WINE QUALITY PREDICTION USING MACHINE LEARNING

An Application Development – 2 (Project) Report Submitted
In partial fulfillment of the requirement for the award of the degree of

Bachelor of Technology
in
Computer Science and Engineering (Data Science)
by

ENUMULA MANOJ KUMAR	**21N31A6713**
GADIDAS THARUN	**21N31A6714**
GAJJELA SANJAY	**21N31A6715**

Under the guidance of

A.NAVEEN KUMAR
Asst.Professor
Department of Emerging Technologies
MRCET (Autonomous Institution, UGC Govt. of India)

MRCET CAMPUS

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING
(EMERGING TECHNOLOGIES)
MALLA REDDY COLLEGE OF ENGINEERING AND TECHNOLOGY
(Autonomous Institution - UGC, Govt. of India)
(Affiliated to JNTU, Hyderabad, Approved by AICTE, Accredited by NBA & NAAC – 'A' Grade, ISO 9001:2015 Certified)
Maisammaguda (v), Near Dullapally, Via: Kompally, Hyderabad – 500 100, Telangana State, India

2023-2024

MALLA REDDY COLLEGE OF ENGINEERING & TECHNOLOGY
(Autonomous Institution – UGC, Govt. of India)
(Sponsored by CMR Educational Society)
Recognized under 2(f) and 12 (B) of UGC ACT 1956
Estd : 2004
(Affiliated to JNTUH, Hyderabad, Approved by AICTE- Accredited by NBA & NAAC– 'A' Grade - ISO 9001:2015 Certified)

CERTIFICATE

This is to certify that this is the Bonafide record of the project titled "**Wine Quality Prediction Using Machine Learning**", submitted by **E. MANOJ KUMAR** (21N31A6713) **G. THARUN (**21N31A6714) and **G. SANJAY (**21N31A6715) of **B.Tech III YEAR – II Semester** in the partial fulfillment of the requirements for the degree of **Bachelor of Technology** in **Computer Science and Engineering (Data Science)**, Dept. of CSE (Emerging Technologies) during the year 2023-2024. The results embodied in this project report have not been submitted to any other university or institute for the award of any degree or diploma.

Project Guide
Department of CSE(ET)

Project Coordinator
Department of CSE (ET)

Overall Project Coordinator
Department of CSE(ET)

HEAD OF THE DEPARTMENT
(Emerging Technology)

EXTERNAL EXAMINER

Date of Viva-Voce Examination Held On:_____

DECLARATION

We hereby declare that the project entitled "**Wine Quality Prediction Using Machine Learning**" submitted to **Malla Reddy College of Engineering and Technology,** affiliated to Jawaharlal Nehru Technological University Hyderabad (JNTUH) as part of III Year B.Tech – II Semester and for the partial fulfillment of the requirement for the award of **Bachelor of Technology** in **Computer Science and Engineering (Data Science)** is a result of original research work done by me.

It is further declared that the project report or any part thereof has not been previously submitted to any University or Institute for the award of degree or diploma.

G. SANJAY - 21N31A6715

G. THARUN - 21N31A6714

E. MANOJ KUMAR - 21N31A6713

ACKNOWLEDGEMENTS

We feel ourself honored and privileged to place our warm salutation to our college "Malla Reddy College of Engineering and Technology (Autonomous Institution – UGC Govt. of India) and our Principal **Dr. S Srinivasa Rao,** Professor who gave us the opportunity to do the Application Development -1 (Project) during our III Year B.Tech and profound the technical skills.

We express our heartiest thanks to our Director **Dr. V S K Reddy,** Professor for encouraging us in every aspect of our project and helping us realize our full potential.

We also thankful to our Head of the Department **Dr. M V Kamal,** Professor for providing training and guidance, excellent infrastructure and a nice atmosphere for completing this project successfully.

We would like to express our sincere gratitude and indebtedness to our project supervisor **Dr. M Gayatri,** Project Coordinator for his valuable suggestions and interest throughout the course of this project.

We convey our heartfelt thanks to our over all Project Coordinator
Dr. P Dileep, Professor for allowing for their regular guidance and constant encouragement during our dissertation work.

We would like to express our sincere gratitude and indebtedness to our project supervisor **A Naveen Kumar,** Asst. professor for his valuable suggestions and interest throughout the course of this project.

We would like to thank all our supporting **staff** of the Department of CSE (Emerging Technologies) and even all other department who have been helpful directly and in-directly in making our project a success.

Finally, we would like to take this opportunity to thank our **families** for their support and blessings for completion of our project that gave us the strength to do our project.

E.MANOJKUMAR (21N31A6713)
G.THARUN (21N31A6714)
G.SANJAY (21N31A6715)

ABSTRACT

- This study focuses on predicting the quality of wine using various machine learning algorithm implemented in Python. Wine quality assessment is a crucial task in the wine industry, as it helps in maintaining and improving the standards of wine production, this research aims to provide winemakers with an efficient tool to evaluate and enhance wine quality.

- The dataset used for this analysis comprises various physicochemical properties of wines, such as acidity levels, residual sugar, alcohol content, and pH, along with their corresponding quality ratings. Initially, exploratory data analysis (EDA) techniques are employed to gain insights into the dataset's characteristics and distributions.

- Several machine learning algorithms including Decision Trees, Random Forest, Support Vector Machines (SVM) are implemented to build predictive models. In conclusion, this research demonstrates the feasibility and efficacy of utilizing machine learning techniques in Python for wine quality prediction.

TABLE OF CONTENTS

CHAPTER 1

INTRODUCTION

- In the realm of winemaking, ensuring high-quality products is paramount. Every step contributes to the final taste and character of the wine, it's now possible to augment these traditional methods with data-driven approaches for more accurate and efficient quality prediction.

- The goal of this study is to demonstrate how ML algorithms can be utilized to predict wine quality reliably, providing winemakers with valuable insights for process optimization and quality control

- Several machine learning algorithms including Decision Trees, Random Forest and Support Vector Machines (SVM) are implemented to build predictive models

- Overall, this introduction lays the foundation for understanding the subsequent sections, results of wine quality prediction using ML and Python.

1.1PROBLEM DEFINITION

- **Objective:** The primary objective is to develop a machine learning model that can predict the quality of wine accurately based on its characteristics. The quality of wine is often rated on a scale, typically from 1 to 10 or from 0 to 10.

- **Input Data:** The input data consists of various features that describe the properties of the wine. These features may include:

- **Output:** The output is the predicted quality of the wine. This can be a numerical value indicating the quality score or a categorical label (e.g., low, medium, high)

- **Data Collection**: Gathering a dataset containing information about different wines along with their quality ratings. This dataset should be sufficiently large and diverse to ensure that the model generalizes well to new, unseen data.

- **Data Preprocessing:** This step involves cleaning the data (handling missing values, outliers, etc.), and possibly transforming features (normalization, standardization, encoding categorical variables, etc.) to prepare it for the machine learning algorithm

- **Model Selection:** Choosing appropriate machine learning algorithms for the task. This could include regression algorithms (for predicting a numerical quality score) or classification algorithms

- **Model Training:** Training the selected machine learning model on the preprocessed data. This involves splitting the data into training and testing sets, and then using the training set to teach the model to make accurate predictions.

- **Model Evaluation:** Assessing the performance of the trained model using appropriate evaluation metrics. For regression tasks, metrics like MSE, RMSE , or R-squared can be used..

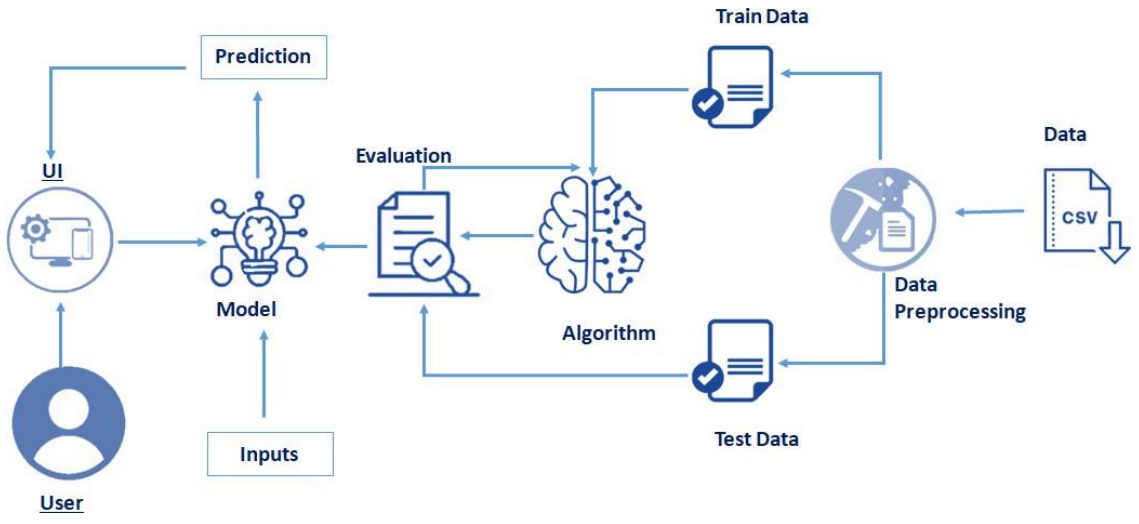
- **Deployment:** Deploying the trained model into production, where it can be used to predict the quality of new, unseen wines. This may involve building an application interface or integrating the model into an existing system

CHAPTER 2

2.1 EXISTING SYSTEM

- Existing systems for wine quality prediction using machine learning (ML) and Python have garnered significant attention due to their potential to revolutionize the wine industry. These systems leverage ML algorithms to analyze various physicochemical properties of wines and predict their quality ratings. Here are some key components and approaches commonly found in existing systems:

1. Data Collection and Preprocessing
2. Exploratory Data Analysis (EDA)
3. Feature Engineering
4. Model Selection and Training
5. Evaluation Metrics
6. Deployment and Integration
7. Continuous Improvement and Monitoring



ww

2.2 PROPOSED SYSTEM

- The proposed system, Wine Predict, aims to provide winemakers with a powerful tool for accurately predicting wine quality based on physicochemical properties. By leveraging machine learning techniques in Python, Wine Predict offers a data-driven solution to optimize production processes, improve quality control measures, and ultimately enhance the overall quality of wines.

- Evaluate various ML algorithms including Decision Trees, Random Forest, Support Vector Machines (SVM), Gradient Boosting, and Neural Networks.

	count	mean	std	min	25%	50%	75%	max
fixed acidity	6487.0	7.216579	1.296750	3.80000	6.40000	7.00000	7.70000	15.90000
volatile acidity	6489.0	0.339691	0.164649	0.08000	0.23000	0.29000	0.40000	1.58000
citric acid	6494.0	0.318722	0.145265	0.00000	0.25000	0.31000	0.39000	1.66000
residual sugar	6495.0	5.444326	4.758125	0.60000	1.80000	3.00000	8.10000	65.80000
chlorides	6495.0	0.056042	0.035036	0.00900	0.03800	0.04700	0.06500	0.61100
free sulfur dioxide	6497.0	30.525319	17.749400	1.00000	17.00000	29.00000	41.00000	289.00000
total sulfur dioxide	6497.0	115.744574	56.521855	6.00000	77.00000	118.00000	156.00000	440.00000
density	6497.0	0.994697	0.002999	0.98711	0.99234	0.99489	0.99699	1.03898
pH	6488.0	3.218395	0.160748	2.72000	3.11000	3.21000	3.32000	4.01000
sulphates	6493.0	0.531215	0.148814	0.22000	0.43000	0.51000	0.60000	2.00000
alcohol	6497.0	10.491801	1.192712	8.00000	9.50000	10.30000	11.30000	14.90000
quality	6497.0	5.818378	0.873255	3.00000	5.00000	6.00000	6.00000	9.00000

CHAPTER 3
REQUIRED SYSTEM

For implementing a wine quality prediction system using machine learning, you'll need several components. Here's an outline of the required system:

- **Data Collection and Storage:** You need a mechanism to collect and store wine-related data. This could involve sourcing datasets from repositories like UCI Machine Learning Repository or collecting data from wineries directly. Data storage could be in a relational database like MySQL or in a NoSQL database like MongoDB, depending on the volume and structure of your data

- **Data Preprocessing Pipeline:** Develop a pipeline to preprocess the raw data before feeding it into the machine learning models. This pipeline may include steps like data cleaning (handling missing values, outliers), feature engineering (creating new features or transforming existing ones), and data normalization or standardization.

- **Machine Learning Model Training:** Choose appropriate machine learning algorithms based on the nature of the problem (regression or classification) and the characteristics of the data. Train these models using the preprocessed data. Popular algorithms for regression tasks include linear regression, random forest regression, and gradient boosting regression. For classification tasks, algorithms like logistic regression, random forest classification, and support vector machines can be used.

- **Model Evaluation**: Evaluate the trained models using appropriate evaluation metrics to assess their performance. This step helps you understand how well the models are generalizing to new, unseen data. Cross-validation techniques like k-fold cross-validation can be used to ensure robust evaluation.

- **Hyperparameter Tuning:** Fine-tune the hyperparameters of the machine learning models to improve their performance further. This can be done using techniques like grid search, random search, or Bayesian optimization.

- **Model Deployment:** Once you have a trained and optimized model, deploy it into production. This involves integrating the model into an application or system where it can make predictions on new data. You may need to build an API or a web interface to serve predictions.

- **Monitoring and Maintenance:** Continuously monitor the deployed model's performance and retrain it periodically with new data to keep it up-to-date. This step ensures that the model maintains its accuracy over time and adapts to changes in the data distribution.

- **Documentation and Support:** Document the system thoroughly, including how to use it, how it was built, and any relevant information about the machine learning models used. Provide support channels for users who may have questions or encounter issues with the system.

3.1 SOFTWARE REQUIRMENTS

- **Python Libraries:**

  Pandas: Used for data manipulation and analysis.

  NumPy: Used for numerical operations and computations.

  Scikit-learn: Provides tools for machine learning tasks such as model training, evaluation, and preprocessing.
  Matplotlib and Seaborn: Libraries for data visualization.

- **Jupyter Notebook** or IDEs for development.
- **Visual Studio Code:** For editing Python scripts and other files.
- **Operating System :** While Python is cross-platform, ensure compatibility with your chosen operating system (Windows, macOS, Linux).
- Kaggle for required Data sets

FRONT END:

- **Graphical User Interface (GUI) Toolkit:** Choose a Python GUI toolkit such as Tkinter , Py Qt, or w x Python for building the frontend interface.
- **Input Fields**: Develop input fields for users to input wine attributes.
- **Output Display**: Display predicted wine quality to users within the GUI.

BACK END:

- **Python Web Framework:** Choose a Python web framework such as Flask or Django to handle HTTP requests and responses.
- **Machine Learning Model Integration**: Integrate the trained machine learning model into the backend server to make predictions.
- **Data Preprocessing**: Implement data preprocessing logic if necessary before passing the input to the machine learning model.
- **Documentation**: Provide documentation for users on how to interact with the application and interpret the results.

3.2 HARDWARE REQUIRMENTS

- **Processor (CPU):**
  A modern multi-core processor (e.g., Intel Core i5 or i7, AMD Ryzen 5 or 7) should be sufficient for development and training small to medium-sized machine learning models.

- **Memory (RAM):**
  At least 8 GB of RAM is recommended for handling data manipulation and running machine learning algorithms.

- **Storage:**
  A solid-state drive (SSD) is preferable for faster data access, but a standard hard disk drive (HDD) with sufficient capacity (at least 256 GB) should also work.

- **Internet Connection:**
  A stable internet connection is required for data retrieval (if using online datasets), software installations, and accessing documentation.

