CONTENT PAGE

ABSTRACT

With the advent of the Internet and social media, while hundreds of people have benefitted from the vast sources of information available, there has been an enormous increase in the rise of cyber-crimes. According to a 2019 report in the Economics Times, India has witnessed a 457% rise in cybercrime in the five year span between 2011 and 2016. Most speculate that this is due to impact of social media such as Instagram on our daily lives. While these definitely help in creating a sound social network, creation of user accounts in these sites usually needs just an email-id. A real life person can create multiple fake IDs and hence impostors can easily be made. Unlike the real world scenario where multiple rules and regulations are imposed to identify oneself in a unique manner (for example while issuing one's passport or driver's license), in the virtual world of social media, admission does not require any such checks. In this project, we study the different accounts of Instagram, in particular and try to assess an account as fake or real.

INTRODUCTION & MOTIVATION

Having the ability to check the authenticity of a user's following is crucial for brands looking to work with influencers. Social Media is one of the most important platforms, especially for youth, to express themselves to the world.

This platform can be used by them as a way of interacting with same type of people and age group, or to present their views. However, use of technology has also constrained with various implications – humans can misuse the technology to cause harm and spread hatred via the same social media platform.

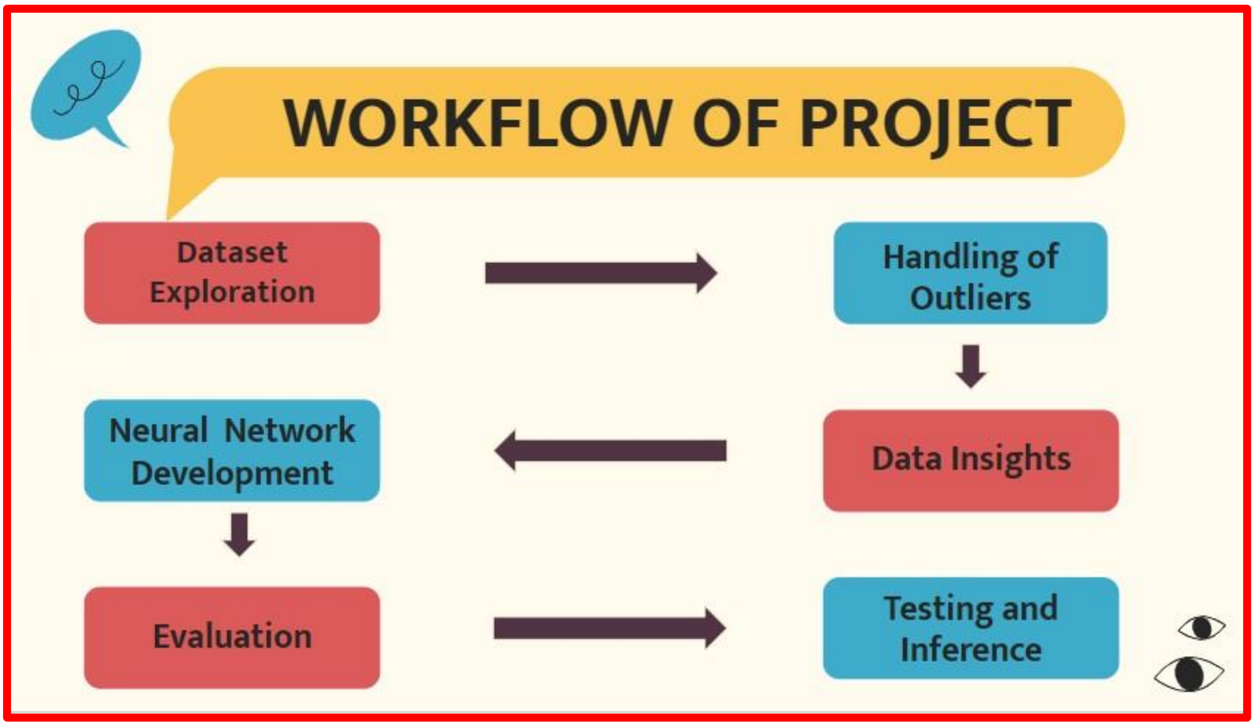
Keeping this is mind, we have tried to perform a basic solution to this problem via deep learning algorithm implementation over a dataset to check with respect to various social media platform – Instagram's attributes , can a neural network actually help to predict a fake or real user profile.

Proposed Method with Flow Diagram

An artificial neural network (ANN) is a computing system designed to simulate how the human brain analyzes and processes information. It is the foundation of artificial intelligence (AI) and solves problems that would prove impossible or difficult by human or statistical standards.

