ASSIGNMENT 4

HARSH KUMAR

21BDS0391

harsh.kumar2021d@vitstudent.ac.in

```
HARSH KUMAR
21BDS0391
harsh.kumar2021d@vitstudent.ac.in
```

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler
from sklearn.ensemble import RandomForestClassifier
from sklearn.metrics import classification_report, accuracy_score
```

1) Loading the Dataset

```
data = pd.read_csv('winequality-red.csv')
```

```
data.head()
```

	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
4	7.4	0.70	0.00	1.9	0.076	11.0	34.0	0.9978	3.51

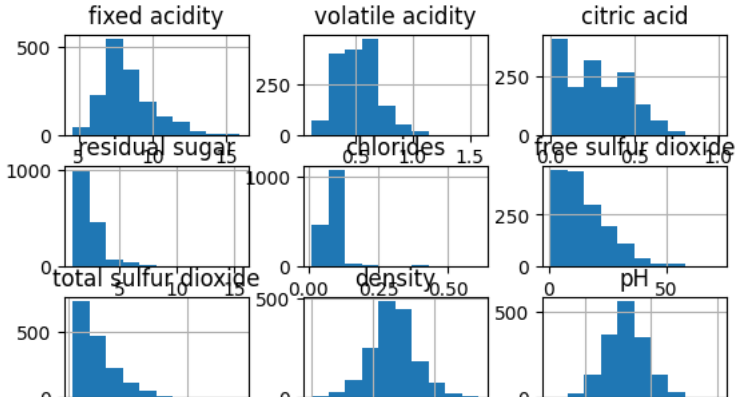
2) Data preprocessing including visualization

```
# Data preprocessing and visualization
print(data.isnull().sum())
```

```
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
```

```
# Visualize data distribution or correlations
data.hist()
```

```
array([[<Axes: title={'center': 'fixed acidity'}>,
        <Axes: title={'center': 'volatile acidity'}>,
        <Axes: title={'center': 'citric acid'}>],
       [<Axes: title={'center': 'residual sugar'}>,
        <Axes: title={'center': 'chlorides'}>,
        <Axes: title={'center': 'free sulfur dioxide'}>],
       [<Axes: title={'center': 'total sulfur dioxide'}>,
        <Axes: title={'center': 'density'}>,
        <Axes: title={'center': 'pH'}>],
       [<Axes: title={'center': 'sulphates'}>,
        <Axes: title={'center': 'alcohol'}>,
        <Axes: title={'center': 'quality'}>]], dtype=object)
```

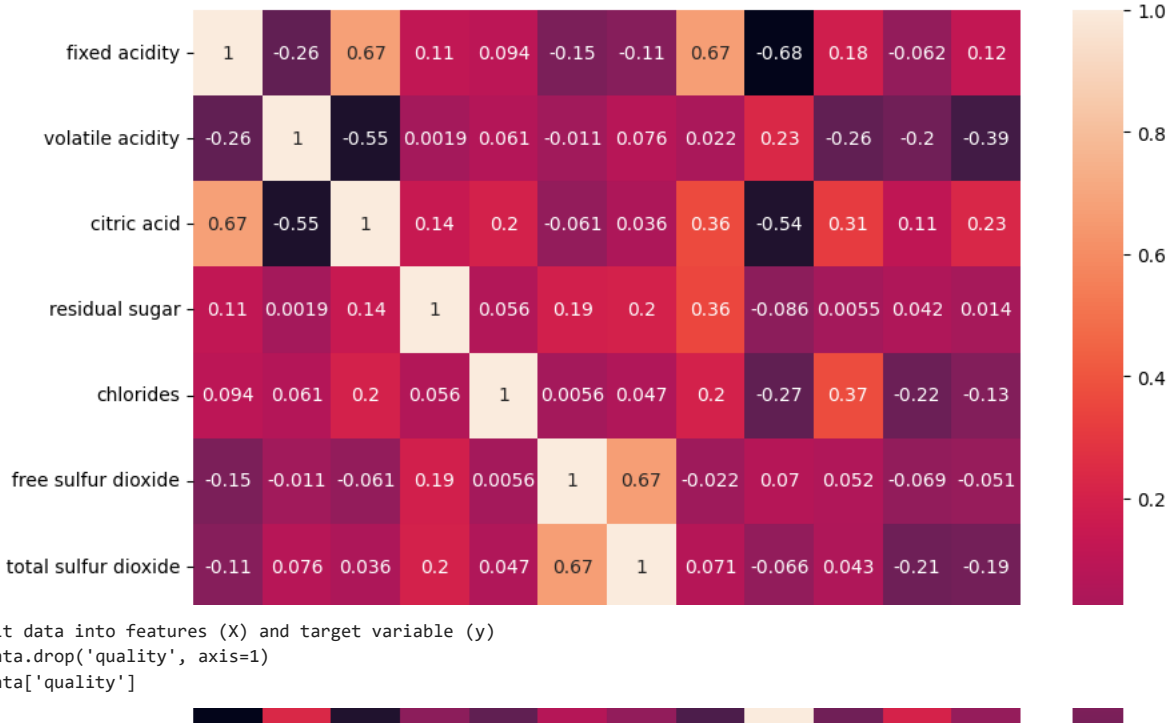



