```
from google.colab import drive
drive.mount('/content/drive')
```

```
Mounted at /content/drive
```

```
import pandas as pd
import matplotlib.pyplot as plt
from sklearn import tree
from sklearn.model_selection import train_test_split, cross_val_score, GridSearchCV
from sklearn.metrics import classification_report, confusion_matrix, accuracy_score, precision_score, recall_score
```

```
url = 'https://archive.ics.uci.edu/ml/machine-learning-databases/wine-quality/winequality-white.csv'
df = pd.read_csv(url,sep = ";")
print(df.head())
```

```
   fixed acidity  volatile acidity  citric acid  residual sugar  chlorides  \
0            7.0              0.27         0.36            20.7      0.045
1            6.3              0.30         0.34             1.6      0.049
2            8.1              0.28         0.40             6.9      0.050
3            7.2              0.23         0.32             8.5      0.058
4            7.2              0.23         0.32             8.5      0.058

   free sulfur dioxide  total sulfur dioxide  density    pH  sulphates  \
0                 45.0                 170.0   1.0010  3.00       0.45
1                 14.0                 132.0   0.9940  3.30       0.49
2                 30.0                  97.0   0.9951  3.26       0.44
3                 47.0                 186.0   0.9956  3.19       0.40
4                 47.0                 186.0   0.9956  3.19       0.40

   alcohol  quality
0      8.8        6
1      9.5        6
2     10.1        6
3      9.9        6
4      9.9        6
```

```
# 1. Check for null values
print(df.head())
print(df.isnull().sum())
```

```
   fixed acidity  volatile acidity  citric acid  residual sugar  chlorides  \
0            7.0              0.27         0.36            20.7      0.045
1            6.3              0.30         0.34             1.6      0.049
2            8.1              0.28         0.40             6.9      0.050
3            7.2              0.23         0.32             8.5      0.058
4            7.2              0.23         0.32             8.5      0.058

   free sulfur dioxide  total sulfur dioxide  density    pH  sulphates  \
0                 45.0                 170.0   1.0010  3.00       0.45
1                 14.0                 132.0   0.9940  3.30       0.49
2                 30.0                  97.0   0.9951  3.26       0.44
3                 47.0                 186.0   0.9956  3.19       0.40
4                 47.0                 186.0   0.9956  3.19       0.40

   alcohol  quality
0      8.8        6
1      9.5        6
2     10.1        6
3      9.9        6
4      9.9        6
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
```

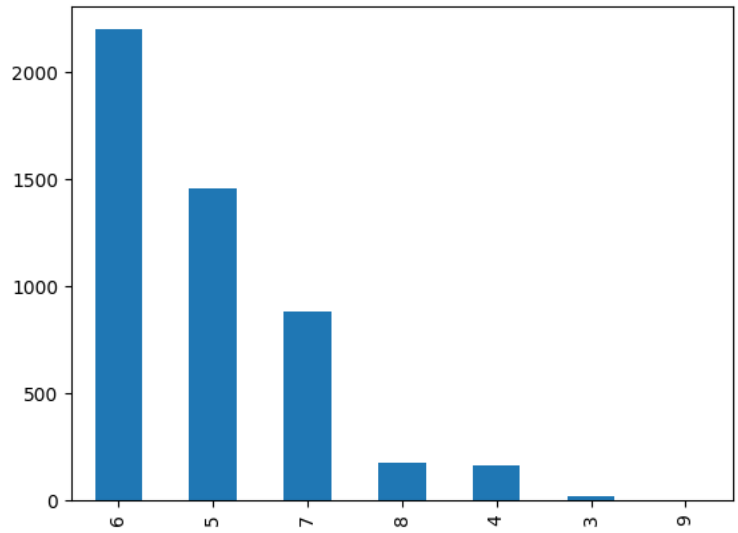
```
# 2. Replace null values if any
df.fillna(df.mean(), inplace=True)
print(df.head())
```

```
   fixed acidity  volatile acidity  citric acid  residual sugar  chlorides  \
0            7.0              0.27         0.36            20.7      0.045
1            6.3              0.30         0.34             1.6      0.049
2            8.1              0.28         0.40             6.9      0.050
```

```
