In [47]:

```
from IPython.core.display import display, HTML
display(HTML("<style>.container { width:100% !important; }</style>"))
display(HTML("<style>.output_result { max-width:100% !important; }</style>"))
```

In [3]:

```
import pandas as pd
import numpy as np
```

In [7]:

```
from sklearn import tree
from sklearn.model_selection import train_test_split
from sklearn.metrics import classification_report
import graphviz
from sklearn.preprocessing import StandardScaler
```

In [4]:

```
data = pd.read_csv("breast_cancer_data.csv")
```

In [5]:

```
data.shape
```

Out[5]:

```
(569, 32)
```

In [6]:

```
data.diagnosis = [1 if i=="M" else 0 for i in data.diagnosis]
```

In [10]:

```
X_data = data.drop(['id','diagnosis'],axis = 1)
y_data = data.diagnosis
```

In [11]:

```
X_train,X_test, y_train,y_test = train_test_split(X_data,data.diagnosis, test_size=0.2)
```

In [8]:

```
DecisionTree = tree.DecisionTreeClassifier(criterion="gini")
```

In [12]:

```
DecisionTree = DecisionTree.fit(X_train,y_train)
```

In [14]:

```
y_pred = DecisionTree.predict(X_test)
```

In [13]:

```
def gini_index(p: float):
    """Gini index for a given binary class ratio."""
    return 2 * p * (1 - p)
```

In [45]:

```
print("1.1) The Gini Index for the entire dataset is : {}".format(gini_index(len(data[d
ata.diagnosis == 0])/len(data)))) #arg for gini function is class_ratio
```

```
1.1) The Gini Index for the entire dataset is : 0.46753006607546925
```

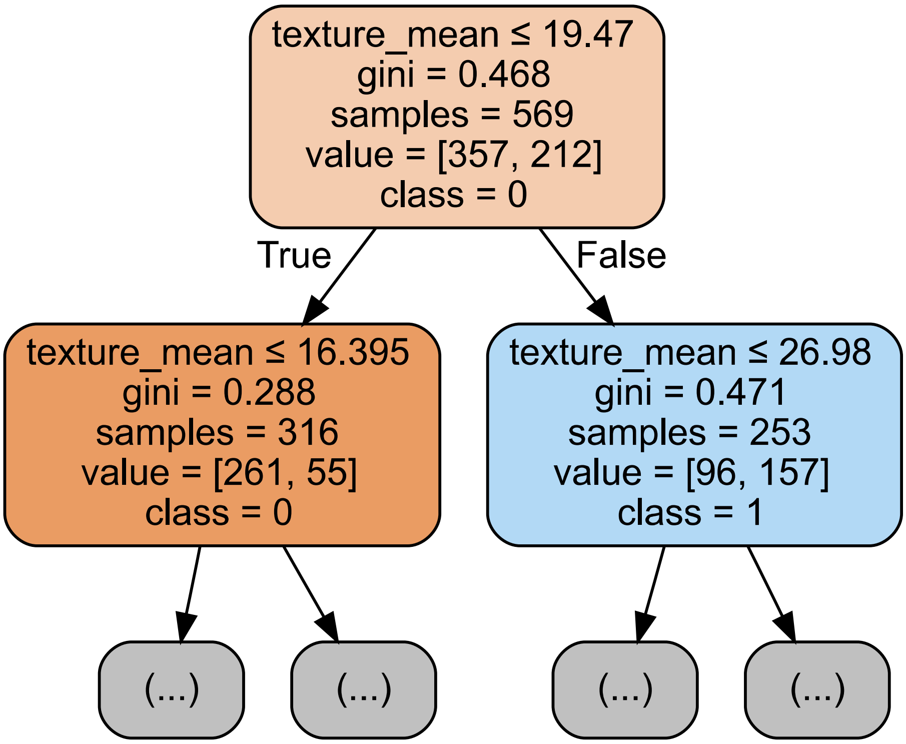
In [26]:

```
DecisionTree_1b = tree.DecisionTreeClassifier(criterion="gini",splitter="best")

X_data_1b = np.asarray(X_data.texture_mean).reshape(-1, 1)
DecisionTree_1b = DecisionTree_1b.fit(X_data_1b,y_data)

dot_data = tree.export_graphviz(
    decision_tree=DecisionTree_1b,
    out_file=None,
    feature_names=['texture_mean'],#X_train.columns,
    class_names=["0", "1"],
    filled=True,
    rounded=True,
    special_characters=True,
    max_depth=1,
)
graph = graphviz.Source(dot_data)
graph.render("cancer_tree")
graph
```

Out[26]:



In [27]:

```
print("1.2) The splitting point for texture_mean with gini index as the error criterion
is : 19.47")
```

```
1.2) The splitting point for texture_mean with gini index as the error cri
terion is : 19.47
```

