**Report On:** Implementation ANN, DNN & CNN on datasets

Course Title : Artificial Intelligence Sessional
Course Code : CSE 342
Report No : 02
Submitted to : Sourav Adhikary
Lecturer, Department of CSE
Port City International University

Submitted by: Asibur Rahman
Student ID: CSE01606557
Batch: CSE $16^{th}$
Program: B.SC in CSE
Date Of Submission:

Department of Computer Science & Engineering
Port City International University

# INDEX

## Program no: 01

## Program name: ANN implementation using python to an Income Classification dataset.

## Objective: My goal is to implement ANN model using python to an Income Classification dataset.

## Description:

ANN is a modeling technique inspired by the human nervous system that allows learning by example from representative data that describes a physical phenomenon or a decision process. A unique feature of ANN is that they are able to establish empirical relationships between independent and dependent variables, and extract subtle information and complex knowledge from representative data sets. The relationships between independent and dependent variables can be established without assumptions about any mathematical representation of the phenomena. ANN models provide certain advantages over regression-based models including its capacity to deal with noisy data.

# Source Code:

IncomeClassification(ANN).ipynb

```python
import pandas as pd
data = pd.read_csv("income_evaluation.csv")
```

```python
from google.colab import drive
drive.mount('/content/drive')
```

```python
data.head(15)
```

```python
data.describe()
```

```python
data.shape
```

```python
data[' marital-status'].value_counts()
```

```python
from sklearn import preprocessing
ncode = preprocessing.LabelEncoder()
data[' marital-status'] = ncode.fit_transform(data[' marital-status'])
```

```python
data[' marital-status'].value_counts()
```

```python
data[' workclass'].value_counts()
```

IncomeClassification(ANN).ipynb

```python
from sklearn import preprocessing
ncode = preprocessing.LabelEncoder()
data[' workclass'] = ncode.fit_transform(data[' workclass'])
```

```python
data[' workclass'].value_counts()
```

```python
data[' education'].value_counts()
```

```python
from sklearn import preprocessing
ncode = preprocessing.LabelEncoder()
data[' education'] = ncode.fit_transform(data[' education'])
```

```python
data[' education'].value_counts()
```

```python
data[' occupation'].value_counts()
```

```python
from sklearn import preprocessing
ncode = preprocessing.LabelEncoder()
data[' occupation'] = ncode.fit_transform(data[' occupation'])
```

```python
data[' occupation'].value_counts()
```

IncomeClassification(ANN).ipynb

File Edit View Insert Runtime Tools Help All changes saved

+ Code + Text

```python
data[' occupation'].value_counts()
```

```python
from sklearn import preprocessing
ncode = preprocessing.LabelEncoder()
data[' occupation'] = ncode.fit_transform(data[' occupation'])
```

```python
data[' occupation'].value_counts()
```

```python
data[' relationship'].value_counts()
```

```python
from sklearn import preprocessing
ncode = preprocessing.LabelEncoder()
data[' relationship'] = ncode.fit_transform(data[' relationship'])
```

```python
data[' relationship'].value_counts()
```

```python
data[' race'].value_counts()
```

```python
from sklearn import preprocessing
ncode = preprocessing.LabelEncoder()
data[' race'] = ncode.fit_transform(data[' race'])
```

IncomeClassification(ANN).ipynb

File Edit View Insert Runtime Tools Help All changes saved

+ Code + Text

```python
data[' race'].value_counts()
```

```python
data[' sex'].value_counts()
```

```python
from sklearn import preprocessing
ncode = preprocessing.LabelEncoder()
data[' sex'] = ncode.fit_transform(data[' sex'])
```

```python
data[' sex'].value_counts()
```

```python
data[' native-country'].value_counts()
```

```python
from sklearn import preprocessing
ncode = preprocessing.LabelEncoder()
data[' native-country'] = ncode.fit_transform(data[' native-country'])
```

```python
data[' native-country'].value_counts()
```

```python
data_y = data[' income'].copy()
data_x = data.drop([' income'],axis=1)
data_y.value_counts()
```

IncomeClassification(ANN).ipynb

File Edit View Insert Runtime Tools Help

+ Code + Text

```
data_y = ncode.fit_transform(data_y)
```

```
from keras.datasets import mnist
from keras.models import Sequential
from keras.layers import Dense
from keras.utils import np_utils
import numpy as np
```

```
def baseline_model():
  model = Sequential()
  model.add(Dense(252,activation="relu",input_dim=data_x.shape[1]))
  model.add(Dense(1, kernel_initializer='normal', activation='sigmoid'))
  model.compile(loss='binary_crossentropy', optimizer='adam', metrics=['accuracy'])
  return model
```

```
model = baseline_model()
```

```
model.summary()
```

```
model.fit(data_x,data_y,epochs =100, batch_size= 16, verbose=1)
```

```
col = data_x.columns
```

