Name: Harshal Jaywant Chavan
Class: FYMCA
Division: A
Roll No. 202124
Subject: AIML

Marks: 30 Marks

General Instructions:

- A practical consists of question **of 30 marks.**
- **Viva will be taken at the time of practical as well as after the practical if required.**
- The figures to the right indicate full marks.
- Create a folder with name of your seat Number in the folder "MCA_SEM_II_2021" on the desktop.
- You are allowed to use help files / documentation of the software/language that you are using.
- If you are using any additional information, state it clearly.
- Once you finish with the code show it to the examiner for testing.

Q.1	Implement Ada Boost algorithm and deploy it using Flask Library.	20
Q.2	Give Rules as predicate expressions for the following: 1. Rani is happy if she sings. 2. Jack and Jill are friends if both of them love to play football. 3. Pooja and Reema are friends if they like each other. 4. Tom and Ben are enemies if they don't like each other. 5. Ben is an uncle of George if Tom is a parent of George and Ben is a brother of Tom.	10

Q2.

=>

Filename: prac-exam.pl

```
likes(pooja,reema).

likes(reema,pooja).

likes(akshata,vighnesh).

likes(vighnesh,rhutika).

likes(tom,vishal).

likes(ben,anagha).

hobby(rani,singing).

hobby(ashish,football).

hobby(yogesh,driving).

activity(singing,rani).

loves(jack,football).

loves(jill,football).
```

brother(ben,tom).

brother(tom,ben).

parent(tom,george).

uncle(X,Y):- parent(tom,Y);brother(X,tom).

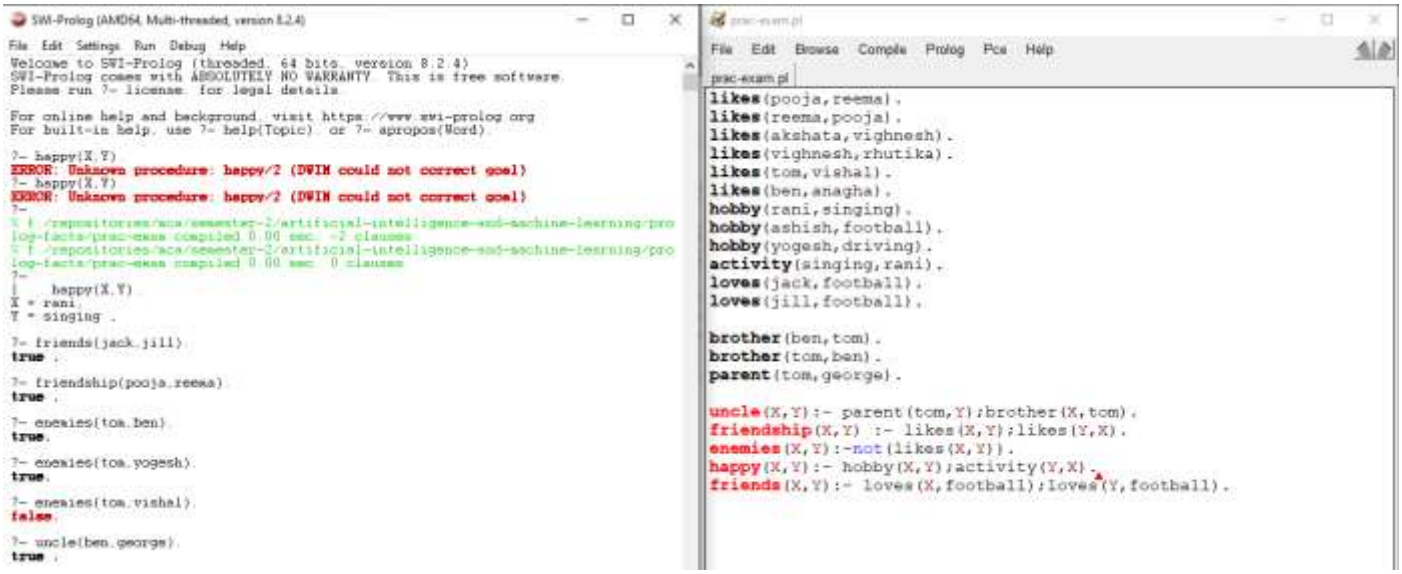
friendship(X,Y) :- likes(X,Y);likes(Y,X).