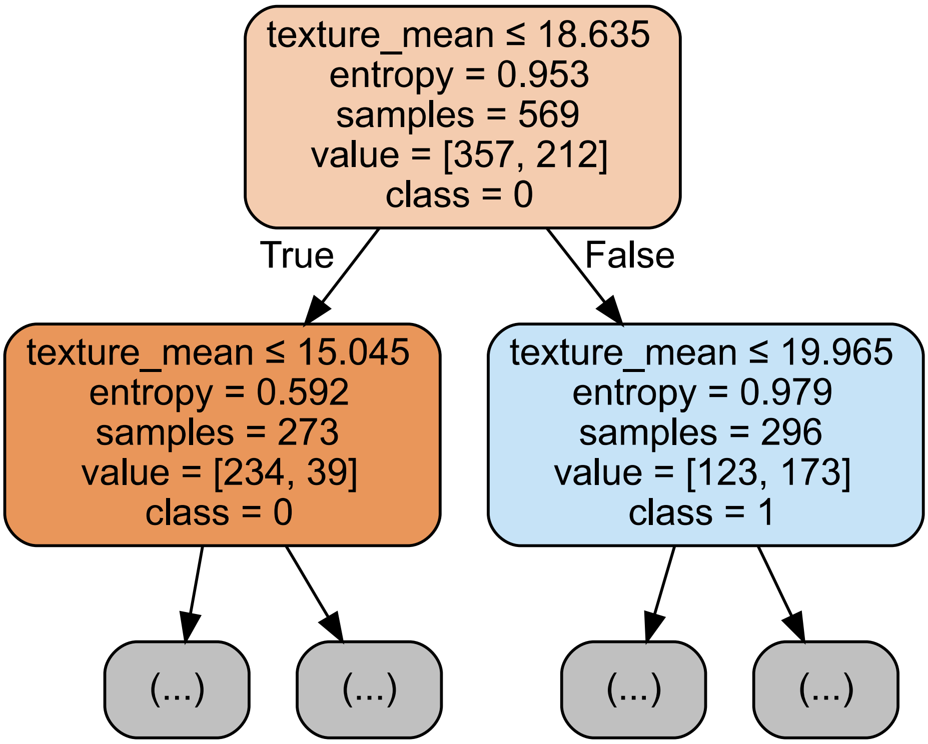
In [28]:

```
DecisionTree_1c = tree.DecisionTreeClassifier(criterion="entropy",splitter="best")

X_data_1c = np.asarray(X_data.texture_mean).reshape(-1, 1)
DecisionTree_1c = DecisionTree_1c.fit(X_data_1c,y_data)

dot_data = tree.export_graphviz(
    decision_tree=DecisionTree_1c,
    out_file=None,
    feature_names=['texture_mean'],#X_train.columns,
    class_names=["0", "1"],
    filled=True,
    rounded=True,
    special_characters=True,
    max_depth=1,
)
graph = graphviz.Source(dot_data)
graph.render("cancer_tree")
graph
```

Out[28]:



In [29]:

```
print("1.3) The splitting point for texture_mean with entropy as the error criterion is
: 18.635")
```

```
1.3) The splitting point for texture_mean with entropy as the error criter
ion is : 18.635
```

Problem 2

In [30]:

```
from keras.models import Sequential
from keras.layers import Dense
from keras.wrappers.scikit_learn import KerasClassifier
from sklearn.model_selection import cross_val_score
from sklearn.preprocessing import LabelEncoder
from sklearn.model_selection import StratifiedKFold
from sklearn.preprocessing import StandardScaler
from sklearn.pipeline import Pipeline
```

In [37]:

```
def create_model():
    model = Sequential()
    model.add(Dense(5, input_shape = (None, 3),activation='relu'))
    model.add(Dense(1, activation='sigmoid'))
    # Compile model
    model.compile(loss='binary_crossentropy', optimizer='adam', metrics=['accuracy'])
    return model
```

In [42]:

```
model = create_model()

model.summary()

print("\n2.1) Total number of parameters in the model is : 26")
```

```
Model: "sequential_2"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 dense_4 (Dense)             (None, None, 5)           20

 dense_5 (Dense)             (None, None, 1)           6

=================================================================
Total params: 26
Trainable params: 26
Non-trainable params: 0
_________________________________________________________________

2.1) Total number of parameters in the model is : 26
```

In [ ]:

```
#using the same dataset since the same attributes are used
```

In [31]:

```
features = [
    "radius_extreme",
    "texture_extreme",
    "perimeter_extreme",
]
```

In [32]:

```
scaler = StandardScaler()
scaler.fit(data[features].values)
X = scaler.transform(data[features].values)
y = data.diagnosis.values.reshape((-1, 1))
```

