National University of Sciences and Technology (NUST)
School of Electrical Engineering and Computer Science

Department of Computing

CS370: Artificial Intelligence

Class: BSCS-10AB

Lab 05: Naïve Bayes Classifier

Date: 02-03-2023

Instructor: Dr. Hashir Kiani

Lab Engineer: Ms Shakeela Bibi

Lab Task: Naive Bayes Classifier Implementation

Train a Naïve Bayes classifier considering the following table data. The 'Status' represents the (class) label, while department, age, and salary are attributes/predictors.

department	*status*	*age*	*salary*
sales	senior	31...35	46K...50K
sales	junior	26...30	26K...30K
sales	junior	31...35	31K...35K
systems	junior	21...25	46K...50K
systems	senior	31...35	66K...70K
systems	junior	26...30	46K...50K
systems	senior	41...45	66K...70K
marketing	senior	36...40	46K...50K
marketing	junior	31...35	41K...45K
secretary	senior	46...50	36K...40K
secretary	junior	26...30	26K...30K

Once you have your classifier, test the following unseen instances:

(a) Marketing, 31…35, 46K…50K

(b) Sales, 31…35, 66K…70K

Note that you might encounter the 'Zero Frequency Problem'. The simplest solution for zero frequency problem is Laplace smoothing. For help, refer to the following YouTube video.

https://www.youtube.com/watch?v=gCI-ZC7irbY

You might need to solve the problem manually first for ease, and then implement the classifier.

