# Report - Laboration 2

MACHINELEARNING ON THE MNIST HAND DRAWN DIGITS IMAGE DATASET

## Task 1 — Balance the data.

```
digits_Df_twos=digits_Df.loc[digits_Df['target']==2,:]
digits_Df_twos=digits_Df_twos.iloc[0:170]
X_digits_Df_twos=digits_Df_twos.iloc[:,0:-1]
digits_Df_threes=digits_Df.loc[digits_Df['target']==3,:]
digits_Df_threes=digits_Df_threes.iloc[0:170]
X_digits_Df_threes=digits_Df_threes.iloc[:,0:-1]
X_digits_Df_twos
digits_Df_twos
digits_Df_threes
```

## Task 2 & 3 — Calculate the accuracy, plot the decision tree with different max_depth and explain how it works.

```
digits_Df_23 = digits_Df_twos.append(digits_Df_threes)
digits_Df_23
```

```
/var/folders/jq/4bwq_gcj13q52nl2rthwnssw0000gn/T/ipykernel_34831/893467902.py:1: FutureWarning: The frame.append me
thod is deprecated and will be removed from pandas in a future version. Use pandas.concat instead.
  digits_Df_23 = digits_Df_twos.append(digits_Df_threes)
```

| | pixel_0_0 | pixel_0_1 | pixel_0_2 | pixel_0_3 | pixel_0_4 | pixel_0_5 | pixel_0_6 | pixel_0_7 | pixel_1_0 | pixel_1_1 | ... | pixel_6_7 | pixel_7_0 | pixel_7_1 | pixel_7_2 | pix |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| **2** | 0.0 | 0.0 | 0.0 | 4.0 | 15.0 | 12.0 | 0.0 | 0.0 | 0.0 | 0.0 | ... | 0.0 | 0.0 | 0.0 | 0.0 | |
| **12** | 0.0 | 0.0 | 5.0 | 12.0 | 1.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | ... | 2.0 | 0.0 | 0.0 | 3.0 | |
| **22** | 0.0 | 0.0 | 8.0 | 16.0 | 5.0 | 0.0 | 0.0 | 0.0 | 0.0 | 1.0 | ... | 0.0 | 0.0 | 0.0 | 7.0 | |
| **50** | 0.0 | 0.0 | 0.0 | 5.0 | 14.0 | 12.0 | 2.0 | 0.0 | 0.0 | 0.0 | ... | 0.0 | 0.0 | 0.0 | 0.0 | |
| **51** | 0.0 | 0.0 | 0.0 | 3.0 | 15.0 | 10.0 | 1.0 | 0.0 | 0.0 | 0.0 | ... | 0.0 | 0.0 | 0.0 | 0.0 | |
| **...** | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | |
| **1624** | 0.0 | 1.0 | 11.0 | 16.0 | 16.0 | 12.0 | 0.0 | 0.0 | 0.0 | 8.0 | ... | 0.0 | 0.0 | 0.0 | 14.0 | |
| **1630** | 0.0 | 0.0 | 6.0 | 16.0 | 15.0 | 5.0 | 0.0 | 0.0 | 0.0 | 0.0 | ... | 0.0 | 0.0 | 0.0 | 6.0 | |

```
X_digits_Df_23 = digits_Df_23.iloc[:,0:-1]
X_digits_Df_23
```

