A
Learning Project-II Report
On

"FAKE SOCIAL MEDIA PROFILE DETECTION USING MACHINE LEARNING"

Submitted in partial fulfillment of
The requirements for the 4th Semester Sessional Examination of

BACHELOR OF TECHNOLOGY
IN
COMPUTER SCIENCE & ENGINEERING

By

K SAMPAT KUMAR
DHRUV RAKSHIT
SOURAV MAHAPATRA

Registration No.

22CSEAIML025
22CSEAIML039
22CSEAIML053

DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING

GIET UNIVERSITY, GUNUPUR

2023 - 24

GIET UNIVERSITY,GUNUPUR

Dist. - Rayagada, Odisha-765022, Contact:- +91 7735745535, 06857-250170,172, Visit us:- **www.giet.edu**

Department of Computer Science & Engineering

CERTIFICATE

This is to certify that the project work entitled "FAKE SOCIAL MEDIA PROFILE DETECTION USING MACHINE LEARNING" is done by K. SAMPAT KUMATR, DHRUV RAKSHIT, SOURAV MAHAPATRA. 22CSEAIML025, 22CSEAIML039, 22CSEAIML053 in partial fulfillment of the requirements for the 4th Semester Sessional Examination of Bachelor of Technology in Computer Science and Engineering during the academic year 2023-24. This work is submitted to the department as a part of evaluation of 4th Semester Learning Project-II.

Proctors/Class Teacher

Project Coordinator, 2nd Year

HoD, CSE, 2nd Year

ACKNOWLEDGEMENT

We express our sincere gratitude to DR. B.B. BISWAL of Computer science and engineering for giving us an opportunity to accomplish the project. Without his active support and guidance, this project report has not been successfully completed.

We also thanks Mr. Bhavani Sankar Panda, Project Coordinator, Dr. Premanshu Sekhar Rath, Head of the Department of Computer Science and Engineering, Prof. (Dr.) Kakita Murali Gopal, Dy. Dean, Computational Science, SOET for their consistent support, guidance and help.

We also thanks to our friends and family members for their constant support during the execution of the project.

K. SAMPAT KUMAR (22CSEAIML025)
DHRUV RAKSHIT (22CSEAIML039)
SOURAV MAHAPATRA (22CSEAIML053)

CONTENT PAGE

ABSTRACT

The social life of everyone has become associated with online social networks. These sites have made a drastic change in the way we pursue our social life. Making friends and keeping in contact with them and their updates has become easier. But with their rapid growth, many problems like fake profiles, and online impersonation have also grown. Fake profiles often spam legitimate users, posting inappropriate or illegal content.

Our project "IMPOSTER HUB" helps legitimate users to identify fake accounts on social networking sites using a machine learning model trained on previously available data. It is a user-friendly website that provides the user option to enter the attribute of that profile they want to check for authenticity. These attributes are passed into the machine learning model running on the backend which classifies the profile to be fake or real.

There are numerous cases where produced accounts have been effectively distinguished utilizing machine adapting techniques however the amount of research work is very low to recognize counterfeit characters made by people. For bots the ML models used various features to calculate the no. of followers to the no. of friends that an account has on social media platforms. The no. of friends to the no. of followers of any account are easily available in the account profiles and no rights are violated of any accounts. In order to accomplish the task of detecting the fake accounts we establish a forged human account.

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 networkactually help to predict a fake or real user profile.

Purpose:

This project aims to develop a deep learning model that can identify fake follower profiles on Instagram. Fake followers are inauthentic accounts created to artificially inflate a user's follower count. This project will help brands identify genuine influencers with real, engaged audiences, leading to more effective influencer marketing campaigns.

Project Scope:

This project will focus on analyzing Instagram data, specifically focusing on user profile attributes. The deep learning model will be trained on a dataset containing labeled user profiles (real vs. fake). The model will learn to identify patterns and features associated with fake accounts.

Product Features:

- **Fake Follower Detection:** The core functionality of the system will be to analyze an Instagram user's profile and predict the likelihood of having a significant portion of fake followers.

- **Engagement Rate Analysis:** The model can potentially be expanded to analyze engagement metrics (likes, comments) alongside profile attributes, providing a more comprehensive assessment of follower authenticity.