3            7.2             0.23         0.32             8.5      0.058
4            7.2             0.23         0.32             8.5      0.058

   free sulfur dioxide  total sulfur dioxide  density    pH  sulphates  \
0                 45.0                 170.0   1.0010  3.00       0.45
1                 14.0                 132.0   0.9940  3.30       0.49
2                 30.0                  97.0   0.9951  3.26       0.44
3                 47.0                 186.0   0.9956  3.19       0.40
4                 47.0                 186.0   0.9956  3.19       0.40

   alcohol  quality
0      8.8        6
1      9.5        6
2     10.1        6
3      9.9        6
4      9.9        6
```

```
# 3. Remove records with many null values if any
df = df.dropna(thresh=len(df.columns)-2)
```

```
# 4. Draw the stat of number of each quality wine
df['quality'].value_counts().plot(kind='bar')
plt.show()
```




```
# 5. Map the Quality to numbers
quality_mapping = {quality: i for i, quality in enumerate(df['quality'].unique())}
df['quality'] = df['quality'].map(quality_mapping)
```

```
# 6. Construct the DT using ID3 and CART
X = df.drop('quality', axis=1)
y = df['quality']
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)
```

```
# ID3
clf_id3 = tree.DecisionTreeClassifier(criterion='entropy')
clf_id3 = clf_id3.fit(X_train, y_train)
```

```
# CART
clf_cart = tree.DecisionTreeClassifier(criterion='gini')
clf_cart = clf_cart.fit(X_train, y_train)
```

```
# 7. Measure the accuracy both the DT using K-FOLD Cross validation
scores_id3 = cross_val_score(clf_id3, X, y, cv=5)
scores_cart = cross_val_score(clf_cart, X, y, cv=5)
```

```
# 8. Print classification report, its confusion matrix, the accuracy, precision and recall score
y_pred_id3 = clf_id3.predict(X_test)
y_pred_cart = clf_cart.predict(X_test)
```

```
print(classification_report(y_test, y_pred_id3))
print(confusion_matrix(y_test, y_pred_id3))
print(accuracy_score(y_test, y_pred_id3))
print(precision_score(y_test, y_pred_id3, average='weighted'))
print(recall_score(y_test, y_pred_id3, average='weighted'))
```

```
                  precision    recall  f1-score   support

               0       0.64      0.66      0.65       432
               1       0.66      0.58      0.62       291
               2       0.60      0.60      0.60       192
               3       0.35      0.49      0.41        35
               4       0.28      0.36      0.32        25
               5       0.00      0.00      0.00         5
               6       0.00      0.00      0.00         0

        accuracy                           0.61       980
       macro avg       0.36      0.38      0.37       980
    weighted avg       0.61      0.61      0.61       980

    [[284  73  53  10   9   0   3]
     [ 92 169  11   6  13   0   0]
     [ 53   6 116  15   1   0   1]
     [ 11   0   7  17   0   0   0]
     [  3   8   3   0   9   2   0]
     [  3   0   2   0   0   0   0]
     [  0   0   0   0   0   0   0]]
    0.6071428571428571
    0.6149160753847505
    0.6071428571428571
    /usr/local/lib/python3.10/dist-packages/sklearn/metrics/_classification.py:1344: UndefinedMetricWarning: Recall and F-sc