Artificial Neural Networks are primarily designed to mimic and simulate the functioning of the human brain. Using the mathematical structure, it is ANN constructed to replicate the biological neurons.

The concept of ANN follows the same process as that of a natural neural net. The objective of ANN is to make the machines or systems understand and ape how a human brain makes a decision and then ultimately takes action. Inspired by the human brain, the fundamentals of neural networks are connected through neurons or nodes.



MODULES OF THE PROJECT

- **Module I - Initial Data Exploration**: It is the initial step in data analysis in which we use data visualization and statistical techniques to describe dataset characterizations, such as size, quantity, and accuracy, in order to better understand the nature of the data.

- **Module II - Data Wrangling:** In this process, cleaning and unifying of messy and complex data sets takes place for easy access and analysis. With the amount of data and data sources rapidly growing and expanding, it is getting increasingly essential for large amounts of available data to be organized for analysis.

- **Module III - Data Insights:** Basic statistical and visual analysis with respect to scraped datasets, which can help to provide basic overview of how data needs to be cleaned or further processed with respect to core neural network development

- **Module IV - Core Neural Network Development:** This module comprises of core neural network development – a basic artificial neural network (ANN), which takes input of basic attributes of independent features of dataset and tries to predict target feature – fake or not.

- **Module V – Evaluation:** After neural network development, this module is being implemented in order to check how the model is actually performing training wise and how it performs on unseen test data – accuracy and loss of model.

- **Module VI - Testing and Inference:** Once the desired and tuned model is obtained, this module is implemented in order to test model (saved model and later loaded for future use) on random unseen data attributes to determine whether the user is fake or not.

IMPLEMENTATION REQUIREMENTS

1)Initial Packages – Pandas, NumPy, Matplotlib, Seaborn – for basic statistical analysis and mathematical insights

2)TensorFlow - TensorFlow is a free and open-source software library for machine learning and artificial intelligence. It can be used across a range of tasks but has a particular focus on training and inference of deep neural networks

3)Scikit-Learn - Scikit-learn is a free software machine learning library for the Python programming language

4) Python – Python based programming language interface in order to run and execute the application

5)Google Colab - Colab is a free Jupyter notebook environment that runs entirely in the cloud – cloud based instance which helps to set up a virtual python based environments and run machine learning or deep learning models

OUTPUT SCREENSHOTS

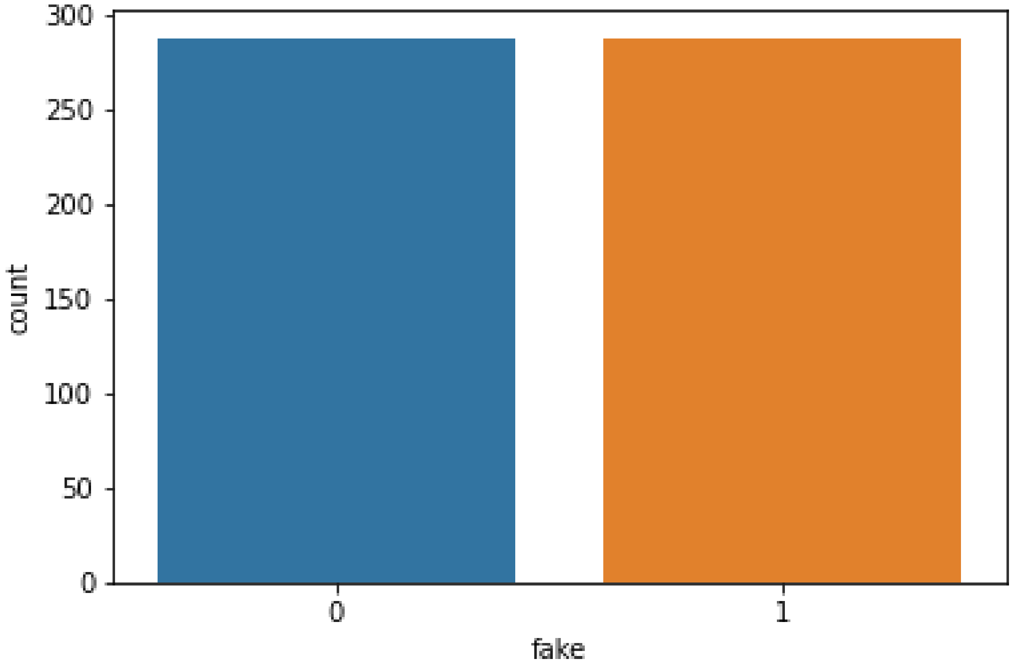
Load Data (Pre-processing)