- **Influencer Evaluation Tool:** By integrating with influencer marketing platforms or dashboards, the system can be used as a tool for brands to evaluate potential influencers based on the authenticity of their follower base.

Benefits:

- **Improved Influencer Marketing:** Brands can invest in influencer marketing campaigns with greater confidence, knowing they're reaching real audiences.
- **Enhanced Brand Reputation:** Partnering with authentic influencers fosters trust and strengthens brand image.
- **Fairer Influencer Landscape:** Discourages the use of fake followers and promotes organic audience growth.

Future Considerations:

- Expanding data analysis to include other social media platforms.
- Incorporating real-time data analysis for continuous monitoring of follower authenticity.
- Developing an API to integrate the model with influencer marketing platforms seamlessly.

Review of Existing methods and their Limitations

1. Machine Learning Techniques:

- **Strengths:** This approach analyzes user data (profile information, posting history, etc.) to identify patterns indicative of fakeness. Algorithms like Random Forest and Support Vector Machines can achieve good accuracy.
- **Limitations:** Machine learning models require large, labeled datasets to train effectively. Creating such datasets can be expensive and time-consuming. Additionally, these models might struggle to adapt to new tactics employed by creators of fake profiles.

2. Social Network Analysis:

- **Strengths:** This method examines a profile's connections and interactions with other accounts. Factors like sudden spikes in followers or a lack of reciprocal connections can raise red flags.
- **Limitations:** Social network analysis is often limited to publicly available information. Fake profiles can be designed to have seemingly legitimate connections, making them difficult to detect solely through this method.

3. Content Analysis:

- **Strengths:** By analyzing the content a profile posts (text, images, etc.), detectors can look for characteristics like unusual language patterns, generic content, or a sudden surge in activity.
- **Limitations:** Content analysis might struggle with sophisticated fake profiles that can mimic human posting behavior. Furthermore, it can't always distinguish between automated bots and genuine accounts with similar posting styles.

Overall Limitations:

Evolving Techniques: As detection methods improve, creators of fake profiles adapt their tactics. This creates an ongoing arms race where developers need to constantly refine their techniques.

Data Privacy: Collecting and analyzing user data raises privacy concerns. Finding the right balance between effective detection and user privacy is crucial.

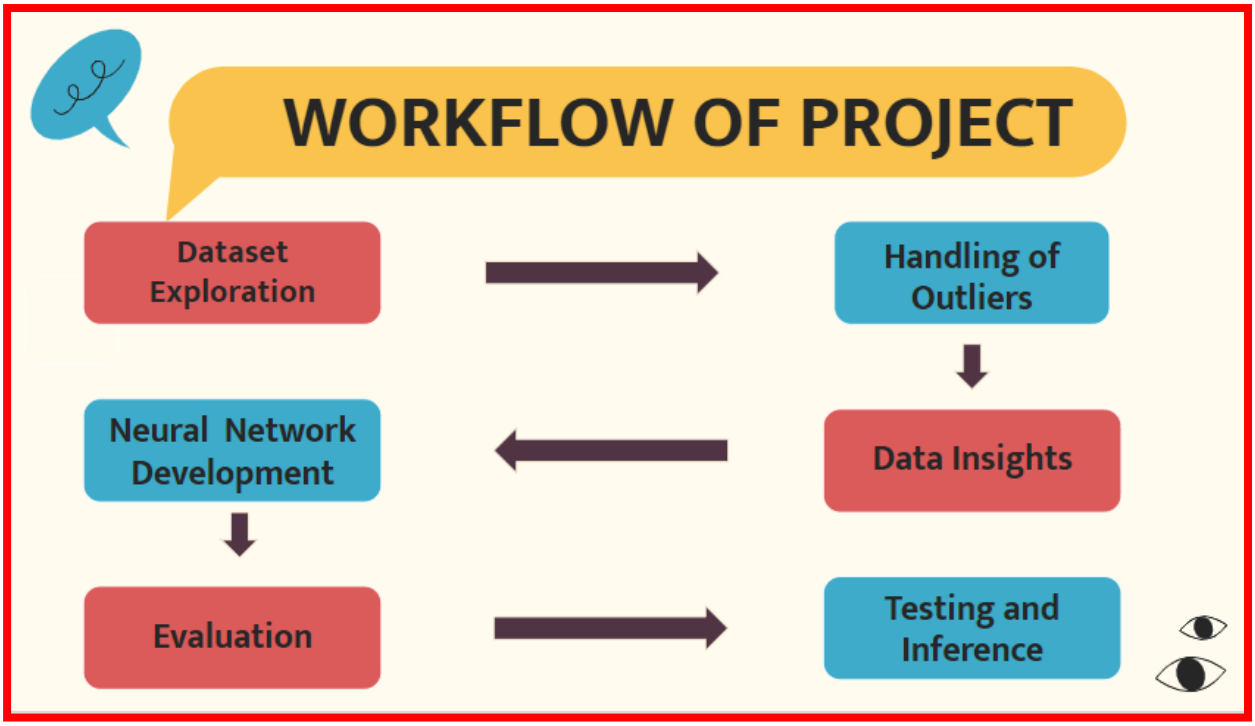
Real-time Detection: Many current methods struggle with real-time detection, allowing fake profiles to operate for a period before being identified.

