**FAKE PROFILE DETECTION IN SOCIAL MEDIA PLATFORMS**

*Prasanna Kumar Jeyaraman*

*1001848479*

*IE 6318- DATA MINING*

**ABSTRACT:**

Identity is a thing that is separate from a person but related to them. The name of a person is a typical illustration. Another illustration is a passport, which includes the holder's name, nationality, date of birth, place of birth, digitally collected fingerprints, and an image. A private and public key that adheres to a Public Key Infrastructure is a third illustration. Identity should, in general, be distinct in the sense that each identifying object should only be used to refer to one individual at most. False identities are crucial to advanced persistent threats (APT), which are coordinated, persistent, and difficult attempts to compromise targets in commercial, non-profit, and governmental organizations. False identities are frequently used in spamming, inflating the number of users in an app to increase its popularity, and other harmful behaviors. A typical scenario for the use of false identities involves using social media platforms to impersonate someone or create a false identity to build trust with the target, which is then exploited: for information gathering for spear phishing attacks, for spear phishing attack deployment, or for directly interacting to obtain the information of interest. All social media sites have an increasing number of bogus profiles, which is detrimental on many different levels. Social media platforms are populated by tens of millions of bogus users, endangering the security and privacy of users. To improve user experience and safety and to identify false users in these networks, it is crucial. From this perspective, the verification of an authentic profile is gaining more and more attention. Someone's psychological well-being can be harmed by false information and impersonation. Even while this issue mostly impacts teenagers, impersonation and false accounts have a substantial impact on cyberbullying, and consequently, on the psychological health and mental stability of teenagers.

**INTRODUCTION:**

**OBJECTIVE:**

All social media sites have an increasing number of bogus profiles, which is detrimental on many different levels. From this perspective, the verification of an authentic profile is gaining more and more attention. Someone's psychological well-being can be harmed by false information and impersonation. Even while this issue mostly impacts teenagers, impersonation and false accounts have a substantial impact on cyberbullying, and consequently, on the psychological health and mental stability of teenagers. Social media platforms are populated by tens of millions of bogus users, endangering the security and privacy of users. To improve user experience and safety and to identify false users in these networks, it is crucial.

**LITERATURE REVIEW:**

Facebook, one of the most prominent social media platforms, has approximately 1,8 billion users as of this writing. According to Facebook's annual report, between 5,5% and 11,2% of monthly active users globally in 2013–2014 were false (duplicate, unwanted, etc.). (Facebook, 2014). In their 2014 article, Adikari and Dutta describe how to spot fake LinkedIn profiles. Using only limited profile data as input, the paper demonstrates that fake profiles can be identified with 84% accuracy and 2.44% false negatives. Principal component analysis, SVMs, and neural networks are used as techniques. Features like the number of languages a person speaks, education, skills, recommendations, interests, and awards are used, among others. As a starting point, characteristics of profiles that are known to be fake and posted on particular websites are used.

The goal of Chu et al. (2010) is to distinguish between Twitter accounts run by humans, bots, and cyborgs (i.e., humans and bots collaborating). The detection of spamming accounts is accomplished using an Orthogonal Sparse Bigram (OSB) text classifier that uses pairs of words as features as part of the formulation of the detection issue. The algorithm was able to correctly differentiate between the bots and the human-operated accounts with the help of other detecting components that evaluated the regularity of tweets and some account characteristics like the frequency and types of URLs and the use of APIs. The aim of the Lee et al. (2010) study was also to identify spam accounts on Twitter and MySpace. The list of features in this analysis was increased in comparison to Chu et al.'s study to include both the quantity and type of linkages. The Decorate metaclassifier was found to have the best classification accuracy after several classifiers from the Weka machine learning package were tested. Z. Yang et al. (2011) used a similar method, albeit with a much smaller collection of features, to find bogus Renren accounts. In order to reflect the characteristics of the social graphs, the clustering coefficient was used as a metric. These characteristics were utilized to create an SVM classifier that produced classifications that were 99% accurate.

According to Krombholz et al. (2015), social engineering attacks can be divided into physical techniques (like dumpster diving), social approaches (depending on socio-psychological techniques), reverse social engineering (attacker tries to make victim think she is a trustworthy entity with the intention of getting victim to approach attacker for assistance), technical approaches, and socio-technical approaches. Unfortunately, the techniques used to detect crowdturfing require the presence of a large-scale activity and, like the procedures used to detect spamming campaigns above, are scarcely able to identify a small-footprint action carried out as part of a targeted attack.

