LAB PRACTICE - 4: MINI-PROJECT (DATA SCIENCE)

TOPIC: FAKE NEWS DETECTION

COURSE FACULTY: Prof. Leena A. Deshpande

GROUP MEMBERS:

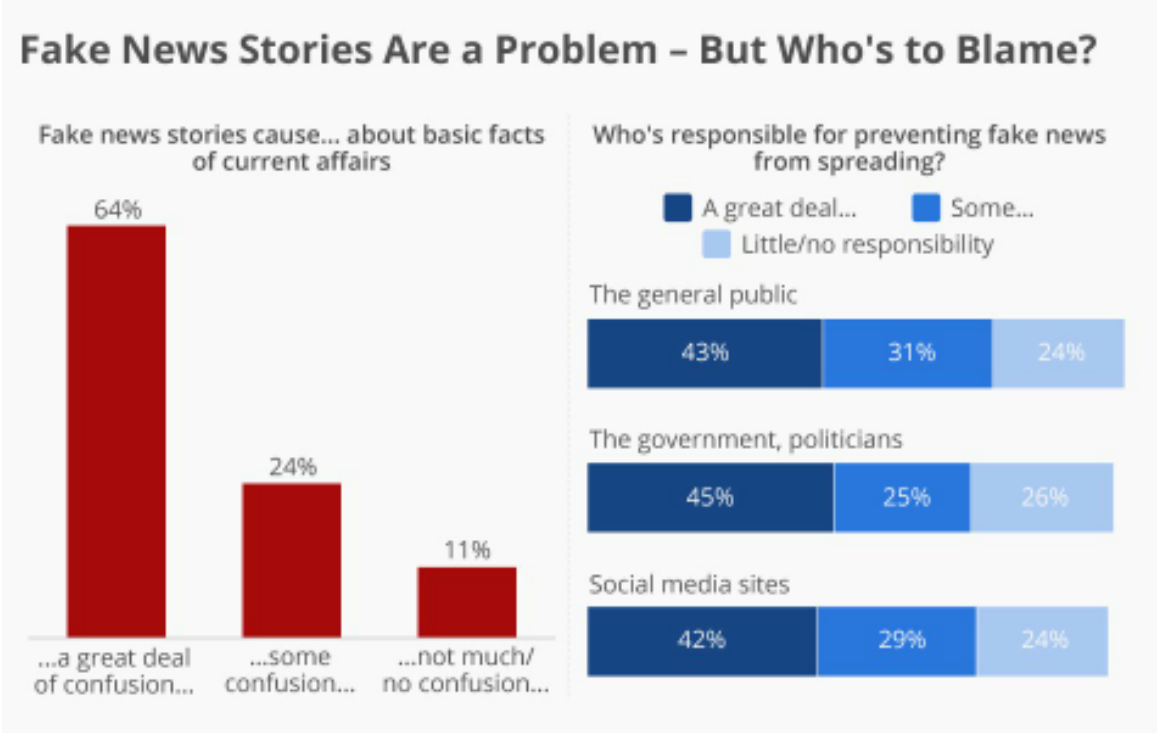
- Pratiksha Navarkle – 21820071 (B2)
- Pallavi Sable - 21820076 (B2)
- Mitali Mehta - 17U081 (B1)
- Sachi Kothari - 17U550 (B2)

CONTENT:

- Description
- Introduction
- Objective
- How the data is created?
- Process flow of project:
- Concepts
- Pre-processing of Data
- Classifier: Models
- Code
- Output

DESCRIPTION:

- Do you trust all the news you hear from social media?
- All news are not real, right?
- So how will you detect the fake news? (mainly spread via social media)
- The answer is Python.
- By practicing this advanced python project of detecting fake news, you will easily make a difference between real and fake news.



