|  |  |  |  |
| --- | --- | --- | --- |
|  |  | **CONTENTS** |  |
| **S.NO** | **NAME** | | **PAGENO** |
|  | **List of Figures** | | i |
|  | **List of Screenshots** | | ii |
|  | **ABSTRACT** | | iii |
| **1** | **INTRODUCTION** | | 1 |
|  | 1.1 | Problem Statement |  |
| **2** | **LITERATURE SURVEY AND METHODOLOGY** | | 3 |
|  | 2.1 | Support Vector Machine |  |
|  | 2.2 | Random Forest |  |
|  | 2.3 | Neural Network |  |
| **3** | **SYSTEM ANALYSIS AND DESCRIPTION** | | 9 |
|  | 3.1 | Existing System |  |
|  | 3.2 | Proposed System |  |
|  | 3.3 | Feasibility Study |  |
| **4** | 3.3.1 Economic Feasibility  3.3.2 Technical Feasibility  3.3.3 Social Feasibility  **MODULES** | | 12 |
|  | 4.1 | Data Collection |  |
|  | 4.2 | Data Preprocessing |  |
|  | 4.3 | Extraction of Feature Set/Training Data |  |
|  | 4.4 | Classification |  |
|  | 4.5 | Confusion Matrix |  |
|  | 4.6 | Supervised Machine Learning |  |
|  |  |  |  |
| **5** | **SOFTWARE REQUIREMENTS SPECIFICATIONS** | | 14 |
|  | 5.1 | Software Requirements |  |
|  | 5.2 | Hardware Requirements |  |
| **6** | **IMPLEMENTATION** | | 16 |
|  | 6.1 Explanation of Attributes | |  |
|  | 6.2 Sample Code | |  |
|  | 6.3 Software Environment | |  |
| **7** | **INPUT AND OUTPUT DESIGN** | | 27 |
|  | 7.1 Input Design | |  |
|  | 7.2 Objectives | |  |
|  | 7.3 Output Design | |  |
| **8** | **ALGORITHMS** | | 29 |

|  |  |  |  |
| --- | --- | --- | --- |
| **9** | **TESTING** | | 45 |
|  | 9.1 | Types of testing |  |
|  |  | 9.1.1 Unit Testing |  |
|  |  | 9.1.2 Integration Testing |  |
|  |  | 9.1.3 Functional Testing |  |
|  |  | 9.1.4 System Testing |  |
|  |  | 9.1.5 White Box Testing |  |
|  |  | 9.1.6 Black Box Testing |  |
|  | 9.2 | Test strategy and approaches |  |
|  |  | 9.2.1 Test Objectives |  |
|  |  | 9.2.2 Integration Testing |  |
|  |  | 9.2.3 Acceptance Testing |  |
| **10** | **EXPERIMENTAL RESULTS AND DISCUSSION** | | 48 |
|  | 10.1 Random Forest Algorithm | |  |
|  | 10.2 Support Vector Machine | |  |
|  | 10.3 Neural Network | |  |
| **11** | **OUTPUT SCREENS** | | 52 |
| **12** | **CONCLUSION** | | 58 |
| **13** | **FUTURE WORK** | | 59 |
| **11** | **BIBLIOGRAPHY** | | 60 |

**LIST OF FIGURES**

|  |  |  |  |
| --- | --- | --- | --- |
| **S.NO** | **FIGURE NAME** | **PAGE NUMBERS** |  |
|  |  |
|  |  |  |  |
| 1 | MACHINE LEARNING PROCESS | 6 |  |
|  |  |  |  |
| 2 | PROPOSED METHODOLOGY | 6 |  |
|  |  |  |  |
| 3 | ATTRIBUTES THAT DEFINE A DATASSET | 16 |  |
|  |  |  |  |
| 4 | DJANGO FRAMEWORK | 25 |  |
|  |  |  |  |
| 5 | VISUAL STUDIO INTERFACE | 26 |  |
|  |  |  |  |
| 6 | WORKING OF RANDOM FOREST ALGORITHM | 29 |  |
|  |  |  |  |
| 7 | SVM WORKING EXAMPLE | 35 |  |
|  |  |  |  |
| 8 | NEURAL NETWORK LAYERS | 40 |  |
|  |  |  |  |
| 9 | ANN WORKING | 42 |  |
|  |  |  |  |

**LIST OF SCREENSHOTS**

|  |  |  |
| --- | --- | --- |
| **S.NO** | **SCREEN SHOT NAME** | **PAGE NUMBERS** |
|  |  |  |
| 1 | DATASET SAMPLE | 52 |
|  |  |  |
| 2 | EXTRACTED COLUMNS/FEATURES | 52 |
|  |  |  |
| 3 | CONFUSION MATRIX WITHOUT NORMALIZATION IN NN | 53 |
|  |  |  |
| 4 | NORMALIZED CONFUSION MATRIX IN NN | 53 |
|  |  |  |
| 5 | ROC CURVE IN NN | 54 |
|  |  |  |
| 6 | CONFUSION MATRIX WITHOUT NORMALIZATION IN RANDOM FOREST | 54 |
|  |  |  |
| 7 | NORMALIZED CONFUSION MATRIX IN RANDOM FOREST | 55 |
|  |  |  |
| 8 | ROC CURVE IN RANDOM FOREST | 55 |
|  |  |  |
| 9 | SVM LEARNING CURVE | 56 |
|  |  |  |
| 10 | CONFUSION MATRIX WITHOUT NORMALIZATION IN SVM | 56 |
|  |  |  |
| 11 | NORMALIZED CONFUSION MATRIX IN SVM | 57 |
|  |  |  |
| 12 | ROC CURVE IN SVM | 57 |
|  |  |  |

**ABSTRACT**

In the present generation, On-Line social networks (OSNs) have become increasingly popular, which impacts people's social lives and impel them to become associated with various social media sites. Social Networks are the essential platforms through which many activities such as promotion, communications, agenda creation, advertisements, and news creation have started to be done. Adding new friends and keeping in contact with them and their updates has become easier. Researchers have been studying these online social networks to see the impact they make on the people. Some malicious accounts are used for purposes such as misinformation and agenda creation. Detection of malicious account is significant. The methods based on machine learning-based were used to detect fake accounts that could mislead people. The dataset is pre-processed using various python libraries and a comparison model is obtained to get a feasible algorithm suitable for the given dataset [2]. An attempt to detect fake accounts on the social media platforms is determined by various Machine Learning algorithms. The classification performances of the algorithms Random Forest, Neural Network and Support Vector Machines are used for the detection of fake accounts.

**CHAPTER-1**

**INTRODUCTION**

Online Social Networks (OSNs), such as Facebook, Twitter and LinkedIn, have become increasingly popular over the last few years. People use OSNs to keep in touch with each other’s, share news, organize events, and even run their own e-business. Facebook community continues to grow with more than 2.2 billion monthly active users and 1.4 billion daily active users, with an increase of 11% on a year-overyear basis. For the purpose to detect fake accounts on the social media platforms the dataset generated was pre-processed and fake accounts were determined by machine learning algorithms.[3] The classification performances of the algorithms Random Forest, Neural Network and Support Vector Machines are used for the detection of fake accounts. The accuracy rates of detecting fake accounts using the mentioned algorithms are compared and the algorithm with the best accuracy rate is noted. These OSN have made a drastic change in the way we pursue our social life. Making new friends, keeping in contact with them and knowing their updates has become easier. But with the rapid growth of social media many problems like fake profiles, online impersonation have also grown. Fake accounts can be either human-generated, computer generated (also referred as “bots”), or cyborgs [1]. A cyborg is half-human, half-bot account [1]. Such an account is manually opened by a human, but from then onwards the actions are automated by a bot. To become member of the OSN the user has to create his profile by entering information like name, photo, date of birth, Email ID, graduation details, place of work, home town, interests and so on [2][3]. Some of the fields are mandatory and some are optional and it varies from one OSN to the another. These websites are popular because of people’s interest in finding friends, sharing pictures, tagging people in group photos, sharing their ideas and views on common topics, maintain good business relationship and general interest with others. In this paper we came up with a framework in which automatic detection of fake profiles is possible and is efficient. This framework uses classification techniques like Support Vector Machine, Random Forest and Deep Neural Networks to classify the profiles into fake or genuine classes. As it is an automatic detection method, it can be applied easily by OSN which has millions of profile where profiles cannot be examined manually .We evaluate whether readily available and engineered features that are used for the successful detection, using machine learning models.

**1.1 PROBLEM STATEMENT**

Nowadays, Online Social Media is dominating the world in several ways. Day by day the number of users using social media is increasing drastically. The main advantage of online social media is that we can connect to people easily and communicate with them in a better way. This provided a new way of a potential attack, such as fake identity, false information, etc. A recent survey suggests that the number of accounts present in the social media is much greater than the users using it. This suggests that fake accounts have been increased in the recent years. Online social media providers face difficulty in identifying these fake accounts. The need for identifying these fake accounts is that social media is flooded with false information, advertisements, etc.

Traditional methods cannot distinguish between real and fake accounts efficiently. Improvement in fake account creation made the previous works outdated. The new models created used different approaches such as automatic posts or comments, spreading false information or spam with advertisements to identify fake accounts. Due to the increase in the creation of the fake accounts different algorithms with different attributes are use. Previously use algorithms like naïve bayes, support vector machine, random forest has become inefficient in finding the fake accounts.

**CHAPTER-2**

**LITERATURE SURVEY**

Sarah Khaled et al. presented a new algorithm, SVM-NN, to provide efficient detection for fake Twitter accounts and bots, feature selection and dimension reduction techniques. This proposed algorithm (SVM-NN) uses less number of features, while still being able to correctly classify about 98% of the accounts of our training dataset [1].

Sreenivas Kumacham et al. proposed a machine learning model to predict the student placements using various Machine Learning algorithms that include J48, Naïve Bayes, Random Forest etc., The model tries to obtain the results from various algorithms and these results are compared to predict the best algorithm for any given dataset.[2]

A literature review is a survey of scholarly sources on a specific topic. It provides an overview of current knowledge, allowing you to identify relevant theories, methods, and gaps in the existing research.

**2.1 WHAT IS MACHINE LEARNING?**

Machine Learning is the field of study that gives computers the capability to learn without being explicitly programmed. ML is one of the most exciting technologies that one would have ever come across. As it is evident from the name, it gives the computer that makes it more similar to humans: The ability to learn. Machine learning is actively being used today, perhaps in many more places than one would expect.

Machine learning is a tool for turning information into knowledge. In the past 50 years, there has been an explosion of data. This mass of data is useless unless we analyze it and find the patterns hidden within. Machine learning techniques are used to automatically find the valuable underlying patterns within complex data that we would otherwise struggle to discover. The hidden patterns and knowledge about a problem can be used to predict future events and perform all kinds of complex decision making.

The terms used in Machine learning are:

**Dataset:** A set of data examples that contain features important to solving the problem.

**Features:** Important pieces of data that help us understand a problem. These are fed in to a MachineLearning algorithm to help it learn.

**Model:** The representation (internal model) of a phenomenon that a Machine Learning algorithm haslearnt. It learns this from the data it is shown during training. The model is the output you get after training an algorithm. For example, a decision tree algorithm would be trained and produce a decision tree model.

**The Process used in Machine learning is of 5 steps:**

**Data Collection:** Collect the data that the algorithm will learn from.

**Data Preparation:** Format and engineer the data into the optimal format, extracting important features and performing dimensionality reduction.

**Training**: Also known as the fitting stage, where the machine learning algorithm actually learns by showing thecollected and prepared data.

**Evaluation:** Test the model to see how well it performs.

**Tuning:** Fine tune the model to maximize its performance.

**Machine Learning Approaches:**

There are many approaches that can be taken when conducting Machine Learning. They are supervised, unsupervised, semi-supervised and reinforcement learning. Each form of Machine Learning has differing approaches, but they all follow the same underlying process and theory. This explanation covers the general Machine Leaning concept and then focuses in on each approach.

Supervised and unsupervised are well established approaches and the most commonly used. Semi-supervised and Reinforcement Learning are newer and more complex but have shown impressive results.

**Supervised Learning:**

In supervised learning, the goal is to learn the mapping (the rules) between a set of inputs and outputs.

.

**Classification:**

Classification is used to group the similar data points into different sections in order to classify them.

Machine Learning is used to find the rules that explain how to separate the different data points.

**Regression:**

Regression is another form of supervised learning. The difference between classification and regression is that regression outputs a number rather than a class. Therefore, regression is useful when predicting number based problems like stock market prices, the temperature for a given day, or the probability of an event.

**Unsupervised Learning:**

In unsupervised learning, only input data is provided in the examples. There are no labelled example outputs to aim for. But it may be surprising to know that it is still possible to find many interesting and complex patterns hidden within data without any labels.

To find the interesting structures in unlabelled data, we use density estimation. The most common form is clustering. Among others, there is also dimensionality reduction, latent variable models and anomaly detection. More complex unsupervised techniques involve neural networks like Auto-encoders and Deep Belief Networks.

**Semi supervised learning:**

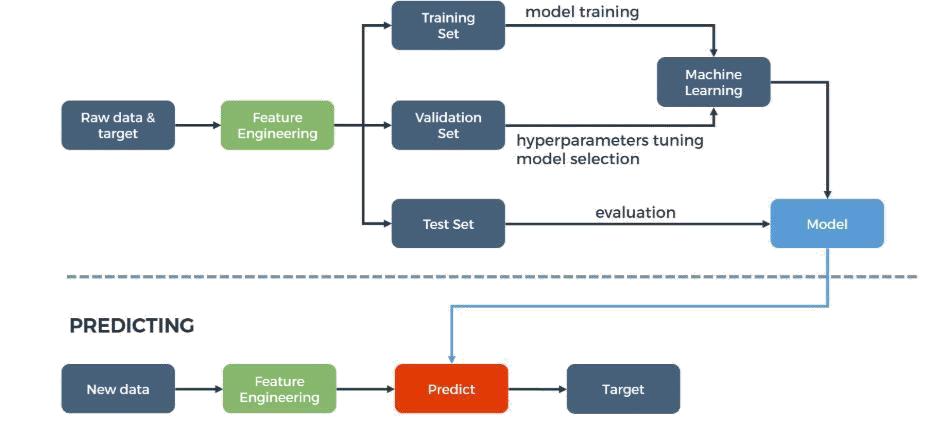
Semi-supervised learning is a mix between supervised and unsupervised approaches. The learning process isn’t closely supervised with example outputs for every single input, but we also don’t let the algorithm do its own thing and provide no form of feedback. Semi-supervised learning takes the middle road.

By being able to mix together a small amount of labelled data with a much larger unlabelled dataset it reduces the burden of having enough labelled data. Therefore, it opens up many more problems to be solved with machine learning. Generative Adversarial Networks (GANs) have been a recent breakthrough with incredible results.

**Reinforcement learning:**

This approach is less common and much more complex, but it has generated incredible results.

In this approach, occasional positive and negative feedback is used to reinforce behaviours.



**Fig Machine learning process.**

**METHODOLOGY**

Proposed system is equipped with various Machine Learning tasks and the architecture followed is as shown below. The proposed system collects the dataset which are preprocessed by providing a framework of algorithms using which we can detect fake profiles in Facebook by comparing the accuracy of three machine learning algorithms and the algorithm with very high efficiency is found for the given dataset.

****

**Fig Proposed Methodology**

The different ways in which an algorithm can model a problem is based on its interaction with the experience or environment for the model preparation process that helps in choosing the most appropriate algorithm for the given input data in order to get the best result.

* 1. **SUPPORT VECTOR MACHINE (SVM)**

Support-vector machines (SVMs, also support-vector networks) are the supervised learning models with associated learning algorithms that analyze data used for classification and regression analysis. For the given labeled training data (supervised learning), the algorithm outputs an optimal hyper plane which categorizes new examples.

“Support Vector Machine” (SVM) is a supervised machine learning algorithm which can be used for both classification and regression challenges. However, it is mostly used in classification problems. In this algorithm, we plot each data item as a point in n-dimensional space (where n is number of features you have) with the value of each feature being the value of a particular coordinate.

Then, we perform classification by finding the hyper-plane that differentiate the two classes very well (look at the below snapshot).Support Vectors are simply the co-ordinates of individual observation. Support Vector Machine is a frontier which best segregates the two classes (hyper-plane/ line).More formally, a support vector machine constructs a hyper plane or set of hyper planes in a high- or infinite-dimensional space, which can be used for classification, regression, or other tasks like outliers detection. Intuitively, a good separation is achieved by the hyper plane that has the largest distance to the nearest training-data point of any class (so-called functional margin), since in general the larger the margin the lower the generalization error of the classifier. Whereas the original problem may be stated in a finite dimensional space, it often happens that the sets to discriminate are not linearly separable in that space. For this reason, it was proposed that the original finite-dimensional space be mapped into a much higher-dimensional space, presumably making the separation easier in that space.

* 1. **NEURAL NETWORKS**

A neural network is a network or circuit of neurons, or in a modern sense, an artificial neural network, composed of artificial neurons or nodes. A neural network (NN), in the case of artificial neurons is an interconnected group of natural or artificial neurons that uses a mathematical model for information.

Neural networks (NNs) can be defined as “The algorithms in machine learning are implemented by using the structure of neural networks. These neural networks model the data using artificial neurons. Neural networks thus mimic the functioning of the brain.” The ‘thinking’ or processing that a brain carries out is the result of these neural networks in action. The Neural networks algorithm tries to improve the performance of the model by using smart computational methods to create new and better performing types of prediction and detection model.

**2.3 RANDOM FOREST**

Random forest algorithm is a supervised classification algorithm. As the name suggest, this algorithm creates the forest with a number of trees. In general, the more trees in the forest the more robust the forest looks like. In the same way in the random forest classifier, the higher the number of trees in the forest gives the high accuracy results. The random forest is a model made up of many decision trees. When training the model using Random forest algorithm, each tree in a random forest learns from a random sample of the data points and the samples drawn with replacement are known as bootstrapping in which some samples will be used multiple times in a single tree.

**CHAPTER-3**

**SYSTEM ANALYSIS AND DESCRIPTION**

**3.1 EXISTING SYSTEM**

The existing systems use very fewer factors to decide whether an account is fake or not. The factors largely affect the way decision making occurs. When the number of factors is low, the accuracy of the decision making is reduced significantly. There is an exceptional improvement in fake account creation, which is unmatched by the software or application used to detect the fake account. Due to the advancement in creation of fake account, existing methods have turned obsolete. The most common algorithm used by fake account detection Applications is the Random forest algorithm. The algorithm has few downsides such as inefficiency to handle the categorical variables which has different number of levels.

## ****DISADVANTAGES****

* Because of Privacy Issues the Facebook dataset is very limited and a lot of details are not made public.

**3.2 PROPOSED SYSTEM**

The proposed system uses random forest, SVM and neural networks algorithm to identify the fake account. It is efficient when it has the correct inputs and when it has all the inputs. When some of the inputs are missing it becomes difficult for the algorithm to produce the output. To overcome such difficulties in the proposed systems we used a gradient boosting algorithm. Random forest algorithm which uses decision trees as its main component. We also changed the way we find the fake accounts i.e., we introduced new methods to find the account. The methods used are spam commenting, engagement rate and artificial activity. These inputs are used to form decision trees that are used in the gradient boosting algorithm. This algorithm gives us an output even if some inputs are missing. This is the major reason for choosing this algorithm. Due to the use of this algorithm we were able to get highly accurate results.

1. Classification starts from the selection of profile that needs to be classified.

2. Once the profile is selected, the useful features are extracted for the purpose of classification.

3. The extracted features are then fed to trained classifier.

4. Classifier is trained regularly as new data is fed into the classifier.

5. Classifier then determines whether the profile is genuine or fake.

6. The result of classification algorithm is then verified and feedback is fed back into the classifier.

7. As the number of training data increases the classifier becomes more and more accurate in predicting the fake profiles.

## ****ADVANTAGES****

* The social networking sites are making our social lives better but nevertheless there are a lot of issues with using these social networking sites.
* The issues are privacy, online bullying, potential for misuse, trolling, etc. These are done mostly by using fake profiles.
* In this project, we came up with a framework through which we can detect a fake profile using machine learning algorithms so that the social life of people become secured.

**3.3 FEASIBILITY STUDY**

The feasibility of the project is analyzed in this phase and business proposal is put forth with a very general plan for the project and some cost estimates. During system analysis the feasibility study of the proposed system is to be carried out. This is to ensure that the proposed system is not a burden to the company. For feasibility analysis, some understanding of the major requirements for the system is essential.

**Three key considerations involved in the feasibility analysis are,**

* ECONOMICAL FEASIBILITY
* TECHNICAL FEASIBILITY
* SOCIAL FEASIBILITY

**3.3.1 Economical Feasibility**

This study is carried out to check the economic impact that the system will have on the organization. The amount of fund that the company can pour into the research and development of the system is limited. The expenditures must be justified. Thus the developed system as well within the budget and this was achieved because most of the technologies used are freely available. Only the customized products had to be purchased.

### 3.3.2 Technical Feasibility

### This study is carried out to check the technical feasibility, that is, the technical requirements of the system. Any system developed must not have a high demand on the available technical resources. This will lead to high demands on the available technical resources. This will lead to high demands being placed on the client. The developed system must have a modest requirement, as only minimal or null changes are required for implementing this system.

**3.3.3 Social Feasibility**

The aspect of study is to check the level of acceptance of the system by the user. This includes the process of training the user to use the system efficiently. The user must not feel threatened by the system, instead must accept it as a necessity. The level of acceptance by the users solely depends on the methods that are employed to educate the user about the system and to make him familiar with it. His level of confidence must be raised so that he is also able to make some constructive criticism, which is welcomed, as he is the final user of the system.

**CHAPTER-4**

**MODULES**

**4.1 DATASET COLLECTION**

Data set of both fake and genuine profiles with various attributes like number of friends, followers, status count. Dataset is divided into training and testing data. Classification algorithm are trained using training dataset and testing data set is used to determine the efficiency of algorithm .From the dataset used 80% of both (real and fake ) are used to prepare a training data set and 20% of both profiles are used to prepare a testing dataset.

**4.2 DATA PREPROCESSING**

Data Preprocessing or Data cleaning, Data is cleansed through processes such as filling in missing values, smoothing the noisy data, or resolving the inconsistencies in the data. And also used to removing the unwanted data. Commonly used as a preliminary data mining practice, data preprocessing transforms the data into a format that will be more easily and effectively processed for the purpose of the user.