+ Code + Text

```
from sklearn.preprocessing import Normalizer
```

```
norm= Normalizer()
data_x=norm.fit_transform(data_x)
data_x=pd.DataFrame(data_x, columns=col)
```

```
from sklearn.model_selection import train_test_split
x_train,x_test,y_train,y_test=train_test_split(data_x,data_y,train_size=9/10)
```

```
model.fit(x_train,y_train,epochs=200,batch_size=64, verbose =1)
```

```
model.evaluate(x_test,y_test)
```

```
scores = model.evaluate(x_test, y_test, verbose=0)
```

```
print("accuracy",(scores[1]*100))
```

```
predictions = (model.predict(x_test)>0.5).astype("int32").ravel()
```

IncomeClassification(ANN).ipynb

File Edit View Insert Runtime Tools Help

+ Code + Text

```
from sklearn.metrics import accuracy_score
from sklearn.metrics import precision_score
from sklearn.metrics import recall_score
from sklearn.metrics import f1_score
precision = precision_score(y_test,predictions)
precision
```

```
recall = recall_score(y_test, predictions)
```

```
from sklearn.metrics import classification_report
print(classification_report(y_test, predictions))
```

```
from sklearn.metrics import confusion_matrix
from matplotlib import pyplot as plt
conf_matrix = confusion_matrix(y_true= y_test, y_pred=predictions)

fig, ax = plt.subplots(figsize=(7.5, 7.5))
ax.matshow(conf_matrix, cmap=plt.cm.Blues, alpha=0.3)
for i in range(conf_matrix.shape[0]):
    for j in range(conf_matrix.shape[1]):
        ax.text(x=j, y=i,s=conf_matrix[i, j], va='center', ha='center', size='xx-large')

plt.xlabel('Predictions', fontsize=18)
plt.ylabel('Actuals', fontsize=18)
plt.title('Confusion Matrix', fontsize=18)
plt.show()
```

## Dataset:

```
[60] data.head(15)
```

| | age | workclass | fnlwgt | education | education-num | marital-status | occupation | relationship | race | sex | capital-gain | capital-loss | hours-per-week | native-country | income |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 0 | 39 | State-gov | 77516 | Bachelors | 13 | Never-married | Adm-clerical | Not-in-family | White | Male | 2174 | 0 | 40 | United-States | <=50K |
| 1 | 50 | Self-emp-not-inc | 83311 | Bachelors | 13 | Married-civ-spouse | Exec-managerial | Husband | White | Male | 0 | 0 | 13 | United-States | <=50K |
| 2 | 38 | Private | 215646 | HS-grad | 9 | Divorced | Handlers-cleaners | Not-in-family | White | Male | 0 | 0 | 40 | United-States | <=50K |
| 3 | 53 | Private | 234721 | 11th | 7 | Married-civ-spouse | Handlers-cleaners | Husband | Black | Male | 0 | 0 | 40 | United-States | <=50K |
| 4 | 28 | Private | 338409 | Bachelors | 13 | Married-civ-spouse | Prof-specialty | Wife | Black | Female | 0 | 0 | 40 | Cuba | <=50K |
| 5 | 37 | Private | 284582 | Masters | 14 | Married-civ-spouse | Exec-managerial | Wife | White | Female | 0 | 0 | 40 | United-States | <=50K |
| 6 | 49 | Private | 160187 | 9th | 5 | Married-spouse-absent | Other-service | Not-in-family | Black | Female | 0 | 0 | 16 | Jamaica | <=50K |
| 7 | 52 | Self-emp-not-inc | 209642 | HS-grad | 9 | Married-civ-spouse | Exec-managerial | Husband | White | Male | 0 | 0 | 45 | United-States | >50K |

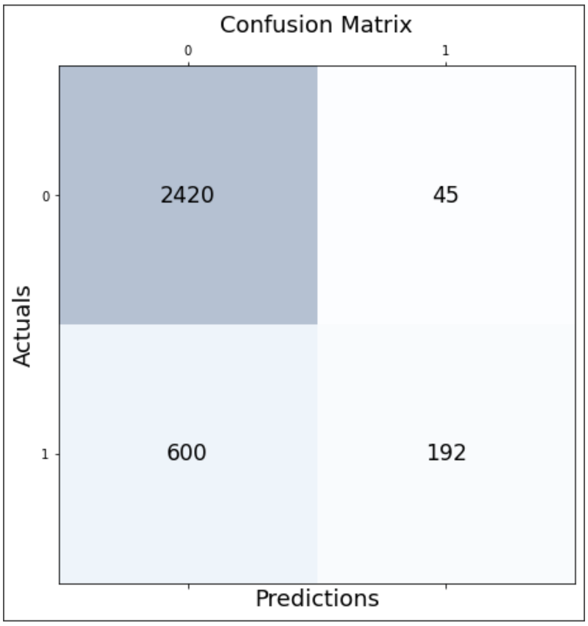
## Output:

```
[101] print("accuracy",(scores[1]*100))

accuracy 80.19649982452393
```

