Documentaion of Wine quality study program

submitted by: Muhammed Shafeeq S

step1 :The code is importing the necessary libraries for data manipulation and visualization.

```
import  pandas as pd
import numpy  as np
from sklearn.preprocessing  import StandardScaler
import  matplotlib.pyplot as plt
import  seaborn as sns
```

step2:

The code is reading two separate CSV files, `winequality-red.csv`, and concatenating them into a single dataframe called `winedata`.The `red_winedata` and `white_winedata` dataframes are created by reading the respective CSV files using the `pd.read_csv()` function. The `sep=';'` parameter specifies that the CSV files are separated by semicolons. The `concat()` function is then used to concatenate the two dataframes vertically, ignoring the index, and the result is stored in the `winedata` dataframe. The concatenation is done twice, which seems to be redundant.

```
red_winedata = pd.read_csv(r'D:\pythons\wine+quality\winequality-red.csv', sep=';')

white_winedata = pd.read_csv(r'D:\pythons\wine+quality\winequality-red.csv', sep=';')

winedata = pd.concat([red_winedata, white_winedata], ignore_index=True)

winedata = pd.concat([red_winedata, white_winedata], ignore_index=True)
```

step 3: The code is performing data cleaning and preprocessing tasks on the `winedata` dataframe.

```
missing_values =  winedata.isnull().sum()

winedata = winedata.drop_duplicates()

winedata.dtypes

winedata['quality'] =  winedata['quality'].astype(int)
```

step 4:

The code is performing feature scaling on the numeric features of the `winedata` dataframe using
the `StandardScaler` class from the `sklearn.preprocessing` module.

```
from  sklearn.preprocessing  import StandardScaler

scaler = StandardScaler()

numeric_features = winedata.select_dtypes(include=[float]).columns
```

```
winedata[numeric_features] =  scaler.fit_transform(winedata[numeric_features])
```

step 5:

The code is splitting the `winedata` dataframe into training and testing sets for machine learning modeling.

```
from sklearn.model_selection import train_test_split

X = winedata.drop(columns=['quality'])
y = winedata['quality']
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2,random_state=42)

winedata.to_csv('cleaned_winedata.csv',index=False)

print(winedata.head())
```

step 6:

The code is reading the cleaned wine data from the 'cleaned_winedata.csv' file into a new dataframe called `winedata`.Then, it calculates the summary statistics of the `winedata` dataframe using the `describe()` function and stores the result in the `summary_stats` variable.

```
winedata = pd.read_csv('cleaned_winedata.csv')


summary_stats = winedata.describe()
```

step 7:

The code is creating a correlation matrix for the `winedata` dataframe using the `corr()` function. The correlation matrix shows the pairwise correlation between all the numeric features in the dataframe.

```
correlation_matrix = winedata.corr()
plt.figure(figsize=(10, 8))
sns.heatmap(correlation_matrix, annot=True, cmap="coolwarm")
plt.title("Correlation Matrix")
plt.show()


plt.figure(figsize=(9, 5))

plt.title("Distribution of Quality by Wine Type")
plt.show()
```

step 8 :

The code is creating a figure with a specific size using `plt.figure(figsize=(12, 5))`.

Then, it defines a list of features to plot, which includes 'alcohol' and 'sulphates'.

```
plt.figure(figsize=(12, 5))
features_to_plot = ['alcohol', 'sulphates']
for feature in features_to_plot:
    sns.histplot(winedata[feature], kde=True, bins=30)
plt.show()
```

step 9 :

The code is creating a figure with a specific size of 8 inches by 5 inches using `plt.figure(figsize=(8, 5))`. Then, it is creating a countplot using `sns.countplot()` to visualize the distribution of wine quality scores. The `x='quality'` parameter specifies that the x-axis should represent the 'quality' column from the `winedata` dataframe. The `data=winedata` parameter specifies the dataframe to use for plotting. The `plt.title()`, `plt.xlabel()`, and `plt.ylabel()` functions are used to set the title, x-axis label, and y-axis

label, respectively. Finally, `plt.show()` is used to display the plot.

```
plt.figure(figsize=(8, 5))
sns.countplot(x='quality', data=winedata)
plt.title("Distribution of Wine Quality Scores")
plt.xlabel("Quality Score")
plt.ylabel("Count")
plt.show()
```

step 10 :

The line of code `correlation_with_quality = winedata.corr()['quality'].sort_values(ascending=False)` is calculating the correlation coefficients between the 'quality' column of the `winedata` dataframe and all other columns. It then sorts the correlation coefficients in descending order. The resulting `correlation_with_quality` variable contains a Series object with the column names as the index and the corresponding correlation coefficients as the values.

```
correlation_with_quality = winedata.corr()['quality'].sort_values(ascending=False)
```

step 11:

The code is creating a figure with a specific size of 9 inches by 5 inches using `plt.figure(figsize=(9, 5))`. Then, it is creating a bar plot using `sns.barplot()` to visualize the correlation coefficients between the features and the wine quality. The `x=correlation_with_quality.values` parameter specifies that the x-axis should represent the correlation coefficients, and the `y=correlation_with_quality.index` parameter specifies that the y-axis should represent the feature names. The `plt.title()`, `plt.xlabel()`, and `plt.show()` functions are used to set the title, x-axis label, and display the plot, respectively.

```
plt.figure(figsize=(9, 5))
sns.barplot( x=correlation_with_quality.values , y=correlation_with_quality.index )
plt.title("Correlation of Features with Wine Quality")
plt.xlabel("Correlation Coefficient")
plt.show()
```

step 12 :

This code is importing the `RandomForestRegressor` class from the `sklearn.ensemble` module. It then creates an instance of the `RandomForestRegressor` class called `model` with a specified random state of 42.
The `RandomForestRegressor` is a machine learning model that uses an ensemble of decision trees to perform regression tasks.

```
from sklearn.ensemble import RandomForestRegressor


model = RandomForestRegressor(random_state=42)
model.fit(X_train, y_train)


feature_importance = model.feature_importances_
feature_names = X.columns
```

step : 13

The code is creating a DataFrame called `feature_importance_df` with two columns: 'Feature' and 'Importance'. The 'Feature' column contains the names of the features from the `X` dataframe, and the 'Importance' column contains the corresponding feature importances calculated by the `RandomForestRegressor` model.

```
feature_importance_df =  pd.DataFrame({'Feature':  feature_names,
'Importance': feature_importance})
feature_importance_df  = feature_importance_df. sort_values(by=
 'Importance', ascending=False )

top_n = 10
print("Top", top_n, "Important Features:")
print(feature_importance_df.head(top_n))
```

step 14:

The code is selecting the top `n` important features from the `feature_importance_df` dataframe, where `n` is set to 10. It then creates a bar plot to visualize the importance of these top features. The x-axis represents the feature names, and the y-axis represents the importance values. The plot is labeled with appropriate axis labels and a title. The `rotation=45` parameter in `plt.xticks()` is used to rotate the x-axis labels by 45 degrees for better readability. Finally, `plt.show()` is used to display the plot.

```
top_n = 10
top_features = feature_importance_df.head(top_n)


plt.figure(figsize=(9, 5))
plt.bar(top_features['Feature'], top_features['Importance'])
plt.xlabel('Feature')
plt.ylabel('Importance')
plt.title('Top {} Important Features'.format(top_n))
plt.xticks(rotation=45)
plt.show()
```