

```
import pandas as pd

# Load dataset
url = "https://archive.ics.uci.edu/ml/machine-learning-databases/wine-quality/winequality-red.csv"
data = pd.read_csv(url, sep=';') # sep means this dataset is seperated by ";"

# Explore dataset
data.head()# Display first 5 rows
```



	fixed acidity	volatile acidity	citric acid	residual sugar	chlorides	free sulfur dioxide	total sulfur dioxide	density	pH	sulphates	alcohol
0	7.4	0.70	0.00	1.9	0.076	11.0	34.0	0.9978	3.51	0.56	9.4
1	7.8	0.88	0.00	2.6	0.098	25.0	67.0	0.9968	3.20	0.68	9.8
2	7.8	0.76	0.04	2.3	0.092	15.0	54.0	0.9970	3.26	0.65	9.8
3	11.2	0.28	0.56	1.9	0.075	17.0	60.0	0.9980	3.16	0.58	9.8
4	7.4	0.70	0.00	1.9	0.076	11.0	34.0	0.9978	3.51	0.56	9.4

Next steps:

[Generate code with data](#)[View recommended plots](#)[New interactive sheet](#)

```
#checking missing values
print("Missing Values:\n",data.isnull().sum())
```



```
Missing Values:
fixed acidity      0
volatile acidity   0
citric acid        0
residual sugar     0
chlorides          0
free sulfur dioxide 0
total sulfur dioxide 0
density            0
pH                 0
sulphates          0
alcohol            0
quality            0
dtype: int64
```

```
#normalize numerical features
from sklearn.preprocessing import MinMaxScaler

scaler=MinMaxScaler()
data.iloc[:, :-1]=scaler.fit_transform(data.iloc[:, :-1]) #fit refer to compute min max #-1 means drop the
```

```
import tensorflow as tf
from tensorflow import keras
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import LabelBinarizer
```

```
#split dataset into features (X) and Target (y)
X=data.iloc[:, :-1].values #All colums except last row
y=data.iloc[:, -1].values # last column (Wine Quality)
```


```
#convert target variable into categorical labels
lb=LabelBinarizer()
y=lb.fit_transform(y)
```

```
#split into training and test tests
X_train,X_test,y_train,y_test=train_test_split(X,y,test_size=0.2,random_state=42) #0.2 means 20%
```

```
#Define the deep learning model
model=keras.Sequential([
    keras.layers.Dense(64,activation='relu',input_shape=(X_train.shape[1],)),
    keras.layers.Dense(32,activation='relu'),
    keras.layers.Dense(len(lb.classes_),activation='softmax') #softmax is used to output layer for mult
])
```

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

```
#Display model summary
model.summary()
```

 /usr/local/lib/python3.11/dist-packages/keras/src/layers/core/dense.py:87: UserWarning: Do not pass super().__init__(activity_regularizer=activity_regularizer, **kwargs)

Model: "sequential"

Layer (type)	Output Shape	Param #
dense (Dense)	(None, 64)	768
dense_1 (Dense)	(None, 32)	2,080
dense_2 (Dense)	(None, 6)	198

Total params: 3,046 (11.90 KB)

Trainable params: 3,046 (11.90 KB)

