WALEED48581@GMAIL.COM

BSCS 8TH A. (MORNING)

FEDRAL URDU UNIVERSITY OF ARTS SCIENCE & TECHNOLOGY

Fake news detection through image document

FYP-P2 DOCUMENTATION BY: REHAN QAYYUM & WALEED MUGHAL

**Chapter: 5**

**Text Extraction from Image:**

The text extraction from image is the first module of the research which is done by using a tool name ‘tesseract’, that is especially for optical character recognition OCR. It recognizes and read text embedded in images.

Here are some examples of extracting and then reading text from image using python-tesseract tool:

A picture containing text, screenshot, computer, monitor

Description automatically generatedA picture containing text, screenshot, monitor, computer

Description automatically generatedA picture containing text, screenshot, monitor, screen

Description automatically generated

**Data Gathering**

The data for the research can be gathered by using many techniques like interviews, observations, questionnaires. In detecting fake news through images data is needed which contains large amount of information that will be suitable for training the model. For this purpose, there are many online websites that offer the free datasets for researchers or a researcher can make the dataset on their own, but this will be a very devastating process and may include many malfunctions, to avoid these hurdles dataset from online website will be preferable. Kaggle.com is used for the gathering of dataset that was needed in the project.

A picture containing text, screenshot, monitor, computer

Description automatically generated

A picture containing text, screenshot, computer, monitor

Description automatically generated

Here, some important libraries were imported that are very useful in the project regarding the cleansing, gathering, processing, training, and testing data.

**Data Processing**

The data processing is very important phase in the research based and AI based projects. Data processing techniques are then applied to the selected dataset like data cleaning, finding null values, replacing, or filling null values, removing garbage, etc. like were done in the research as:A picture containing text, screenshot, computer, monitor

Description automatically generatedA picture containing text, screenshot, computer, monitor