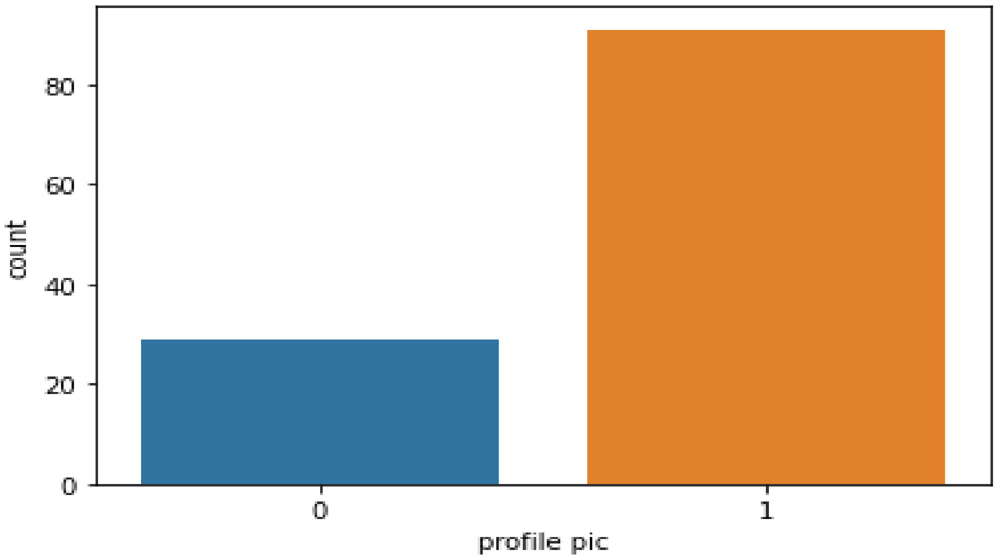
```
pd.read_csv(test_data_path)
```

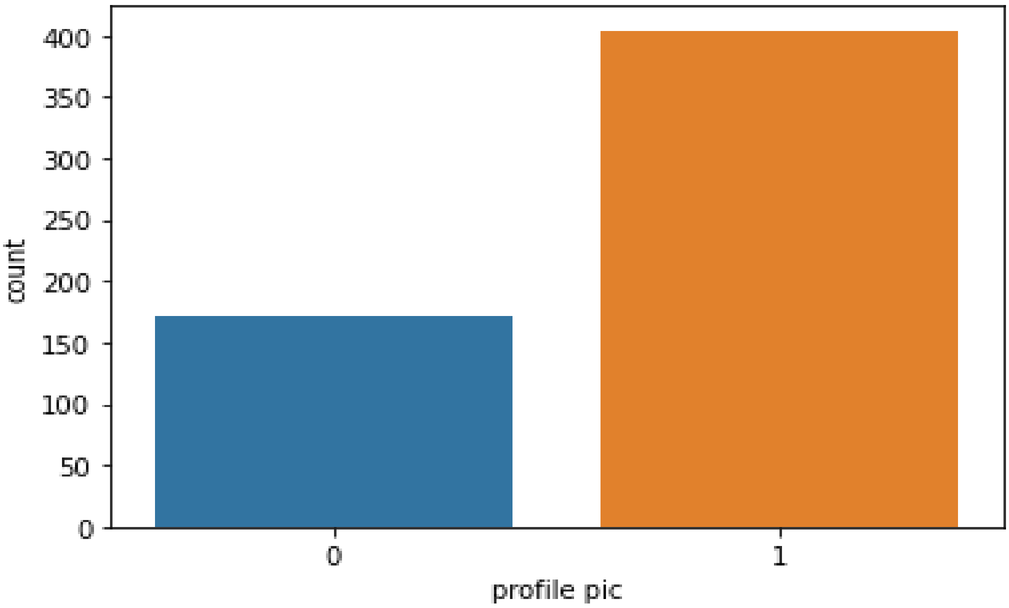
	profile pic	nums/length username	fullname words	nums/length fullname	name==username	description length	external URL	private	#posts	#followers	#follows	fake
0	1	0.33	1	0.33	1	30	0	1	35	488	604	0
1	1	0.00	5	0.00	0	64	0	1	3	35	6	0
2	1	0.00	2	0.00	0	82	0	1	319	328	668	0
3	1	0.00	1	0.00	0	143	0	1	273	14890	7369	0
4	1	0.50	1	0.00	0	76	0	1	6	225	356	0
...	...	...	...	...	...	...	...	...	...	...	...	...
115	1	0.29	1	0.00	0	0	0	0	13	114	811	1
116	1	0.40	1	0.00	0	0	0	0	4	150	164	1
117	1	0.00	2	0.00	0	0	0	0	3	833	3572	1
118	0	0.17	1	0.00	0	0	0	0	1	219	1695	1
119	1	0.44	1	0.00	0	0	0	0	3	39	68	1

