```python
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import LabelEncoder
from sklearn import tree
import random

path = 'drug.csv'
data = pd.read_csv(path)
print('data')
print(data)
```

```
data
     Age Sex      BP Cholesterol  Na_to_K   Drug
0     23   F    HIGH        HIGH   25.355  drugY
1     47   M     LOW        HIGH   13.093  drugC
2     47   M     LOW        HIGH   10.114  drugC
3     28   F  NORMAL        HIGH      NaN  drugX
4     61   F     LOW        HIGH   18.043  drugY
..   ...  ..     ...         ...      ...    ...
195   56   F     LOW        HIGH   11.567  drugC
196   16   M     LOW        HIGH   12.006  drugC
197   52   M  NORMAL        HIGH    9.894  drugX
198   23   M  NORMAL         NaN   14.020  drugX
199   40   F     LOW      NORMAL   11.349  drugX

[200 rows x 6 columns]
```

```python
missing_values = data.isnull().sum()
print("Missing values in each column in data:")
print(missing_values)
```

```
Missing values in each column in data:
Age            0
Sex            0
BP             2
Cholesterol    2
Na_to_K        1
Drug           0
dtype: int64
```

```python
data = data.dropna()
print('data after removing missing values')
print(data)
```

```
data after removing missing values
     Age Sex      BP Cholesterol  Na_to_K   Drug
0     23   F    HIGH        HIGH   25.355  drugY
1     47   M     LOW        HIGH   13.093  drugC
2     47   M     LOW        HIGH   10.114  drugC
```

```
4      61   F      LOW        HIGH   18.043  drugY
5      22   F   NORMAL        HIGH    8.607  drugX
..    ...  ..      ...         ...      ...    ...
194    46   F     HIGH        HIGH   34.686  drugY
195    56   F      LOW        HIGH   11.567  drugC
196    16   M      LOW        HIGH   12.006  drugC
197    52   M   NORMAL        HIGH    9.894  drugX
199    40   F      LOW      NORMAL   11.349  drugX

[195 rows x 6 columns]
```

```python
def categorize_features(data):
    categorical_features =
data.select_dtypes(include=['object']).columns.tolist()
    numerical_features =
data.select_dtypes(include=[np.number]).columns.tolist()

    return categorical_features, numerical_features


categorical, numerical = categorize_features(data)

print("Categorical Features:", categorical)
```

```
Categorical Features: ['Sex', 'BP', 'Cholesterol', 'Drug']
```

```python
# the features and targets are separated
num_of_cols = data.shape[1]
X = data.iloc[:, 0:num_of_cols - 1]
y = data.iloc[:, num_of_cols - 1:num_of_cols]
print("features")
print(X)
print("----------------------------------------------")
print("targets")
print(y)
```

```
features
     Age Sex      BP Cholesterol  Na_to_K
0     23   F    HIGH        HIGH   25.355
1     47   M     LOW        HIGH   13.093
2     47   M     LOW        HIGH   10.114
4     61   F     LOW        HIGH   18.043
5     22   F  NORMAL        HIGH    8.607
..   ...  ..     ...         ...      ...
194   46   F    HIGH        HIGH   34.686
195   56   F     LOW        HIGH   11.567
196   16   M     LOW        HIGH   12.006
197   52   M  NORMAL        HIGH    9.894
199   40   F     LOW      NORMAL   11.349

[195 rows x 5 columns]
```

```
---------------------------------------------
targets
      Drug
0    drugY
1    drugC
2    drugC
4    drugY
5    drugX
..     ...