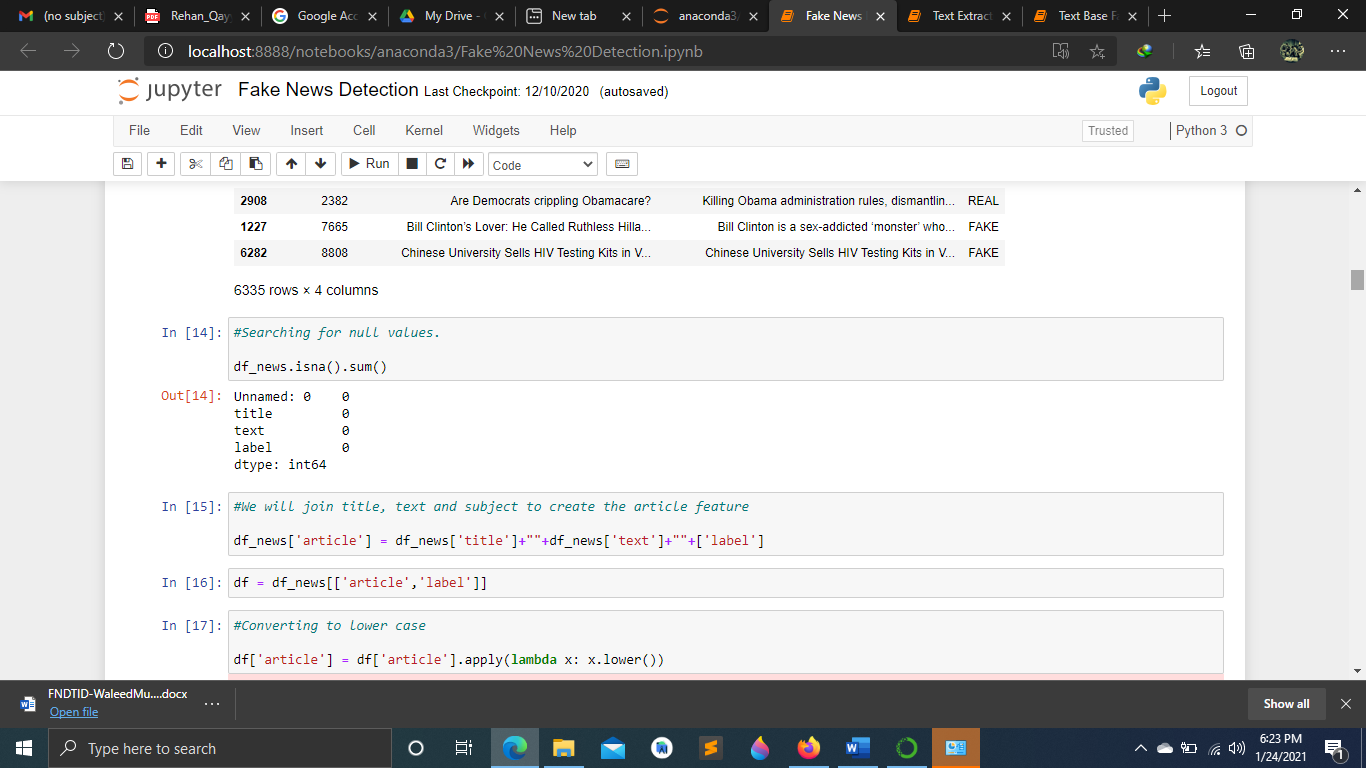
Description automatically generated

A picture containing text, screenshot, computer, monitor

Description automatically generatedA picture containing text, screenshot, computer, monitor

Description automatically generated

**Finding null values**



A picture containing text, screenshot, computer, monitor

Description automatically generated

**Data after applying pre-processing techniques.**

**Feature Extraction:**

Feature extraction through dataset by applying different models, classifiers. The **‘sklearn.feature\_extraction’** module can be used to extract features in a format supported by machine learning algorithms from datasets consisting of formats such as **text** and **image**.

**Loading Features:**

The class DictVectorizer can be used to convert feature arrays represented as lists of standard Python dict objects to the NumPy/SciPy representation used by scikit-learn estimators. While not particularly fast to process, Python’s dict has the advantages of being convenient to use, being sparse (absent features need not be stored) and storing feature names in addition to values.

DictVectorizer implements what is called one-of-K or “one-hot” coding for categorical (aka nominal, discrete) features. Categorical features are “attribute-value” pairs where the value is restricted to a list of discrete of possibilities without ordering (e.g. topic identifiers, types of objects, tags, names…).

**Count Vectorizer:**

Convert a collection of text documents to a matrix of token counts. This implementation produces a sparse representation of the counts using scipy.sparse.csr\_matrix.

**Tf-idf term:**

In a large text corpus, some words will be very present (e.g., “the”, “a”, “is” in English) hence carrying very little meaningful information about the actual contents of the document. If we were to feed the direct count data directly to a classifier those very frequent terms would shadow the frequencies of rarer yet more interesting terms.

To re-weight the count features into floating point values suitable for usage by a classifier it is very common to use the tf–idf transform. Tf means term-frequency while tf–idf means term-frequency times inverse document-frequency.

Using the TfidfTransformer’s default settings, TfidfTransformer(norm='l2', use\_idf=True, smooth\_idf=True, sublinear\_tf=False) the term frequency, the number of times a term occurs in a given document, is multiplied with idf component.

A picture containing text, screenshot, monitor, computer

Description automatically generatedA picture containing text, screenshot, computer, monitor

Description automatically generatedA picture containing text, screenshot, computer, monitor

Description automatically generated

**Decoding Text Files:**

Text is made of characters, but files are made of bytes. These bytes represent characters according to some encoding. To work with text files in Python, their bytes must be decoded to a character set called Unicode. Common encodings are ASCII, Latin-1 (Western Europe), KOI8-R (Russian) and the universal encodings UTF-8 and UTF-16. Many others exist**.**

The text feature extractors in scikit-learn know how to decode text files, but only if we tell them what encoding the files are in. The **Count Vectorizer** takes an encoding parameter for this purpose. For modern text files, the correct encoding is probably UTF-8, which is therefore the default (encoding="utf-8").

If the text we are loading is not actually encoded with UTF-8, however, we will get a UnicodeDecodeError. The vectorizers can be told to be silent about decoding errors by setting the decode\_error parameter to either "ignore" or "replace".