CHAPTER 4

SYSTEM DESIGN

4.1 DATAFLOW DIAGRAMS

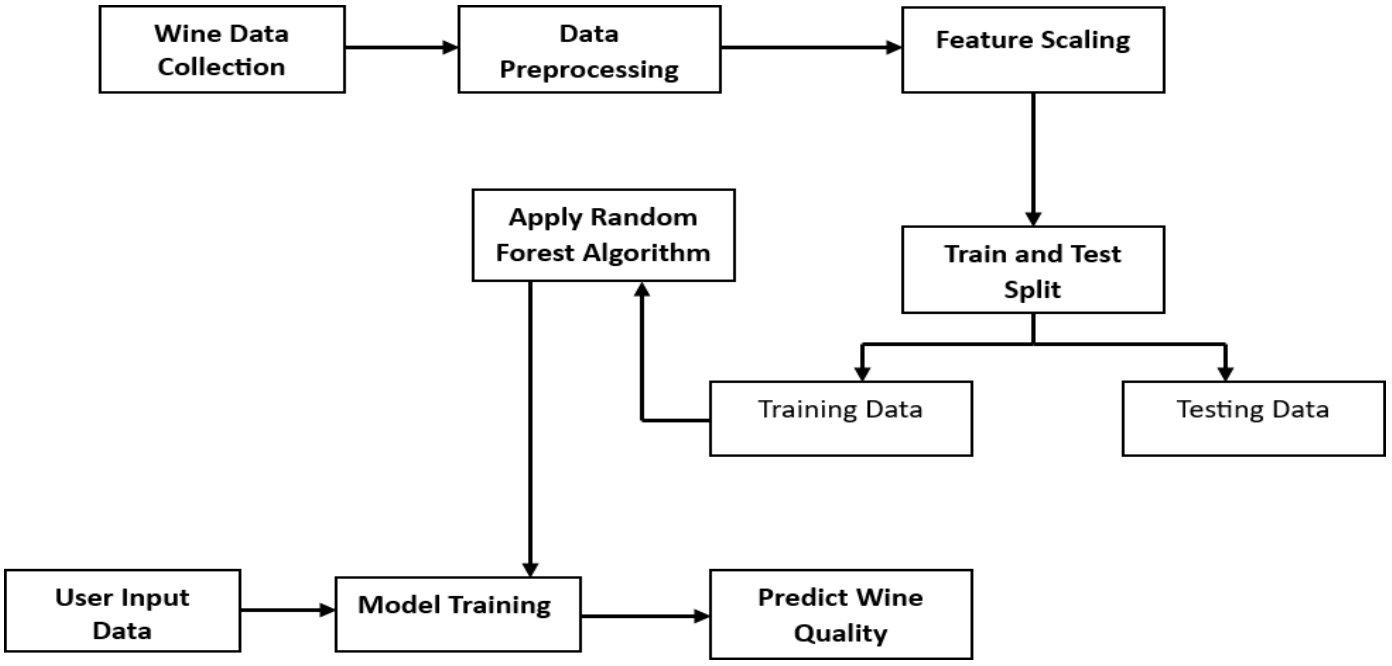


Fig 1. System Architecture

- The system architecture for wine quality prediction using machine learning typically involves several components working together. At the core is the machine learning model, which receives input data on wine attributes and quality ratings. This model is trained using historical data and deployed within a larger system architecture that includes modules for data preprocessing, feature engineering, and model evaluation.
- Data preprocessing modules handle tasks such as data cleaning, normalization, and encoding of categorical variables. Feature engineering modules select relevant features and possibly create new ones to enhance prediction accuracy. Model evaluation modules assess the performance of trained models using validation techniques.
- Deployment modules integrate the trained model into production systems, enabling real-time predictions. Additionally, monitoring and maintenance modules continuously monitor the deployed model's performance, ensuring it remains accurate over time and can be updated or retrained as needed.
- This system architecture facilitates the end-to-end process of wine quality prediction, from data ingestion to model deployment and maintenance, providing a scalable and robust framework for predictive analytics in the wine industry.

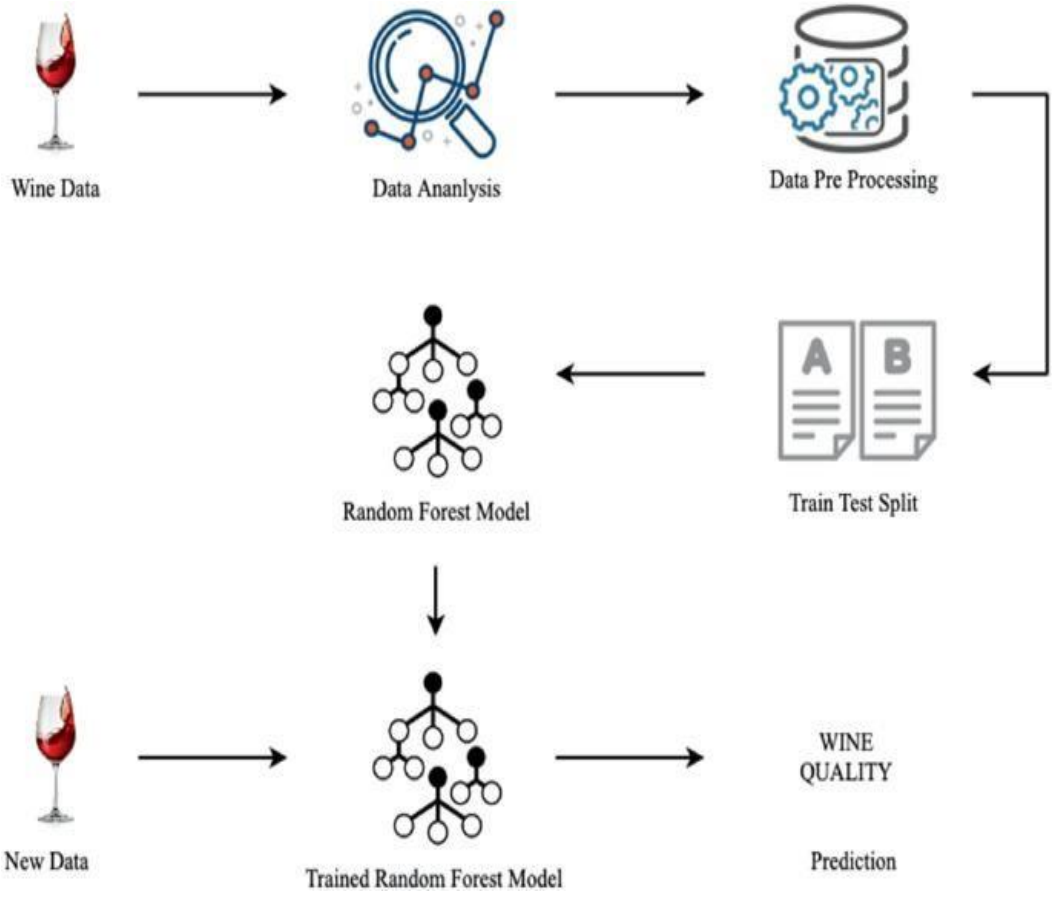


Fig 2. Flowchart

- The wine quality prediction process using machine learning begins with data collection, where relevant information about wine attributes and quality ratings is gathered. Subsequently, the collected data undergoes preprocessing to clean any inconsistencies, handle missing values, and normalize the features. Feature selection and engineering follow, aiming to identify the most significant predictors and possibly create new features that enhance prediction accuracy.
- The dataset is then split into training, validation, and test sets to facilitate model development and evaluation. Next, a suitable machine learning algorithm is selected, and the model is trained on the training data while tuning hyperparameters to optimize performance. Evaluation is conducted using the validation set to ensure the model generalizes well to unseen data.
- After satisfactory performance is confirmed, the model undergoes testing on the independent test set to provide an unbiased estimate of its predictive capabilities. Once validated, the model is deployed into production systems, where it can generate predictions for new instances of wine quality. Continuous monitoring and maintenance ensure the model's performance remains optimal over time.
- Additionally, a feedback loop is established to gather insights from model usage and improve its accuracy and relevance based on new data or user feedback.

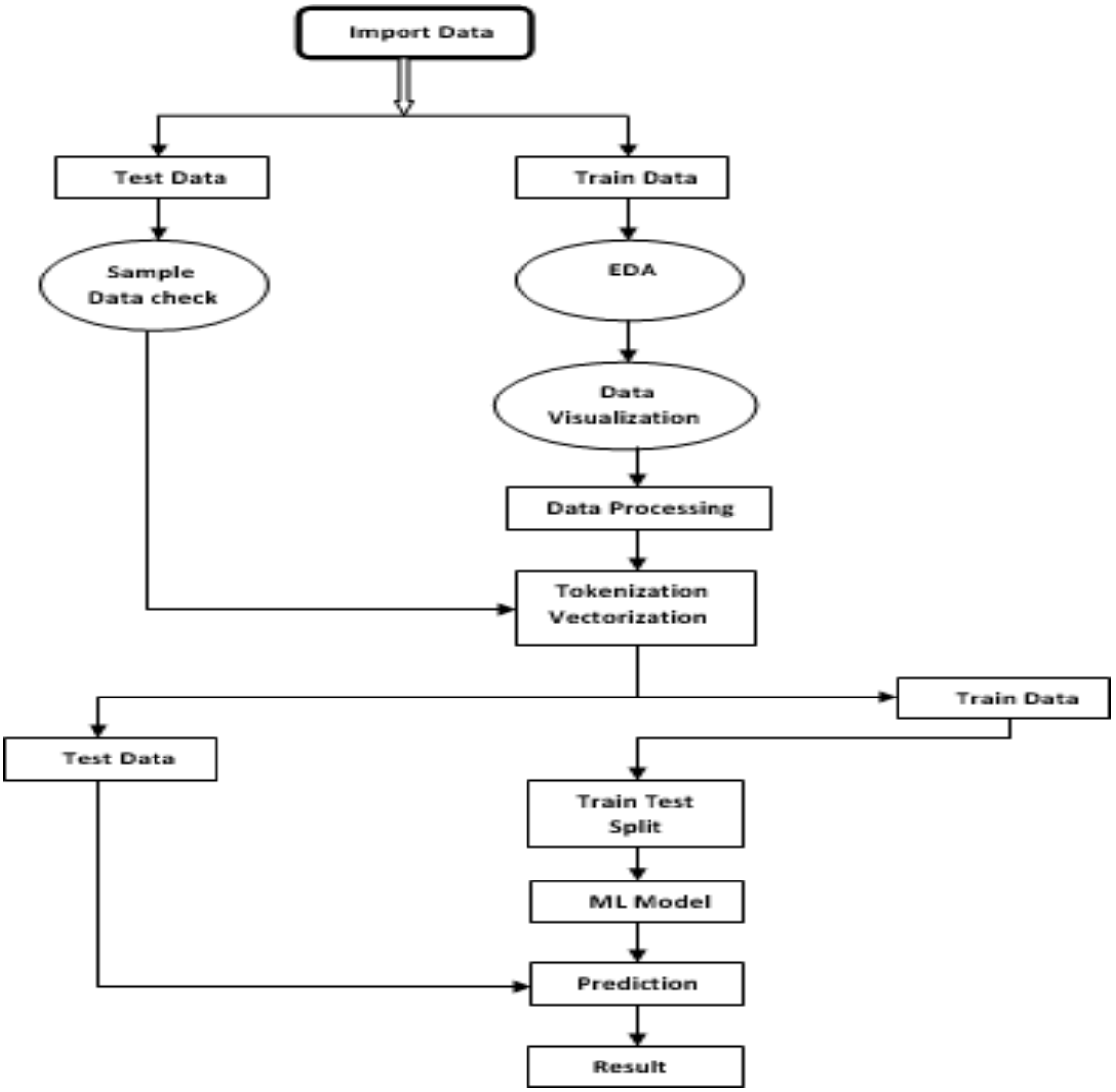


Fig 3. Work Flow

- The workflow for wine quality prediction using machine learning involves several key stages. Initially, relevant data on wine attributes and quality ratings is collected and preprocessed to ensure consistency and reliability.
- Following this, feature selection and engineering techniques are applied to identify the most influential features and possibly create new ones. The dataset is then split into training and testing sets for model development and evaluation. A suitable machine learning algorithm is selected and trained on the training data, with hyperparameters fine-tuned to optimize performance.
- The trained model is then evaluated on the test set to assess its predictive accuracy and generalization capabilities. Once the model demonstrates satisfactory performance, it is deployed into production systems, where it can make predictions on new instances of wine quality.
- Continuous monitoring and maintenance of the deployed model ensure its effectiveness over time, with periodic updates and retraining to adapt to evolving data and user needs.

4.2 UML DIGRAM

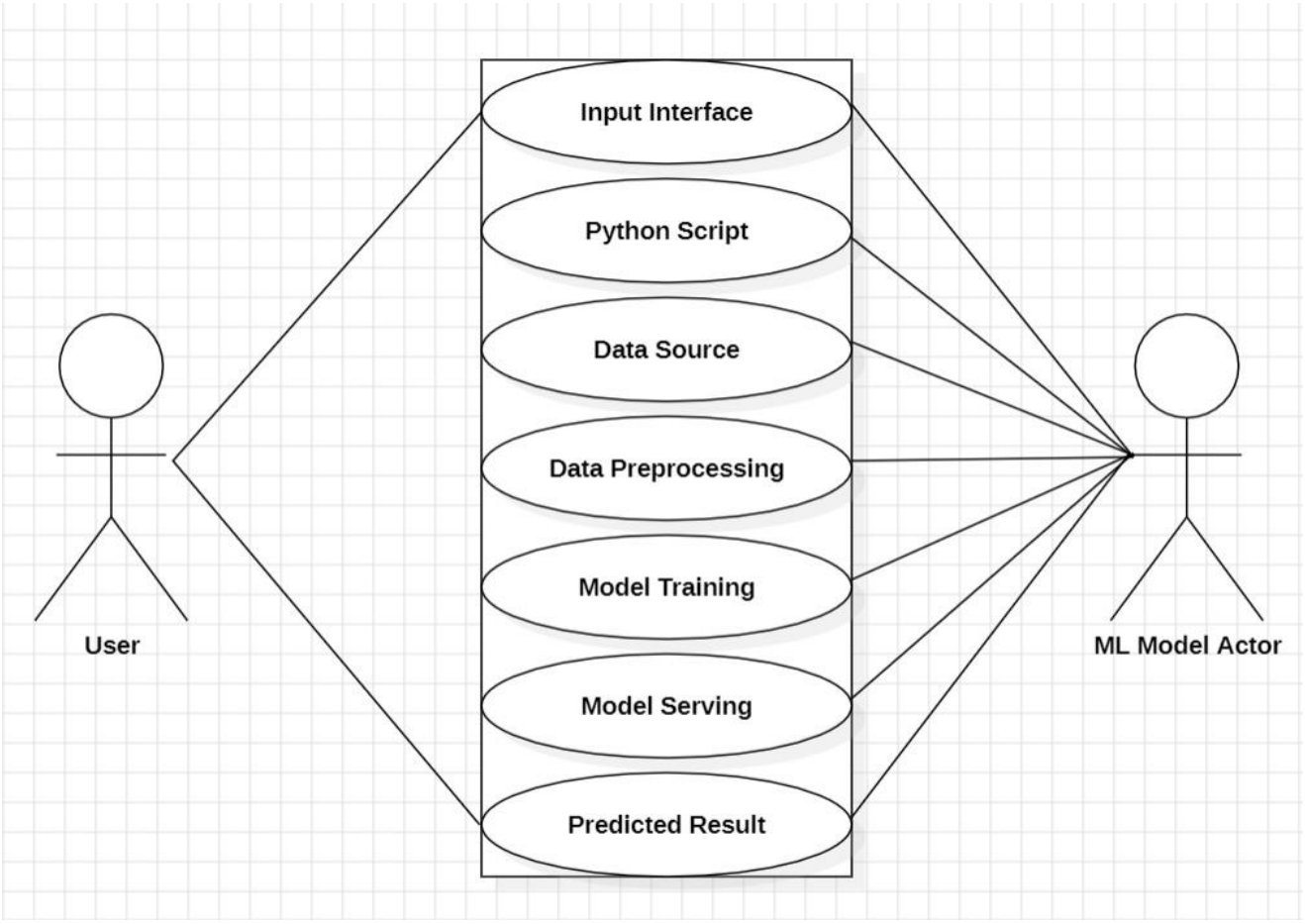
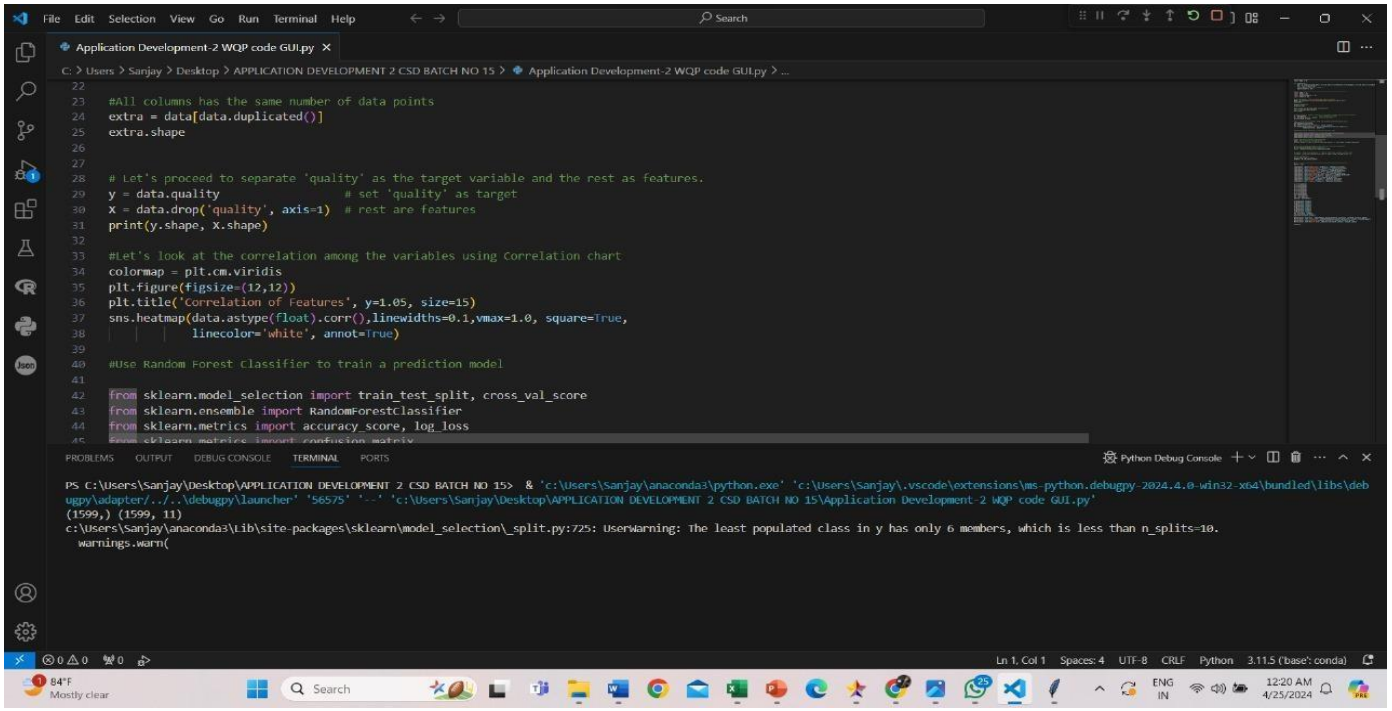