Code

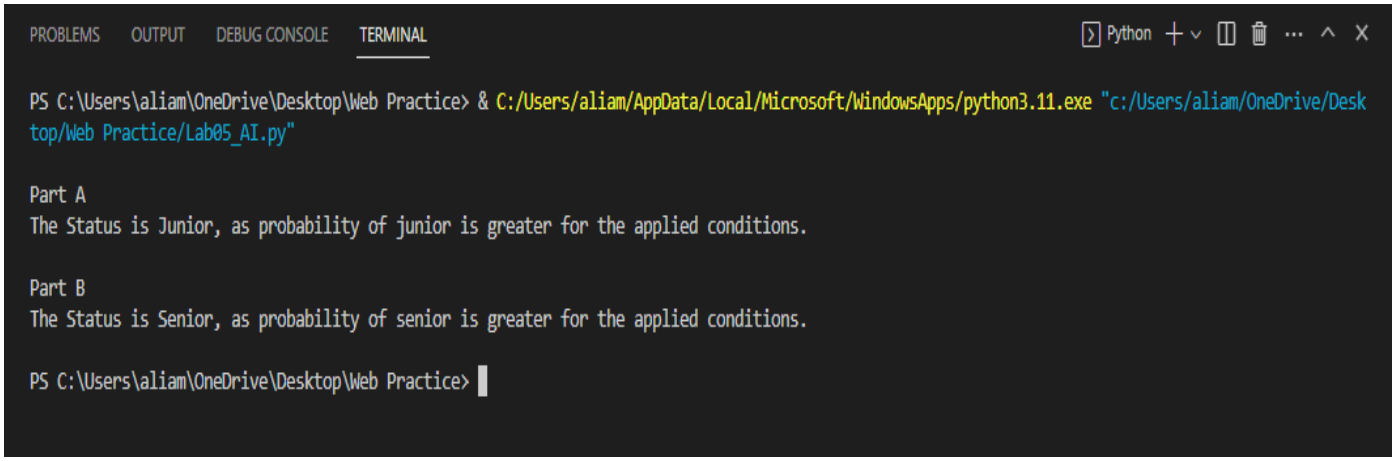
```
# Define a function to add data to department, age, and salary dictionaries
def addDataSet(dept, ag, sal, status):
    global department, age, salary

    if status == "senior":
        # If department is already in the dictionary, increment its count
        if (dept in department_s):
            department_s[dept] = department_s[dept]+1
        else:
            # Otherwise, add it with a count of 1
            department_s[dept] = 1

        # If age is already in the dictionary, increment its count
        if (ag in age_s):
            age_s[ag] = age_s[ag]+1
        else:
            # Otherwise, add it with a count of 1
            age_s[ag] = 1

        # If salary is already in the dictionary, increment its count
        if (sal in salary_s):
            salary_s[sal] = salary_s[sal]+1
        else:
            # Otherwise, add it with a count of 1
            salary_s[sal] = 1
    else:
        # If department is already in the dictionary, increment its count
        if (dept in department_j):
            department_j[dept] = department_j[dept]+1
        else:
            # Otherwise, add it with a count of 1
            department_j[dept] = 1

        # If age is already in the dictionary, increment its count
        if (ag in age_j):
            age_j[ag] = age_j[ag]+1
        else:
            # Otherwise, add it with a count of 1
            age_j[ag] = 1

        # If salary is already in the dictionary, increment its count
        if (sal in salary_j):
            salary_j[sal] = salary_j[sal]+1
        else:
            # Otherwise, add it with a count of 1
            salary_j[sal] = 1


if __name__ == "__main__":
    # Define empty dictionaries for department, age, and salary
    department_s = dict()  # dictionary to store the counts for department for senior employees
    department_j = dict()  # dictionary to store the counts for department for junior employees
    age_s = dict()  # dictionary to store the counts for age range for senior employees
    age_j = dict()  # dictionary to store the counts for age range for junior employees
```

```
salary_s = dict()  # dictionary to store the counts for salary range for senior employees
salary_j = dict()  # dictionary to store the counts for salary range for junior employees

# Define lists for department, age, salary, and status data
department_list = ["sales", "sales", "sales", "systems", "systems",
                   "systems", "systems", "marketing", "marketing", "secretary", "secretary"]  # list of departments
age_list = ["31...35", "26...30", "31...35", "21...25", "31...35", "26...30", "41...45", "36...40", "31...35",
            "46...50", "26...30"]  # list of age ranges
salary_list = ["46k...50k", "26k...30k", "31k...35k", "46k...50k", "66k...70k", "46k...50k", "66k...70k",
               "46k...50k", "41k...45k", "36k...40k", "26k...30k"]  # list of salary ranges
status_list = ["senior", "junior", "junior", "junior", "senior",
               "junior", "senior", "senior", "junior", "senior", "junior"]  # list of employment status

# Iterate through the data lists and add the data to the dictionaries using the addDataSet function
for i in range(11):
    addDataSet(department_list[i], age_list[i],
               salary_list[i], status_list[i])

department_list_s = department_s.keys()  # list of departments for senior employees
age_list_s = age_s.keys()  # list of age ranges for senior employees
salary_list_s = salary_s.keys()  # list of salary ranges for senior employees

department_list_j = department_j.keys()  # list of departments for junior employees
age_list_j = age_j.keys()  # list of age ranges for junior employees
salary_list_j = salary_j.keys()  # list of salary ranges for junior employees

probaility_dict_s = dict()  # dictionary to store probabilities for senior employees
probaility_dict_j = dict()  # dictionary to store probabilities for junior employees

sum_department_s = 0
for i in department_s.values():
    sum_department_s = sum_department_s+i  # calculate total count of departments for senior employees

sum_department_j = 0
for i in department_j.values():
    sum_department_j = sum_department_j+i  # calculate total count of departments for junior employees

sum_age_s = 0
for i in age_s.values():
    sum_age_s = sum_age_s+i  # calculate total count of age ranges for senior employees

sum_age_j = 0
for i in age_j.values():
    sum_age_j = sum_age_j+i  # calculate total count of age ranges for junior employees

sum_salary_s = 0  # initializing a variable to store the sum of salaries for seniors

for i in salary_s.values():  # loop through the salary values for seniors
    sum_salary_s = sum_salary_s+i  # add each salary to the sum_salary_s variable

sum_salary_j = 0  # initializing a variable to store the sum of salaries for juniors
