**

***VIRTUAL SUMMER INTERNSHIP***

*June – 2020*

*Student Details:*

*Mohit Jain,17103297,B8*

*Submitted in partial fulfillment of the Degree of Bachelors of Technology in Computer Science and Engineering*

*Department of Computer Science and Engineering & Information Technology*

**ACKNOWLEDGEMENT**

I would like to express special thanks of gratitude to my mentor Ms. Sakshi for teaching me, helping me with the Project as well as guiding me with the relevant resources, and ultimately providing me with an opportunity to make and present this report.

**INTRODUCTION**

The use of social networks such as Facebook, Twitter, Google+, Instagram, and LinkedIn is on the rise. Individuals and organizations use social networks to express their views, advertise their products, and express future policies of their companies and organizations. By expanding the use of social networks, malicious users seek to violate the privacy of other users and abuse their names and credentials by creating fake accounts, which has become a concern for users. Hence, social networks providers are trying to detect malicious users and fake accounts in order to eliminate them from social networking environments. Creating fake accounts in social networks causes more damage than any other cybercrime.

Removing fake accounts has attracted the attention of many researches; thus, extensive researches have been carried out on the identification of fake accounts in social networks. Different approaches are proposed and to find fake accounts based on attribute similarity, similarity of friend networks, profile analysis for a time interval, and similarity of attribute together with IP address.The problems in discovering fake accounts in previous researches are stated below:

1. The use of similarity measures that do not consider the strength of the network of friendships shared among users, while we believe that the more the shared friendship network of the two users is connected, the greater the similarity of the users is.

2. Due to the high volume of information, the use of machine learning techniques leads to overfitting problem.

3. In some previous works, in order to implement the proposed methods, some normal users were assumed to be fake and this is because the number of fake users is lower than that of the fake users in datasets. The above assumption is completely wrong and, thus, will dispute the logic of learning.

Graph Analysis is used in many applications, such as displaying circuit diagrams to detect SHAPE, image matching, and social network analysis. The networks' graph is analyzed in order to solve most of the social network problems. Therefore, graph similarity measures reduce the complexity of graph analysis problems by using different techniques.

A **data set** (or **dataset**) is a collection of data. In the case of tabular data, a data set corresponds to one or more database table where every column of a table represents a particular variable, and each row corresponds to a given record of the data set in question. The data set lists values for each of the variables, such as height and weight of an object, for each member of the data set. Each value is known as a datum. Data sets can also consist of a collection of documents or files.

In the open data discipline, data set is the unit to measure the information released in a public open data repository. The European Open Data portal aggregates more than half a million data sets. In this field other definitions have been proposed, but currently there is not an official one. Some other issues (real-time data sources, non-relational data sets, etc.) increases the difficulty to reach a consensus about it.

**Problem Statement**

The current report explores various aspects associated with the social networking accounts like Facebook, Instagram, Twitter, Linkedin.And tries to find out whether account can be considered fake or not! The specific research questions explained in this study are as follows:

1. Relation between profile pic with fake accounts through graph and Pearson correlation Coefficient.
2. Relation between follows with fake accounts through graph and Pearson correlation Coefficient.
3. Relation between followers with fake accounts through graph and Pearson correlation Coefficient.
4. Relation between Username along with different scenarios with fake accounts through graph and Pearson correlation Coefficient.
5. Relation between External URL along with different scenarios with fake accounts through graph and Pearson correlation Coefficient.
6. Relation between Posts along with different scenarios with fake accounts through graph and Pearson correlation Coefficient.

**Technology Involved**

Let us begin our learning on Data Science with Python by first understanding of data science. Data science is all about finding and exploring data in the real world and using that knowledge to solve business problems.

When it comes to data science, we need some sort of programming language or tool, like Python. Although there are other tools for data science, like R and SAS, we will focus on Python and how it is beneficial for data science in this article. Python as a programming language has become very popular in recent times. It has been used in data science, IOT, AI, and other technologies, which has added to its popularity.  Python is used as a programming language for data science because it contains costly tools from a mathematical or statistical perspective. It is one of the significant reasons why data scientists around the world use Python. If you track the trends over the past few years, you will notice that Python has become the programming language of choice, particularly for data science.

There are several other reasons why Python is one of the most used programming languages for data science, including:

* Speed - Python is relatively faster than other programming languages
* Availability - There are a significant number of packages available that other users have developed, which can be reused
* Design goal - The syntax roles in Python are intuitive and easy to understand, thereby helping in building applications with a readable codebase

 The Python Language Reference describes the exact syntax and semantics of the Python language, this library reference manual describes the standard library that is distributed with Python. It also describes some of the optional components that are commonly included in Python distributions. Python’s standard library is very extensive, offering a wide range of facilities as indicated by the long table of contents listed below. The library contains built-in modules (written in C) that provide access to system functionality such as file I/O that would otherwise be inaccessible to Python programmers, as well as modules written in Python that provide standardized solutions for many problems that occur in everyday programming. Some of these modules are explicitly designed to encourage and enhance the portability of Python programs by abstracting away platform-specifics into platform-neutral APIs. Python is one of the most popular languages used by data scientists and software developers alike for data science tasks. It can be used to predict outcomes, automate tasks, streamline processes, and offer business intelligence insights.

