In [ ]:

```
import pandas as pd
import numpy as np
import seaborn as sns
import matplotlib.pyplot as plt
```

Q1. Import the dataset and examine the variables. Use descriptive statistics and visualizations to understand the distribution and relationships between the variables.

In [4]:

```
df = pd.read_csv('diabetes.csv')
```

In [7]:

```
df.describe()
```

Out[7]:

	Pregnancies	Glucose	BloodPressure	SkinThickness	Insulin	BMI	DiabetesPedigreeFunction	Age	Outcome
count	768.000000	768.000000	768.000000	768.000000	768.000000	768.000000	768.000000	768.000000	768.000000
mean	3.845052	120.894531	69.105469	20.536458	79.799479	31.992578	0.471876	33.240885	0.348958
std	3.369578	31.972618	19.355807	15.952218	115.244002	7.884160	0.331329	11.760232	0.476951
min	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.078000	21.000000	0.000000
25%	1.000000	99.000000	62.000000	0.000000	0.000000	27.300000	0.243750	24.000000	0.000000
50%	3.000000	117.000000	72.000000	23.000000	30.500000	32.000000	0.372500	29.000000	0.000000
75%	6.000000	140.250000	80.000000	32.000000	127.250000	36.600000	0.626250	41.000000	1.000000
max	17.000000	199.000000	122.000000	99.000000	846.000000	67.100000	2.420000	81.000000	1.000000

In [12]:

```
x = df.corr()
x
```

Out[12]:

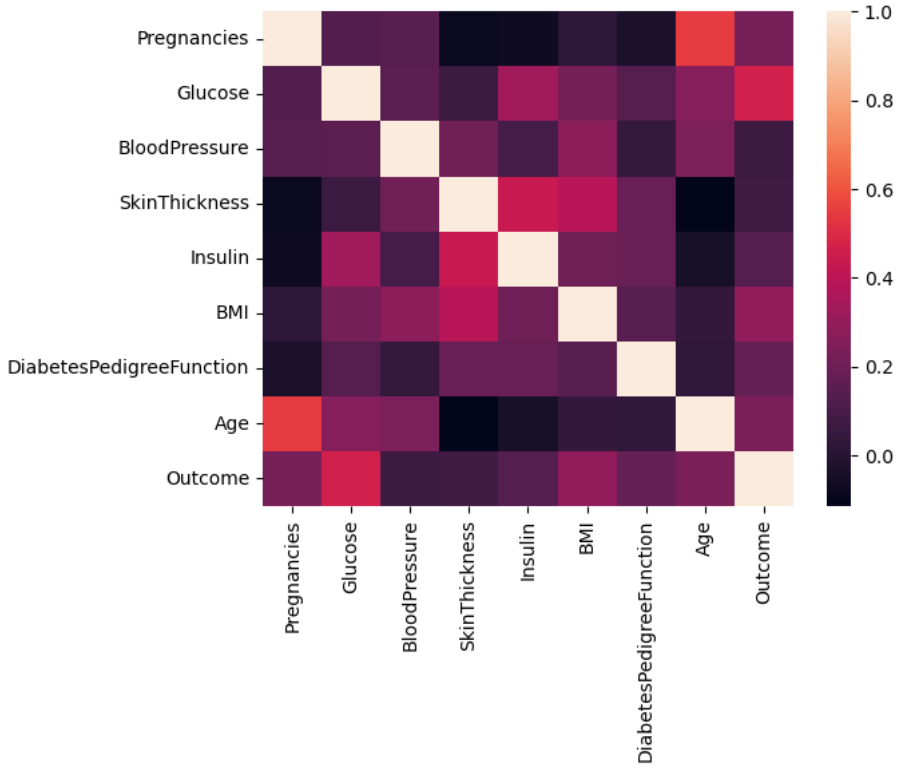
	Pregnancies	Glucose	BloodPressure	SkinThickness	Insulin	BMI	DiabetesPedigreeFunction	Age	Outcome
Pregnancies	1.000000	0.129459	0.141282	-0.081672	-0.073535	0.017683	-0.033523	0.544341	0.221898
Glucose	0.129459	1.000000	0.152590	0.057328	0.331357	0.221071	0.137337	0.263514	0.466581
BloodPressure	0.141282	0.152590	1.000000	0.207371	0.088933	0.281805	0.041265	0.239528	0.065068
SkinThickness	-0.081672	0.057328	0.207371	1.000000	0.436783	0.392573	0.183928	-0.113970	0.074752
Insulin	-0.073535	0.331357	0.088933	0.436783	1.000000	0.197859	0.185071	-0.042163	0.130548
BMI	0.017683	0.221071	0.281805	0.392573	0.197859	1.000000	0.140647	0.036242	0.292695
DiabetesPedigreeFunction	-0.033523	0.137337	0.041265	0.183928	0.185071	0.140647	1.000000	0.033561	0.173844
Age	0.544341	0.263514	0.239528	-0.113970	-0.042163	0.036242	0.033561	1.000000	0.238356
Outcome	0.221898	0.466581	0.065068	0.074752	0.130548	0.292695	0.173844	0.238356	1.000000