**Training Dataset by Applying Classifiers:**

In the research we apply 2 different classifiers on the dataset these are:

1. Naïve Bayes.
2. SVM.

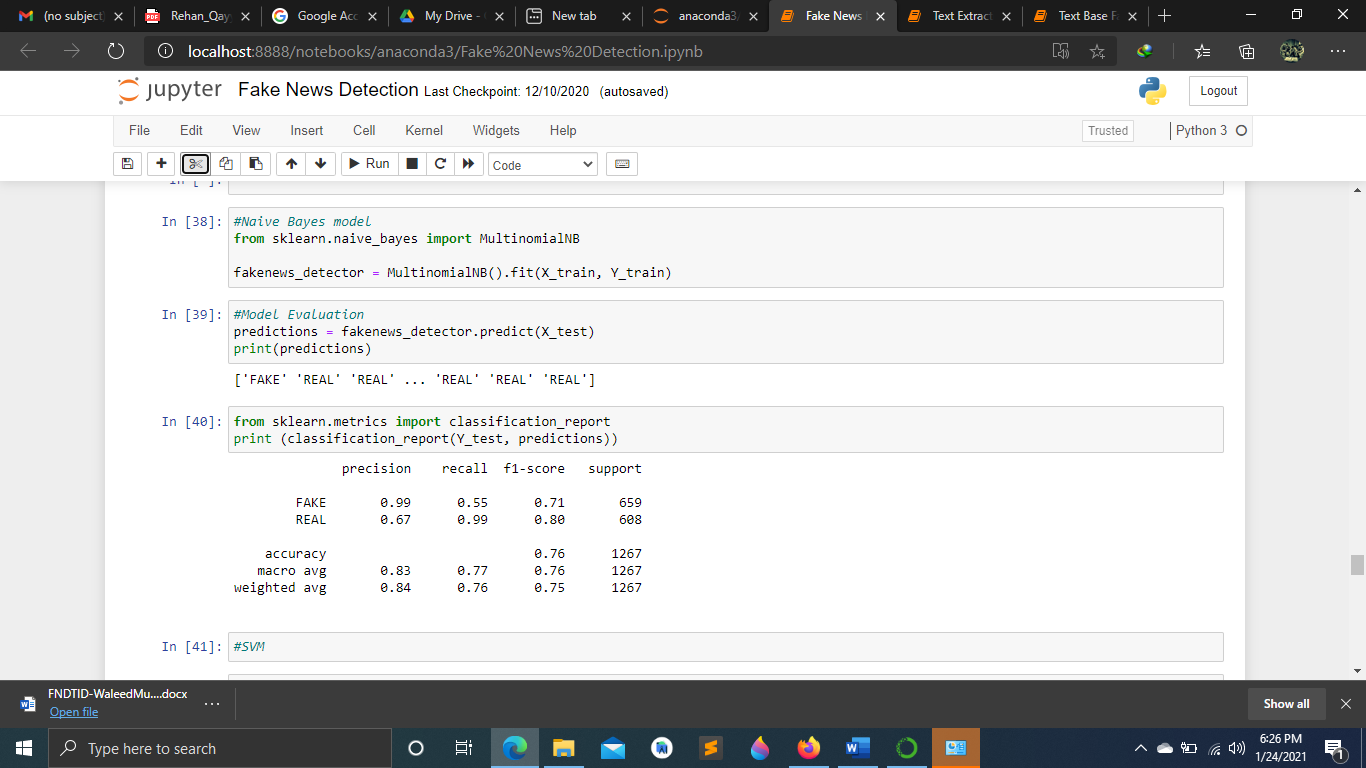
**Naïve Bayes:** Naive Bayes classifiers are a collection of classification algorithms based on Bayes’ Theorem. It is not a single algorithm but a family of algorithms where all of them share a common principle, i.e., every pair of features being classified is independent of each other.

