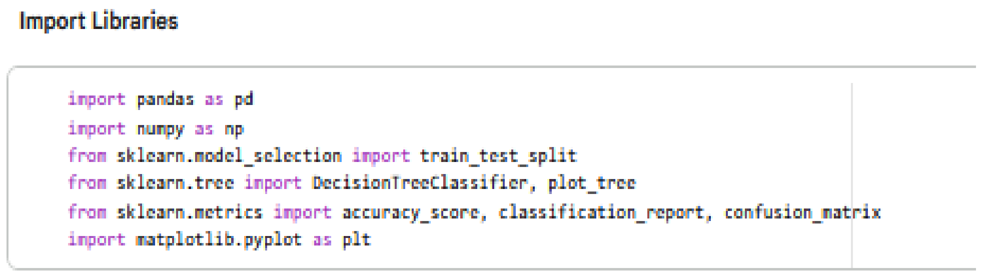
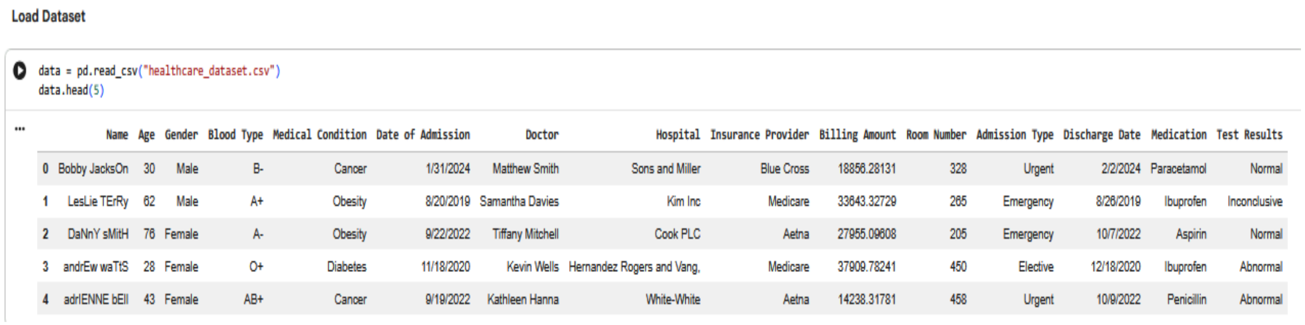
Task 3: Build a decision tree classifier to predict whether a customer will purchase a product or service based on their demographic and behavioral data. Use a dataset such as the Bank Marketing dataset from the UCI Machine Learning Repository.



Import Libraries

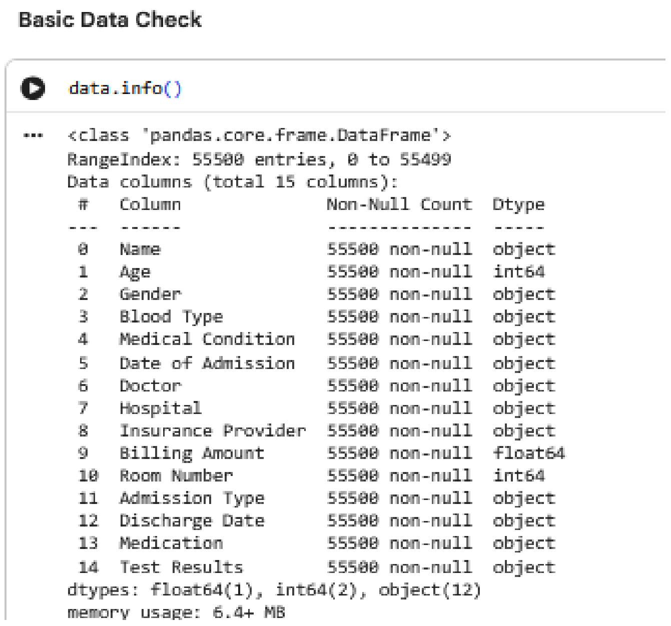
```
import pandas as pd
import numpy as np
from sklearn.model_selection import train_test_split
from sklearn.tree import DecisionTreeClassifier, plot_tree
from sklearn.metrics import accuracy_score, classification_report, confusion_matrix
import matplotlib.pyplot as plt
```