## *Data Mining*

#### 1. [Scrapy](https://github.com/scrapy/scrapy)

One of the most popular Python data science libraries, Scrapy helps to build crawling programs (spider bots) that can retrieve structured data from the web – for example, URLs or contact info. It's a great tool for scraping data used in, for example, Python machine learning models.

#### 2. [BeautifulSoup](https://www.crummy.com/software/BeautifulSoup/bs4/doc/)

BeautifulSoup is another really popular library for web crawling and data scraping. If you want to collect data that’s available on some website but not via a proper CSV or API, BeautifulSoup can help you scrape it and arrange it into the format you need.

## *Data Processing and Modeling*

#### 3. [NumPy](https://github.com/numpy/numpy)

NumPy (Numerical Python) is a perfect tool for scientific computing and performing basic and advanced array operations.The library offers many handy features performing operations on n-arrays and matrices in Python. It helps to process arrays that store values of the same data type and makes performing math operations on arrays (and their vectorization) easier. In fact, the vectorization of mathematical operations on the NumPy array type increases performance and accelerates the execution time.

#### 4. [SciPy](https://github.com/scipy/scipy)

This useful library includes modules for linear algebra, integration, optimization, and statistics. Its main functionality was built upon NumPy, so its arrays make use of this library. SciPy works great for all kinds of scientific programming projects (science, mathematics, and engineering). It offers efficient numerical routines such as numerical optimization, integration, and others in submodules. The extensive documentation makes working with this library really easy.

#### 5. [Pandas](https://github.com/pandas-dev/pandas)

Pandas is a library created to help developers work with "labeled" and "relational" data intuitively. It's based on two main data structures: "Series" (one-dimensional, like a list of items) and "Data Frames" (two-dimensional, like a table with multiple columns). Pandas allows converting data structures to DataFrame objects, handling missing data, and adding/deleting columns from DataFrame, imputing missing files, and plotting data with histogram or plot box. It’s a must-have for data wrangling, manipulation, and visualization.

#### 6. [Keras](https://github.com/keras-team/keras)

Keras is a great library for building neural networks and modeling. It's very straightforward to use and provides developers with a good degree of extensibility. The library takes advantage of other packages, (Theano or TensorFlow) as its backends. Moreover, Microsoft integrated CNTK (Microsoft Cognitive Toolkit) to serve as another backend. It's a great pick if you want to experiment quickly using compact systems – the minimalist approach to design really pays off!

#### 7. [SciKit-Learn](https://github.com/scikit-learn/scikit-learn)

This is an industry-standard for data science projects based in Python. Scikits is a group of packages in the SciPy Stack that were created for specific functionalities – for example, image processing. Scikit-learn uses the math operations of SciPy to expose a concise interface to the most common machine learning algorithms. Data scientists use it for handling standard machine learning and data mining tasks such as clustering, regression, model selection, dimensionality reduction, and classification. Another advantage? It comes with quality documentation and offers high performance.

#### 8. [PyTorch](https://github.com/pytorch/pytorch)

PyTorch is a framework that is perfect for data scientists who want to perform deep learning tasks easily. The tool allows performing tensor computations with GPU acceleration. It's also used for other tasks – for example, for creating dynamic computational graphs and calculating gradients automatically. PyTorch is based on Torch, which is an open-source deep learning library implemented in C, with a wrapper in Lua.

#### 9. [TensorFlow](https://github.com/tensorflow/tensorflow)

TensorFlow is a popular Python framework for machine learning and deep learning, which was developed at Google Brain. It's the best tool for tasks like object identification, speech recognition, and many others. It helps in working with artificial neural networks that need to handle multiple data sets. The library includes various layer-helpers (tflearn, tf-slim, skflow), which make it even more functional. TensorFlow is constantly expanded with its new releases – including fixes in potential security vulnerabilities or improvements in the integration of TensorFlow and GPU.

#### 10. [XGBoost](https://github.com/dmlc/xgboost)

Use this library to implement machine learning algorithms under the Gradient Boosting framework. XGBoost is portable, flexible, and efficient. It offers parallel tree boosting that helps teams to resolve many data science problems. Another advantage is that developers can run the same code on major distributed environments such as Hadoop, SGE, and MPI.

## *Data Visualization*

#### 11. [Matplotlib](https://github.com/matplotlib/matplotlib)

This is a standard data science library that helps to generate data visualizations such as two-dimensional diagrams and graphs (histograms, scatterplots, non-Cartesian coordinates graphs). Matplotlib is one of those plotting libraries that are really useful in data science projects — it  provides an object-oriented API for embedding plots into applications. It's thanks to this library that Python can compete with scientific tools like MatLab or Mathematica. However, developers need to write more code than usual while using this library for generating advanced visualizations. Note that popular plotting libraries work seamlessly with Matplotlib.