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 throughneurons 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.Here are some key aspects to explore:

- **Data Acquisition:**
  - Source**:** Identify the source of your data. Will you be collecting it directly from Instagram's API (if allowed) or using a third-party dataset provider?
  - Labeling: How will you obtain labels for user profiles (real vs. fake)? Consider manual labeling, using existing datasets with labels, or potentially exploring weakly supervised learning techniques.
  - Data Volume: Assess the size and quality of the available data. Is there enough data to train a robust model effectively?
- **Data Understanding:**
  - Features: Identify the features available in your dataset that might be relevant to identifying fake followers. This could include profile attributes like account age, number of posts, follower-to-following ratio, bio content, and typical engagement metrics (likes, comments per post).
  - Exploratory Data Analysis (EDA): Use visualization techniques like histograms, box plots, and scatter plots to understand the distribution of features for real vs. fake profiles. Look for patterns or outliers that might be indicative of fake accounts.
  - Data Cleaning: Address any missing values or inconsistencies in your data. Techniques like imputation or data deletion might be necessary for specific features.

- **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.Here are some common data wrangling tasks you might encounter:

- Missing Values: Address missing values in your data using techniques like imputation (filling in missing values with estimated values) or deletion. The chosen approach will depend on the specific feature and the amount of missing data.

- Inconsistent Formats: Standardize data formats for consistency. For example, ensure dates are formatted consistently and text data is free of extra spaces or special characters.
- Outlier Detection: Identify and handle outliers, which are data points that fall far outside the expected range. You might choose to remove outliers, winsorize them (cap their values to a certain threshold), or investigate them further to understand their cause.
- Feature Engineering: Create new features from existing ones if they might be more informative for the model. For example, calculate engagement rate as average likes/comments per post.

- **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.Here's how this module will help:

- Understanding Feature Distributions: Use techniques like histograms and boxplots to visualize the distribution of features for real vs. fake profiles. This can reveal patterns and potential differences between the two groups.
- Identifying Correlations: Explore correlations between features using techniques like scatter plots and correlation coefficients. This can help you understand how different features might be related to each other and whether they might be redundant for model training.
- Data Cleaning Validation: Use visualizations to validate your data cleaning efforts. For example, ensure missing values have been addressed appropriately, and outliers are no longer skewing the data distribution.

- **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. Here's an overview of the process:

- Choosing an Architecture: Several neural network architectures can be suitable for this task. A common choice is a Multi-Layer Perceptron (MLP), which is a feedforward network with multiple hidden layers. Convolutional Neural Networks (CNNs) might also be explored if image data (profile pictures) is included as a feature.
- Defining the Network: Specify the number of hidden layers, the number of neurons in each layer, and the activation functions used in each layer. The activation function introduces non-linearity into the network, allowing it to learn complex relationships between features.

- Training the Network: Train the network on your prepared data. This involves feeding the model with labeled examples (profile features and corresponding labels - real or fake) and iteratively adjusting the network weights to minimize the prediction error. Common optimization algorithms like Adam or Stochastic Gradient Descent (SGD) are used for this purpose.

