Microspecialization Project -> Bank Credit Approval

Classification Algorithms

```
import pandas as pd
import numpy as np
import seaborn as sns
from sklearn.model_selection import train_test_split
from sklearn.tree import DecisionTreeClassifier
from sklearn.metrics import accuracy_score
from sklearn import tree
import matplotlib.pyplot as plt
import warnings
warnings.filterwarnings('ignore')
%matplotlib inline
```

Loading the Dataset

```
df = pd.read_excel('data.xlsx')
df.head()
```

```
   Variable_1  Variable_2  Variable_3  Variable_4  Variable_5
Variable_6  \
0           1       22.08       11.46           2           4
4
1           0       22.67        7.00           2           8
4
2           0       29.58        1.75           1           4
4
3           0       21.67       11.50           1           5
3
4           1       20.17        8.17           2           6
4

   Variable_7  Variable_8  Variable_9  Variable_10  Variable_11
Variable_12  \
0       1.585           0           0            0            1
2
1       0.165           0           0            0            0
2
2       1.250           0           0            0            1
2
3       0.000           1           1           11            1
2
4       1.960           1           1           14            0
```

```
2
```

```
   Variable_13  Variable_14  Target
0          100         1213       0
1          160            1       0
2          280            1       0
3            0            1       1
4           60          159       1
```

Preprocessing the Dataset

```
# Checking for Null Values
df.isnull().sum()
```

```
Variable_1     0
Variable_2     0
Variable_3     0
Variable_4     0
Variable_5     0
Variable_6     0
Variable_7     0
Variable_8     0
Variable_9     0
Variable_10    0
Variable_11    0
Variable_12    0
Variable_13    0
Variable_14    0
Target         0
dtype: int64
```

```
df.describe()
```

```
       Variable_1  Variable_2  Variable_3  Variable_4  Variable_5
Variable_6  \
count  690.000000  690.000000  690.000000  690.000000  690.000000
690.000000
mean     0.678261   31.568203    4.758725    1.766667    7.372464
4.692754
std      0.467482   11.853273    4.978163    0.430063    3.683265
1.992316
min      0.000000   13.750000    0.000000    1.000000    1.000000
1.000000
25%      0.000000   22.670000    1.000000    2.000000    4.000000
4.000000
50%      1.000000   28.625000    2.750000    2.000000    8.000000
4.000000
75%      1.000000   37.707500    7.207500    2.000000   10.000000
5.000000
max      1.000000   80.250000   28.000000    3.000000   14.000000
```