#### 12. [Seaborn](https://github.com/mwaskom/seaborn)

Seaborn is based on Matplotlib and serves as a useful Python machine learning tool for visualizing statistical models – heatmaps and other types of visualizations that summarize data and depict the overall distributions. When using this library, you get to benefit from an extensive gallery of visualizations (including complex ones like time series, joint plots, and violin diagrams).

#### 13. [Bokeh](https://github.com/bokeh/bokeh)

This library is a great tool for creating interactive and scalable visualizations inside browsers using JavaScript widgets. Bokeh is fully independent of Matplotlib. It focuses on interactivity and presents visualizations through modern browsers – similarly to Data-Driven Documents (d3.js). It offers a set of graphs, interaction abilities (like linking plots or adding JavaScript widgets), and styling.

#### 14. [Plotly](https://github.com/plotly/plotly.py)

This web-based tool for data visualization that offers many useful out-of-box graphics – you can find them on the [Plot.ly website](https://plot.ly/). The library works very well in interactive web applications. Its creators are busy expanding the library with new graphics and features for supporting multiple linked views, animation, and crosstalk integration.

#### 15. [pydot](https://github.com/pydot/pydot)

This library helps to generate oriented and non-oriented graphs. It serves as an interface to Graphviz (written in pure Python). You can easily show the structure of graphs with the help of this library. That comes in handy when you're developing algorithms based on neural networks and decision trees.

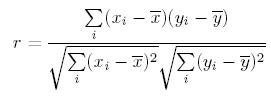
**Working**

In this analysis, We will plot the graph between two columns of a dataset and try to get a trend through graph and also use Pearson Formula of Correlation Coefficient.

**Formula used :**

The Pearson correlation coefficient is used to measure the strength of a linear association between two variables,where the value r=1 means a perfect positive correlation.

Equation



**Imported Libraries**

import pandas as pd

import numpy as np

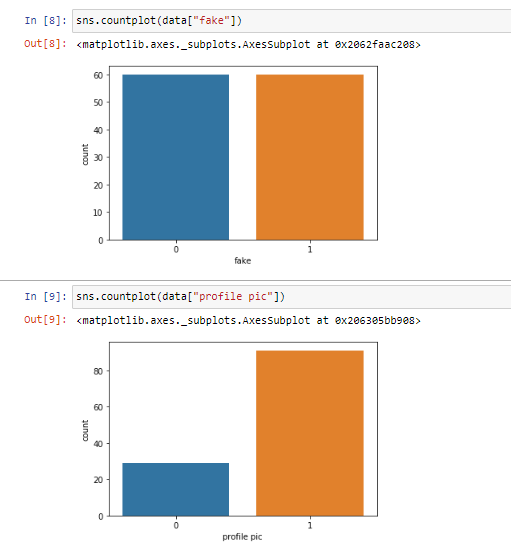
import seaborn as sns

import matplotlib.pyplot as plt

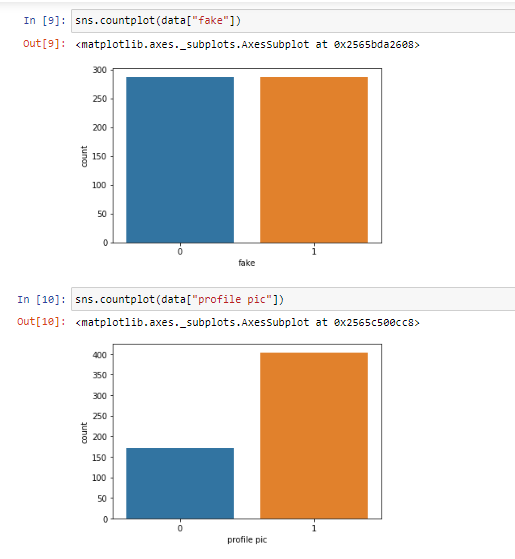
**Datasets:**

1. Insta\_fake1
2. Insta\_fake2

**1). Trend between Profile Pic & Fake**

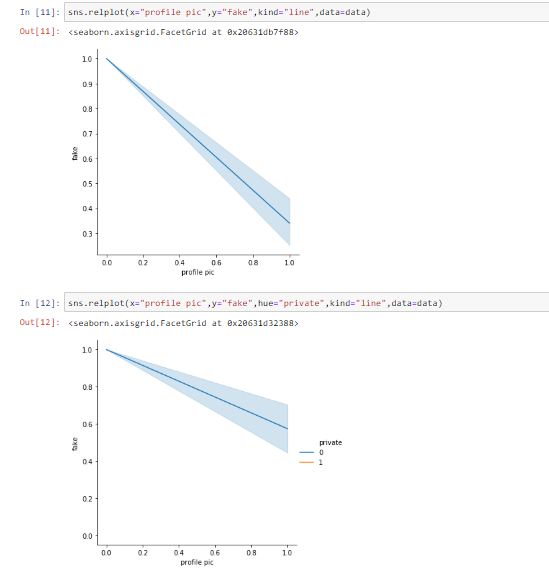


**Fig (1) Shows count of columns “profile pic” and “fake” of dataset insta\_fake1 where 0 denotes “No” and 1 denotes “yes”.**