194  drugY
195  drugC
196  drugC
197  drugX
199  drugX

[195 rows x 1 columns]

# First experiment
print('First experiment')
print('-----------------')
# Generate a list of 5 unique random numbers
random_numbers = random.sample(range(1, 101), 5)
print('random_numbers: ', random_numbers)
print()
highest = [0,0,0]
count = 1
for random_seed in random_numbers:
    # the data is shuffled and split into training and testing sets
    X_train, X_test, y_train, y_test = train_test_split(X, y,
shuffle=True, test_size=0.30, random_state=random_seed)

    # categorical features are encoded
    categorical_columns_X = categorical[:-1]
    label_encoder_X = LabelEncoder()

    for i in range(len(categorical_columns_X)):
        X_train[categorical_columns_X[i]] =
label_encoder_X.fit_transform(X_train[categorical_columns_X[i]])
        X_test[categorical_columns_X[i]] =
label_encoder_X.transform(X_test[categorical_columns_X[i]])


    # categorical targets are encoded
    categorical_columns_y = categorical[-1:]
    label_encoder_y = LabelEncoder()
    y_train[categorical_columns_y[0]] =
label_encoder_y.fit_transform(y_train[categorical_columns_y[0]])
    y_test[categorical_columns_y[0]] =
label_encoder_y.transform(y_test[categorical_columns_y[0]])
```

```
    maxD = random.randint(2, 4)
    model = tree.DecisionTreeClassifier(criterion="entropy",
max_depth=maxD)
    model.fit(X_train, y_train)
    y_pred = model.predict(X_test)
    print('Experiment #'+str(count)+':')
    accuracy = model.score(X_test, y_test)
    print("Accuracy: ", accuracy)
    print("Tree depth:", model.tree_.max_depth)
    print("Total number of nodes:", model.tree_.node_count)
    count+=1
    if highest[0] < accuracy:
        highest = [accuracy, model.tree_.max_depth,
model.tree_.node_count]
    elif highest[0] == accuracy:
        if highest[1] < model.tree_.max_depth:
            highest = [accuracy, model.tree_.max_depth,
model.tree_.node_count]

    print()
    print("-" * 50)
    print()
```

```
First experiment
-----------------
random_numbers:  [95, 46, 55, 20, 88]

Experiment #1:
Accuracy:  1.0
Tree depth: 4
Total number of nodes: 11

--------------------------------------------------

Experiment #2:
Accuracy:  0.864406779661017
Tree depth: 2
Total number of nodes: 5

--------------------------------------------------

Experiment #3:
Accuracy:  0.9830508474576272
Tree depth: 4
Total number of nodes: 11

--------------------------------------------------

Experiment #4:
Accuracy:  0.8305084745762712
```

```
Tree depth: 2
Total number of nodes: 5

---------------------------------------------------

Experiment #5:
Accuracy:  0.8983050847457628
Tree depth: 3
Total number of nodes: 9

---------------------------------------------------


print('The highest overall performance')
print("Accuracy: ", highest[0])
print("Tree depth:", highest[1])
print("Total number of nodes:", highest[2])