120 rows × 12 columns

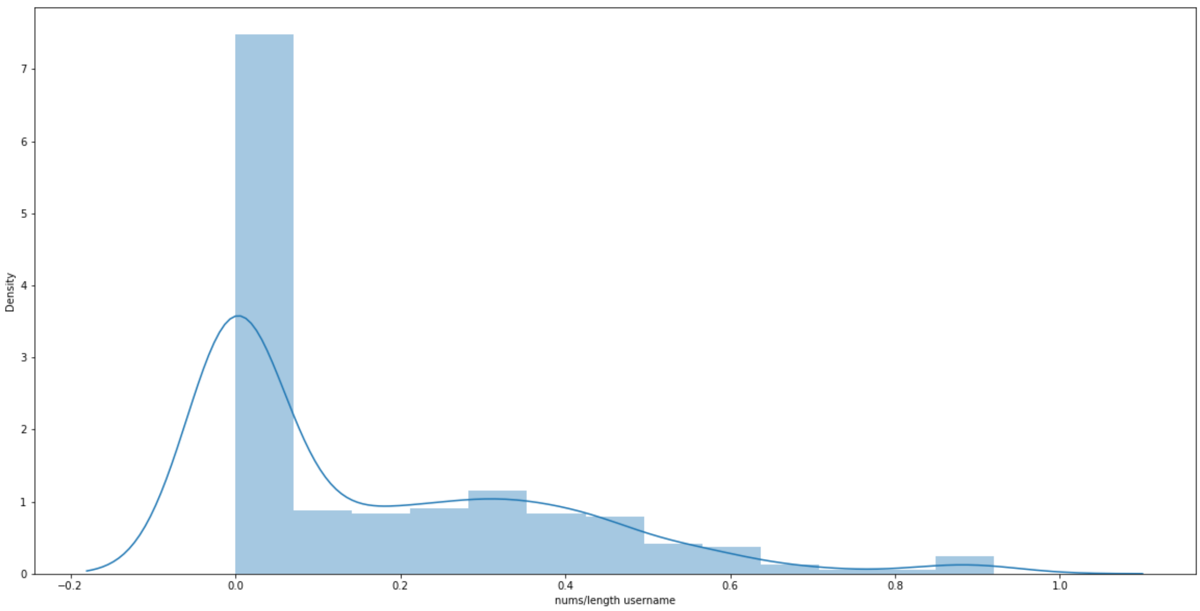
Bar Plot – Visualization (Data Insights)