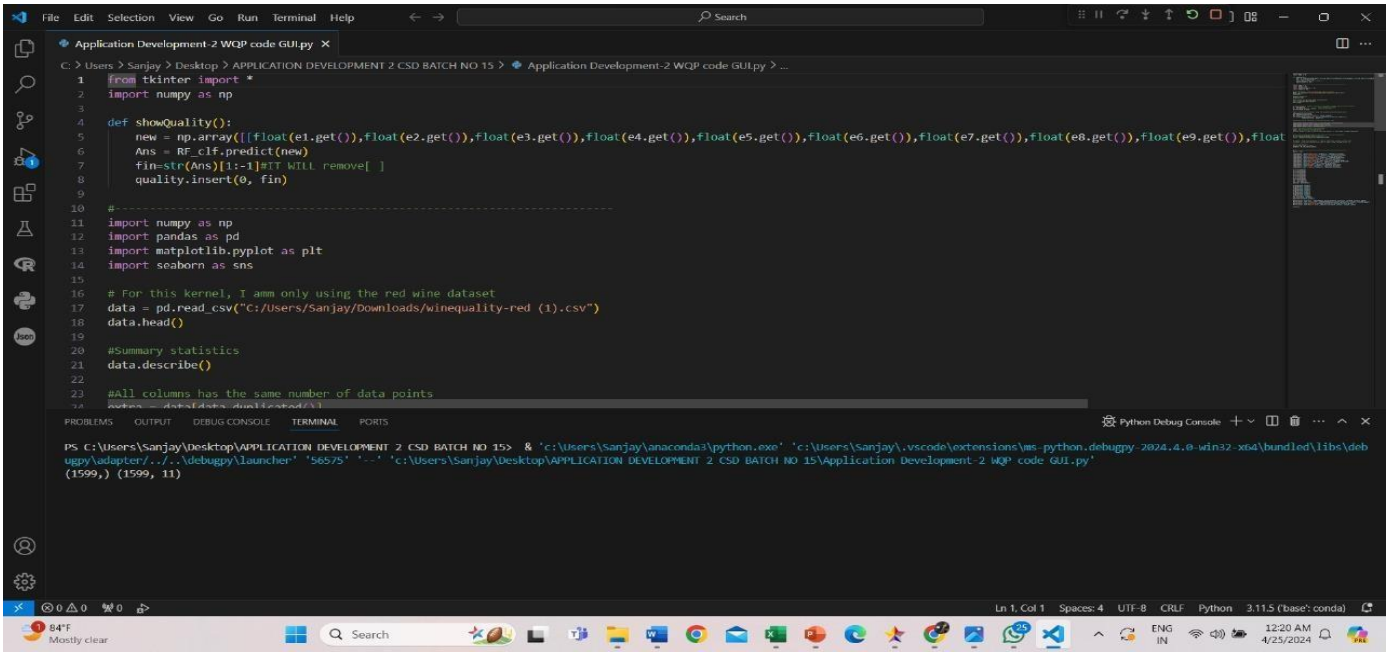
Fig 4.Use Case Diagram

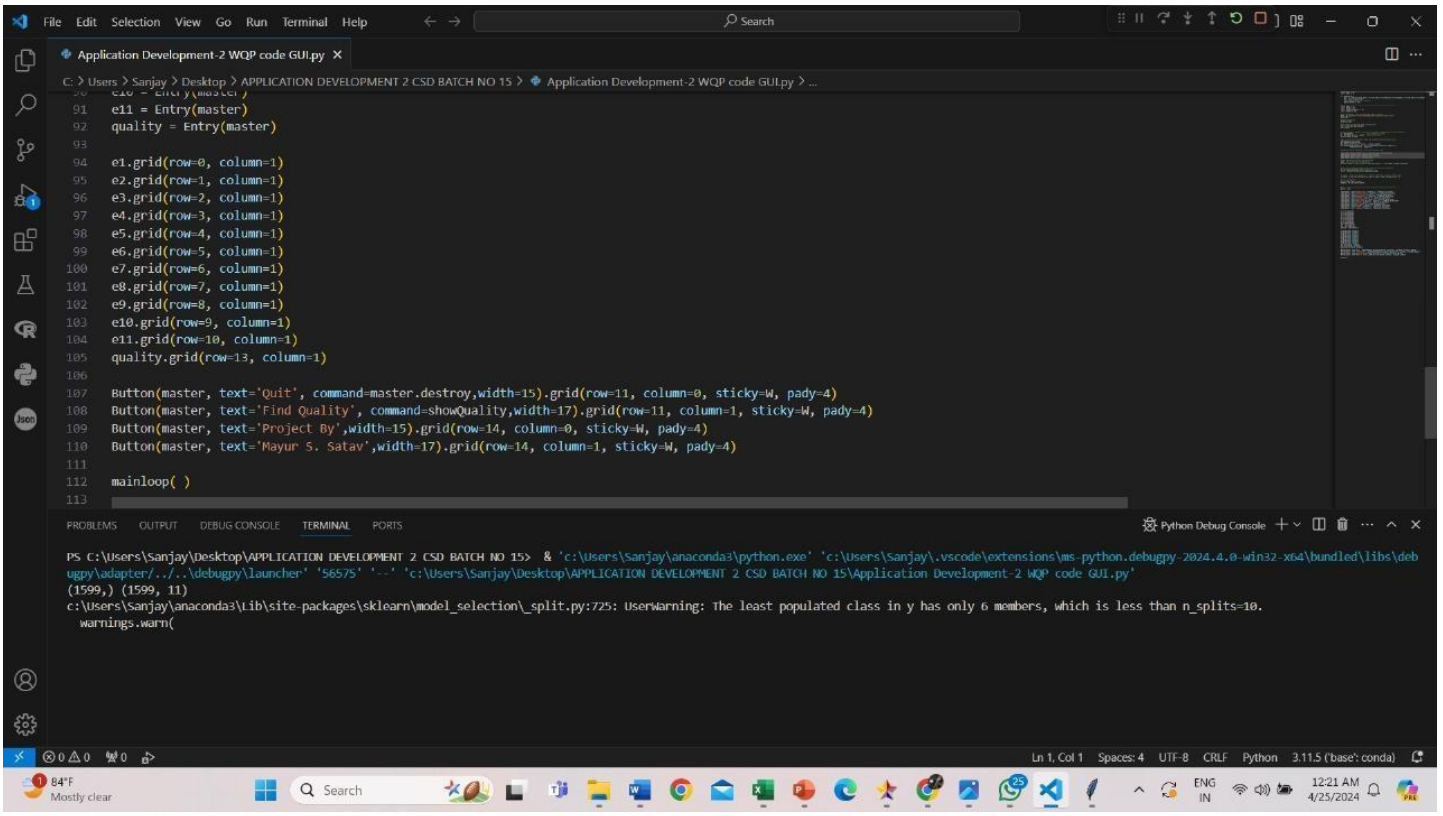
- In a use case diagram for wine quality prediction using machine learning, the primary actors are typically the Data Scientist, who interacts with the machine learning pipeline, and the System, which encompasses the various components involved in data processing, model training, and prediction.
- Use cases include "Collect Data," where the System gathers information on wine attributes and quality ratings; "Preprocess Data," involving cleaning, handling missing values, and normalization; "Train Model," where the Data Scientist trains machine learning algorithms on the processed data; "Evaluate Model," to assess the performance of the trained models; "Deploy Model," to integrate the model into production systems for real-time prediction; and "Monitor Model," for continuous performance monitoring and maintenance.
- Additionally, there may be use cases for "Feature Selection/Engineering" and "Hyperparameter Tuning" within the "Train Model" phase, reflecting the iterative nature of model development