  - **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. Here are some key metrics used for model evaluation:

- Accuracy: The percentage of profiles correctly classified as real or fake.
- Precision: The proportion of predicted fake profiles that are actually fake.
- Recall: The proportion of actual fake profiles that are correctly identified by the model.
- F1-score: A harmonic mean of precision and recall, providing a balanced view of model performance.

Evaluation Techniques:

- Train-Test Split: During data preprocessing, the dataset is typically split into training and testing sets. The model is trained on the training set and evaluated on the unseen testing set to assess its generalization ability (performance on data it hasn't seen before).
- Cross-Validation: To obtain a more robust evaluation, techniques like k-fold cross-validation can be employed. Here, the data is divided into k folds, and the model is trained and evaluated k times, each time using a different fold for testing. The average performance across all folds provides a more reliable estimate of the model's generalizability.

Additional Considerations:

- Visualization Techniques: Techniques like confusion matrices can be used to visualize the model's performance breakdown. This helps identify areas for improvement, such as cases where the model is frequently misclassifying real profiles as fake or vice versa.
- Class Imbalance: If your dataset has a significant imbalance between real and fake profiles (e.g., many more real profiles than fake ones), it can impact the model's performance. Techniques like oversampling (duplicating minority class examples) or undersampling (removing majority class examples) can be explored to address this issue.

- **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. This module focuses on:

- Model Deployment: Save the trained model in a format that can be used for inference (making predictions on new, unseen data). Popular options include TensorFlow SavedModel or ONNX.
- Real-World Testing: Test the model on unseen data to assess its performance in a practical setting. This might involve collecting new user profile data from Instagram or other sources and feeding it into the model to predict the likelihood of being fake.
- API Integration (Optional): If you're aiming to integrate the model into a larger application or platform, consider developing an API (Application Programming Interface) that allows other systems to interact with your model and receive predictions.

Continuous Improvement:

Machine learning models are not static. As you collect more data and gain insights from real-world usage, you can continuously improve your model by:

- Retraining: Re-train the model on a larger dataset or with updated data to improve its accuracy over time.
- Fine-tuning: Fine-tune the model's hyperparameters or explore alternative network architectures to potentially enhance performance.

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

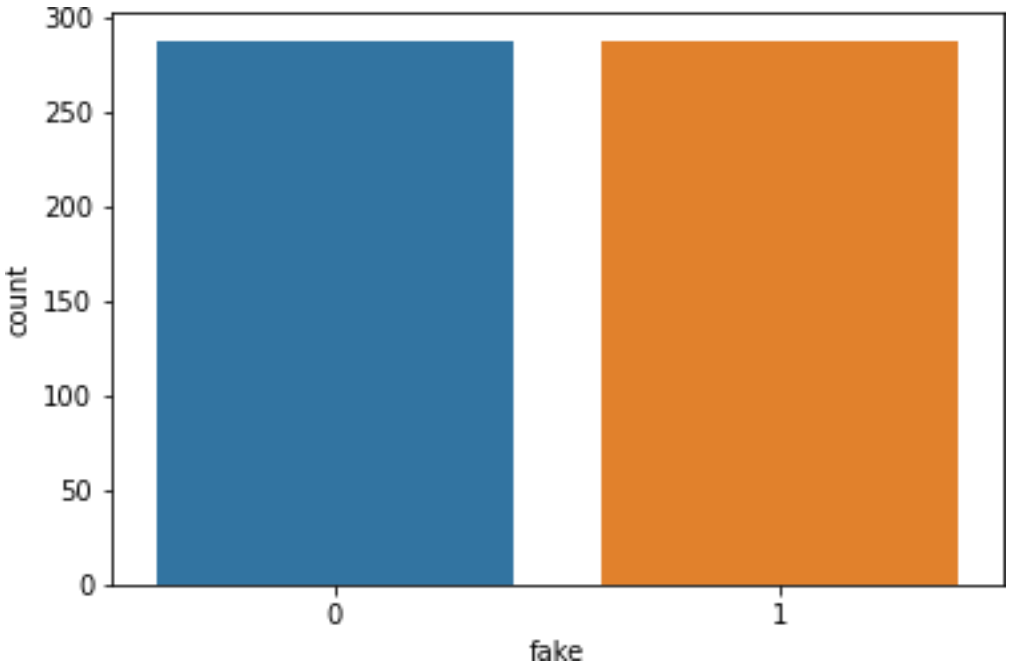
Load Data (Pre-processing)