```
9.000000

       Variable_7  Variable_8  Variable_9  Variable_10  Variable_11  \
count  690.000000  690.000000  690.000000    690.00000   690.000000
mean     2.223406    0.523188    0.427536      2.40000     0.457971
std      3.346513    0.499824    0.495080      4.86294     0.498592
min      0.000000    0.000000    0.000000      0.00000     0.000000
25%      0.165000    0.000000    0.000000      0.00000     0.000000
50%      1.000000    1.000000    0.000000      0.00000     0.000000
75%      2.625000    1.000000    1.000000      3.00000     1.000000
max     28.500000    1.000000    1.000000     67.00000     1.000000

       Variable_12  Variable_13    Variable_14      Target
count   690.000000   690.000000     690.000000  690.000000
mean      1.928986   184.014493    1018.385507    0.444928
std       0.298813   172.159274    5210.102598    0.497318
min       1.000000     0.000000       1.000000    0.000000
25%       2.000000    80.000000       1.000000    0.000000
50%       2.000000   160.000000       6.000000    0.000000
75%       2.000000   272.000000     396.500000    1.000000
max       3.000000  2000.000000  100001.000000    1.000000
```

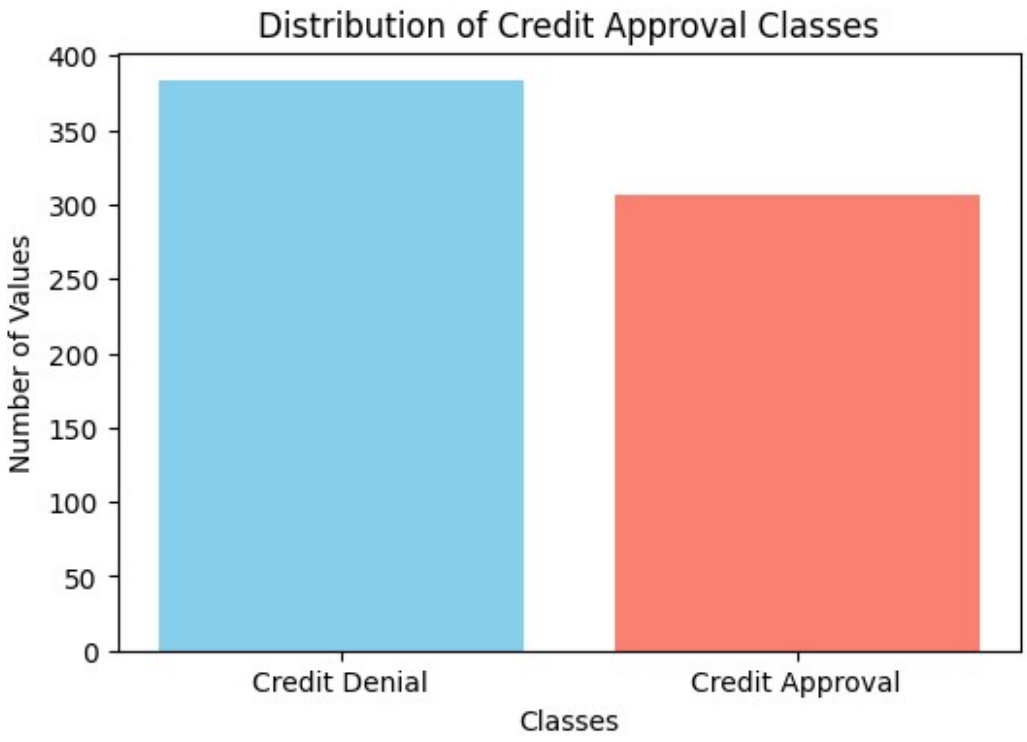
Exploratory Data Analysis

```
# Count the occurrences of each class
class_counts = df['Target'].value_counts()