| | pixel_0_0 | pixel_0_1 | pixel_0_2 | pixel_0_3 | pixel_0_4 | pixel_0_5 | pixel_0_6 | pixel_0_7 | pixel_1_0 | pixel_1_1 | ... | pixel_6_6 | pixel_6_7 | pixel_7_0 | pixel_7_1 | pixel |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| **2** | 0.0 | 0.0 | 0.0 | 4.0 | 15.0 | 12.0 | 0.0 | 0.0 | 0.0 | 0.0 | ... | 5.0 | 0.0 | 0.0 | 0.0 | |
| **12** | 0.0 | 0.0 | 5.0 | 12.0 | 1.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | ... | 8.0 | 2.0 | 0.0 | 0.0 | |
| **22** | 0.0 | 0.0 | 8.0 | 16.0 | 5.0 | 0.0 | 0.0 | 0.0 | 0.0 | 1.0 | ... | 3.0 | 0.0 | 0.0 | 0.0 | |
| **50** | 0.0 | 0.0 | 0.0 | 5.0 | 14.0 | 12.0 | 2.0 | 0.0 | 0.0 | 0.0 | ... | 0.0 | 0.0 | 0.0 | 0.0 | |
| **51** | 0.0 | 0.0 | 0.0 | 3.0 | 15.0 | 10.0 | 1.0 | 0.0 | 0.0 | 0.0 | ... | 3.0 | 0.0 | 0.0 | 0.0 | |
| **...** | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | |
| **1624** | 0.0 | 1.0 | 11.0 | 16.0 | 16.0 | 12.0 | 0.0 | 0.0 | 0.0 | 8.0 | ... | 0.0 | 0.0 | 0.0 | 0.0 | |
| **1630** | 0.0 | 0.0 | 6.0 | 16.0 | 15.0 | 5.0 | 0.0 | 0.0 | 0.0 | 0.0 | ... | 0.0 | 0.0 | 0.0 | 0.0 | |