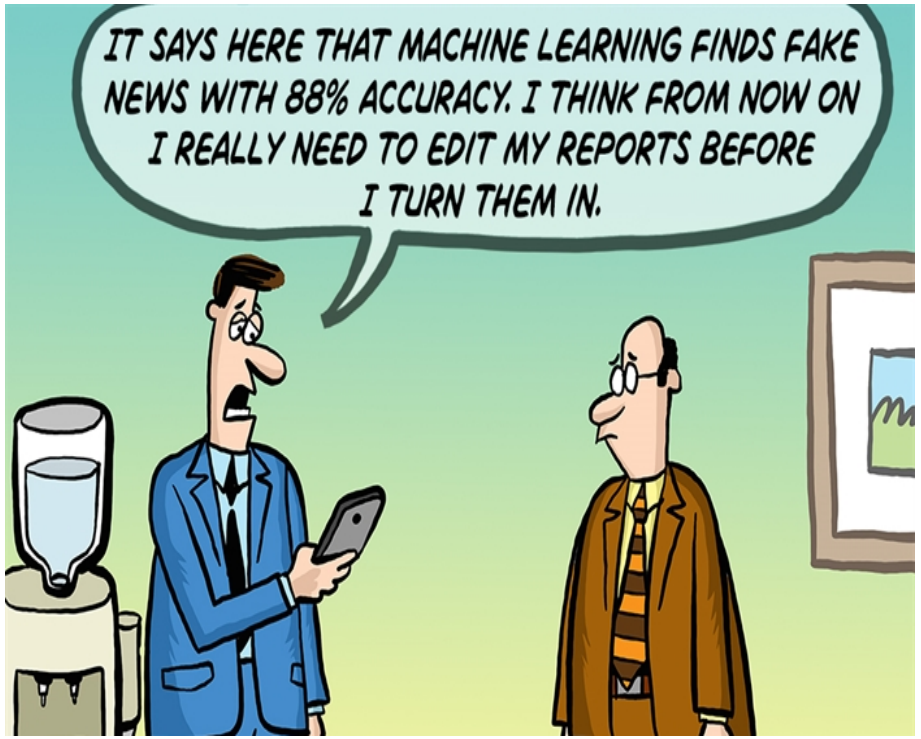
INTRODUCTION:

- WHAT IS FAKE NEWS?

A type of yellow journalism, fake news encapsulates pieces of news that may be hoaxes and is generally spread through social media and other online media.

This is often done to further or impose certain ideas and is often achieved with political agendas.

Such news items may contain false and/or exaggerated claims, and may end up being viral by algorithms, and users may end up in a filter bubble.



OBJECTIVE:

- To detect the level of fakeness of a news
- To test, train and validate data
- To build a model to accurately classify a piece of news as TRUE or FALSE.
- Identify features (words, entities, phrases) in news which are fraudulent in nature.
- Use various natural language processing techniques and ML-algorithms to classify fake news articles using sci-kit libraries from python.
- Using sklearn, we will build a TfidfVectorizer on our dataset.
- Initialize a Passive Aggressive Classifier and fit the model and at the end, the accuracy score and the confusion matrix will tell us how well our model fares.

HOW THE DATA IS CREATED?

- Here we have a LIAR dataset obtained from Google.
- Originally LIAR dataset contains 3 files in .tsv format for test, train and validation.
- The data set is then converted to .csv format and only the necessary columns and attributes are used
- We did some research and have obtained this as a result :

LIAR : A BENCHMARK DATASET FOR FAKE NEWS DETECTION

William Yang Wang, "Liar, Liar Pants on Fire“ : A New Benchmark Dataset for Fake News Detection, to appear in Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (ACL 2017), short paper, Vancouver, BC, Canada, July 30-August 4, ACL.

HOW THE DATA IS CREATED: DESCRIPTION

The original dataset contained 13 variables/columns for train, test and validation sets as follows
:

- Column 1: the ID of the statement ([ID].json).
- Column 2: the label. (Label class contains: True, Mostly-true, Half-true, Barely-true, FALSE, Pants-fire)
- Column 3: the statement.
- Column 4: the subject(s).
- Column 5: the speaker.
- Column 6: the speaker's job title.
- Column 7: the state info.
- Column 8: the party affiliation.
- Column 9-13: the total credit history count, including the current statement.
- 9: barely true counts.
- 10: false counts.
- 11: half true counts.
- 12: mostly true counts.
- 13: pants on fire counts.
- Column 14: the context (venue / location of the speech or statement).

HOW THE DATA IS CREATED: DESCRIPTION (CONTINUED)

To make things simple we have chosen only 2 variables from this original dataset for this classification. The other variables can be added later to add some more complexity and enhance the features.