**4.3 EXTRACTION OF FEATURE SET/TRAINING DATA**

Features are selected to apply classification algorithms. The classification algorithm is being discussed further. Attributes are selected as features if they are not dependent on other attributes and they increase efficiency of the classification. After selection of attributes, the dataset of profiles that are already classified as fake or genuine are needed for the training purpose of the classification algorithm. We have used a publicly available dataset of 1337 fake users and 1481 genuine users consisting of various attributes including listed count, status count, number of friends, followers count, favourites, languages known, sex code.

**4.4 CLASSIFICATION**

Classification is the process of categorizing a data object into categories called classes based upon features/attributes associated with that data object. Classification uses a classifier, an algorithm that processes the attributes of each data object and outputs a class based upon this information. In this project, we use Support Vector Machine as a classifier.

Support Vector Machine is an elegant and robust technique for classification on a large data set not unlike the data sets of Social Network with several millions of profiles. Algorithms used for classification are Support Vector Machine, Random Forest and Deep Neural Networks.

**4.5 CONFUSION MATRIX**

Confusion Matrix is a technique for describing the performance of a classification algorithm. Confusion Matrix is used to give you a better idea of what your classification model is getting right and what types of errors it is making. All the algorithm results are plotted in confusion matrix to know where the error has occurred.

**4.6 SUPERVISED MACHINE LEARNING**

Miller et al [6] proposed that supervised machine learning models require a label included in the corpus to predict the expected outcome. With unsupervised machine learning the data is unlabeled and data are being grouped based on the similarity of the data considered. It is not practical to search the class consisting of fake accounts. The norm is to train a one class support vector machine on the minority class.

**CHAPTER-5**

**SOFTWARE REQUIREMENTS SPECIFICATIONS**

**5.1 HARDWARE REQUIREMENTS**

* **System :** Pentium IV 2.4 GHz.
* **Hard Disk :** 40 GB.
* **Floppy Drive :** 1.44 Mb.
* **Monitor :** 14’ Colour Monitor.
* **Mouse :** Optical Mouse
* **Ram :** 512 Mb.

**5.2 SOFTWARE REQUIREMENTS**

Software requirements specifications place an important role in creating quality software solutions. Specification is basically a representation process. Requirements are represented in a manner that ultimately leads to successful software implementation. Requirements may be specified in a variety of ways. However, there are some guidelines worth following:

* Representation format and content should be relevant to the problem.
* Information contained within the specification should be nested.
* Diagrams and other notational forms should be restricted in number and consistent in use.
* Representation should be revisable.
* **Operating system :** Windows 7 Ultimate/8/10.
* **Coding Language :** Python 3.7.
* **IDE :** PyCharm Community Edition.
* **Designing :** Jupyter Notebook.
* **Data Base :** Social Network Users Dataset’s

**Libraries required**

* matplotlib==1.4.3
* numpy==1.9.2
* pandas==0.16.2
* scikit-learn==0.17
* sexmachine==0.1.1
* pybrain==0.31

**CHAPTER-6**

**IMPLEMENTATION**

**6.1 EXPLANATION OF ATTRIBUTES**

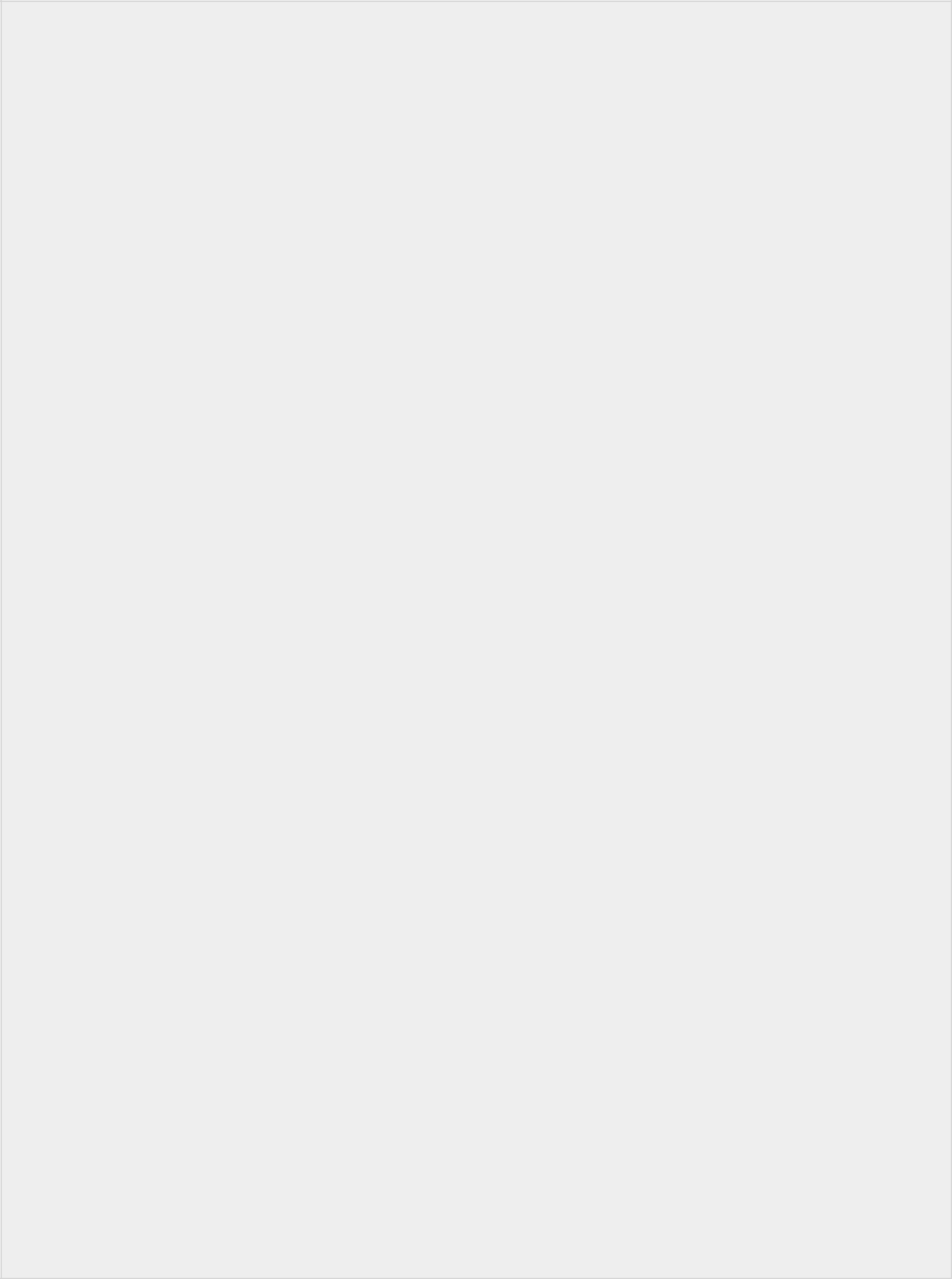
Attribute importance is a supervised function that identifies and ranks the attributes that are most important in predicting a target attribute.[4] Raw machine learning data contains a mixture of attributes, some of which are relevant to making predictions.



**Table 1: Attributes that define a Dataset**

**6.2 SAMPLE CODE**

**Neural Network.py**

****

**Neural Network.py**

# coding: utf-8

### detect the fake profiles in online social networks using Neural Network

# In[1]:

import sys

import csv

import os

import datetime

import math

import numpy as np

import pandas as pd

import matplotlib.pyplot as plt

from datetime import datetime

import sexmachine.detector as gender

from sklearn.preprocessing import Imputer

from sklearn import cross\_validation

from sklearn import metrics

from sklearn import preprocessing

from sklearn.linear\_model import LinearRegression

from sklearn.ensemble import RandomForestClassifier

from sklearn.decomposition import PCA

from sklearn.cross\_validation import StratifiedKFold, train\_test\_split

from sklearn.grid\_search import GridSearchCV

from sklearn.metrics import accuracy\_score

from sklearn.learning\_curve import learning\_curve

from sklearn.metrics import roc\_curve, auc ,roc\_auc\_score

from sklearn.metrics import confusion\_matrix

from sklearn.metrics import classification\_report

get\_ipython().magic(u'matplotlib inline')

from pybrain.structure import SigmoidLayer

from pybrain.datasets import ClassificationDataSet

from pybrain.utilities import percentError

from pybrain.tools.shortcuts import buildNetwork

from pybrain.supervised.trainers import BackpropTrainer

from pybrain.structure.modules import SoftmaxLayer

from pybrain.tools.xml.networkwriter import NetworkWriter

from pybrain.tools.xml.networkreader import NetworkReader

####### function for reading dataset from csv files

# In[2]:

def read\_datasets():

""" Reads users profile from csv files """

genuine\_users = pd.read\_csv("data/users.csv")

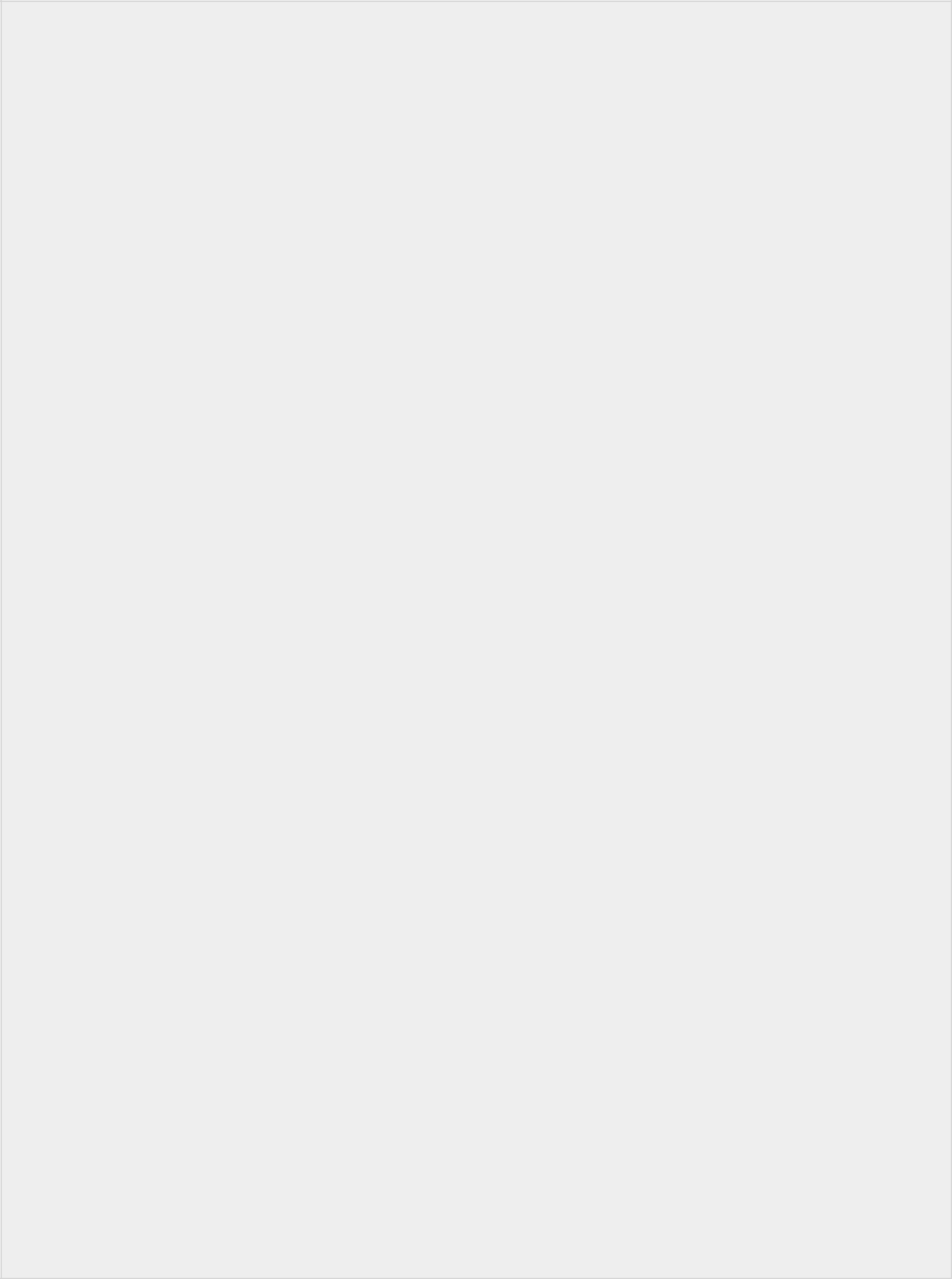
fake\_users = pd.read\_csv("data/fusers.csv")

# print genuine\_users.columns

# print genuine\_users.describe()

#print fake\_users.describe()

x=pd.concat([genuine\_users,fake\_users])

****

y=len(fake\_users)\*[0] + len(genuine\_users)\*[1]

return x,y

####### function for predicting sex using name of person

# In[3]:

def predict\_sex(name):

sex\_predictor = gender.Detector(unknown\_value=u"unknown",case\_sensitive=False)

first\_name= name.str.split(' ').str.get(0)

sex= first\_name.apply(sex\_predictor.get\_gender)

sex\_dict={'female': -2, 'mostly\_female': -1,'unknown':0,'mostly\_male':1, 'male': 2}

sex\_code = sex.map(sex\_dict).astype(int)

return sex\_code

####### function for feature engineering

# In[4]:

def extract\_features(x):

lang\_list = list(enumerate(np.unique(x['lang'])))

lang\_dict = { name : i for i, name in lang\_list }

x.loc[:,'lang\_code'] = x['lang'].map( lambda x: lang\_dict[x]).astype(int)

x.loc[:,'sex\_code']=predict\_sex(x['name'])

feature\_columns\_to\_use = ['statuses\_count','followers\_count','friends\_count','favourites\_count','listed\_count','sex\_code','lang\_code']

x=x.loc[:,feature\_columns\_to\_use]

return x

####### function for plotting confusion matrix

# In[5]:

def plot\_confusion\_matrix(cm, title='Confusion matrix', cmap=plt.cm.Blues):

target\_names=['Fake','Genuine']

plt.imshow(cm, interpolation='nearest', cmap=cmap)

plt.title(title)

plt.colorbar()

tick\_marks = np.arange(len(target\_names))

plt.xticks(tick\_marks, target\_names, rotation=45)

plt.yticks(tick\_marks, target\_names)

plt.tight\_layout()

plt.ylabel('True label')

plt.xlabel('Predicted label')

####### function for plotting ROC curve

# In[6]:

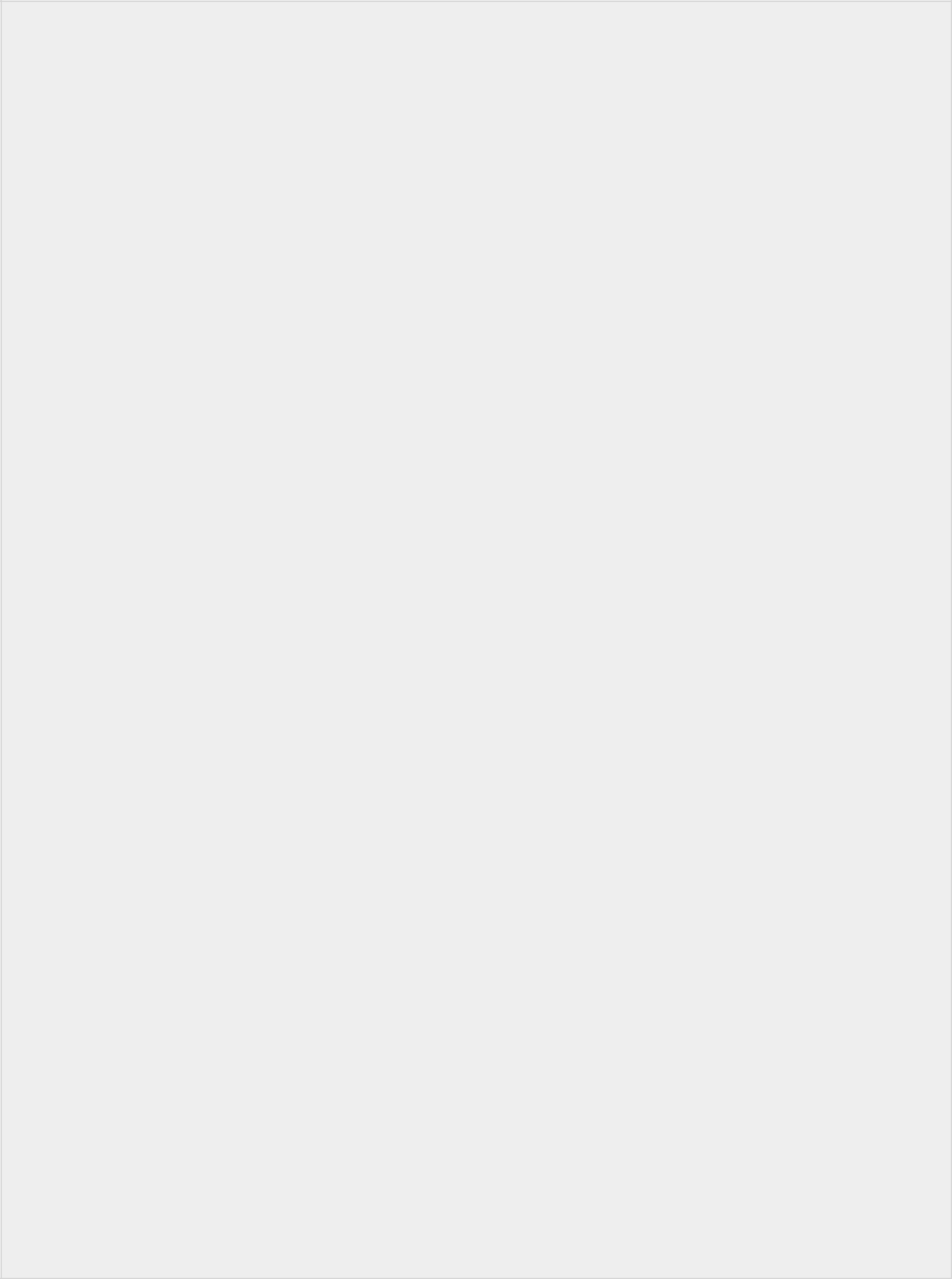
def plot\_roc\_curve(y\_test, y\_pred):

false\_positive\_rate, true\_positive\_rate, thresholds = roc\_curve(y\_test, y\_pred)

print "False Positive rate: ",false\_positive\_rate

print "True Positive rate: ",true\_positive\_rate

roc\_auc = auc(false\_positive\_rate, true\_positive\_rate)

****

plt.title('Receiver Operating Characteristic')

plt.plot(false\_positive\_rate, true\_positive\_rate, 'b',

label='AUC = %0.2f'% roc\_auc)

plt.legend(loc='lower right')

plt.plot([0,1],[0,1],'r--')

plt.xlim([-0.1,1.2])

plt.ylim([-0.1,1.2])

plt.ylabel('True Positive Rate')

plt.xlabel('False Positive Rate')

plt.show()

####### Function for training data using Neural Network

# In[7]:

def train(X,y):

""" Trains and predicts dataset with a Neural Network classifier """

ds = ClassificationDataSet( len(X.columns), 1,nb\_classes=2)

for k in xrange(len(X)):

ds.addSample(X.iloc[k],np.array(y[k]))

tstdata, trndata = ds.splitWithProportion( 0.20 )

trndata.\_convertToOneOfMany( )

tstdata.\_convertToOneOfMany( )

input\_size=len(X.columns)

target\_size=1

hidden\_size = 5

fnn=None

if os.path.isfile('fnn.xml'):

fnn = NetworkReader.readFrom('fnn.xml')

else:

fnn = buildNetwork( trndata.indim, hidden\_size , trndata.outdim, outclass=SoftmaxLayer )

trainer = BackpropTrainer( fnn, dataset=trndata,momentum=0.05, learningrate=0.1 , verbose=False, weightdecay=0.01)

trainer.trainUntilConvergence(verbose = False, validationProportion = 0.15, maxEpochs = 100, continueEpochs = 10 )

NetworkWriter.writeToFile(fnn, 'oliv.xml')

predictions=trainer.testOnClassData (dataset=tstdata)

return tstdata['class'],predictions

# In[8]:

print "reading datasets.....\n"

x,y=read\_datasets()

x.describe()

# In[9]:

print "extracting featues.....\n"

x=extract\_features(x)

print x.columns

print x.describe()

# In[10]:

print "training datasets.......\n"

y\_test,y\_pred =train(x,y)

# In[11]:

print 'Classification Accuracy on Test dataset: ' ,accuracy\_score(y\_test, y\_pred)

# In[12]:

print 'Percent Error on Test dataset: ' ,percentError(y\_pred,y\_test)