| **1632** | 0.0 | 0.0 | 6.0 | 15.0 | 15.0 | 4.0 | 0.0 | 0.0 | 0.0 | 6.0 | ... | 0.0 | 0.0 | 0.0 | 0.0 | |
| **1639** | 0.0 | 0.0 | 7.0 | 15.0 | 16.0 | 10.0 | 0.0 | 0.0 | 0.0 | 0.0 | ... | 1.0 | 0.0 | 0.0 | 0.0 | |
| **1644** | 0.0 | 0.0 | 8.0 | 15.0 | 16.0 | 6.0 | 0.0 | 0.0 | 0.0 | 0.0 | ... | 1.0 | 0.0 | 0.0 | 0.0 | |

340 rows × 64 columns

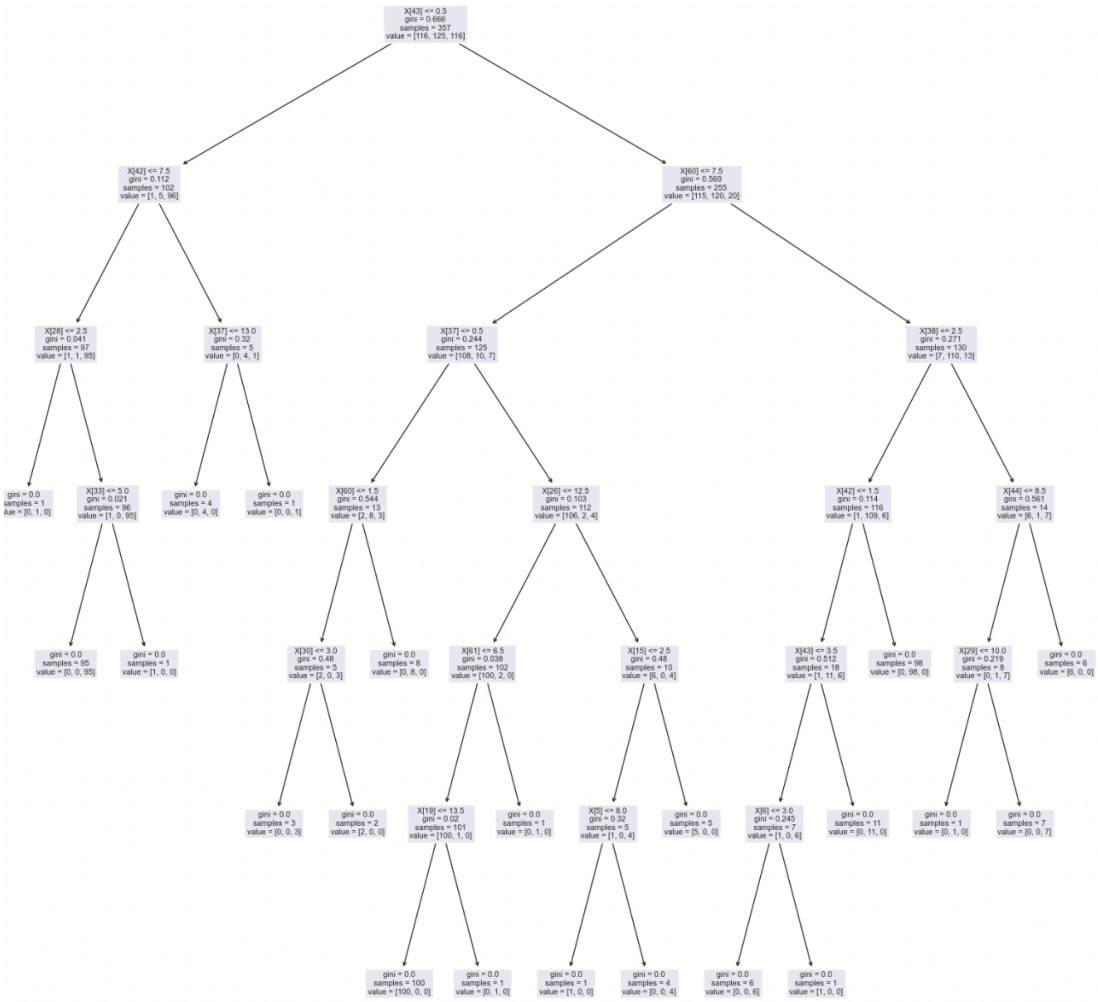
```
Y_digits_Df_23 = digits_Df_23.iloc[:,64]
Y_digits_Df_23
```

```
2        2
12       2
22       2
50       2
51       2
        ..
1624     3
1630     3
1632     3
1639     3
1644     3
Name: target, Length: 340, dtype: int64
```

```
from sklearn.model_selection import train_test_split
x_train, x_test, y_train, y_test = train_test_split(X_digits_Df_23, Y_digits_Df_23,
                                                    test_size=0.30, random_state=40)
```

```
from sklearn.tree import DecisionTreeClassifier
model = DecisionTreeClassifier(max_depth = None, random_state = 40)
model.fit(x_train, y_train)
#model.score(x_train, y_train)
model.score(x_test, y_test)
```

```
0.9607843137254902
```