Non-trainable params: 0 (0.00 B)

```
history=model.fit(X_train,y_train,epochs=50,batch_size=32,validation_split=0.2)
```



```

Epoch 30/50
32/32 ————— 0s 5ms/step - accuracy: 0.6069 - loss: 0.9494 - val_accuracy: 0.6758 -
Epoch 31/50
32/32 ————— 0s 4ms/step - accuracy: 0.5883 - loss: 0.9731 - val_accuracy: 0.6641 -
Epoch 32/50
32/32 ————— 0s 6ms/step - accuracy: 0.6134 - loss: 0.9511 - val_accuracy: 0.6328 -
Epoch 33/50
32/32 ————— 0s 5ms/step - accuracy: 0.6132 - loss: 0.9456 - val_accuracy: 0.6758 -
Epoch 34/50
32/32 ————— 0s 6ms/step - accuracy: 0.6077 - loss: 0.9126 - val_accuracy: 0.6367 -
Epoch 35/50
32/32 ————— 0s 5ms/step - accuracy: 0.6294 - loss: 0.9270 - val_accuracy: 0.6328 -
Epoch 36/50
32/32 ————— 0s 6ms/step - accuracy: 0.6254 - loss: 0.9060 - val_accuracy: 0.6523 -
Epoch 37/50