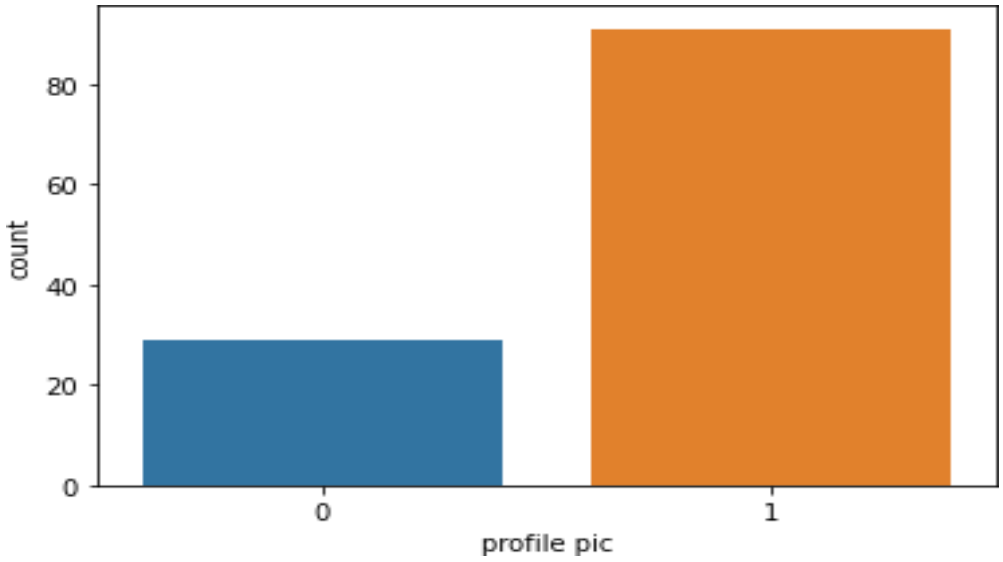
```
pd.read_csv(test_data_path)
```

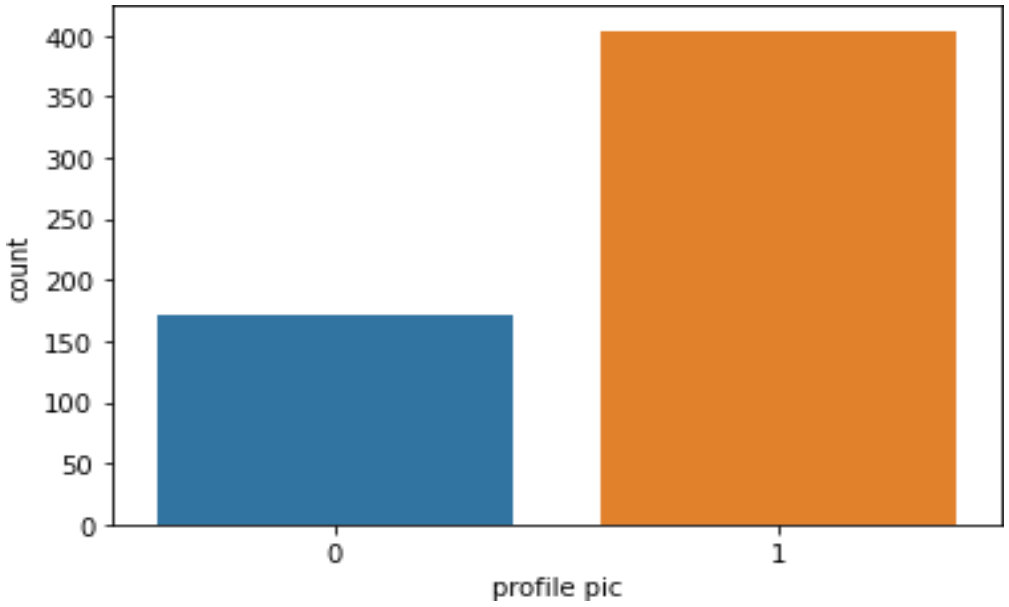
	profile pic	nums/length username	fullname words	nums/length fullname	name==username	description length	external URL	private	#posts	#followers	#follows	fake
0	1	0.33	1	0.33	1	30	0	1	35	488	604	0
1	1	0.00	5	0.00	0	64	0	1	3	35	6	0
2	1	0.00	2	0.00	0	82	0	1	319	328	668	0
3	1	0.00	1	0.00	0	143	0	1	273	14890	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