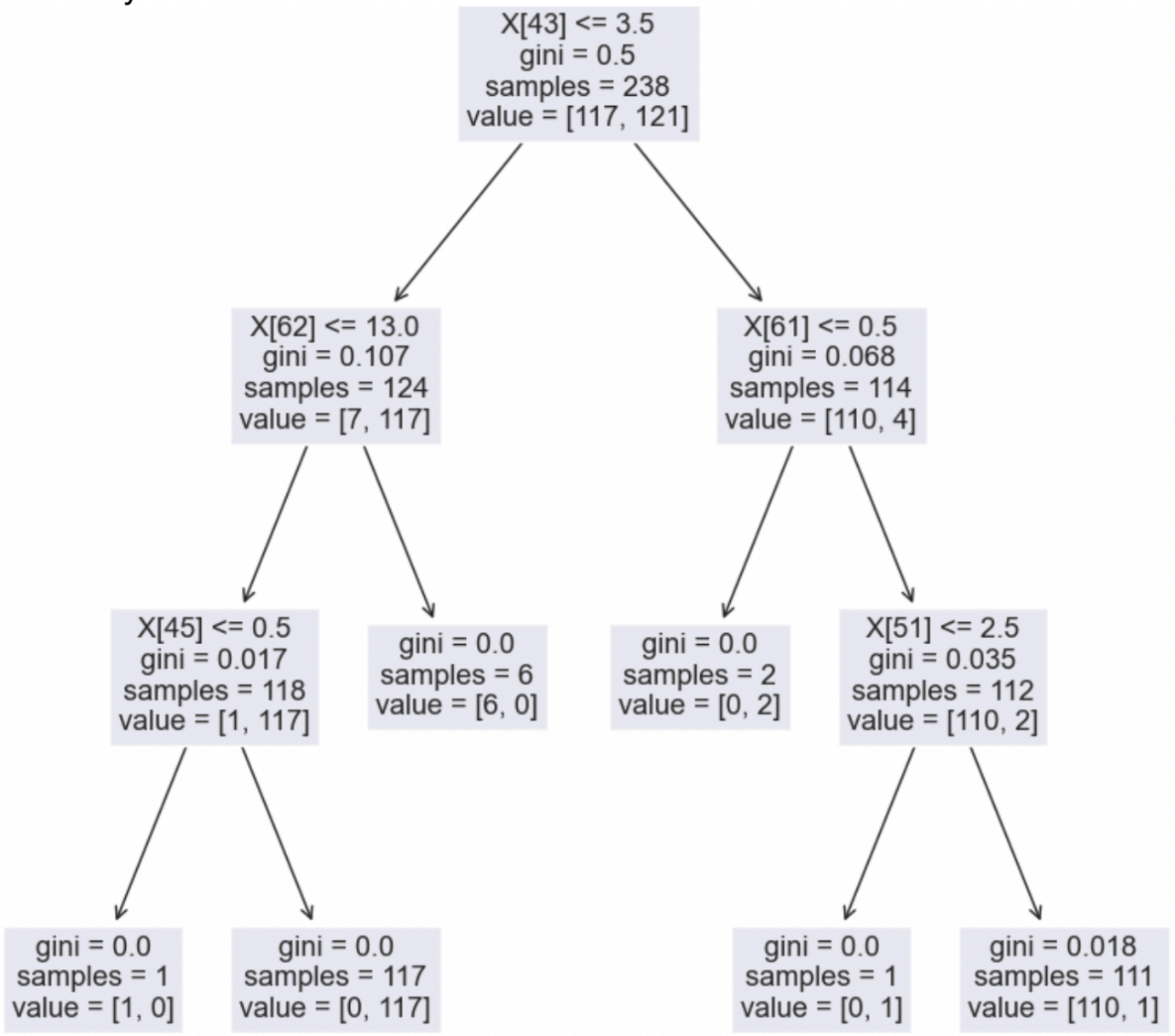


The maximum depth of the decision tree is not specified here, so the decision tree we get is shown in the figure above, and the calculated accuracy is about 0.96.

```
from sklearn.tree import DecisionTreeClassifier
model = DecisionTreeClassifier(max_depth = 3, random_state = 40)
model.fit(x_train, y_train)
#model.score(x_train, y_train)
model.score(x_test, y_test)
```

```
0.9215686274509803
```

If we specify a maximum depth of 3, we will get a decision tree as shown in the figure below, and find that the accuracy has decreased, now the accuracy is about 0.92.




Taking this decision tree as an example, the working process of the decision tree is as follows:
First judge whether the value of the 43rd column in the X part is less than or equal to 3.5, if it is true, then judge whether the value of the 62nd column is

less than or equal to 13, but if it is false, then judge whether the value of the 61st column is less than or equal to 0.5 ...
By analogy, judge whether the situation in the node is true, if it is true, continue to move down to the left, if it is false, go to the bottom right, until 0 appears in the value, and end.

## Task 4 — 7/8/9

```
digits_Df_sevens=digits_Df.loc[digits_Df['target']==7,:]
digits_Df_sevens=digits_Df_sevens.iloc[0:170]
digits_Df_eights=digits_Df.loc[digits_Df['target']==8,:]
digits_Df_eights=digits_Df_eights.iloc[0:170]
digits_Df_nines=digits_Df.loc[digits_Df['target']==9,:]
digits_Df_nines=digits_Df_nines.iloc[0:170]
digits_Df_nines
```

```
digits_Df_78 = digits_Df_sevens.append(digits_Df_eights)
digits_Df_789 = digits_Df_78.append(digits_Df_nines)
digits_Df_789
```

```
X_digits_Df_789 = digits_Df_789.iloc[:,0:-1]
X_digits_Df_789
```