The highest overall performance
Accuracy:  1.0
Tree depth: 4
Total number of nodes: 11

# Second experiment
print('Second experiment')
print('------------------')

# categorical features are encoded
categorical_columns_X = categorical[:-1]
label_encoder_X = LabelEncoder()

for i in range(len(categorical_columns_X)):
    X[categorical_columns_X[i]] =
label_encoder_X.fit_transform(X[categorical_columns_X[i]])

print("X after encoding")
print(X)

# categorical targets are encoded
categorical_columns_y = categorical[-1:]
label_encoder_y = LabelEncoder()
y[categorical_columns_y[0]] =
label_encoder_y.fit_transform(y[categorical_columns_y[0]])

print()
print("y after encoding")
print(y)

Second experiment
-----------------
X after encoding
```

```
     Age  Sex  BP  Cholesterol  Na_to_K
0     23    0   0            0   25.355
1     47    1   1            0   13.093
2     47    1   1            0   10.114
4     61    0   1            0   18.043
5     22    0   2            0    8.607
..   ...  ...  ..          ...      ...
194   46    0   0            0   34.686
195   56    0   1            0   11.567
196   16    1   1            0   12.006
197   52    1   2            0    9.894
199   40    0   1            1   11.349

[195 rows x 5 columns]

y after encoding
     Drug
0       4
1       2
2       2
4       4
5       3
..    ...
194     4
195     2
196     2
197     3
199     3

[195 rows x 1 columns]

print('Second experiment')
print('-----------------')
trainSizes = [30, 40, 50, 60, 70]
means_accuracy = []
means_nodes = []
for trainSize in trainSizes:
    print('train size:', trainSize)
    # Generate a list of 5 unique random numbers
    random_numbers = random.sample(range(1, 101), 5)
    print('random_numbers: ', random_numbers)
    print()
    accuracies = []
    sizes = []
    for random_seed in random_numbers:
        # the data is shuffled and split into training and testing
sets
        X_train, X_test, y_train, y_test = train_test_split(X, y,
shuffle=True, train_size=(trainSize/100), random_state=random_seed)
```

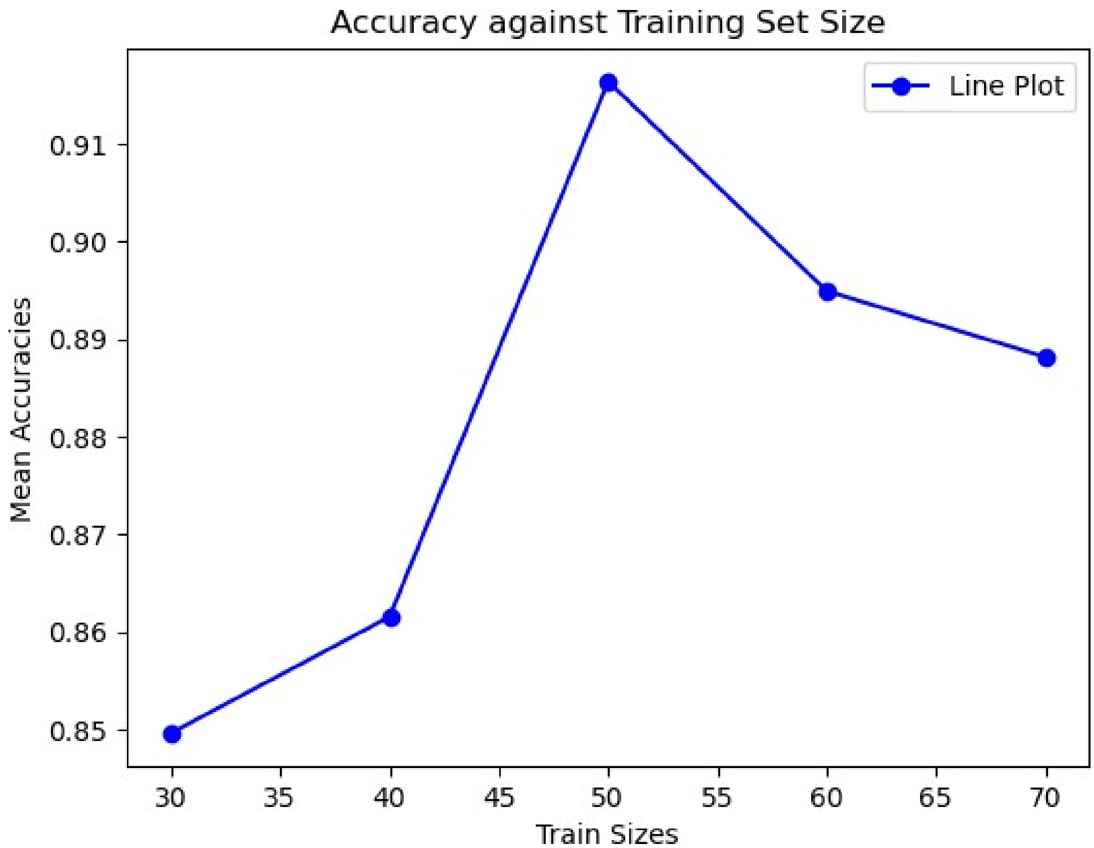
```
        maxD = random.randint(2, 4)
        model = tree.DecisionTreeClassifier(criterion="entropy",
max_depth=maxD)
        model.fit(X_train, y_train)
        y_pred = model.predict(X_test)
        accuracy = model.score(X_test, y_test)
        size = model.tree_.node_count
        accuracies.append(accuracy)
        sizes.append(size)

    print('Mean, Maximum, and Minimum accuracy')
    print('-----------------------------------')
    mean_acc = sum(accuracies)/len(accuracies)
    maximum_acc = max(accuracies)
    minimum_acc = min(accuracies)
    print('Mean: ', mean_acc)
    print('Maximum: ', maximum_acc)
    print('Minimum: ', minimum_acc)
    print()
    means_accuracy.append(mean_acc)
    print('Mean, Maximum, and Minimum tree size')
    print('-----------------------------------')
    mean_node = sum(sizes)/len(sizes)