In [11]:

```
import seaborn as sns
sns.heatmap(x)
```

Out[11]:

```
<AxesSubplot:>
```



Q2. Preprocess the data by cleaning missing values, removing outliers, and transforming categorical variables into dummy variables if necessary.

In [15]:

```
df[df.isnull()].count()
```

Out[15]:

```
Pregnancies                 0
Glucose                     0
BloodPressure               0
SkinThickness               0
Insulin                     0
BMI                         0
DiabetesPedigreeFunction    0
Age                         0
Outcome                     0
dtype: int64
```

we can show that no missing value

In [20]:

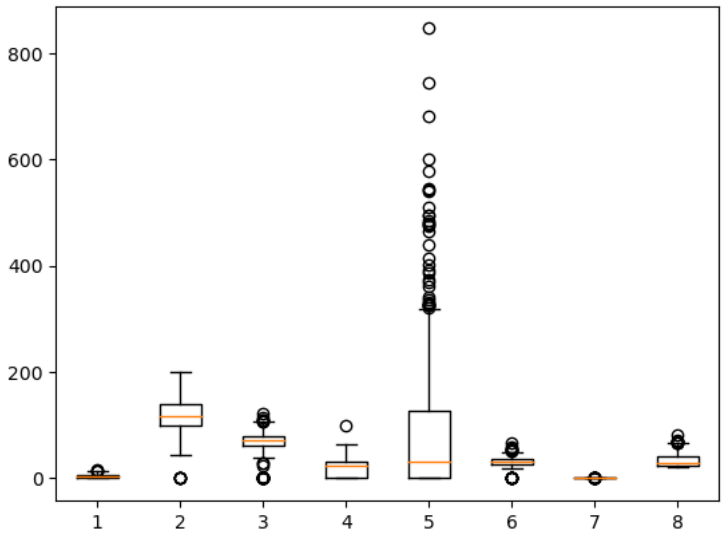
```
data = df.iloc[0,0:8]
```

In [30]:

```
plt.boxplot([df['Pregnancies'],df['Glucose'],df['BloodPressure'],df['SkinThickness'],df['Insulin'],df['BMI'],df['DiabetesPedigreeFunction
```