| | pixel_0_0 | pixel_0_1 | pixel_0_2 | pixel_0_3 | pixel_0_4 | pixel_0_5 | pixel_0_6 | pixel_0_7 | pixel_1_0 | pixel_1_1 | ... | pixel_6_6 | pixel_6_7 | pixel_7_0 | pixel_7_1 | pixe |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 7 | 0.0 | 0.0 | 7.0 | 8.0 | 13.0 | 16.0 | 15.0 | 1.0 | 0.0 | 0.0 | ... | 0.0 | 0.0 | 0.0 | 0.0 | |
| 17 | 0.0 | 0.0 | 1.0 | 8.0 | 15.0 | 10.0 | 0.0 | 0.0 | 0.0 | 3.0 | ... | 0.0 | 0.0 | 0.0 | 0.0 | |
| 27 | 0.0 | 0.0 | 0.0 | 8.0 | 14.0 | 14.0 | 2.0 | 0.0 | 0.0 | 0.0 | ... | 0.0 | 0.0 | 0.0 | 0.0 | |
| 43 | 0.0 | 0.0 | 0.0 | 9.0 | 15.0 | 12.0 | 0.0 | 0.0 | 0.0 | 0.0 | ... | 0.0 | 0.0 | 0.0 | 0.0 | |
| 44 | 0.0 | 0.0 | 9.0 | 16.0 | 16.0 | 16.0 | 5.0 | 0.0 | 0.0 | 1.0 | ... | 0.0 | 0.0 | 0.0 | 0.0 | |
| ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | |
| 1662 | 0.0 | 0.0 | 7.0 | 15.0 | 16.0 | 12.0 | 0.0 | 0.0 | 0.0 | 12.0 | ... | 0.0 | 0.0 | 0.0 | 0.0 | |
| 1665 | 0.0 | 0.0 | 1.0 | 10.0 | 13.0 | 12.0 | 5.0 | 0.0 | 0.0 | 0.0 | ... | 0.0 | 0.0 | 0.0 | 0.0 | |
