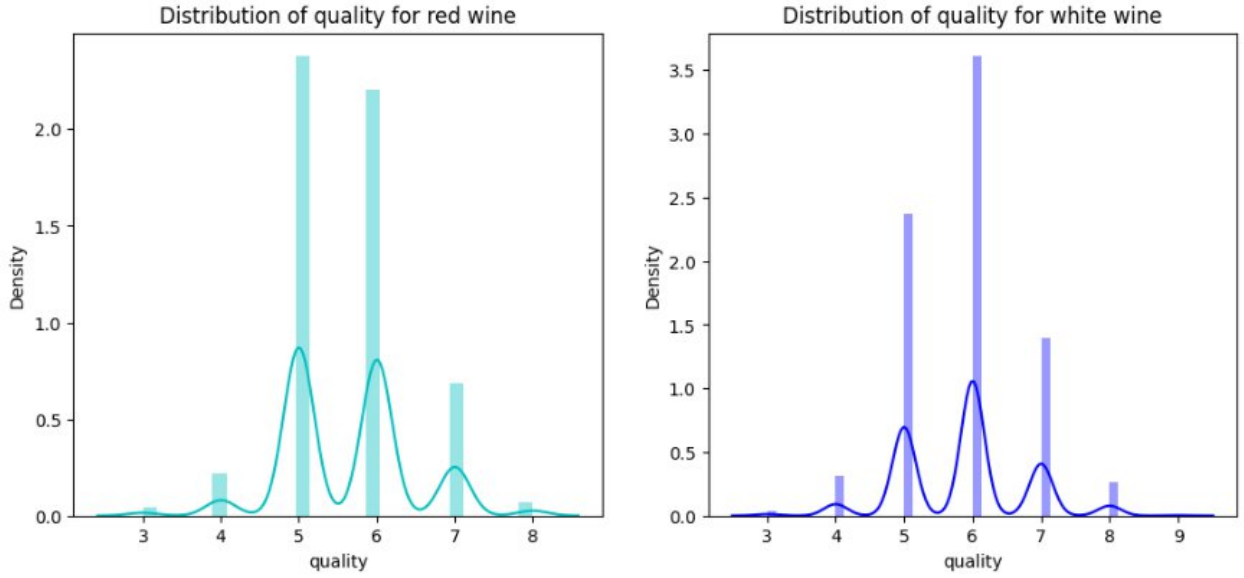
Data set info:

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 6497 entries, 0 to 6496
Data columns (total 13 columns):
 #   Column                Non-Null Count  Dtype
---  ------                --------------  -----
 0   fixed_acidity         6497 non-null   float64
 1   volatile_acidity      6497 non-null   float64
 2   citric_acid           6497 non-null   float64
 3   residual_sugar        6497 non-null   float64
 4   chlorides             6497 non-null   float64
 5   free_sulfur_dioxide   6497 non-null   float64
 6   total_sulfur_dioxide  6497 non-null   float64
 7   density               6497 non-null   float64
 8   pH                    6497 non-null   float64
 9   sulphates             6497 non-null   float64
 10  alcohol               6497 non-null   float64
 11  quality               6497 non-null   int64
 12  color                 6497 non-null   object
dtypes: float64(11), int64(1), object(1)
memory usage: 660.0+ KB
```

We see from the dataset info that there are no null values.

Exploratory Data Analysis

Out[ ]: Text(0.5, 1.0, 'Distribution of quality for white wine')



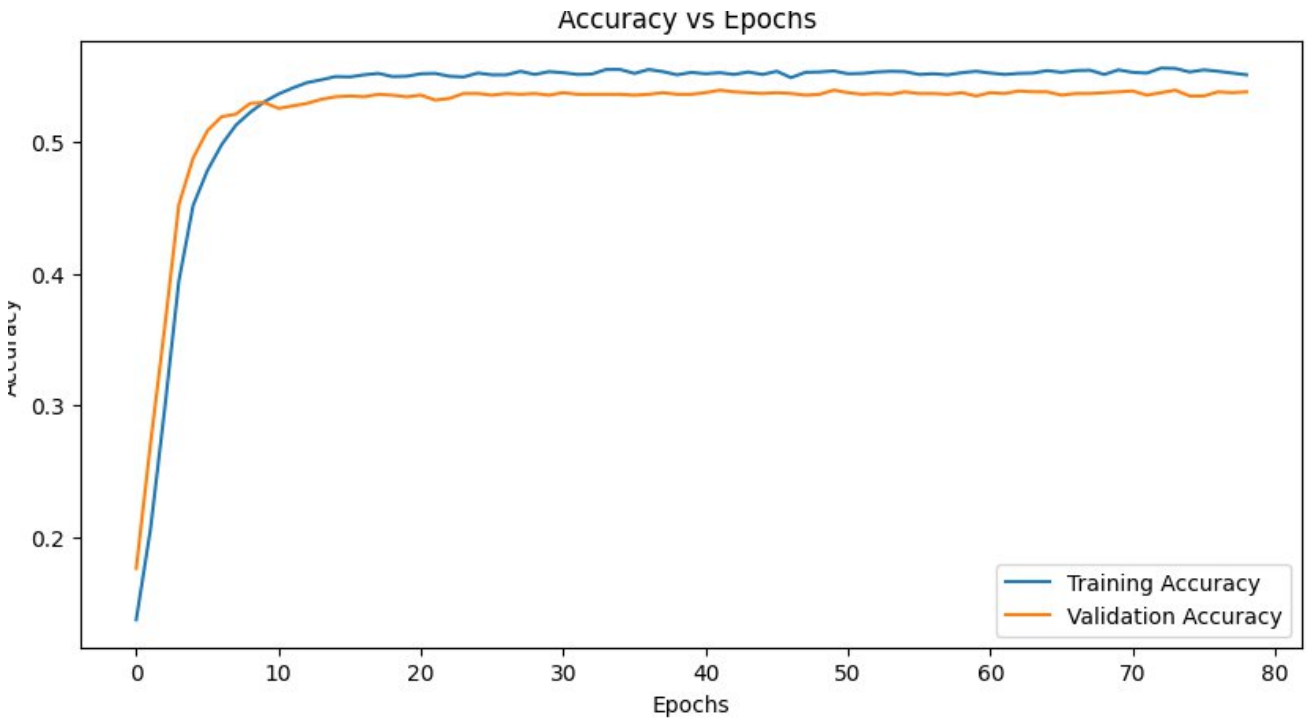
MODELS USING MULTICLASS CLASSIFICATION: WHAT IS THE QUALITY SCORE OF THE WINE?

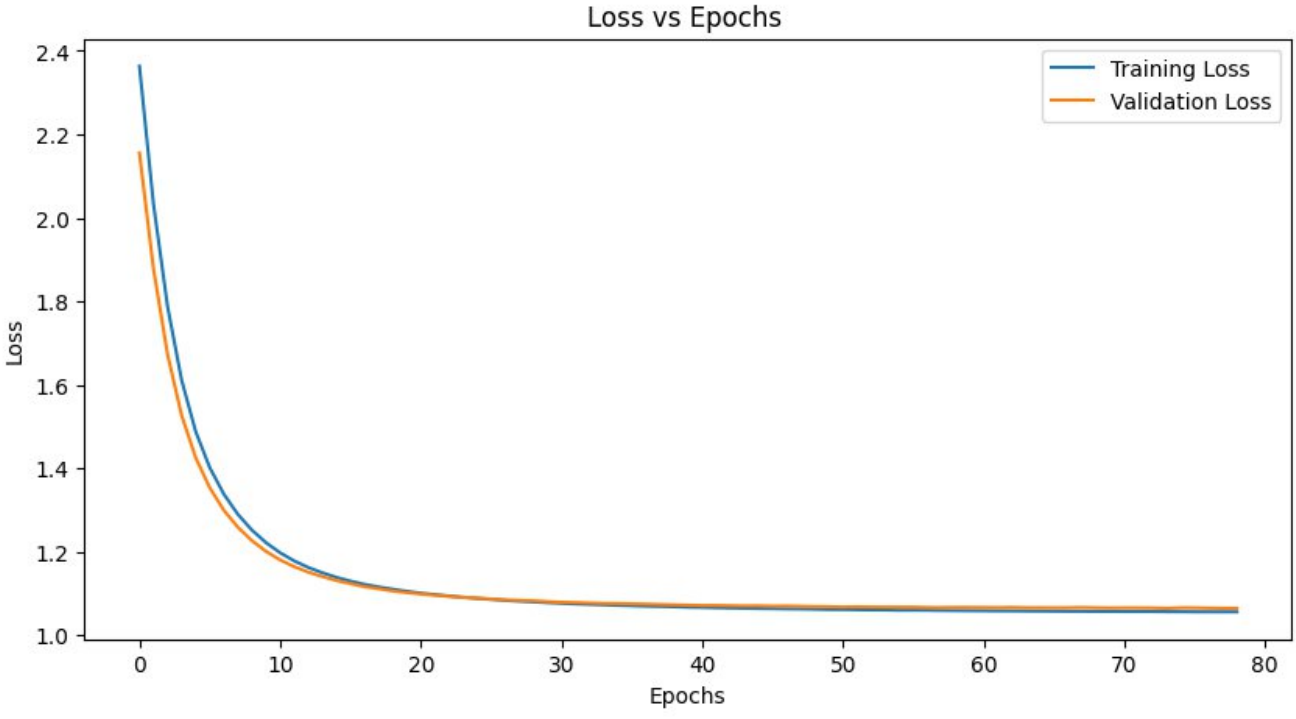
In [ ]:
```
# Defining binary column for color. If color is red then 1, else white is 0