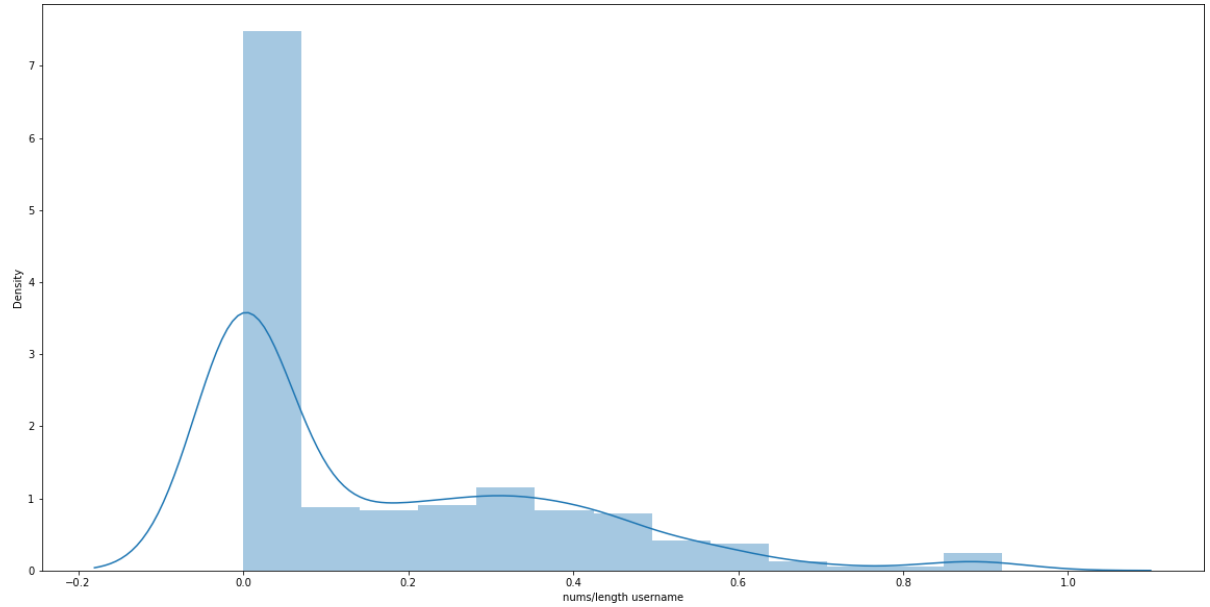
Bar Plot – Visualization (Data Insights)



