**Fake social media accounts detection**

**Source code :**

import pandas as pd

import matplotlib.pyplot as plt

import numpy as np

import seaborn as sns

import tensorflow as tf

from tensorflow import keras

from tensorflow.keras.layers import Dense, Activation, Dropout

from tensorflow.keras.optimizers import Adam

from tensorflow.keras.metrics import Accuracy

from sklearn import metrics

from sklearn.preprocessing import LabelEncoder

from sklearn.metrics import classification\_report,accuracy\_score,roc\_curve,confusion\_matrix

import os

for dirname, \_, filenames in os.walk('/kaggle/input'):

for filename in filenames:

print(os.path.join(dirname, filename))

import warnings

warnings.filterwarnings("ignore")

def fxn():

warnings.warn("deprecated", DeprecationWarning)

with warnings.catch\_warnings():

warnings.filterwarnings("ignore", category=DeprecationWarning)

fxn()

# Load the training dataset

instagram\_df\_train=pd.read\_csv('/kaggle/input/instagram-fake-spammer-genuine-accounts/train.csv')

instagram\_df\_train

# Load the testing data

instagram\_df\_test=pd.read\_csv('/kaggle/input/instagram-fake-spammer-genuine-accounts/test.csv')

instagram\_df\_test

instagram\_df\_train.head()

instagram\_df\_train.tail()

# Getting dataframe info

instagram\_df\_train.info()

# Get the statistical summary of the dataframe

instagram\_df\_train.describe()

# Checking if null values exist

instagram\_df\_train.isnull().sum()

# Get the number of unique values in the "profile pic" feature

instagram\_df\_train['profile pic'].value\_counts()

# Get the number of unique values in "fake" (Target column)

instagram\_df\_train['fake'].value\_counts()

# Visualize the data

sns.countplot(instagram\_df\_train['fake'])

plt.show()

# Visualize the private column data

sns.countplot(instagram\_df\_train['private'])

plt.show()

# Visualize the "profile pic" column data

sns.countplot(instagram\_df\_train['profile pic'])

plt.show()

# Visualize the data

plt.figure(figsize = (20, 10))

sns.distplot(instagram\_df\_train['nums/length username'])

plt.show()

# Visualize the data

plt.figure(figsize = (20, 10))

sns.distplot(instagram\_df\_train['nums/length username'])

plt.show()

# Correlation plot

plt.figure(figsize=(20, 20))

cm = instagram\_df\_train.corr()

ax = plt.subplot()

sns.heatmap(cm, annot = True, ax = ax)

# Training and testing dataset (inputs)

X\_train = instagram\_df\_train.drop(columns = ['fake'])

X\_test = instagram\_df\_test.drop(columns = ['fake'])

X\_trainplt.show()

# Training and testing dataset (Outputs)

y\_train = instagram\_df\_train['fake']

y\_test = instagram\_df\_test['fake']

y\_train

y\_train = tf.keras.utils.to\_categorical(y\_train, num\_classes = 2)

y\_test = tf.keras.utils.to\_categorical(y\_test, num\_classes = 2)y\_train

y\_train

import tensorflow.keras

from tensorflow.keras.models import Sequential

from tensorflow.keras.layers import Dense, Dropout

model = Sequential()

model.add(Dense(50, input\_dim=11, activation='relu'))

model.add(Dense(150, activation='relu'))

model.add(Dropout(0.3))

model.add(Dense(150, activation='relu'))

model.add(Dropout(0.3))

model.add(Dense(25, activation='relu'))

model.add(Dropout(0.3))

model.add(Dense(2,activation='softmax'))

model.summary()

model.compile(optimizer = 'adam', loss = 'categorical\_crossentropy', metrics = ['accuracy'])

epochs\_hist = model.fit(X\_train, y\_train, epochs = 50, verbose = 1, validation\_split = 0.1)

plt.plot(epochs\_hist.history['loss'])

plt.plot(epochs\_hist.history['val\_loss'])

plt.title('Model Loss Progression During Training/Validation')

plt.ylabel('Training and Validation Losses')

plt.xlabel('Epoch Number')

plt.legend(['Training Loss', 'Validation Loss'])

plt.show()

print(classification\_report(test, predicted\_value))

plt.figure(figsize=(10, 10))

cm=confusion\_matrix(test, predicted\_value)

sns.heatmap(cm, annot=True)

plt.show()