Naïve bayes usually divide dataset into two parts, namely, **feature matrix** and the **response vector.**

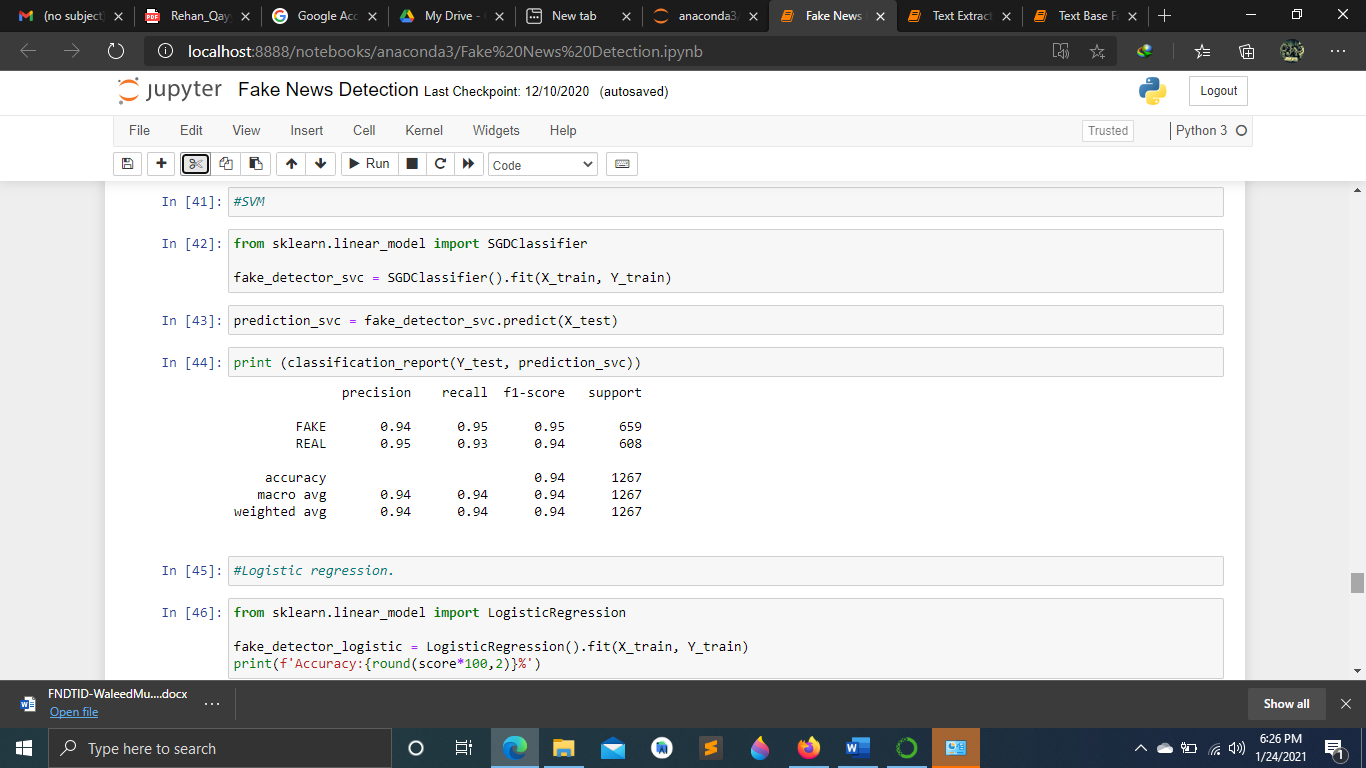
* **Feature matrix** contains all the vectors(rows) of dataset in which each vector consists of the value of dependent features.
* **Response vector** contains the value of class variable (prediction or output) for each row of feature matrix.