Below are the columns used to create 3 datasets that have been in used in this project

- Column 1: Statement (News headline or text).
- Column 2: Label (Label class contains: True, False

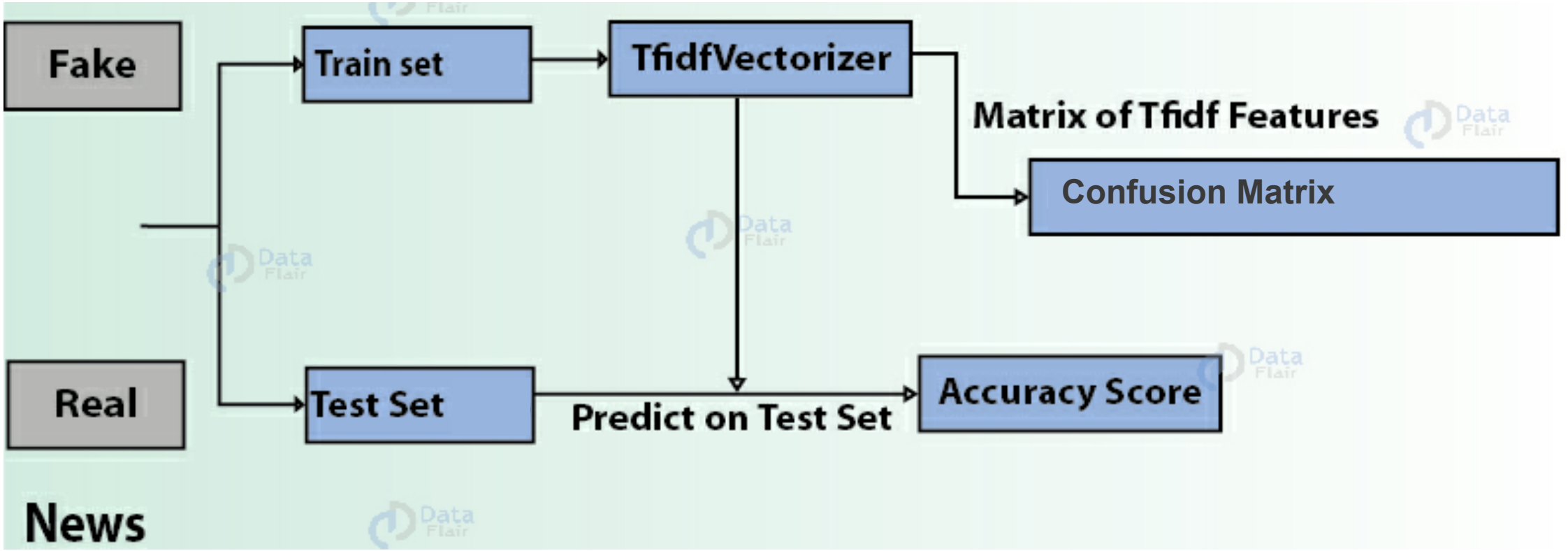
HOW THE DATA IS CREATED: DESCRIPTION (CONTINUED)