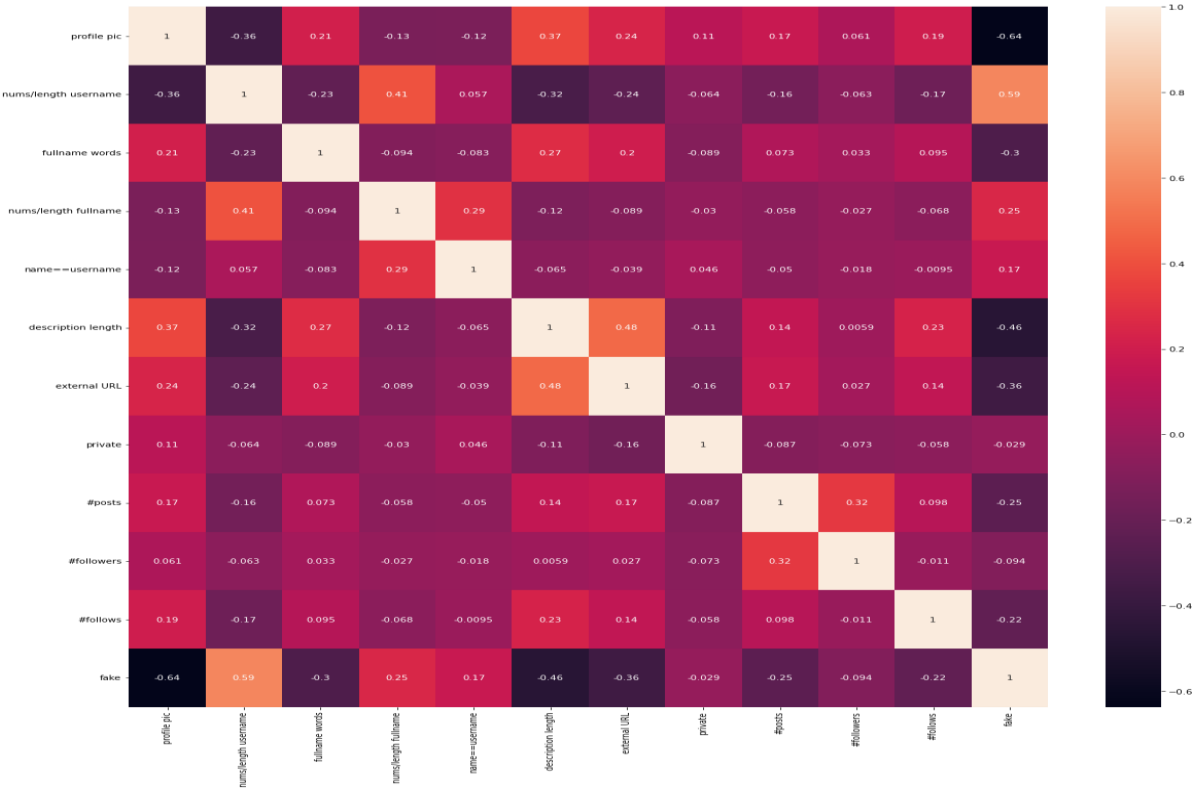

80
60
40
20
0
count
0
1
profile pic


400
350
300
250
200
150
100
50
0
count
0
1
profile pic

KDE Plot (Data Insights)



Heat Map – Correlation Check (Data Insights)