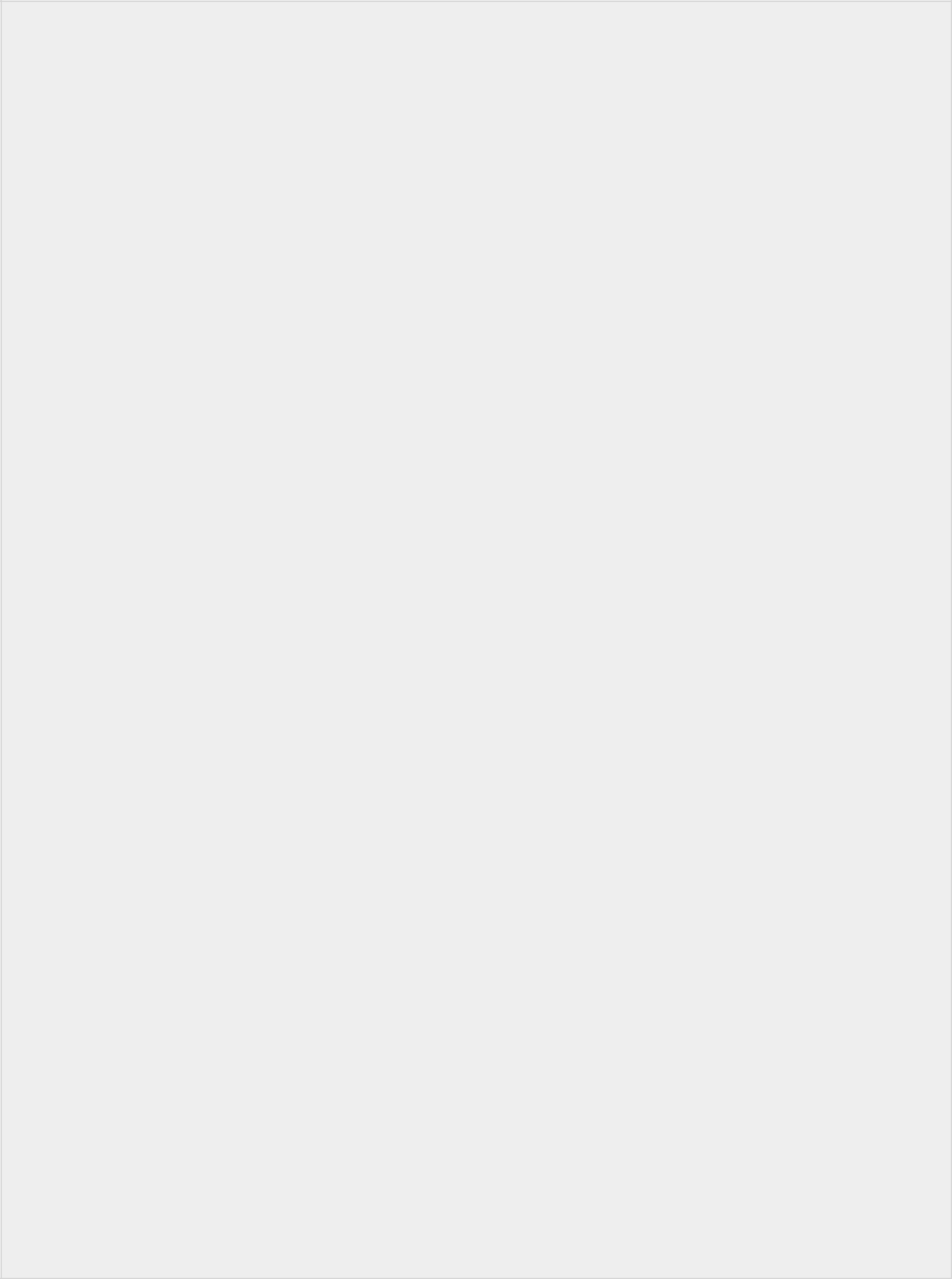
# In[13]:

cm=confusion\_matrix(y\_test, y\_pred)

print('Confusion matrix, without normalization')

print(cm)

plot\_confusion\_matrix(cm)

****

# In[11]:

print 'Classification Accuracy on Test dataset: ' ,accuracy\_score(y\_test, y\_pred)

# In[12]:

print 'Percent Error on Test dataset: ' ,percentError(y\_pred,y\_test)

# In[13]:

cm=confusion\_matrix(y\_test, y\_pred)

print('Confusion matrix, without normalization')

print(cm)

plot\_confusion\_matrix(cm)

# In[14]:

cm\_normalized = cm.astype('float') / cm.sum(axis=1)[:, np.newaxis]

print('Normalized confusion matrix')

print(cm\_normalized)

plot\_confusion\_matrix(cm\_normalized, title='Normalized confusion matrix')

# In[15]:

print(classification\_report(y\_test, y\_pred, target\_names=['Fake','Genuine']))

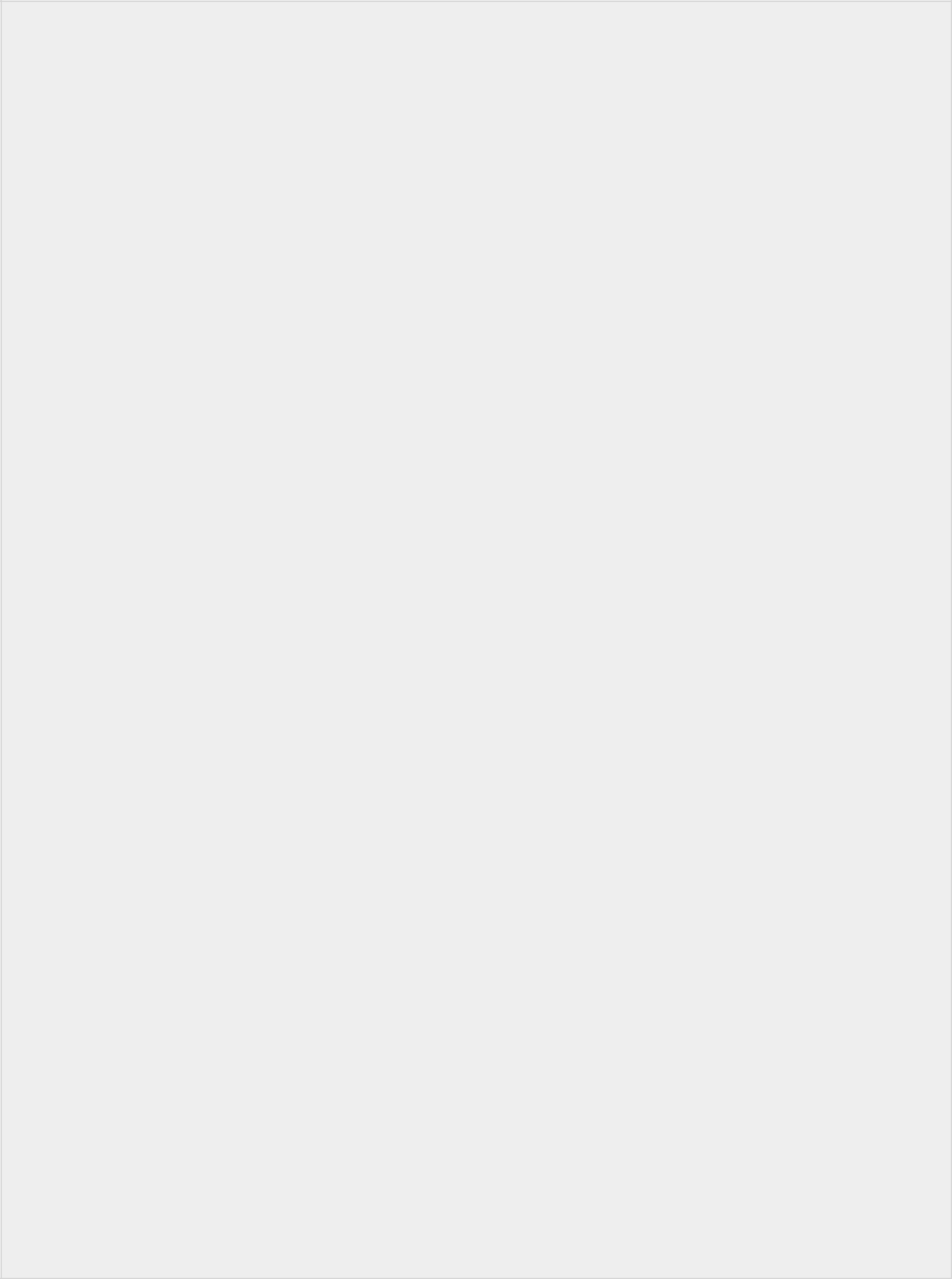
# In[16]:

s=roc\_auc\_score(y\_test, y\_pred)

print "roc\_auc\_score : ",s

# In[17]:

plot\_roc\_curve(y\_test, y\_pred)

**RandomForest.py**

**Random Forest.py**

# coding: utf-8

### Detect fake profiles in online social networks using Random Forest

# In[54]:

import sys

import csv

import datetime

import numpy as np

import pandas as pd

import matplotlib.pyplot as plt

from datetime import datetime

import sexmachine.detector as gender

from sklearn.preprocessing import Imputer

from sklearn import cross\_validation

from sklearn import metrics

from sklearn import preprocessing

from sklearn.metrics import roc\_curve, auc

from sklearn.ensemble import RandomForestClassifier

from sklearn.cross\_validation import StratifiedKFold, train\_test\_split

from sklearn.grid\_search import GridSearchCV

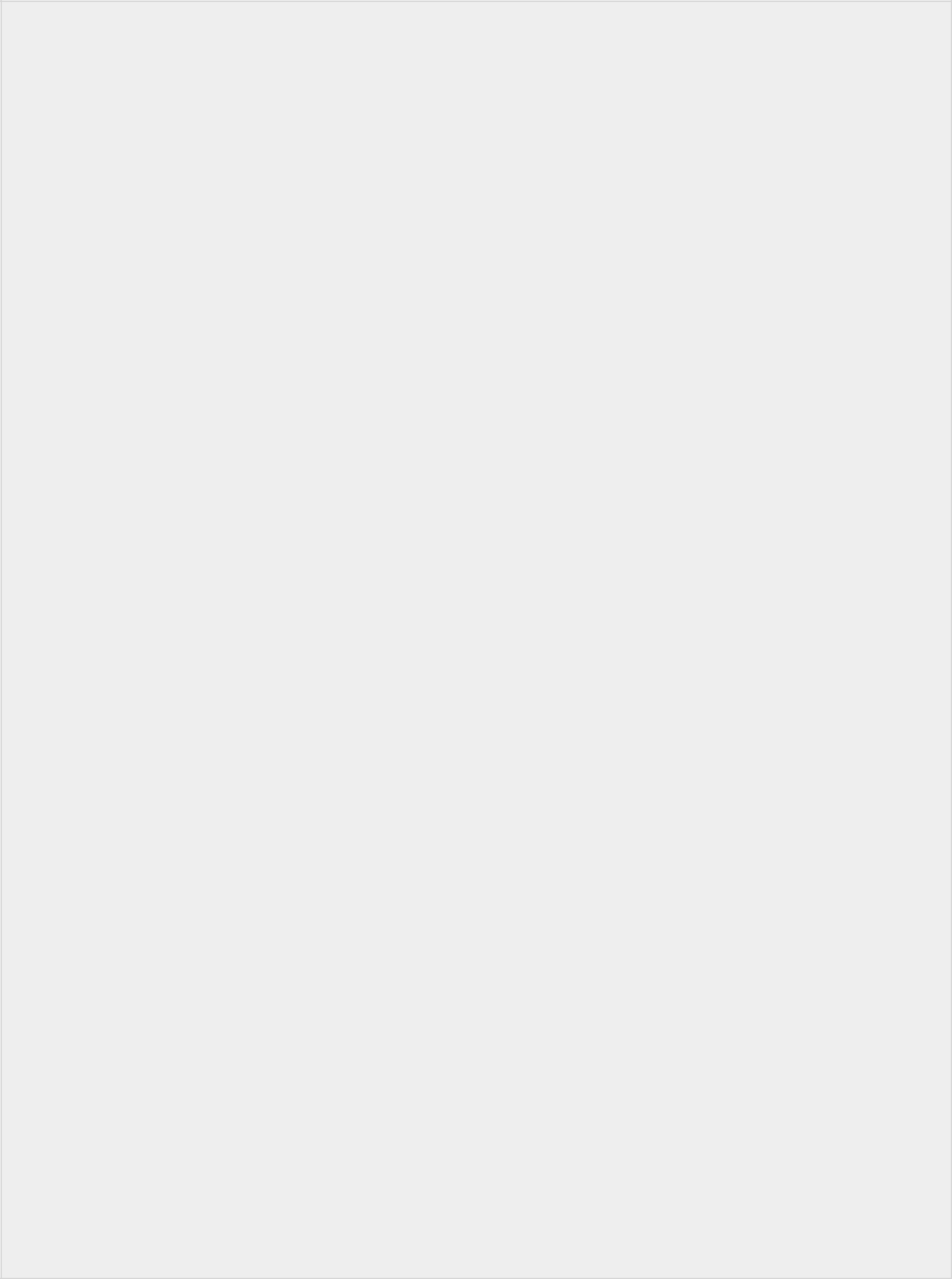
from sklearn.metrics import accuracy\_score

from sklearn.learning\_curve import learning\_curve

from sklearn.metrics import classification\_report

from sklearn.metrics import confusion\_matrix

get\_ipython().magic(u'matplotlib inline')

****

from sklearn.metrics import roc\_curve, auc

from sklearn.ensemble import RandomForestClassifier

from sklearn.cross\_validation import StratifiedKFold, train\_test\_split

from sklearn.grid\_search import GridSearchCV

from sklearn.metrics import accuracy\_score

from sklearn.learning\_curve import learning\_curve

from sklearn.metrics import classification\_report

from sklearn.metrics import confusion\_matrix

get\_ipython().magic(u'matplotlib inline')

####### function for reading dataset from csv files

# In[55]:

def read\_datasets():

""" Reads users profile from csv files """

genuine\_users = pd.read\_csv("data/users.csv")

fake\_users = pd.read\_csv("data/fusers.csv")

# print genuine\_users.columns

# print genuine\_users.describe()

#print fake\_users.describe()

x=pd.concat([genuine\_users,fake\_users])

y=len(fake\_users)\*[0] + len(genuine\_users)\*[1]

return x,y

####### function for predicting sex using name of person

# In[56]:

def predict\_sex(name):

sex\_predictor = gender.Detector(unknown\_value=u"unknown",case\_sensitive=False)

first\_name= name.str.split(' ').str.get(0)

sex= first\_name.apply(sex\_predictor.get\_gender)

sex\_dict={'female': -2, 'mostly\_female': -1,'unknown':0,'mostly\_male':1, 'male': 2}

sex\_code = sex.map(sex\_dict).astype(int)

return sex\_code

####### function for feature engineering

# In[57]:

def extract\_features(x):

lang\_list = list(enumerate(np.unique(x['lang'])))

lang\_dict = { name : i for i, name in lang\_list }

x.loc[:,'lang\_code'] = x['lang'].map( lambda x: lang\_dict[x]).astype(int)

x.loc[:,'sex\_code']=predict\_sex(x['name'])

feature\_columns\_to\_use = ['statuses\_count','followers\_count','friends\_count','favourites\_count','listed\_count','sex\_code','lang\_code']

x=x.loc[:,feature\_columns\_to\_use]

return x

####### function for ploting learning curve

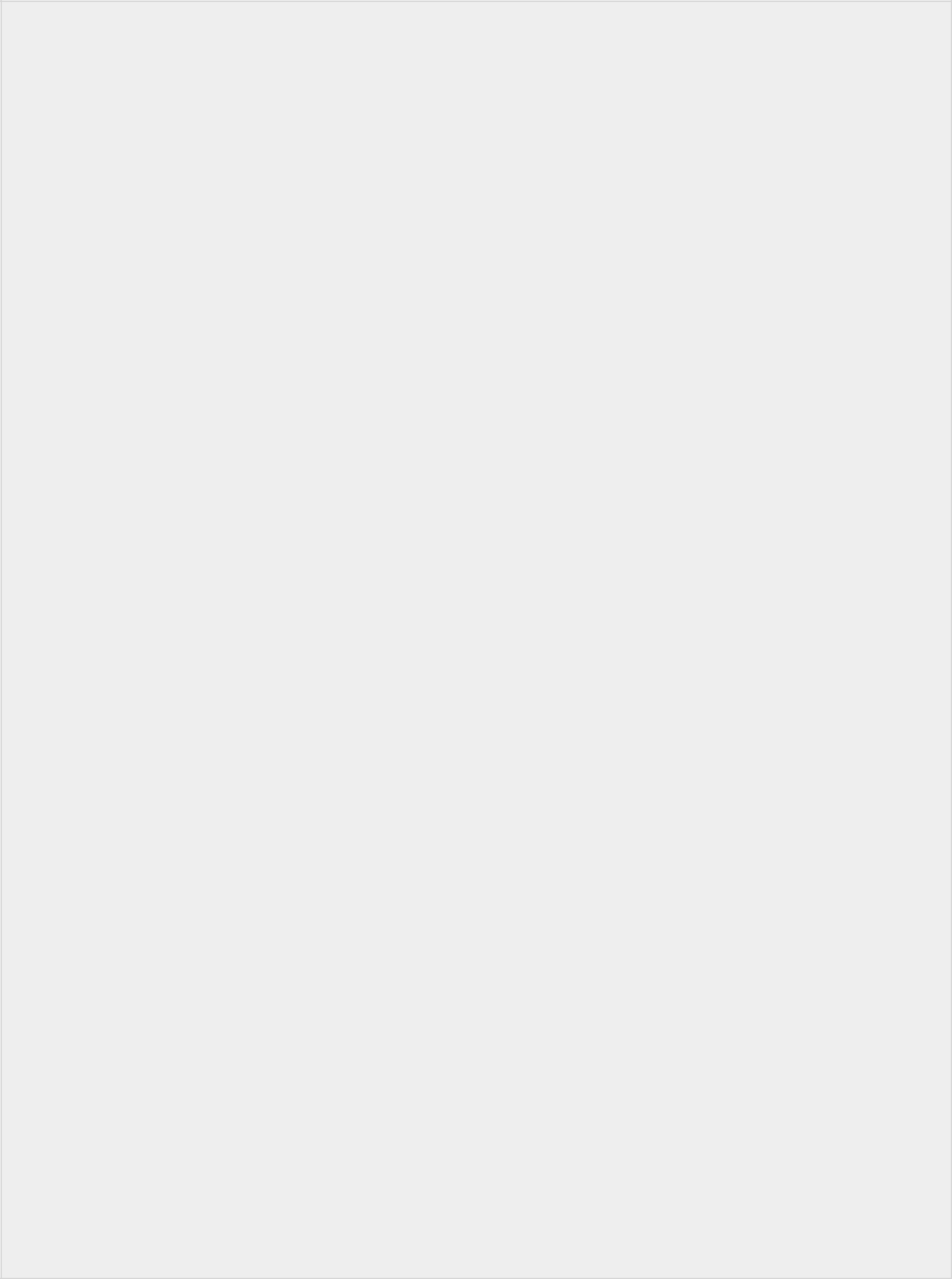
# In[60]:

def plot\_learning\_curve(estimator, title, X, y, ylim=None, cv=None,

n\_jobs=1, train\_sizes=np.linspace(.1, 1.0, 5)):

plt.figure()

plt.title(title)

****

if ylim is not None:

plt.ylim(\*ylim)

plt.xlabel("Training examples")

plt.ylabel("Score")

train\_sizes, train\_scores, test\_scores = learning\_curve(

estimator, X, y, cv=cv, n\_jobs=n\_jobs, train\_sizes=train\_sizes)

train\_scores\_mean = np.mean(train\_scores, axis=1)

train\_scores\_std = np.std(train\_scores, axis=1)

test\_scores\_mean = np.mean(test\_scores, axis=1)

test\_scores\_std = np.std(test\_scores, axis=1)

plt.grid()

plt.fill\_between(train\_sizes, train\_scores\_mean - train\_scores\_std,

train\_scores\_mean + train\_scores\_std, alpha=0.1,

color="r")

plt.fill\_between(train\_sizes, test\_scores\_mean - test\_scores\_std,

test\_scores\_mean + test\_scores\_std, alpha=0.1, color="g")

plt.plot(train\_sizes, train\_scores\_mean, 'o-', color="r",

label="Training score")

plt.plot(train\_sizes, test\_scores\_mean, 'o-', color="g",

label="Cross-validation score")

plt.legend(loc="best")

return plt

####### function for plotting confusion matrix

# In[61]:

def plot\_confusion\_matrix(cm, title='Confusion matrix', cmap=plt.cm.Blues):

target\_names=['Fake','Genuine']

plt.imshow(cm, interpolation='nearest', cmap=cmap)

plt.title(title)

plt.colorbar()

tick\_marks = np.arange(len(target\_names))

plt.xticks(tick\_marks, target\_names, rotation=45)

plt.yticks(tick\_marks, target\_names)

plt.tight\_layout()

plt.ylabel('True label')

plt.xlabel('Predicted label')

####### function for plotting ROC curve

# In[62]:

def plot\_roc\_curve(y\_test, y\_pred):

false\_positive\_rate, true\_positive\_rate, thresholds = roc\_curve(y\_test, y\_pred)

print "False Positive rate: ",false\_positive\_rate

print "True Positive rate: ",true\_positive\_rate

roc\_auc = auc(false\_positive\_rate, true\_positive\_rate)

plt.title('Receiver Operating Characteristic')

plt.plot(false\_positive\_rate, true\_positive\_rate, 'b',

label='AUC = %0.2f'% roc\_auc)

plt.legend(loc='lower right')

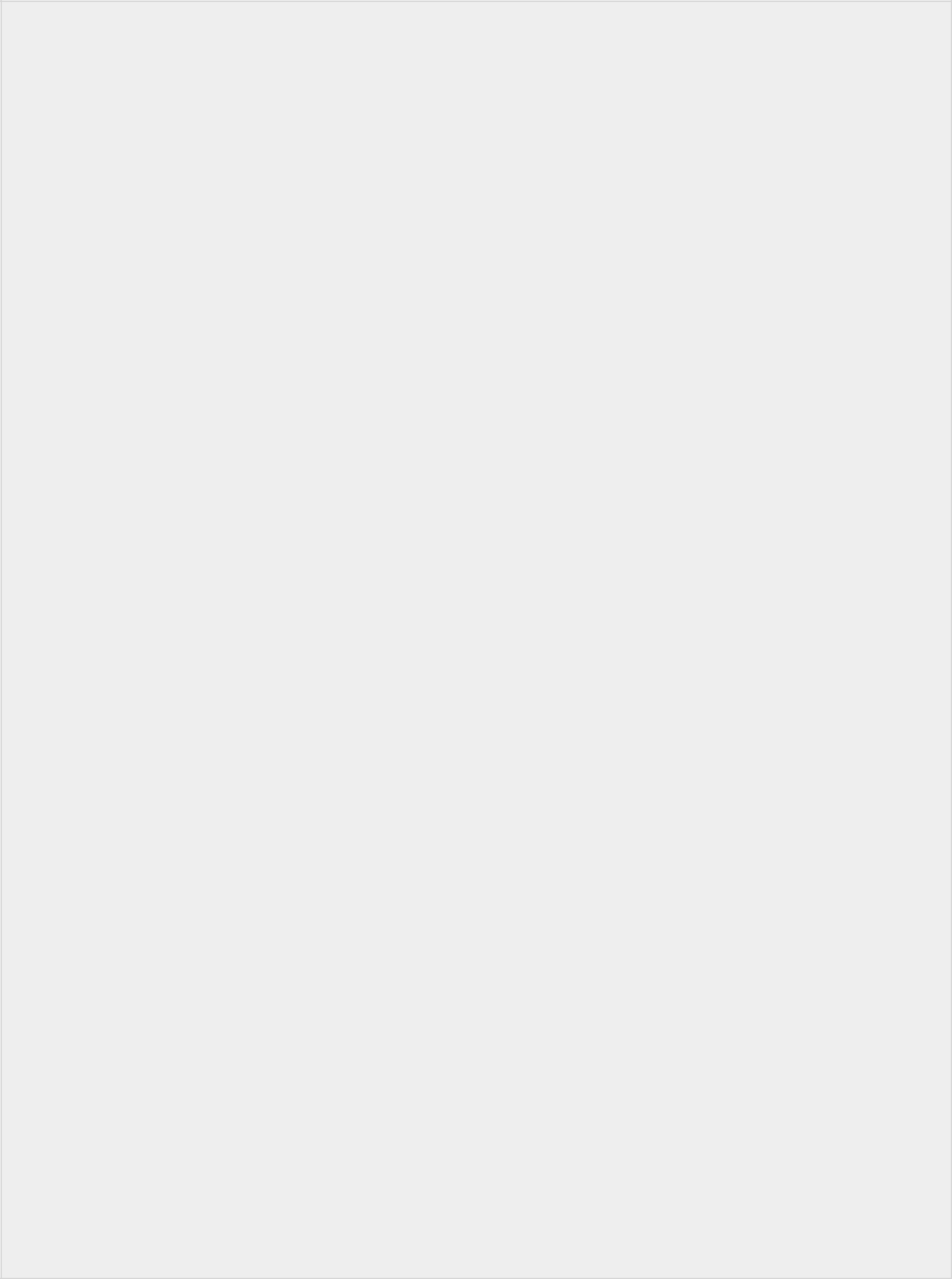
plt.plot([0,1],[0,1],'r--')

plt.xlim([-0.1,1.2])

plt.ylim([-0.1,1.2])

plt.ylabel('True Positive Rate')

plt.xlabel('False Positive Rate')

