Richard Nadar
RA1911030010109
5-4-22

ARTIFICIAL INTELLIGENCE (18CSC305J) LAB

EXPERIMENT 9: Implementation of uncertain methods for an application

AIM: To implement uncertain methods for an application.

CODE:

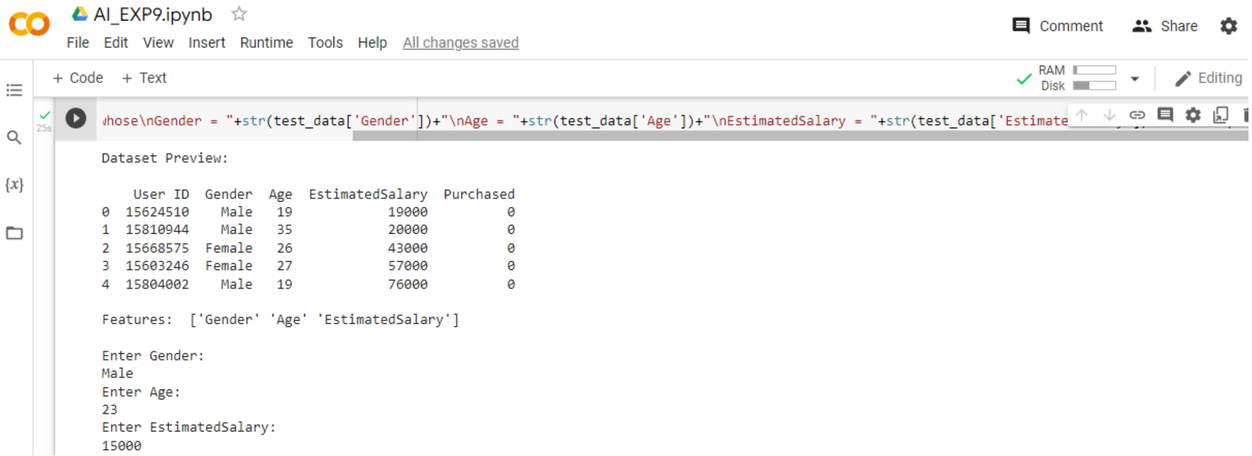
Language used: Python

```
import pandas as pd
import numpy as np

#Importing Dataset
dataset = pd.read_csv('Social_Network_Ads.csv')
print("Dataset Preview:\n")
print(dataset.head())

def calculate_entropy(d_label):
   classes,class_counts = np.unique(d_label,return_counts=True)
   entropy_value = np.sum([(-
class_counts[i]/np.sum(class_counts))*np.log2(class_counts[i]/np.sum(class_counts))
for i in range(len(classes))])
   return entropy_value

def calculate_infogain(dataset, feature, c_label):
   dataset_entropy = calculate_entropy(dataset[c_label])
   values,feat_counts = np.unique(dataset[feature],return_counts=True)
   weighted_feature_entropy =
np.sum([(feat_counts[i]/np.sum(feat_counts))*calculate_entropy(dataset.where(datase
t[feature]==values[i]).dropna()[c_label]) for i in range(len(values))])
   feature_info_gain = dataset_entropy - weighted_feature_entropy
   return feature_info_gain
```

```
def create_dtree(dataset, features, c_label, parent):
    datum = np.unique(dataset[c_label], return_counts=True)
    unique_data = np.unique(dataset[c_label])

    if len(unique_data) <= 1:
        return unique_data[0]

    elif len(dataset) == 0:
        return unique_data[np.argmax(datum[1])]

    elif len(features) == 0:
        return parent

    else:
        parent = unique_data[np.argmax(datum[1])]
        item_values = [calculate_infogain(dataset, feature, c_label) for feature in features]
        optimum_feature_index = np.argmax(item_values)
        optimum_feature = features[optimum_feature_index]
        decision_tree = {optimum_feature: {}}
        features = [i for i in features if i != optimum_feature]

        for value in np.unique(dataset[optimum_feature]):
            min_data = dataset.where(dataset[optimum_feature] == value).dropna()
            min_tree = create_dtree(min_data, features, c_label, parent)
            decision_tree[optimum_feature][value] = min_tree
        return (decision_tree)

def predict_purchase(test_data, decision_tree):
    for nodes in decision_tree.keys():
        value = test_data[nodes]
        decision_tree = decision_tree[nodes][value]
        prediction = 0
        if type(decision_tree) is dict:
            prediction = predict_purchase(test_data, decision_tree)
        else:
            prediction = decision_tree
            break
    return prediction
```

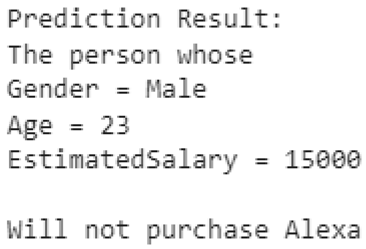
```
#Extracting Features from Dataset
features = dataset.columns[1:-1]
c_label = 'Purchased'
parent = None
print("\nFeatures: ", features.values)

#Creation of decision tree
decision_tree = create_dtree(dataset, features, c_label, parent)

#Enter values for predicting results
test_data = {}
print("\nEnter Gender: ")
gender = input()
print("Enter Age: ")
age = int(input())
print("Enter EstimatedSalary: ")
salary = int(input())

test_data['Gender'] = gender
test_data['Age'] = age
test_data['EstimatedSalary'] = salary

t_data=pd.Series(test_data)

#Making Predictions
prediction = predict_purchase(t_data, decision_tree)
if(prediction>=1):
   pur = 'Will purchase Alexa'
else:
   pur = 'Will not purchase Alexa'

print("\nPrediction Result:\nThe person whose\nGender = 
"+str(test_data['Gender'])+"\nAge = "+str(test_data['Age'])+"\nEstimatedSalary = 
"+str(test_data['EstimatedSalary'])+"\n\n" + pur)
```

OUTPUT:



```
whose\nGender = "+str(test_data['Gender'])+"\nAge = "+str(test_data['Age'])+"\nEstimatedSalary = "+str(test_data['Estimate
```

```
Dataset Preview:

     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

Features:  ['Gender' 'Age' 'EstimatedSalary']

Enter Gender:
Male
Enter Age:
23
Enter EstimatedSalary:
15000
```



```
Prediction Result:
The person whose
Gender = Male
Age = 23
EstimatedSalary = 15000

Will not purchase Alexa
```

RESULT: Hence, we successfully implemented uncertain methods for an application and verified the output and documented the result.