```
corr = data.corr()
corr
```

	fixed acidity	volatile acidity	citric acid	residual sugar	chlorides	free sulfur dioxide	total sulfur dioxide
fixed acidity	1.000000	-0.256131	0.671703	0.114777	0.093705	-0.153794	-0.113181
volatile acidity	-0.256131	1.000000	-0.552496	0.001918	0.061298	-0.010504	0.076470
citric acid	0.671703	-0.552496	1.000000	0.143577	0.203823	-0.060978	0.035533
residual sugar	0.114777	0.001918	0.143577	1.000000	0.055610	0.187049	0.203028
chlorides	0.093705	0.061298	0.203823	0.055610	1.000000	0.005562	0.047400
free sulfur dioxide	-0.153794	-0.010504	-0.060978	0.187049	0.005562	1.000000	0.667666
total sulfur dioxide	-0.113181	0.076470	0.035533	0.203028	0.047400	0.667666	1.000000
density	0.668047	0.022026	0.364947	0.355283	0.200632	-0.021946	0.071269
pH	-0.682978	0.234937	-0.541904	-0.085652	-0.265026	0.070377	-0.066495
sulphates	0.183006	-0.260987	0.312770	0.005527	0.371260	0.051658	0.042947
alcohol	-0.061668	-0.202288	0.109903	0.042075	-0.221141	-0.069408	-0.205654
quality	0.124052	-0.390558	0.226373	0.013732	-0.128907	-0.050656	-0.185100

```
plt.subplots(figsize=(10,10))
sns.heatmap(corr,annot=True)
```

```
<Axes: >
```




```
# Split data into features (X) and target variable (y)
X = data.drop('quality', axis=1)
y = data['quality']
```

```
X.head()
```

	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
4	7.4	0.70	0.00	1.9	0.076	11.0	34.0	0.9978	3.51

```
y.head()
```

```
0    5
1    5
2    5
3    6
4    5
Name: quality, dtype: int64
```

```
# Split data 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)
```

```
X_train
```

	fixed acidity	volatile acidity	citric acid	residual sugar	chlorides	free sulfur dioxide	total sulfur dioxide	density	
493	8.7	0.690	0.31	3.0	0.086	23.0	81.0	1.00020	3
354	6.1	0.210	0.40	1.4	0.066	40.5	165.0	0.99120	3
342	10.9	0.390	0.47	1.8	0.118	6.0	14.0	0.99820	3
834	8.8	0.685	0.26	1.6	0.088	16.0	23.0	0.99694	3
705	8.4	1.035	0.15	6.0	0.073	11.0	54.0	0.99900	3
...	...	...	...	...	...	...	...	...	
1130	9.1	0.600	0.00	1.9	0.058	5.0	10.0	0.99770	3
1294	8.2	0.635	0.10	2.1	0.073	25.0	60.0	0.99638	3
860	7.2	0.620	0.06	2.7	0.077	15.0	85.0	0.99746	3
1459	7.9	0.200	0.35	1.7	0.054	7.0	15.0	0.99458	3
1126	5.8	0.290	0.26	1.7	0.063	3.0	11.0	0.99150	3

1279 rows × 11 columns

```
y_train
```

```
493     6
354     6
342     6
834     5
705     5
       ..
1130    6
1294    6
860     5
1459    7
1126    6
Name: quality, Length: 1279, dtype: int64
```

```
X_test
```

	fixed acidity	volatile acidity	citric acid	residual sugar	chlorides	free sulfur dioxide	total sulfur dioxide	density	
803	7.7	0.560	0.08	2.50	0.114	14.0	46.0	0.99710	3
124	7.8	0.500	0.17	1.60	0.082	21.0	102.0	0.99600	3
350	10.7	0.670	0.22	2.70	0.107	17.0	34.0	1.00040	3
682	8.5	0.460	0.31	2.25	0.078	32.0	58.0	0.99800	3
1326	6.7	0.460	0.24	1.70	0.077	18.0	34.0	0.99480	3
...	...	...	...	...	...	...	...	...	
1259	6.8	0.640	0.00	2.70	0.123	15.0	33.0	0.99538	3
1295	6.6	0.630	0.00	4.30	0.093	51.0	77.5	0.99558	3
1155	8.3	0.600	0.25	2.20	0.118	9.0	38.0	0.99616	3
963	8.8	0.270	0.39	2.00	0.100	20.0	27.0	0.99546	3
704	9.1	0.765	0.04	1.60	0.078	4.0	14.0	0.99800	3

320 rows × 11 columns