****

plt.show()

####### Function for training data using Random Forest

# In[63]:

def train(X\_train,y\_train,X\_test):

""" Trains and predicts dataset with a Random Forest classifier """

clf=RandomForestClassifier(n\_estimators=40,oob\_score=True)

clf.fit(X\_train,y\_train)

print("The best classifier is: ",clf)

# Estimate score

scores = cross\_validation.cross\_val\_score(clf, X\_train,y\_train, cv=5)

print scores

print('Estimated score: %0.5f (+/- %0.5f)' % (scores.mean(), scores.std() / 2))

title = 'Learning Curves (Random Forest)'

plot\_learning\_curve(clf, title, X\_train, y\_train, cv=5)

plt.show()

# Predict

y\_pred = clf.predict(X\_test)

return y\_test,y\_pred

# In[64]:

print "reading datasets.....\n"

x,y=read\_datasets()

x.describe()

# In[65]:

print "extracting featues.....\n"

x=extract\_features(x)

print x.columns

print x.describe()

# In[66]:

print "spliting datasets in train and test dataset...\n"

X\_train,X\_test,y\_train,y\_test = train\_test\_split(x, y, test\_size=0.20, random\_state=44)

# In[67]:

print "training datasets.......\n"

y\_test,y\_pred = train(X\_train,y\_train,X\_test)

# In[68]:

print 'Classification Accuracy on Test dataset: ' ,accuracy\_score(y\_test, y\_pred)

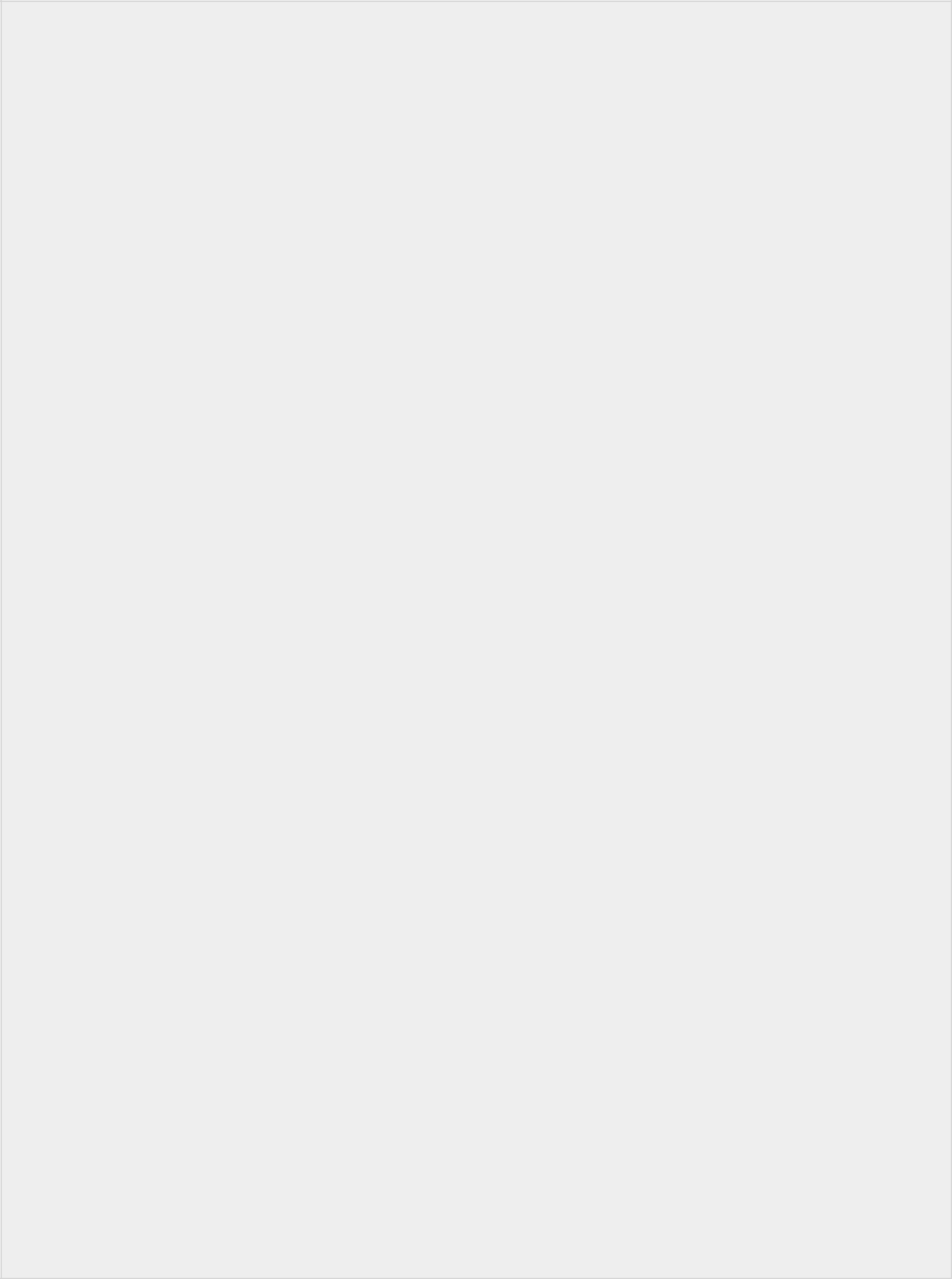
# In[70]:

cm=confusion\_matrix(y\_test, y\_pred)

print('Confusion matrix, without normalization')

print(cm)

plot\_confusion\_matrix(cm)

****

# In[71]:

cm\_normalized = cm.astype('float') / cm.sum(axis=1)[:, np.newaxis]

print('Normalized confusion matrix')

print(cm\_normalized)

plot\_confusion\_matrix(cm\_normalized, title='Normalized confusion matrix')

# In[72]:

print(classification\_report(y\_test, y\_pred, target\_names=['Fake','Genuine']))

# In[73]:

plot\_roc\_curve(y\_test, y\_pred)

**6.3 SOFTWARE ENVIRONMENT**

**Python**

Python is a general-purpose interpreted, interactive, object-oriented, and high-level programming language. An [interpreted language](https://en.wikipedia.org/wiki/Interpreted_language), Python has a design philosophy that emphasizes code [readability](https://en.wikipedia.org/wiki/Readability) (notably using [whitespace](https://en.wikipedia.org/wiki/Whitespace_character) indentation to delimit [code blocks](https://en.wikipedia.org/wiki/Code_block) rather than curly brackets or keywords), and a syntax that allows programmers to express concepts in fewer [lines of code](https://en.wikipedia.org/wiki/Source_lines_of_code) than might be used in languages such as [C++](https://en.wikipedia.org/wiki/C%2B%2B)or [Java](https://en.wikipedia.org/wiki/Java_(programming_language)). It provides constructs that enable clear programming on both small and large scales. Python interpreters are available for many [operating systems](https://en.wikipedia.org/wiki/Operating_system). [CPython](https://en.wikipedia.org/wiki/CPython), the [reference implementation](https://en.wikipedia.org/wiki/Reference_implementation) of Python, is [open source](https://en.wikipedia.org/wiki/Open_source) software and has a community-based development model, as do nearly all of its variant implementations. CPython is managed by the non-profit [Python Software Foundation](https://en.wikipedia.org/wiki/Python_Software_Foundation). Python features a [dynamic type](https://en.wikipedia.org/wiki/Dynamic_type) system and automatic [memory management](https://en.wikipedia.org/wiki/Memory_management). It supports multiple [programming paradigms](https://en.wikipedia.org/wiki/Programming_paradigm), including [object-oriented](https://en.wikipedia.org/wiki/Object-oriented_programming), [imperative](https://en.wikipedia.org/wiki/Imperative_programming), [functional](https://en.wikipedia.org/wiki/Functional_programming) and [procedural](https://en.wikipedia.org/wiki/Procedural_programming), and has a large and comprehensive [standard library](https://en.wikipedia.org/wiki/Standard_library).

**Django**

Django is a high-level Python Web framework that encourages rapid development and clean, pragmatic design. Built by experienced developers, it takes care of much of the hassle of Web development, so you can focus on writing your app without needing to reinvent the wheel. It’s free and open source.

Django's primary goal is to ease the creation of complex, database-driven websites. Django emphasizes [reusability](https://en.wikipedia.org/wiki/Reusability) and "pluggability" of components, rapid development, and the principle of [don't repeat yourself](https://en.wikipedia.org/wiki/Don%27t_repeat_yourself). Python is used throughout, even for settings files and data models.



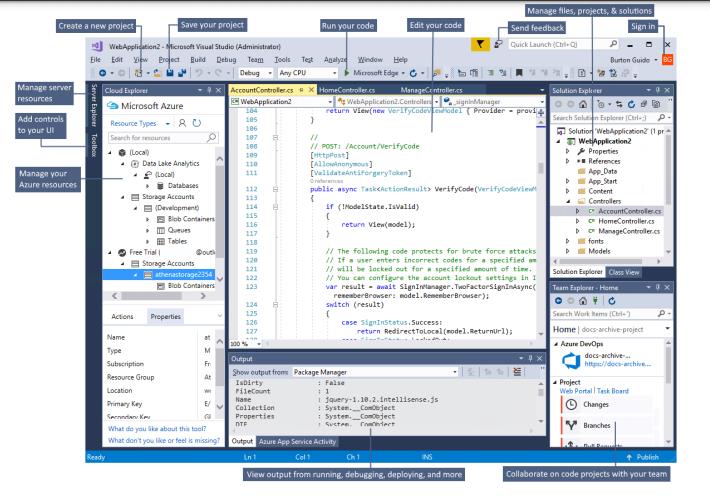
Fig: Django Framework

Django also provides an optional administrative [create, read, update and delete](https://en.wikipedia.org/wiki/Create,_read,_update_and_delete) interface that is generated dynamically through [introspection](https://en.wikipedia.org/wiki/Introspection_(computer_science)) and configured via admin models.

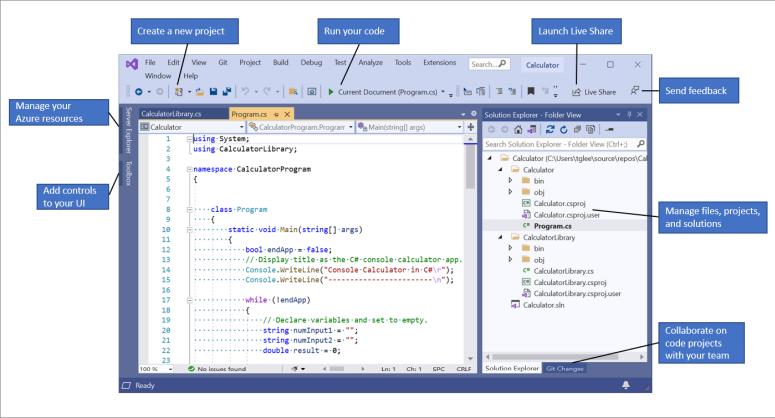


**Visual Studio IDE**

The Visual Studio integrated development environment is a creative launching pad that you can use to edit, debug, and build code, and then publish an app. An integrated development environment (IDE) is a feature-rich program that can be used for many aspects of software development. Over and above the standard editor and debugger that most IDEs provide, Visual Studio includes compilers, code completion tools, graphical designers, and many more features to ease the software development process.

****

This image shows Visual Studio with an open project and several key tool windows you'll likely use: Solution Explorer (top right) lets you view, navigate, and manage your code files. Solution Explorer can help organize your code by grouping the files into solutions and projects. The editor window (center), where you'll likely spend a majority of your time, displays file contents. This is where you can edit code or design a user interface such as a window with buttons and text boxes. The Output window (bottom center) is where Visual Studio sends notifications such as debugging and error messages, compiler warnings, publishing status messages, and more. Each message source has its own tab. Git Changes (bottom right) lets you track work items and share code with others using version control technologies such as Git and GitHub.

****

**CHAPTER-7**

**INPUT AND OUTPUT DESIGNS**

**7.1 INPUT DESIGN**

The input design is the link between the information system and the user. It comprises the developing specification and procedures for data preparation and those steps are necessary to put transaction data in to a usable form for processing can be achieved by inspecting the computer to read data from a written or printed document or it can occur by having people keying the data directly into the system. The design of input focuses on controlling the amount of input required, controlling the errors, avoiding delay, avoiding extra steps and keeping the process simple. The input is designed in such a way so that it provides security and ease of use with retaining the privacy. Input Design considered the following things:

* What data should be given as input?
* How the data should be arranged or coded?
* The dialog to guide the operating personnel in providing input.
* Methods for preparing input validations and steps to follow when error occur.

**7.2 OBJECTIVES**

1. Input Design is the process of converting a user-oriented description of the input into a computer-based system. This design is important to avoid errors in the data input process and show the correct direction to the management for getting correct information from the computerized system.

2. It is achieved by creating user-friendly screens for the data entry to handle large volume of data. The goal of designing input is to make data entry easier and to be free from errors. The data entry screen is designed in such a way that all the data manipulates can be performed. It also provides record viewing facilities.

3. When the data is entered it will check for its validity. Data can be entered with the help of screens. Appropriate messages are provided as when needed so that the user will not be in maize of instant. Thus the objective of input design is to create an input layout that is easy to follow.

**7.3 OUTPUT DESIGN**

A quality output is one, which meets the requirements of the end user and presents the information clearly. In any system results of processing are communicated to the users and to other system through outputs. In output design it is determined how the information is to be displaced for immediate need and also the hard copy output. It is the most important and direct source information to the user. Efficient and intelligent output design improves the system’s relationship to help user decision-making.

1. Designing computer output should proceed in an organized, well thought out manner; the right output must be developed while ensuring that each output element is designed so that people will find the system can use easily and effectively. When analysis design computer output, they should Identify the specific output that is needed to meet the requirements.

2. Select methods for presenting information.

3. Create document, report, or other formats that contain information produced by the system.

The output form of an information system should accomplish one or more of the following objectives.

* Convey information about past activities, current status or projections of the Future.
* Signal important events, opportunities, problems, or warnings.
* Trigger an action.
* Confirm an action.

**CHAPTER-8**

**ALGORITHMS**

# 8.1 Random Forest Algorithm

Random Forest is a popular machine learning algorithm that belongs to the supervised learning technique. It can be used for both Classification and Regression problems in ML. It is based on the concept of **ensemble learning,** which is a process of combining multiple classifiers to solve a complex problem and to improve the performance of the model.

As the name suggests, **"Random Forest is a classifier that contains a number of decision trees on various subsets of the given dataset and takes the average to improve the predictive accuracy of that dataset."** Instead of relying on one decision tree, the random forest takes the prediction from each tree and based on the majority votes of predictions, and it predicts the final output.

**The greater number of trees in the forest leads to higher accuracy and prevents the problem of overfitting.**

The below diagram explains the working of the Random Forest algorithm:



**Fig: Working of Random Forest Algorithm**

#### Note: To better understand the Random Forest Algorithm, you should have knowledge of the Decision Tree Algorithm.

## Assumptions for Random Forest

## Since the random forest combines multiple trees to predict the class of the dataset, it is possible that some decision trees may predict the correct output, while others may not. But together, all the trees predict the correct output. Therefore, below are two assumptions for a better Random forest classifier:

* There should be some actual values in the feature variable of the dataset so that the classifier can predict accurate results rather than a guessed result.
* The predictions from each tree must have very low correlations.

## Why use Random Forest?

Below are some points that explain why we should use the Random Forest algorithm:

* It takes less training time as compared to other algorithms.
* It predicts output with high accuracy, even for the large dataset it runs efficiently.
* It can also maintain accuracy when a large proportion of data is missing.

## How does Random Forest algorithm work?

## Random Forest works in two-phase first is to create the random forest by combining N decision tree, and second is to make predictions for each tree created in the first phase.

The Working process can be explained in the below steps and diagram:

**Step-1:** Select random K data points from the training set.

**Step-2:** Build the decision trees associated with the selected data points (Subsets).

**Step-3:** Choose the number N for decision trees that you want to build.

**Step-4:** Repeat Step 1 & 2.

**Step-5:** For new data points, find the predictions of each decision tree, and assign the new data points to the category that wins the majority votes.

The working of the algorithm can be better understood by the below example:

**Example:** Suppose there is a dataset that contains multiple fruit images. So, this dataset is given to the Random forest classifier. The dataset is divided into subsets and given to each decision tree. During the training phase, each decision tree produces a prediction result, and when a new data point occurs, then based on the majority of results, the Random Forest classifier predicts the final decision. Consider the below image:



## Fig : Random Forest Example

## Applications of Random Forest

There are mainly four sectors where Random forest mostly used:

1. **Banking:** Banking sector mostly uses this algorithm for the identification of loan risk.
2. **Medicine:** With the help of this algorithm, disease trends and risks of the disease can be identified.
3. **Land Use:** We can identify the areas of similar land use by this algorithm.
4. **Marketing:** Marketing trends can be identified using this algorithm.

## Advantages of Random Forest

* Random Forest is capable of performing both Classification and Regression tasks.
* It is capable of handling large datasets with high dimensionality.
* It enhances the accuracy of the model and prevents the overfitting issue.

## Disadvantages of Random Forest

* Although random forest can be used for both classification and regression tasks, it is not more suitable for Regression tasks.

# 8.2 Support Vector Machine Algorithm

Support Vector Machine or SVM is one of the most popular Supervised Learning algorithms, which is used for Classification as well as Regression problems. However, primarily, it is used for Classification problems in Machine Learning.

The goal of the SVM algorithm is to create the best line or decision boundary that can segregate n-dimensional space into classes so that we can easily put the new data point in the correct category in the future. This best decision boundary is called a hyperplane.

SVM chooses the extreme points/vectors that help in creating the hyperplane. These extreme cases are called as support vectors, and hence algorithm is termed as Support Vector Machine. Consider the below diagram in which there are two different categories that are classified using a decision boundary or hyperplane:



**Fig: SVM Hyperplanes**

**Example:** SVM can be understood with the example that we have used in the KNN classifier. Suppose we see a strange cat that also has some features of dogs, so if we want a model that can accurately identify whether it is a cat or dog, so such a model can be created by using the SVM algorithm. We will first train our model with lots of images of cats and dogs so that it can learn about different features of cats and dogs, and then we test it with this strange creature. So as support vector creates a decision boundary between these two data (cat and dog) and choose extreme cases (support vectors), it will see the extreme case of cat and dog. On the basis of the support vectors, it will classify it as a cat. Consider the below diagram:



**Fig: SVM Example**

SVM algorithm can be used for **Face detection, image classification, text categorization,** etc.

## Types of SVM

**SVM can be of two types:**

* **Linear SVM:** Linear SVM is used for linearly separable data, which means if a dataset can be classified into two classes by using a single straight line, then such data is termed as linearly separable data, and classifier is used called as Linear SVM classifier.
* **Non-linear SVM:** Non-Linear SVM is used for non-linearly separated data, which means if a dataset cannot be classified by using a straight line, then such data is termed as non-linear data and classifier used is called as Non-linear SVM classifier.

## Hyperplane and Support Vectors in the SVM algorithm

**Hyperplane:** There can be multiple lines/decision boundaries to segregate the classes in n-dimensional space, but we need to find out the best decision boundary that helps to classify the data points. This best boundary is known as the hyperplane of SVM.

The dimensions of the hyperplane depend on the features present in the dataset, which means if there are 2 features (as shown in image), then hyperplane will be a straight line. And if there are 3 features, then hyperplane will be a 2-dimension plane.

We always create a hyperplane that has a maximum margin, which means the maximum distance between the data points.

**Support Vectors:**

The data points or vectors that are the closest to the hyperplane and which affect the position of the hyperplane are termed as Support Vector. Since these vectors support the hyperplane, hence called a Support vector.

## How does SVM works?

**Linear SVM:**

The working of the SVM algorithm can be understood by using an example. Suppose we have a dataset that has two tags (green and blue), and the dataset has two features x1 and x2. We want a classifier that can classify the pair(x1, x2) of coordinates in either green or blue. Consider the below image:



So as it is 2-d space so by just using a straight line, we can easily separate these two classes. But there can be multiple lines that can separate these classes. Consider the below image:



Hence, the SVM algorithm helps to find the best line or decision boundary; this best boundary or region is called as a **hyperplane**. SVM algorithm finds the closest point of the lines from both the classes. These points are called support vectors. The distance between the vectors and the hyperplane is called as **margin**. And the goal of SVM is to maximize this margin. The **hyperplane** with maximum margin is called the **optimal hyperplane**.