| 1676 | 0.0 | 1.0 | 8.0 | 13.0 | 15.0 | 5.0 | 0.0 | 0.0 | 0.0 | 8.0 | ... | 12.0 | 0.0 | 0.0 | 0.0 | |
| 1686 | 0.0 | 0.0 | 8.0 | 14.0 | 12.0 | 3.0 | 0.0 | 0.0 | 0.0 | 6.0 | ... | 3.0 | 0.0 | 0.0 | 0.0 | |

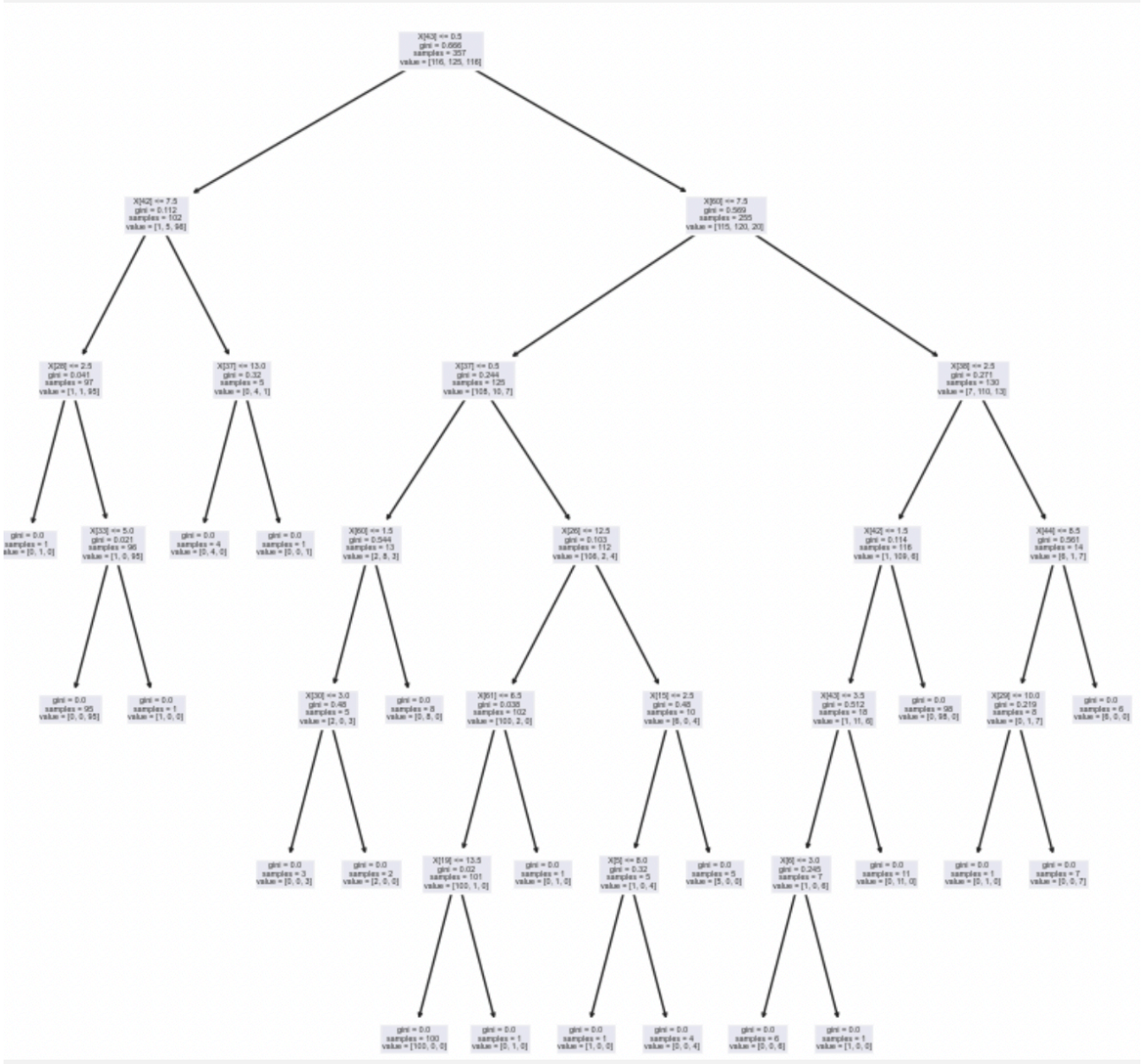
```
Y_digits_Df_789 = digits_Df_789.iloc[:,64]
Y_digits_Df_789
```

```
7       7
17      7
27      7
43      7
44      7
       ..
1662    9
1665    9
1676    9
1686    9
1696    9
Name: target, Length: 510, dtype: int64
```

```
from sklearn.model_selection import train_test_split
x_train, x_test, y_train, y_test = train_test_split(X_digits_Df_789, Y_digits_Df_789,
                                                    test_size=0.30, random_state=40)
```

```
from sklearn.tree import DecisionTreeClassifier
model = DecisionTreeClassifier(max_depth = None, random_state = 40)
model.fit(x_train, y_train)
#model.score(x_train, y_train)
model.score(x_test, y_test)
```