enemies(X,Y):-not(likes(X,Y)).

happy(X,Y):- hobby(X,Y);activity(Y,X).

friends(X,Y):- loves(X,football);loves(Y,football).

Screenshots:



Q1

=>

import pandas as pd

ds = pd.read_csv(r'addsdataset.csv')

X = ds.iloc[:,[2,3]].values

y = ds.iloc[:, 4].values

from sklearn.model_selection import train_test_split

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size = 0.25, random_state = 0, shuffle = False)

```
from sklearn.preprocessing import StandardScaler
sd = StandardScaler()
X_train = sd.fit_transform(X_train)
X_test = sd.transform(X_test)

from sklearn.ensemble import AdaBoostClassifier
classifier = AdaBoostClassifier()
classifier.fit(X_train, y_train)
y_pred = classifier.predict(X_test)
y_test


y_pred


from sklearn.metrics import confusion_matrix, accuracy_score
ac = accuracy_score(y_test, y_pred)
ac

from sklearn.ensemble import RandomForestClassifier
RF = RandomForestClassifier(max_depth = 2, random_state = 0)
classifierNew = AdaBoostClassifier(base_estimator = RF, n_estimators = 100, learning_rate = 0.01,
random_state = 0)
classifierNew.fit(X_train, y_train)



y_pred = classifierNew.predict(X_test)
ac = accuracy_score(y_test, y_pred)

with open('model.pkl','wb') as file:
   pickle.dump(classifier, file)
with open('modelNew.pkl','wb') as file:
   pickle.dump(classifierNew, file)
```

```
import flask
from flask import Flask, request
import pickle

model_adaboost = pickle.load(open('modelNew.pkl', 'rb'))

app = Flask(__name__)
@app.route('/', methods = ['GET', 'POST'])
def main():
   return "Ada boost with flask"

@app.route('/classify', methods = ['GET'])
def classify():
   if flask.request.method == 'GET':
      Age = request.args.get('age') # we will call the data from API using Postman
      EstimatedSalary = request.args.get('salary')
      prediction = model_adaboost.predict([[Age, EstimatedSalary]])
      print(prediction)

      if prediction == 1:
         return "there is a chance to purchase things"
      else:
         return "sorry, no chance"
   else:
      return "Select GET method"

if __name__ == '__main__':
   app.run()
```

Screenshots:

In [14]:
```
import pandas as pd
ds = pd.read_csv(r'addsdataset.csv')
```

In [15]:
```
ds.head()
```

Out[15]:

	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

In [16]:
```
X = ds.iloc[:,[2,3]].values
y = ds.iloc[:, 4].values
```

In [17]:
```
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size = 0.25, random_state = 0, shuffle = False)
```

In [18]:
```
from sklearn.ensemble import AdaBoostClassifier
classifier = AdaBoostClassifier()
classifier.fit(X_train,y_train)
```

Out[18]:
```
AdaBoostClassifier()
```

In [19]:
```
from sklearn.ensemble import AdaBoostClassifier
classifier = AdaBoostClassifier()
classifier.fit(X_train, y_train)
y_pred = classifier.predict(X_test)
y_test
```