A picture containing text, screenshot, monitor, computer

Description automatically generated



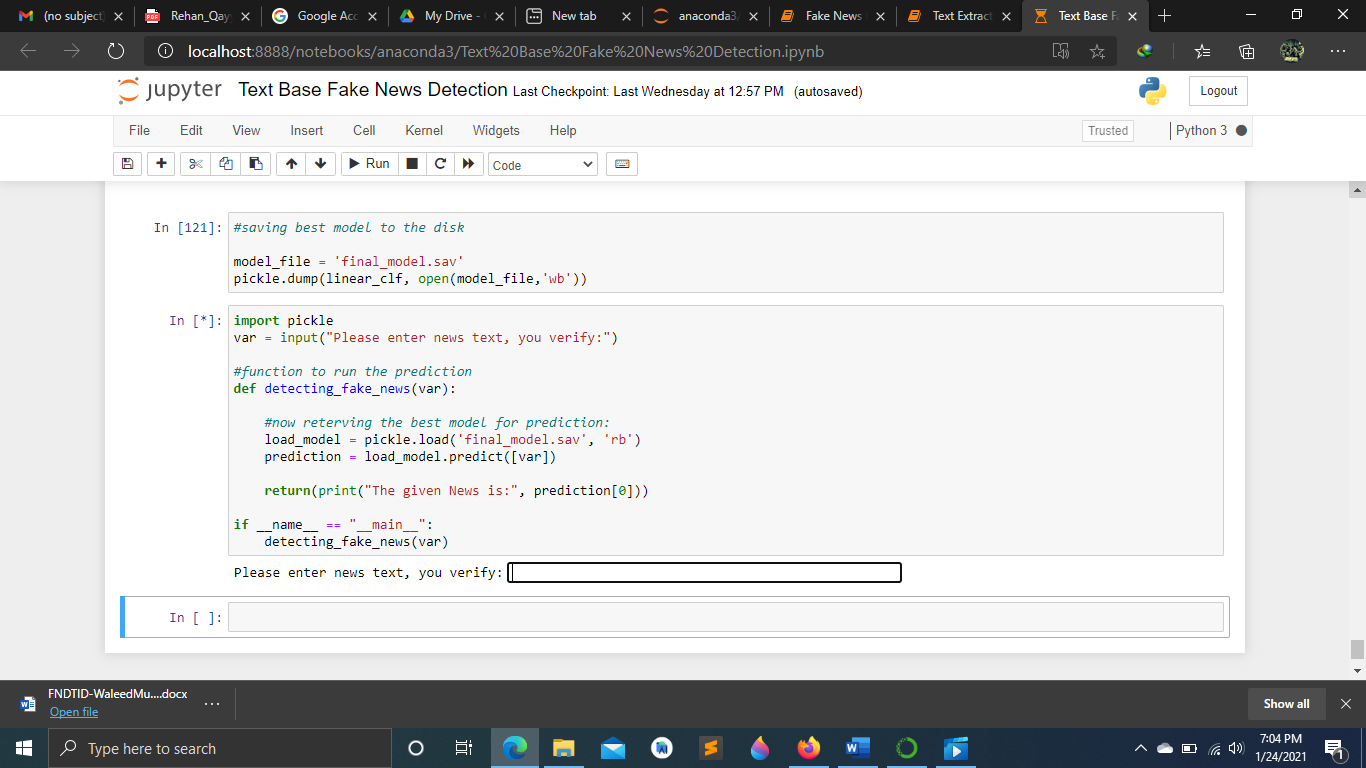
**SVM(Support Vector Machine):**

****

**Chapter 6**

**RESULTS & DISCUSSION**

**Result:**

****

**Chapter 7**

**CONCLUSIONS**

This work aims to improve an accuracy and performance of Fake News Detection Through Image Document. We evaluated variants of a SVM & Naïve Bayes to avoid complex pre-processing, costly feature extraction. Through extensive evaluation using a dataset, the present work suggests the role of various hyperparameters. We also verified that fine tuning of hyper-parameters is essential in improving an accuracy and performance of our model. We achieved an accuracy rate of 94-96% using Naïve Bayes model and SVM model respectively, which gives better results. TensorFlow was used while building these models. Alongside these many python libraries were used (imported) for several purposes and different techniques were applied to achieve our desired accuracy of the model. Comparing to other research methods, this architecture works better by improving the accuracy of the classification models. Utilizing these machine learning techniques, a high amount of accuracy can be obtained.

**References:**

* <https://scikit-learn.org/stable/modules/generated/sklearn.feature_extraction.text.CountVectorizer.html>
* <https://scikit-learn.org/stable/modules/feature_extraction.html#text-feature-extraction>