300
250
200
150
100
50
0
count
0
1
fake

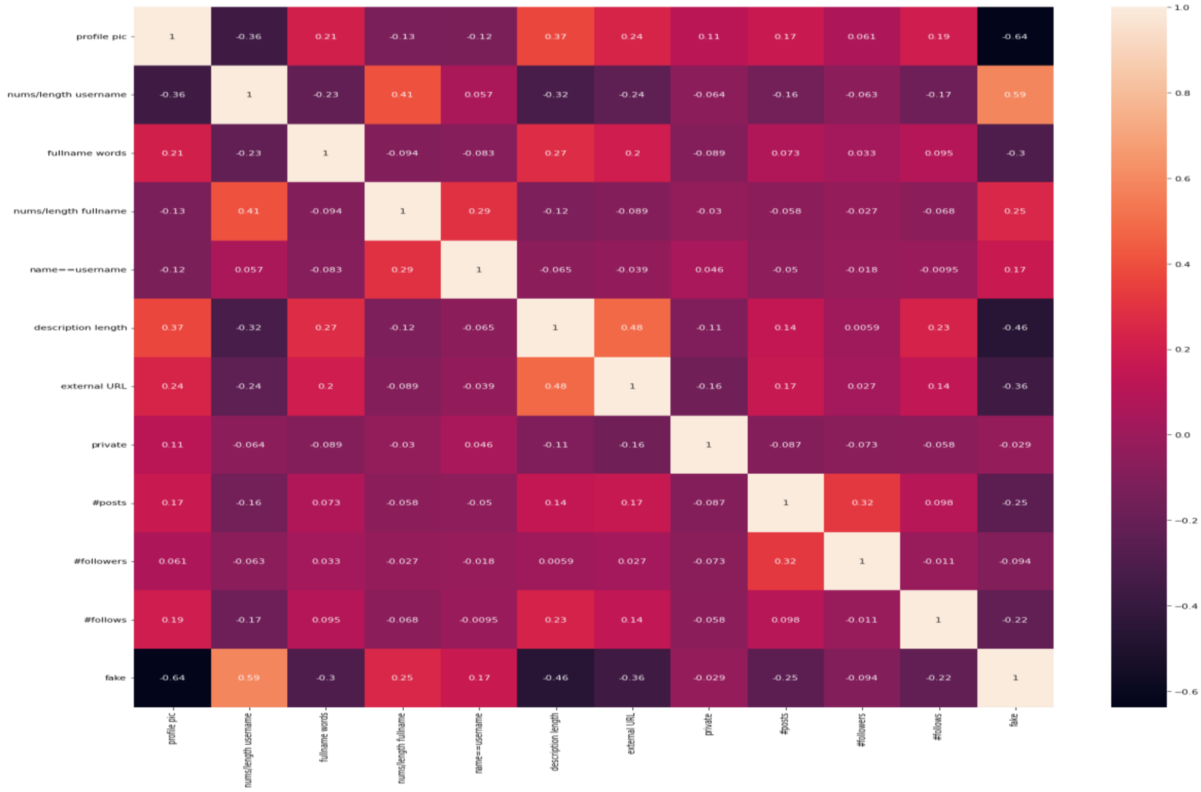

80
60
40
20
0
count
0
1
profile pic



KDE Plot (Data Insights)



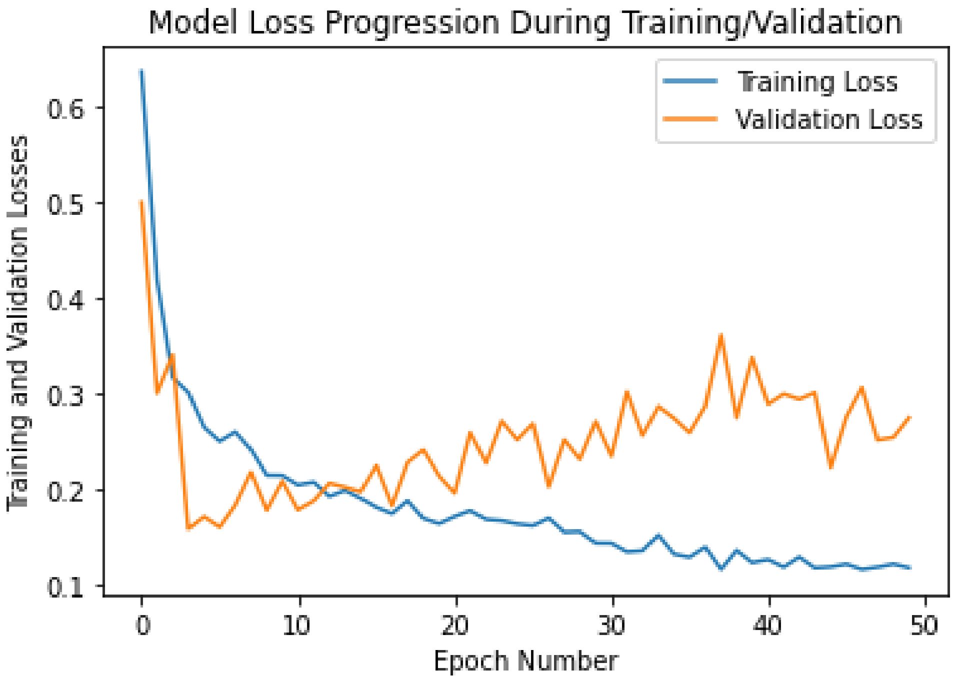
Heat Map – Correlation Check (Data Insights)