32/32 ————— 0s 6ms/step - accuracy: 0.6283 - loss: 0.9178 - val_accuracy: 0.6602 -
Epoch 38/50
32/32 ————— 0s 5ms/step - accuracy: 0.5810 - loss: 0.9477 - val_accuracy: 0.6328 -
Epoch 39/50
32/32 ————— 0s 6ms/step - accuracy: 0.6135 - loss: 0.9148 - val_accuracy: 0.6641 -
Epoch 40/50
32/32 ————— 0s 5ms/step - accuracy: 0.6397 - loss: 0.9191 - val_accuracy: 0.6602 -
Epoch 41/50
32/32 ————— 0s 5ms/step - accuracy: 0.5938 - loss: 0.9463 - val_accuracy: 0.6562 -
Epoch 42/50
32/32 ————— 0s 5ms/step - accuracy: 0.6247 - loss: 0.8984 - val_accuracy: 0.6484 -
Epoch 43/50
32/32 ————— 0s 5ms/step - accuracy: 0.6095 - loss: 0.9141 - val_accuracy: 0.6602 -
Epoch 44/50
32/32 ————— 0s 5ms/step - accuracy: 0.6218 - loss: 0.9182 - val_accuracy: 0.6797 -
Epoch 45/50
32/32 ————— 0s 6ms/step - accuracy: 0.6170 - loss: 0.9263 - val_accuracy: 0.6758 -