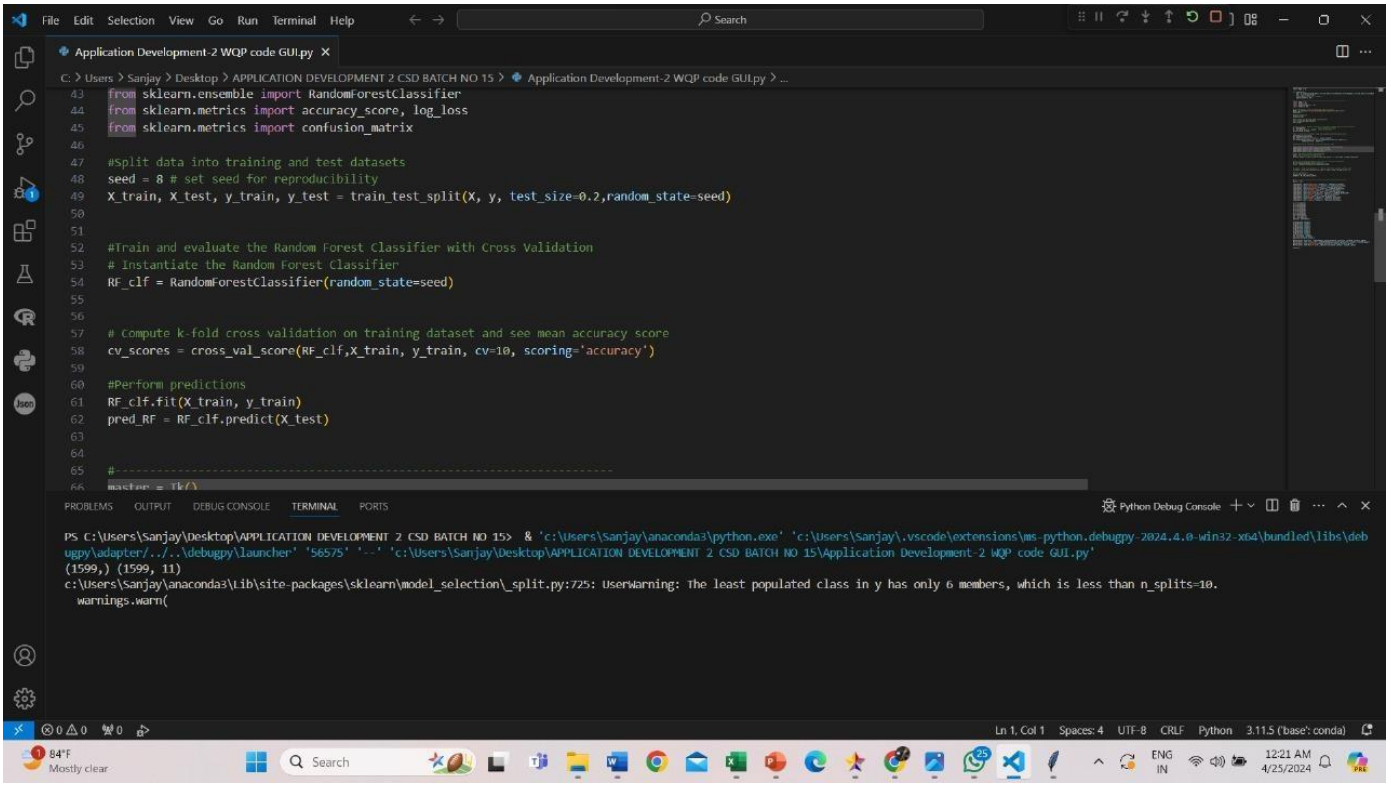
CHAPTER 5

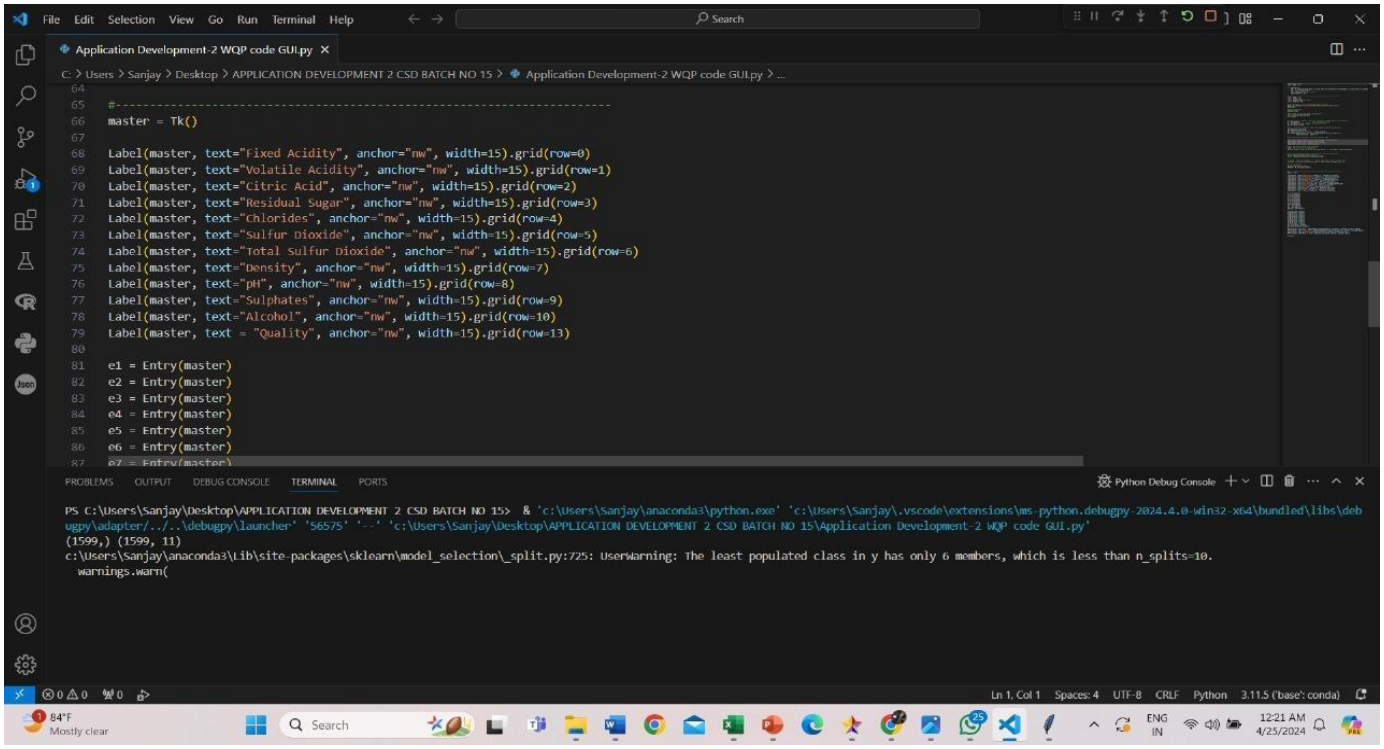
IMPLEMENTATION

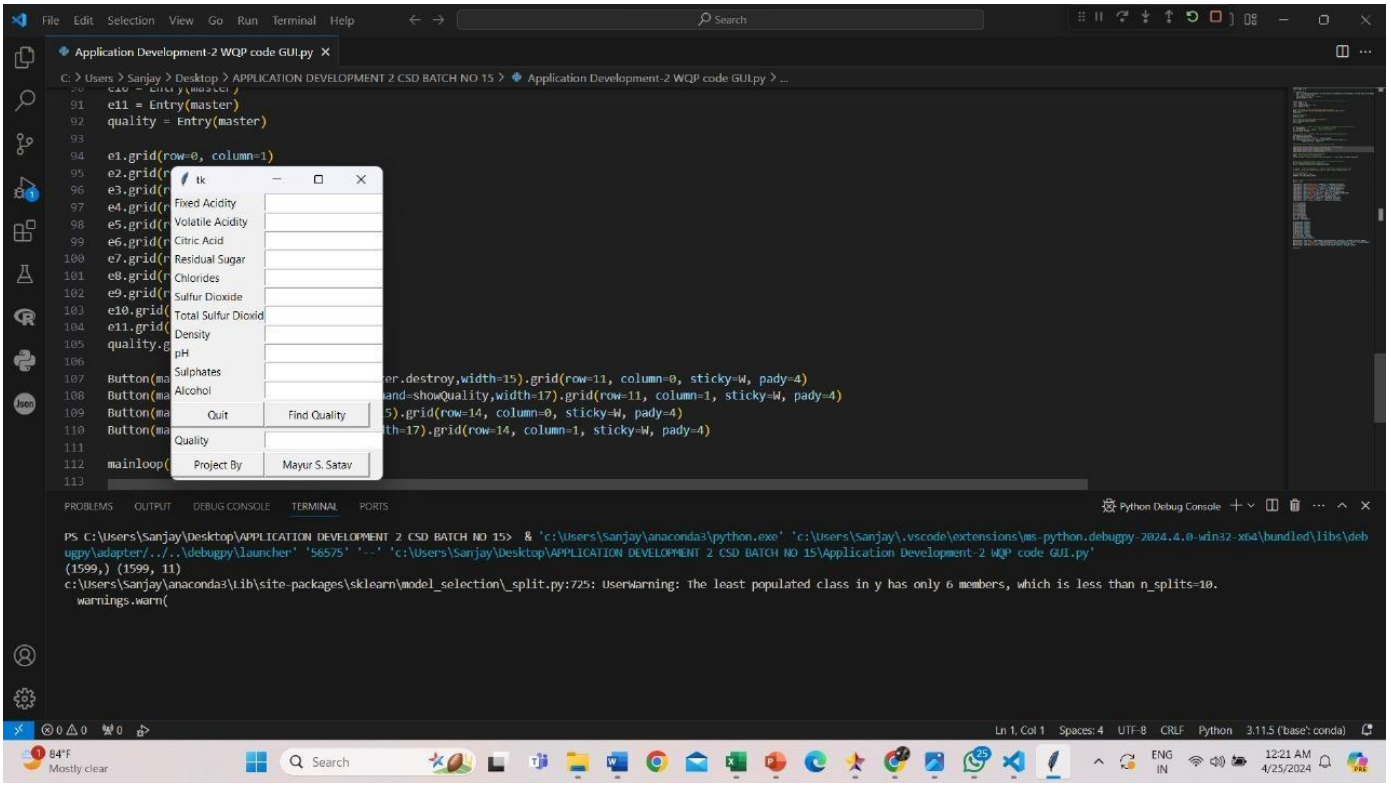




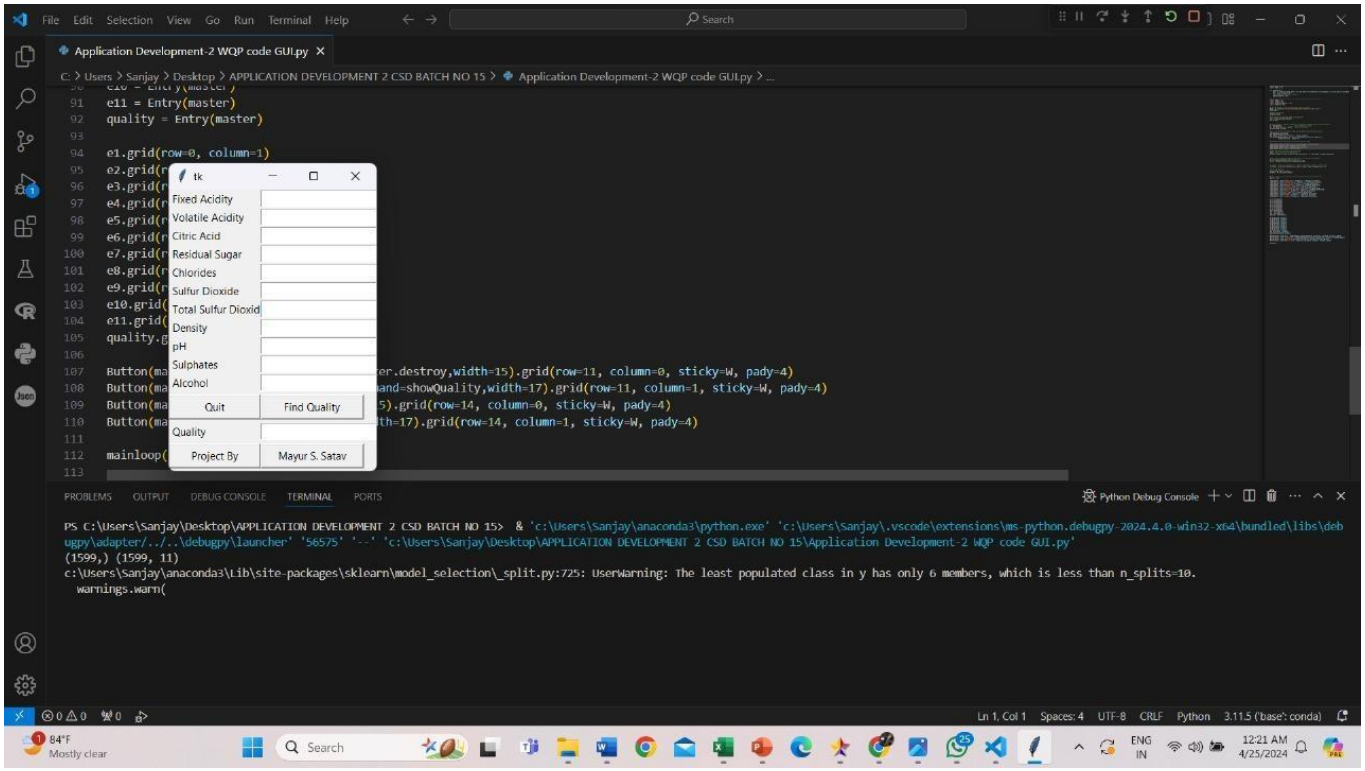

File Edit Selection View Go Run Terminal Help
Application Development-2 WQP code GUI.py
C: > Users > Sanjay > Desktop > APPLICATION DEVELOPMENT 2 CSD BATCH NO 15 > Application Development-2 WQP code GUI.py > ...
e11 = Entry(master)
quality = Entry(master)
e1.grid(row=0, column=1)
e2.grid(row=1, column=1)
e3.grid(row=2, column=1)
e4.grid(row=3, column=1)
e5.grid(row=4, column=1)
e6.grid(row=5, column=1)
e7.grid(row=6, column=1)
e8.grid(row=7, column=1)
e9.grid(row=8, column=1)
e10.grid(row=9, column=1)
e11.grid(row=10, column=1)
quality.grid(row=13, column=1)
Button(master, text='Quit', command=master.destroy,width=15).grid(row=11, column=0, sticky=W, pady=4)
Button(master, text='Find Quality', command=showQuality,width=17).grid(row=11, column=1, sticky=W, pady=4)
Button(master, text='Project By',width=15).grid(row=14, column=0, sticky=W, pady=4)
Button(master, text='Mayur S. Satav',width=17).grid(row=14, column=1, sticky=W, pady=4)
mainloop( )
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS
Python Debug Console
PS C:\Users\Sanjay\Desktop\APPLICATION DEVELOPMENT 2 CSD BATCH NO 15> & 'c:\Users\Sanjay\anaconda3\python.exe' 'c:\Users\Sanjay\.vscode\extensions\ms-python.debugpy-2024.4.0-win32-x64\bundled\libs\debugpy\adapter/../..\debugpy\launcher' '56575' '--' 'c:\Users\Sanjay\Desktop\APPLICATION DEVELOPMENT 2 CSD BATCH NO 15\Application Development-2 WQP code GUI.py'
(1599,) (1599, 11)
c:\Users\Sanjay\anaconda3\Lib\site-packages\sklearn\model_selection_split.py:725: UserWarning: The least populated class in y has only 6 members, which is less than n_splits=10.
warnings.warn(
Ln 1, Col 1 Spaces: 4 UTF-8 CRLF Python 3.11.5 ('base': conda)
84°F Mostly clear
Search
ENG IN
12:21 AM 4/25/2024


File Edit Selection View Go Run Terminal Help
Application Development-2 WQP code GUI.py
C: > Users > Sanjay > Desktop > APPLICATION DEVELOPMENT 2 CSD BATCH NO 15 > Application Development-2 WQP code GUI.py > ...
from sklearn.ensemble import RandomForestClassifier
from sklearn.metrics import accuracy_score, log_loss
from sklearn.metrics import confusion_matrix
#Split data into training and test datasets
seed = 8 # set seed for reproducibility
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2,random_state=seed)
#Train and evaluate the Random Forest Classifier with Cross Validation
# Instantiate the Random Forest Classifier
RF_clf = RandomForestClassifier(random_state=seed)
# Compute k-fold cross validation on training dataset and see mean accuracy score
cv_scores = cross_val_score(RF_clf,X_train, y_train, cv=10, scoring='accuracy')
#Perform predictions
RF_clf.fit(X_train, y_train)
pred_RF = RF_clf.predict(X_test)
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS
Python Debug Console
PS C:\Users\Sanjay\Desktop\APPLICATION DEVELOPMENT 2 CSD BATCH NO 15> & 'c:\Users\Sanjay\anaconda3\python.exe' 'c:\Users\Sanjay\.vscode\extensions\ms-python.debugpy-2024.4.0-win32-x64\bundled\libs\debugpy\adapter/../..\debugpy\launcher' '56575' '--' 'c:\Users\Sanjay\Desktop\APPLICATION DEVELOPMENT 2 CSD BATCH NO 15\Application Development-2 WQP code GUI.py'
(1599,) (1599, 11)
c:\Users\Sanjay\anaconda3\Lib\site-packages\sklearn\model_selection_split.py:725: UserWarning: The least populated class in y has only 6 members, which is less than n_splits=10.
warnings.warn(
Ln 1, Col 1 Spaces: 4 UTF-8 CRLF Python 3.11.5 ('base': conda)
84°F Mostly clear
Search
ENG IN
12:21 AM 4/25/2024

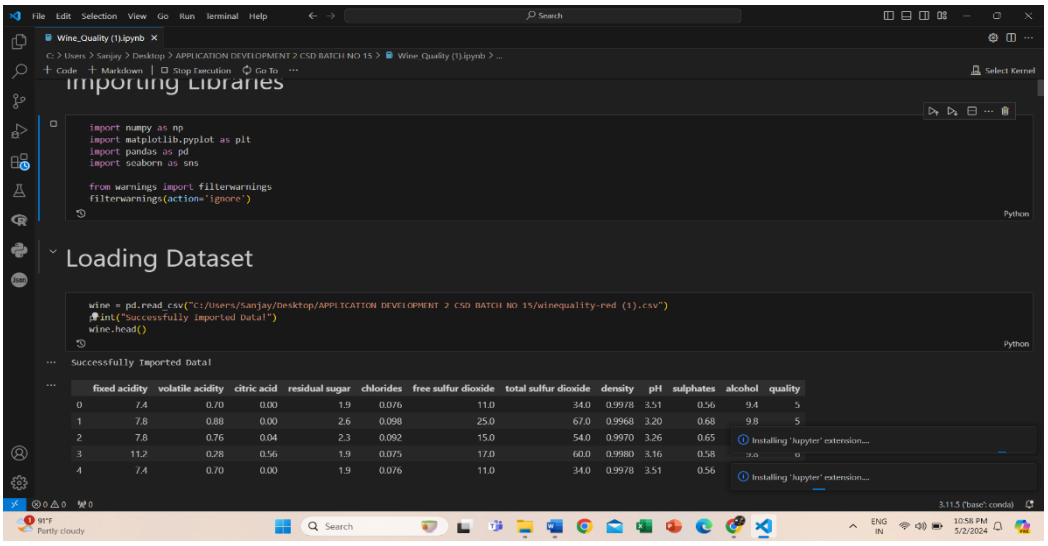

Application Development-2 WQP code GUI.py
C: > Users > Sanjay > Desktop > APPLICATION DEVELOPMENT 2 CSD BATCH NO 15 > Application Development-2 WQP code GUI.py > ...
master = Tk()
Label(master, text="Fixed Acidity", anchor="nw", width=15).grid(row=0)
Label(master, text="Volatile Acidity", anchor="nw", width=15).grid(row=1)
Label(master, text="Citric Acid", anchor="nw", width=15).grid(row=2)
Label(master, text="Residual Sugar", anchor="nw", width=15).grid(row=3)
Label(master, text="Chlorides", anchor="nw", width=15).grid(row=4)
Label(master, text="Sulfur Dioxide", anchor="nw", width=15).grid(row=5)
Label(master, text="Total Sulfur Dioxide", anchor="nw", width=15).grid(row=6)
Label(master, text="Density", anchor="nw", width=15).grid(row=7)
Label(master, text="pH", anchor="nw", width=15).grid(row=8)
Label(master, text="Sulphates", anchor="nw", width=15).grid(row=9)
Label(master, text="Alcohol", anchor="nw", width=15).grid(row=10)
Label(master, text = "Quality", anchor="nw", width=15).grid(row=13)
e1 = Entry(master)
e2 = Entry(master)
e3 = Entry(master)
e4 = Entry(master)
e5 = Entry(master)
e6 = Entry(master)
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS
Python Debug Console
PS C:\Users\Sanjay\Desktop\APPLICATION DEVELOPMENT 2 CSD BATCH NO 15> & 'c:\Users\Sanjay\anaconda3\python.exe' 'c:\Users\Sanjay\.vscode\extensions\ms-python.debugpy-2024.4.0-win32-x64\bundled\libs\debugpy\adapter/../..\debugpy\launcher' '56575' '--' 'c:\Users\Sanjay\Desktop\APPLICATION DEVELOPMENT 2 CSD BATCH NO 15\Application Development-2 WQP code GUI.py'
(1599,) (1599, 11)
c:\Users\Sanjay\anaconda3\Lib\site-packages\sklearn\model_selection_split.py:725: UserWarning: The least populated class in y has only 6 members, which is less than n_splits=10.
warnings.warn(
Ln 1, Col 1 Spaces: 4 UTF-8 CRLF Python 3.11.5 ('base': conda)
84°F Mostly clear
Search
ENG IN
12:21 AM 4/25/2024

