**IMPLEMENTATION**

**MODULES:**

* Data Collection
* Dataset
* Data Preparation
* Model Selection
* Analyze and Prediction
* Accuracy on test set
* Saving the Trained Model

**MODULES DESCSRIPTION:**

**Data Collection:**

In the first module of Fake Profile Detection on Social Networking, we developed the system to get the input dataset. Data collection process is the first real step towards the real development of a machine learning model, collecting data. This is a critical step that will cascade in how good the model will be, the more and better data that we get; the better our model will perform. There are several techniques to collect the data, like web scraping, manual interventions. Our dataset is placed in the project and it’s located in the model folder. The dataset is referred from the popular standard dataset repository kaggle where all the researchers refer it. The dataset consists of about Instagram account. The following is the URL for the dataset referred from kaggle.

Link: <https://www.kaggle.com/datasets/jayaprakashpondy/instagram-fake-profile>

**Dataset:**

The dataset consists of 576 individual data. There are 12 columns in the dataset, which are described below.

Profile pic: user has profile picture or not

Nums/length username: ratio of number of numerical chars in username to its length

Fullname words: full name in word tokens

Nums/length fullname: ratio of number of numerical characters in full name to its length

Name==username: Are username and full name literally the same

Description length: Bio length in characters

External URL: Has external URL or not

Private: Private or not

#Posts: Number of posts

#Followers: Number of followers

#Follows - Number of follows.

Fake: Yes or No

**Data Preparation:**

Wrangle data and prepare it for training. Clean that which may require it (remove duplicates, correct errors, deal with missing values, normalization, data type conversions, etc.). Randomize data, which erases the effects of the particular order in which we collected and/or otherwise prepared our data. Visualize data to help detect relevant relationships between variables or class imbalances (bias alert!), or perform other exploratory analysis. Split into training and evaluation sets

**Two model used :**

* Random Forest Classifier
* Decision Tree Classifier

**1. Random Forest Classifier**

**Model Selection:**

We used Random Forest Classifier machine learning algorithm, We got a accuracy of 100% on train set so we implemented this Aalgorithm.

**The Random Forests Algorithm**

Let’s understand the algorithm in layman’s terms. Suppose you want to go on a trip and you would like to travel to a place which you will enjoy.

So what do you do to find a place that you will like? You can search online, read reviews on travel blogs and portals, or you can also ask your friends.

Let’s suppose you have decided to ask your friends, and talked with them about their past travel experience to various places. You will get some recommendations from every friend. Now you have to make a list of those recommended places. Then, you ask them to vote (or select one best place for the trip) from the list of recommended places you made. The place with the highest number of votes will be your final choice for the trip.

In the above decision process, there are two parts. First, asking your friends about their individual travel experience and getting one recommendation out of multiple places they have visited. This part is like using the decision tree algorithm. Here, each friend makes a selection of the places he or she has visited so far.

The second part, after collecting all the recommendations, is the voting procedure for selecting the best place in the list of recommendations. This whole process of getting recommendations from friends and voting on them to find the best place is known as the random forests algorithm.

It technically is an ensemble method (based on the divide-and-conquer approach) of decision trees generated on a randomly split dataset. This collection of decision tree classifiers is also known as the forest. The individual decision trees are generated using an attribute selection indicator such as information gain, gain ratio, and Gini index for each attribute. Each tree depends on an independent random sample. In a classification problem, each tree votes and the most popular class is chosen as the final result. In the case of regression, the average of all the tree outputs is considered as the final result. It is simpler and more powerful compared to the other non-linear classification algorithms.

*How does the algorithm work?*

It works in four steps:

Select random samples from a given dataset.

Construct a decision tree for each sample and get a prediction result from each decision tree.

Perform a vote for each predicted result.

Select the prediction result with the most votes as the final prediction.

*Finding important features*

Random forests also offer a good feature selection indicator. Scikit-learn provides an extra variable with the model, which shows the relative importance or contribution of each feature in the prediction. It automatically computes the relevance score of each feature in the training phase. Then it scales the relevance down so that the sum of all scores is 1.

This score will help you choose the most important features and drop the least important ones for model building.

Random forest uses gini importance or mean decrease in impurity (MDI) to calculate the importance of each feature. Gini importance is also known as the total decrease in node impurity. This is how much the model fit or accuracy decreases when you drop a variable. The larger the decrease, the more significant the variable is. Here, the mean decrease is a significant parameter for variable selection. The Gini index can describe the overall explanatory power of the variables.

**Analyze and Prediction:**

In the actual dataset, we chose only 11 features:

**Profile pic:** user has profile picture or not

**Nums/length username:** ratio of number of numerical chars in username to its length

**Fullname words:** full name in word tokens

**Nums/length fullname:** ratio of number of numerical characters in full name to its length

**Name==username:** Are username and full name literally the same

**Description length:** Bio length in characters

**External URL:** Has external URL or not

**Private:** Private or not

**#Posts:** Number of posts

**#Followers:** Number of followers

**Fake :** Yes or No

**Accuracy on test set:**

After training and evaluating the model on the validation set, the accuracy of the model will be assessed on the test set. The accuracy on the test set will be an important metric for evaluating the model's performance. We got an accuracy of 93% on test set.

**Saving the Trained Model:**

Once you’re confident enough to take your trained and tested model into the production-ready environment, the first step is to save it into a .h5 or .pkl file using a library like pickle.

Make sure you have pickle installed in your environment.

Next, let’s import the module and dump the model into .pkl file.

**2.** **Decision Tree Classifier**

**Model Selection:**

We used decision tree classifier machine learning algorithm, We got a accuracy of 92% on train set so we implemented this algorithm.