**Fig: SVM working example**

**Non-Linear SVM:**

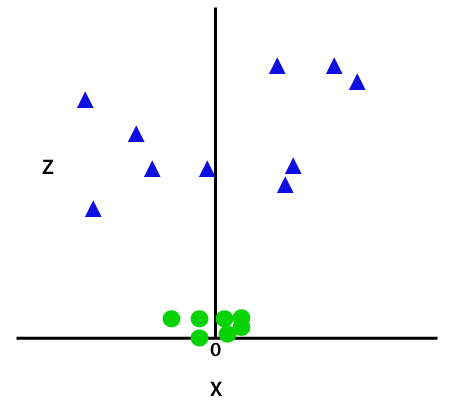
If data is linearly arranged, then we can separate it by using a straight line, but for non-linear data, we cannot draw a single straight line. Consider the below image:



So to separate these data points, we need to add one more dimension. For linear data, we have used two dimensions x and y, so for non-linear data, we will add a third dimension z. It can be calculated as:

z=x2 +y2

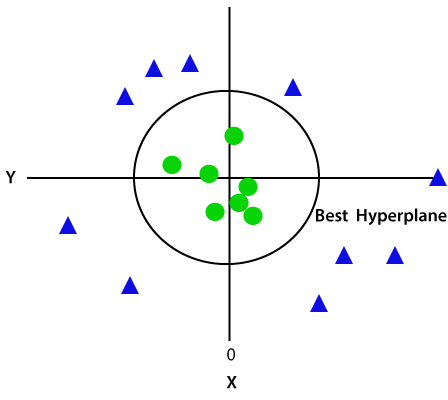
By adding the third dimension, the sample space will become as below image:



So now, SVM will divide the datasets into classes in the following way. Consider the below image:



Since we are in 3-d Space, hence it is looking like a plane parallel to the x-axis. If we convert it in 2d space with z=1, then it will become as:



Hence we get a circumference of radius 1 in case of non-linear data.

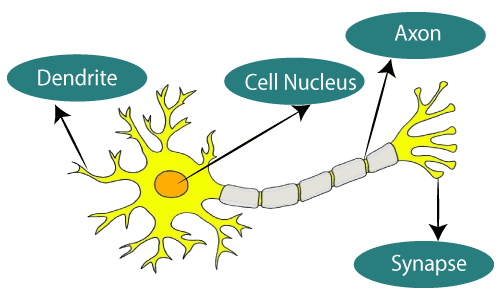
# 8.3 Artificial Neural Network

The term "Artificial neural network" refers to a biologically inspired sub-field of artificial intelligence modeled after the brain. An Artificial neural network is usually a computational network based on biological neural networks that construct the structure of the human brain. Similar to a human brain has neurons interconnected to each other, artificial neural networks also have neurons that are linked to each other in various layers of the networks. These neurons are known as nodes.

Artificial neural network tutorial covers all the aspects related to the artificial neural network. In this tutorial, we will discuss ANNs, Adaptive resonance theory, Kohonen self-organizing map, Building blocks, unsupervised learning, Genetic algorithm, etc.

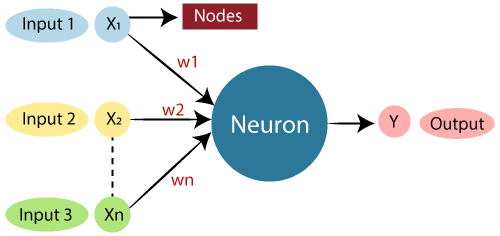
## What is Artificial Neural Network?

The term "**Artificial Neural Network**" is derived from Biological neural networks that develop the structure of a human brain. Similar to the human brain that has neurons interconnected to one another, artificial neural networks also have neurons that are interconnected to one another in various layers of the networks. These neurons are known as nodes.



**The given figure illustrates the typical diagram of Biological Neural Network.**

**The typical Artificial Neural Network looks something like the given figure.**



Dendrites from Biological Neural Network represent inputs in Artificial Neural Networks, cell nucleus represents Nodes, synapse represents Weights, and Axon represents Output.

Relationship between Biological neural network and artificial neural network:

|  |  |
| --- | --- |
| **Biological Neural Network** | **Artificial Neural Network** |
| Dendrites | Inputs |
| Cell nucleus | Nodes |
| Synapse | Weights |
| Axon | Output |

An **Artificial Neural Network** in the field of **Artificial intelligence** where it attempts to mimic the network of neurons makes up a human brain so that computers will have an option to understand things and make decisions in a human-like manner. The artificial neural network is designed by programming computers to behave simply like interconnected brain cells.

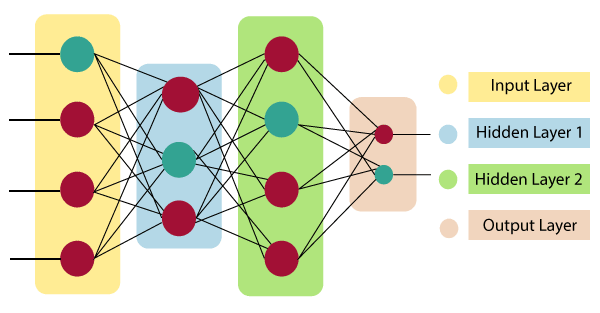
There are around 1000 billion neurons in the human brain. Each neuron has an association point somewhere in the range of 1,000 and 100,000. In the human brain, data is stored in such a manner as to be distributed, and we can extract more than one piece of this data when necessary from our memory parallelly. We can say that the human brain is made up of incredibly amazing parallel processors.

We can understand the artificial neural network with an example, consider an example of a digital logic gate that takes an input and gives an output. "OR" gate, which takes two inputs. If one or both the inputs are "On," then we get "On" in output. If both the inputs are "Off," then we get "Off" in output. Here the output depends upon input. Our brain does not perform the same task. The outputs to inputs relationship keep changing because of the neurons in our brain, which are "learning."

## The architecture of an artificial neural network:

To understand the concept of the architecture of an artificial neural network, we have to understand what a neural network consists of. In order to define a neural network that consists of a large number of artificial neurons, which are termed units arranged in a sequence of layers. Lets us look at various types of layers available in an artificial neural network.

Artificial Neural Network primarily consists of three layers:



**Fig: Neural Network Layers**

**Input Layer:**

As the name suggests, it accepts inputs in several different formats provided by the programmer.

**Hidden Layer:**

The hidden layer presents in-between input and output layers. It performs all the calculations to find hidden features and patterns.

**Output Layer:**

The input goes through a series of transformations using the hidden layer, which finally results in output that is conveyed using this layer.

The artificial neural network takes input and computes the weighted sum of the inputs and includes a bias. This computation is represented in the form of a transfer function.

What is Artificial Neural Network

It determines weighted total is passed as an input to an activation function to produce the output.

Activation functions choose whether a node should fire or not. Only those who are fired make it to the output layer. There are distinctive activation functions available that can be applied upon the sort of task we are performing.

## Advantages of Artificial Neural Network (ANN)

**Parallel processing capability:**

Artificial neural networks have a numerical value that can perform more than one task simultaneously.

**Storing data on the entire network:**

Data that is used in traditional programming is stored on the whole network, not on a database. The disappearance of a couple of pieces of data in one place doesn't prevent the network from working.

**Capability to work with incomplete knowledge:**

After ANN training, the information may produce output even with inadequate data. The loss of performance here relies upon the significance of missing data.

**Having a memory distribution:**

For ANN is to be able to adapt, it is important to determine the examples and to encourage the network according to the desired output by demonstrating these examples to the network. The succession of the network is directly proportional to the chosen instances, and if the event can't appear to the network in all its aspects, it can produce false output.

**Having fault tolerance:**

Extortion of one or more cells of ANN does not prohibit it from generating output, and this feature makes the network fault-tolerance.

## Disadvantages of Artificial Neural Network:

**Assurance of proper network structure:**

There is no particular guideline for determining the structure of artificial neural networks. The appropriate network structure is accomplished through experience, trial, and error.

**Unrecognized behavior of the network:**

It is the most significant issue of ANN. When ANN produces a testing solution, it does not provide insight concerning why and how. It decreases trust in the network.

**Hardware dependence:**

Artificial neural networks need processors with parallel processing power, as per their structure. Therefore, the realization of the equipment is dependent.

**Difficulty of showing the issue to the network:**

ANNs can work with numerichajal data. Problems must be converted into numerical values before being introduced to ANN. The presentation mechanism to be resolved here will directly impact the performance of the network. It relies on the user's abilities.

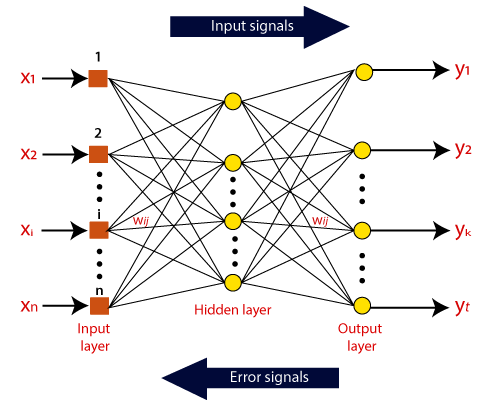
**The duration of the network is unknown:**

The network is reduced to a specific value of the error, and this value does not give us optimum results.

Science artificial neural networks that have steeped into the world in the mid-20th century are exponentially developing. In the present time, we have investigated the pros of artificial neural networks and the issues encountered in the course of their utilization. It should not be overlooked that the cons of ANN networks, which are a flourishing science branch, are eliminated individually, and their pros are increasing day by day. It means that artificial neural networks will turn into an irreplaceable part of our lives progressively important.

## How do artificial neural networks work?

Artificial Neural Network can be best represented as a weighted directed graph, where the artificial neurons form the nodes. The association between the neurons outputs and neuron inputs can be viewed as the directed edges with weights. The Artificial Neural Network receives the input signal from the external source in the form of a pattern and image in the form of a vector. These inputs are then mathematically assigned by the notations x(n) for every n number of inputs.



Afterward, each of the input is multiplied by its corresponding weights ( these weights are the details utilized by the artificial neural networks to solve a specific problem ). In general terms, these weights normally represent the strength of the interconnection between neurons inside the artificial neural network. All the weighted inputs are summarized inside the computing unit.

If the weighted sum is equal to zero, then bias is added to make the output non-zero or something else to scale up to the system's response. Bias has the same input, and weight equals to 1. Here the total of weighted inputs can be in the range of 0 to positive infinity. Here, to keep the response in the limits of the desired value, a certain maximum value is benchmarked, and the total of weighted inputs is passed through the activation function.

The activation function refers to the set of transfer functions used to achieve the desired output. There is a different kind of the activation function, but primarily either linear or non-linear sets of functions. Some of the commonly used sets of activation functions are the Binary, linear, and Tan hyperbolic sigmoidal activation functions. Let us take a look at each of them in details:

## Binary:

In binary activation function, the output is either a one or a 0. Here, to accomplish this, there is a threshold value set up. If the net weighted input of neurons is more than 1, then the final output of the activation function is returned as one or else the output is returned as 0.

## Sigmoidal Hyperbolic:

The Sigmoidal Hyperbola function is generally seen as an "**S**" shaped curve. Here the tan hyperbolic function is used to approximate output from the actual net input. The function is defined as:

**F(x) = (1/1 + exp(-????x))**

Where ???? is considered the Steepness parameter.

## Types of Artificial Neural Network:

There are various types of Artificial Neural Networks (ANN) depending upon the human brain neuron and network functions, an artificial neural network similarly performs tasks. The majority of the artificial neural networks will have some similarities with a more complex biological partner and are very effective at their expected tasks. For example, segmentation or classification.

### Feedback ANN:

In this type of ANN, the output returns into the network to accomplish the best-evolved results internally. As per the **University of Massachusetts**, Lowell Centre for Atmospheric Research. The feedback networks feed information back into itself and are well suited to solve optimization issues. The Internal system error corrections utilize feedback ANNs.

### Feed-Forward ANN:

A feed-forward network is a basic neural network comprising of an input layer, an output layer, and at least one layer of a neuron. Through assessment of its output by reviewing its input, the intensity of the network can be noticed based on group behavior of the associated neurons, and the output is decided. The primary advantage of this network is that it figures out how to evaluate and recognize input patterns.

**CHAPTER-9**

**TESTING**

The purpose of testing is to discover errors. Testing is the process of trying to discover every conceivable fault or weakness in a work product. It provides a way to check the functionality of components, sub assemblies, assemblies and/or a finished product It is the process of exercising software with the intent of ensuring that the Software system meets its requirements and user expectations and does not fail in an unacceptable manner. There are various types of test. Each test type addresses a specific testing requirement.

### 9.1 TYPES OF TESTS

### 9.1.1 Unit testing

Unit testing involves the design of test cases that validate that the internal program logic is functioning properly, and that program inputs produce valid outputs. All decision branches and internal code flow should be validated. It is the testing of individual software units of the application .it is done after the completion of an individual unit before integration. This is a structural testing, that relies on knowledge of its construction and is invasive. Unit tests perform basic tests at component level and test a specific business process, application, and/or system configuration. Unit tests ensure that each unique path of a business process performs accurately to the documented specifications and contains clearly defined inputs and expected results.

**9.1.2 Integration testing**

Integration tests are designed to test integrated software components to determine if they actually run as one program. Testing is event driven and is more concerned with the basic outcome of screens or fields. Integration tests demonstrate that although the components were individually satisfaction, as shown by successfully unit testing, the combination of components is correct and consistent. Integration testing is specifically aimed at exposing the problems that arise from the combination of components.

**9.1.3 Functional test**

Functional tests provide systematic demonstrations that functions tested are available as specified by the business and technical requirements, system documentation, and user manuals.

Functional testing is centered on the following items:

Valid Input : identified classes of valid input must be accepted.

Invalid Input : identified classes of invalid input must be rejected.

Functions : identified functions must be exercised.

Output : identified classes of application outputs must be exercised.

Systems/Procedures : interfacing systems or procedures must be invoked.

Organization and preparation of functional tests is focused on requirements, key functions, or special test cases. In addition, systematic coverage pertaining to identify Business process flows; data fields, predefined processes, and successive processes must be considered for testing. Before functional testing is complete, additional tests are identified and the effective value of current tests is determined.

**9.1.4 System Test**

System testing ensures that the entire integrated software system meets requirements. It tests a configuration to ensure known and predictable results. An example of system testing is the configuration oriented system integration test. System testing is based on process descriptions and flows, emphasizing pre-driven process links and integration points.

**9.1.5 White Box Testing**

White Box Testing is a testing in which in which the software tester has knowledge of the inner workings, structure and language of the software, or at least its purpose. It is purpose. It is used to test areas that cannot be reached from a black box level.

**9.1.6 Black Box Testing**

Black Box Testing is testing the software without any knowledge of the inner workings, structure or language of the module being tested. Black box tests, as most other kinds of tests, must be written from a definitive source document, such as specification or requirements document, such as specification or requirements document. It is a testing in which the software under test is treated, as a black box .You cannot “see” into it. The test provides inputs and responds to outputs without considering how the software works.

**Unit Testing**

Unit testing is usually conducted as part of a combined code and unit test phase of the software lifecycle, although it is not uncommon for coding and unit testing to be conducted as two distinct phases.

**9.2 TEST STRATEGY AND APPROACH**

Field testing will be performed manually and functional tests will be written in detail.

**9.2.1 Test objectives**

* All field entries must work properly.
* Pages must be activated from the identified link.
* The entry screen, messages and responses must not be delayed.

**Features to be tested**

* Verify that the entries are of the correct format
* No duplicate entries should be allowed
* All links should take the user to the correct page.

# 9.2.2 Integration Testing

Software integration testing is the incremental integration testing of two or more integrated software components on a single platform to produce failures caused by interface defects.

The task of the integration test is to check that components or software applications, e.g. components in a software system or – one step up – software applications at the company level – interact without error.

**Test Results:** All the test cases mentioned above passed successfully. No defects encountered.

**9.2.3 Acceptance Testing**

User Acceptance Testing is a critical phase of any project and requires significant participation by the end user. It also ensures that the system meets the functional requirements.

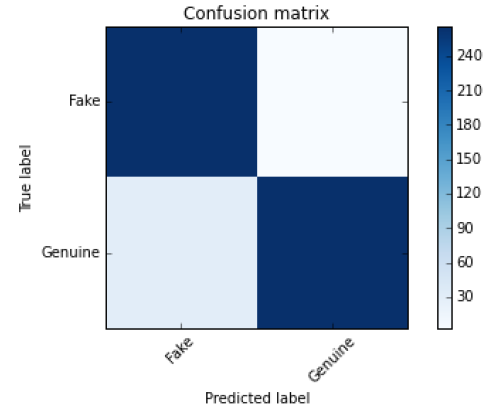
**Test Results:** All the test cases mentioned above passed successfully. No defects encountered.

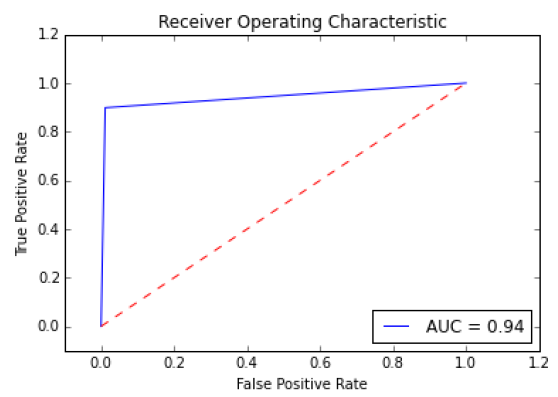
**CHAPTER-10**

**EXPERIMENTAL RESULTS AND DISCUSSION**

**10.1 PERFORMANCE OF MODEL USING RANDOM FOREST ALGORITHM**

The random forest is a model made up of many decision trees. When training the model using Random forest algorithm, each tree in a random forest learns from a random sample of the data points and the samples drawn with replacement are known as bootstrapping in which some samples will be used multiple times in a single tree.

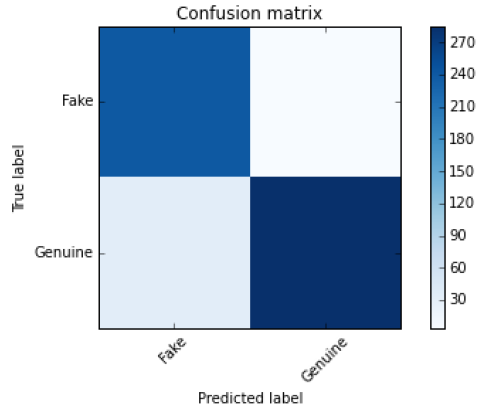
****

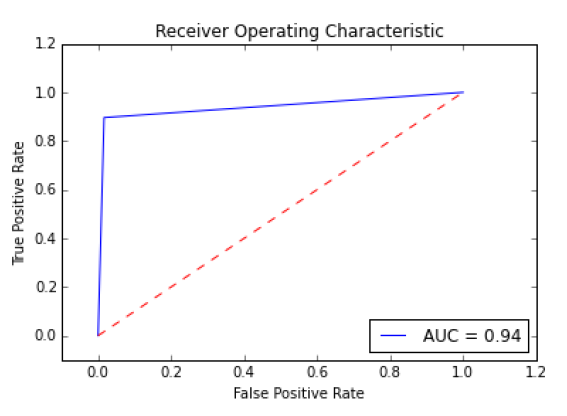
****

**10.2 PERFORMANCE OF MODEL USING SUPPORT VECTOR MACHINE ALGORITHM**

In many supervised learning tasks, labeling instances to create a training set is time consuming and costly; thus, finding ways to minimize the number of labeled instances is beneficial. The Support Vector Machine algorithm is used to minimize the instances by improving efficiency. In this algorithm, we plot each data item as a point in n-dimensional space (where n is number of features you have) with the value of each feature being the value of a particular coordinate.

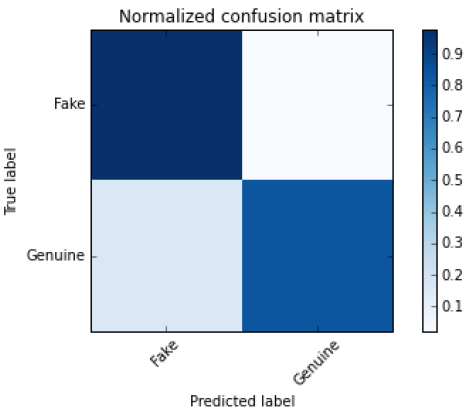
We then perform the detection of fake accounts through classification technique by finding the hyper-plane that differentiate the two classes very well (look at the below snapshot).

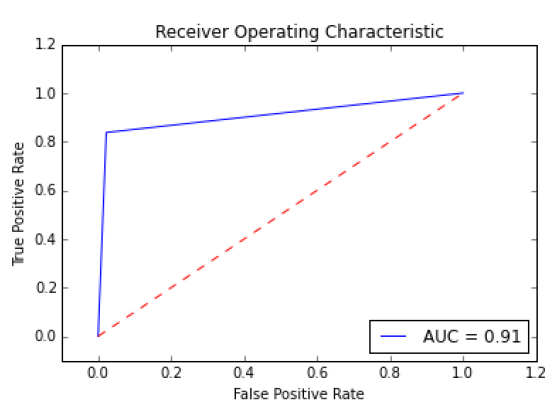
****

****

**10.3 PERFORMANCE OF MODEL USING NEURAL NETWORKS ALGORITHM**

Neural networks (NNs) can be defined as “The algorithms in machine learning are implemented by using the structure of neural networks. These neural networks model the data using artificial neurons. Neural networks thus mimic the functioning of the brain.” The ‘thinking’ or processing that a brain carries out is the result of these neural networks in action. The Neural networks algorithm tries to improve the performance of the model by using smart computational methods to create new and better performing types of prediction and detection model.