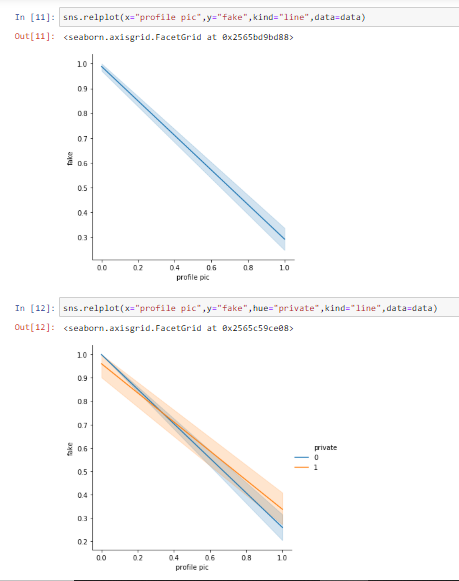


**Fig (2) Shows count of columns “profile pic” and “fake” of dataset insta\_fake2 where 0 denotes “No” and 1 denotes “yes”.**

Plotting shows that no of fake accounts are equal to no of valid accounts. Whereas most of the accouns are found to have profile pic.



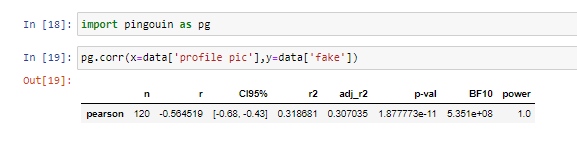
**Fig (3) Shows the trend between “profile pic” and “fake” with and without using a dependency of “private” accounts of dataset “insta\_fake1” where 0 denotes “No” and 1 denotes “Yes”.**



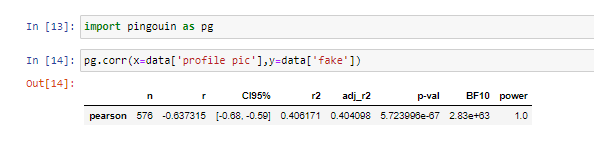
**Fig (4) Shows the trend between “profile pic” and “fake” with and without using a dependency of “private” accounts of dataset “insta\_fake2” where 0 denotes “No” and 1 denotes “Yes”.**

Plotting shows that there is more chance of account to be fake when there is no profile pic and same trend can be seen with using Privacy as dependency.

**Mathematical Computation**



**Fig (5) Shows Pearson correlation Coefficient between “Profile Pic” and “fake” of dataset “insta\_fake1”.**



**Fig (6) Shows Pearson correlation Coefficient between “Profile Pic” and “fake” of dataset “insta\_fake2”.**

n = sample size;

r = correlation coefficient(-0.56,-0.63), negative slope but quite high.

CI95% = 95% confidence interval around the correlation coefficient.

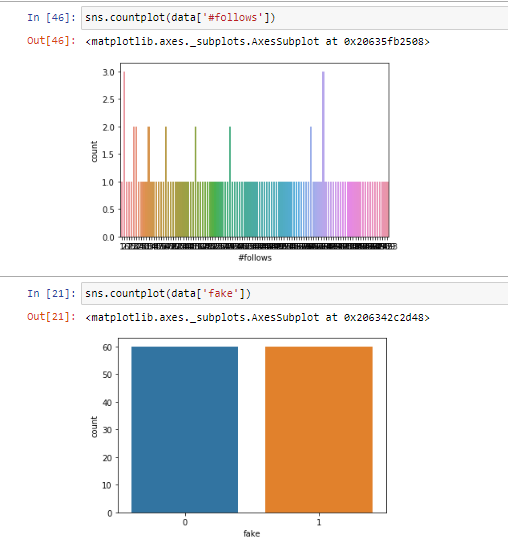
r 2 & adj\_r2 = r squared and adjust r-squared respectively.

p val = p-value of test. we can reject hypothesis that two variables are not correlated if p-val <0.05. In our case, it’s quite high.

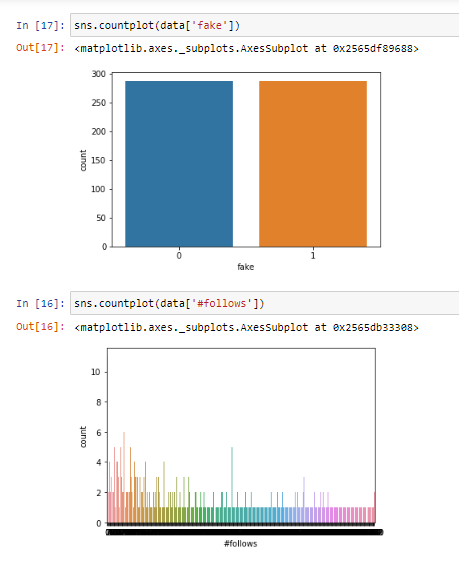
BF10 = Bayes factor of the test which measures the statistical significance of the test. Since the value is so large that it indicates that two variables are strongly correlated.