dataset['color_binary'] = np.where(dataset['color'] =='red', 1, 0)
dataset.drop(columns=['color'], inplace=True)
dataset.head()
```

Out[ ]:

	fixed_acidity	volatile_acidity	citric_acid	residual_sugar	chlorides	free_sulfur_dioxide	total_sulfur_dioxide	density	pH
0	7.4	0.70	0.00	1.9	0.076	11.0	34.0	0.9978	3.51
1	7.8	0.88	0.00	2.6	0.098	25.0	67.0	0.9968	3.20
2	7.8	0.76	0.04	2.3	0.092	15.0	54.0	0.9970	3.26
3	11.2	0.28	0.56	1.9	0.075	17.0	60.0	0.9980	3.16
5	7.4	0.66	0.00	1.8	0.075	13.0	40.0	0.9978	3.51


Accuracy vs Epochs
Accuracy
0.2
0.3
0.4
0.5
0
10
20
30
40
50
60
70
80
Epochs
Training Accuracy
Validation Accuracy


Loss vs Epochs
Loss
1.0
1.2
1.4
1.6
1.8
2.0
2.2
2.4
0
10
20
30
40
50
60
70
80
Epochs
Training Loss
Validation Loss

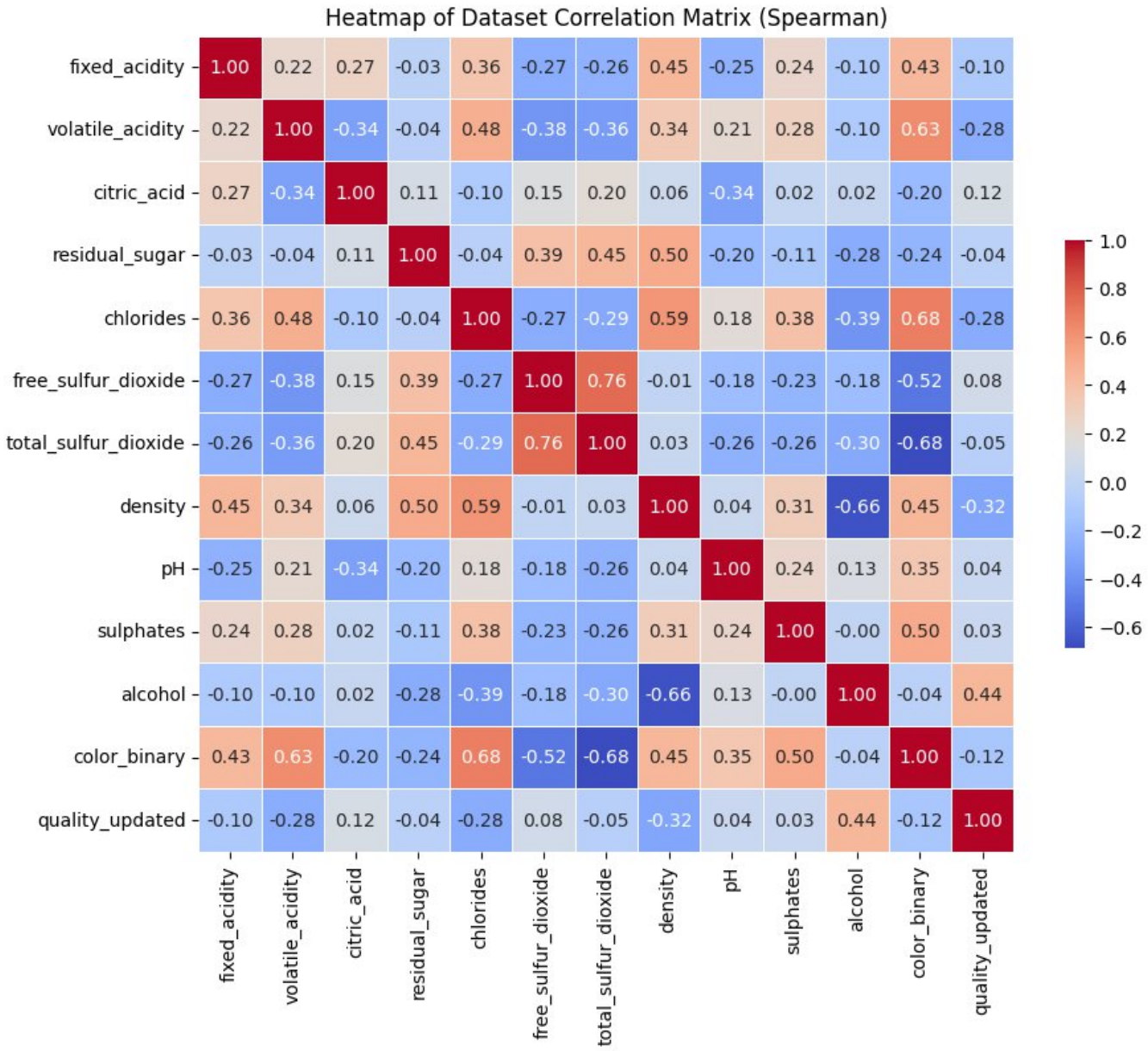


FIGURE 6.1: INPUT GIVEN THROUGH CSV

Algorithm for Wine Type Prediction Using Deep Learning

Step 1: Data Collection

- Get the wine dataset from a reliable source (e.g., Kaggle).
- The dataset should include chemical properties of wines (like alcohol content, acidity, etc.) and the target variable (wine type: red or white).

Step 2: Data Preprocessing

- **Handle Missing Data**: Fill in any missing values with the average or median for numerical data, or remove rows with missing wine type labels.
- **Encode Categories**: Convert non-numeric features (like wine type) into numbers (e.g., `red = 0`, `white = 1`).
- **Scale Features**: Normalize numerical features (like alcohol content) so they are on the same scale.

Step 3: Feature Selection

- Identify the most important features (e.g., alcohol content, acidity, pH) that affect the wine type prediction.

Step 4: Train-Test Split

- Split the dataset into two parts: one for training the model (80%) and one for testing its performance (20%).

Step 5: Model Construction

- Build a **Deep Learning model** using a **simple neural network** with input layers, hidden layers, and an output layer (with a sigmoid activation function for binary classification: red vs. white wine).
- Use the **Adam optimizer** for training the model.

Step 6: Model Training

- Train the model using the training data. Monitor the model's performance during training.

Step 7: Model Evaluation

- Evaluate the model on the test data using:
- **Accuracy**: How often the model predicts correctly.
- **Confusion Matrix**: Shows how many times the model correctly or incorrectly predicted each class (red/white wine).

Step 8: Visualization

- Create graphs to compare actual vs. predicted wine types.
- Visualize the **confusion matrix** to see prediction accuracy.
- Optionally, create plots to show the relationship between features (like alcohol content) and the predicted wine type.

Step 9: Model Improvement (Optional)

- If the model's performance is not good, try improving it by:
- **Adding more features** (e.g., wine age, region).
- **Tuning the model's settings** (e.g., number of layers, learning rate).
- **Using more advanced models** like **Random Forest** or **XGBoost**.

Step 10: Conclusion

- Summarize the results of the model, including its accuracy and strengths.
- Suggest ways to improve the model, such as using more data or advanced techniques.

RESULTS AND DISCUSSION