Epoch 46/50
32/32 ————— 0s 6ms/step - accuracy: 0.6372 - loss: 0.9095 - val_accuracy: 0.6484 -
Epoch 47/50
32/32 ————— 0s 7ms/step - accuracy: 0.6335 - loss: 0.8817 - val_accuracy: 0.6484 -
Epoch 48/50
32/32 ————— 0s 7ms/step - accuracy: 0.6184 - loss: 0.8992 - val_accuracy: 0.6602 -
Epoch 49/50
32/32 ————— 0s 5ms/step - accuracy: 0.6287 - loss: 0.9174 - val_accuracy: 0.6758 -
Epoch 50/50
32/32 ————— 0s 5ms/step - accuracy: 0.6484 - loss: 0.9055 - val_accuracy: 0.6836 -

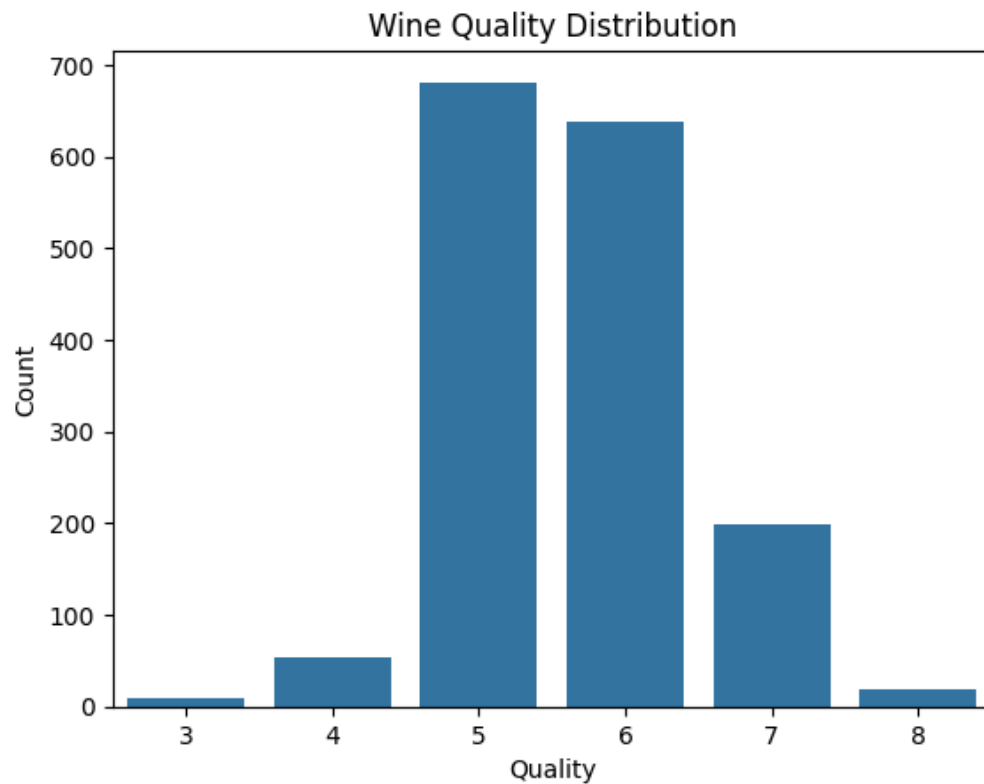
```

```

import seaborn as sns
import matplotlib.pyplot as plt

#Display distribution of wine quality
sns.countplot(data=data,x='quality')
plt.title("Wine Quality Distribution")
plt.xlabel("Quality")
plt.ylabel("Count")
plt.show()

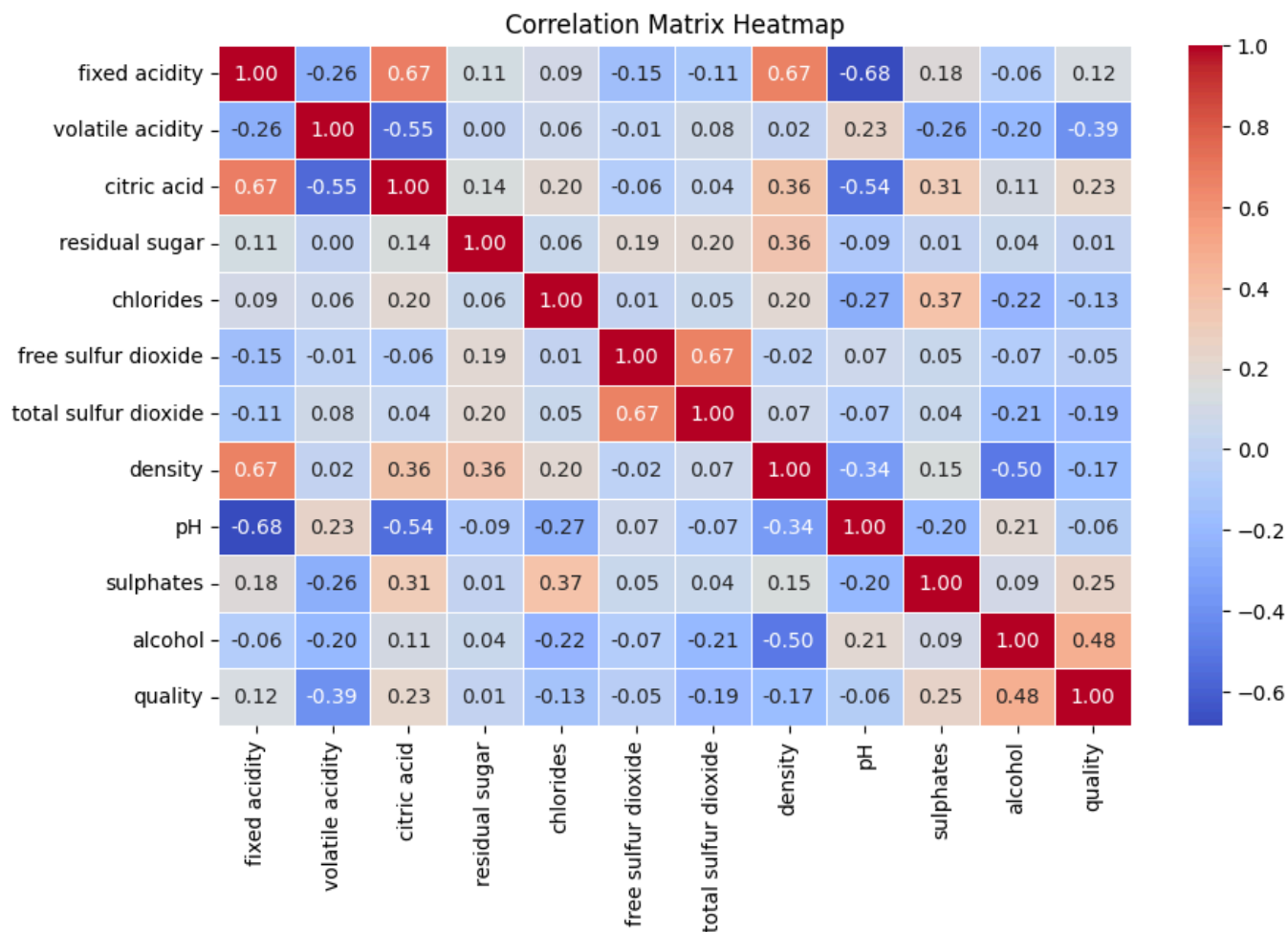
```