Power = achieved power of the test. Higher the value the more robust our test is. In this case,1 means we can greatly confident in our test ability to detect significant effect.

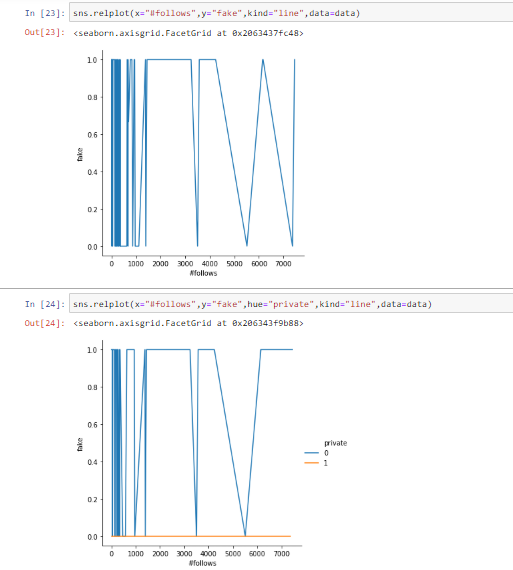
**2). Follows vs Fake**



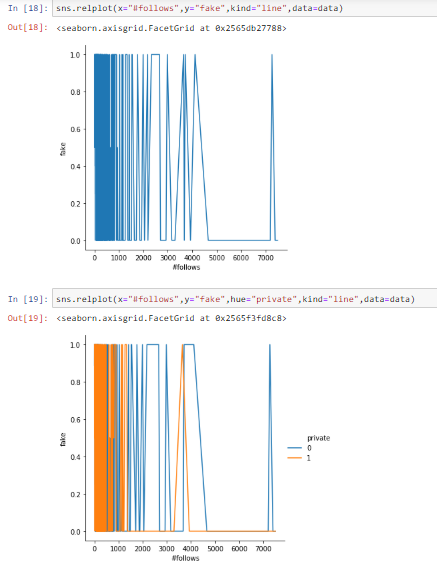
**Fig (1.2.1) Denoting count of both column**



**Fig (2.2.1) Denoting count of both column**

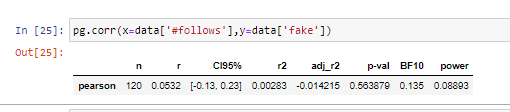


**Fig (1.2.2) Denotes the trend between “#follows” and “fake” with and without using a dependency of “private”**

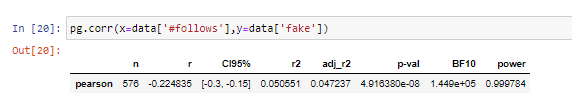


**Fig (2.2.2) Denotes the trend between “#follows” and “fake” with and without using a dependency of “private”**

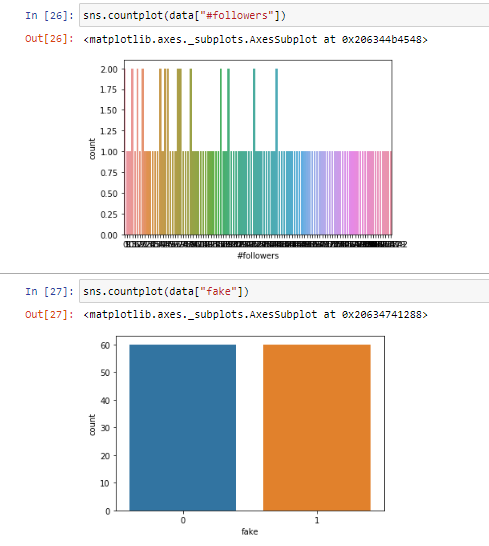
**Mathematical Formula**



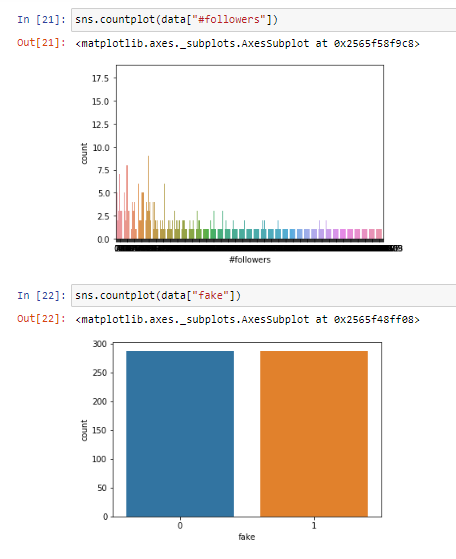
**Fig (1.2.3) Pearson correlation Coefficient between “#follows” and “fake”**