## Classification Report:

```
[106] from sklearn.metrics import classification_report
      print(classification_report(y_test, predictions))

              precision    recall  f1-score   support

           0       0.80      0.98      0.88      2465
           1       0.81      0.24      0.37       792

    accuracy                           0.80      3257
   macro avg       0.81      0.61      0.63      3257
weighted avg       0.80      0.80      0.76      3257
```

## Confusion Matrix:




## Discussion:

ANNs have been utilized to speed up dependability investigation of foundations dependent upon cataclysmic events and to foresee establishment settlements. ANNs have additionally been utilized for building black-box models in geoscience: hydrology, sea displaying and seaside designing, and geomorphology. ANNs have been utilized in online protection, with the target to segregate between authentic exercises and malevolent ones. For instance, AI has been utilized for characterizing Android malware, for recognizing areas having a place with danger entertainers and for identifying URLs representing a security risk. Research is in progress on ANN frameworks intended for entrance testing, for recognizing botnets, Mastercards cheats and organization interruptions.

## Program no: 02

**Program name:** CNN implementation using python to an Income Classification dataset.

**Objective:** My goal is to implement CNN model using python to an Income Classification dataset.

## Description:

CNN is a particular type of feed-forward neural network in AI. It is widely used for image recognition. CNN represents the input data in the form of multidimensional arrays. It works well for a large number of labeled data. CNN extract the each and every portion of input image, which is known as receptive field. It assigns weights for each neuron based on the significant role of the receptive field. So that it can discriminate the importance of neurons from one another. classification. Deep Learning thus recognizes objects in an image by using a CNN. CNNs are playing a major role in diverse tasks/functions like image processing problems, computer vision tasks like localization and segmentation, video analysis, to recognize obstacles in self-driving cars, as well as speech recognition in natural language processing.

## Source Code:

IncomeClassification(CNN).ipynb

```python
import pandas as pd
data = pd.read_csv("income_evaluation.csv")
```

```python
from google.colab import drive
drive.mount('/content/drive')
```

```python
data.head(15)
```

```python
data.describe()
```

```python
data.shape
```

```python
data[' marital-status'].value_counts()
```

```python
from sklearn import preprocessing
ncode = preprocessing.LabelEncoder()
data[' marital-status'] = ncode.fit_transform(data[' marital-status'])
```

```python
data[' marital-status'].value_counts()
```

```python
data[' workclass'].value_counts()
```

IncomeClassification(CNN).ipynb

```python
from sklearn import preprocessing
ncode = preprocessing.LabelEncoder()
data[' workclass'] = ncode.fit_transform(data[' workclass'])
```

```python
data[' workclass'].value_counts()
```

```python
data[' education'].value_counts()
```

```python
from sklearn import preprocessing
ncode = preprocessing.LabelEncoder()
data[' education'] = ncode.fit_transform(data[' education'])
```

```python
data[' education'].value_counts()
```

```python
data[' occupation'].value_counts()
```

```python
from sklearn import preprocessing
ncode = preprocessing.LabelEncoder()
data[' occupation'] = ncode.fit_transform(data[' occupation'])
```

```python
data[' occupation'].value_counts()
```

```python
data[' relationship'].value_counts()
```

IncomeClassification(CNN).ipynb

File Edit View Insert Runtime Tools Help All changes saved

\+ Code + Text

```
from sklearn import preprocessing
ncode = preprocessing.LabelEncoder()
data[' relationship'] = ncode.fit_transform(data[' relationship'])
```

```
data[' relationship'].value_counts()
```

```
data[' race'].value_counts()
```

```
from sklearn import preprocessing
ncode = preprocessing.LabelEncoder()
data[' race'] = ncode.fit_transform(data[' race'])
```

```
data[' race'].value_counts()
```

```
data[' sex'].value_counts()
```

```
from sklearn import preprocessing
ncode = preprocessing.LabelEncoder()
data[' sex'] = ncode.fit_transform(data[' sex'])
```

```
data[' sex'].value_counts()
```

```
data[' native-country'].value_counts()
```