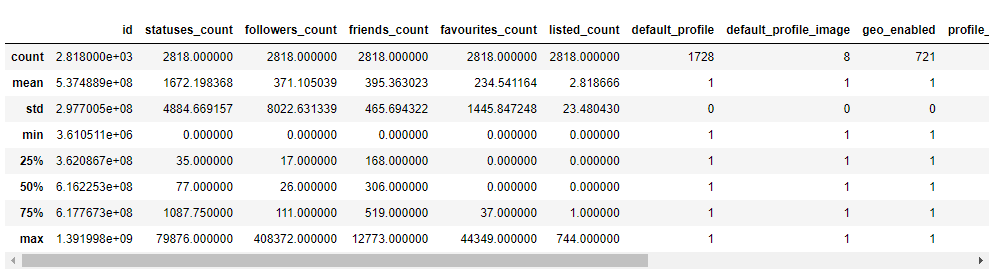




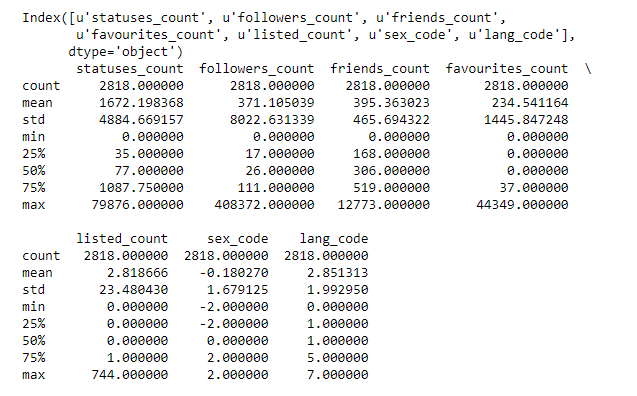
**CHAPTER-11**

**OUTPUT SCREENS**

**Dataset sample:**

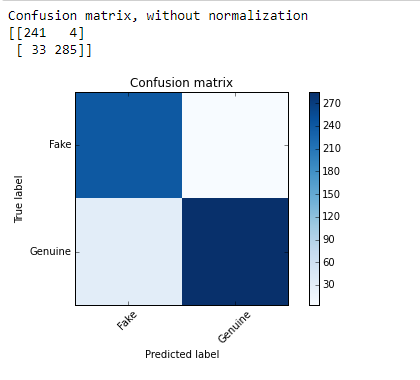
****

**Extracted columns**

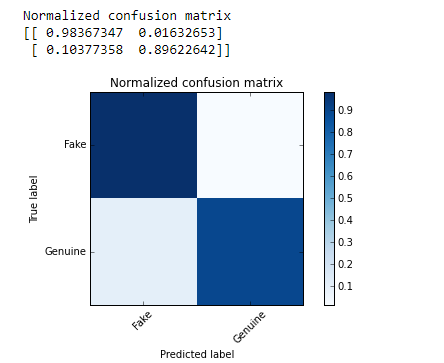


**NEURAL NETWORK**

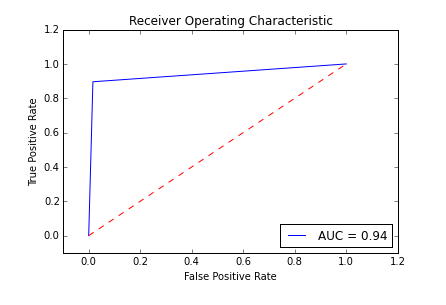
**Confusion matrix without normalization in Neural networks**

****

**Normalized Confusion matrix in Neural Networks**

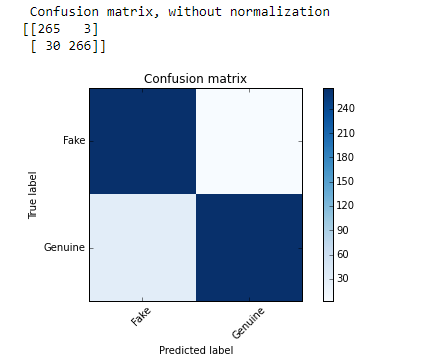
****

**ROC Curve in Neural Networks**

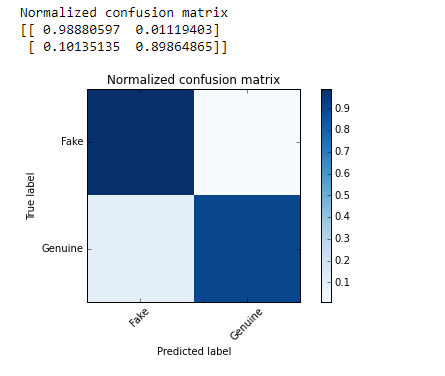
****

**RANDOM FOREST CLASSIFIER**

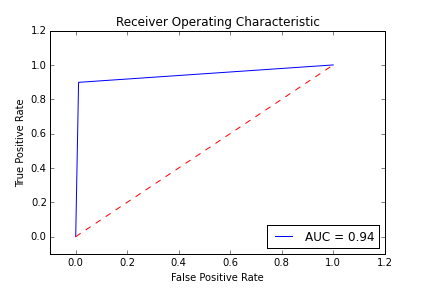
**Confusion matrix without normalization in Random Forest**

****

**Normalized confusion matrix in Random Forest**

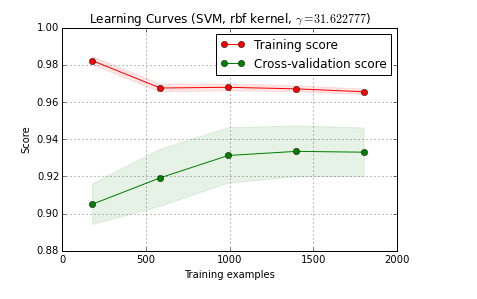
****

**Roc curve in Random Forest**

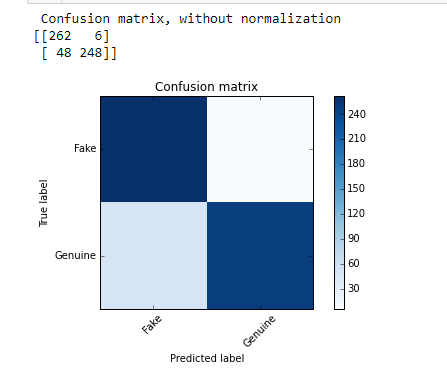
****

**SUPPORT VECTOR MACHINE**

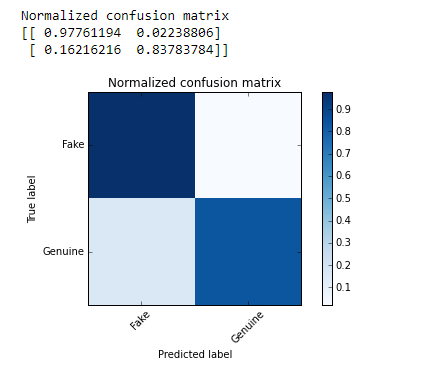
**SVM Learning Curve**

****

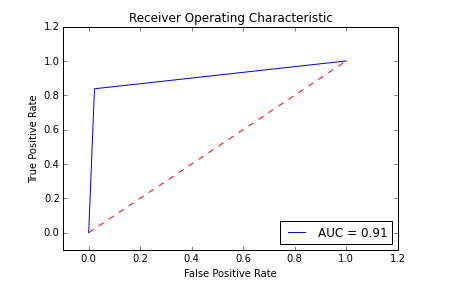
**SVM Confusion Matrix Without Normalisation**

****

**SVM Normalised Confusion Matrix**

****

**SVM ROC Curve**

****

**CHAPTER-12**

**CONCLUSION**

Through utilization of different kinds of Machine Learning Algorithms, this paper is aimed to exploit different aspects of dataset which has not been deeply considered in literature and to find a good way of detection of the fake and automated accounts. In this paper we have presented a Machine Learning pipeline for detecting fake accounts in online social networks. Rather than making a prediction using one single algorithm, our system uses three different classification algorithms to determine whether or not an account in the provided dataset is a fake account or not. Our evaluation using Support Vector Machine, Random Forest and Neural Networks showed strong performance, and the comparison of the accuracy of prediction seemed to be higher using Support Vector Machine for the given dataset. The Accuracy of detecting fake accounts is found to be higher using Random Forest Algorithm followed by Neural Networks Algorithm for a given dataset. As a future work,[5] recurrent neural networks can be utilized for the time series user data for a better detection of fake accounts and the algorithms can be applied to various social online platforms such as Instagram, LinkedIn and Twitter to detect the fake accounts.

## ****CHAPTER-13****

## ****FUTURE WORK****

Since we have limited data to train the classifier, our approach is facing a high variance problem which can be observed in the learning curve as follows High variance problems can usually be mitigated by increasing the size of the dataset which should not be of much concern to Social Networks Organizations which already have fairly large datasets.

It was also noticed that using the feature set provided by PCA, results a very low classification accuracy, while the correlation feature set results high classification accuracy. This happened because PCA do dimension reduction and generate a new features base on linear combination of original features. But the correlation, and other feature selection techniques select the best set of original features, not linear combination of all features. On other words feature selection select the most effective original features, but PCA do a linear combination of the original features event they are not effective. The correlation feature set records a remarkable accuracy among the other feature selection technique sets, because correlation technique not only select the best features, but also removes the redundancy.

**CHAPTER-14**

**BIBLIOGRAPHY**

[1] S. Khaled, N. El-Tazi and H. M. O. Mokhtar, "Detecting Fake Accounts on Social Media," 2018 IEEE International Conference on Big Data (Big Data), Seattle, WA, USA, 2018, pp. 3672- 3681.

[2] Rao, K. Sreenivasa, N. Swapna, and P. Praveen Kumar. "Educational data mining for student placement prediction using machine learning algorithms." Int. J. Eng. Technol. Sci 7.1.2 (2018): 43-46.

[3] Y. Boshmaf, D. Logothetis, G. Siganos, J. Lería, J. Lorenzo, M. Ripeanu, K. Beznosov, H. Halawa, "Íntegro: Leveraging victim prediction for robust fake account detection in large scale osns", Computers & Security, vol. 61, pp. 142-168, 2016.

[4] N. Singh, T. Sharma, A. Thakral and T. Choudhury, "Detection of Fake Profile in Online Social Networks Using Machine Learning," 2018 International Conference on Advances in Computing and Communication Engineering (ICACCE), Paris, 2018, pp. 231-234.

[5] D. M. Freeman, "Detecting clusters of fake accounts in online social networks", 8th ACM Workshop on Artificial Intelligence and Security, pp. 91-101.

[6] (2018) Quarterly earning reports. Internet draft. [Online]. Available: <https://investor.fb.com/home/default.aspx>

[7] (2018) Statista.twitter: number of monthly active users 2010-2018. Internet draft. [Online]. Available: <https://www.statista.com/statistics/282087/number-of-monthlyactive-twitter-users/>

[8] Y. Boshmaf, M. Ripeanu, K. Beznosov, and E. Santos-Neto, “Thwarting fake osn accounts by predicting their victims,” in Proceedings of the 8th ACM Workshop on Artificial Intelligence and Security. ACM, 2015, pp. 81–89.

[9] (2018) Facebook publishes enforcement numbers for the first time. Internet draft. [Online]. Available: <https://newsroom.fb.com/news/2018/05/enforcement-numbers/>

[10] (2013) Banque populaire dis-moi combien damis tu as sur facebook, je te dirai si ta banque va taccorder un prłt. Internet draft. [Online]. Available: <http://bigbrowser.blog.lemonde.fr/2013/09/19/popularitedis-moi-combien-damis-tu-as-sur-facebook-je-te-dirai-si-ta-banqueva-taccorder-un-pret/>

[11] S.-T. Sun, Y. Boshmaf, K. Hawkey, and K. Beznosov, “A billion keys, but few locks: the crisis of web single sign-on,” in Proceedings of the 2010 New Security Paradigms Workshop. ACM, 2010, pp. 61–72.

[12] S. Fong, Y. Zhuang, and J. He, “Not every friend on a social network can be trusted: Classifying imposters using decision trees,” in Future Generation Communication Technology (FGCT), 2012 International Conference on. IEEE, 2012, pp. 58–63.

[13] K. Thomas, C. Grier, D. Song, and V. Paxson, “Suspended accounts in retrospect: an analysis of twitter spam,” in Proceedings of the 2011 ACM SIGCOMM conference on Internet measurement conference. ACM, 2011, pp. 243–258.

[14] Y. Boshmaf, I. Muslukhov, K. Beznosov, and M. Ripeanu, “The socialbot network: when bots socialize for fame and money,” in Proceedings of the 27th annual computer security applications conference. ACM, 2011, pp. 93–102.

[15] J. Ratkiewicz, M. Conover, M. Meiss, B. Gonc¸alves, S. Patil, A. Flammini, and F. Menczer, “Truthy: mapping the spread of astroturf in microblog streams,” in Proceedings of the 20th international conference companion on World wide web. ACM, 2011, pp. 249–252.

[16] (2018) How concerned are you that there are fake accounts and bots on social media platforms that are used to try to sell you things or influence you? Internet draft. [Online]. Available: <https://www.statista.com/statistics/881017/fakesocial-media-accounts-bots-influencing-selling-purchases-usa/>

[17] (2012) Buying their way to twitter fame. Internet draft. [Online]. Available: <https://investor.fb.com/home/default.aspxhttp://www.nytimes.com/2012/08/23/fashion/twitterfollowers-for-sale.html?smid=pl-share>

[18] (2017) Welcome to the era of the bot as political boogeyman. Internet draft. [Online]. Available: https://www.washingtonpost.com/news/politics/wp/2017/06/12/welcometo-the-era-of-the-bot-as-political-boogeyman/?utmterm = .2271ba8db710

[19] (2018) Human or ’bot’? doubts over italian comic beppe grillo’s twitter followers. Internet draft. [Online]. Available: <https://www.telegraph.co.uk/technology/twitter/9421072/Human-orbot-Doubts-over-Italian-comic-Beppe-Grillos-Twitter-followers.html>

[20] Y. Boshmaf, D. Logothetis, G. Siganos, J. Ler´ıa, J. Lorenzo, M. Ripeanu, K. Beznosov, and H. Halawa, “´Integro: Leveraging victim prediction for robust fake account detection in large scale osns,” Computers & Security, vol. 61, pp. 142–168, 2016.

[21] Q. Cao, M. Sirivianos, X. Yang, and T. Pregueiro, “Aiding the detection of fake accounts in large scale social online services,” in Proceedings of the 9th USENIX conference on Networked Systems Design and Implementation. USENIX Association, 2012, pp. 15–15.

[22] L. Alvisi, A. Clement, A. Epasto, S. Lattanzi, and A. Panconesi, “Sok: The evolution of sybil defense via social networks,” in Security and Privacy (SP), 2013 IEEE Symposium on. IEEE, 2013, pp. 382–396.

[23] P. Patel, K. Kannoorpatti, B. Shanmugam, S. Azam, and K. C. Yeo, “A theoretical review of social media usage by cyber-criminals,” in Computer Communication and Informatics (ICCCI), 2017 International Conference on. IEEE, 2017, pp. 1–6.

Received December 5, 2017, accepted January 12, 2018, date of publication January 23, 2018, date of current version March 9, 2018.

*Digital Object Identifier 10.1109/ACCESS.2018.2796018*

Using Machine Learning to Detect Fake Identities:

Bots vs Humans

ESTÉE VAN DER WALTAND JAN ELOFF

Department of Computer Science, University of Pretoria, Pretoria 0002, South Africa

Corresponding author: Estée Van Der Walt (estee.vanderwalt@gmail.com)

**ABSTRACT** There are a growing number of people who hold accounts on social media platforms (SMPs) but hide their identity for malicious purposes. Unfortunately, very little research has been done to date to detect fake identities created by humans, especially so on SMPs. In contrast, many examples exist of cases where fake accounts created by bots or computers have been detected successfully using machine learning models. In the case of bots these machine learning models were dependent on employing engineered features, such as the ‘‘friend-to-followers ratio.’’ These features were engineered from attributes, such as ‘‘friend-count’’ and ‘‘follower-count,’’ which are directly available in the account profiles on SMPs. The research discussed in this paper applies these same engineered features to a set of fake human accounts in the hope of advancing the successful detection of fake identities created by humans on SMPs.

**INDEX TERMS** Big data, bots, data science, fake accounts, fake identities, identity deception, social media,

veracity.

# INTRODUCTION

Identity deception on big data platforms (like social media) is an increasing problem, due to the continued growth and exponential evolvement of these platforms. Social media is one of the preferred means of communication [1] and has become a target for spammers and scammers alike [2]. Cyberthreats like spamming, which involves the sending of unsolicited emails, are common in email applications. These same threats - and more - now emerge on social media platforms (SMPs), although in different manifestations.

Much can be learned about people’s behaviour and needs through analysing their interactions with one another. Habits and topics of conversations can be evaluated to deliver a better service or product to customers and ultimately to people at large [1], [3]. The same information can however also be used against people, very often in a deceptive way. For example, a cluster of people may influence an opinion [4] when the other participants in the conversation are unaware that the ‘‘people’’ in the cluster are not real.



Since the detection of fake social engagement is quite challenging [5], this vulnerability is greatly abused [6]. We believe that these fake accounts can be attributed to, among others, the following factors:

* *The privacy policies of SMPs not expecting persons to reveal their true identity* [7]. The authenticity of people is constantly being questioned [1], and this can detrimentally affect [2] those who are falsely accused

or misled. An example is the case of cyberbullying [8] where children are bullied online through the spreading of false rumours.

* *Malicious individuals and groups on SMPs striving to spread chaos and pandemonium.* A recent example was the spreading of fake news about Hurricane Sandy in the US [9]. False news about the hurricane went viral and became a main source of information for those affected by the storm.
* *The gamification of sites, with more ‘‘likes’’ or ‘‘followers’’ inadvertently meaning greater popularity and higher social ratings* [2]. This trend drives people to find new means to artificially or manually [2] stay ahead of their competitors. By analogy, the most popular candidate in a political election usually receives most of the votes [10].
* *The ease with which false accounts and actions can be obtained.* An example is false accounts being bought online at a marketplace [11] at minimal cost, or delivered through crowdsourcing services [12]. It is even possible to buy Twitter followers and Facebook ‘‘likes’’ online [5].

|  |
| --- |
| 2169-3536 2018 IEEE. Translations and content mining are permitted for academic research only.  6540 Personal use is also permitted, but republication/redistribution requires IEEE permission. VOLUME 6, 2018  See http://www.ieee.org/publications\_standards/publications/rights/index.html for more information. |

Fake accounts can be either human-generated, computergenerated (also referred to as ‘‘bots’’), or cyborgs [13]. A cyborg is a half-human, half-bot account [13]. Such an account is manually opened by a human, but from then onwards the actions are automated by a bot. Variations exist

between bots and human accounts. For example, bots are known as ‘‘Sybil’’ accounts when the accounts are fake [1], [12] and not stolen from legitimate users [14]. On the other hand, fake human accounts are known as ‘‘trolls’’ when their purpose is to defame the character of another person [8]. Regardless of the origin of the account, the malicious intent of these fake identities is as follows:

* To change the actions of an individual or group examples are online extremism; terrorist propaganda; and radicalisation campaigns [15].
* To change perceptions of an individual or group examples are changing the creditworthiness of an account [16]; spreading rumours and false news [9], [17]; defaming someone’s character [8], [17]; polarising opinions [4], [8]; influencing popularity [11], [18]; and skewing perceptions [10].
* To hide the malicious activity of an individual or group - examples are identity impersonation [8]; identity theft [11]; cyberbullying [8]; dissemination of pornography [17]; and fraud [11].
* To spread malware - examples are the creation of false communications to steal credentials [19]; or misdirecting users to fake web sites [20].

Past research has done much to detect fake identities generated by bots. Machine learning [16], [21] has been used to not only detect bots on SMPs but also identify the intent of the bot [21]. Fake identities can possibly be detected by various approaches. These can include, amongst others, the detection of fake content linked to the account [10], investigating the account profile itself [1], or using non-verbal indicators, for example the time between opening an account and the first entry posted [22].

The problem is that very little has been done so far to detect actual human identities that are fake. The field of psychology has provided suggestions as to what constitutes a fake identity [23]–[25]. Humans are just as responsible for the malicious intents found on SMPs and they therefore warrant the same attention. The difference, according to the authors, is that fake bot accounts target groups at large, whereas fake humanaccountsrathertendtotargetspecificindividuals.This could lead to severe consequences for the targeted individual.

In a previous study [26] we investigated the use of social media attributes found in Twitter, with the aim of detecting instances of identity deception by humans on SMPs. We found that standard attributes alone, such as the number of friend and followers that are available through application programming interfaces (APIs) and describing accounts in SMPs [27]–[29] like Twitter, were not sufficient to successfully detect fake identities created by humans.

In this paper we evaluate whether readily available and engineered features that are used for the successful detection, using machine learning models, of fake identities created by bots or computers can be used to detect fake identities created by humans. This is done in the hope that similar features can serve as a catalyst for uncovering identity deception by humans on SMPs.

# RELATED WORK

Seeing that very little has been done so far to detect actual fake human identities on SMPs, we looked towards past research addressing similar problems. Spam behaviour found in emails and SMS, for example, shows similar malicious intent with fake accounts spreading false rumours [30]. Spamming occurs when electronic media such as emails, SMSs and SMPs are used to send unsolicited content to an individual or group [31]. Besides spam, fake identities are also present on SMPs in the form of bots.