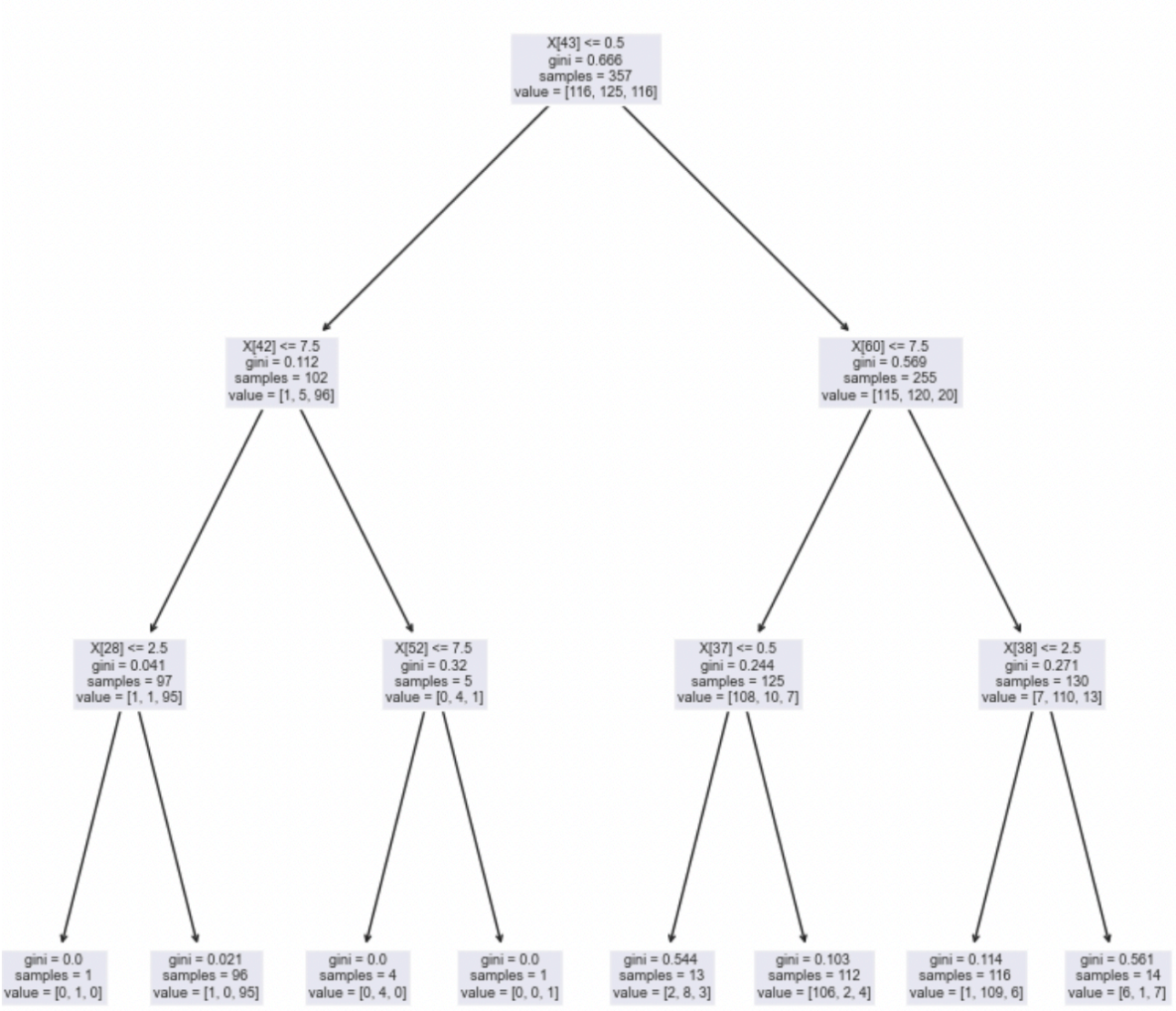
```
0.9607843137254902
```



The maximum depth of the decision tree is not specified here, so the decision tree we get is shown in the figure above, and the calculated accuracy is about 0.96.

```
from sklearn.tree import DecisionTreeClassifier
model = DecisionTreeClassifier(max_depth = 3, random_state = 40)
model.fit(x_train, y_train)
#model.score(x_train, y_train)
model.score(x_test, y_test)
```

```
0.8758169934640523
```




If we specify a maximum depth of 3, we will get a decision tree as shown in the figure above, and find that the accuracy has decreased, now the accuracy is about 0.88.

## Task 5 — Print the arrays

Output the predicted value(y_pred) and the true value(y_test). Comparing the two arrays, it can be found that some of the predicted values do not match the true value. For example, the last one has a predicted value of 7 but a true value of 9.

```
y_pred = model.predict(x_test)
y_pred
```

```
array([9, 9, 9, 9, 9, 9, 7, 7, 7, 7, 7, 9, 8, 8, 8, 9, 7, 7, 9, 8, 7, 7,
       7, 7, 8, 9, 7, 9, 8, 7, 9, 8, 7, 9, 8, 9, 9, 9, 7, 8, 7, 9, 7, 9,
       7, 8, 8, 9, 7, 9, 8, 7, 7, 9, 9, 9, 7, 9, 9, 8, 8, 9, 8, 7, 8, 9,
       7, 9, 9, 8, 7, 8, 8, 9, 8, 9, 9, 9, 9, 7, 8, 7, 7, 7, 8, 7, 8, 9,
       8, 9, 9, 7, 7, 9, 8, 7, 9, 9, 7, 8, 9, 7, 9, 9, 8, 8, 7, 9, 8, 8,
       8, 8, 8, 8, 7, 9, 8, 8, 7, 9, 8, 7, 8, 9, 8, 8, 9, 7, 8, 7, 7, 8,
       7, 8, 9, 9, 7, 7, 9, 7, 7, 8, 9, 9, 9, 9, 7, 7, 8, 9, 9, 9, 7])
```