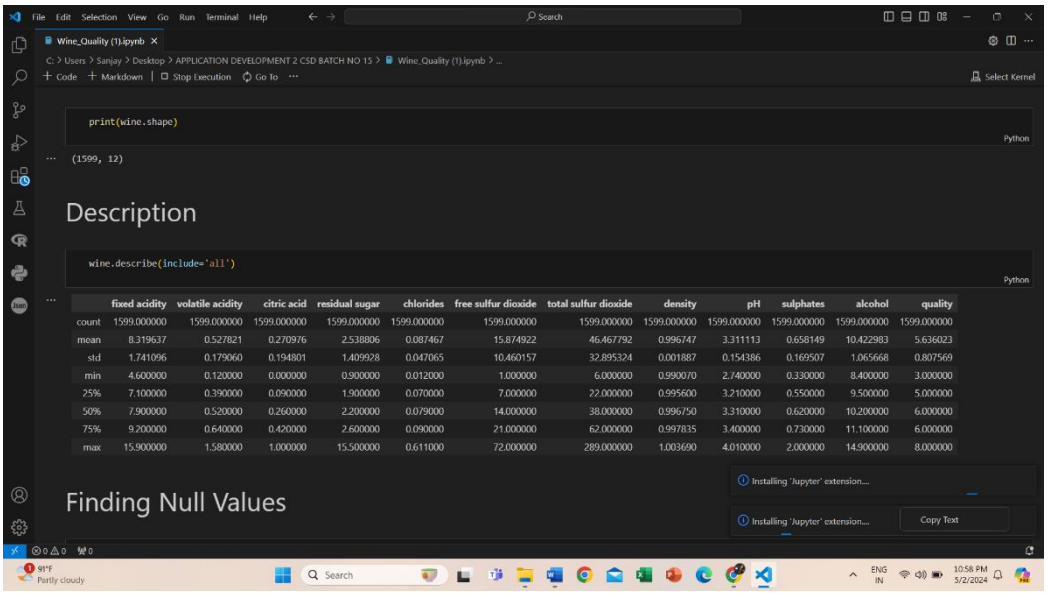

Application Development-2 WQP code GUI.py
C: > Users > Sanjay > Desktop > APPLICATION DEVELOPMENT 2 CSD BATCH NO 15 > Application Development-2 WQP code GUI.py > ...
e11 = Entry(master)
quality = Entry(master)
e1.grid(row=0, column=1)
tk
Fixed Acidity
Volatile Acidity
Citric Acid
Residual Sugar
Chlorides
Sulfur Dioxide
Total Sulfur Dioxid
Density
pH
Sulphates
Alcohol
Quit
Find Quality
Quality
Project By
Mayur S. Satav
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS
Python Debug Console
PS C:\Users\Sanjay\Desktop\APPLICATION DEVELOPMENT 2 CSD BATCH NO 15> & 'c:\Users\Sanjay\anaconda3\python.exe' 'c:\Users\Sanjay\.vscode\extensions\ms-python.debugpy-2024.4.0-win32-x64\bundled\libs\debugpy\adapter/../..\debugpy\launcher' '56575' '--' 'c:\Users\Sanjay\Desktop\APPLICATION DEVELOPMENT 2 CSD BATCH NO 15\Application Development-2 WQP code GUI.py'
(1599,) (1599, 11)
c:\Users\Sanjay\anaconda3\Lib\site-packages\sklearn\model_selection_split.py:725: UserWarning: The least populated class in y has only 6 members, which is less than n_splits=10.
warnings.warn(
Ln 1, Col 1 Spaces: 4 UTF-8 CRLF Python 3.11.5 ('base': conda)
84°F Mostly clear
Search
ENG IN
12:21 AM 4/25/2024

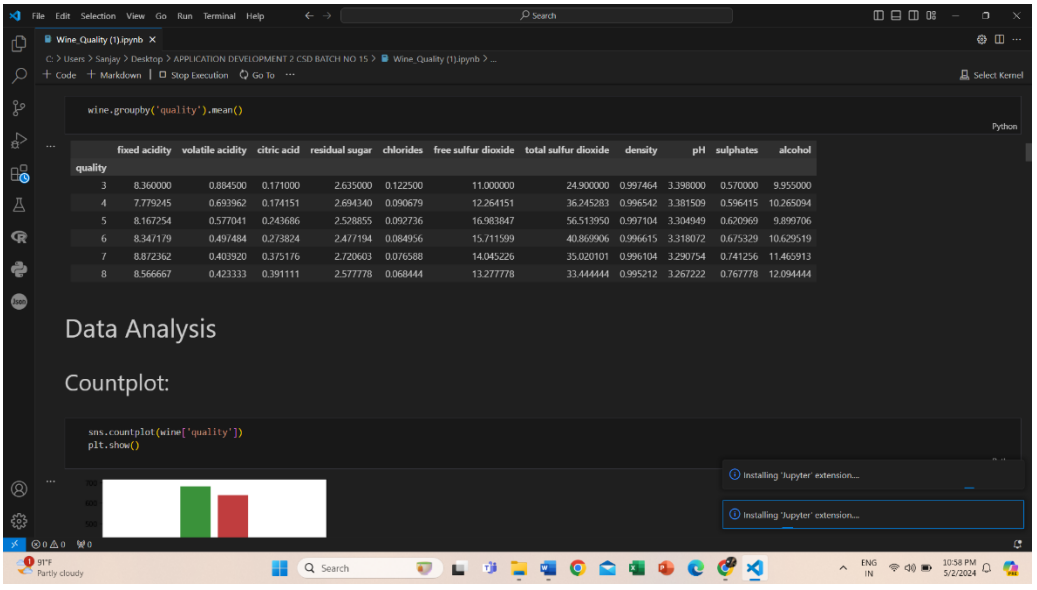
User Interface:

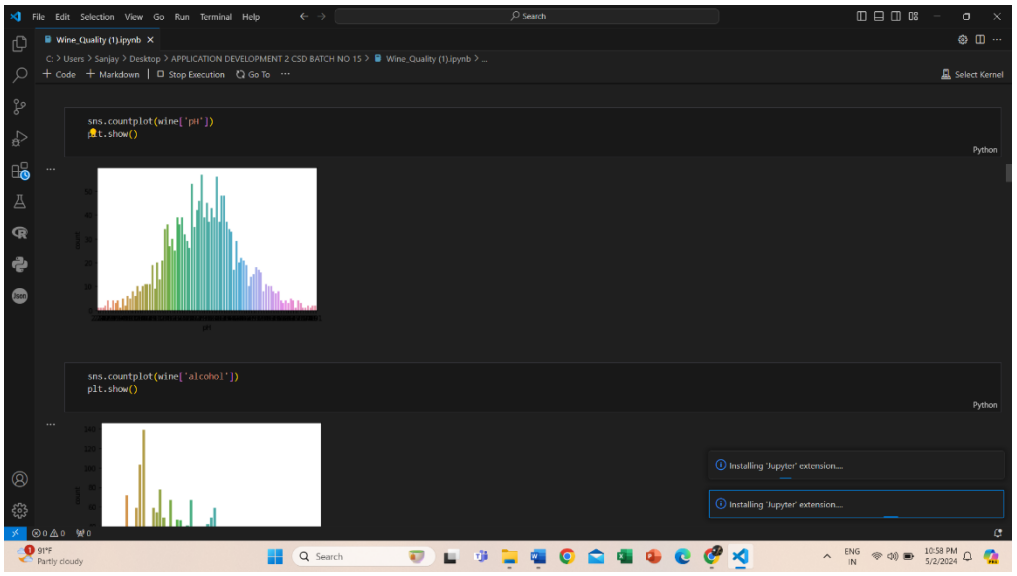


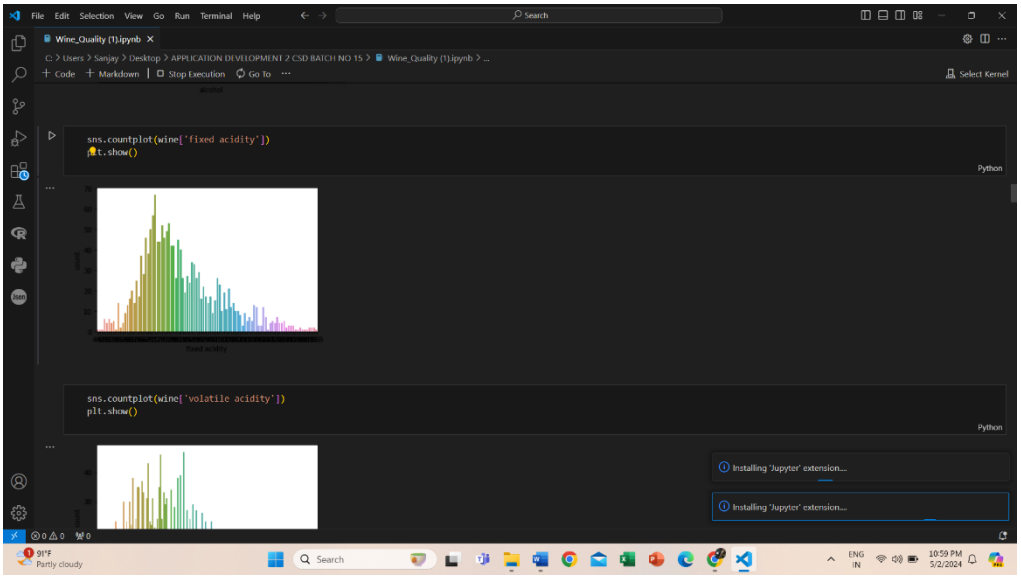
Data Preprocessing Implementation:

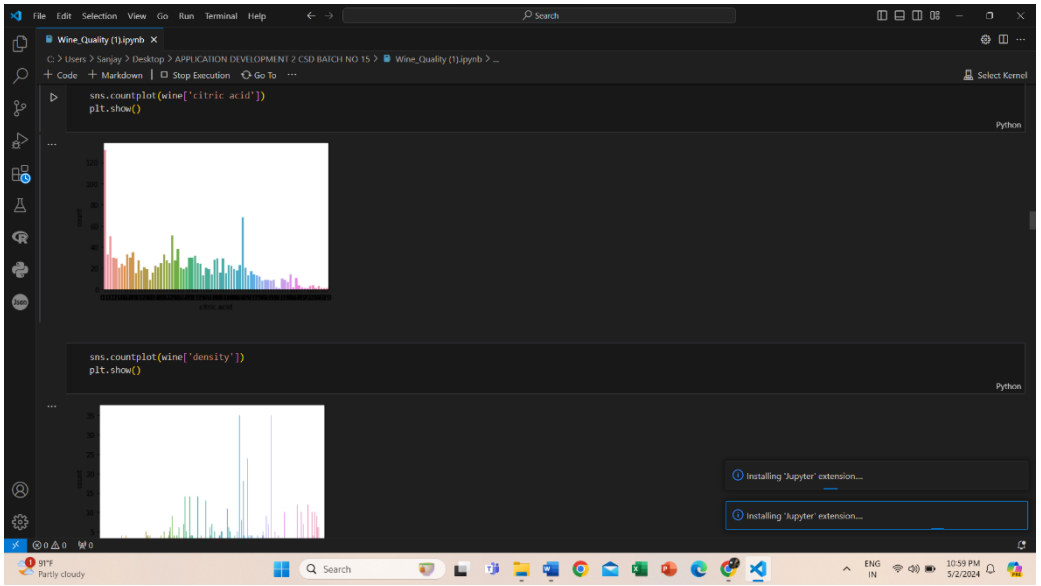







sns.countplot(wine['pH'])
plt.show()
sns.countplot(wine['alcohol'])
plt.show()

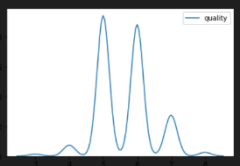

sns.countplot(wine['fixed acidity'])
plt.show()
sns.countplot(wine['volatile acidity'])
plt.show()


sns.countplot(wine['citric acid'])
plt.show()
sns.countplot(wine['density'])
plt.show()

KDE plot:

```
sns.kdeplot(wine.query('quality > 2').quality)
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x24217cefc48>
```



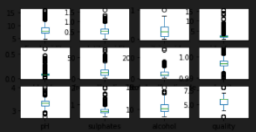
Distplot:

```
sns.distplot(wine['alcohol'])
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x24217e2c148>
```

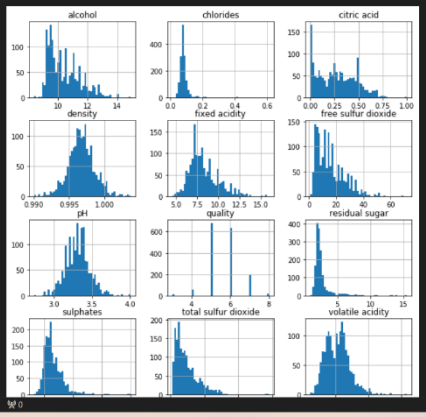
```
wine.plot(kind ='box',subplots = True, layout =(4,4),sharex = False)
```

```
fixed acidity           AxesSubplot(0.125,0.71587;0.168478x0.16413)
volatile acidity        AxesSubplot(0.327174,0.71587;0.168478x0.16413)
citric acid             AxesSubplot(0.529348,0.71587;0.168478x0.16413)
residual sugar          AxesSubplot(0.731522,0.71587;0.168478x0.16413)
chlorides               AxesSubplot(0.125,0.518913;0.168478x0.16413)
free sulfur dioxide     AxesSubplot(0.327174,0.518913;0.168478x0.16413)
total sulfur dioxide    AxesSubplot(0.529348,0.518913;0.168478x0.16413)
density                 AxesSubplot(0.731522,0.518913;0.168478x0.16413)
pH                      AxesSubplot(0.125,0.321957;0.168478x0.16413)
sulphates               AxesSubplot(0.327174,0.321957;0.168478x0.16413)
alcohol                 AxesSubplot(0.529348,0.321957;0.168478x0.16413)
quality                 AxesSubplot(0.731522,0.321957;0.168478x0.16413)
dtype: object
```



```
wine.plot(kind ='density',subplots = True, layout =(4,4),sharex = False)
```

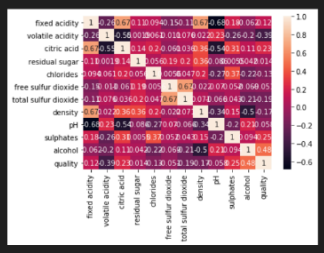
```
wine.hist(figsize=(10,10),bins=50)
plt.show()
```



Heatmap for expressing correlation

```
corr = wine.corr()
sns.heatmap(corr,annot=True)
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x1df1779a288>
```

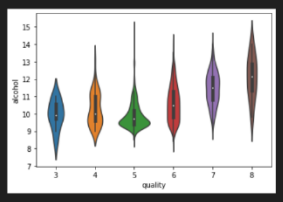


Pair Plot:

Violinplot:

```
sns.violinplot(x='quality', y='alcohol', data=wine)
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x242182ba6c8>
```



Feature Selection

```
# Create Classification version of target variable
wine['goodquality'] = [1 if x >= 7 else 0 for x in wine['quality']]# Separate feature variables and target variable
X = wine.drop(['quality','goodquality'], axis = 1)
Y = wine['goodquality']
```

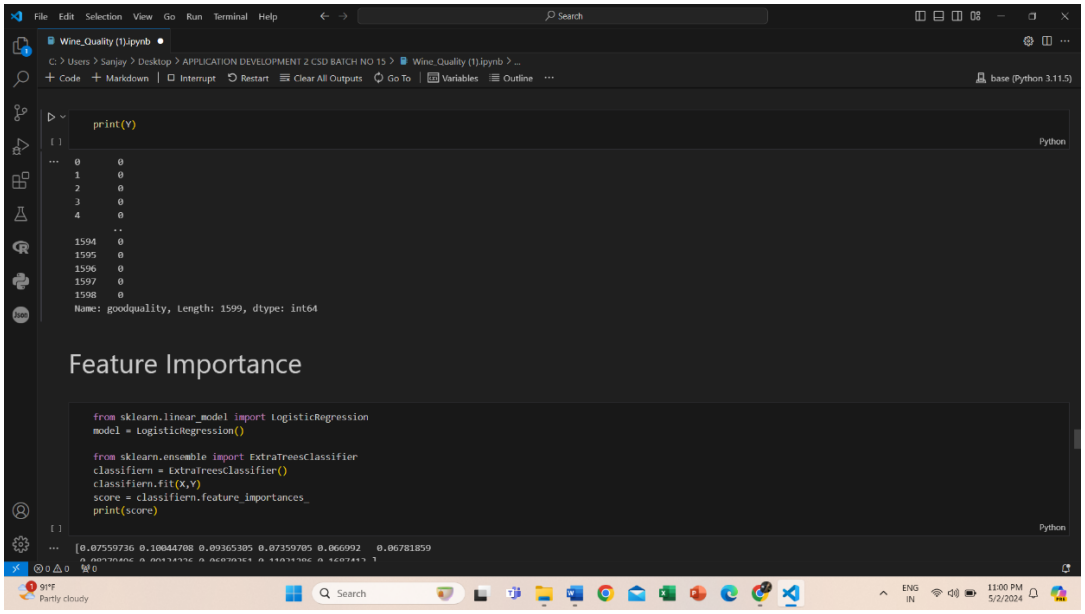
```
# See proportion of good vs bad wines
wine['goodquality'].value_counts()
```