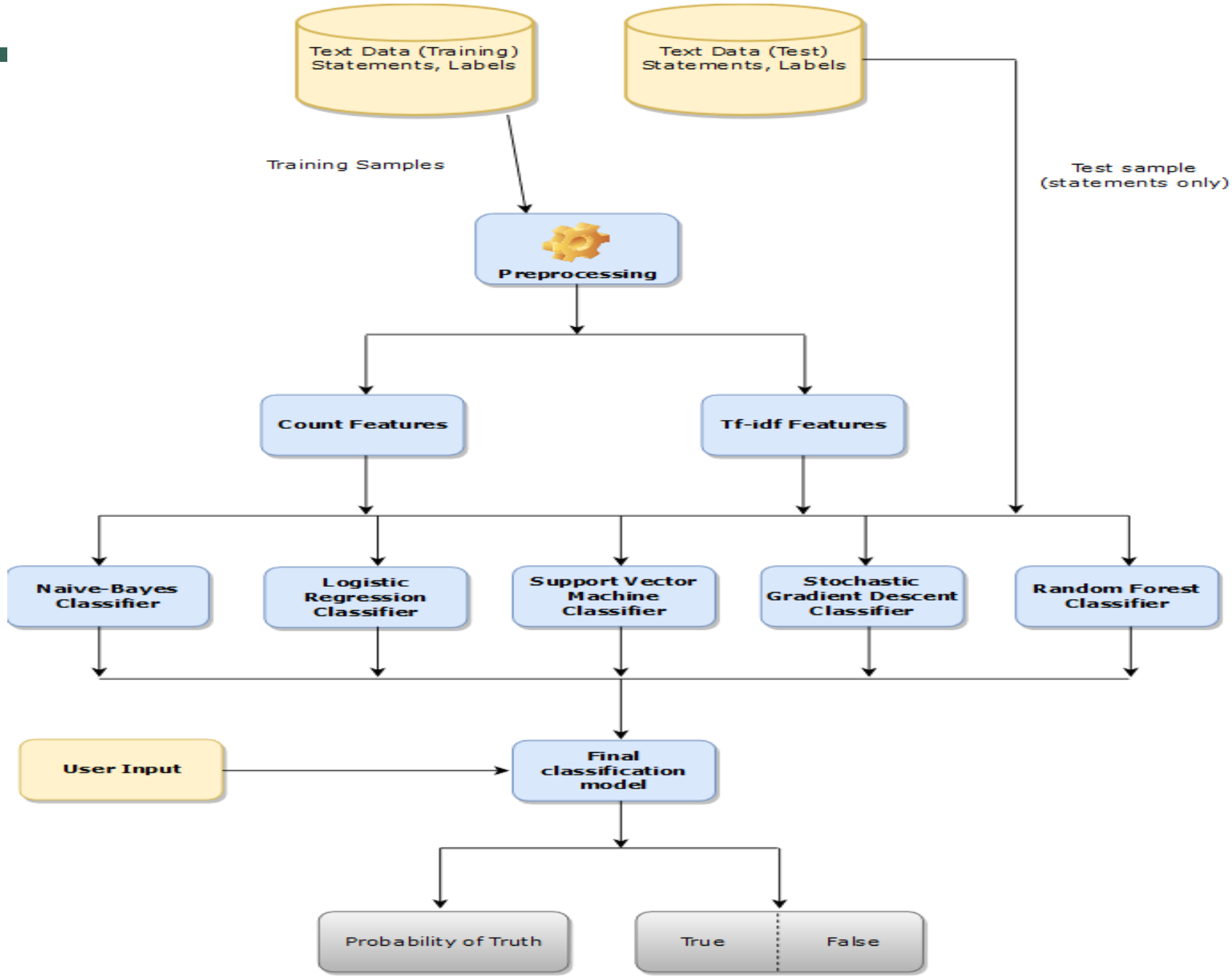
You will see that newly created dataset has only 2 classes as compared to 6 from original classes. Below is method used for reducing the number of classes.

- Original -- New
- True -- True
- Mostly-true -- True
- Half-true -- True
- Barely-true -- False
- False -- False
- Pants-fire -- False

The dataset used for this project were in csv format named train.csv, test.csv and valid.csv(Yo). The original datasets are in "liar" folder in .tsv format.

PROCESS FLOW OF PROJECT:




Text Data (Training)
Statements, Labels
Text Data (Test)
Statements, Labels
Training Samples
Test sample
(statements only)
Preprocessing
Count Features
Tf-idf Features
Naive-Bayes Classifier
Logistic Regression Classifier
Support Vector Machine Classifier
Stochastic Gradient Descent Classifier
Random Forest Classifier
User Input
Final classification model
Probability of Truth
True
False

CONCEPTS:

What is a TfidfVectorizer?

- TF (Term Frequency): The number of times a word appears in a document is its Term Frequency. A higher value means a term appears more often than others, and so, the document is a good match when the term is part of the search terms.
- IDF (Inverse Document Frequency): Words that occur many times a document, but also occur many times in many others, may be irrelevant. IDF is a measure of how significant a term is in the entire corpus.