      _warn_prf(average, modifier, msg_start, len(result))
    /usr/local/lib/python3.10/dist-packages/sklearn/metrics/_classification.py:1344: UndefinedMetricWarning: Recall and F-sc
      _warn_prf(average, modifier, msg_start, len(result))
    /usr/local/lib/python3.10/dist-packages/sklearn/metrics/_classification.py:1344: UndefinedMetricWarning: Recall and F-sc
      _warn_prf(average, modifier, msg_start, len(result))
    /usr/local/lib/python3.10/dist-packages/sklearn/metrics/_classification.py:1344: UndefinedMetricWarning: Recall is ill-d
      _warn_prf(average, modifier, msg_start, len(result))
```

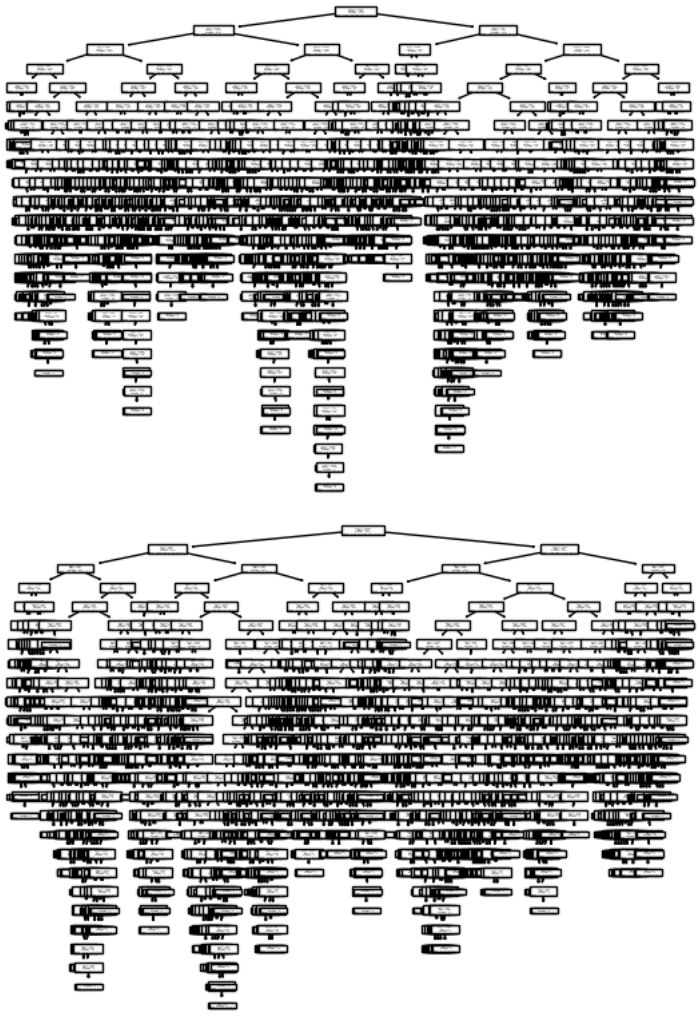
```
print(classification_report(y_test, y_pred_cart))
print(confusion_matrix(y_test, y_pred_cart))
print(accuracy_score(y_test, y_pred_cart))
print(precision_score(y_test, y_pred_cart, average='weighted'))
print(recall_score(y_test, y_pred_cart, average='weighted'))
```

```
                  precision    recall  f1-score   support

               0       0.65      0.62      0.64       432
               1       0.66      0.65      0.65       291
               2       0.61      0.58      0.59       192
               3       0.30      0.43      0.35        35
               4       0.21      0.32      0.25        25
               5       0.00      0.00      0.00         5
               6       0.00      0.00      0.00         0

        accuracy                           0.61       980
       macro avg       0.35      0.37      0.36       980
    weighted avg       0.62      0.61      0.61       980

    [[270  80  52  14  14   2   0]
     [ 74 189  11   3  14   0   0]
     [ 53   8 112  17   2   0   0]
     [  9   0  10  15   0   0   1]
     [  7   9   0   1   8   0   0]
     [  2   2   0   0   1   0   0]
     [  0   0   0   0   0   0   0]]
    0.6061224489795919
    0.6162194223774473
    0.6061224489795919
    /usr/local/lib/python3.10/dist-packages/sklearn/metrics/_classification.py:1344: UndefinedMetricWarning: Recall and F-sc
      _warn_prf(average, modifier, msg_start, len(result))
    /usr/local/lib/python3.10/dist-packages/sklearn/metrics/_classification.py:1344: UndefinedMetricWarning: Recall and F-sc
      _warn_prf(average, modifier, msg_start, len(result))
    /usr/local/lib/python3.10/dist-packages/sklearn/metrics/_classification.py:1344: UndefinedMetricWarning: Recall and F-sc
      _warn_prf(average, modifier, msg_start, len(result))
    /usr/local/lib/python3.10/dist-packages/sklearn/metrics/_classification.py:1344: UndefinedMetricWarning: Recall is ill-d
      _warn_prf(average, modifier, msg_start, len(result))
```

```
# 9. Use cross validation to tune the hyperparameters of the tree
parameters = {'max_depth':range(3,20)}
clf = GridSearchCV(tree.DecisionTreeClassifier(), parameters, n_jobs=4)
clf.fit(X=X, y=y)
tree_model = clf.best_estimator_
print (clf.best_score_, clf.best_params_)
```

```
    0.515312793145859 {'max_depth': 3}
```

```
# 10. plot_decision_trees obtained
tree.plot_tree(clf_id3)
plt.show()
tree.plot_tree(clf_cart)
plt.show()
```



```
# 11. Plot the confusion matrix for the classified model
confusion_matrix(y_test, y_pred_id3)