IncomeClassification(CNN).ipynb

File Edit View Insert Runtime Tools Help All changes saved

\+ Code + Text

```
from sklearn import preprocessing
ncode = preprocessing.LabelEncoder()
data[' native-country'] = ncode.fit_transform(data[' native-country'])
```

```
data[' native-country'].value_counts()
```

```
data_y = data[' income'].copy()
data_x = data.drop([' income'],axis=1)
data_y.value_counts()
```

```
data_y = ncode.fit_transform(data_y)
```

```
from sklearn.model_selection import train_test_split
x_train,x_test,y_train,y_test=train_test_split(data_x,data_y,train_size=9/10)
```

```
from keras.datasets import mnist
from keras.models import Sequential
from keras.layers import Dense,Dropout, Activation, Flatten,Convolution1D, MaxPooling1D
from keras.utils import np_utils
import numpy as np
```

```python
def baseline_model():
    model = Sequential()
    model.add(Convolution1D(8,3, activation="relu", input_shape= (x_train.shape[1],1)))
    model.add(MaxPooling1D(pool_size=2))
    model.add(Flatten())
    model.add(Dense(64, activation ='relu'))
    model.add(Dense(1, activation='sigmoid'))
    model.compile(loss='binary_crossentropy',optimizer="adam",metrics=['accuracy'])
    return model
```

```python
model = baseline_model()
```

```python
model.summary()
```

```python
model.fit(data_x,data_y,epochs =100, batch_size= 16)
```

```python
col = data_x.columns
```

```python
model.evaluate(x_test,y_test)
```

```python
scores = model.evaluate(x_test, y_test, verbose=0)
```

```python
print("accuracy",(scores[1]*100))
```

IncomeClassification(CNN).ipynb

File Edit View Insert Runtime Tools Help

+ Code + Text

```python
from sklearn.metrics import f1_score
precision = precision_score(y_test,predictions)
precision
```

```python
recall = recall_score(y_test, predictions)
```

```python
f1 = f1_score(y_test,predictions)
f1
```

```python
from sklearn.metrics import classification_report
print(classification_report(y_test, predictions))
```

```python
from sklearn.metrics import confusion_matrix
from matplotlib import pyplot as plt
conf_matrix = confusion_matrix(y_true= y_test, y_pred=predictions)

fig, ax = plt.subplots(figsize=(7.5, 7.5))
ax.matshow(conf_matrix, cmap=plt.cm.Blues, alpha=0.3)
for i in range(conf_matrix.shape[0]):
    for j in range(conf_matrix.shape[1]):
        ax.text(x=j, y=i,s=conf_matrix[i, j], va='center', ha='center', size='xx-large')

plt.xlabel('Predictions', fontsize=18)
plt.ylabel('Actuals', fontsize=18)
plt.title('Confusion Matrix', fontsize=18)
plt.show()
```

# Dataset:

```
[4] data.head(15)
```

| | age | workclass | fnlwgt | education | education-num | marital-status | occupation | relationship | race | sex | capital-gain | capital-loss | hours-per-week | native-country | income |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 0 | 39 | State-gov | 77516 | Bachelors | 13 | Never-married | Adm-clerical | Not-in-family | White | Male | 2174 | 0 | 40 | United-States | <=50K |
| 1 | 50 | Self-emp-not-inc | 83311 | Bachelors | 13 | Married-civ-spouse | Exec-managerial | Husband | White | Male | 0 | 0 | 13 | United-States | <=50K |
| 2 | 38 | Private | 215646 | HS-grad | 9 | Divorced | Handlers-cleaners | Not-in-family | White | Male | 0 | 0 | 40 | United-States | <=50K |
| 3 | 53 | Private | 234721 | 11th | 7 | Married-civ-spouse | Handlers-cleaners | Husband | Black | Male | 0 | 0 | 40 | United-States | <=50K |
| 4 | 28 | Private | 338409 | Bachelors | 13 | Married-civ-spouse | Prof-specialty | Wife | Black | Female | 0 | 0 | 40 | Cuba | <=50K |
| 5 | 37 | Private | 284582 | Masters | 14 | Married-civ-spouse | Exec-managerial | Wife | White | Female | 0 | 0 | 40 | United-States | <=50K |
| 6 | 49 | Private | 160187 | 9th | 5 | Married-spouse-absent | Other-service | Not-in-family | Black | Female | 0 | 0 | 16 | Jamaica | <=50K |
| 7 | 52 | Self-emp-not-inc | 209642 | HS-grad | 9 | Married-civ-spouse | Exec-managerial | Husband | White | Male | 0 | 0 | 45 | United-States | >50K |

