Practical 6

Aim : To implement a Machine Learning Classification model using a Decision Tree Classifier algorithm and enhance the model by K Fold and GridSearchCV cross-validation.

In [70]:
```
import numpy as np
import pandas as pd
from sklearn.model_selection import train_test_split, GridSearchCV, KFold
from sklearn.tree import DecisionTreeClassifier
from sklearn.metrics import accuracy_score
```

Step 1: Load the Data

In [71]:
```
data = pd.read_csv(r"Practical5.csv")
#X = data.iloc[:, [1, 2, 3, 4, 5, 6, 7]].values
#y = data.iloc[:, -1].values
X = data.drop(columns=['Outcome'])  # Features
y = data['Outcome']
```

Data preprocessing

In [72]:
```
data.head()
```

Out[72]:

	Pregnancies	Glucose	BloodPressure	SkinThickness	Insulin	BMI	DiabetesPedigreeFunct
0	6	148	72	35	0	33.6	0.
1	1	85	66	29	0	26.6	0.
2	8	183	64	0	0	23.3	0.
3	1	89	66	23	94	28.1	0.
4	0	137	40	35	168	43.1	2.

Data enconding

In [73]:
```
#from sklearn.preprocessing import LabelEncoder
#le = LabelEncoder()
#X[:,0] = le.fit_transform(X[:,0])
X_encoded = pd.get_dummies(X)
```

In [74]:
```
X_train, X_test, y_train, y_test = train_test_split(X_encoded, y, test_size
```

Model building

In [75]:
```
dt_classifier = DecisionTreeClassifier()
```

In [76]:
```
param_grid = {
    'criterion': ['gini', 'entropy'],
    'max_depth': [3, 5, 7, 10],
    'min_samples_split': [2, 5, 10],
    'min_samples_leaf': [1, 2, 4]
}
```

KFold Cross Validation

In [77]:
```
kf = KFold(n_splits=5, shuffle=True, random_state=42)
```

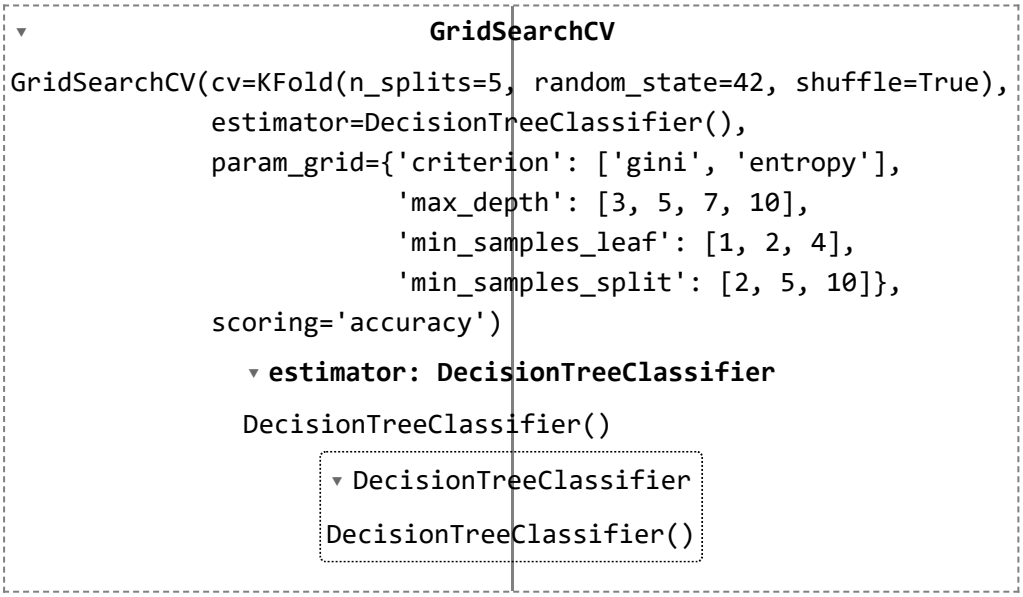
Create a GridSearchCV object

In [78]:
```
grid_search = GridSearchCV(dt_classifier, param_grid, cv=kf, scoring='accur
```

Fit the GridSearchCV object to the training data

In [79]:
```
grid_search.fit(X_train, y_train)
```

Out[79]:



Get the best parameters and best score

In [80]:
```
best_params = grid_search.best_params_
best_score = grid_search.best_score_
```

Instantiate the Decision Tree Classifier with the best parameters

In [81]:
```
best_dt_classifier = DecisionTreeClassifier(**best_params)
```

Train the classifier with the entire training data

In [82]:
```
best_dt_classifier.fit(X_train, y_train)
```

Out[82]:
```
DecisionTreeClassifier(max_depth=7, min_samples_leaf=4, min_samples_split
=10)
```

Make predictions on the test data

In [83]:
```
y_pred = best_dt_classifier.predict(X_test)
```

Calculate accuracy

In [84]:
```
accuracy = accuracy_score(y_test, y_pred)
print("Accuracy:", accuracy)
print("Best Parameters:", best_params)
print("Best Score:", best_score)
```

```
Accuracy: 0.7402597402597403
Best Parameters: {'criterion': 'gini', 'max_depth': 7, 'min_samples_leaf':
4, 'min_samples_split': 10}
Best Score: 0.7492469678795148
```

In [85]:
```
from sklearn.tree import plot_tree
import matplotlib.pyplot as plt

class_names = data.columns[[8]].tolist()
# Plot the decision tree
plt.figure(figsize=(100,100))
plot_tree(best_dt_classifier,  filled=True)
plt.show()
```