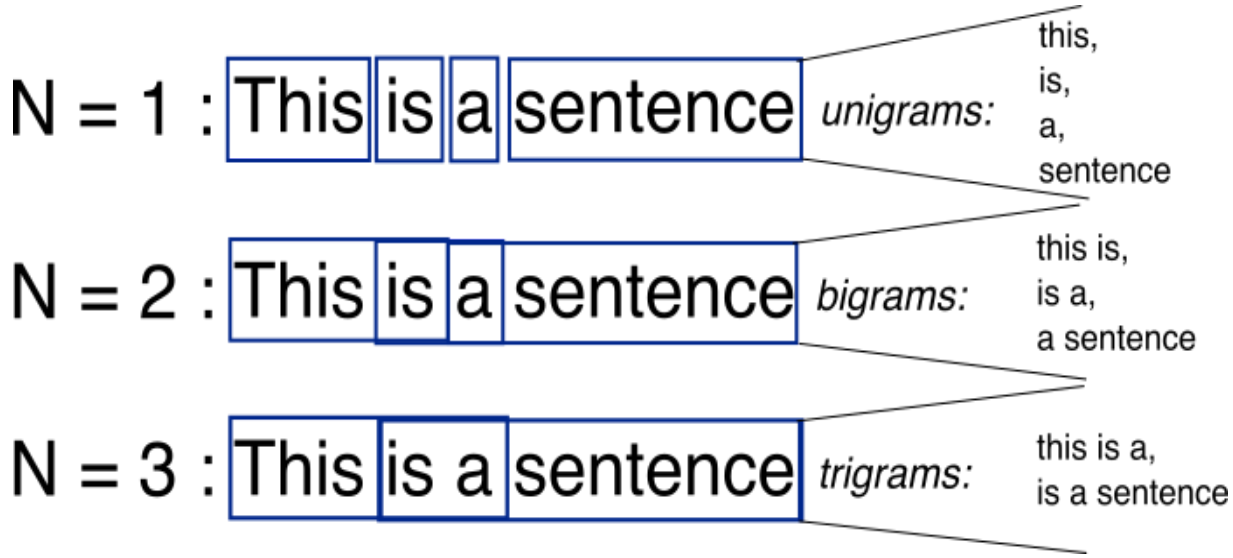
Stemming: Stemming is the process of producing morphological variants of a root/base word. Stemming programs are commonly referred to as stemming algorithms or stemmers. A stemming algorithm reduces the words eg. “chocolates”, “chocolatey”, “choco” to the root word

CONCEPTS: (CONTINUED)

- **Tokenization**

Tokenization is the act of breaking up a sequence of strings into pieces such as words, keywords, phrases, symbols and other elements called tokens. Tokens can be individual words, phrases or even whole sentences. In the process of tokenization, some characters like punctuation marks are discarded. The tokens become the input for another process like parsing and text mining

- **ngrams, unigrams, bigrams**

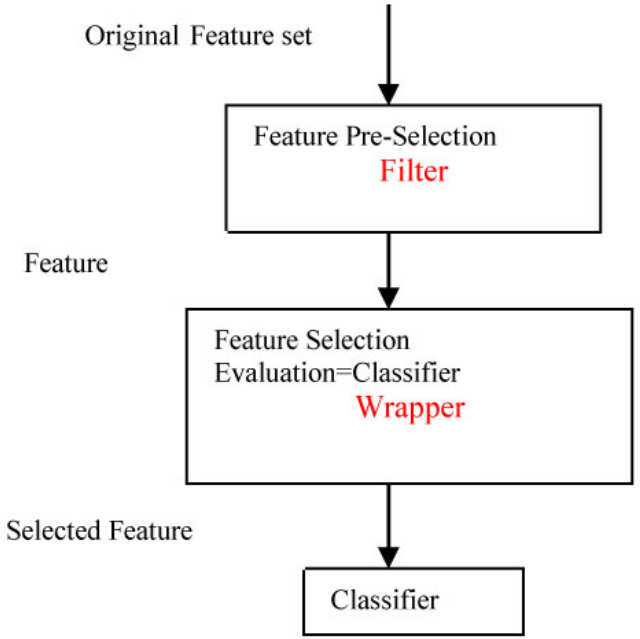


CONCEPTS: (CONTINUED)

- **Feature Selection**

Feature Selection is the process where you automatically or manually select those features which contribute most to your prediction variable or output in which you are interested in.

Having irrelevant features in your data can decrease the accuracy of the models and make your model learn based on irrelevant features.



- **Bag of words**

The bag-of-words (BOW) model is a representation that turns arbitrary text into fixed-length vectors by counting how many times each word appears. This process is often referred to as vectorization.