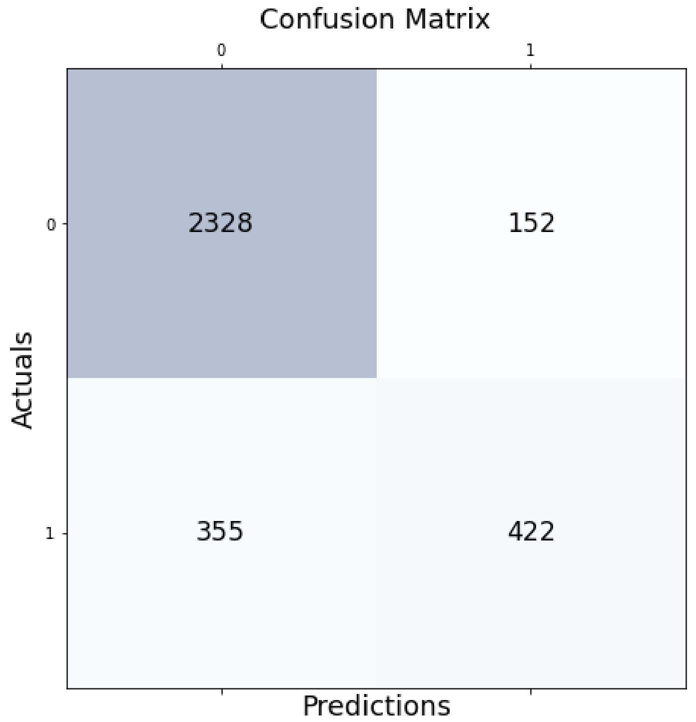
# Output:

```
[42] print("accuracy",(scores[1]*100))

     accuracy 84.43352580070496
```

# Classification Report:

```
[47] from sklearn.metrics import classification_report
     print(classification_report(y_test, predictions))

                   precision    recall  f1-score   support

               0       0.87      0.94      0.90      2480
               1       0.74      0.54      0.62       777

        accuracy                           0.84      3257
       macro avg       0.80      0.74      0.76      3257
    weighted avg       0.84      0.84      0.84      3257
```

## Confusion Matrix:




## Discussion:

In deep learning, a convolutional neural network (CNN/ConvNet) is a class of deep neural networks, most commonly applied to analyze visual imagery. Now when we think of a neural network, we think about matrix multiplications but that is not the case with ConvNet. It uses a special technique called Convolution. Now in mathematics convolution is a mathematical operation on two functions that produces a third function that expresses how the shape of one is modified by the other.

## Program no: 03

**Program name:** CNN implementation using python to an Image dataset.

**Objective:** My goal is to implement CNN model using python to an Image dataset.

## Description:

A Convolutional Neural Network, also known as CNN or ConvNet, is a class of neural networks that specializes in processing data that has a grid-like topology, such as an image. A digital image is a binary representation of visual data. The CIFAR10 dataset contains 60,000 color images in 10 classes, with 6,000 images in each class. The dataset is divided into 50,000 training images and 10,000 testing images. The classes are mutually exclusive and there is no overlap between them.

# Source Code:

Image Classification CNN

File Edit View Insert Runtime Tools Help

\+ Code + Text

```
[11] import tensorflow as tf

     from tensorflow.keras import datasets, layers, models
     import matplotlib.pyplot as plt
```

```
[12] (train_images, train_labels), (test_images, test_labels) = datasets.cifar10.load_data()

     # Normalize pixel values to be between 0 and 1
     train_images, test_images = train_images / 255.0, test_images / 255.0
```

```
class_names = ['airplane', 'automobile', 'bird', 'cat', 'deer',
               'dog', 'frog', 'horse', 'ship', 'truck']

plt.figure(figsize=(10,10))
for i in range(25):
    plt.subplot(5,5,i+1)
    plt.xticks([])
    plt.yticks([])
    plt.grid(False)
    plt.imshow(train_images[i])
    # The CIFAR labels happen to be arrays,
    # which is why you need the extra index
    plt.xlabel(class_names[train_labels[i][0]])
plt.show()
```

```
[14] model = models.Sequential()
     model.add(layers.Conv2D(32, (3, 3), activation='relu', input_shape=(32, 32, 3)))
     model.add(layers.MaxPooling2D((2, 2)))
     model.add(layers.Conv2D(64, (3, 3), activation='relu'))
     model.add(layers.MaxPooling2D((2, 2)))
     model.add(layers.Conv2D(64, (3, 3), activation='relu'))
```

Image Classification CNN

File Edit View Insert Runtime Tools Help All changes saved

\+ Code + Text

```
[15] model.summary()
```

```
[16] model.add(layers.Flatten())
     model.add(layers.Dense(64, activation='relu'))
     model.add(layers.Dense(10))
```