**DATA IDENTIFICATION:**

Preparing the data is the initial step in any machine learning problem. the fake profile database is compiled using Kaggle. We'll be using a Kaggle dataset for this issue. The CSV file from Kaggle includes data on 786 actual users and fake users of the social networking platform Instagram, along with 13 other factors. The data contains a variety of varibles in it. The "Profile Pic" field is one variable that shows whether or not each user's account is linked to a profile photo. Another option is "Nums/Length Username," which can refer to either the length of a user's username string or the amount of numeric characters it contains.The number of words in each user's full name is also indicated by a variable called "Fullname Words," which is present. Either the length of the complete name string or the number of numeric characters can be represented by the "Nums/Length Fullname" variable.

The "Name/Username" field could reveal whether a user is employing a pseudonym or whether their name matches their username. The length of each user's profile description or bio is indicated by the "Description Length" variable. Whether a person has included a website or link in their profile is shown by the "External URL" variable. A user's private or public account status is indicated by the "Private" variable.Additional user activity-related variables include "Posts," which denotes how many posts each user has made on the platform, "Followers," which denotes how many followers each user has, and "Follows," which denotes how many accounts each user is following.The final variable, "Fake," specifies whether or not a user account has been classified as fraudulent.

A picture containing text, screenshot, number, font

Description automatically generated

**DATA VISUALIZATION:**

The data can be used to train a machine-learning model once it has been collected and cleaned. The Gaussian Naïve Bayes and KNN with Manhattan distance Classifier will all be trained using this cleaned data. To assess the models' quality, a confusion matrix will be used. Once the three models have been trained, we will combine their predictions to identify fake profiles from given set of social media profile data. This strengthens and improves the accuracy of our total prediction.

The pandas library, which is used for data analysis and manipulation, is first imported into the code. The train\_test\_split function from Sklearn's model\_selection module is then imported, which is used to divide a dataset into training and testing sets. Each feature in this dataset only comprises 0s and 1s, and the dataset is free of null values. Before classifying anything, I made sure the target column is balanced. I'm using a bar plot to see if the dataset is balanced or not.

A blue and orange squares with white text

Description automatically generated with low confidence

*Figure 1*

Using seaborn's countplot() method, a count plot was produced as the first visualization (Figure 1). By counting the occurrences of each distinct value in a categorical variable, the count plot illustrates the distribution of observations in the dataset. Assuming that the categorical variable is phony in this instance, the count plot displays the proportion of observations that are fake in comparison to those that are not. Users can comprehend the ratio of fake to actual observations using the graphic that is produced, and they can also spot any dataset imbalances.

A picture containing diagram, plot, line, screenshot

Description automatically generated

*Figure 2*

A distribution plot was produced in the second visualization (Figure 2) using the distplot() function of seaborn. The distribution plot creates a histogram of the variable's values to show the distribution of a continuous variable. The distribution plot displays the distribution of the ratio of numbers to length of each observation's username in this case where the continuous variable is nums/length username. Users can comprehend the overall distribution of the variable and any patterns or trends in the data by using the visualization that is produced.

A picture containing pattern, colorfulness, square, rectangle

Description automatically generated

*Figure 3*

The heatmap in the above plot illustrates the multicollinearity of the features, which is being checked to see which features should be dropped in order to improve the accuracy of false profile deduction. This visualization can provide valuable insights into the relationships between the different variables in the dataset, helping users identify which variables are strongly or weakly correlated with each other. This information can be used to make informed decisions about which variables to include or exclude in further analysis.

**CLASSIFICATION MODELS:**

After removing Null values and converting the labels to numerical representations, I divided the data into training and testing groups. I organized the data into an 80:20 structure, where 80% of the data will be used to train the model and 20% will be used to evaluate its performance.

We have used the following Machine learning predictive models

1. Gaussian Naïve Bayes
2. KNN with Manhattan distance

**Gaussian Naïve Bayes:**

A screenshot of a computer program

Description automatically generated with low confidence

This code is evaluating a Gaussian Naive Bayes classifier model. The model is trained on the training set (X\_train and y\_train) and then used to make predictions on the test set (X\_test). The predicted values are stored in y\_pred. Next, the code calculates the evaluation metrics for the model on the test set. The accuracy, precision, recall, and F1 score are all calculated using functions from the scikit-learn library. These metrics provide insight into how well the model is able to classify instances correctly.

Finally, the evaluation metrics are printed to the console. The accuracy indicates the proportion of correctly classified instances, while precision, recall, and F1 score provide information about the model's ability to identify true positives, avoid false positives, and balance precision and recall. Overall, this code provides an efficient way to evaluate the performance of a Gaussian Naive Bayes classifier on a given dataset. The output of the evaluation metrics can be used to compare the performance of different models or to tune hyperparameters for improved performance.

*Output:*

A screenshot of a computer

Description automatically generated with medium confidence

This output shows the evaluation metrics for a Gaussian Naive Bayes classifier model. The model has been trained on a dataset and tested on a separate test dataset.