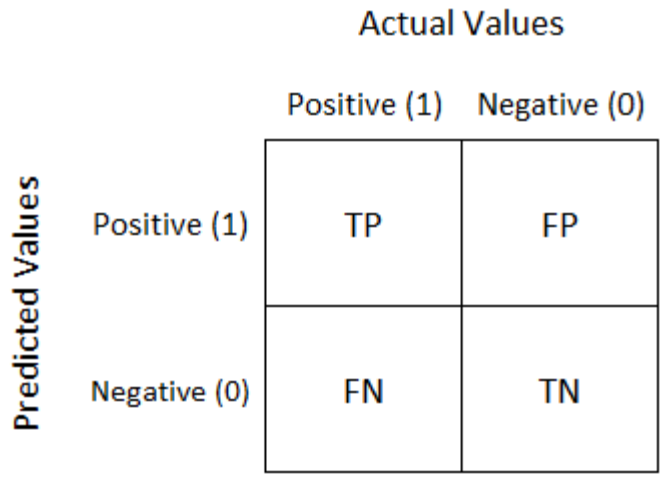
CONCEPTS: (CONTINUED)

- **POS tagging**

The process of classifying words into their parts of speech and labeling them accordingly is known as part-of-speech tagging or POS-tagging, or simply tagging.

- **Confusion Matrix**

It is a performance measurement for machine learning classification problem where output can be two or more classes. It is a table with 4 different combinations of predicted and actual values. It is extremely useful for measuring Recall, Precision, Specificity, Accuracy and most importantly AUC-ROC Curve.



STEPS PERFORMED FOR PRE-PROCESSING OF DATA:

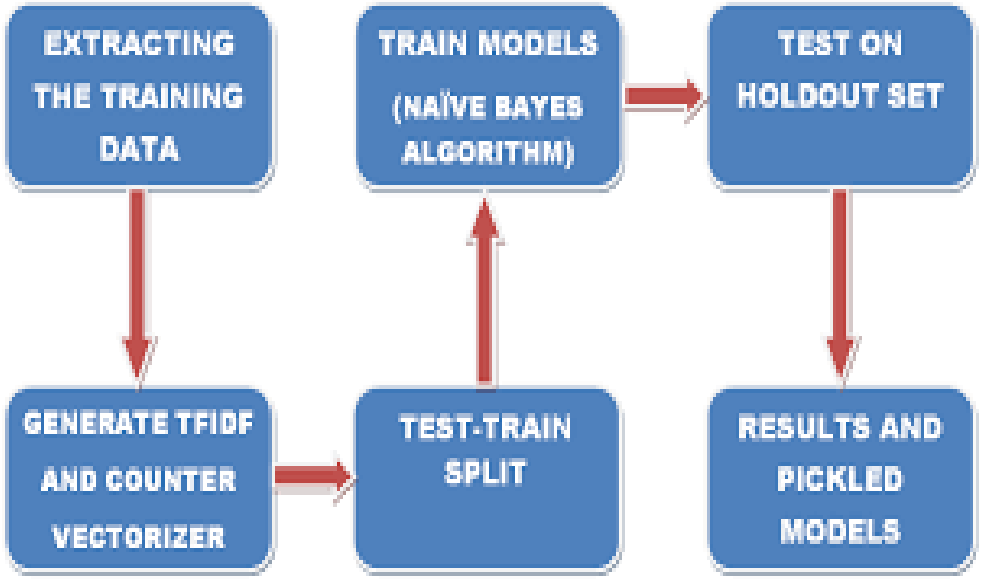
Our data has been pre-processed in Dataprep.py

1. Importing all the necessary libraries
2. Read all the data from the dataset files
3. Observation of data
4. Distribution of classes for prediction
5. Check the integrity of data (missing value labels)
6. Check training, testing and validation of data to be fairly or evenly distributed between the classes or not
7. Perform Stemming
8. Process the data
9. Creation of ngrams, unigrams, bigrams
10. Tokenization of Data

CLASSIFIERS: TRAINING MODEL

- In this Project we have used 5 types of model
- To scrutinize the data and come down to a solution
- The 5 models are :
  1. Naïve - bayes
  2. Logistic Regression
  3. Linear Support Vector Machine
  4. Stochastic gradient decent
  5. Random forest classifiers

Note: Not all models have been executed, but coded.



CLASSIFIERS: TRAINING MODEL (CONTINUED)

- **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. The fundamental Naive Bayes assumption is that each feature makes an: Independent and equal contribute to the outcome.

- **Stochastic gradient decent**

in Stochastic Gradient Descent, a few samples are selected randomly instead of the whole data set for each iteration. In Gradient Descent, there is a term called "batch" which denotes the total number of samples from a dataset that is used for calculating the gradient for each iteration. In typical Gradient Descent optimization, like Batch Gradient Descent, the batch is taken to be the whole dataset.

- **Linear support vector machine (SVM)**

A support vector machine (SVM) is a supervised machine learning model that uses classification algorithms for two-group classification problems. After giving an SVM model sets of labeled training data for each category, they're able to categorize new text.