# Create a bar plot
plt.figure(figsize=(6, 4))
plt.bar(class_counts.index, class_counts.values, color=['skyblue',
'salmon'])

# Add labels and title
plt.xticks([0, 1], ['Credit Denial', 'Credit Approval'])  # Setting
the x-ticks to show class labels
plt.xlabel('Classes')
plt.ylabel('Number of Values')
plt.title('Distribution of Credit Approval Classes')

# Show plot
plt.show()
```

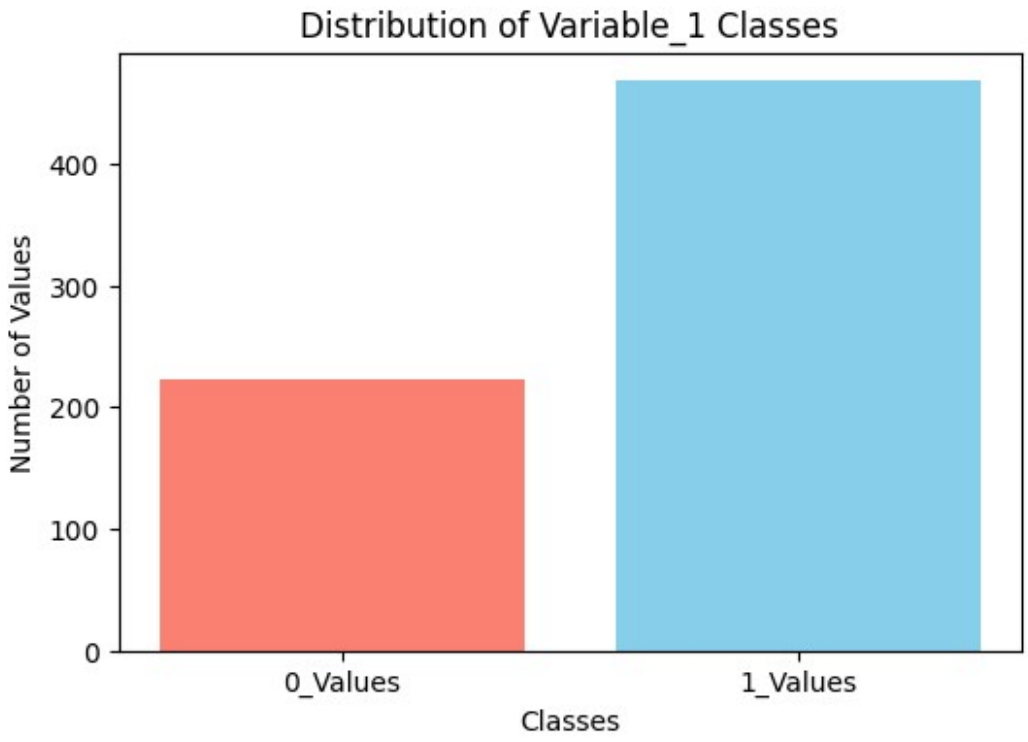



```
# Count the occurrences of each class of Variable 1
class_counts = df['Variable_1'].value_counts()

# Create a bar plot
plt.figure(figsize=(6, 4))
plt.bar(class_counts.index, class_counts.values, color=['skyblue',
'salmon'])

# Add labels and title
plt.xticks([0, 1], ['0_Values', '1_Values'])  # Setting the x-ticks to
show class labels
plt.xlabel('Classes')
plt.ylabel('Number of Values')
plt.title('Distribution of Variable_1 Classes')

# Show plot
plt.show()
```

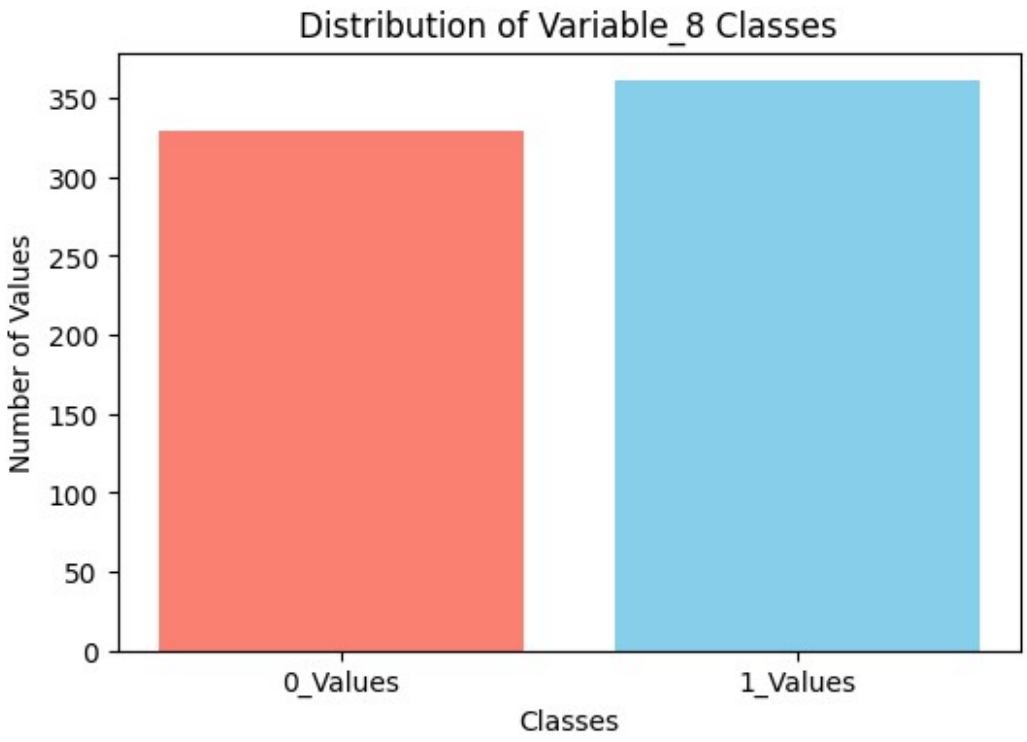



```
# Count the occurrences of each class of Variable 8
class_counts = df['Variable_8'].value_counts()

# Create a bar plot
plt.figure(figsize=(6, 4))
plt.bar(class_counts.index, class_counts.values, color=['skyblue',
'salmon'])

# Add labels and title
plt.xticks([0, 1], ['0_Values', '1_Values'])  # Setting the x-ticks to
show class labels