In [33]:

```
# forward pass for a simple 2-layer NN, with 3 hidden units
np.random.seed(10)

def sigmoid(x):
    """Calculates sigmoid function."""
    return 1. / (1 + np.exp(-x))
def relu(x):
    """Calculates relu function."""
    return np.maximum(0.0,x)
```

In [34]:

```
# parameters for the first layer
W_1 = np.ones(shape=(5, X.shape[1]))
print(f"Shape of W_1 is {W_1.shape}")

b_1 = np.ones(shape=(5, 1))*0.1
print(f"Shape of b_1 is {b_1.shape}")

# parameters for the second layer
W_2 = np.ones(shape=(1, 5))
print(f"Shape of W_2 is {W_2.shape}")

b_2 = np.ones(shape=(1, 1))*0.1
print(f"Shape of b_1 is {b_2.shape}")

# calculate the forward propagation
Z_1 = X @ W_1.T
print(f"\nShape of Z_1 is {Z_1.shape}")
print("Samples for Z_1:")
print(Z_1[:5])

A_1 = relu(Z_1 + b_1.T)
print(f"Shape of A_1 is {A_1.shape}")
print("Samples for A_1:")
print(A_1[:5])

Z_2 = A_1 @ W_2.T
print(f"\nShape of Z_2 is {Z_2.shape}")
print("Samples for Z_2:")
print(Z_1[:5])

A_2 = Y_hat = sigmoid(Z_2 + b_2.T)
print(f"Shape of A_2 is {A_2.shape}")
print("Samples for A_2:")
print(A_2[:5])
```

```
Shape of W_1 is (5, 3)
Shape of b_1 is (5, 1)
Shape of W_2 is (1, 5)
Shape of b_1 is (1, 1)

Shape of Z_1 is (569, 5)
Samples for Z_1:
[[ 2.83099678  2.83099678  2.83099678  2.83099678  2.83099678]
 [ 2.97185021  2.97185021  2.97185021  2.97185021  2.97185021]
 [ 2.83537107  2.83537107  2.83537107  2.83537107  2.83537107]
 [-0.39741967 -0.39741967 -0.39741967 -0.39741967 -0.39741967]
 [ 1.17034432  1.17034432  1.17034432  1.17034432  1.17034432]]
Shape of A_1 is (569, 5)
Samples for A_1:
[[2.93099678 2.93099678 2.93099678 2.93099678 2.93099678]
 [3.07185021 3.07185021 3.07185021 3.07185021 3.07185021]
 [2.93537107 2.93537107 2.93537107 2.93537107 2.93537107]
 [0.         0.         0.         0.         0.        ]
 [1.27034432 1.27034432 1.27034432 1.27034432 1.27034432]]

Shape of Z_2 is (569, 1)
Samples for Z_2:
[[ 2.83099678  2.83099678  2.83099678  2.83099678  2.83099678]
 [ 2.97185021  2.97185021  2.97185021  2.97185021  2.97185021]
 [ 2.83537107  2.83537107  2.83537107  2.83537107  2.83537107]
 [-0.39741967 -0.39741967 -0.39741967 -0.39741967 -0.39741967]
 [ 1.17034432  1.17034432  1.17034432  1.17034432  1.17034432]]
Shape of A_2 is (569, 1)
Samples for A_2:
[[0.99999961]
 [0.99999981]
 [0.99999962]
 [0.52497919]
 [0.99842468]]
```

In [41]:

```
avg_prediction = sum((A_2)/A_2.shape[0])[0]
print("2.2) The average prediction is : {}".format(avg_prediction))
```

```
2.2) The average prediction is : 0.7134461335701471
```

In [46]:

```
log_loss = -np.mean(np.multiply(y, np.log(Y_hat+1E-16)) + np.multiply(1 - y, np.log(1 - Y_hat+1E-16)))
print("2.3) The log loss error metric is : {}".format(log_loss))
```

```
2.3) The log loss error metric is : 0.6811257843182167
```

Answers

Problem 1

1) The Gini Index for the entire dataset is : 0.4675300607546925

2) The splitting point for texture_mean with gini index as the error criterion is : 19.47

3) The splitting point for texture_mean with entropy as the error criterion is : 18.635

Problem 2

1) Total number of parameters in the model is : 26

2) The average prediction after one pass is : 0.7134461335701471

3) The log loss error metric is : 0.6811257843182167

Problem 3

Prove that, in Lloyd's algorithm for K-means clustering, for a given cluster with observations, the averaged pairwise within-cluster squared L2 norm equals to the sum of squared L2 norm of each point (in the cluster) to its cluster center. That is:

$\frac{1}{n}\sum_{i,j\in C}||x_i - x_j||_2^2 = 2\sum_{i\in C}||x_i - \mu||_2^2,$

$where\ \ \mu = \frac{1}{n}\sum_i x_i\ \ is\ the\ centroid\ of\ the\ cluster.$

Proof :