```
y_test
```

```
803     6
124     5
350     6
682     5
1326    6
       ..
1259    6
1295    5
1155    5
963     6
704     4
Name: quality, Length: 320, dtype: int64
```

```
# Standardize features (if needed)
scaler = StandardScaler()
X_train = scaler.fit_transform(X_train)
X_test = scaler.transform(X_test)
```

```
X_train
```

```
array([[ 0.21833164,  0.88971201,  0.19209222, ...,  1.09349989,
         0.45822284,  1.12317723],
       [-1.29016623, -1.78878251,  0.65275338, ..., -0.40043872,
        -0.40119696,  1.40827174],
       [ 1.49475291, -0.78434707,  1.01104539, ..., -0.07566946,
         0.51551749, -0.58738978],
       ...,
       [-0.65195559,  0.49909822, -1.08752211, ...,  1.28836145,
        -0.68767023, -0.87248428],
       [-0.24582155, -1.84458448,  0.39683051, ...,  0.05423824,
         0.80199076,  1.40827174],
       [-1.46422367, -1.34236676, -0.06383064, ...,  0.50891521,
        -0.68767023,  2.92877575]])
```

```
X_test
```

```
array([[-3.61859850e-01,  1.64286407e-01, -9.85152962e-01, ...,
        -4.65392578e-01, -1.34389336e-04, -7.77452782e-01],
       [-3.03840702e-01, -1.70525408e-01, -5.24491803e-01, ...,
         5.08915214e-01, -1.03143815e+00, -8.72484283e-01],
       [ 1.37871461e+00,  7.78108067e-01, -2.68568937e-01, ...,
```

```
 -2.05577167e-01,  1.83329452e+00, -4.92358280e-01],
...,
[-1.37449586e-02,  3.87494284e-01, -1.15015218e-01, ...,
 -1.04997725e+00, -7.44964886e-01, -5.87389780e-01],
[ 2.76350785e-01, -1.45397070e+00,  6.01568807e-01, ...,
 -1.04997725e+00,  1.71749571e-01,  7.43051230e-01],
[ 4.50408230e-01,  1.30822677e+00, -1.18989125e+00, ...,
 -1.40623314e-01, -6.87670232e-01, -6.82421281e-01]])
```

3) Machine Learning Model building

```
# Machine Learning Model building
# Here, we'll use a random forest classifier as an example. You can choose any other suitable algorithm.
clf = RandomForestClassifier(n_estimators=100, random_state=42)
clf.fit(X_train, y_train)
```

```
RandomForestClassifier
RandomForestClassifier(random_state=42)
```

4) Evaluate the model

```
# Evaluate the model
y_pred = clf.predict(X_test)
accuracy = accuracy_score(y_test, y_pred)
classification_report_result = classification_report(y_test, y_pred)
```

```
/usr/local/lib/python3.10/dist-packages/sklearn/metrics/_classification.py:1344: UndefinedMetricWarning: Precision and F-score are
  _warn_prf(average, modifier, msg_start, len(result))
/usr/local/lib/python3.10/dist-packages/sklearn/metrics/_classification.py:1344: UndefinedMetricWarning: Precision and F-score are
  _warn_prf(average, modifier, msg_start, len(result))
/usr/local/lib/python3.10/dist-packages/sklearn/metrics/_classification.py:1344: UndefinedMetricWarning: Precision and F-score are
  _warn_prf(average, modifier, msg_start, len(result))
```

```
print(f'Accuracy: {accuracy}')
```

```
Accuracy: 0.659375
```

```
print('Classification Report:\n', classification_report_result)
```

```
Classification Report:
                precision    recall  f1-score   support

           3       0.00      0.00      0.00         1
           4       0.00      0.00      0.00        10
           5       0.71      0.74      0.72       130
           6       0.63      0.70      0.66       132
           7       0.64      0.55      0.59        42
           8       0.00      0.00      0.00         5

    accuracy                           0.66       320
   macro avg       0.33      0.33      0.33       320
weighted avg       0.63      0.66      0.64       320
```

5) Test with random observation

```
manual_input = np.array([[9.6, 0.84, 0.09, 2.0, 0.090, 15.0, 50.0, 0.9990, 3.55, 0.61, 9.5]])
predicted_quality = clf.predict(manual_input)
```

```
predicted_quality
```

```
array([5])
```