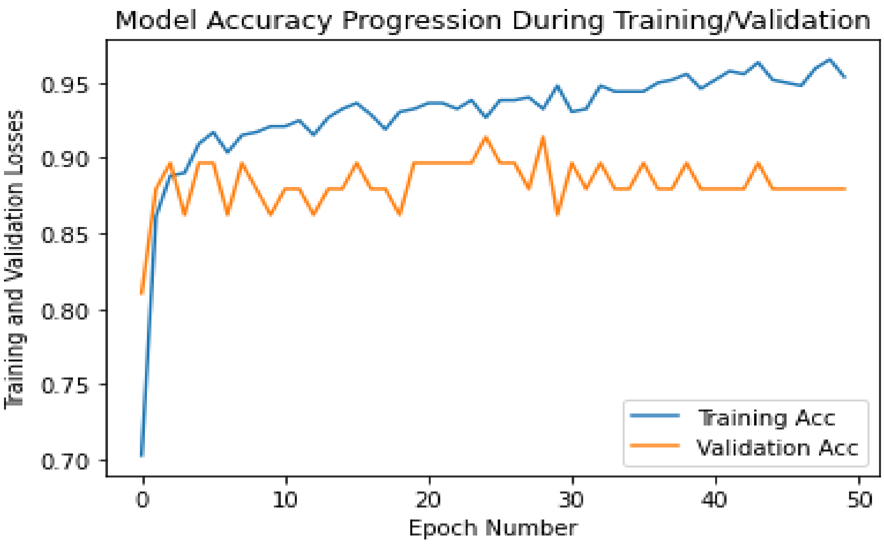
Model Training- (Sequential Training)

```
Model: "sequential"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 dense (Dense)               (None, 50)                600

 dense_1 (Dense)             (None, 150)               7650

 dropout (Dropout)           (None, 150)               0

 dense_2 (Dense)             (None, 150)               22650

 dropout_1 (Dropout)         (None, 150)               0

 dense_3 (Dense)             (None, 25)                3775

 dropout_2 (Dropout)         (None, 25)                0

 dense_4 (Dense)             (None, 2)                 52

=================================================================
Total params: 34,727
Trainable params: 34,727
Non-trainable params: 0
_________________________________________________________________
Epoch 1/50
17/17 [==============================] - 1s 24ms/step - loss: 0.6356 - accuracy: 0.6583 - val_loss: 0.4993 - val_accuracy: 0.8103
Epoch 2/50
17/17 [==============================] - 0s 7ms/step - loss: 0.4191 - accuracy: 0.8707 - val_loss: 0.3006 - val_accuracy: 0.8276
Epoch 3/50
17/17 [==============================] - 0s 6ms/step - loss: 0.3170 - accuracy: 0.8919 - val_loss: 0.3410 - val_accuracy: 0.8276
Epoch 4/50
17/17 [==============================] - 0s 7ms/step - loss: 0.3015 - accuracy: 0.8958 - val_loss: 0.1594 - val_accuracy: 0.9138
Epoch 5/50
17/17 [==============================] - 0s 6ms/step - loss: 0.2653 - accuracy: 0.9054 - val_loss: 0.1720 - val_accuracy: 0.8966
Epoch 6/50
17/17 [==============================] - 0s 6ms/step - loss: 0.2506 - accuracy: 0.9131 - val_loss: 0.1611 - val_accuracy: 0.9138
Epoch 7/50
17/17 [==============================] - 0s 6ms/step - loss: 0.2604 - accuracy: 0.9093 - val_loss: 0.1841 - val_accuracy: 0.8966
Epoch 8/50
17/17 [==============================] - 0s 7ms/step - loss: 0.2420 - accuracy: 0.9151 - val_loss: 0.2184 - val_accuracy: 0.8966
Epoch 9/50
17/17 [==============================] - 0s 7ms/step - loss: 0.2153 - accuracy: 0.9266 - val_loss: 0.1787 - val_accuracy: 0.8966
Epoch 10/50
17/17 [==============================] - 0s 7ms/step - loss: 0.2151 - accuracy: 0.9286 - val_loss: 0.2093 - val_accuracy: 0.8966
Epoch 11/50
17/17 [==============================] - 0s 6ms/step - loss: 0.2052 - accuracy: 0.9189 - val_loss: 0.1791 - val_accuracy: 0.8966
Epoch 12/50
17/17 [==============================] - 0s 6ms/step - loss: 0.2081 - accuracy: 0.9266 - val_loss: 0.1888 - val_accuracy: 0.9138
Epoch 13/50
17/17 [==============================] - 0s 6ms/step - loss: 0.1933 - accuracy: 0.9189 - val_loss: 0.2070 - val_accuracy: 0.9138
Epoch 14/50
17/17 [==============================] - 0s 6ms/step - loss: 0.1995 - accuracy: 0.9286 - val_loss: 0.2029 - val_accuracy: 0.9138
Epoch 15/50
17/17 [==============================] - 0s 6ms/step - loss: 0.1913 - accuracy: 0.9170 - val_loss: 0.1981 - val_accuracy: 0.9138
```