Out[19]:
```
array([1, 1, 1, 1, 0, 0, 0, 1, 1, 0, 0, 1, 0, 1, 0, 1, 1, 0, 1, 0, 1, 1,
       0, 1, 1, 0, 0, 0, 1, 1, 0, 1, 0, 0, 1, 0, 1, 0, 0, 1, 1, 0, 0, 1,
       1, 0, 1, 1, 0, 0, 1, 0, 1, 0, 1, 1, 1, 0, 1, 0, 1, 1, 1, 0, 1, 1,
       1, 1, 0, 1, 1, 1, 0, 1, 0, 1, 0, 0, 1, 1, 0, 1, 1, 1, 1, 1, 1, 0,
       1, 1, 1, 1, 1, 1, 0, 1, 1, 1, 0, 1], dtype=int64)
```

In [20]:
```
y_pred
```

Out[20]:
```
array([1, 1, 1, 0, 0, 0, 1, 1, 1, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 1, 1,
       0, 1, 1, 0, 0, 0, 1, 1, 0, 1, 0, 0, 0, 0, 1, 0, 0, 1, 1, 0, 0, 0,
       1, 0, 1, 1, 0, 0, 0, 0, 1, 0, 1, 1, 1, 0, 0, 0, 1, 1, 0, 0, 1, 1,
       0, 0, 0, 1, 1, 1, 0, 1, 0, 0, 0, 0, 0, 1, 0, 1, 1, 1, 1, 0, 1, 0,
       1, 1, 1, 1, 0, 1, 0, 0, 1, 1, 0, 1], dtype=int64)
```

In [21]:
```
from sklearn.metrics import confusion_matrix, accuracy_score
ac = accuracy_score(y_test, y_pred)
ac
```

Out[22]:
```
AdaBoostClassifier(base_estimator=RandomForestClassifier(max_depth=2,
                                                         random_state=0),
                   learning_rate=0.01, n_estimators=100, random_state=0)
```

In [23]:
```
y_pred = classifierNew.predict(X_test)
ac = accuracy_score(y_test, y_pred)
```

In [25]:
```
import pickle
with open('model.pkl','wb') as file:
    pickle.dump(classifier, file)
with open('modelNew.pkl','wb') as file:
    pickle.dump(classifierNew, file)
```

In [26]:
```
import flask
from flask import Flask, request
```

In [27]:
```
model_adaboost = pickle.load(open('modelNew.pkl', 'rb'))
```

In [28]:
```
app = Flask(__name__)
```

In [29]:
```
@app.route('/', methods = ['GET', 'POST'])
def main():
```

In [14]:
```
import pandas as pd
ds = pd.read_csv(r'addsdataset.csv')
```

In [15]:
```
ds.head()
```

Out[15]:

	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

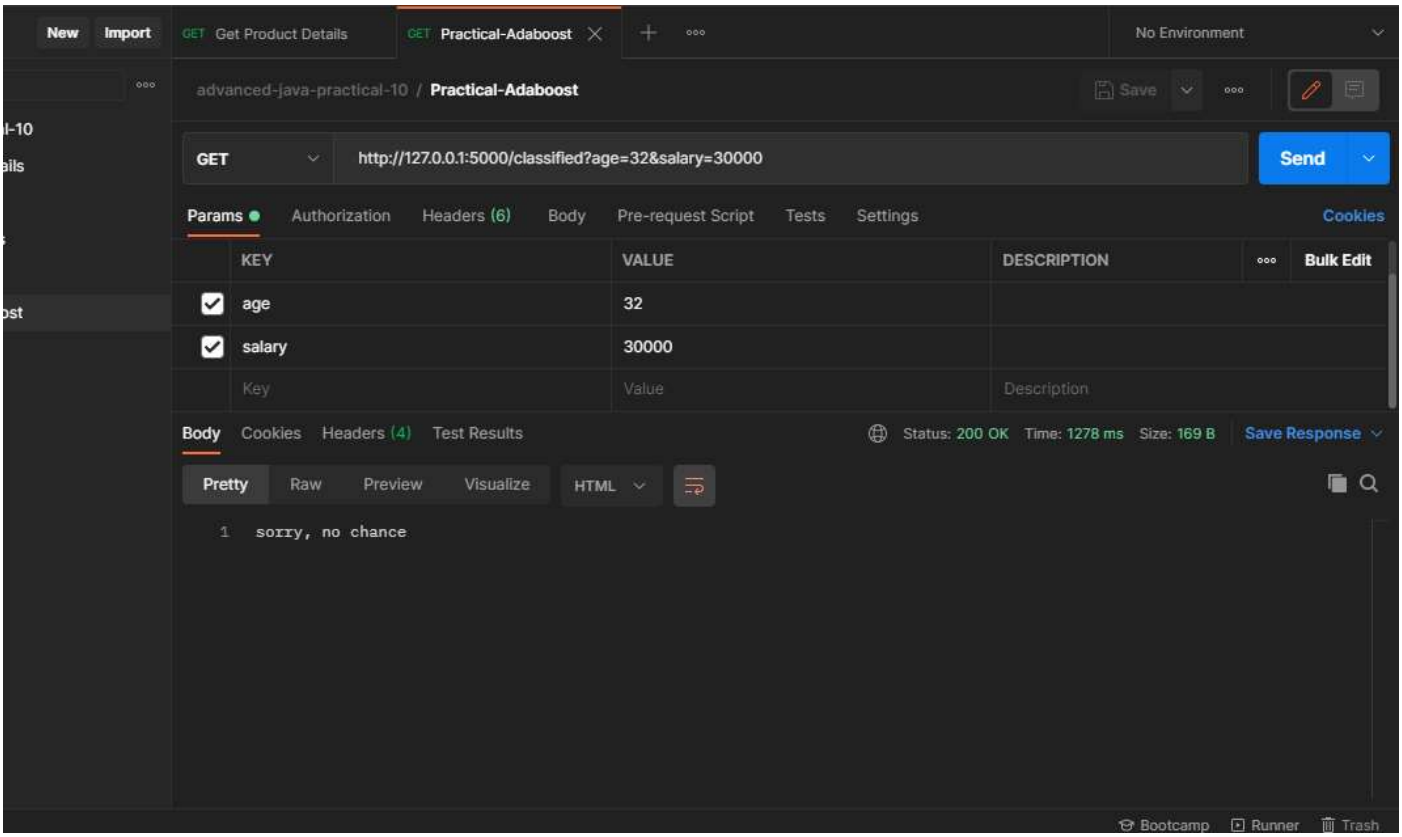
In [16]:
```
X = ds.iloc[:,[2,3]].values
y = ds.iloc[:, 4].values
```

In [17]:
```
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size = 0.25, random_state = 0, shuffle = False)
```

In [*]:
```
@app.route('/classified', methods = ['GET'])
def classified():
    if flask.request.method == 'GET':
        Age = request.args.get('age') # we will call the data from API using Postman
        EstimatedSalary = request.args.get('salary')
        prediction = model_adaboost.predict([[Age, EstimatedSalary]])
        print(prediction)
        if prediction == 1:
            return "there is a chance to purchase things"
        else:
            return "sorry, no chance"
    else:
        return "Select GET method"
if __name__ == '__main__':
    app.run()
```

```
 * Serving Flask app "__main__" (lazy loading)
 * Environment: production
   WARNING: This is a development server. Do not use it in a production deployment.
   Use a production WSGI server instead.
 * Debug mode: off
```

Output:


New
Import
GET Get Product Details
GET Practical-Adaboost
No Environment
advanced-java-practical-10 / Practical-Adaboost
Save
GET
http://127.0.0.1:5000/classified?age=32&salary=30000
Send
Params
Authorization
Headers (6)
Body
Pre-request Script
Tests
Settings
Cookies
KEY
VALUE
DESCRIPTION
Bulk Edit
age
32
salary
30000
Key
Value
Description
Body
Cookies
Headers (4)
Test Results
Status: 200 OK
Time: 1278 ms
Size: 169 B
Save Response
Pretty
Raw
Preview
Visualize
HTML
1 sorry, no chance
Bootcamp
Runner
Trash