Out[30]:

```
{'whiskers': [<matplotlib.lines.Line2D at 0x1a1f12a78e0>,
  <matplotlib.lines.Line2D at 0x1a1f12a7c40>,
  <matplotlib.lines.Line2D at 0x1a1f1147a30>,
  <matplotlib.lines.Line2D at 0x1a1f1147be0>,
  <matplotlib.lines.Line2D at 0x1a1f11467c0>,
  <matplotlib.lines.Line2D at 0x1a1f1146b50>,
  <matplotlib.lines.Line2D at 0x1a1f11458b0>,
  <matplotlib.lines.Line2D at 0x1a1f1145130>,
  <matplotlib.lines.Line2D at 0x1a1f12ffbb0>,
  <matplotlib.lines.Line2D at 0x1a1f12fff70>,
  <matplotlib.lines.Line2D at 0x1a1f1302940>,
  <matplotlib.lines.Line2D at 0x1a1f1302cd0>,
  <matplotlib.lines.Line2D at 0x1a1f12a3670>,
  <matplotlib.lines.Line2D at 0x1a1f12a38b0>,
  <matplotlib.lines.Line2D at 0x1a1f13064c0>,
  <matplotlib.lines.Line2D at 0x1a1f11454f0>],
 'caps': [<matplotlib.lines.Line2D at 0x1a1f12a7190>,
  <matplotlib.lines.Line2D at 0x1a1f1140f40>,
  <matplotlib.lines.Line2D at 0x1a1f1141a60>,
  <matplotlib.lines.Line2D at 0x1a1f11410d0>,
  <matplotlib.lines.Line2D at 0x1a1f1146eb0>,
  <matplotlib.lines.Line2D at 0x1a1f11490a0>,
  <matplotlib.lines.Line2D at 0x1a1f1145340>,
  <matplotlib.lines.Line2D at 0x1a1f127f8e0>,
  <matplotlib.lines.Line2D at 0x1a1f1300eb0>,
  <matplotlib.lines.Line2D at 0x1a1f1300280>,
  <matplotlib.lines.Line2D at 0x1a1f13040a0>,
  <matplotlib.lines.Line2D at 0x1a1f1304400>,
  <matplotlib.lines.Line2D at 0x1a1f12a3e50>,
  <matplotlib.lines.Line2D at 0x1a1f1148730>,
  <matplotlib.lines.Line2D at 0x1a1f1300d60>,
  <matplotlib.lines.Line2D at 0x1a1f13005e0>],
 'boxes': [<matplotlib.lines.Line2D at 0x1a1f12a7520>,
  <matplotlib.lines.Line2D at 0x1a1f11476d0>,
  <matplotlib.lines.Line2D at 0x1a1f1146040>,
  <matplotlib.lines.Line2D at 0x1a1f1149be0>,
  <matplotlib.lines.Line2D at 0x1a1f12f8040>,
  <matplotlib.lines.Line2D at 0x1a1f13026a0>,
  <matplotlib.lines.Line2D at 0x1a1f1304fa0>,
  <matplotlib.lines.Line2D at 0x1a1f1306130>],
 'medians': [<matplotlib.lines.Line2D at 0x1a1f12af370>,
  <matplotlib.lines.Line2D at 0x1a1eb7b16a0>,
  <matplotlib.lines.Line2D at 0x1a1f1149640>,
  <matplotlib.lines.Line2D at 0x1a1f127d6d0>,
  <matplotlib.lines.Line2D at 0x1a1f13009d0>,
  <matplotlib.lines.Line2D at 0x1a1f13046a0>,
  <matplotlib.lines.Line2D at 0x1a1f1148070>,
  <matplotlib.lines.Line2D at 0x1a1f12f87c0>],
 'fliers': [<matplotlib.lines.Line2D at 0x1a1f1147220>,
  <matplotlib.lines.Line2D at 0x1a1f0ee2670>,
  <matplotlib.lines.Line2D at 0x1a1f11499d0>,
  <matplotlib.lines.Line2D at 0x1a1f127d8e0>,
  <matplotlib.lines.Line2D at 0x1a1f1302340>,
  <matplotlib.lines.Line2D at 0x1a1f1304a30>,
  <matplotlib.lines.Line2D at 0x1a1f1148a60>,
  <matplotlib.lines.Line2D at 0x1a1f12f82b0>],
 'means': []}
```



we can see 5 numer feature insulin has many outlier

Q3. Split the dataset into a training set and a test set. Use a random seed to ensure reproducibility.

In [66]:

```
X = df.drop('Outcome',axis = 1)
y = df['Outcome']
from sklearn.model_selection import train_test_split
X_train,X_test,y_train,y_test = train_test_split(X,y,random_state=42,test_size= 0.2)
X_train.shape,X_test.shape,y_train.shape,y_test.shape
```

Out[66]:

```
((614, 8), (154, 8), (614,), (154,))
```

Q4. Use a decision tree algorithm, such as ID3 or C4.5, to train a decision tree model on the training set. Use cross-validation to optimize the hyperparameters and avoid overfitting.

In [68]:

```
from sklearn.tree import DecisionTreeClassifier
from sklearn import tree
model = DecisionTreeClassifier(criterion='entropy',max_depth = 3)
model.fit(X_train,y_train)
y_pred = model.predict(X_test)
y_pred
```

Out[68]:

```
array([0, 0, 0, 0, 0, 0, 0, 0, 1, 1, 0, 1, 1, 0, 0, 0, 0, 1, 1, 0, 0,
       0, 0, 1, 1, 0, 0, 0, 0, 1, 1, 1, 1, 0, 1, 1, 0, 0, 1, 0, 0, 0, 0,
       0, 1, 0, 0, 0, 1, 0, 1, 1, 0, 0, 0, 1, 0, 0, 1, 1, 0, 0, 0, 0, 1,
       0, 1, 0, 1, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 1, 0, 1, 1, 0,
       0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 1, 0, 1, 0, 0, 1, 1, 0, 0, 1, 0, 0,
       0, 1, 0, 1, 0, 0, 1, 0, 0, 0, 0, 0, 0, 1, 0, 0, 1, 1, 1, 1, 1,
       0, 0, 1, 0, 0, 1, 1, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 1, 0, 0, 1, 0],
      dtype=int64)
```

Q5. Evaluate the performance of the decision tree model on the test set using metrics such as accuracy, precision, recall, and F1 score. Use confusion matrices and ROC curves to visualize the results.

