University of Information Technology and Sciences (UITS)

Department of Information Technology

Lab Report No.: 5

IT-452 : Machine Learning

# Random Forest and ANN implementation

*Submitted To:*

Sumaiya Akhtar Mitu
Lecturer,
Department of IT ,
UITS
Email:sumaiya.akhtar@uits.edu.bd

*Submitted By:*

Name:Nazmul Zaman
Student ID:2014755055
Department of IT, UITS

10 June 2023 Department of IT, UITS © All rights reserved.

# Contents

# 1 Abstract

In this lab report I learn what is Random Forest and ANN algorithm in Machine Learning and learn deep things of both algorithm and how to implementation using different data set .

# 2 Random Forest

Random forest is a commonly-used machine learning algorithm trademarked by Leo Breiman and Adele Cutler, which combines the output of multiple decision trees to reach a single result. Its ease of use and flexibility have fueled its adoption, as it handles both classification and regression problems.

## 2.1 Why we use Random Forest

It can perform both regression and classification tasks. A random forest produces good predictions that can be understood easily. It can handle large datasets efficiently. The random forest algorithm provides a higher level of accuracy in predicting outcomes over the decision tree algorithm

## 2.2 Working Procedure of Random Forest

Step 1: Select random samples from a given data or training set.
Step 2: This algorithm will construct a decision tree for every training data.
Step 3: Voting will take place by averaging the decision tree.
Step 4: Finally, select the most voted prediction result as the final prediction result.

## 2.3 Applications of Random Forest

1.Predicting customer behavior
2.Consumer demand or stock price fluctuations
3.Identifying fraud
4.Diagnosing patients

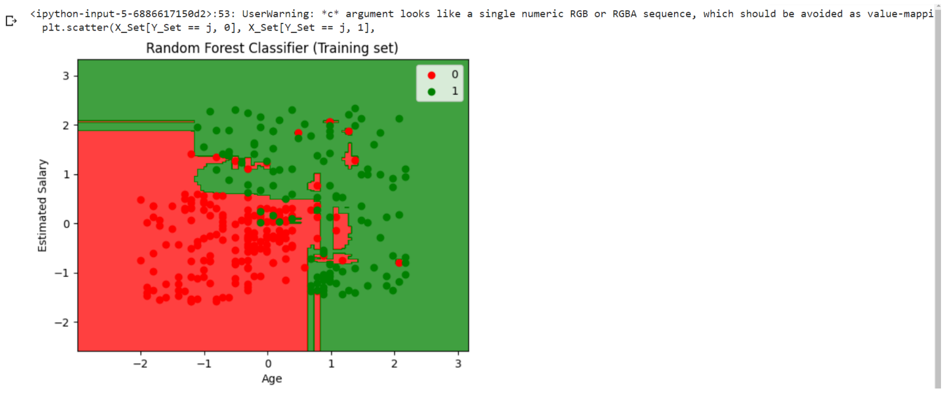
## 2.4 Code For Random Forest

```
# -*- coding: utf-8 -*-
"""labfinal.ipynb

Automatically generated by Colaboratory.

Original file is located at
    https://colab.research.google.com/drive/1T-kWntc5At55VrQZqNiJfZ6moW9c67D2
"""

#nazmul zaman

# Importing the libraries

import numpy as np
import matplotlib.pyplot as plt
import pandas as pd

# Importing the datasets

datasets = pd.read_csv('https://raw.githubusercontent.com/mahesh147/Random-Forest-Classifier/master/Social_Network_Ads.csv')
X = datasets.iloc[:, [2,3]].values
Y = datasets.iloc[:, 4].values


# Splitting the dataset into the Training set and Test set

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)

# Feature Scaling

from sklearn.preprocessing import StandardScaler
sc_X = StandardScaler()
X_Train = sc_X.fit_transform(X_Train)
X_Test = sc_X.transform(X_Test)

# Fitting the classifier into the Training set

from sklearn.ensemble import RandomForestClassifier
classifier = RandomForestClassifier(n_estimators = 200, criterion = 'entropy', random_state = 0)
classifier.fit(X_Train,Y_Train)

# Predicting the test set results

Y_Pred = classifier.predict(X_Test)

# Making the Confusion Matrix
```

```python

from sklearn.metrics import confusion_matrix
cm = confusion_matrix(Y_Test, Y_Pred)

# Showing  the Training set results

from matplotlib.colors import ListedColormap
X_Set, Y_Set = X_Train, Y_Train
X1, X2 = np.meshgrid(np.arange(start = X_Set[:, 0].min() - 1, stop = X_Set[:, 0].max() + 1, step = 0.01),
                          np.arange(start = X_Set[:, 1].min() - 1, stop = X_Set[:, 1].max() + 1, step = 0.01))
plt.contourf(X1, X2, classifier.predict(np.array([X1.ravel(), X2.ravel()]).T).reshape(X1.shape),
                alpha = 0.75, cmap = ListedColormap(('red', 'green')))
plt.xlim(X1.min(), X1.max())
plt.ylim(X2.min(), X2.max())
for i, j in enumerate(np.unique(Y_Set)):
     plt.scatter(X_Set[Y_Set == j, 0], X_Set[Y_Set == j, 1],
                    c = ListedColormap(('red', 'green'))(i), label = j)
plt.title('Random Forest Classifier (Training set)')
plt.xlabel('Age')
plt.ylabel('Estimated Salary')
plt.legend()
plt.show()

# Show Test set results

from matplotlib.colors import ListedColormap
X_Set, Y_Set = X_Test, Y_Test
X1, X2 = np.meshgrid(np.arange(start = X_Set[:, 0].min() - 1, stop = X_Set[:, 0].max() + 1, step = 0.01),
                          np.arange(start = X_Set[:, 1].min() - 1, stop = X_Set[:, 1].max() + 1, step = 0.01))
plt.contourf(X1, X2, classifier.predict(np.array([X1.ravel(), X2.ravel()]).T).reshape(X1.shape),
                alpha = 0.75, cmap = ListedColormap(('red', 'green')))
plt.xlim(X1.min(), X1.max())
plt.ylim(X2.min(), X2.max())
for i, j in enumerate(np.unique(Y_Set)):
     plt.scatter(X_Set[Y_Set == j, 0], X_Set[Y_Set == j, 1],
                    c = ListedColormap(('red', 'green'))(i), label = j)
plt.title('Random Forest Classifier (Test set)')
plt.xlabel('Age')
plt.ylabel('Estimated Salary')
plt.legend()
plt.show()
```