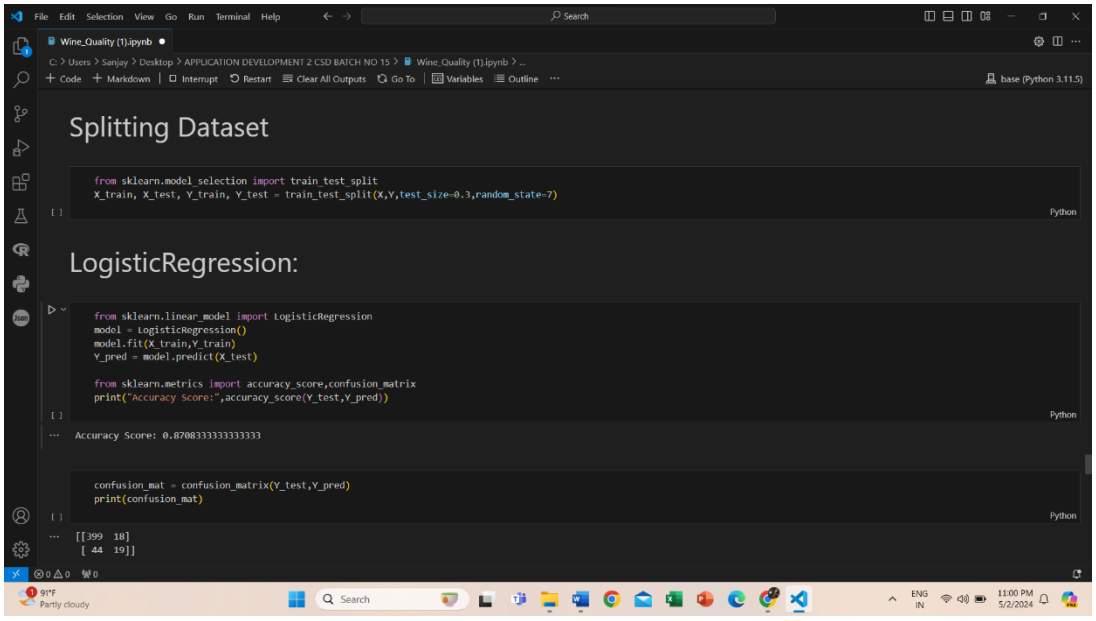
```
0    1382
1     217
Name: goodquality, dtype: int64
```

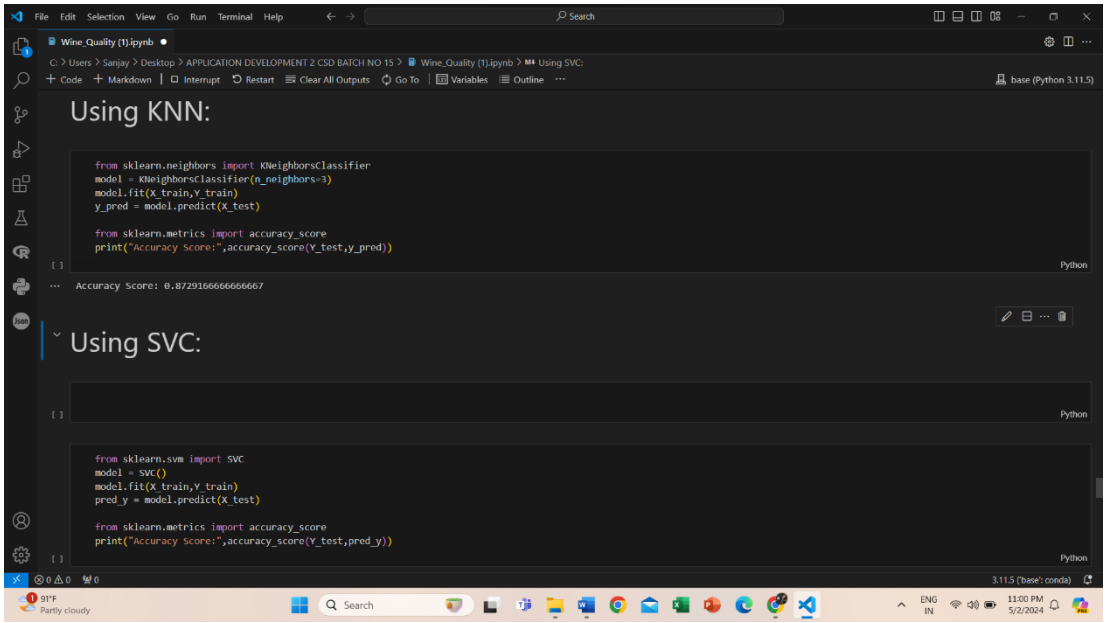
```
x
```

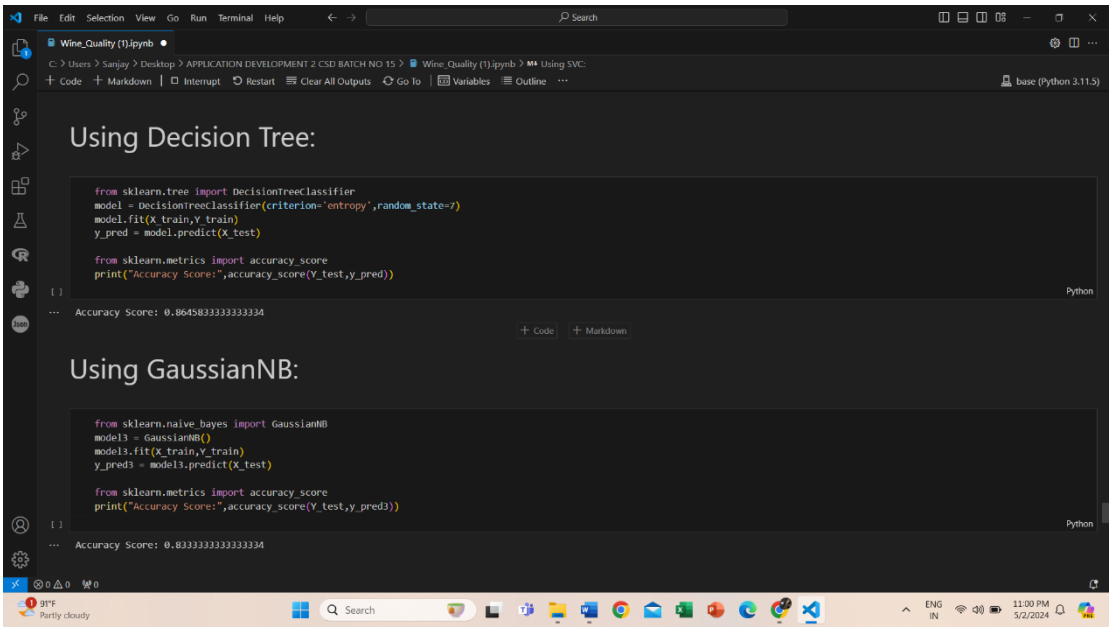
	fixed acidity	volatile acidity	citric acid	residual sugar	chlorides	free sulfur dioxide	total sulfur dioxide	density	pH	sulphates	alcohol
0	7.4	0.700	0.00	1.9	0.076	11.0	34.0	0.99780	3.51	0.56	9.4
1	7.8	0.880	0.00	2.6	0.098	25.0	67.0	0.99680	3.20	0.68	9.8
2	7.8	0.760	0.04	2.3	0.092	15.0	54.0	0.99700	3.26	0.65	9.8
3	11.2	0.280	0.56	1.9	0.075	17.0	60.0	0.99800	3.16	0.58	9.8
4	7.4	0.700	0.00	1.9	0.076	11.0	34.0	0.99780	3.51	0.56	9.4
...	...	...	...	...	...	...	...	...	...	...	...
1594	6.2	0.600	0.08	2.0	0.090	32.0	44.0	0.99490	3.45	0.58	10.5
1595	5.9	0.550	0.10	2.2	0.062	39.0	51.0	0.99512	3.52	0.76	11.2
1596	6.3	0.510	0.13	2.3	0.076	29.0	40.0	0.99574	3.42	0.75	11.0
1597	5.9	0.645	0.12	2.0	0.075	32.0	44.0	0.99547	3.57	0.71	10.2
1598	6.0	0.310	0.47	3.6	0.067	18.0	42.0	0.99549	3.39	0.66	11.0

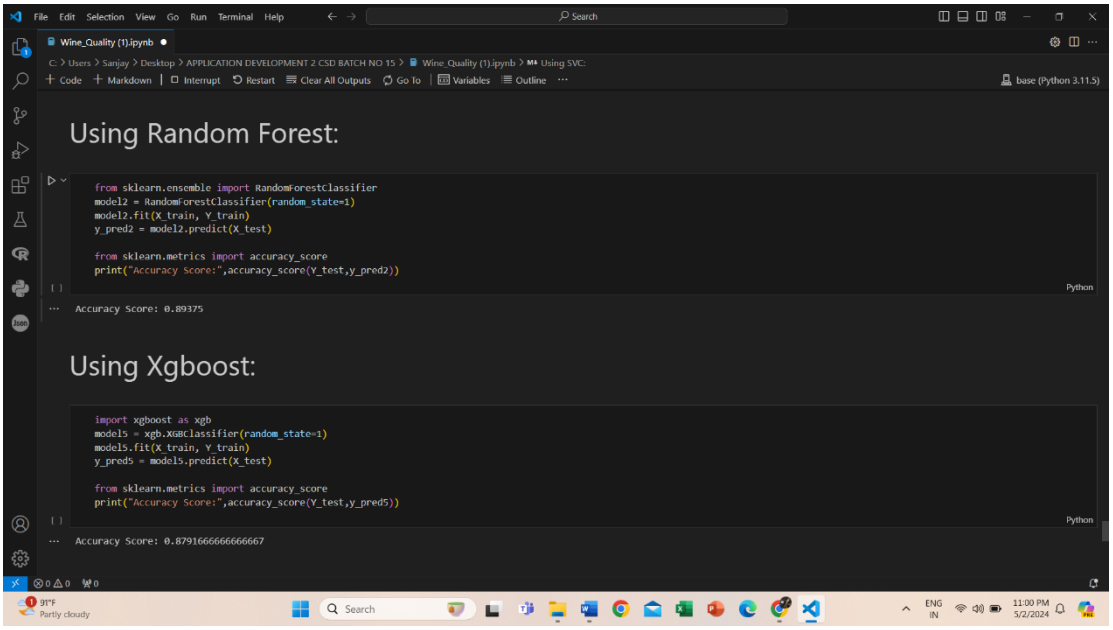
1599 rows × 11 columns

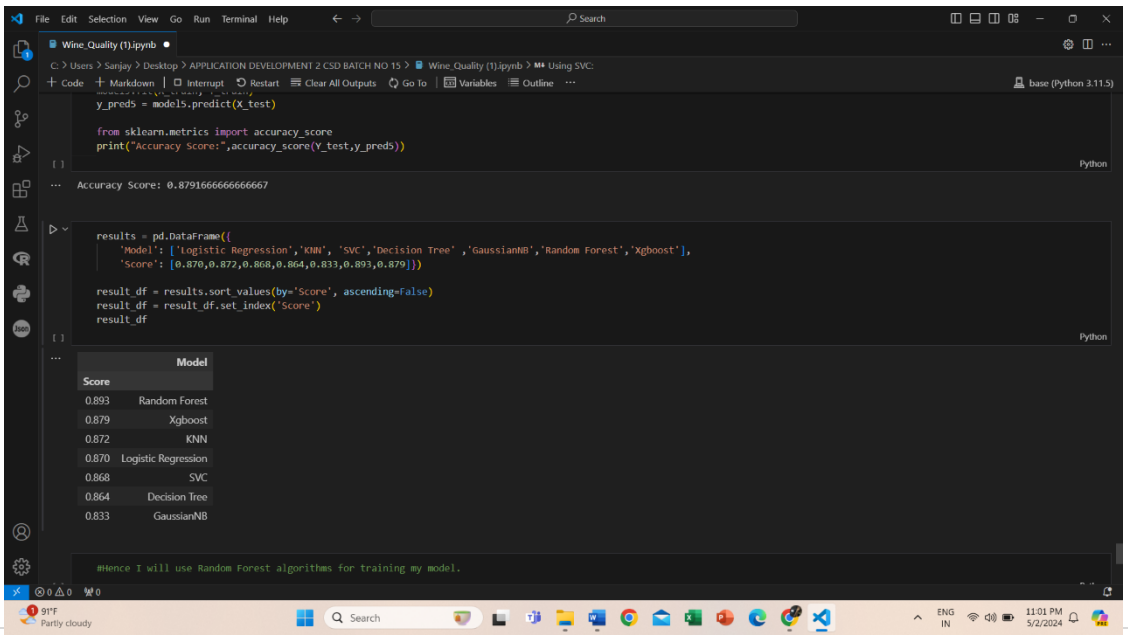

print(Y)
0 0
1 0
2 0
3 0
4 0
1594 0
1595 0
1596 0
1597 0
1598 0
Name: goodquality, Length: 1599, dtype: int64
Feature Importance
from sklearn.linear_model import LogisticRegression
model = LogisticRegression()
from sklearn.ensemble import ExtraTreesClassifier
classifiern = ExtraTreesClassifier()
classifiern.fit(X,Y)
score = classifiern.feature_importances_
print(score)
[0.07559736 0.10044708 0.09365305 0.07359705 0.066992 0.06781859


Splitting Dataset
from sklearn.model_selection import train_test_split
X_train, X_test, Y_train, Y_test = train_test_split(X,Y,test_size=0.3,random_state=7)
LogisticRegression:
from sklearn.linear_model import LogisticRegression
model = LogisticRegression()
model.fit(X_train,Y_train)
Y_pred = model.predict(X_test)
from sklearn.metrics import accuracy_score,confusion_matrix
print("Accuracy Score:",accuracy_score(Y_test,Y_pred))
Accuracy Score: 0.8708333333333333
confusion_mat = confusion_matrix(Y_test,Y_pred)
print(confusion_mat)
[[399 18]
[44 19]]


Using KNN:
from sklearn.neighbors import KNeighborsClassifier
model = KNeighborsClassifier(n_neighbors=3)
model.fit(X_train,Y_train)
y_pred = model.predict(X_test)
from sklearn.metrics import accuracy_score
print("Accuracy Score:",accuracy_score(Y_test,y_pred))
Accuracy Score: 0.8729166666666667
Using SVC:
from sklearn.svm import SVC
model = SVC()
model.fit(X_train,Y_train)
pred_y = model.predict(X_test)
from sklearn.metrics import accuracy_score
print("Accuracy Score:",accuracy_score(Y_test,pred_y))