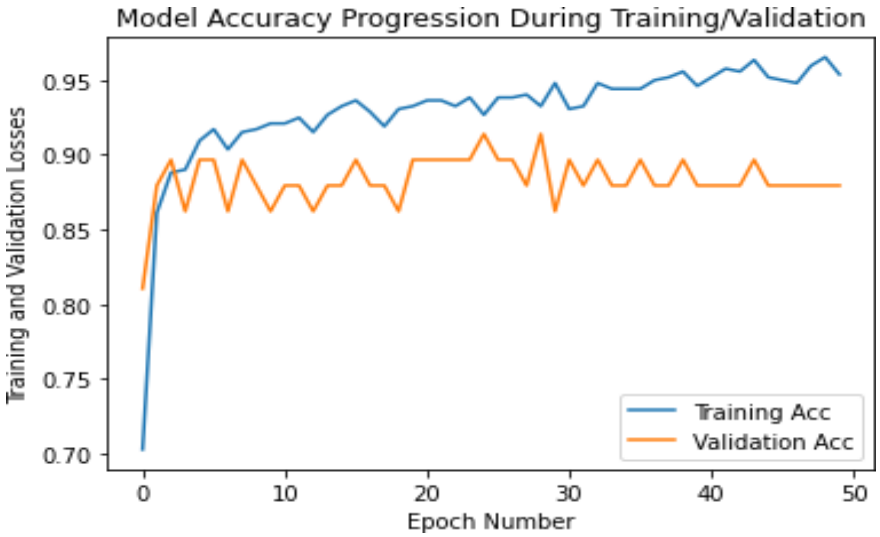


Model Training- (Sequential Training)

```
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)

Training Progress - Accuracy(Training)



Classification Report (Evaluation)

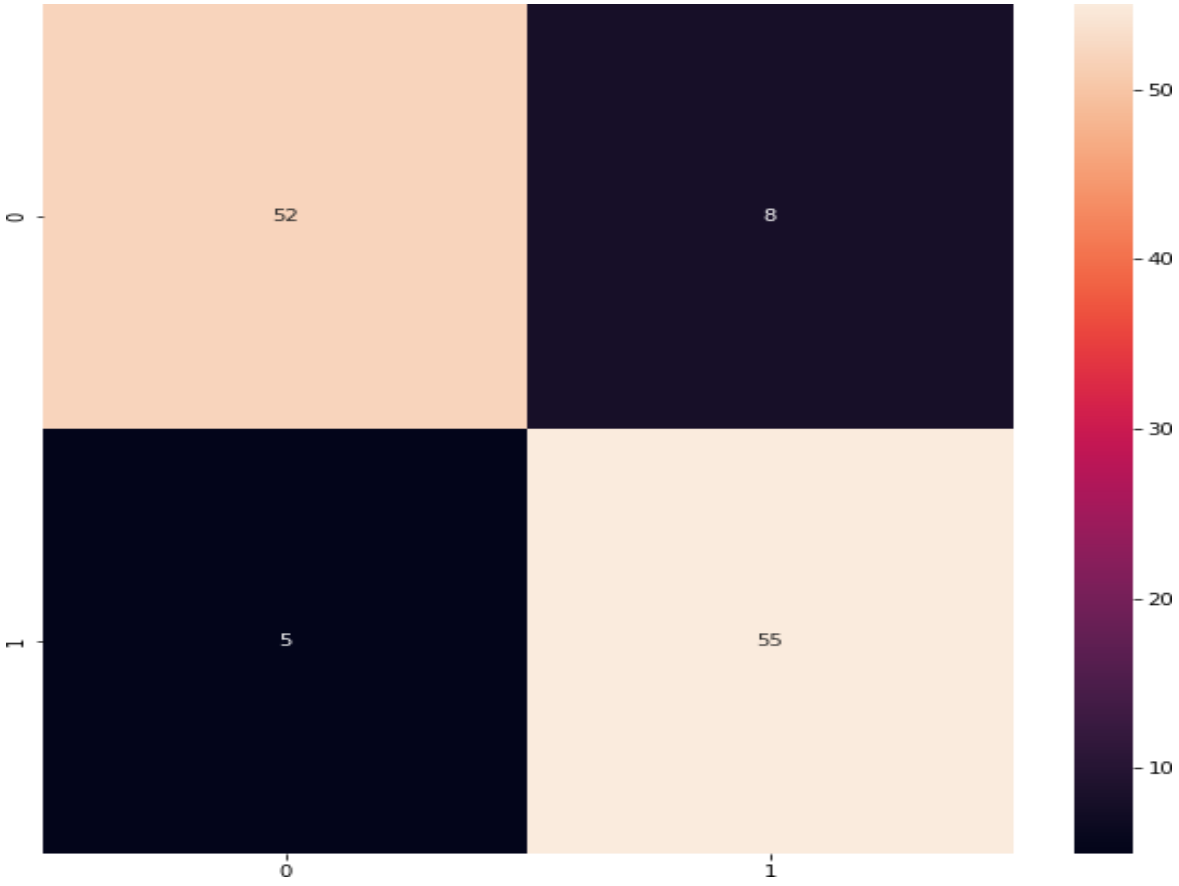
```
print("Accuracy : ", get_avg(model_training_progress['Accuracy']) * 100)

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
```

Confusion Matrix (Evaluation)



CONCLUSION

This project proposes leveraging deep learning, specifically Artificial Neural Networks (ANNs), for enhanced detection of fake social media profiles. By analyzing a well-distributed dataset of user activity attributes, the framework aims to learn patterns indicative of suspicious behavior. This allows the model to predict the probability of an account being fake with improved accuracy.

Enhancing Detection with Multimodal Deep Learning:
The proposed approach can be further strengthened by incorporating additional data sources beyond user activity. Here's how:
Scraping More Metadata: Extracting visual features like images, posts, captions, and even time spent on the platform can provide valuable insights.
Ensemble Deep Learning Models: Utilizing a combination of deep learning models, such as Convolutional Neural Networks (CNNs) for image analysis and Recurrent Neural Networks (RNNs) for text analysis (captions and posts), creates a multimodal approach. This allows the model to leverage the strengths of each model for a more comprehensive understanding of a profile's authenticity.

Furthermore by integrating deep learning and multimodal analysis, this project offers a powerful framework for detecting fake social media profiles. The ability to learn from diverse data sources like user activity, visuals, and textual content empowers the model to make more informed predictions. This comprehensive approach paves the way for a safer and more trustworthy online environment.

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 - 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-Detection-main/insta_train.csv'

test_data_path = 'datasets/Fake-Instagram-Profile-Detection-main/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 = []
```