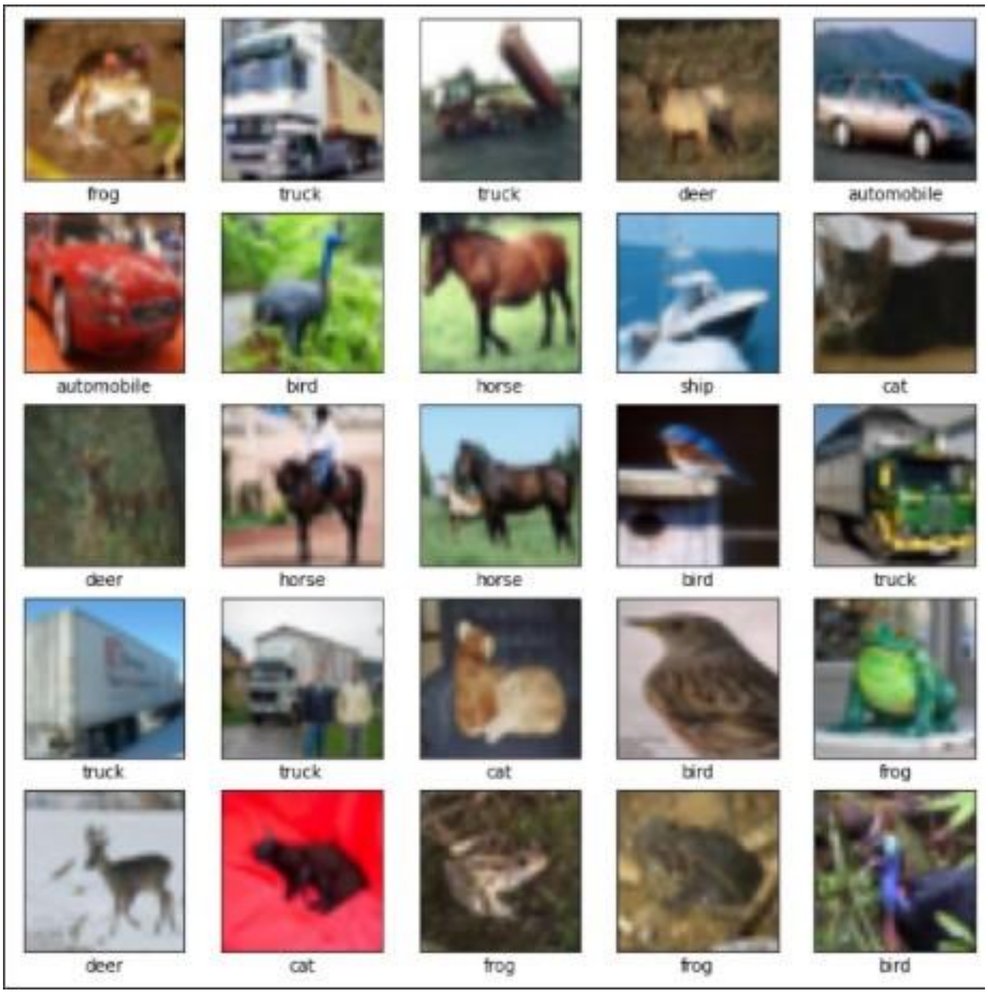
```
model.summary()
```

```
[21] model.compile(optimizer='adam',
                   loss=tf.keras.losses.SparseCategoricalCrossentropy(from_logits=True),
                   metrics=['accuracy'])

     history = model.fit(train_images, train_labels, epochs=20,
                         validation_data=(test_images, test_labels))
```