Training Progress - Loss (Training)



<u>Training Progress - Accuracy(Training)</u>



<u>Classification Report (Evaluation)</u>

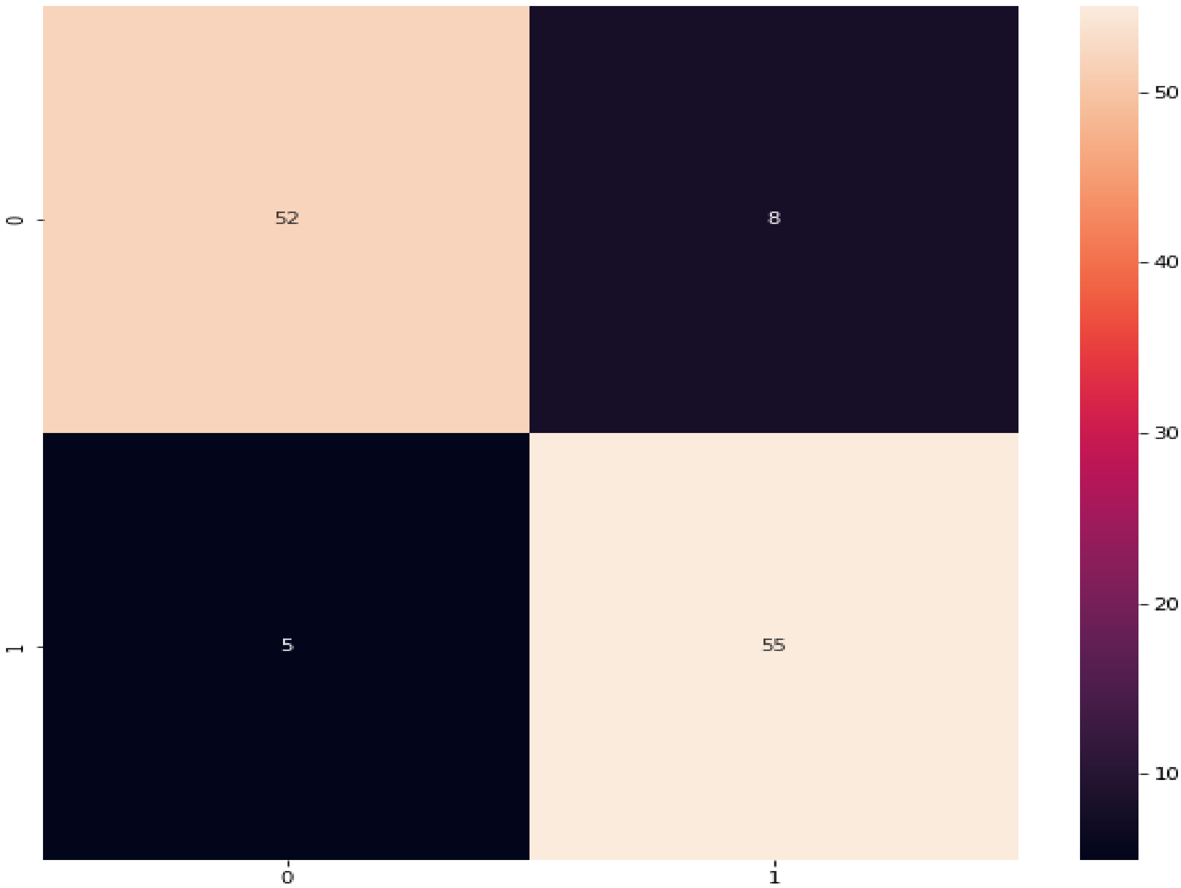
```
print("Accuracy : ", get_avg(model_training_progress['Accuracy']) * 100)

print("Validation Accuracy : ", get_avg(model_training_progress['Validation_Accuracy']) * 100)


print("Loss : ", get_avg(model_training_progress['Loss']) * 100)


print("Validation Loss : ", get_avg(model_training_progress['Validation Loss']) * 100)
```

```
Accuracy :  92.91891884803772
Validation Accuracy :  88.31034588813782
Loss :  17.7891463637352
Validation Loss :  26.413013011217117
```

```
predicted = model.predict(X_test)

predicted_value = []
test = []
for i in predicted:
    predicted_value.append(np.argmax(i))

for i in y_test:
    test.append(np.argmax(i))

print(classification_report(test, predicted_value))
```

```
              precision    recall  f1-score   support

           0       0.91      0.87      0.89        60
           1       0.87      0.92      0.89        60

    accuracy                           0.89       120
   macro avg       0.89      0.89      0.89       120
weighted avg       0.89      0.89      0.89       120
```