**Decision Tree Classification Algorithm**

Decision Tree is a Supervised learning technique that can be used for both classification and Regression problems, but mostly it is preferred for solving Classification problems. It is a tree-structured classifier, where internal nodes represent the features of a dataset, branches represent the decision rules and each leaf node represents the outcome.

In a Decision tree, there are two nodes, which are the Decision Node and Leaf Node. Decision nodes are used to make any decision and have multiple branches, whereas Leaf nodes are the output of those decisions and do not contain any further branches.

The decisions or the test are performed on the basis of features of the given dataset.

It is a graphical representation for getting all the possible solutions to a problem/decision based on given conditions.

It is called a decision tree because, similar to a tree, it starts with the root node, which expands on further branches and constructs a tree-like structure.

In order to build a tree, we use the CART algorithm, which stands for Classification and Regression Tree algorithm.

A decision tree simply asks a question, and based on the answer (Yes/No), it further split the tree into sub-trees.

Below diagram explains the general structure of a decision tree:

Note: A decision tree can contain categorical data (YES/NO) as well as numeric data.



*Why use Decision Trees?*

There are various algorithms in Machine learning, so choosing the best algorithm for the given dataset and problem is the main point to remember while creating a machine learning model. Below are the two reasons for using the Decision tree:

Decision Trees usually mimic human thinking ability while making a decision, so it is easy to understand.

The logic behind the decision tree can be easily understood because it shows a tree-like structure.

*How does the Decision Tree algorithm Work?*

In a decision tree, for predicting the class of the given dataset, the algorithm starts from the root node of the tree. This algorithm compares the values of root attribute with the record (real dataset) attribute and, based on the comparison, follows the branch and jumps to the next node.

For the next node, the algorithm again compares the attribute value with the other sub-nodes and move further. It continues the process until it reaches the leaf node of the tree. The complete process can be better understood using the below algorithm:

Step-1: Begin the tree with the root node, says S, which contains the complete dataset.

Step-2: Find the best attribute in the dataset using Attribute Selection Measure (ASM).

Step-3: Divide the S into subsets that contains possible values for the best attributes.

Step-4: Generate the decision tree node, which contains the best attribute.

Step-5: Recursively make new decision trees using the subsets of the dataset created in step -3. Continue this process until a stage is reached where you cannot further classify the nodes and called the final node as a leaf node.

Example: Suppose there is a candidate who has a job offer and wants to decide whether he should accept the offer or Not. So, to solve this problem, the decision tree starts with the root node (Salary attribute by ASM). The root node splits further into the next decision node (distance from the office) and one leaf node based on the corresponding labels. The next decision node further gets split into one decision node (Cab facility) and one leaf node. Finally, the decision node splits into two leaf nodes (Accepted offers and Declined offer). Consider the below diagram:



*Attribute Selection Measures*

While implementing a Decision tree, the main issue arises that how to select the best attribute for the root node and for sub-nodes. So, to solve such problems there is a technique which is called as Attribute selection measure or ASM. By this measurement, we can easily select the best attribute for the nodes of the tree. There are two popular techniques for ASM, which are:

Information Gain

Gini Index

*1. Information Gain:*

Information gain is the measurement of changes in entropy after the segmentation of a dataset based on an attribute.

It calculates how much information a feature provides us about a class.

According to the value of information gain, we split the node and build the decision tree.

A decision tree algorithm always tries to maximize the value of information gain, and a node/attribute having the highest information gain is split first. It can be calculated using the below formula:

Information Gain= Entropy(S)- [(Weighted Avg) \*Entropy(each feature)

Entropy: Entropy is a metric to measure the impurity in a given attribute. It specifies randomness in data. Entropy can be calculated as:

Entropy(s)= -P(yes)log2 P(yes)- P(no) log2 P(no)

Where,

S= Total number of samples

P(yes)= probability of yes

P(no)= probability of no

*2. Gini Index:*

Gini index is a measure of impurity or purity used while creating a decision tree in the CART(Classification and Regression Tree) algorithm.

An attribute with the low Gini index should be preferred as compared to the high Gini index.

It only creates binary splits, and the CART algorithm uses the Gini index to create binary splits.

Gini index can be calculated using the below formula:

Gini Index= 1- ∑jPj2

Pruning: Getting an Optimal Decision tree

Pruning is a process of deleting the unnecessary nodes from a tree in order to get the optimal decision tree.

A too-large tree increases the risk of overfitting, and a small tree may not capture all the important features of the dataset. Therefore, a technique that decreases the size of the learning tree without reducing accuracy is known as Pruning. There are mainly two types of tree pruning technology used:

**Analyze and Prediction:**

In the actual dataset, we chose only 11 features :

**Profile pic:** user has profile picture or not

**Nums/length username:** ratio of number of numerical chars in username to its length

**Fullname words:** full name in word tokens

**Nums/length fullname:** ratio of number of numerical characters in full name to its length

**Name==username:** Are username and full name literally the same

**Description length:** Bio length in characters

**External URL:** Has external URL or not

**Private:** Private or not

**#Posts:** Number of posts

**#Followers:** Number of followers

**Fake :** Yes or No

**Accuracy on test set:**

After training and evaluating the model on the validation set, the accuracy of the model will be assessed on the test set. The accuracy on the test set will be an important metric for evaluating the model's performance. We got an accuracy of 92% on test set.

**Saving the Trained Model:**

Once you’re confident enough to take your trained and tested model into the production-ready environment, the first step is to save it into a .h5 or .pkl file using a library like pickle.

Make sure you have pickle installed in your environment.

Next, let’s import the module and dump the model into .pkl file.