plt.xlabel('Classes')
plt.ylabel('Number of Values')
plt.title('Distribution of Variable_8 Classes')

# Show plot
plt.show()
```

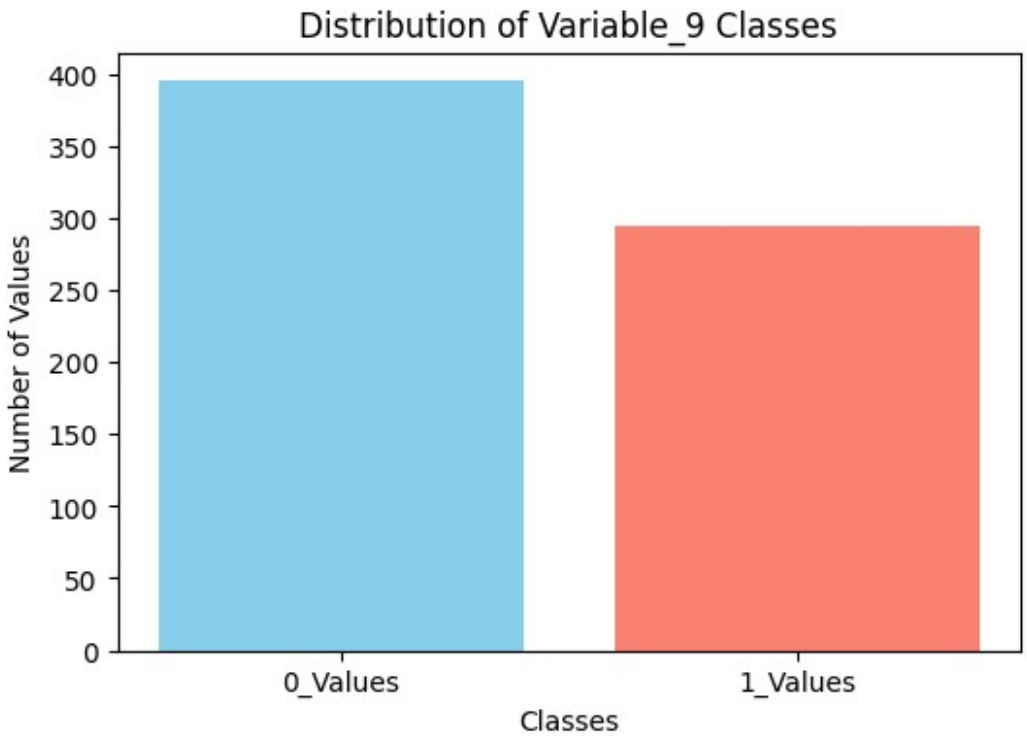



```
# Count the occurrences of each class of Variable 1
class_counts = df['Variable_9'].value_counts()

# Create a bar plot
plt.figure(figsize=(6, 4))
plt.bar(class_counts.index, class_counts.values, color=['skyblue',
'salmon'])

# Add labels and title
plt.xticks([0, 1], ['0_Values', '1_Values'])  # Setting the x-ticks to
show class labels
plt.xlabel('Classes')
plt.ylabel('Number of Values')
plt.title('Distribution of Variable_9 Classes')

# Show plot
plt.show()
```

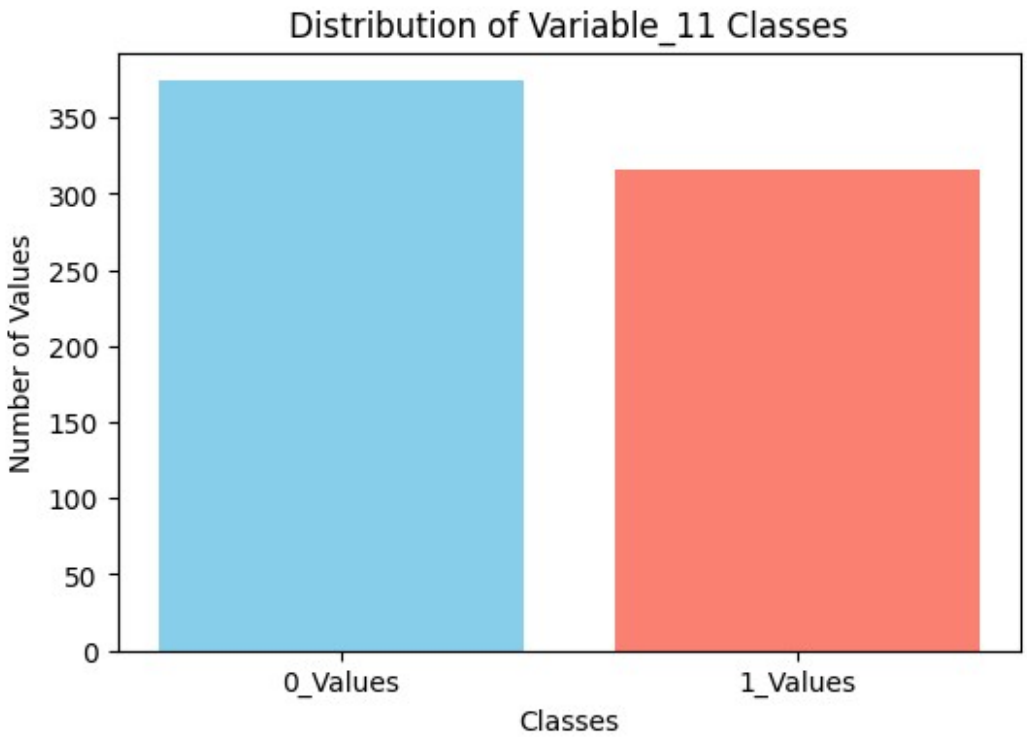



```
# Count the occurrences of each class of Variable 1
class_counts = df['Variable_11'].value_counts()

# Create a bar plot
plt.figure(figsize=(6, 4))
plt.bar(class_counts.index, class_counts.values, color=['skyblue',
'salmon'])

# Add labels and title
plt.xticks([0, 1], ['0_Values', '1_Values'])  # Setting the x-ticks to
show class labels
plt.xlabel('Classes')
plt.ylabel('Number of Values')
plt.title('Distribution of Variable_11 Classes')

# Show plot
plt.show()
```

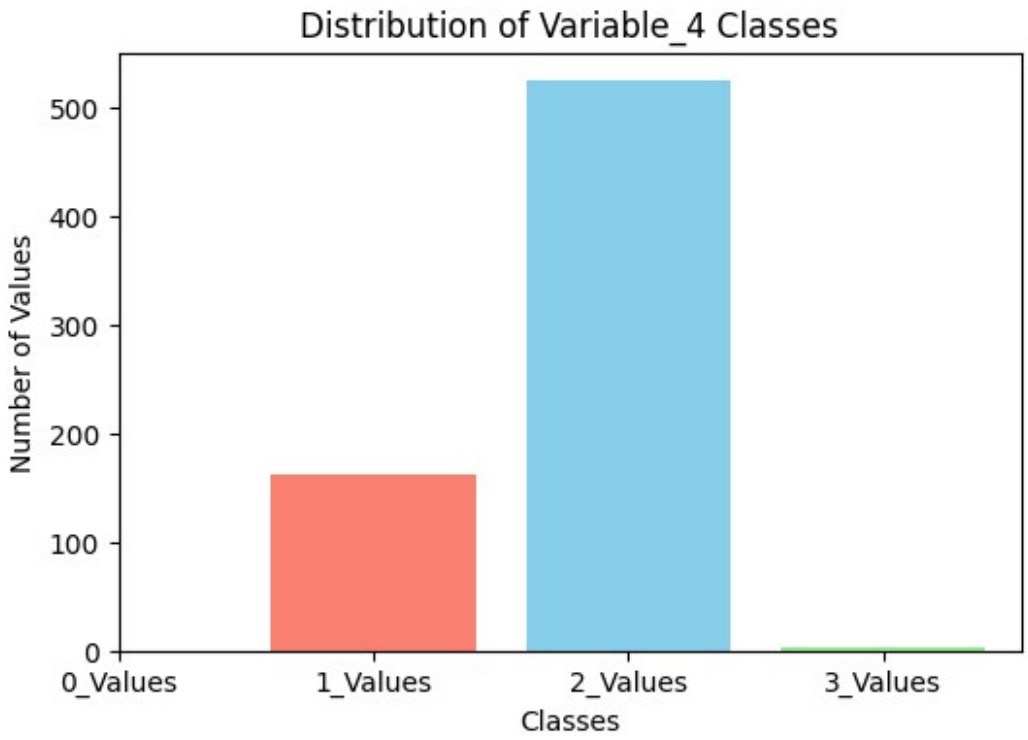



```
# Count the occurrences of each class of Variable 1
class_counts = df['Variable_4'].value_counts()