The accuracy of the model is 0.7083, which means that the model has classified 70.83% of instances correctly. The precision of the model is 0.625, which means that out of all the instances that the model has classified as positive, only 62.5% of them are actually positive. The recall of the model is 0.909, which means that out of all the actual positive instances in the test dataset, the model has correctly identified 90.9% of them.

The F1 score is a weighted harmonic mean of precision and recall, and it provides a balanced measure of the model's performance. The F1 score for this model is 0.7407, which is a reasonable performance for a classifier model. Overall, this output indicates that the Gaussian Naive Bayes classifier is moderately effective at classifying instances correctly on this particular dataset, with room for improvement. The specific values of the evaluation metrics can be used to compare the performance of different models or to fine-tune the parameters of the model.

**KNN with Manhattan distance:**

**A screenshot of a computer code

Description automatically generated with medium confidence**

The code above trains a KNN (k-nearest neighbors) classifier on the given dataset and tests its accuracy using several metrics. First, the code initializes a KNN classifier with 5 neighbors and a Manhattan distance metric using the KNeighborsClassifier() function from scikit-learn's neighbors module. The classifier is then trained on the training set using the fit() function, where X\_train contains the input features and y\_train contains the corresponding output labels. Next, the trained classifier is used to predict the labels of the test set using the predict() function, and the accuracy of the predictions is measured using four different metrics: accuracy, precision, recall, and F1 score. These metrics are computed using the accuracy\_score(), precision\_score(), recall\_score(), and f1\_score() functions from scikit-learn's metrics module, respectively.

Finally, the code creates a confusion matrix using the confusion\_matrix() function from scikit-learn's metrics module, which shows the number of true positives, true negatives, false positives, and false negatives for the test set predictions. The confusion matrix is then printed to the console. Overall, this code demonstrates how to train and test a KNN classifier on a given dataset, and provides a number of metrics to evaluate the accuracy of the classifier's predictions. The resulting scores and confusion matrix can be used to assess the performance of the model and identify any areas where it may need to be improved.

*Output:*

A screenshot of a computer

Description automatically generated with medium confidence

The train\_score of the trained model is 0.8854, which means that the model was able to predict the correct labels for 88.54% of the samples in the training set. The test\_score of the trained model is 0.9167, which means that the model was able to predict the correct labels for 91.67% of the samples in the test set. These scores suggest that the model is able to generalize well to new, unseen data.

The confusion matrix shows the number of true positives, false positives, true negatives, and false negatives for the test set predictions. The first row of the confusion matrix shows that there were 12 true negatives and 1 false positive, while the second row shows that there was 1 false negative and 10 true positives. This indicates that the model is able to accurately identify both the positive and negative classes, with only a small number of misclassifications. Overall, the output provides valuable information on the performance of the KNN classifier and can be used to evaluate the accuracy of the model's predictions and identify any areas where the model may need to be improved.

**RESULTS:**

|  |  |
| --- | --- |
| **Prediction Model** | **Accuracy Rate** |
| Gaussian Naïve Bayes | 70.8% |
| KNN with Manhattan distance | 91.6% |

As demonstrated above, this prediction model is validated using three different classifiers. It can be seen clearly that KNN performs better than Gaussian model.

**CONCLUSION:**

* It is possible to train a machine learning model to estimate the possibility that a social media account is fraudulent using these and other pertinent information.
* The development of an accurate fake social media account detection algorithm, however, would probably necessitate a sizable and varied dataset of actual and fake accounts, as well as rigorous feature engineering and model selection.
* Additionally, it would be crucial to keep the model updated as phony accounts develop and alter over time.

**REFERENCE:**

[1] Krombholz, K., Hobel, H., Huber, M., Weippl, E., 2015. Advanced Social Engineering Attacks. J Inf Secur Appl 22, 113–122. doi:10.1016/j.jisa.2014.09.005.

[2] Krombholz, K., Merkl, D., Weippl, E., 2012. Fake identities in social media: A case study on the sustainability of the Facebook business model. J. Serv. Sci. Res. 4, 175–212. doi:10.1007/s12927-012-0008-z.

[3] Chu, Z., Gianvecchio, S., Wang, H., Jajodia, S., 2010. Who is Tweeting on Twitter: Human, Bot, or Cyborg?, in: Proceedings of the 26th Annual Computer Security Applications Conference, ACSAC ’10. ACM, New York, NY, USA, pp. 21–30. doi:10.1145/ 1920261.1920265.

[4] Zang, W., Zhang, P., Wang, X., Shi, J., Guo, L., 2013. Detecting Sybil Nodes in Anonymous Communication Systems. Procedia Comput. Sci., First International Conference on Information Technology and Quantitative Management 17, 861–869. doi:10.1016/ j.procs.2013.05.110.