Figure 1:

## 2.5 Output



```
<ipython-input-5-6886617150d2>:53: UserWarning: *c* argument looks like a single numeric RGB or RGBA sequence, which should be avoided as value-mappi
  plt.scatter(X_Set[Y_Set == j, 0], X_Set[Y_Set == j, 1],
```

Figure 2:

## 2.6 Output

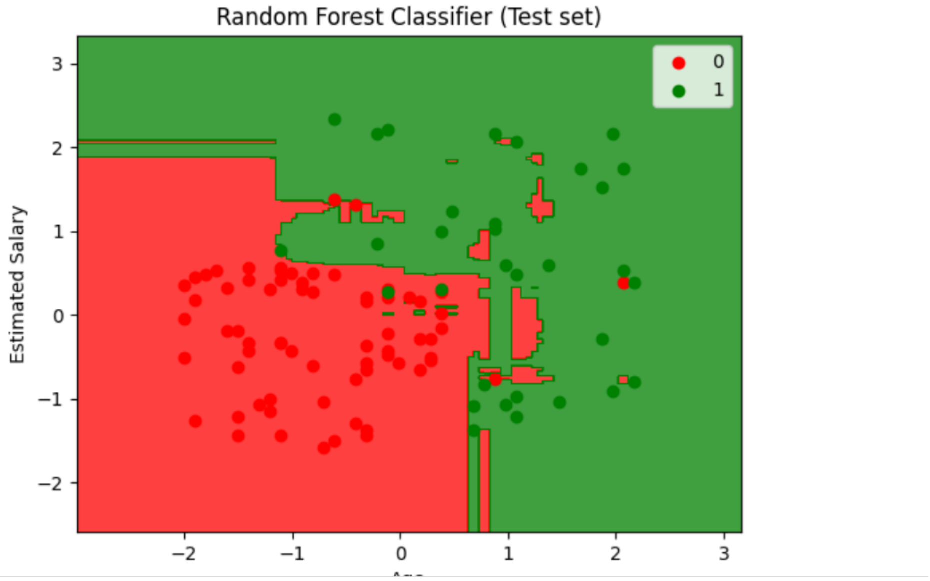


Figure 3:

# 3 ANN

Neural networks, also known as artificial neural networks (ANNs) or simulated neural networks (SNNs), are a subset of machine learning and are at the heart of deep learning algorithms. Their name and structure are inspired by the human brain, mimicking the way that biological neurons signal to one another.

## 3.1 Why we use ANN

Artificial neural networks are used for a range of applications, including image recognition, speech recognition, machine translation, and medical diagnosis. The fact that ANN learns from sample data sets is a significant advantage. The most typical application of ANN is for random function approximation.

