from sklearn.model\_selection import train\_test\_split  
from sklearn.tree import DecisionTreeClassifier  
from sklearn.datasets import load\_iris  
from sklearn import metrics  
import matplotlib.pyplot as plt  
from sklearn.tree import plot\_tree  
import numpy as np  
import pandas as pd

# data = load\_iris()  
# X = data.data   
# y = data.target

df = pd.read\_csv('https://raw.githubusercontent.com/mahesh147/Decision-Tree-Classifier/refs/heads/master/Social\_Network\_Ads.csv')  
df.head()

User ID Gender Age EstimatedSalary Purchased  
0 15624510 Male 19 19000 0  
1 15810944 Male 35 20000 0  
2 15668575 Female 26 43000 0  
3 15603246 Female 27 57000 0  
4 15804002 Male 19 76000 0

df['Gender'] = df['Gender'].map({'Male': 1, 'Female': 0})

X = df.drop(['User ID', 'Purchased'], axis=1)  
y = df[['Purchased']]

X\_train, X\_test, y\_train, y\_test = train\_test\_split(X, y, test\_size=0.2, random\_state=42)  
model = DecisionTreeClassifier(max\_depth=3)  
model.fit(X\_train, y\_train)

DecisionTreeClassifier(max\_depth=3)

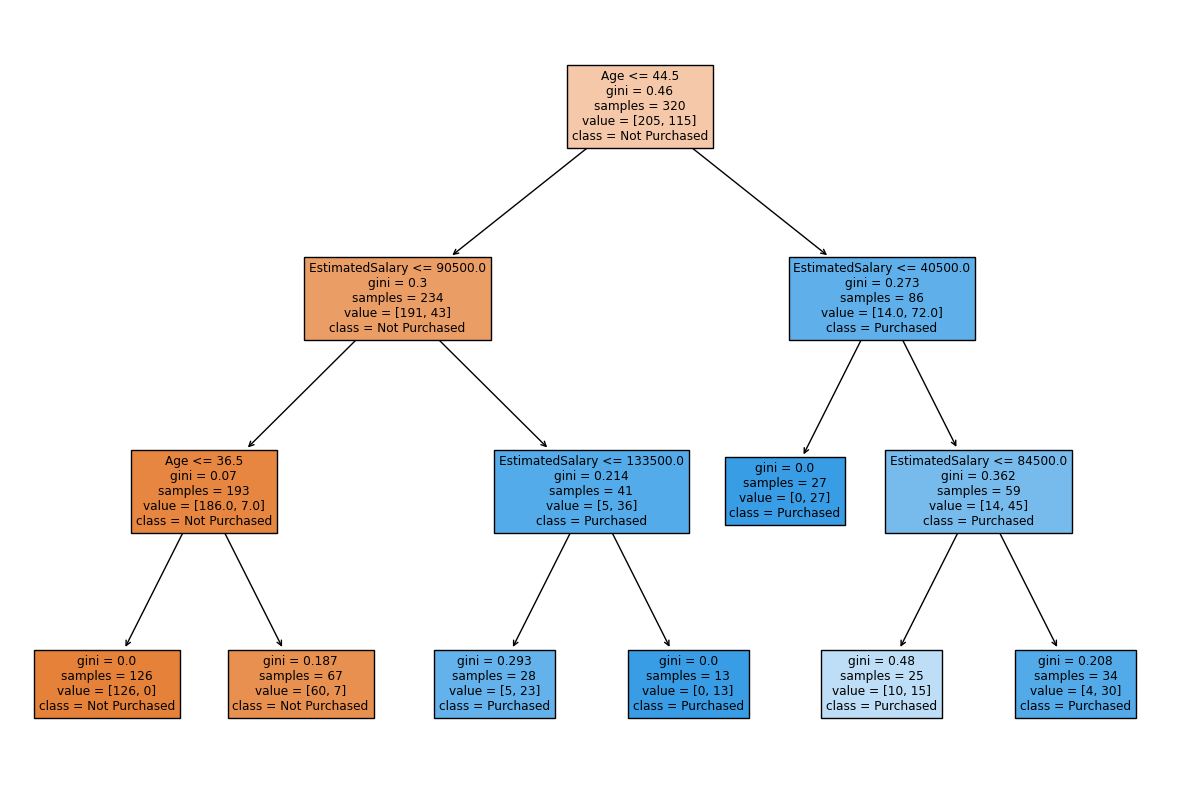
y\_pred = model.predict(X\_test)

accuracy = metrics.accuracy\_score(y\_test, y\_pred)  
print(f"Model aniqligi: {accuracy:.2f}")

Model aniqligi: 0.91

feature\_names = ["Gender", "Age", "EstimatedSalary"]  
class\_names = ["Not Purchased", "Purchased"]

plt.figure(figsize=(15, 10))  
plot\_tree(model, filled=True, feature\_names=feature\_names, class\_names=class\_names)  
plt.show()



4-topshiriq

class ExpertSystem:  
 def \_\_init\_\_(self):  
   
 self.rules = [  
 {"condition": lambda x: x['budget'] < 15000 and x['usage'] == "city", "recommendation": "Toyota Yaris"},  
 {"condition": lambda x: x['budget'] < 15000 and x['usage'] == "highway", "recommendation": "Honda Civic"},  
 {"condition": lambda x: x['budget'] >= 15000 and x['budget'] < 30000 and x['usage'] == "city", "recommendation": "BMW 3 Series"},  
 {"condition": lambda x: x['budget'] >= 15000 and x['budget'] < 30000 and x['usage'] == "highway", "recommendation": "Audi A4"},  
 {"condition": lambda x: x['budget'] >= 30000 and x['usage'] == "city", "recommendation": "Tesla Model 3"},  
 {"condition": lambda x: x['budget'] >= 30000 and x['usage'] == "highway", "recommendation": "Mercedes-Benz E-Class"},  
 ]  
  
 def ask\_user(self):  
 """Foydalanuvchidan kerakli ma'lumotlarni so'rash."""  
 budget = float(input("Avtomobil uchun byudjetni kiriting (USD): "))  
 usage = input("Avtomobil qanday maqsadda ishlatiladi? (city / highway): ").lower()  
   
 user\_data = {'budget': budget, 'usage': usage}  
   
 return user\_data  
  
 def information(self, user\_data):  
 """Foydalanuvchi ma'lumotlariga asoslangan tavsiyalar berish."""  
 for rule in self.rules:  
 if rule['condition'](user\_data):  
 return f"Sizga tavsiya qilinadi: {rule['recommendation']}"  
   
 return "Afsuski, sizning talablaringizga mos keladigan avtomobil tavsiyasi topilmadi."

user\_data = ExpertSystem().ask\_user()

recommendation = ExpertSystem().information(user\_data)  
  
print(recommendation)

Sizga tavsiya qilinadi: Tesla Model 3