<u>Confusion Matrix (Evaluation)</u>



CONCLUSION

The proposed project majorly focuses on how deep learning algorithms - Artificial Neural Network or ANNs can be leveraged for better insights exploration over a well distributed dataset. The proposed framework exhibits how different attributes with respect to user's activity can be learned or analysed by machine learning or deep learning algorithms to predict any suspicious activity and tell the probability of that specific account being a fake or genuine one.

Furthermore, this algorithm can be improved by scraping more metadata - like visual features - images, posts, captions, activity spend time and heavy deep learning models can be ensemble - like multimodal deep learning for even better results.

REFERENCES

1. Instagram Fake Spammer Dataset - Kaggle
2. Easy ways to analyse if account is fake or not - WikiBlog
3. Tensorflow - Basic Code Base
4. Instagram Fake and Automated Account Detection - Fatih Cagatay Akyon; M. Esat Kalfaoglu

APPENDIX A - Source Code

```
#Initial Data Exploration and Data Wrangling

  import pandas as

pd import numpy as

np

 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
```

```
train_data_path = 'datasets/Fake-Instagram-Profile-
Detectionmain/insta_train.csv'

test_data_path = 'datasets/Fake-Instagram-Profile-
Detectionmain/insta_test.csv'

 pd.read_csv(test_data_path)



576 + 120



train_data_path =
'datasets/Insta_Fake_Profile_Detection/train.csv'

test_data_path =
'datasets/Insta_Fake_Profile_Detection/test.csv'

 pd.read_csv(train_data_path)



# Load the training dataset

instagram_df_train=pd.read_csv(train_data_path)

instagram_df_train



# Load the testing data

instagram_df_test=pd.read_csv(test_data_path)

instagram_df_test

 instagram_df_train.head()
instagram_df_train.tail()
```

```
 instagram_df_test.head()

instagram_df_test.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()

 instagram_df_test.info()

 instagram_df_test.describe()
```

```
instagram_df_test.isnull().sum()

 instagram_df_test['fake'].value_counts()



# Perform Data Visualizations



# 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()



# Correlation plot plt.figure(figsize=(20,
20))
```

```
cm = instagram_df_train.corr() ax =

plt.subplot() sns.heatmap(cm, annot =

True, ax = ax) plt.show()

 sns.countplot(instagram_df_test['fake'])

 sns.countplot(instagram_df_test['private'])

 sns.countplot(instagram_df_test['profile

pic'])



# Preparing Data to Train the Model



# Training and testing dataset (inputs)

X_train = instagram_df_train.drop(columns = ['fake'])

X_test = instagram_df_test.drop(columns = ['fake'])

X_train



X_test



# Training and testing dataset (Outputs)

y_train = instagram_df_train['fake'] y_test

= instagram_df_test['fake']

 y_train
```

```
y_test


# Scale the data before training the model from

sklearn.preprocessing import StandardScaler, MinMaxScaler

scaler_x = StandardScaler()

X_train = scaler_x.fit_transform(X_train)

X_test = scaler_x.transform(X_test)


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_test


# print the shapes of training and testing datasets

X_train.shape, X_test.shape, y_train.shape, y_test.shape

Training_data = len(X_train)/( len(X_test) + len(X_train) ) *
100

Training_data
```

```
Testing_data = len(X_test)/( len(X_test) + len(X_train) ) * 100

Testing_data



# Building and Training Deep Training Model

 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)
```

```
# Access the Performance of the model

 print(epochs_hist.history.keys())

 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()



  predicted =

model.predict(X_test)

 predicted_value =
[]
```

```
test = [] for i in

predicted:

    predicted_value.append(np.argmax(i))

    for i in

y_test:

    test.append(np.argmax(i))

 print(classification_report(test,

predicted_value))

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

cm=confusion_matrix(test, predicted_value)

sns.heatmap(cm, annot=True) plt.show()
```

APPENDIX B - Github Project Link

Project Link - https://github.com/harshgeek4coder/18CSC305J_AI_Insta_Fake_Profile_Detection