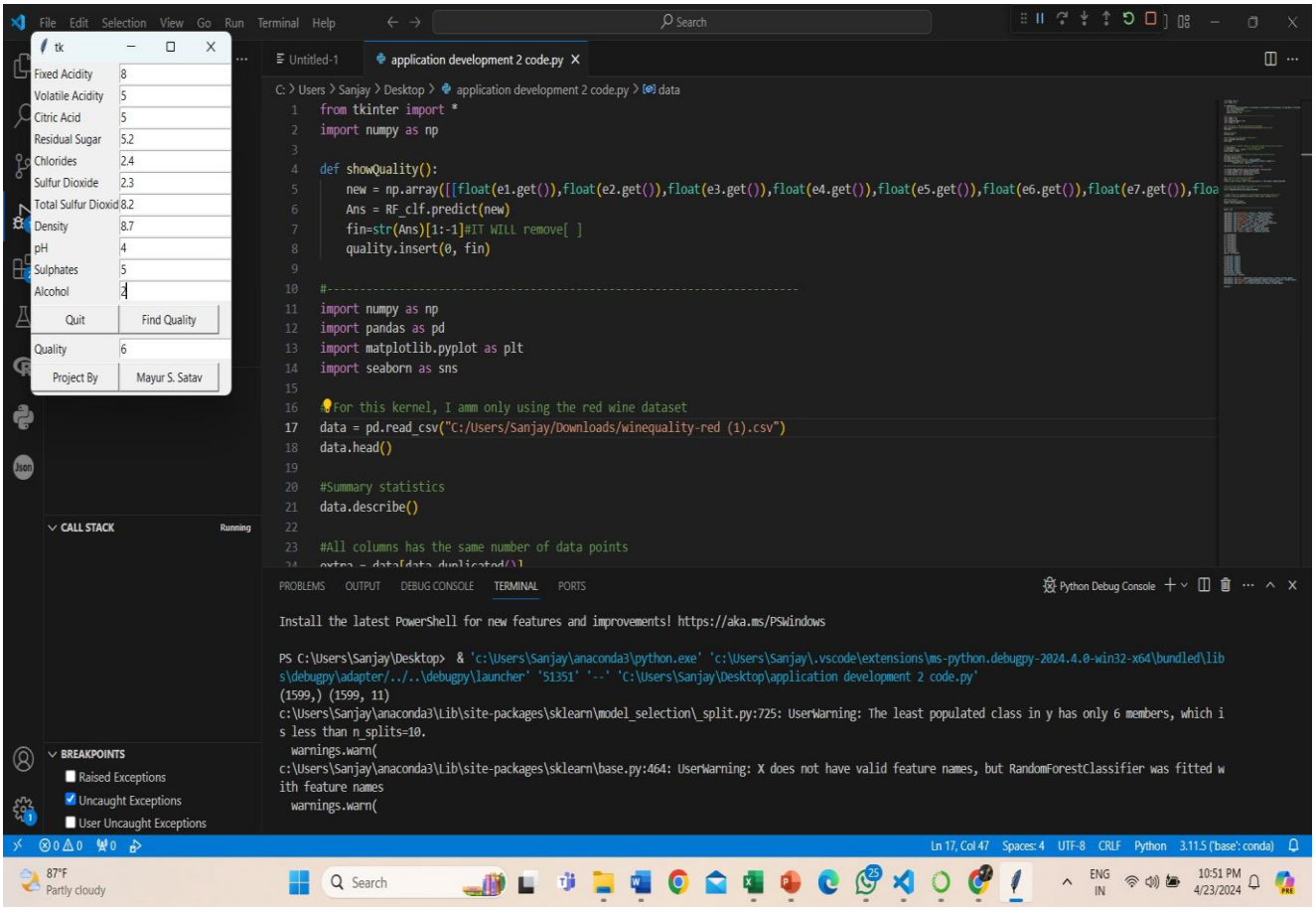

Using Decision Tree:
from sklearn.tree import DecisionTreeClassifier
model = DecisionTreeClassifier(criterion='entropy',random_state=7)
model.fit(X_train,Y_train)
y_pred = model.predict(X_test)
from sklearn.metrics import accuracy_score
print("Accuracy Score:",accuracy_score(Y_test,y_pred))
Accuracy Score: 0.8645833333333334
Using GaussianNB:
from sklearn.naive_bayes import GaussianNB
model3 = GaussianNB()
model3.fit(X_train,Y_train)
y_pred3 = model3.predict(X_test)
from sklearn.metrics import accuracy_score
print("Accuracy Score:",accuracy_score(Y_test,y_pred3))
Accuracy Score: 0.8333333333333334


Using Random Forest:
from sklearn.ensemble import RandomForestClassifier
model2 = RandomForestClassifier(random_state=1)
model2.fit(X_train, Y_train)
y_pred2 = model2.predict(X_test)
from sklearn.metrics import accuracy_score
print("Accuracy Score:",accuracy_score(Y_test,y_pred2))
Accuracy Score: 0.89375
Using Xgboost:
import xgboost as xgb
model5 = xgb.XGBClassifier(random_state=1)
model5.fit(X_train, Y_train)
y_pred5 = model5.predict(X_test)
from sklearn.metrics import accuracy_score
print("Accuracy Score:",accuracy_score(Y_test,y_pred5))
Accuracy Score: 0.8791666666666667

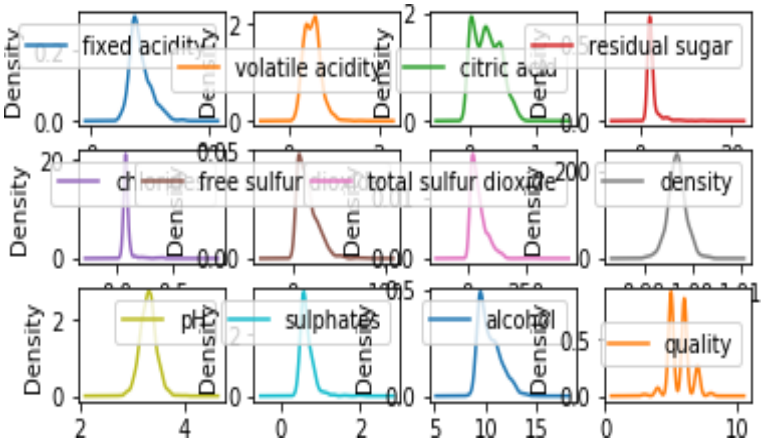

y_pred5 = model5.predict(X_test)
from sklearn.metrics import accuracy_score
print("Accuracy Score:",accuracy_score(Y_test,y_pred5))
Accuracy Score: 0.8791666666666667
results = pd.DataFrame({
'Model': ['Logistic Regression','KNN', 'SVC','Decision Tree' ,'GaussianNB','Random Forest','Xgboost'],
'Score': [0.870,0.872,0.868,0.864,0.833,0.893,0.879]})
result_df = results.sort_values(by='Score', ascending=False)
result_df = result_df.set_index('Score')
result_df
Model
Score
0.893 Random Forest
0.879 Xgboost
0.872 KNN
0.870 Logistic Regression
0.868 SVC
0.864 Decision Tree
0.833 GaussianNB
#Hence I will use Random Forest algorithms for training my model.

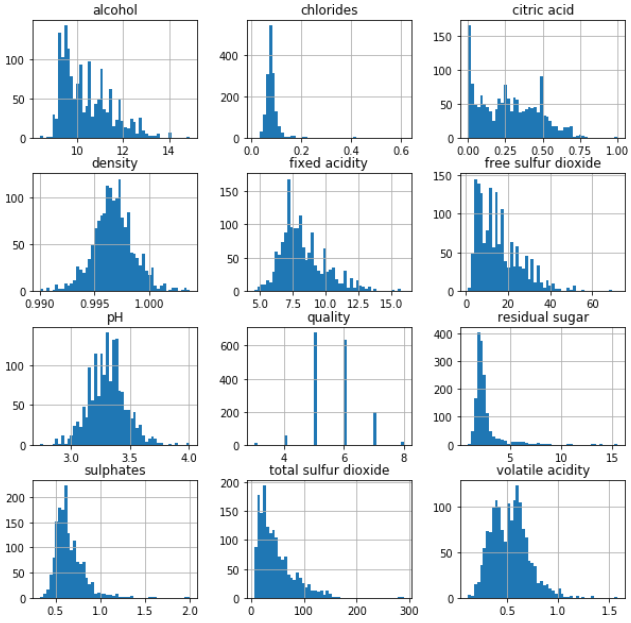
CHAPTER 6
OUTPUT SCREEN

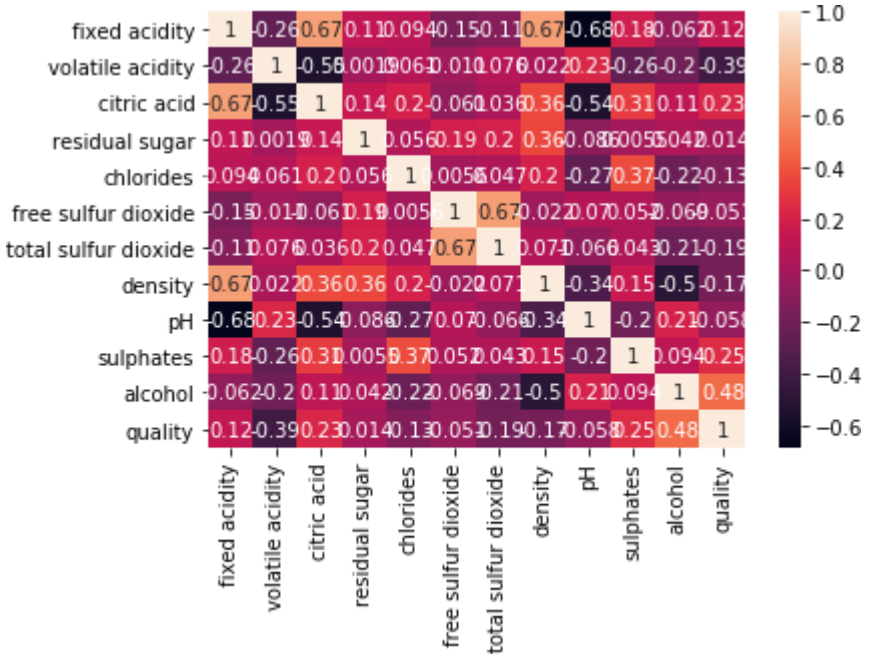
Out Screen of GUII Code :

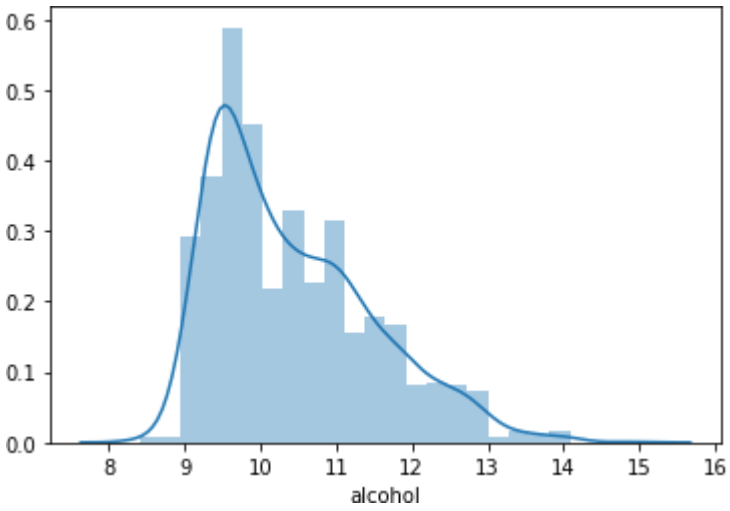


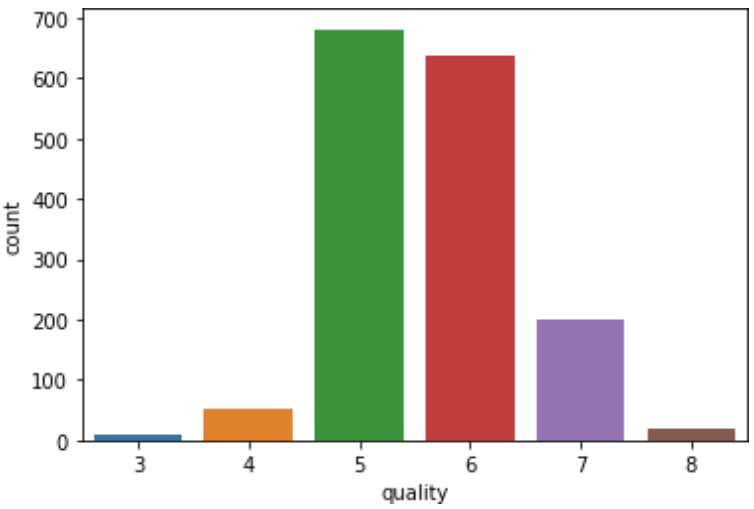
Score	Model
0.893	Random Forest
0.879	Xgboost
0.872	KNN
0.870	Logistic Regression
0.868	SVC
0.864	Decision Tree
0.833	GaussianNB






fixed acidity
volatile acidity
citric acid
residual sugar
chlorides
free sulfur dioxide
total sulfur dioxide
density
pH
sulphates
alcohol
quality
1.0
0.8
0.6
0.4
0.2
0.0
−0.2
−0.4
−0.6


0.6
0.5
0.4
0.3
0.2
0.1
0.0
8
9
10
11
12
13
14
15
16
alcohol


700
600
500
400
300
200
100
0
count
3
4
5
6
7
8
quality

CHAPTER 7
ALGORITHMS

It seems like you've listed a few machine learning algorithms! Each of these algorithms serves a different purpose and has its own strengths and weaknesses. Here's a brief overview of each:

- **Random Forest:** It's an ensemble learning method that operates by constructing multiple decision trees during training time and outputs the mode of the classes (classification) or mean prediction (regression) of the individual trees. It's known for its robustness and ability to handle large data sets with high dimensionality.

- **Support Vector Machine (SVC):** It's a powerful supervised learning algorithm used for classification and regression tasks. SVC finds the optimal hyperplane that best separates data points into different classes. It's effective in high-dimensional spaces and is particularly well-suited for cases where the number of dimensions exceeds the number of samples.

- **Decision Tree:** A decision tree is a flowchart-like structure where an internal node represents a "test" on an attribute, each branch represents the outcome of the test, and each leaf node represents a class label. They're simple to understand and interpret, making them useful for visualizing and understanding decision-making processes.

- **K-Nearest Neighbors (KNN):** It's a simple and effective algorithm used for classification and regression tasks. KNN works by finding the 'k' nearest data points in the feature space and assigns a label to a query point based on the majority class among its neighbors (for classification) or the average of the values of its neighbors (for regression).

- **Logistic Regression:** Despite its name, logistic regression is a linear model for binary classification rather than a regression algorithm. It predicts the probability that an instance belongs to a particular class. It's widely used due to its simplicity, ease of implementation, and interpretability.

- Each algorithm has its own set of advantages and disadvantages, and the choice of which one to use depends on factors like the nature of the data, the size of the dataset, computational resources, and the specific problem you're trying to solve.

- **XG Boost (Extreme Gradient Boosting):** XG Boost is an efficient and scalable implementation of gradient boosting for classification and regression tasks. It's known for its speed and performance, often outperforming other gradient boosting implementations. XG Boost builds a sequence of decision trees iteratively, where each tree corrects the errors made by the previous one. It's highly customizable, allowing users to define various hyperparameters to control the boosting process.

- **Gaussian Naive Bayes:** Naive Bayes is a family of probabilistic algorithms based on Bayes' theorem. Gaussian Naive Bayes assumes that the features follow a Gaussian distribution, meaning that continuous features are assumed to be normally distributed. Despite its simplicity and the "naive" assumption of feature independence, Gaussian Naive Bayes can perform well in practice, especially for classification tasks with a small amount of training data.

- Both XGBoost and Gaussian Naive Bayes have their strengths and weaknesses. XGBoost is often favored for its high performance and ability to handle large datasets, while Gaussian Naive Bayes is appreciated for its simplicity and interpretability. The choice between them depends on factors such as the nature of the data, the size of the dataset, and the specific requirements of the problem at hand.

CHAPTER 8
TESTING AND TEST CASE

8.1 TESTING

Testing wine quality prediction using machine learning involves evaluating the performance of trained models on unseen data to assess their accuracy and generalization capabilities. Here's a brief overview of how testing is typically conducted:

- **Test Set Selection:** A portion of the dataset, separate from the training data, is reserved for testing purposes. This test set should be representative of the overall data distribution and contain examples that the model has not been exposed to during training.

- **Model Evaluation Metrics:** Choose appropriate evaluation metrics based on the nature of the prediction task. For wine quality prediction, metrics like Mean Squared Error (MSE), Root Mean Squared Error (RMSE), Mean Absolute Error (MAE), or classification accuracy may be used, depending on whether it's a regression or classification problem.

- **Testing Procedure:** Feed the test set into the trained model and generate predictions. Compare the model's predictions with the actual wine quality ratings in the test set using the chosen evaluation metrics.

- **Performance Analysis:** Analyze the evaluation metrics to assess the model's performance. A lower error or higher accuracy indicates better predictive capability. Additionally, visualizations such as scatter plots comparing predicted vs. actual values can provide insights into the model's behavior.