confusion_matrix(y_test, y_pred_cart)
```

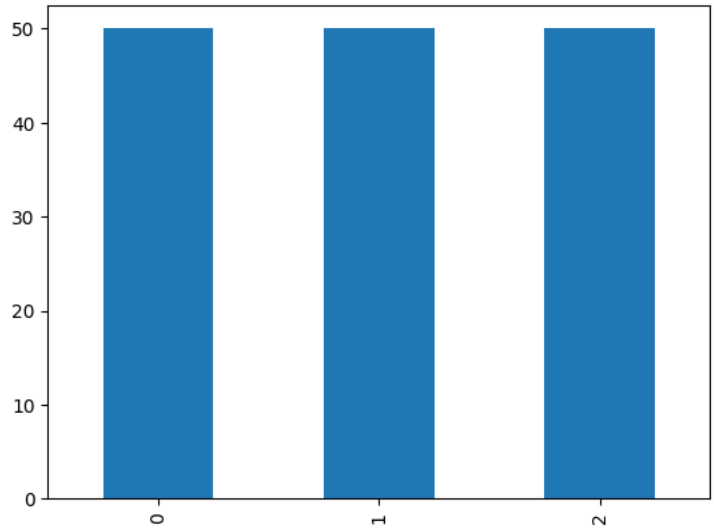
```
array([[270,  80,  52,  14,  14,   2,   0],
       [ 74, 189,  11,   3,  14,   0,   0],
       [ 53,   8, 112,  17,   2,   0,   0],
       [  9,   0,  10,  15,   0,   0,   1],
       [  7,   9,   0,   1,   8,   0,   0],
       [  2,   2,   0,   0,   1,   0,   0],
       [  0,   0,   0,   0,   0,   0,   0]])
```

```
# Confusion Matrix: This is a table that describes the performance of a classification model. Each row represents the instar

# Accuracy: This is the ratio of the number of correct predictions to the total number of predictions. It's calculated as fo

# Accuracy =  Number of Correct Predictions / Total Number of Predictions

# An accuracy of 0.607 means that your model correctly predicted the wine quality approximately 60.7% of the time.

# Precision: This is the ratio of correctly predicted positive observations to the total predicted positives. It's calculate

# Precision = True Positives / (True Positives + False Positives)

# A precision score of 0.6149 means that when your model predicted a wine quality, it was correct approximately 61.49% of th

# Recall (Sensitivity): This is the ratio of correctly predicted positive observations to all the observations in the actual

# Recall = True Positives/ (True Positives + False Negatives)

# A recall score of 0.607 means that your model correctly identified approximately 60.7% of all actual wine qualities.
```

```
#conclusion

# The model handles missing values, visualizes the distribution of wine quality,
# and maps quality to numerical values. It then constructs decision trees,
# evaluates their performance using K-Fold Cross Validation, and prints out various
# metrics such as the confusion matrix, accuracy, precision, and recall. The model also
# tunes the hyperparameters of the tree using GridSearchCV. The accuracy scores obtained are
# moderate, indicating that the model's predictions are correct around 60% of the time.
# This suggests that while the model has learned to some extent from the dataset, there is still
# room for improvement. Possible steps for enhancement could include further hyperparameter tuning,
# feature engineering, or trying more complex models.
```

```
import pandas as pd
import matplotlib.pyplot as plt
from sklearn import tree
from sklearn.model_selection import train_test_split, cross_val_score, GridSearchCV
from sklearn.metrics import classification_report, confusion_matrix, accuracy_score, precision_score, recall_score
from sklearn.datasets import load_iris


# Load the dataset
iris = load_iris()
df = pd.DataFrame(data=iris.data, columns=iris.feature_names)
df['target'] = iris.target


# 1. Check for null values
print(df.isnull().sum())

# 2. Replace null values if any
df.fillna(df.mean(), inplace=True)

# 3. Remove records with many null values if any
df = df.dropna(thresh=len(df.columns)-2)

# 4. Draw the stat of number of each target
df['target'].value_counts().plot(kind='bar')
plt.show()
```

```
sepal length (cm)    0
sepal width (cm)     0
petal length (cm)    0
petal width (cm)     0
target               0