# Create a bar plot
plt.figure(figsize=(6, 4))
plt.bar(class_counts.index, class_counts.values, color=['skyblue',
'salmon', 'lightgreen', 'orange'])

# Add labels and title
plt.xticks([0, 1, 2, 3], ['0_Values', '1_Values', '2_Values',
'3_Values'])  # Setting the x-ticks to show class labels
plt.xlabel('Classes')
plt.ylabel('Number of Values')
plt.title('Distribution of Variable_4 Classes')

# Show plot
plt.show()
```

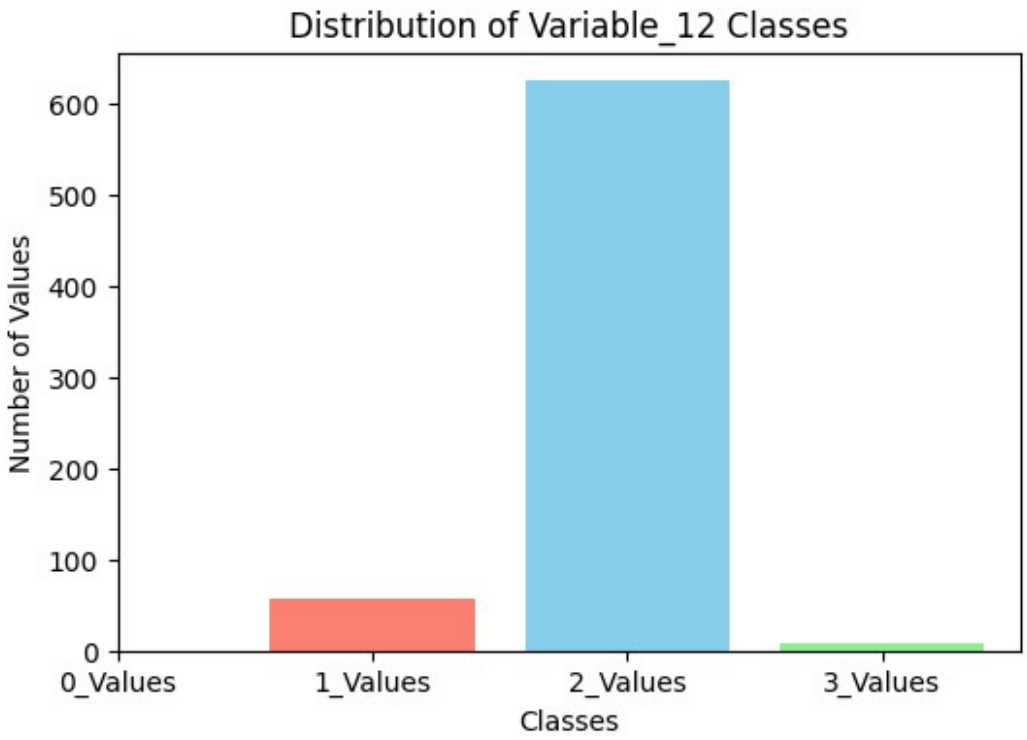



```
# Count the occurrences of each class of Variable 1
class_counts = df['Variable_12'].value_counts()

# Create a bar plot
plt.figure(figsize=(6, 4))
plt.bar(class_counts.index, class_counts.values, color=['skyblue',
'salmon', 'lightgreen', 'orange'])

# Add labels and title
plt.xticks([0, 1, 2, 3], ['0_Values', '1_Values', '2_Values',
'3_Values'])  # Setting the x-ticks to show class labels
plt.xlabel('Classes')
plt.ylabel('Number of Values')
plt.title('Distribution of Variable_12 Classes')

# Show plot
plt.show()
```



Train Test Split

```
df.shape
```

```
(690, 15)
```

```
# Seperating the target value
X = df.values[:, 0:13]
y = df.values[:, 14]

print("Shape of X:", X.shape)
print("Shape of y:", y.shape)
```

```
Shape of X: (690, 13)
Shape of y: (690,)
```

```