for i in salary_j.values():  # loop through the salary values for juniors
    sum_salary_j = sum_salary_j+i  # add each salary to the sum_salary_j variable

print("")  # print an empty line for formatting purposes
for dep in department_list_s:  # loop through the department list for seniors
```

```
    if (dep not in probaility_dict_s):  # check if the department is not already in the probability dictionary
        probaility_dict_s[dep] = department_s[dep]/sum_department_s  # calculate the probability of the
department and add it to the dictionary

  for dep in department_list_j:  # loop through the department list for juniors
    if (dep not in probaility_dict_j):  # check if the department is not already in the probability dictionary
        probaility_dict_j[dep] = department_j[dep]/sum_department_j  # calculate the probability of the
department and add it to the dictionary

  for dep in age_list_s:  # loop through the age list for seniors
    if (dep not in probaility_dict_s):  # check if the age group is not already in the probability dictionary
        probaility_dict_s[dep] = age_s[dep]/sum_age_s  # calculate the probability of the age group and add it
to the dictionary

  for dep in age_list_j:  # loop through the age list for juniors
    if (dep not in probaility_dict_j):  # check if the age group is not already in the probability dictionary
        probaility_dict_j[dep] = age_j[dep]/sum_age_j  # calculate the probability of the age group and add it to
the dictionary

  for dep in salary_list_s:  # loop through the salary list for seniors
    if (dep not in probaility_dict_s):  # check if the salary range is not already in the probability dictionary
        probaility_dict_s[dep] = salary_s[dep]/sum_salary_s  # calculate the probability of the salary range and
add it to the dictionary

  for dep in salary_list_j:  # loop through the salary list for juniors
    if (dep not in probaility_dict_j):  # check if the salary range is not already in the probability dictionary
        probaility_dict_j[dep] = salary_j[dep]/sum_salary_j  # calculate the probability of the salary range and
add it to the dictionary

  probab_senior=5/11  # set the prior probability of a senior employee
  probab_junior=1-probab_senior  # set the prior probability of a junior employee


#-----------------------------------------------------------------------------------------
  print("Part A")
  #Part (A)
  #P(S|M,31...35,46k...50k)
  probability_senior=
probaility_dict_s.get("marketing",1)*probaility_dict_s.get("31...35",1)*probaility_dict_s.get("66k...70k",1)*prob
ab_senior
  #P(J|M,31...35,46k...50k)
  probability_junior=
probaility_dict_j.get("marketing",1)*probaility_dict_j.get("31...35",1)*probaility_dict_j.get("66k...70k",1)*proba
b_junior

  if probability_senior>probability_junior:
    print("The Status is Senior, as probability of senior is greater for the applied conditions.")
    print("")
  else:
    print("The Status is Junior, as probability of junior is greater for the applied conditions.")
    print("")
#-----------------------------------------------------------------------------------------
#-----------------------------------------------------------------------------------------
  print("Part B")
  #Part (B)
  #P(S|S,31...35,66k...70k)
```

```
   probability_senior=
probaility_dict_s.get("sales",1)*probaility_dict_s.get("31...35",1)*probaility_dict_s.get("46k...50k",1)*probab_s
enior
   #P(J|S,31...35,66k...70k)
   probability_junior=
probaility_dict_j.get("marketing",1)*probaility_dict_j.get("31...35",1)*probaility_dict_j.get("46k...50k",1)*proba
b_junior

   if probability_senior>probability_junior:
      print("The Status is Senior, as probability of senior is greater for the applied conditions.")
      print("")
   else:
      print("The Status is Junior, as probability of junior is greater for the applied conditions.")
      print("")
#-------------------------------------------------------------------------------------------
```

Screenshot



Findings

- **For Part A**, the code calculates the probability of the status being senior given the conditions **M, 31...35, 46k...50k**. It calculates the probability of the status being junior using the same conditions and compares the probabilities to determine which status is more likely.
- **For Part B**, the code calculates the probability of the status being senior given the conditions **S, 31...35, 66k...70k**. It calculates the probability of the status being junior using the same conditions and compares the probabilities to determine which status is more likely.
- **Overall**, the code seems to be a basic implementation of a probabilistic model that predicts the status of employees based on their department, age, and salary. However, the implementation of the addDataSet function and the definition of the probabilities could not be seen in the given code snippet.