**Fig (2.2.3) Pearson correlation Coefficient between “#follows” and “fake”**

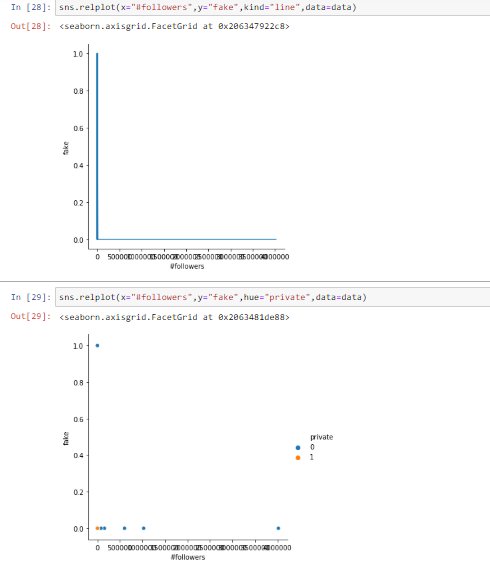
**3). Followers Vs Fake**



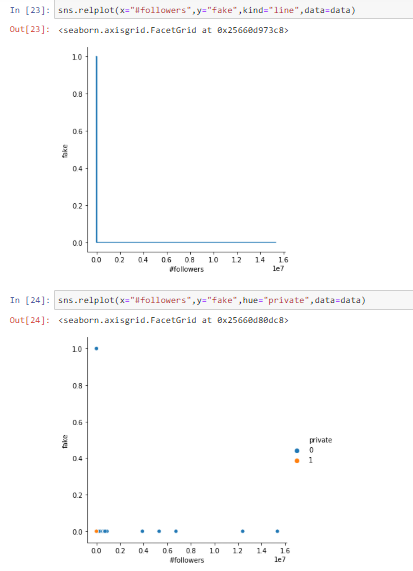
**Fig (1.3.1) Denoting count of both column**