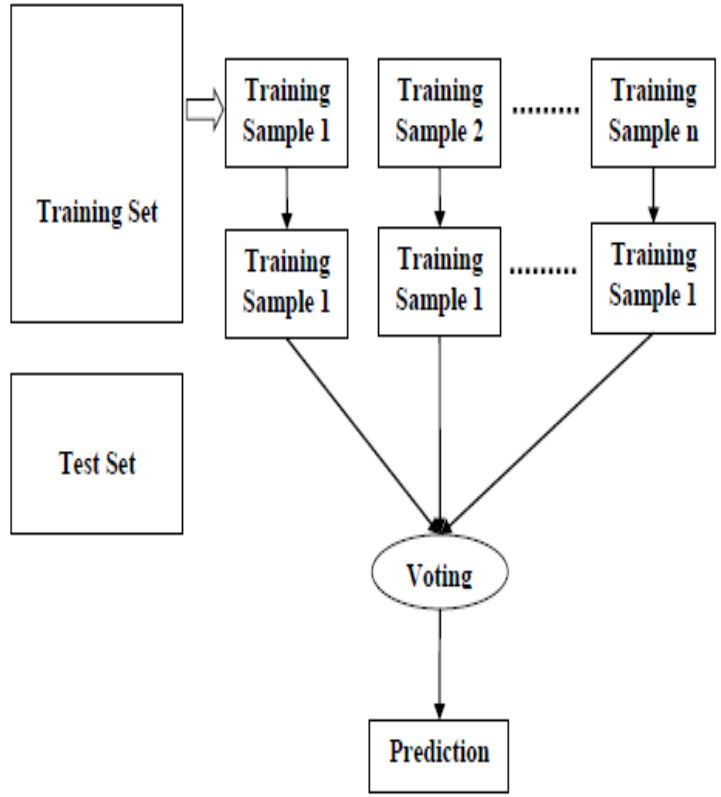
CLASSIFIERS: TRAINING MODEL (CONTINUED)

- **Random forest classifiers**

It is an ensemble tree-based learning algorithm. The Random Forest Classifier is a set of decision trees from randomly selected subset of training set.
It aggregates the votes from different decision trees to decide the final class of the test object. We could have used this model as our second preference because

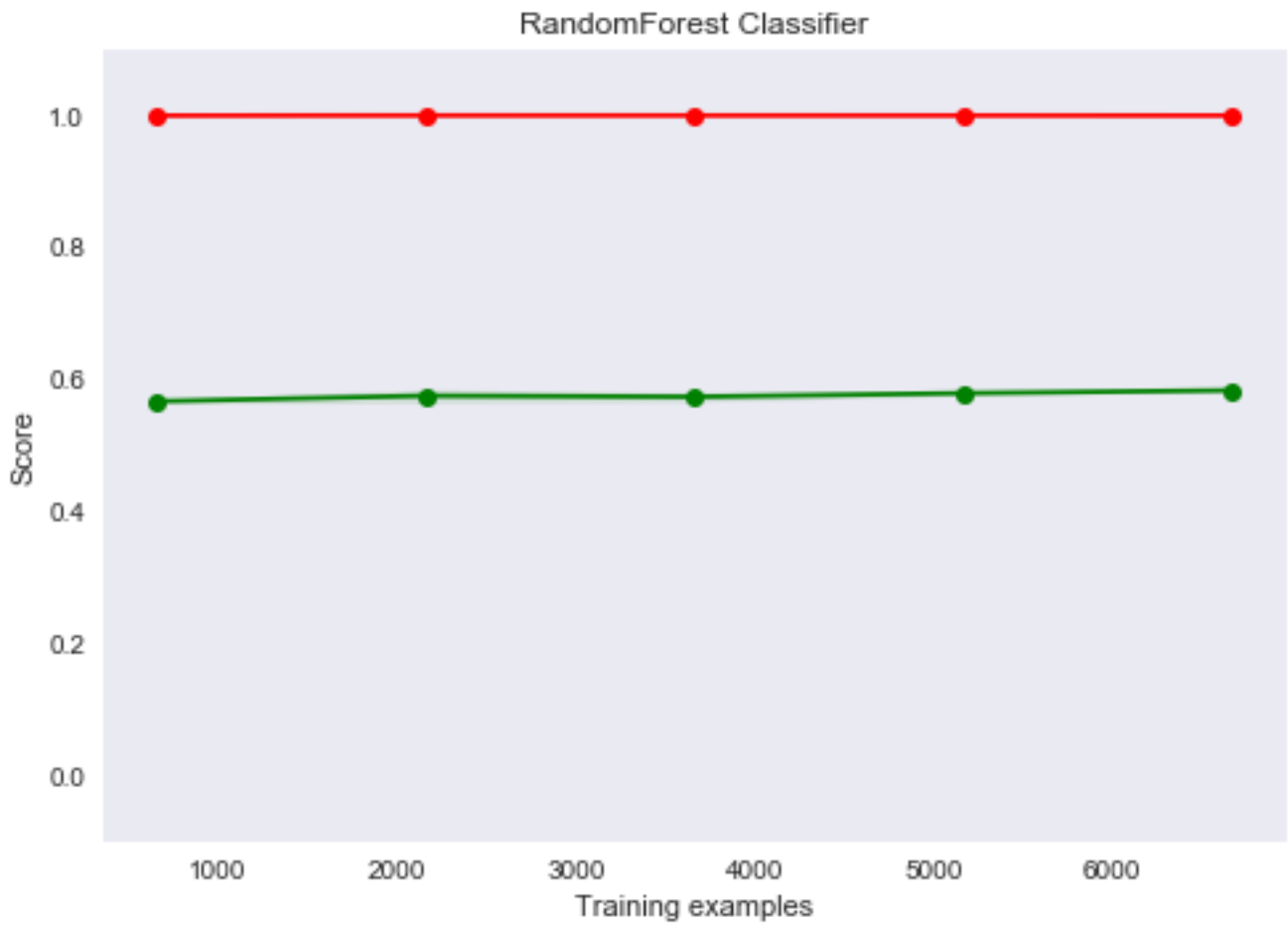
- Random Forest can be used to solve both classification as well as regression problems
- Random Forest works well with both categorical and continuous variables. (our data is categorical)
- It can automatically handle missing values.