$We\ know\ that\ \ \frac{1}{n}\sum_i x_i\ \ is\ the\ centroid\ of\ the\ cluster.$

$K - Means :$

$a)\ C_1, C_2, C_3, \ldots\ldots., C_k = arg\ min\left(\sum_{k=1}^{k} W(C_k)\right) \Longrightarrow (1)$

$where\ W(C_k)\ measures\ within\ cluster\ variations.$

$b)\ C_1\ \cup\ C_2\ \cup\ C_3\ \cup\ \ldots\ldots. \cup C_k\ = \{1, 2, 3, \ldots\ldots., n\}$

$c)\ C_k\ \cap\ C_{k'}\ = \{1, 2, 3, \ldots\ldots., n\}$
$here\ \ C_1, C_2, C_3, \ldots\ldots., C_k\ denote\ sets\ containing\ the\ indices\ of\ the\ observations\ of\ each\ cl$

$W(C_k) = \frac{1}{|C_k|}\sum_{i,i'\in C_k}\sum_{j=1}^{d}\left(x_{ij} - x_{i'j}\right)^2 \Longrightarrow (2)$

$K - means\ is\ based\ on\ minimizing\ the\ pairwise\ distance\ of\ data\ points\ within\ the\ same\ clu$

$Formally\ given\ x_1, \ldots.., x_n \in R^d,\ wepartition\ these\ points\ into\ k\ clusters\ C_1, \ldots\ldots, C_k\ bas$

$Lloyd's\ algorithm :$

$Substituting\ (2)\ in\ (1)\ \ :$

$$\therefore\ \ arg\ min_{C_1,..,C_k}\left(\sum_{i,i'=1}^{k}\frac{1}{|C_k|}\sum_{i,j\in C_k}||x_{ij} - x_{i'j}||^2\right) \Longrightarrow (3)$$

$Rewriting\ in\ simple\ terms\ \ :$

$$min_{C_1,..,C_k}\left(\sum_{l=1}^{k}\frac{1}{|C_l|}\sum_{i,j\in C_l}||x_i-x_j||^2\right)\implies(4)$$

$Here\ C_1,C_2,C_3,.......,C_k\implies l=1,2,....,k.\ Hence\ written\ as\ C_l.$

$Equation\ (4)\ is\ K-means\ in\ a\ simpler\ way.$

$This\ cost\ function\ is\ a\ weighted\ average\ of\ the\ cluster\ variances\ ,with\ weights\ proportional \approx |C_k|$

$Let\ \ \mu=\frac{1}{|C_l|}\sum_{i\in C_l}^{n}x_i\implies(5)$

$Splitting\ (4)\ and\ considering\ the\ LHS:$

$$\therefore\sum_{i,j\in C_l}||x_i-x_j||^2\ =\sum_{i,j\in C_l}\Big(||x_i||^2+||x_j||^2-2<x_i,x_j>\Big)\implies(6)$$

$Applying\ clusters\ in\ (6):$

$$\begin{aligned}\sum_{i,j\in C_l}||x_i-x_j||^2\ &=\sum_{i\in C_l}\left(|C_l|||x_i||^2+\sum_{j\in C_l}||x_j||^2-2|C_l|<x_i,\mu>\right)\\&=2|C_l|\sum_{i\in C_l}||x_i||^2-2|C_l|^2||\mu||^2\implies(7)\\&\therefore=2|C_l|\left[\sum_{i\in C_l}||x_i||^2-|C_l|||\mu||^2\right]\implies(8)\end{aligned}$$

$Considering\ the\ RHS:$

$$\begin{aligned}\sum_{i\in C_l}||x_i-\mu||^2\ &=\sum_{i\in C_l}\Big(||x_i||^2+||\mu||^2-2<x_i,\mu>\Big)\\&=\sum_{i\in C_l}\Big(||x_i||^2+||C_l||\,||\mu||^2-2\,||C_l||\,||\mu||^2\Big)\\\therefore\sum_{i\in C_l}||x_i-\mu||^2\ &=\sum_{i\in C_l}\Big(||x_i||^2-||C_l||\,||\mu||^2\Big)\implies(9)\end{aligned}$$

$Substituting\ (9)\ in\ (8):$

$$\sum_{i,j\in C_l}||x_i-x_j||^2\ =2|C_l|\left[\sum_{i\in C_l}\Big(||x_i||^2-||C_l||\,||\mu||^2\Big)\right]$$

$Rearranging\ the\ equation\ and\ substituting\ C_l=C_k=n,\ we\ get:$

$$\frac{1}{n}\sum_{i,j\in C_l}||x_i-x_j||_2^2\ =2\sum_{i\in C_l}||x_i-\mu||_2^2$$