## 3.2 Code For ANN

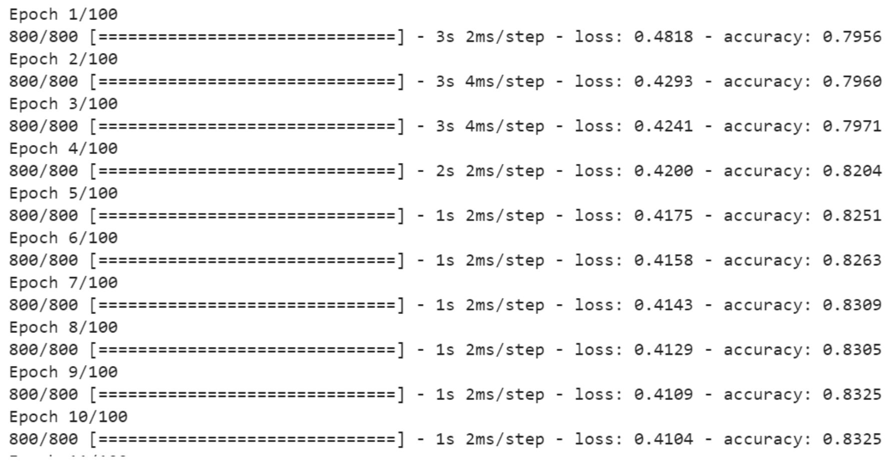
```python
# -*- coding: utf-8 -*-
"""ANN.ipynb

Automatically generated by Colaboratory.

Original file is located at
    https://colab.research.google.com/drive/187vyJh2rFGlgLYDIKm6SA8VQEGn3e4Zh
"""

# Importing the libraries
import numpy as np
import pandas as pd

# Importing the dataset
dataset = pd.read_csv('https://raw.githubusercontent.com/anwarshaikh078/Artificial-Neural-Network/master/Churn_Modelling.csv')
X = dataset.iloc[:, 3:13].values
y = dataset.iloc[:, 13].values

# Encoding categorical data
from sklearn.preprocessing import LabelEncoder, OneHotEncoder
from sklearn.compose import ColumnTransformer

labelencoder_X_1 = LabelEncoder()
X[:, 1] = labelencoder_X_1.fit_transform(X[:, 1])
labelencoder_X_2 = LabelEncoder()
X[:, 2] = labelencoder_X_2.fit_transform(X[:, 2])

ct = ColumnTransformer(
    [('one_hot_encoder', OneHotEncoder(categories='auto'), [1])],
    remainder='passthrough'
)
```

```python
X = np.array(ct.fit_transform(X), dtype=np.float)

# Avoiding the Dummy Variable Trap
X = X[:, 1:]

# Splitting the dataset into the Training set and Test set
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(
    X, y, test_size=0.2, random_state=0
)

# Feature Scaling
from sklearn.preprocessing import StandardScaler
sc = StandardScaler()
X_train = sc.fit_transform(X_train)
X_test = sc.transform(X_test)

# Importing Keras and packages
import keras
from keras.models import Sequential
from keras.layers import Dense

# Initializing the Artificial Neural Network
classifier = Sequential()

# Adding the input layer and first hidden layer
classifier.add(Dense(units=6, kernel_initializer='uniform',
   activation='relu', input_dim=11))

# Adding the second hidden layer
classifier.add(Dense(units=6, kernel_initializer='uniform',
   activation='relu'))

# Adding the output layer
classifier.add(Dense(units=1, kernel_initializer='uniform',
   activation='sigmoid'))

# Compiling the Artificial Neural Network
classifier.compile(optimizer='adam', loss='binary_crossentropy'
   , metrics=['accuracy'])

# Fitting the training set
classifier.fit(X_train, y_train, batch_size=10, epochs=100)

# Predicting the test set results
y_pred = classifier.predict(X_test)
y_pred = (y_pred > 0.5)

# New customer data
new_customer = [[0, 0, 600, 1, 40, 3, 60000, 2, 1, 1, 50000]]

# Feature scaling for the new customer data
new_customer = sc.transform(new_customer)
```

```

# Predicting the exit for the new customer
exit_prediction = classifier.predict(new_customer)
exit_prediction = (exit_prediction > 0.5)

# Printing the prediction
if exit_prediction:
    print("The customer is predicted to exit.")
else:
    print("The customer is predicted to stay.")

# Making the confusion matrix
from sklearn.metrics import confusion_matrix
cm = confusion_matrix(y_test, y_pred)
```

Figure 4:

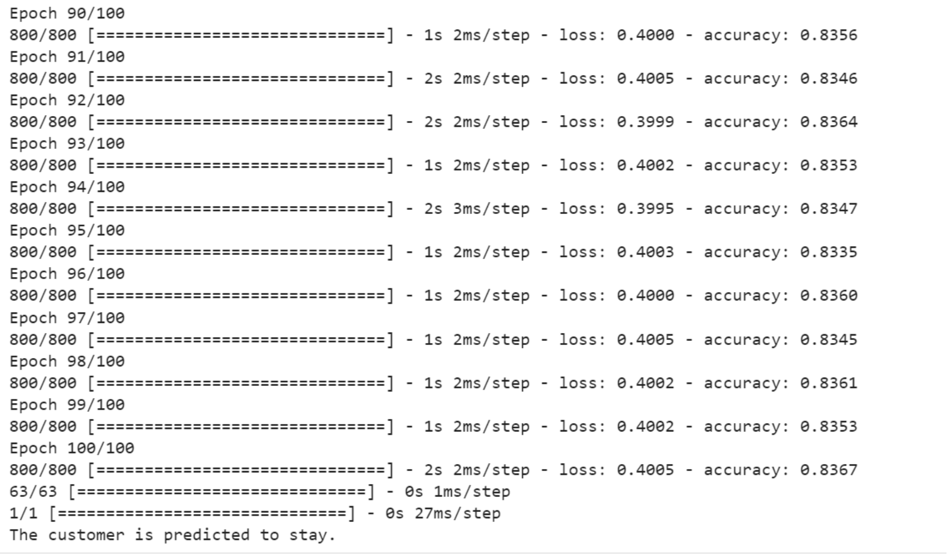
## 3.3 Output



```
Epoch 1/100
800/800 [==============================] - 3s 2ms/step - loss: 0.4818 - accuracy: 0.7956
Epoch 2/100
800/800 [==============================] - 3s 4ms/step - loss: 0.4293 - accuracy: 0.7960
Epoch 3/100
800/800 [==============================] - 3s 4ms/step - loss: 0.4241 - accuracy: 0.7971
Epoch 4/100
800/800 [==============================] - 2s 2ms/step - loss: 0.4200 - accuracy: 0.8204
Epoch 5/100
800/800 [==============================] - 1s 2ms/step - loss: 0.4175 - accuracy: 0.8251
Epoch 6/100
800/800 [==============================] - 1s 2ms/step - loss: 0.4158 - accuracy: 0.8263
Epoch 7/100
800/800 [==============================] - 1s 2ms/step - loss: 0.4143 - accuracy: 0.8309
Epoch 8/100
800/800 [==============================] - 1s 2ms/step - loss: 0.4129 - accuracy: 0.8305
Epoch 9/100
800/800 [==============================] - 1s 2ms/step - loss: 0.4109 - accuracy: 0.8325
Epoch 10/100
800/800 [==============================] - 1s 2ms/step - loss: 0.4104 - accuracy: 0.8325
```

Figure 5: Output shows 100 test cases accuracy and loss and It's too long so that I put here first 10 cases

## 3.4 Output



```
Epoch 90/100
800/800 [==============================] - 1s 2ms/step - loss: 0.4000 - accuracy: 0.8356
Epoch 91/100
800/800 [==============================] - 2s 2ms/step - loss: 0.4005 - accuracy: 0.8346
Epoch 92/100
800/800 [==============================] - 2s 2ms/step - loss: 0.3999 - accuracy: 0.8364
Epoch 93/100
800/800 [==============================] - 1s 2ms/step - loss: 0.4002 - accuracy: 0.8353
Epoch 94/100
800/800 [==============================] - 2s 3ms/step - loss: 0.3995 - accuracy: 0.8347
Epoch 95/100
800/800 [==============================] - 1s 2ms/step - loss: 0.4003 - accuracy: 0.8335
Epoch 96/100
800/800 [==============================] - 1s 2ms/step - loss: 0.4000 - accuracy: 0.8360
Epoch 97/100
800/800 [==============================] - 1s 2ms/step - loss: 0.4005 - accuracy: 0.8345
Epoch 98/100
800/800 [==============================] - 1s 2ms/step - loss: 0.4002 - accuracy: 0.8361
Epoch 99/100
800/800 [==============================] - 1s 2ms/step - loss: 0.4002 - accuracy: 0.8353
Epoch 100/100
800/800 [==============================] - 2s 2ms/step - loss: 0.4005 - accuracy: 0.8367
63/63 [==============================] - 0s 1ms/step
1/1 [==============================] - 0s 27ms/step
The customer is predicted to stay.
```

Figure 6: Output shows 100 test cases accuracy and loss It's too long so that I put here last 10 cases and final output

# 4 Conclusion

In this lab report I learn how to implement Random Forest and ANN algorithm and learning what is this and why we use those algorithm and working procedure and many things.

# 5 References

1.https://www.javatpoint.com/machine-learning-random-forest-algorithm 2.https://www.javatpoint.
neural-network