dtype: int64
```




```
# 5. Map the target to numbers
target_mapping = {target: i for i, target in enumerate(df['target'].unique())}
df['target'] = df['target'].map(target_mapping)

# 6. Construct the DT using ID3 and CART
X = df.drop('target', axis=1)
y = df['target']
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)

# ID3
clf_id3 = tree.DecisionTreeClassifier(criterion='entropy')
clf_id3 = clf_id3.fit(X_train, y_train)


# CART
clf_cart = tree.DecisionTreeClassifier(criterion='gini')
clf_cart = clf_cart.fit(X_train, y_train)

# 7. Measure the accuracy both the DT using K-FOLD Cross validation
scores_id3 = cross_val_score(clf_id3, X, y, cv=5)
scores_cart = cross_val_score(clf_cart, X, y, cv=5)


# 8. Print classification report, its confusion matrix, the accuracy, precision and recall score
y_pred_id3 = clf_id3.predict(X_test)
y_pred_cart = clf_cart.predict(X_test)
```

```
print(classification_report(y_test, y_pred_id3))
print(confusion_matrix(y_test, y_pred_id3))
print(accuracy_score(y_test, y_pred_id3))
print(precision_score(y_test, y_pred_id3, average='weighted'))
print(recall_score(y_test, y_pred_id3, average='weighted'))
```

```
                  precision    recall  f1-score   support

               0       1.00      1.00      1.00        10
               1       1.00      1.00      1.00         9
               2       1.00      1.00      1.00        11

        accuracy                           1.00        30
       macro avg       1.00      1.00      1.00        30
    weighted avg       1.00      1.00      1.00        30

    [[10  0  0]
     [ 0  9  0]
     [ 0  0 11]]
    1.0
    1.0
    1.0
```

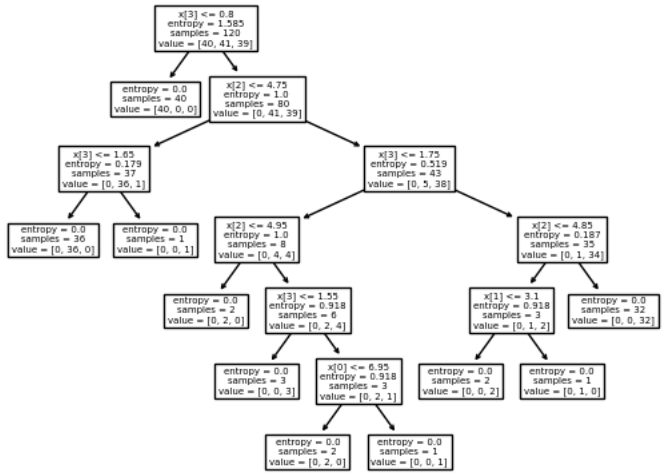
```
print(classification_report(y_test, y_pred_cart))
print(confusion_matrix(y_test, y_pred_cart))
print(accuracy_score(y_test, y_pred_cart))
print(precision_score(y_test, y_pred_cart, average='weighted'))
print(recall_score(y_test, y_pred_cart, average='weighted'))
```

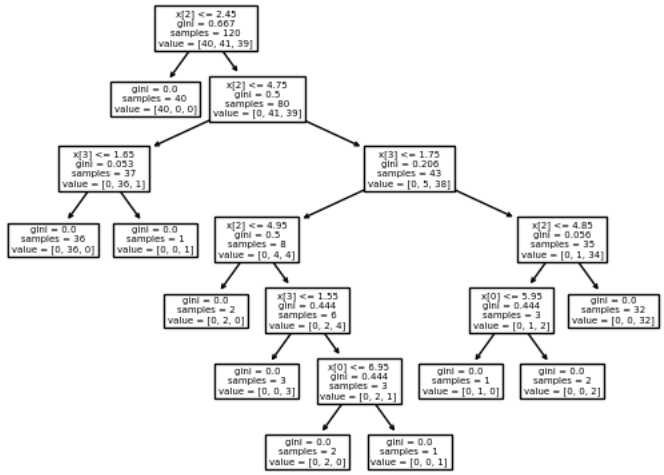
```
                  precision    recall  f1-score   support

               0       1.00      1.00      1.00        10
               1       1.00      1.00      1.00         9
               2       1.00      1.00      1.00        11

        accuracy                           1.00        30
       macro avg       1.00      1.00      1.00        30
    weighted avg       1.00      1.00      1.00        30

    [[10  0  0]
     [ 0  9  0]
     [ 0  0 11]]
    1.0
    1.0
    1.0
```

```
# 9. Use cross validation to tune the hyperparameters of the tree
parameters = {'max_depth':range(3,20)}
clf = GridSearchCV(tree.DecisionTreeClassifier(), parameters, n_jobs=4)
clf.fit(X=X, y=y)
tree_model = clf.best_estimator_
print (clf.best_score_, clf.best_params_)
```

```
    0.9666666666666668 {'max_depth': 5}
```

```
# 10. plot_decision_trees obtained
tree.plot_tree(clf_id3)
plt.show()
tree.plot_tree(clf_cart)
plt.show()
```






```
# 11. Plot the confusion matrix for the classified model
confusion_matrix(y_test, y_pred_id3)
confusion_matrix(y_test, y_pred_cart)
```

```
    array([[10,  0,  0],
           [ 0,  9,  0],
           [ 0,  0, 11]])
```