**Fig (2.3.1) Denoting count of both column**

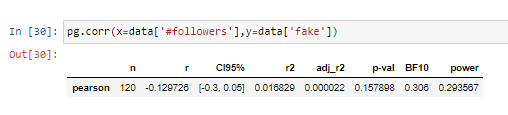


**Fig (1.3.2) Denotes the trend between “#followers” and “fake” with and without using a dependency of “private”**

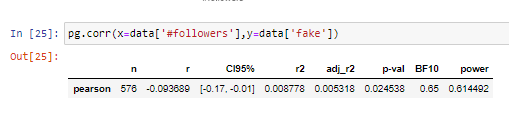


**Fig (2.3.2) Denotes the trend between “#followers” and “fake” with and without using a dependency of “private”**

**Mathematical computation:**

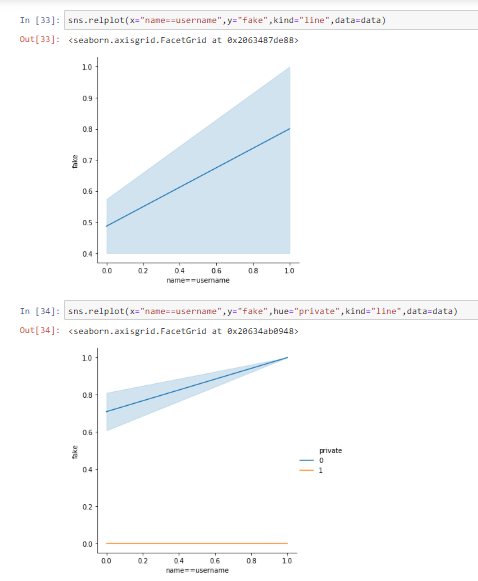


**Fig (1.3.3) Pearson correlation Coefficient between “#followers” and “fake”**

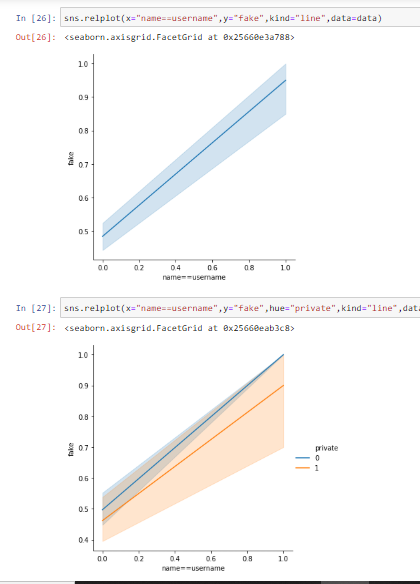


**Fig (2.3.3) Pearson correlation Coefficient between “#followers” and “fake”**

**4). Username Vs Fake**

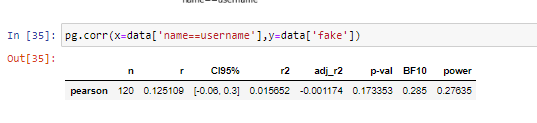


**Fig (1.4.1) Denotes the trend between “Username” and “fake” with and without using a dependency of “private”**

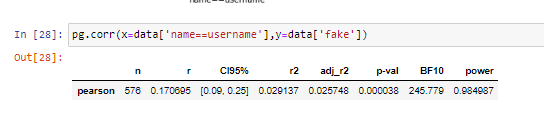


**Fig (2.4.1) Denotes the trend between “username” and “fake” with and without using a dependency of “private”**

**Mathematical Computation**

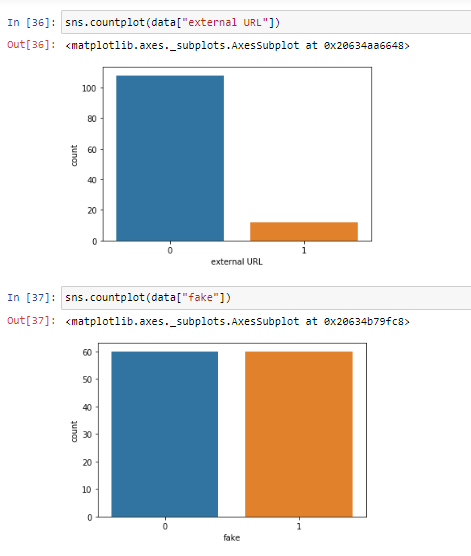


**Fig (1.4.2) Pearson correlation Coefficient between “Username” and “fake”**



**Fig (2.4.2) Pearson correlation Coefficient between “Username” and “fake”**

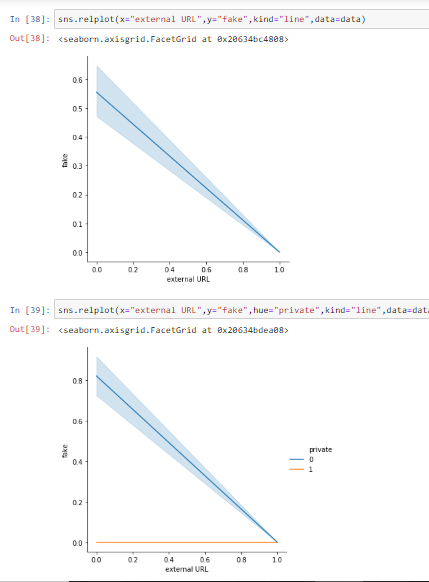
**5).External URL vs Fake**



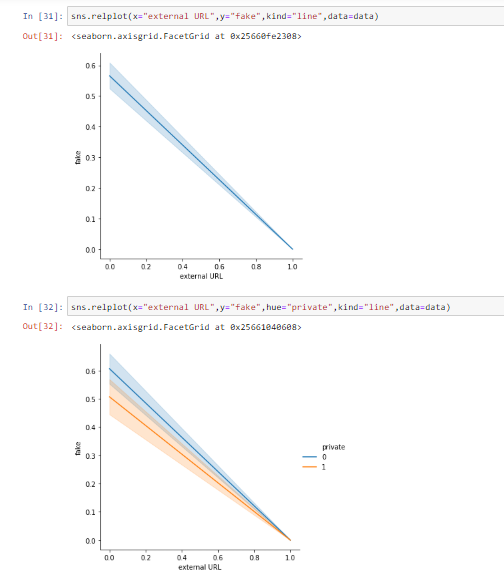
**Fig (1.5.1) Denoting count of both column**