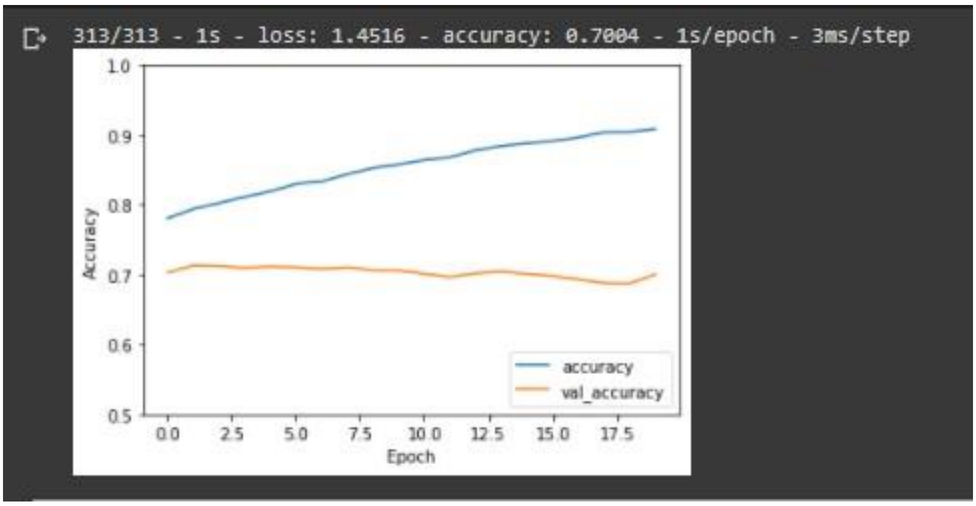
```
[22] plt.plot(history.history['accuracy'], label='accuracy')
     plt.plot(history.history['val_accuracy'], label = 'val_accuracy')
     plt.xlabel('Epoch')
     plt.ylabel('Accuracy')
     plt.ylim([0.5, 1])
     plt.legend(loc='lower right')

     test_loss, test_acc = model.evaluate(test_images,  test_labels, verbose=2)
```

```
[23] print(test_acc)
```

## Dataset:




## Output:




## Discussion:

I got an accuracy of 70.04% from the CIFAR100 dataset.

**Program no: 04**

**Program name:** ANN implementation using python to an Image dataset.

**Objective:** My goal is to implement ANN model using python to an Stellar Classification dataset.

## Description:

The ANNs model is the most common emerging tool that uses neural-based black-box technique for modeling environmental concerns, particularly, in water quality modeling [1]. The major features of ANNs are the measurement of nonlinear methods with input and output datasets, whereas particular functions and soft computing tools are applied as an example. The ANNs usually rely on the three structures: networks, hidden layers, and nodes. This model can be demarcated into different classes.

## Source Code:

Stellar-classification ANN.ipynb

File Edit View Insert Runtime Tools Help

+ Code + Text

```
[3] from google.colab import drive
    drive.mount('/content/drive')
```

```
Mounted at /content/drive
```

```
[4] import pandas as pd

    data = pd.read_csv("star_classification.csv")
    data.head()
```

```
[28] def create_data(file):
         data = pd.read_csv(file)

         class_map = {'STAR': 0, 'GALAXY': 1, 'QSO': 2}
         data['class_'] = data['class'].map(class_map)
         return data
```

```
[6] data.describe()
```

```
[7] numeric_columns = data.select_dtypes(include='number')
    numeric_columns.head()
```

```
[8] nonnumeric_column = data.select_dtypes(include = 'object')
    nonnumeric_column.head()
```

```
from sklearn import preprocessing
label_encoder = preprocessing.LabelEncoder()
nonnumeric_column_encoded = nonnumeric_column.apply(label_encoder.fit_transform)
nonnumeric_column_encoded.head()
```

Stellar-classification ANN.ipynb

File Edit View Insert Runtime Tools Help All changes saved

+ Code + Text

```
[10] together = [nonnumeric_column_encoded, numeric_columns]
     finalData = pd.concat(together, axis=1)
     finalData.head()
```

```
from keras.models import Sequential
from keras.layers import Dense
from keras.utils import np_utils
```

```
[12] data_y = finalData['class']
     data_x = finalData.drop(labels='class', axis=1)
     finalData.shape
```

```
[13] # define baseline model
     def baseline_model():
       model = Sequential()
       model.add(Dense(128, input_dim=data_x.shape[1], kernel_initializer='normal', activation='relu'))
       model.add(Dense(252,activation="relu"))
       model.add(Dense(1, kernel_initializer='normal', activation='sigmoid'))
       model.compile(loss='binary_crossentropy', optimizer='adam', metrics=['accuracy'])
       return model
```

```
[14] from keras.models import Sequential
     from keras.layers import Dense
     from keras.utils import np_utils
     model = baseline_model()
     model.fit(data_x, data_y,  epochs=100,  verbose=1)
```

```
[30] from sklearn.model_selection import train_test_split
```

```
[16] X_train, X_test, y_train, y_test = train_test_split(data_x, data_y, train_size=0.70)
```