Previous research towards understanding and identifying spam behaviour presented techniques like filtering [32], rules [30], and machine learning [16] to detect fake identities. The same techniques, and more, have been applied to SMPs to detect fake bot accounts:

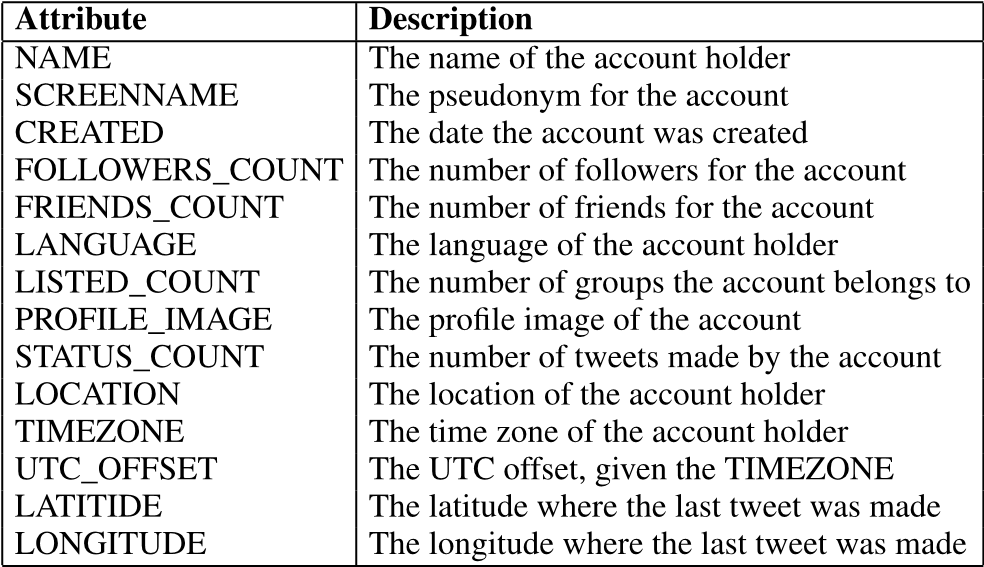
* *Filtering* is mostly reactive: only when a new threat is identified and verified will that sender be added to a blacklist. Similar methods of dealing with spam have been proposed on Twitter to blacklist known malicious URL content and to quarantine known bots [6]. Spam filtering, however, becomes very difficult when spammers use dynamically adaptive and automated strategies to circumvent the proposed methods. This is even more true for SMPs. Humans easily adapt themselves to avoid detection and, in the case of blacklisting, they simply create a new account and fake identity [16] as soon as the current detected account is blacklisted.
* Besides filtering techniques, *rules* have been established to identify fake accounts during detection. Examples of such rules are based on words (such as ‘win’) that are known to belong to spam within the messages [30], [33]. If a message contains such a word or number of words, it is regarded as spam. These same rules have been applied to SMPs with success [17]. The problem, however, is that new words are created constantly, and abbreviated words are common on SMPs, such as ‘lol’ meaning ‘laugh out loud’. This is problematic in the sense that detection rules are usually outdated. More adaptive rules were proposed on SMPs by means of pattern matching [1]. For example, if an account has been tweeting about three or more trending topics, or if an account took part in trending topics but is less than a day old, it can be classified as fake [18]. On Facebook, Fire *et al.* [34] scored friends for deceptiveness by using rules based on similar relationships, tagging, and chat history with others. These rules have success in detecting bot accounts but fail to detect actual fake human accounts. Human behaviour is deemed to be more random [35] than that of bot accounts [14] and thus hard to represent by means of rules.
* *Supervised machine learning* models have been proposed to detect fake accounts. For email spam detection [36], supervised classification machine models like support vector machines (SVMs), decisions trees, Naïve Bayes and neural networks were proposed by

Tuteja [36]. Features were engineered, as input for the models, based on the header and content of the body of the email [36]. For SMS spam detection [37], ten features, inter alia SMS length, were engineered by Choudhary and Jain [37]. These features predicted SMS spam with great success by using supervised machine learning models like random forest, decision trees, J48, logistic regression, and Naïve Bayes. Cresci*et al.* [16] proposed a supervised machine learning model based on those attributes describing the identity of an account only, to detect bots on SMPs. Gupta *et al.* [9] in turn suggested that behaviour, such as the frequency of messages and time of day, provides enough information to detect bots successfully through supervised machine learning models. Supervised machine learning models require a label included in the corpus to predict the expected outcome [38].

* Various *semi-supervised machine learning* models have been proposed. Amongst others, Ebrahimi*et al.* [39] compared a one-class support vector machine model to a Naïve Bayes machine learning model and showed how the one-class SVM outperforms the binary classification model when one of the classes is the minority. The norm is to train a one-class SVM on the minority class [39], [40]. In SMPs it is not practical to mine the minority class consisting of fake accounts [39] or be certain that an account is indeed deceptive [41]. Semisupervised machine learning models require a clear boundary between classes [42].
* *Unsupervised machine learning* was successfully applied by Gu*el al.* [43], Wu *et al.* [44], and Yahyazadeh and Abadi [45]. Their research showed how clustering, which is a common unsupervised machine learning method, can be used to detect bots. With unsupervised machine learning, the data is unlabelled, and data are grouped based on similarity [38]. Clustering works well to detects bots as these bots usually share similar characteristics and has the same purpose. Not the same can necessarily be said of fake human accounts.
* Venkatesan*et al.* [46] presented a *reinforcement* proofof-concept model that rewards itself for detecting bots successfully. Spam in SMPs was detecting by Arif*et al.* [47] whereby the importance of features was used to build a better performing set of rules iteratively. Reinforcement machine learning models require feedback from the environment to adjust and improve. This is not readily available in SMPs.

Given these techniques proposed by previous work, the research at hand will focus on supervised machine learning. The reasons being that supervised machine learning are well suited for classification problems [38], is preferred above unsupervised machine learning techniques for bot detection [4], and have shown good results in past research work detecting bots [16]. We also believe that human deceptive accounts are not as common as bots and therefore less likely to be clustered appropriately through unsupervised machine learning approaches.

**TABLE 1.** Twitter attributes used in a previous study [26].



Supervised machine learning algorithms require a dataset of features with a label classifying each row or outcome. Features are thus the input used by supervised machine learning models to predict an outcome. These features can be the attributesfoundviaAPIsthatdescribesasinglepieceofinformation about an SMP account, like the number of friends. Features can also be engineered by combining attributes from an SMP account, past engineered features, and/or domain knowledge. An example of an engineered feature is the combination of the number of friends and followers to present their relationship as a ratio for input to a machine learning model.

Features used by machine learning models are mostly referred to as ‘‘engineered features’’ as they are a combination of attributes and engineered features. There are however exceptions. In the previous study [26] by the authors, only the attributes in Twitter were used [27]. These attributes are shown in Table 1. It was possible with these attributes alone to identify fake accounts generated by humans, but the result was worse than getting the prediction right by chance.

With SMPs, the engineered features are divided into three distinct groups: data describing the *identity* of the account, the *relationships* of the account to others, and lastly the *behaviour* or messages of the account. This is different from email- or SMS-engineered features in that they only consider features pertaining to the header and body of the message. To detect fake bot accounts in SMPs, various combinations of the above three groups of engineered features were applied to the machine learning models. Cresci*et al.* [16] proposed a lightweight classification model based on the identity of the account (thus, excluding its relationships and behaviour). They suggested that features about the identity of an account are sufficient to detect bots. Gupta *et al.* [9] in turn suggested that behaviour, such as the frequency of messages and time of day, provides more information relevant to deception than the identity of the account itself. Detecting the behaviour through sentiment was also successful for specific topics of interest, for example elections [10]. Given the positive results presented by Cresci*et al.* [16] we propose to also use a similar light weight classifier that only includes data describing the identity of an account. When humans are being deceptive the intent could have detrimental consequences to the targeted individual. The sooner the deception can be detected, the better.

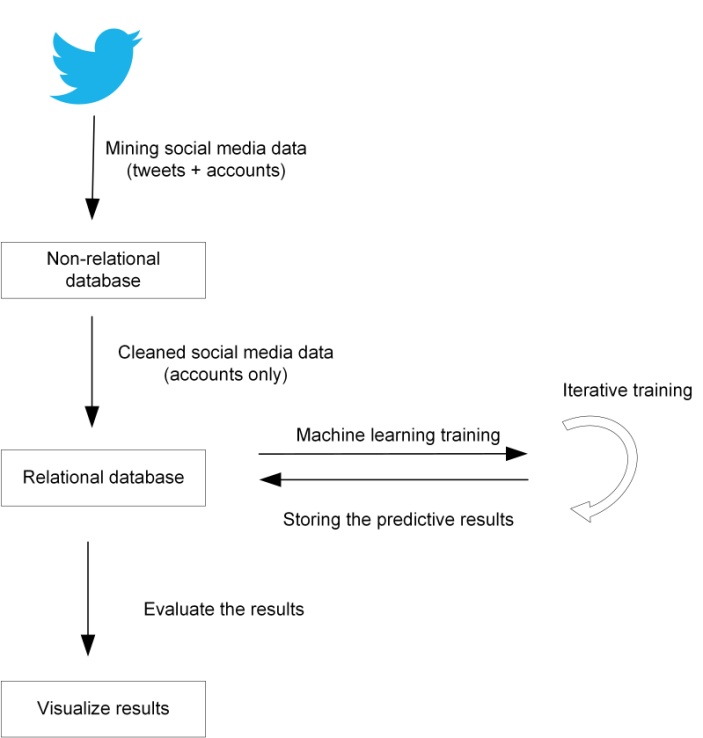
Various options were evaluated to obtain a dataset of potentially deceptive humans, given past research, for SMPs. Some researchers used data from *available datasets*, like paedophiles [48] and extremism groups [15], to label accordingly. To the best of our knowledge, no labelled dataset exists of humans lying specifically about their identity. Other researchers employed the help from the *Amazon Mechanical Turk* crowdsourcing platform where people from the public are paid to label data manually [49], [50]. Due to the volumes of data in SMPs this was not an option for the research at hand. Twitter also indicate *suspended accounts* [15]. This information can be used to label an account. Unfortunately, Twitter does not give the reason for the account being suspended and will thus include more than accounts only suspended for being untruthful about their identity. Furthermore, Zhu [51] listed various other *semi-supervised machine learning* techniques where one approach is to cluster the data first and use the output to label the data. This approach does require the dataset to be clusterable in accord with unsupervised machine learning. Lastly, deceptive accounts can be *manually injected* into the existing corpus gathered. As none of the previous options were viable for the research at hand and collecting deceptive accounts is not practical in real-world examples to date [39], this was the option taken. The accounts were however generated in an informed way.

Past research in psychology has done much work on understanding why people lie [52]–[54]. We looked towards this research showing that people lie about their age [24], [55], gender [25], [55], image [49], location [23], [56], and their name [49], [57]. Using input from the field of psychology, allows for the creation of a set of ‘informed’-deceptive accounts.

Although all previous studies focused on the detection of bots and spam, a few more recent ones have subsequently addressed the detection of deceptive human accounts. Bogdanova*et al.* [48] proposed that the behaviour found in the messages of paedophiles should be used to protect minors. The approach of these researchers relied on sentiment and text analytics to predict deception with an SVM machine learning model. Cyberbullying was addressed by Galán-García*et al.* [8]. They relied on the fact that cyberbullies have a distinctive relationship with the user they target. This knowledge can be used to great effect to identify cyberbullies.Gogoglou*etal.*[58]illustratedhowsocialgraph features and SVMs can aid in highlighting those relationships that are susceptible to online grooming. Lastly, Ferrara *et al.* [15] predicted online extremism using the identity of the user and his/her behaviour features through random forest trees and logistic regression models.

Although the studies mentioned were successful, two shortcomings were found:

* The studies aimed at detecting deceptive humans used engineered features that relied on the behaviour of



**FIGURE 1.** The flow of data to detect identity deception.

the account. Mining account behaviour, which is part of the account’s messages, is computationally costly and time consuming. This is a problem when deception detection should be real-time and not reactive.

* Past research was concerned with accounts being deceptive in general, and not that they were deceptive given their identity. In other words, detecting unsolicited content from accounts constitutes one way to curb deception, but another is to find actual individuals who are lying about their identity.

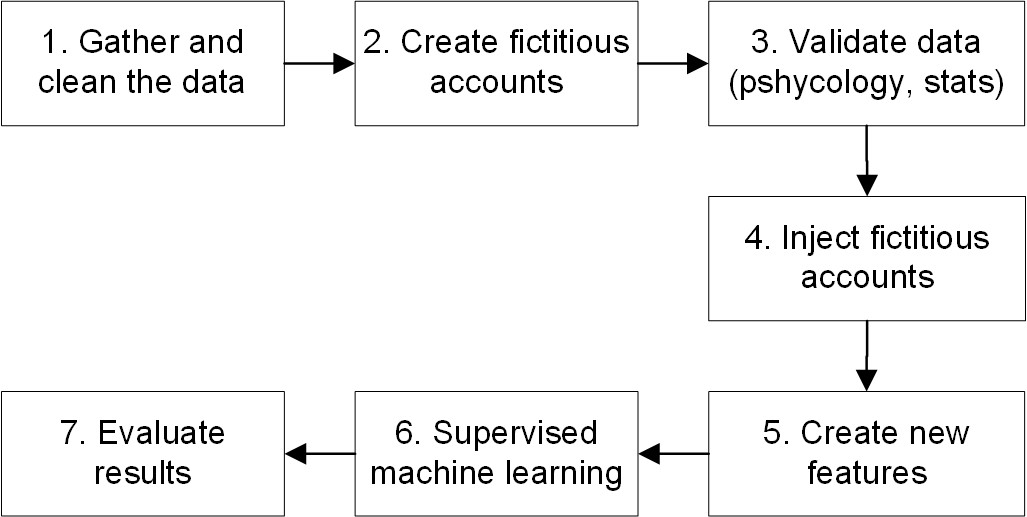
Very little has been done so far to detect actual fake human identities on SMPs, independent of their behaviour. However, bot detection approaches and past research in psychology showed clear promise in detecting fake non-human identities.

We conducted multiple machine learning experiments with existing bot detection approaches to evaluate their efficacy in detecting identity deception committed by humans for malicious purposes. It is hoped that these approaches will address the shortcomings mentioned and serve as a catalyst for uncovering identity deception by humans on SMPs.

# FINDING DECEPTIVE ACCOUNTS

Duringtheprocessofdetectingidentitydeceptionbyhumans, data is mined, cleaned, stored and applied to supervised machine learning models, and the results are evaluated. This flow of data is illustrated in Fig. 1.

For this research, social media data from Twitter was mined using the twitter4J [27] application programming interface (API) and a non-relational database, Hadoop [59]. Non-relational databases cater for the unstructured nature and vast volumes expected from mining Twitter data [60].

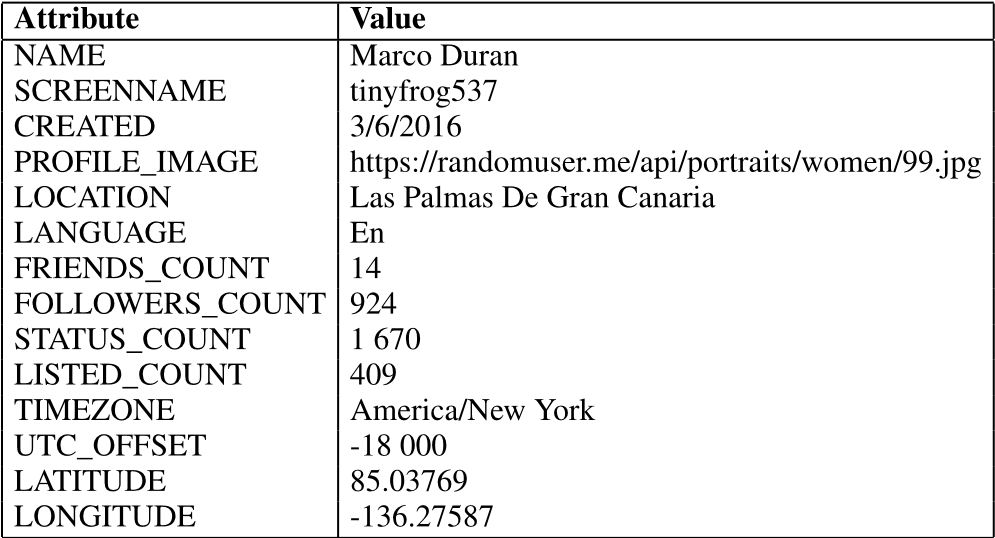
**FIGURE 2.** Research steps.

This resulted in a corpus of over 200 million tweets from 223 796 accounts opened between 2006 and 2017. Thereafter, only data related to the identity of the accounts was injected into a relational database, namely SAP HANA [61]. The account data is required for the proposed experiment to determine if past bot detection approaches can apply to humans as well. A relational in-memory database such as SAP HANA, is well suited to process data at speed [61], althoughanon-relationaldatabasewouldhaveservedthepurpose just as well. Supervised machine learning models were trained with the cleaned data and the results were written back to the relational database. The results were lastly visualised and compared to determine whether features engineered to detect bots could be applied to detect fake human identities on SMPs.

Different research steps were executed to discover deceptive accounts. We were specifically interested in human accounts and, more precisely, identity deception in human accounts. To discover such deceptive accounts, we followed the research steps listed next. Fig. 2 presents the steps in diagram format.

1. The corpus was cleaned of bot and cyborg accounts as far as possible. Previous research suggested various simple rules to distinguish humans from bots [7], [16], for example that human accounts will always have a name and image. We applied the rules determined by Cresci*et al.* [16], such as discarding accounts that have 30 or more followers. The remaining corpus consisted of 154 517 accounts. It is expected that the corpus might still contain some bots and even fake human accounts.Thefewremainingfakeaccountsshouldhave very little effect on the supervised machine learning models as the majority of the class has been classified correctly.
2. 15000fictitiousdeceptiveaccountswerecreatedbythe authors of the current paper. These deceptive accounts were manually created as if by humans and therefore not by bots. The reason being due to ethical considerations for reporting on sensitive content, such as

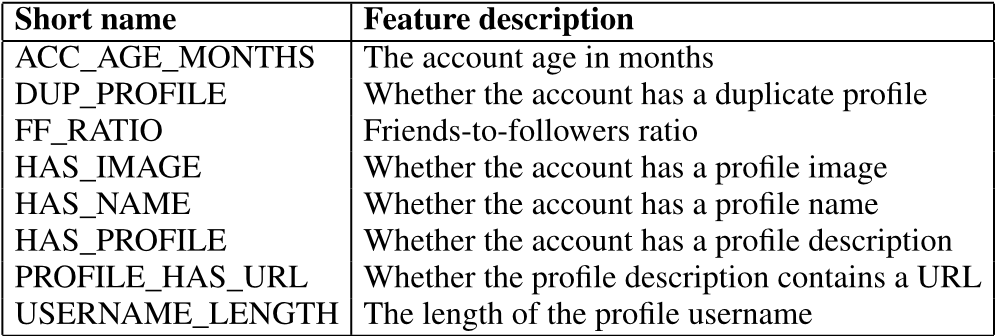
**TABLE 2.** Example of a fictitious deceptive account.



social media data and the lack of an existing dataset for research. By including examples of deceptive accounts, we avoided the risk of reporting sensitive content by mistake. To ensure that the attributes introduced were indeed deceptive, we looked towards research in psychology showing that people lie on their age [24], [55], gender [25], [55], image [49], location [23], [56], and their name [49], [57] the most. We therefore ensured that the new fictitious accounts were deceptive on all 5theseareastocreateaccountsasdeceptiveaspossible. The injected accounts were classified as ‘‘fake’’ and the original corpus accounts were classified as ‘‘human’’. Table 2 presents an example of one fictitious deceptive account. This account is perceived as fake due to a number of potential reasons: the name and screenname are unrelated, the image represents a different gender than suggested by the name, the latitude and longitude coincide somewhere over the Arctic ocean, and New York’s UTC offset is actually -4 and not -18 as suggested.

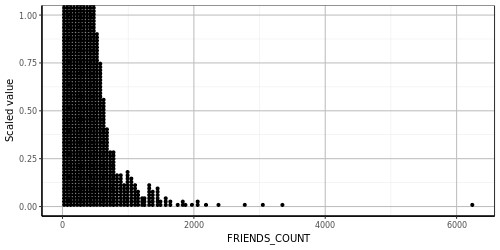
1. To ensure that no bias was introduced with the fictitious accounts, this set was compared with the original corpus by means of two statistical tests. The MannWhitney-U test proofs that the means of the two sets

**TABLE 3.** Engineered features previously used to detect fake bot accounts that were added to the corpus.



are similar per attribute [62]. The Chi Square test for independence proofs that the datasets are not correlated and therefor independent [62]. This means that both the deceptive and original corpus must have similar data and show the same distributions.

1. The next step was to inject these fictitious accounts into the original mined corpus.
2. Up to now, the corpus consisted of attributes found in social media only. The corpus was further enriched with engineered features that were taken from past research [16] and were able to successfully detect bot accounts by using data that describe the identity of the account only. These engineered features are shown in Table 3.
3. Supervised machine learning models were trained, using cross validation and resampling, to detect the accounts denoted as fake in the corpus. The machine learning models used were random forest, boosting, and support vector machines as they had been successfully used in past research towards spam [36], [37] and bot detection [16].
   * For the random forest model, the rf library in R software was used. The random forest model creates many variations of trees. The best outcome will be used to predict identity deception. This model works well for bot detection, as rules are easily represented in tree format [2]. An example would be where accounts that have an image or name are considered human, whereas the rest are denoted as bots. Each of these outcomes represents a different section in the tree.
   * For the boosting model, the Adaboost function in R was used. This is a popular model for detecting bots [10], as different features are assigned different weights to predict the outcome. The model makes use of decision trees [2], which are iteratively adjusted with weights. After each iteration, identity deception detection effectiveness is evaluated. This iterative process is continued until the best result towards identity deception detection is achieved.
   * Lastly, for the support vector model, the svmLinear library in R software was used. This algorithm is typically used to model curves on a



**FIGURE 3.** Distributions of friends.

hyperplane [63], [64]. Trees typically split on single features, whereas SVMs can do so on combinations of features. The SVM algorithm accounts for complex features identifying fake accounts that were missed by trees.

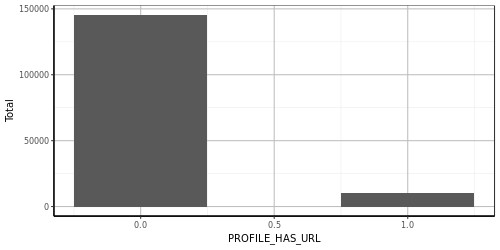
1. Once the supervised machine learning models had been trained, their effectiveness was evaluated. We used the following metrics to determine the effectiveness of each model:
   * Accuracy - this determined how many accounts from the total corpus were correctly identified as fake or not.
   * F1 Score - this was a measure of the harmonic mean between precision and recall. ‘‘Precision’’ refers to how successful the model was at detecting identity deception by humans; ‘‘recall’’ means how successful the model was at filtering out the human accounts that were truthful about their identity.
   * Precision-Recall Area under curve (PR-AUC) - this is the statistical value of the area under the precision-recall curve. The PR-AUC measured how successful the current model

(in totality) was at predicting identity deception by humans.

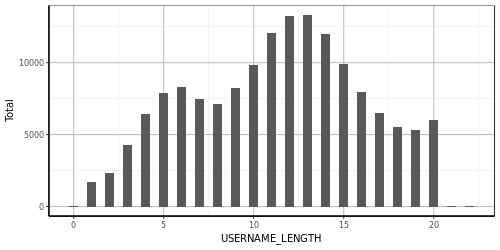
# RESEARCH RESULTS

The engineered features created during step 5 of the research were explored to understand the corpus and it was noted that most accounts had few friends and followers. The distribution of friends is shown in Fig. 3.