**Fig (2.5.1) Denoting count of both column**

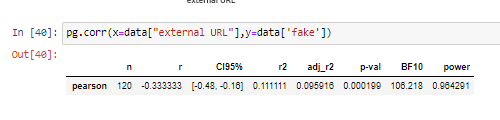


**Fig (1.5.2) Denotes the trend between “External URL” and “fake” with and without using a dependency of “private”**

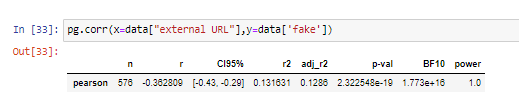


**Fig (2.5.3) Denotes the trend between “External URL” and “fake” with and without using a dependency of “private”**

**Mathematical Computation**

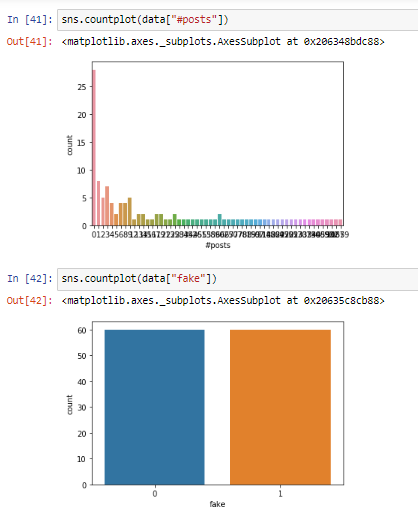


**Fig (1.5.3) Pearson correlation Coefficient between “External URL” and “fake”**

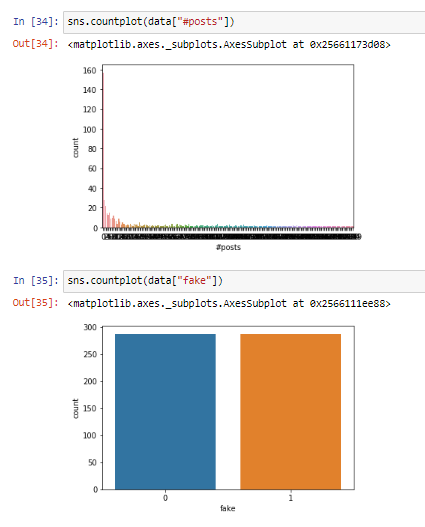


**Fig (2.5.3) Pearson correlation Coefficient between “External URL” and “fake”**

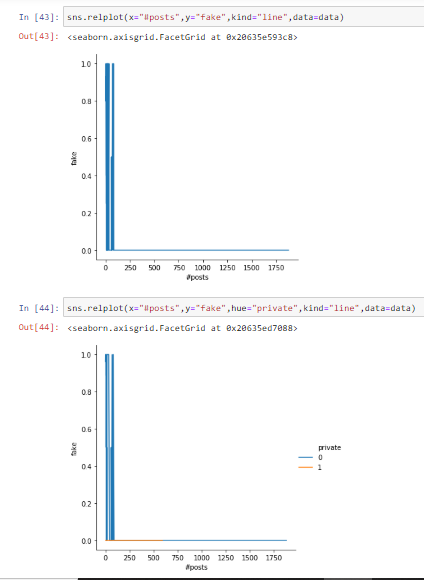
**6). Post vs Fake**



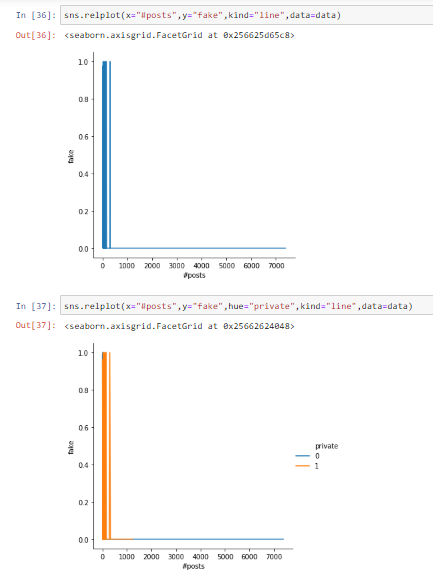
**Fig (1.6.1) Denoting count of both column**



**Fig (2.6.1) Denoting count of both column**

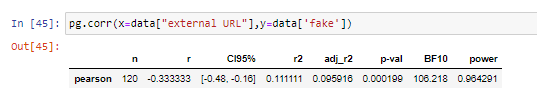


**Fig (1.6.2) Denotes the trend between “Posts” and “fake” with and without using a dependency of “private”**

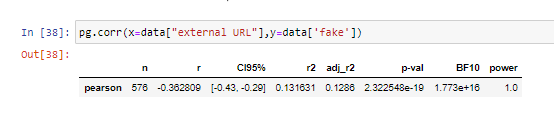


**Fig (2.6.3) Denotes the trend between “Posts” and “fake” with and without using a dependency of “private”**

**Mathematical Computation**



**Fig (1.5.3) Pearson correlation Coefficient between “Posts” and “fake”**



**Fig (2.6.3) Pearson correlation Coefficient between “Posts” and “fake”**

**Important Results :**

1.Accounts with more profile pic have less probability.

2.Accounts with External URL have less chances of being in fake.

3.Accounts with more post,followers,follows have less chance of fake.

4.Private accounts are less fakes

5. Accounts with big description are less fake

**Day to Day Report**

# Week (1): -

**Day (1-3) : -** Search and gain knowledge about the features of the technology, I am going to use that is **Python**. Watched some Introduction videos of python data analysis from which I came to know that this is a good library for Data Analysis.

**Day (4-6) : -** Gone through some important features of Python libraries which I am gonna use in implementing python codes and making them simpler.

**Day(7) :** Submitted week 1 Assignment and had a discussion about future work and got the queries resolved related to libraries.

# Week( 2& 3) : -

**Day (1-5) :** I searched about dataset on Google, Kaggle and Youtube. How they are created, their file format and their uses in modern algorithms.

**Day(6-13) :** I downloaded small size dataset on Social networking sites and others too and gone through them and their entities.

**Day(14) :** Submitted week 2 Assignment and had a discussion about future work and got the queries resolved related to datasets and their uses.

# Week (4 & 5) : -

**Day (1-4) :** I searched about suitable python software to analyse datasets and I installed setup of Python Anaconda (Spyder) and learn about Jupyter Notebook and got basic idea of this.

**Day(5-13) :** I analysed my 5 datasets with graphs and finding relation between two entities and got the possibility of account to be fake seeing is Name, Pic, Address, Privacy and other things.

**Day(14) :** I submitted my work and had conversation with me about the work and future scopes.