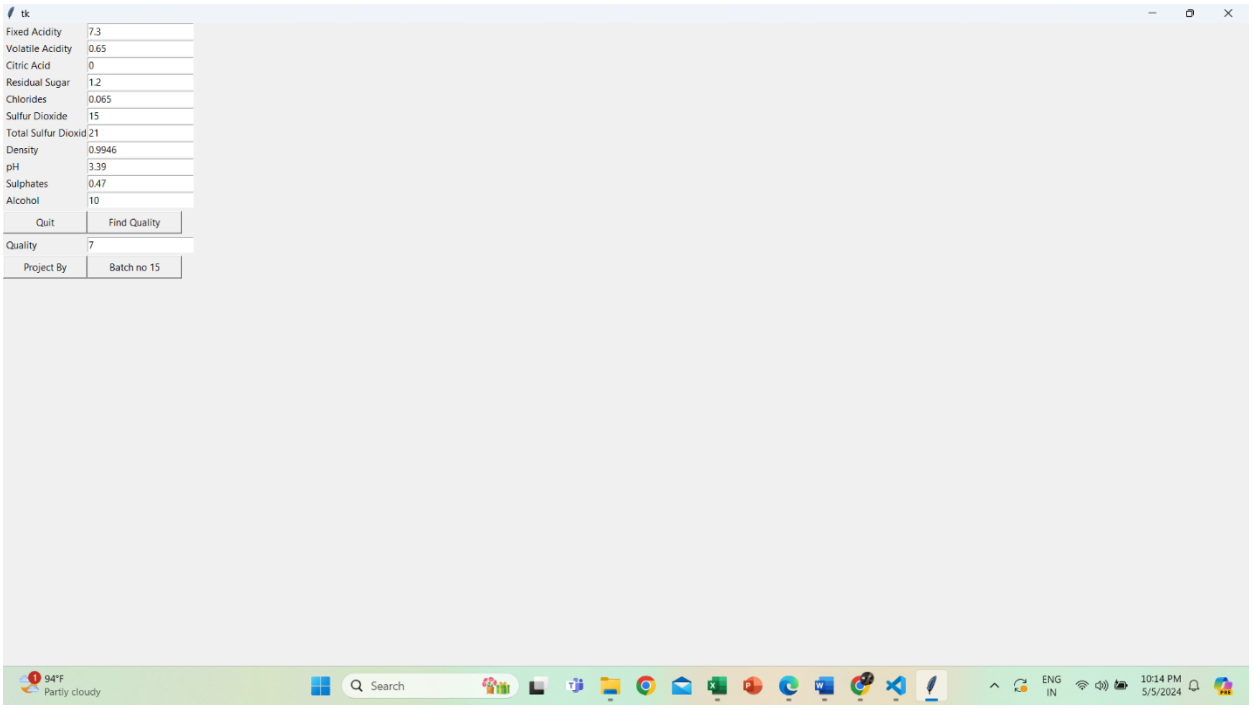
- **Generalization Check:** Ensure that the model performs well not only on the training data but also on unseen test data. If the model's performance on the test set is consistent with its performance on the training set, it suggests good generalization.

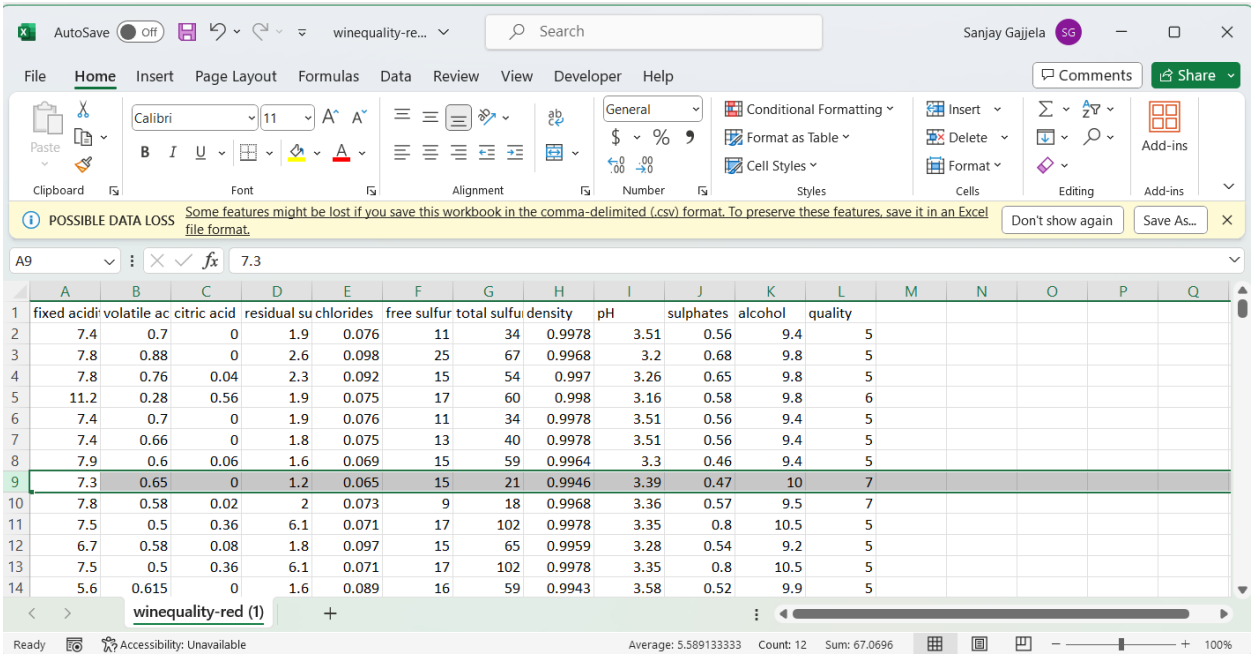
- **Iterative Improvement:** If the model's performance is not satisfactory, consider fine-tuning hyperparameters, adjusting feature selection, or exploring different algorithms. Retest the updated model until satisfactory results are achieved.

- **Final Assessment:** Once the model demonstrates acceptable performance on the test set, it can be considered ready for deployment in real-world applications.

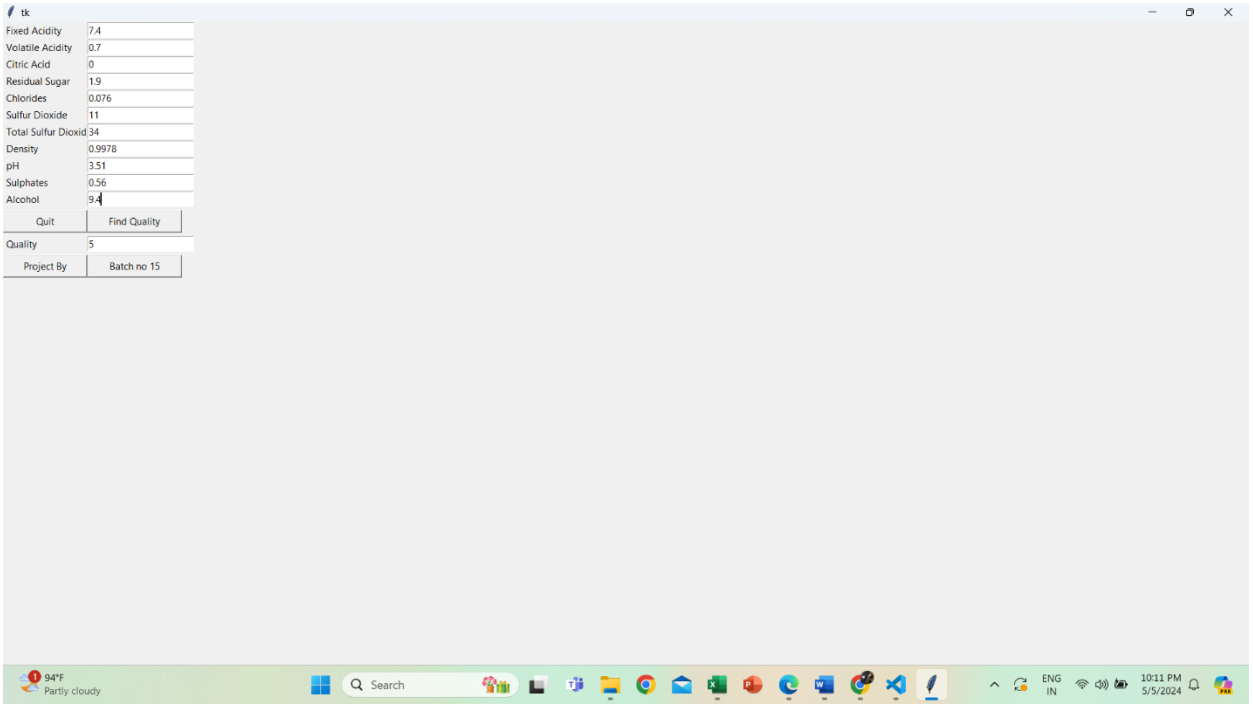
- By rigorously testing machine learning models for wine quality prediction, we can ensure their reliability and effectiveness in practical scenarios, ultimately enhancing decision-making processes in the wine industry.

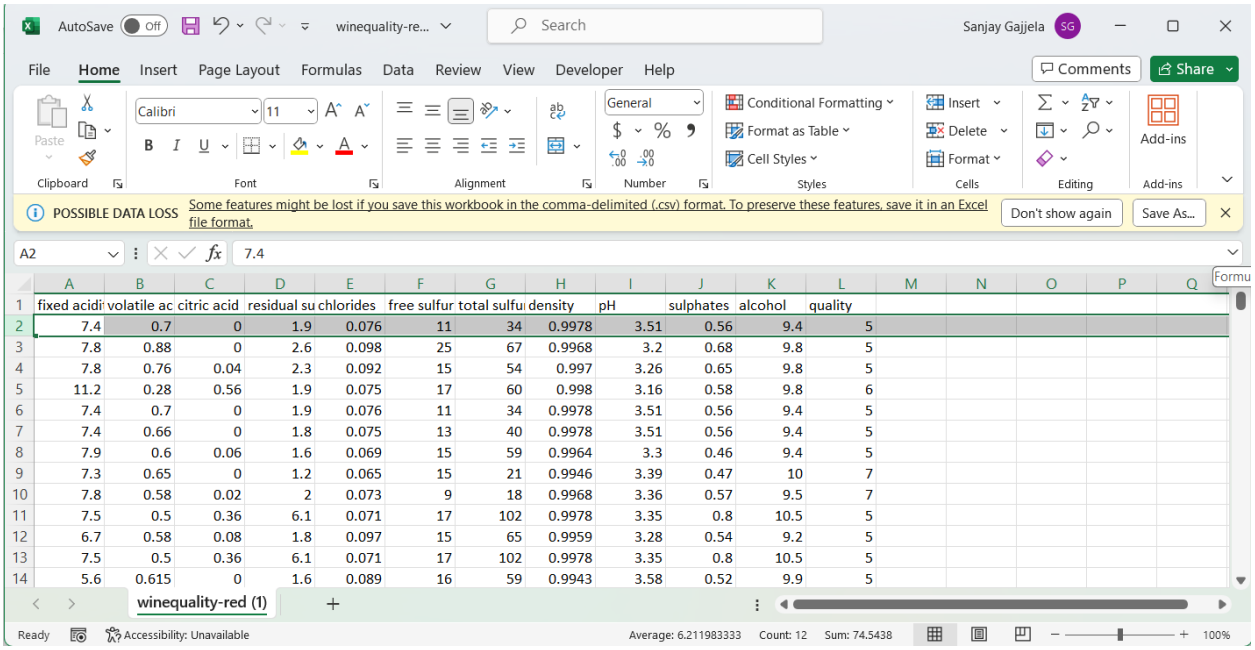
8.2 TEST CASES





	A	B	C	D	E	F	G	H	I	J	K	L
1	fixed acidi	volatile ac	citric acid	residual su	chlorides	free sulfur	total sulfu	density	pH	sulphates	alcohol	quality
2	7.4	0.7	0	1.9	0.076	11	34	0.9978	3.51	0.56	9.4	5
3	7.8	0.88	0	2.6	0.098	25	67	0.9968	3.2	0.68	9.8	5
4	7.8	0.76	0.04	2.3	0.092	15	54	0.997	3.26	0.65	9.8	5
5	11.2	0.28	0.56	1.9	0.075	17	60	0.998	3.16	0.58	9.8	6
6	7.4	0.7	0	1.9	0.076	11	34	0.9978	3.51	0.56	9.4	5
7	7.4	0.66	0	1.8	0.075	13	40	0.9978	3.51	0.56	9.4	5
8	7.9	0.6	0.06	1.6	0.069	15	59	0.9964	3.3	0.46	9.4	5
9	7.3	0.65	0	1.2	0.065	15	21	0.9946	3.39	0.47	10	7
10	7.8	0.58	0.02	2	0.073	9	18	0.9968	3.36	0.57	9.5	7
11	7.5	0.5	0.36	6.1	0.071	17	102	0.9978	3.35	0.8	10.5	5
12	6.7	0.58	0.08	1.8	0.097	15	65	0.9959	3.28	0.54	9.2	5
13	7.5	0.5	0.36	6.1	0.071	17	102	0.9978	3.35	0.8	10.5	5
14	5.6	0.615	0	1.6	0.089	16	59	0.9943	3.58	0.52	9.9	5





	A	B	C	D	E	F	G	H	I	J	K	L
1	fixed acidi	volatile ac	citric acid	residual su	chlorides	free sulfur	total sulfu	density	pH	sulphates	alcohol	quality
2	7.4	0.7	0	1.9	0.076	11	34	0.9978	3.51	0.56	9.4	5
3	7.8	0.88	0	2.6	0.098	25	67	0.9968	3.2	0.68	9.8	5
4	7.8	0.76	0.04	2.3	0.092	15	54	0.997	3.26	0.65	9.8	5
5	11.2	0.28	0.56	1.9	0.075	17	60	0.998	3.16	0.58	9.8	6
6	7.4	0.7	0	1.9	0.076	11	34	0.9978	3.51	0.56	9.4	5
7	7.4	0.66	0	1.8	0.075	13	40	0.9978	3.51	0.56	9.4	5
8	7.9	0.6	0.06	1.6	0.069	15	59	0.9964	3.3	0.46	9.4	5
9	7.3	0.65	0	1.2	0.065	15	21	0.9946	3.39	0.47	10	7
10	7.8	0.58	0.02	2	0.073	9	18	0.9968	3.36	0.57	9.5	7
11	7.5	0.5	0.36	6.1	0.071	17	102	0.9978	3.35	0.8	10.5	5
12	6.7	0.58	0.08	1.8	0.097	15	65	0.9959	3.28	0.54	9.2	5
13	7.5	0.5	0.36	6.1	0.071	17	102	0.9978	3.35	0.8	10.5	5
14	5.6	0.615	0	1.6	0.089	16	59	0.9943	3.58	0.52	9.9	5

CHAPTER 9

CONCLUSION AND FUTURE SCOPE

9.1 CONCLUSION

CONCLUSION: In conclusion, the development of a wine quality prediction system using machine learning and Python offers promising opportunities for enhancing wine production and consumption. By leveraging advanced data analysis techniques and predictive modeling, such a system can provide valuable insights into the factors influencing wine quality and help winemakers optimize their processes.

9.2 FUTURE SCOPE

Looking ahead, there are several venues for future exploration and improvement in this field:

- **Enhanced Prediction Models**: Continued research into advanced machine learning algorithms and techniques can lead to more accurate and prediction models.
- **Integration of Additional Data Sources**: Incorporating additional data sources such as weather patterns, soil composition, and grape variety characteristics can enrich the predictive capabilities of the system
- **Real-time Monitoring and Decision Support**: Developing real-time monitoring capabilities can enable winemakers to make timely decisions and adjustments to optimize wine quality throughout the production process.
- **Personalized Recommendations**: Tailoring recommendations based on individual preferences and tastes can enhance the user experience and satisfaction.

CHAPTER 10

REFERENCES

Predicting wine quality using machine learning (ML) techniques has gained popularity in recent years. Here are some references to get you started:
Link : https://github.com/amberkakkar01/Prediction-of-Wine-Quality

Modeling Wine Preferences by Data Mining from Physicochemical Properties: by Paulo Cortez, António Cerdeira, Fernando Almeida, Telmo Matos, José Reis. This paper explores the use of ML techniques such as decision trees, random forests, and support vector machines (SVM) to predict wine quality based on its physicochemical properties.

Wine Quality Prediction Using Classification Algorithms: A Comparative Study by Alok Kumar Jagadev, Ashok Kumar Jagadev. This study compares the performance of various classification algorithms including k-nearest neighbors (KNN), decision trees, and neural networks for predicting wine quality.

Wine Quality Prediction using Machine Learning Techniques: by Rajib Ranjan Maiti, Srimonti Dutta. This paper investigates the application of ML algorithms such as k-means clustering, decision trees, and SVM for predicting wine quality.

Wine Quality Prediction with Random Forests and Gated Recurrent Units: by Rajeev Agrawal, Brian Guo, Arindam Banerjee. This research explores the use of ensemble methods like random forests and recurrent neural networks (RNNs) for wine quality prediction, achieving competitive results.

Predicting Wine Quality with an Ensemble of Regressors: by Gabriel DeSouza, Amir Joudaki. This paper employs an ensemble learning approach with multiple regression models to predict wine quality.

These references should give you a good starting point for understanding how machine learning techniques can be applied to predict wine quality based on various features and properties.