In [69]:

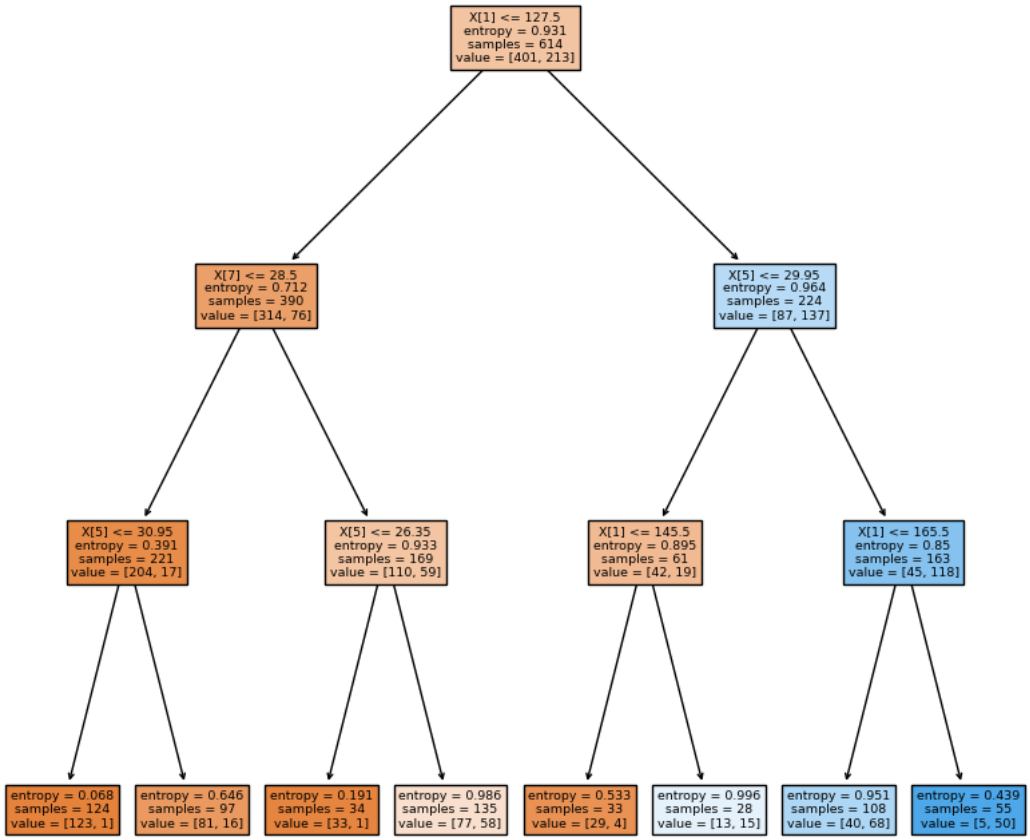
```
from sklearn.metrics import confusion_matrix,accuracy_score,classification_report
print(f'Confusion Matrix = {confusion_matrix(y_test,y_pred)}')
print(f'accuracy score {accuracy_score(y_test,y_pred)}')
print(f'Report = {classification_report(y_test,y_pred)}')
```

```
Confusion Matrix = [[83 16]
 [20 35]]
accuracy score 0.7662337662337663
Report =               precision    recall  f1-score   support

           0       0.81      0.84      0.82        99
           1       0.69      0.64      0.66        55

    accuracy                           0.77       154
   macro avg       0.75      0.74      0.74       154
weighted avg       0.76      0.77      0.76       154
```

Q6. Interpret the decision tree by examining the splits, branches, and leaves. Identify the most important variables and their thresholds. Use domain knowledge and common sense to explain the patterns and trends.

In [70]:

```
plt.figure(figsize=(10,10))
tree.plot_tree(model,filled=True)
plt.show()
```



Q7. Validate the decision tree model by applying it to new data or testing its robustness to changes in the dataset or the environment. Use sensitivity analysis and scenario testing to explore the uncertainty and risks.

In [74]:

```
pred = model.predict([[6,197,70,45,53,30.5,0.15,53]])
pred
```

```
C:\Users\hiren\anaconda3\lib\site-packages\sklearn\base.py:450: UserWarning: X does not have valid feature names, but Decis
ionTreeClassifier was fitted with feature names
  warnings.warn(
```

Out[74]:

```
array([1], dtype=int64)
```