# Splitting the dataset into test and train
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size =
0.3, random_state = 100)

# Function to perform training with Entropy
clf_entropy = DecisionTreeClassifier(criterion='entropy',
random_state=100, max_depth=3,min_samples_leaf=5)
clf_entropy.fit(X_train, y_train)
```

```
DecisionTreeClassifier(criterion='entropy', max_depth=3,
min_samples_leaf=5,
                       random_state=100)
```

```
y_pred = clf_entropy.predict(X_test)
print("Predicted values:")
print(y_pred)
```

```
Predicted values:
[0. 1. 0. 0. 0. 1. 1. 0. 1. 0. 0. 1. 1. 0. 0. 0. 0. 1. 1. 0. 1. 0. 1.
0.
 1. 0. 1. 1. 0. 1. 0. 1. 0. 1. 1. 1. 0. 0. 0. 0. 1. 0. 0. 1. 0. 0. 0.
0.
 0. 1. 0. 0. 1. 1. 0. 0. 0. 1. 1. 0. 1. 0. 0. 0. 1. 1. 0. 1. 1. 1. 1.
0.
 1. 0. 0. 1. 0. 0. 0. 0. 1. 1. 0. 1. 1. 1. 0. 1. 0. 1. 1. 1. 1. 0. 0.
1.
 0. 0. 0. 1. 1. 0. 0. 1. 1. 0. 0. 1. 0. 1. 1. 1. 0. 0. 0. 0. 1. 0. 1.
1.
 0. 1. 1. 0. 0. 0. 1. 1. 0. 0. 0. 1. 0. 0. 1. 1. 0. 0. 0. 1. 0. 1. 0.
0.
 0. 0. 0. 1. 1. 0. 0. 0. 1. 1. 1. 0. 1. 1. 1. 1. 0. 0. 1. 0. 1. 0. 0.
0.
 1. 0. 1. 0. 1. 0. 1. 0. 1. 0. 0. 1. 1. 1. 1. 0. 1. 0. 0. 1. 0. 0. 0.
0.
 0. 1. 0. 0. 1. 1. 1. 1. 0. 1. 1. 0. 1. 0. 0.]
```

```
# Checking the Accuracy of the model
print("Accuracy is: ", accuracy_score(y_test, y_pred)*100)
```

```
Accuracy is:  85.99033816425121
```