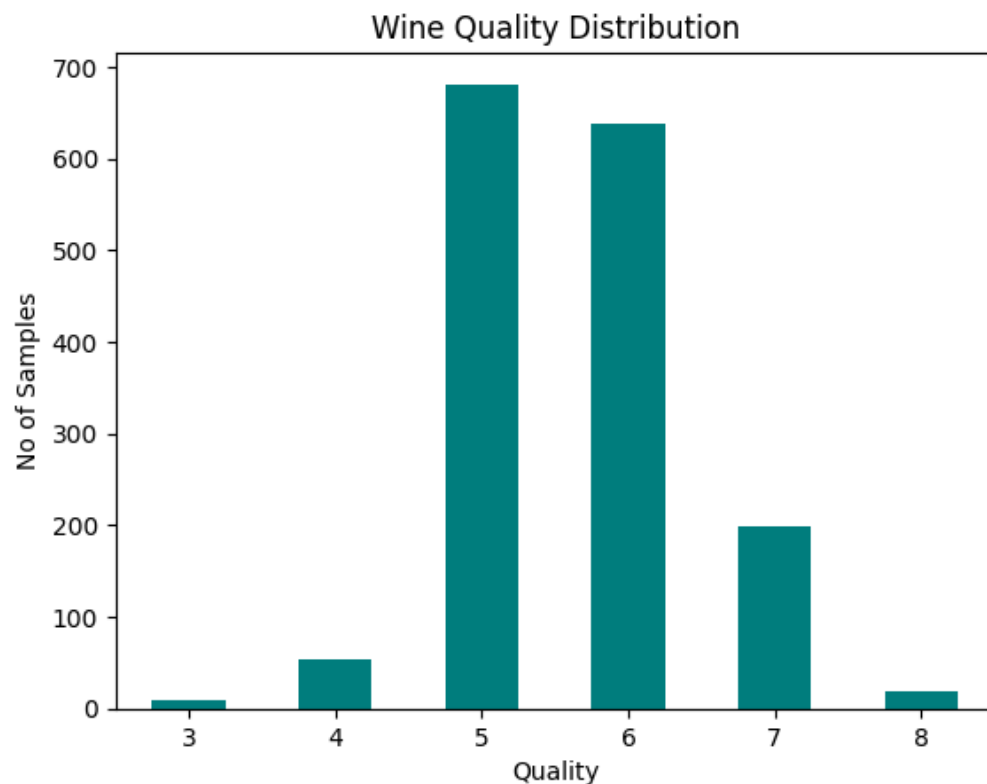
```
import numpy as np

#compute correlation matrix
corr_matrix=data.corr()

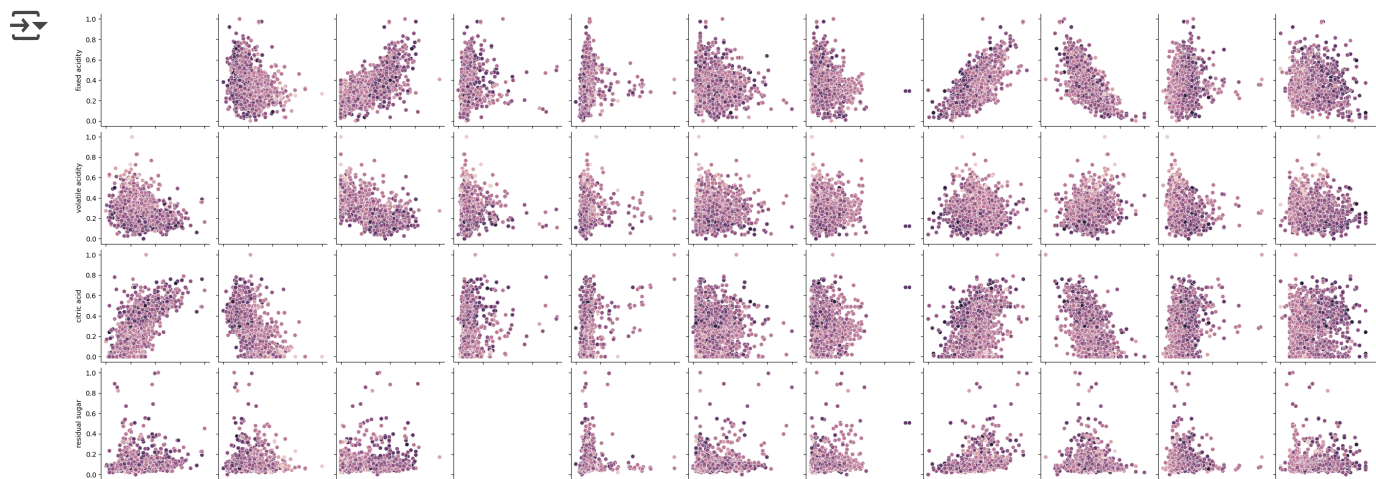
#create heatmap of correlation matrix
plt.figure(figsize=(10,6))
sns.heatmap(corr_matrix,annot=True,cmap='coolwarm', fmt=".2f", linewidths=0.5) # coolwarm means color pale
plt.title("Correlation Matrix Heatmap")
plt.show()
```



```
data["quality"].value_counts().sort_index().plot(kind="bar", color="teal") # Changed 'colr' to 'color'
plt.title("Wine Quality Distribution")
plt.xlabel("Quality")
plt.ylabel("No of Samples")
plt.xticks(rotation=0) #rotation means column name rotation. when we have word names as columns we can use
plt.show()
```



```
sns.pairplot(data,hue="quality", diag_kind='kde') #kde means diagonal kind  
plt.show()
```



```
plt.figure(figsize=(12,8))
for i,col in enumerate (data.columns[:-1]): #Exclude 'quality' column
    plt.subplot(4,3,i+1)
    sns.boxplot(x=data["quality"],y=data[col])
    plt.title(col)
    plt.xlabel("Quality")
    plt.tight_layout()
plt.show()
```