    maximum_node = max(sizes)
    minimum_node = min(sizes)
    print('Mean: ', mean_node)
    print('Maximum: ', maximum_node)
    print('Minimum: ', minimum_node)
    means_nodes.append(mean_node)
    print()
    print('-'*70)
    print()
```

```
Second experiment
-----------------
train size: 30
random_numbers:  [29, 71, 24, 93, 16]

Mean, Maximum, and Minimum accuracy
-----------------------------------
Mean:  0.8496350364963503
Maximum:  0.8978102189781022
Minimum:  0.7956204379562044

Mean, Maximum, and Minimum tree size
-----------------------------------
Mean:  6.6
Maximum:  9
Minimum:  5
```

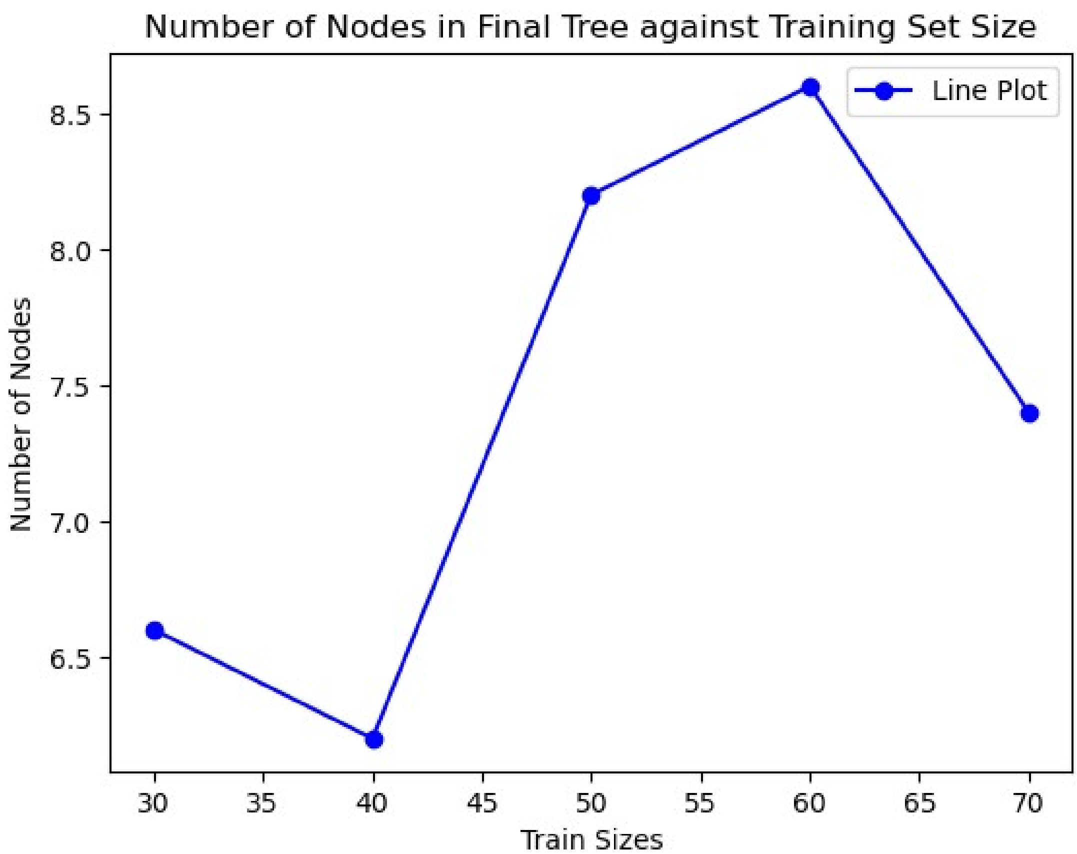
```
----------------------------------------------------------------------

train size: 40
random_numbers:  [29, 48, 70, 80, 52]

Mean, Maximum, and Minimum accuracy
------------------------------------
Mean:  0.8615384615384615
Maximum:  0.9914529914529915
Minimum:  0.811965811965812

Mean, Maximum, and Minimum tree size
------------------------------------
Mean:  6.2
Maximum:  11
Minimum:  5

----------------------------------------------------------------------

train size: 50
random_numbers:  [32, 50, 100, 58, 71]

Mean, Maximum, and Minimum accuracy
------------------------------------
Mean:  0.9163265306122449
Maximum:  1.0
Minimum:  0.8571428571428571

Mean, Maximum, and Minimum tree size
------------------------------------
Mean:  8.2
Maximum:  11
Minimum:  5

----------------------------------------------------------------------

train size: 60
random_numbers:  [1, 39, 19, 8, 82]

Mean, Maximum, and Minimum accuracy
------------------------------------
Mean:  0.8948717948717949
Maximum:  0.9743589743589743
Minimum:  0.8589743589743589

Mean, Maximum, and Minimum tree size
------------------------------------
Mean:  8.6
Maximum:  11
Minimum:  5
```

```
------------------------------------------------------------------------

train size: 70
random_numbers:  [67, 1, 43, 47, 100]

Mean, Maximum, and Minimum accuracy
------------------------------------
Mean:  0.8881355932203391
Maximum:  1.0
Minimum:  0.7966101694915254

Mean, Maximum, and Minimum tree size
------------------------------------
Mean:  7.4
Maximum:  11
Minimum:  5

------------------------------------------------------------------------
```

```
plt.plot(trainSizes,means_accuracy, marker='o', linestyle='-',
color='b', label='Line Plot')
plt.xlabel('Train Sizes')
plt.ylabel('Mean Accuracies')
plt.title('Accuracy against Training Set Size')
plt.legend()
plt.show()
```




```
plt.plot(trainSizes,means_nodes, marker='o', linestyle='-', color='b',
label='Line Plot')
plt.xlabel('Train Sizes')
plt.ylabel('Number of Nodes')
plt.title('Number of Nodes in Final Tree against Training Set Size')
plt.legend()
plt.show()
```


Number of Nodes in Final Tree against Training Set Size
Line Plot
8.5
8.0
7.5
7.0
6.5
Number of Nodes
30
35
40
45
50
55
60
65
70
Train Sizes