Stellar-classification ANN.ipynb

File Edit View Insert Runtime Tools Help All changes saved

\+ Code + Text

```
[17] scores=model.evaluate(X_test,y_test,verbose=0)

print("accuracy",(scores[1]*100))

model.summary()

[20] predictions = (model.predict(X_test)>0.5).astype("int32").ravel()

[21] predictions

predictions = (model.predict(X_test)>0.5).astype("int32").ravel()

[26] from sklearn.metrics import classification_report
     print(classification_report(y_test, predictions))

[27] from sklearn.metrics import confusion_matrix
     from matplotlib import pyplot as plt
     conf_matrix = confusion_matrix(y_true= y_test, y_pred=predictions)

     fig, ax = plt.subplots(figsize=(7.5, 7.5))
     ax.matshow(conf_matrix, cmap=plt.cm.Blues, alpha=0.3)
     for i in range(conf_matrix.shape[0]):
         for j in range(conf_matrix.shape[1]):
             ax.text(x=j, y=i,s=conf_matrix[i, j], va='center', ha='center', size='xx-large')

     plt.xlabel('Predictions', fontsize=18)
     plt.ylabel('Actuals', fontsize=18)
     plt.title('Confusion Matrix', fontsize=18)
     plt.show()
```

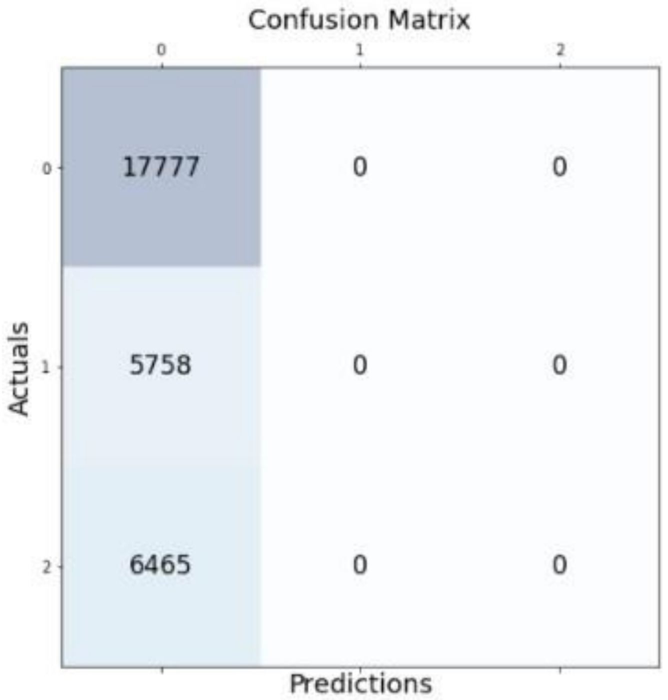
# Dataset:

| | obj_ID | alpha | delta | u | g | r | i | z | run_ID | rerun_ID | cam_col | field_ID | spec_obj_ID | class | redshift | plate | MJD |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 0 | 1.237661e+18 | 135.689107 | 32.494632 | 23.87882 | 22.27530 | 20.39501 | 19.16573 | 18.79371 | 3606 | 301 | 2 | 79 | 6.543777e+18 | GALAXY | 0.634794 | 5812 | 56354 |
| 1 | 1.237665e+18 | 144.826101 | 31.274185 | 24.77759 | 22.83188 | 22.58444 | 21.16812 | 21.61427 | 4518 | 301 | 5 | 119 | 1.176014e+19 | GALAXY | 0.779136 | 10445 | 58158 |
| 2 | 1.237661e+18 | 142.188790 | 35.582444 | 25.26307 | 22.66389 | 20.60976 | 19.34857 | 18.94827 | 3606 | 301 | 2 | 120 | 5.152200e+18 | GALAXY | 0.644195 | 4576 | 55592 |
| 3 | 1.237663e+18 | 338.741038 | -0.402828 | 22.13682 | 23.77656 | 21.61162 | 20.50454 | 19.25010 | 4192 | 301 | 3 | 214 | 1.030107e+19 | GALAXY | 0.932346 | 9149 | 58039 |
| 4 | 1.237680e+18 | 345.282593 | 21.183866 | 19.43718 | 17.58028 | 16.49747 | 15.97711 | 15.54461 | 8102 | 301 | 3 | 137 | 6.891865e+18 | GALAXY | 0.116123 | 6121 | 56187 |

# Output:

```
from sklearn.metrics import classification_report
print(classification_report(y_test, predictions))

              precision    recall  f1-score   support

           0       0.59      1.00      0.74     17777
           1       0.00      0.00      0.00      5758
           2       0.00      0.00      0.00      6465

    accuracy                           0.59     30000
   macro avg       0.20      0.33      0.25     30000
weighted avg       0.35      0.59      0.44     30000
```

## Confusion Matrix:




## Discussion:

The data consists of 100,000 observations of space taken by the SDSS (Sloan Digital Sky Survey). Every observation is described by 17 feature columns and 1 class column which identifies it to be either a star, galaxy or quasar.

Here, I got an accuracy of 59.44% from the dataset Stellar Classification from Kaggle.

**Dataset Link:** https://www.kaggle.com/fedesoriano/stellar-classification-dataset-sdss17.