Next, the data exploration looked at the profile descriptions of these accounts. The exploration showed that not all accounts had a profile description and that some profile descriptions were shared among accounts. A few profile descriptions also contained URLs. The results of profiles having an URL as part of their profile is shown in Fig. 4 where 0 = no and 1 = yes. These exploratory results showed that even though we are dealing with human accounts only, they still show characteristics known to bots, such as having a URL in their profile description. This further affirmed that research previously conducted to detect fake bot accounts on SMPscouldwellbeapplicabletodetectfakehumanidentities too.



**FIGURE 4.** Number of accounts with a URL in the profile.



**FIGURE 5.** Number of accounts per user name length.

**TABLE 4.** Supervised machine learning results.

|  |  |  |  |
| --- | --- | --- | --- |
|  |  |  |  |
|  |  |  |  |

Furthermore, a tailed distribution seemed to occur regarding the length of user names chosen for accounts. Any outliers on this distribution could indicate potential deception. This pattern, which is illustrated in Fig. 5, is useful, as supervised machine learning models will be able to detect this type of anomaly.

In summary, it was shown that even though the corpus had mostly been cleansed of bots, the engineered features that were used in past research to detect bot accounts were still present in the corpus of human accounts. Examples of these features were the duplicates found in human profile descriptions, the fact that certain human profile descriptions contained URLs, and lastly, the fact that some human profiles had no description at all. These features are just as prevalent in bot accounts. Therefore, it is assumed that the same engineered features and supervised machine learning models can be applied to the human accounts in the hope of detecting fake identities. The supervised machine learning results are shown in Table 4.

The overall accuracy across all machine learning models was very high, with the highest being 87.11%. These results could incorrectly indicate that the supervised machine learning models are good predictors of identity deception by

**TABLE 5.** Entropy results.

|  |  |  |  |
| --- | --- | --- | --- |
|  |  |  |  |
|  |  |  |  |

humans on SMPs. The accuracy measure, however, does not account for wrong predictions and suffers in skewed distributions [65], [66]. The specific corpus was a good example of a skewed distribution because only 15 000 accounts of the total corpus were denoted as fake.

Therefore, we looked towards the F1 score and PRAUC results, which account for getting the predictions wrong. At best, an F1 score of 49.75% was achieved from the random forest (rf) machine learning model and a PRAUC score of 49.90%. These results are just below what one would expect from getting the prediction right by chance (50%).

Furthermore, entropy indicated which of the features contributed most towards identity deception detection. The entropy results from each supervised machine learning model are shown in Table 5. Values are indicated as having an importance out of 100, in which case 100 means the model is completely dependent on the feature. The entropy results showed that username and profile could be dependent features towards the detection of identity deception. In simple terms, it means that humans lie about their name and the description of themselves on SMPs when they are trying to be deceptive.

These findings are very closely related to what is already known about the social sciences and psychology. From psychology we know that deceptive people lie about their name [49], [57] and age [24], [55]. We also learn that people lie about their image [49], location [23], [56], and gender [25], [55].

This can be used in going forward to engineer more features in the hope of uncovering identity deception by humans. For example, does the gender presented in the profile image match the gender of the name provided for the account?

# SUMMARY OF THE RESULTS OBTAINED

What we learned and gathered from the experiment discussed in this paper:

* There are many attributes available in SMPs that describe the identity of an SMP account. For example, the name, location, and profile image.
* Humanaccountsandbotaccountshavesimilarattributes and they share similar characteristics. For example, human accounts have a name and so do accounts generated by bots.
* Features can be engineered from SMP attributes similar to what has been engineered in past research to detect fake accounts generated by bots or computers (for example, whether the account is a duplicate of another).
* Engineered features that have been created to detect fake identities generated by bots can be applied to the existing corpus of human accounts.
* The predictive results from the trained machine learning models only yielded a best F1 score of 49.75%. Given that predicting the correct answer by chance alone would be represented as 50%, this is not optimal.
* Even though only three machine learning models were used in the experiments, these machine learning models have been successfully used in the past towards spam and bot detection. Given the results, these machine learning models are unable to detect fake humans.
* Entropy presents an indication of which engineered features performed well and which not. For example, the fact that an account had a duplicate profile seemed to have made a difference in the accuracy of the predictions.

Based on the predictive results from the machine learning models, it seems that existing features and machine learning models used to detect bot accounts are not suited to detect fake human accounts.

# CONCLUSION

The main contribution of this paper is to show that the engineered features that were previously used to detect fake accounts generated by bots are not similarly successful in the detection of fake accounts generated by humans.

This paper reports on a study that focused on detecting fake accounts created by humans, as opposed to those created by bots. We investigated whether the results from past studies to detect bot accounts could be applied successfully to detect fake human accounts. A corpus of human accounts was enriched with engineered features that had previously been used to successfully detect fake accounts created by bots. These features were applied to various supervised machine learning models. The machine learning models were trained to use engineered features without relying on behavioural data. This made it possible for these machine learning models to be trained on very little data, compared to when behavioural data is included.

The findings indicate that engineered features that were previously used to detect fake accounts generated by bots, at best predicted fake accounts generated by humans with an F1 score of 49.75%. This can be attributed to the fact that humans have different characteristics and behaviours than bots which cannot be modelled similarly. Human fake accounts are also not as common as fake accounts generated by bots. Machine learning models might miss these sparse deceptions in the mass.

Future work will investigate the enrichment of the feature set used in the research for this paper by engineering features from the social sciences knowledge domain - especially psychology. The aim will be to enrich the corpus with new features engineered from the same attributes, as used in this study, found on SMPs. It is hoped that these new features will show better results in the detection of identity deception on SMPs.

# ACKNOWLEDGMENT

Estée van der Walt would like to thank the HPI Future SOC research lab in Potsdam, Germany for their assistance and support during the research. The HPI Future SOC research lab provided the SAP HANA instance, research environment, and storage used during the experimentations.

# REFERENCES

1. S. Gurajala, J. S. White, B. Hudson, B. R. Voter, and J. N. Matthews, ‘‘Profile characteristics of fake Twitter accounts,’’ *Big Data Soc.*, vol. 3, no. 2, p. 2053951716674236, 2016, doi: [10.1177/2053951716674236.](http://dx.doi.org/10.1177/2053951716674236)
2. C. Xiao, D. M. Freeman, and T. Hwa, ‘‘Detecting clusters of fake accounts in online social networks,’’ in *Proc. 8th ACM Workshop Artif. Intell. Secur.*, 2015, pp. 91–101.
3. S. Mainwaring, *We First: How Brands and Consumers Use Social Media to Build a Better World*. New York, NY, USA: Macmillan,

2011.

1. V. S. Subrahmanian*et al.* (2016). ‘‘The DARPA Twitter bot challenge.’’ [Online]. Available: https://arxiv.org/abs/1601.05140
2. Y. Li, O. Martinez, X. Chen, Y. Li, and J. E. Hopcroft, ‘‘In a world that counts: Clustering and detecting fake social engagement at scale,’’ in *Proc. 25th Int. Conf. World Wide Web*, 2016, pp. 111–120.
3. K. Thomas, C. Grier, D. Song, and V. Paxson, ‘‘Suspended accounts in retrospect: An analysis of Twitter spam,’’ in *Proc. ACM SIGCOMM Conf. Internet Meas. Conf.*, 2011, pp. 243–258.
4. T. Tuna *et al.*, ‘‘User characterization for online social networks,’’ *SocialNetw. Anal. Mining*, vol. 6, no. 1, p. 104, 2016.
5. P. Galán-García, J. G. De La Puerta, C. L. Gómez, I. Santos, and

P. G. Bringas, ‘‘Supervised machine learning for the detection of troll profiles in Twitter social network: Application to a real case of cyberbullying,’’ *Logic J. IGPL*. vol. 24, no. 1, pp. 42–53, 2015.

1. A. Gupta, H. Lamba, P. Kumaraguru, and A. Joshi, ‘‘Faking sandy: Characterizing and identifying fake images on Twitter during hurricane sandy,’’ in *Proc. 22nd Int. Conf. World Wide Web*, 2013, pp. 729–736.
2. J. P. Dickerson, V. Kagan, and V. S. Subrahmanian, ‘‘Using sentiment to detect bots on Twitter: Are humans more opinionated than bots?’’ in *Proc. IEEE/ACM Int. Conf. Adv. Social Netw. Anal. Mining (ASONAM)*, Aug. 2014, pp. 620–627.
3. S. Gurajala, J. S. White, B. Hudson, and J. N. Matthews, ‘‘Fake Twitter accounts: Profile characteristics obtained using an activity-based pattern detection approach,’’ in *Proc. Int. Conf. Social Media Soc.*, 2015, p. 9.
4. B. Viswanath*et al.*, ‘‘Towards detecting anomalous user behavior in online social networks,’’ in *Proc. UsenixSecur.*, vol. 14. 2014, pp. 223–238.
5. Z. Chu, S. Gianvecchio, H. Wang, and S. Jajodia, ‘‘Who is tweeting on Twitter: Human, bot, or cyborg?’’ in *Proc. 26th Annu. Comput. Secur. Appl. Conf.*, 2010, pp. 21–30.
6. M. Egele, G. Stringhini, C. Kruegel, and G. Vigna, ‘‘Compa: Detecting compromised accounts on social networks,’’ in *Proc. NDSS*, 2013, pp. 1–17.
7. E. Ferrara, W.-Q. Wang, O. Varol, A. Flammini, and A. Galstyan, ‘‘Predicting online extremism, content adopters, and interaction reciprocity,’’ in *Proc. Int. Conf. Social Inform.*, 2016, pp. 22–39.
8. S. Cresci, R. DiPietro, M. Petrocchi, A. Spognardi, andM. Tesconi, ‘‘Fame for sale: Efficient detection of fake Twitter followers,’’ *Decision Support Syst.*, vol. 80, pp. 56–71, Dec. 2015.
9. F. Benevenuto, G. Magno, T. Rodrigues, and V. Almeida, ‘‘Detecting spammers on Twitter,’’ in *Proc. Collaboration, Electron. Messaging, AntiAbuse Spam Conf. (CEAS)*, vol. 6. 2010, p. 12.
10. H. Kwak, C. Lee, H. Park, and S. Moon, ‘‘What is Twitter, a social network or a news media?’’ in *Proc. 19th Int. Conf. World Wide Web*, 2010, pp. 591–600.
11. Z. Zhang and B. B. Gupta, ‘‘Social media security and trustworthiness: Overview and new direction,’’ *Future Generat. Comput. Syst.*, to be published, doi: [10.1016/j.future.2016.10.007.](http://dx.doi.org/10.1016/j.future.2016.10.007)
12. A. K. Jain and B. B. Gupta, ‘‘Phishing detection: Analysis of visual similarity based approaches,’’ *Secur. Commun. Netw.*, vol. 2017, Jan. 2017, Art. no. 5421046. [Online]. Available: https://doi.org/10.1155/ 2017/5421046
13. R. J. Oentaryo, A. Murdopo, P. K. Prasetyo, and E.-P. Lim, ‘‘On profiling bots in social media,’’ in *Proc. Int. Conf. Social Inform.*, 2016, pp. 92–109.
14. M. Tsikerdekis and S. Zeadally, ‘‘Multiple account identity deception detection in social media using nonverbal behavior,’’ *IEEE Trans. Inf. Forensics Security*, vol. 9, no. 8, pp. 1311–1321, Aug. 2014.
15. J. T. Hancock, ‘‘Digital deception,’’ in *Oxford Handbook of Internet Psychology*. London, U.K.: Oxford Univ. Press, 2007, pp. 289–301.
16. G. A. Wang, H. Chen, J. J. Xu, and H. Atabakhsh, ‘‘Automatically detecting criminal identity deception: An adaptive detection algorithm,’’ *IEEE Trans. Syst., Man, Cybern. A, Syst. Humans*, vol. 36, no. 5, pp. 988–999, Sep. 2006.
17. E. Bergen *et al.*, ‘‘The effects of using identity deception and suggesting secrecy on the outcomes of adult-adult and adult-child or-adolescent online sexual interactions,’’ *Victims Offenders*, vol. 9, no. 3, pp. 276–298, 2014.
18. E. Van der Walt and J. H. P. Eloff, ‘‘Protecting minors on social media platforms—A big data science experiment,’’ in *Proc. HPI Cloud Symp.*, 2015, pp. 1–78.
19. Twitter. (2017). *Twitter API*. [Online]. Available: https://dev.twitter. com/overview/api
20. Facebook. (2017). *The Facebook Graph API*. [Online]. Available: https:// developers.facebook.com/docs/graph-api/overview
21. Instagram. (2017). *Instagram API*. [Online]. Available: https://www. instagram.com/developer/
22. M. Jiang, P.Cui, and C. Faloutsos, ‘‘Suspiciousbehavior detection: Current trends and future directions,’’ *IEEE Intell. Syst.*, vol. 31, no. 1, pp. 31–39, Jan. 2016.
23. Wikipedia. (2017). *Spamming*. [Online]. Available: https://en. wikipedia.org/wiki/Spamming
24. M. S. Alishahi, M. Mejri, and N. Tawbi, ‘‘Clustering spam emails into campaigns,’’ in *Proc. Int. Conf. Inf. Syst. Secur. Privacy (ICISSP)*, Feb. 2015, pp. 90–97.
25. P. Hayati and V. Potdar, ‘‘Toward spam 2.0: An evaluation of Web 2.0 anti-spam methods,’’ in *Proc. 7th IEEE Int. Conf. Ind. Inform. (INDIN)*, Jun. 2009, pp. 875–880.
26. M.Fire,D.Kagan,A.Elyashar,andY.Elovici,‘‘Friendorfoe?Fakeprofile identification in online social networks,’’ *Social Netw. Anal. Mining*, vol. 4, no. 1, p. 194, 2014.
27. N. M. Radziwill and M. C. Benton. (2016). ‘‘Bot or not? Deciphering time maps for tweet interarrivals.’’ [Online]. Available: https:// arxiv.org/abs/1605.06555
28. S. K. Tuteja, ‘‘A survey on classification algorithms for email spam filtering,’’ *Int. J. Eng. Sci.*, vol. 6, no. 5, pp. 5937–5940, 2016.
29. N. Choudhary and A. K. Jain, ‘‘Towards filtering of SMS spam messages using machine learning based technique,’’ in *Advanced Informatics for Computing Research*. Singapore: Springer, 2017, pp. 18–30.
30. S. Miller and C. Busby-Earle, ‘‘The impact of different botnet flow feature subsets on prediction accuracy using supervised and unsupervised learning methods,’’ *J. Internet Technol. Secured Trans.*, vol. 5, no. 2, pp. 474–485, Jun. 2016.
31. M. Ebrahimi, C. Y. Suen, O. Ormandjieva, and A. Krzyzak, ‘‘Recognizing predatory chat documents using semi-supervised anomaly detection,’’ *Electron. Imag.*, vol. 2016, no. 17, pp. 1–9, 2016.
32. C. Bellinger, S. Sharma, and N. Japkowicz, ‘‘One-class versus binary classification: Which and when?’’ in *Proc. 11th Int. Conf. Mach. Learn. Appl. (ICMLA)*, vol. 2. Dec. 2012, pp. 102–106.
33. L. M. Jupe, A. Vrij, G. Nahari, S. Leal, and S. A. Mann, ‘‘The lies we live: Using the verifiability approach to detect lying about occupation,’’ *J. Articles Support Null Hypothesis*, vol. 13, no. 1, pp. 1–13, 2016.
34. C. Beleites, K. Geiger, M. Kirsch, S. B. Sobottka, G. Schackert, and R. Salzer, ‘‘Raman spectroscopic grading of astrocytoma tissues: Using soft reference information,’’ *Anal. Bioanal. Chem.*, vol. 400, no. 9, p. 2801, 2011.
35. G. Gu, R. Perdisci, J. Zhang, and W. Lee, ‘‘BotMiner: Clustering analysis of network traffic for protocol-and structure-independent botnet detection,’’ in *Proc. USENIX Secur. Symp.*, vol. 5. 2008, pp. 139–154.
36. W. Wu, J. Alvarez, C. Liu, and H.-M. Sun, ‘‘Bot detection using unsupervised machine learning,’’ *Microsyst. Technol.*, vol. 24, no. 1, pp. 209–217, 2018.
37. M. Yahyazadeh and M. Abadi, ‘‘BotOnus: An online unsupervised method for botnet detection,’’ *ISC Int. J. Inf. Secur.*, vol. 4, no. 1, pp. 51–62, 2012.
38. S. Venkatesan, M. Albanese, A. Shah, R. Ganesan, and S. Jajodia, ‘‘Detecting stealthy botnets in a resource-constrained environment using reinforcement learning,’’ in *Proc. Workshop Moving Target Defense*, 2017, pp. 75–85.
39. M. H. Arif, J. Li, M. Iqbal, and K. Liu, ‘‘Sentiment analysis and spam detection in short informal text using learning classifier systems,’’ in *Soft Computing*. Berlin, Germany: Springer, 2017, pp. 1–11.
40. D. Bogdanova, P. Rosso, and T. Solorio, ‘‘Exploring high-level features for detecting cyberpedophilia,’’ *Comput. Speech Lang.*, vol. 28, no. 1, pp. 108–120, 2014.
41. K. Stanton, S. Ellickson-Larew, and D. Watson, ‘‘Development and validation of a measure of online deception and intimacy,’’ *Per. Individual Differences*, vol. 88, pp. 187–196, Jan. 2016.
42. M.A.Al-Garadi,K.D.Varathan,andS.D.Ravana,‘‘Cybercrimedetection in online communications: The experimental case of cyberbullying detection in the Twitter network,’’ *Comput. Hum. Behav.*, vol. 63, pp. 433–443, Oct. 2016.
43. X. Zhu, ‘‘Semi-supervised learning literature survey,’’ Dept. Comput. Sci., Univ. Wisconsin-Madison, Madison, WI, USA, Tech. Rep. TR 1530, 2005.
44. M. Drouin, D. Miller, S. M. J. Wehle, and E. Hernandez, ‘‘Why do people lie online? ‘Because everyone lies on the Internet,’’’ *Comput. Hum. Behav.*, vol. 64, pp. 134–142, Nov. 2016.
45. R. Rong, D. Houser, and A. Y. Dai, ‘‘Money or friends: Social identity and deception in networks,’’ *Eur. Econ. Rev.*, vol. 90, pp. 56–66, Nov. 2016.
46. V. L. Rubin, ‘‘Deception detection and rumor debunking for social media,’’ in*TheSAGEHandbookofSocialMediaResearchMethods*.NewburyPark, CA, USA: SAGE, 2017, p. 342.
47. C. L. Toma, J. T. Hancock, and N. B. Ellison, ‘‘Separating fact from fiction: An examination of deceptive self-presentation in online dating profiles,’’ *Per. Social Psychol. Bull.*, vol. 34, no. 8, pp. 1023–1036, 2008.
48. A. Caspi and P. Gorsky, ‘‘Online deception: Prevalence, motivation, and emotion,’’ *CyberPsychol. Behav.*, vol. 9, no. 1, pp. 54–59, 2006.
49. S. Utz, ‘‘Types of deception and underlying motivation: What people think,’’ *Social Sci. Comput. Rev.*, vol. 23, no. 1, pp. 49–56, 2005.
50. A. Gogoglou, Z. Theodosiou, T. Kounoudes, A. Vakali, and Y. Manolopoulos, ‘‘Early malicious activity discovery in microblogs by social bridges detection,’’ in *Proc. IEEE Int. Symp. Signal Process. Inf. Technol. (ISSPIT)*, Dec. 2016, pp. 132–137.
51. A.S. Foundation. (2014). *The Hadoop Distributed File System: Architecture and Design*. [Online]. Available: http://hadoop.apache. org/docs/r2.4.1/hadoop-project-dist/hadoop-hdfs/HdfsDesign.html
52. R. Kannadasan, R. Shaikh, and P. Parkhi, ‘‘Survey on big data technologies,’’*Int. J. Adv. Eng. Res.*, vol. 3, no. 3, pp. 1–11, 2013.
53. *SAP Hana*, SAP SE, Walldorf, Germany, 2017.
54. C. R. Kothari, *Research Methodology: Methods and Techniques*. New Age International, 2004. [Online]. Available: https://books.google.co.uk/ books?id=hZ9wSHysQDYC
55. A. M. Meligy, H. M. Ibrahim, and M. F. Torky, ‘‘Identity verification mechanism for detecting fake profiles in online social networks,’’ *Int. J. Comput. Netw. Inf. Secur.*, vol. 9, no. 1, pp. 31–39, 2017.
56. S. T. Peddinti, K. W. Ross, and J. Cappos. (2017). ‘‘Mining anonymity: Identifying sensitive accounts on Twitter.’’ [Online]. Available: https:// arxiv.org/abs/1702.00164
57. G. Menardi and N. Torelli, ‘‘Training and assessing classification rules with imbalanced data,’’ *Data Mining Knowl. Discovery*, vol. 28, no. 1, pp. 92–122, 2014.
58. L. A. Jeni, J. F. Cohn, and F. De La Torre, ‘‘Facing imbalanced data–recommendations for the use of performance metrics,’’ in *Proc. Hum. Assoc. Conf. Affect. Comput. Intell. Interact. (ACII)*, Sep. 2013, pp. 245–251.

 ESTÉE VAN DER WALT was born in

Johannesburg, South Africa, in 1977. She received the B.Sc. and M.Sc. degrees in computer science from the University of Johannesburg, South Africa, in 1997 and 2001 respectively. She is currently pursuing the Ph.D. degree in information technology with the University of Pretoria, South Africa.

Her research interests include cybersecurity and identity deception on social media platforms, and

in particular,the protection ofhumans on thesebig data platforms.She makes use of machine learning and data mining techniques to not only understand this type of deception but also find a solution to detect and warn authorities about identity deception.

She was a recipient of the Best Poster Award at the 3rd International Conference on Information Systems Security and Privacy that was hosted in Porto, Portugal, in 2017.

JANELOFF received the Ph.D. degree in computer science in 1985. Up to 2015, he was appointed as the Research Director for SAP Research, Africa. He is currently appointed as the Deputy Dean for research and postgraduate studies with the Faculty of Engineering, Built Environment and IT, and as a Full Professor in computer science with the University of Pretoria, South Africa.

Since 2007, he has been an Associate Editor of the *Computers & Security journal* and an Editorial Member of the *International Computer Fraud & Security bulletin* (Elsevier). He is an internationally recognized Researcher and has published 113 peer reviewed papers with 3537 citations.