Load Dataset

```
data = pd.read_csv("healthcare_dataset.csv")
data.head(5)
```

	Name	Age	Gender	Blood Type	Medical Condition	Date of Admission	Doctor	Hospital	Insurance Provider	Billing Amount	Room Number	Admission Type	Discharge Date	Medication	Test Results
0	Bobby JacksOn	30	Male	B-	Cancer	1/31/2024	Matthew Smith	Sons and Miller	Blue Cross	18856.28131	328	Urgent	2/2/2024	Paracetamol	Normal
1	LesLie TErRy	62	Male	A+	Obesity	8/20/2019	Samantha Davies	Kim Inc	Medicare	33643.32729	265	Emergency	8/26/2019	Ibuprofen	Inconclusive
2	DaNnY sMitH	76	Female	A-	Obesity	9/22/2022	Tiffany Mitchell	Cook PLC	Aetna	27955.09608	205	Emergency	10/7/2022	Aspirin	Normal
3	andrEw waTtS	28	Female	O+	Diabetes	11/18/2020	Kevin Wells	Hernandez Rogers and Vang,	Medicare	37909.78241	450	Elective	12/18/2020	Ibuprofen	Abnormal
4	adrIENNE bEll	43	Female	AB+	Cancer	9/19/2022	Kathleen Hanna	White-White	Aetna	14238.31781	458	Urgent	10/9/2022	Penicillin	Abnormal



Basic Data Check

```
data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 55500 entries, 0 to 55499
Data columns (total 15 columns):
 #   Column              Non-Null Count  Dtype  
---  ------              --------------  -----  
 0   Name                55500 non-null  object 
 1   Age                 55500 non-null  int64  
 2   Gender              55500 non-null  object 
 3   Blood Type          55500 non-null  object 
 4   Medical Condition   55500 non-null  object 
 5   Date of Admission   55500 non-null  object 
 6   Doctor              55500 non-null  object 
 7   Hospital            55500 non-null  object 
 8   Insurance Provider  55500 non-null  object 
 9   Billing Amount      55500 non-null  float64
 10  Room Number         55500 non-null  int64  
 11  Admission Type      55500 non-null  object 
 12  Discharge Date      55500 non-null  object 
 13  Medication          55500 non-null  object 
 14  Test Results        55500 non-null  object 
dtypes: float64(1), int64(2), object(12)
memory usage: 6.4+ MB
```

```
data.isnull().sum()
```

	0
Name	0
Age	0
Gender	0
Blood Type	0
Medical Condition	0
Date of Admission	0
Doctor	0
Hospital	0
Insurance Provider	0
Billing Amount	0
Room Number	0
Admission Type	0
Discharge Date	0
Medication	0
Test Results	0

dtype: int64

Data Cleaning

Fill Missing Values

```
for col in data.columns:
    if data[col].dtype == 'object':
        data[col].fillna(data[col].mode()[0], inplace=True)
    else:
        data[col].fillna(data[col].mean(), inplace=True)
```

```
/tmp/ipython-input-834479767.py:3: FutureWarning: A value is trying to
The behavior will change in pandas 3.0. This inplace method will never

For example, when doing 'df[col].method(value, inplace=True)', try usi


  data[col].fillna(data[col].mode()[0], inplace=True)
/tmp/ipython-input-834479767.py:5: FutureWarning: A value is trying to
The behavior will change in pandas 3.0. This inplace method will never

For example, when doing 'df[col].method(value, inplace=True)', try usi


  data[col].fillna(data[col].mean(), inplace=True)
```

Remove Duplicate Rows

```
data.drop_duplicates(inplace=True)
```

Select Features and Target

```
# Admission Type
X = data[['Age', 'Gender', 'Blood Type', 'Medical Condition',
          'Insurance Provider', 'Billing Amount']]

y = data['Admission Type']
```

Convert Categorical Columns into Numbers

```
X = pd.get_dummies(X)
y = pd.get_dummies(y)
```

Train-Test Split

```
X_train, X_test, y_train, y_test = train_test_split(
    X, y, test_size=0.2, random_state=42
)
```

Train Decision Tree Model

```
model = DecisionTreeClassifier(max_depth=5)
model.fit(X_train, y_train)
```

```
DecisionTreeClassifier
DecisionTreeClassifier(max_depth=5)
```

Prediction

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

Model Evaluation

Accuracy

```
accuracy = accuracy_score(y_test, y_pred)
print("Model Accuracy:", accuracy)
```

```
Model Accuracy: 0.00227396761870111
```

Confusion Matrix

```
confusion_matrix(y_test.values.argmax(axis=1),
y_pred.argmax(axis=1))
```

```
array([[3657,    5,    3],
       [3677,    5,    9],
       [3626,    7,    5]])
```

Classification Report

```
print(classification_report(
    y_test.values.argmax(axis=1),
    y_pred.argmax(axis=1)
))
```

```
...              precision    recall  f1-score   support

           0       0.33      1.00      0.50      3665
           1       0.29      0.00      0.00      3691
           2       0.29      0.00      0.00      3638

    accuracy                           0.33     10994
   macro avg       0.31      0.33      0.17     10994
weighted avg       0.31      0.33      0.17     10994
```

Visualize Decision Tree

```
plt.figure(figsize=(20,10))
plot_tree(model,
          feature_names=X.columns,
          filled=True)
plt.show()
```

```
...
```