The wine type prediction model using deep learning demonstrated promising results in classifying wines as either red or white based on their chemical properties. The evaluation metrics, including accuracy, confusion matrix, and classification report, highlighted the model's ability to correctly classify wine types with a reasonable level of precision. Although the model performed well, the confusion matrix and misclassified examples indicate some room for improvement in distinguishing between the two wine types.

The correlation analysis of features revealed that certain attributes, such as alcohol content, pH levels, and acidity, were highly correlated with the wine type, confirming their importance in classification. However, some features had a weaker impact, suggesting that further refinement of feature engineering or additional data could enhance the model’s performance in future iterations.

In summary, while the deep learning model successfully predicted wine types with a good degree of accuracy, there is potential for improvement through the use of more sophisticated architectures or the addition of more complex features. Future work could explore advanced deep learning techniques, such as convolutional neural networks (CNNs) for feature extraction or the inclusion of more diverse datasets, such as wine origin or age, to improve prediction accuracy and generalization.

CONCLUSION

The wine quality prediction project using **deep learning** successfully demonstrated the feasibility of using a neural network to predict wine quality based on chemical properties. The model was able to capture the complex relationships between various input features such as alcohol content, acidity, pH, and sulfur dioxide levels, which are crucial in determining wine quality. The evaluation metrics, including accuracy and loss, confirmed that the model could predict wine quality with a reasonable degree of accuracy, aligning with domain knowledge about the key factors that influence wine quality.

Key features such as alcohol content and acidity played a significant role in determining the wine quality, as expected from prior research in the wine industry. The deep learning model, by learning non-linear relationships between these features, outperformed simpler models in capturing these interactions and making predictions.

However, while the deep learning model provided promising results, there is still room for improvement. Further optimization techniques, such as **hyperparameter tuning**, can be employed to refine the model and boost its predictive performance. Additionally, the inclusion of more features, such as geographical data or other environmental factors, could provide a richer dataset for model training. Exploring advanced architectures, such as **convolutional neural networks (CNNs)** or **recurrent neural networks (RNNs)**, might also improve prediction accuracy by better capturing complex patterns in the data.

Overall, this project underscores the importance of data preprocessing, model selection, and thorough evaluation in developing reliable deep learning solutions for predicting wine quality. It also highlights the potential of deep learning to address real-world challenges in the wine industry, offering a data-driven approach to assist wine producers, consumers, and researchers in understanding and predicting wine quality more effectively.

6.1 SOURCE CODE