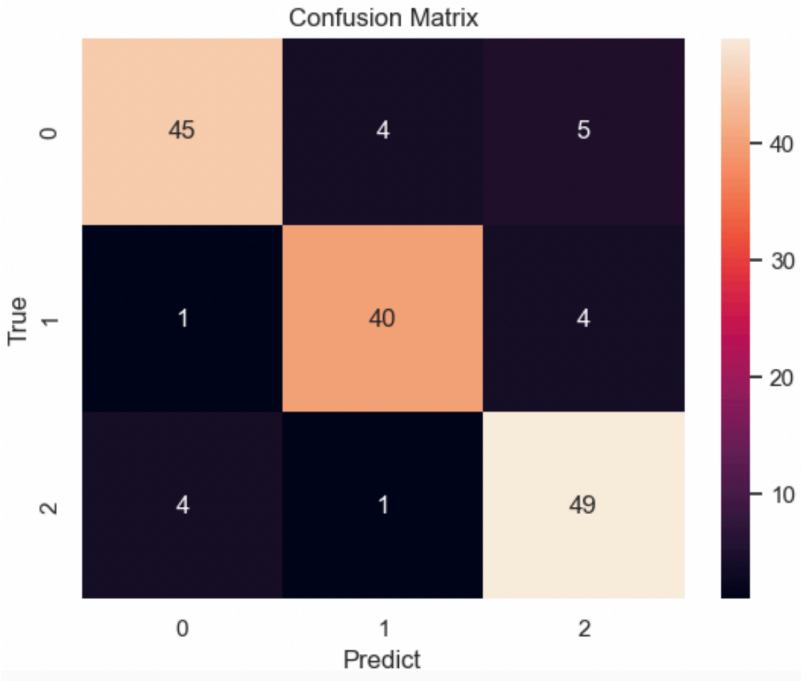
```
y_test = y_test.values
y_test
```

```
array([9, 9, 9, 9, 9, 9, 7, 7, 7, 7, 7, 9, 8, 7, 8, 7, 9, 7, 9, 8, 7, 9,
       7, 7, 9, 9, 7, 8, 8, 7, 9, 7, 7, 9, 8, 9, 9, 9, 7, 8, 7, 9, 7, 9,
       7, 8, 7, 7, 7, 9, 8, 7, 7, 9, 9, 7, 7, 9, 9, 8, 8, 9, 8, 7, 8, 9,
       7, 9, 7, 7, 7, 8, 8, 9, 8, 7, 9, 9, 9, 7, 8, 7, 7, 7, 8, 7, 8, 9,
       8, 9, 8, 7, 7, 8, 8, 8, 9, 9, 7, 8, 9, 7, 9, 9, 8, 8, 9, 9, 8, 8,
       8, 8, 8, 8, 7, 9, 8, 8, 7, 8, 8, 7, 8, 9, 8, 8, 9, 7, 8, 7, 7, 8,
       7, 8, 9, 9, 7, 7, 9, 7, 7, 8, 9, 9, 9, 9, 7, 7, 8, 9, 9, 9, 9])
```

## Task 6 — Plot the confusion_matrix.

```
import seaborn as sns
from sklearn.metrics import confusion_matrix
```

```
sns.set()
fig,ax = plt.subplots()
y_true = y_test
y_pred = y_pred
C = confusion_matrix(y_true, y_pred)
sns.heatmap(C,annot=True,ax=ax)
ax.set_title('Confusion Matrix')
ax.set_xlabel('Predict')
ax.set_ylabel('True')
```



## Task 7 — Use LogisticRegression to calculate the accuracy.

```
from sklearn.linear_model import LogisticRegression
modelLR=LogisticRegression()
modelLR.fit(x_train,y_train)
modelLR.score(x_test,y_test)
```

```
/Users/zaltman/opt/anaconda3/lib/python3.9/site-package
bfgs failed to converge (status=1):
STOP: TOTAL NO. of ITERATIONS REACHED LIMIT.

Increase the number of iterations (max_iter) or scale t
    https://scikit-learn.org/stable/modules/preprocessi
Please also refer to the documentation for alternative
    https://scikit-learn.org/stable/modules/linear_mode
  n_iter_i = _check_optimize_result(
```

```
0.9869281045751634
```

Clearly, the accuracy of the LogisticRegression algorithm is higher than that of the decision tree, and the accuracy is about 0.99.