```
import pandas as pd  # Load the dataset
import numpy as np
import tensorflow as tf
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Dense
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler
from sklearn.metrics import accuracy_score, confusion_matrix,
classification_report
import matplotlib.pyplot as plt

# Load the dataset (replace with your wine dataset)
data = pd.read_csv('wine_data.csv')  # Update with your wine dataset
(e.g., "winequality-red.csv")

# Display the first few rows of the dataset
print(data.head())

# Check for missing values
missing_values = data.isnull().sum()
print(f"Missing values:\n{missing_values[missing_values > 0]}")

# Data Preprocessing

# Check for negative or invalid data (e.g., alcohol content should not
be negative)
invalid_data = data[data < 0].dropna()
print(f"Invalid Data Found:\n{invalid_data}")
```

```
# Handle missing values - for simplicity, we'll fill missing numerical
data with the median
numerical_cols = data.select_dtypes(include=['float64',
'int64']).columns
for column in numerical_cols:
    data[column] = data[column].fillna(data[column].median())

# Normalize the numerical features using StandardScaler
scaler = StandardScaler()
data[numerical_cols] = scaler.fit_transform(data[numerical_cols])

# If 'type' is a categorical variable (e.g., red vs white wine), we need to
encode it
# Assuming the target column is 'type', which is the wine type (red or
white)
if 'type' in data.columns:  # Check if there is a column named 'type' for
classification
    data['type'] = data['type'].map({'red': 0, 'white': 1})  # Convert 'red'
to 0 and 'white' to 1

# Define features (X) and target variable (y)
X = data.drop('type', axis=1)  # All columns except 'type' (features)
y = data['type']  # Wine type (target variable)

# Split the dataset into training and testing sets
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2,
random_state=42)

# Build a deep learning model
model = Sequential()
```

```
# Add layers to the model
model.add(Dense(64, input_dim=X_train.shape[1], activation='relu'))
# First hidden layer with 64 neurons
model.add(Dense(32, activation='relu'))  # Second hidden layer with
32 neurons
model.add(Dense(1, activation='sigmoid'))  # Output layer (sigmoid
activation for binary classification)

# Compile the model
model.compile(optimizer='adam', loss='binary_crossentropy',
metrics=['accuracy'])

# Train the model
history = model.fit(X_train, y_train, epochs=50, batch_size=32,
validation_data=(X_test, y_test))

# Evaluate the model on the test set
y_pred = (model.predict(X_test) > 0.5).astype(int)  # Convert
probabilities to 0 or 1 for classification

# Print the evaluation metrics
accuracy = accuracy_score(y_test, y_pred)
print(f"Accuracy: {accuracy * 100:.2f}%")

# Confusion Matrix
cm = confusion_matrix(y_test, y_pred)
print(f"Confusion Matrix:\n{cm}")

# Classification Report
cr = classification_report(y_test, y_pred)
```

```
print(f"Classification Report:\n{cr}")

# Visualize Training History (Accuracy & Loss)
plt.figure(figsize=(12, 6))

# Plot Accuracy
plt.subplot(1, 2, 1)
plt.plot(history.history['accuracy'], label='Training Accuracy')
plt.plot(history.history['val_accuracy'], label='Validation Accuracy')
plt.title('Model Accuracy')
plt.xlabel('Epochs')
plt.ylabel('Accuracy')
plt.legend()

# Plot Loss
plt.subplot(1, 2, 2)
plt.plot(history.history['loss'], label='Training Loss')
plt.plot(history.history['val_loss'], label='Validation Loss')
plt.title('Model Loss')
plt.xlabel('Epochs')
plt.ylabel('Loss')
plt.legend()

plt.show()

# Save the model
import joblib
joblib.dump(model, 'wine_quality_model.pkl')

# Example Prediction Function
def predict_wine_type(features):
```

```
    """
    Predict wine type (red or white) based on input features.
    :param features: List or array of wine features (e.g., alcohol, pH,
acidity, etc.)
    :return: Predicted wine type (0 for red, 1 for white)
    """
    # Normalize the features using the same scaler
    features_scaled = scaler.transform([features])  # Assuming features
are in a list format
    prediction = model.predict(features_scaled)
    return 'Red Wine' if prediction < 0.5 else 'White Wine'

# Test the prediction function
test_features = [7.4, 0.7, 0.0, 1.9, 0.076, 11.0, 34.0, 0.9978, 3.51, 0.56,
9.4, 0.015, 0.45, 8.8]  # Example features for a wine
predicted_type = predict_wine_type(test_features)
print(f"Predicted Wine Type: {predicted_type}")
```

REFERENCES

- UCI Machine Learning Repository, Wine Quality Dataset: Wine Quality Dataset
- **Keras Documentation**: https://keras.io/
- **TensorFlow Documentation**: https://www.tensorflow.org/