But we discard this classifier because it needs longer training period - Random Forest require much more time to train as compared to decision trees as it generates a lot of trees and makes decision on the majority of votes which will increase complexity and reduce accuracy



CLASSIFIERS: TRAINING MODEL
RANDOM FOREST CLASSIFIERS (CONTINUED)

- Anyway we implemented the model and this is how we did it and got a result
- #code
- As you can see we cannot get a distinct variation
- Note
- Red False
- Green True



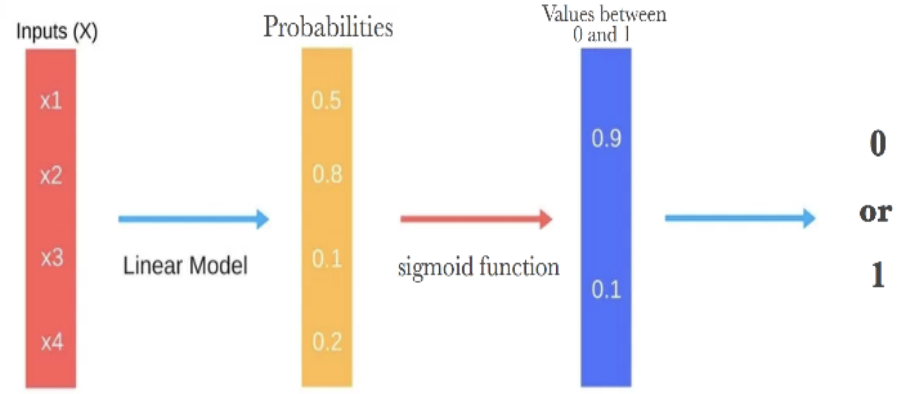
CLASSIFIERS: TRAINING MODEL (CONTINUED)

- **Logistic Regression (Best Classifier for the project)**

Logistic regression is a statistical model that in its basic form uses a logistic function to model a binary dependent variable, although many more complex extensions exist. In regression analysis, logistic regression (or logit regression) is estimating the parameters of a logistic model (a form of binary regression)

We use this model for prediction because:

- Dataset is linearly separable. (True or False)
- Gives a measure of how relevant a predictor (coefficient size) is.
- Gives direction of association of predictor (positive or negative/bool).
- Efficient to train



CLASSIFIERS: TRAINING MODEL LOGISTIC REGRESSION (CONTINUED)

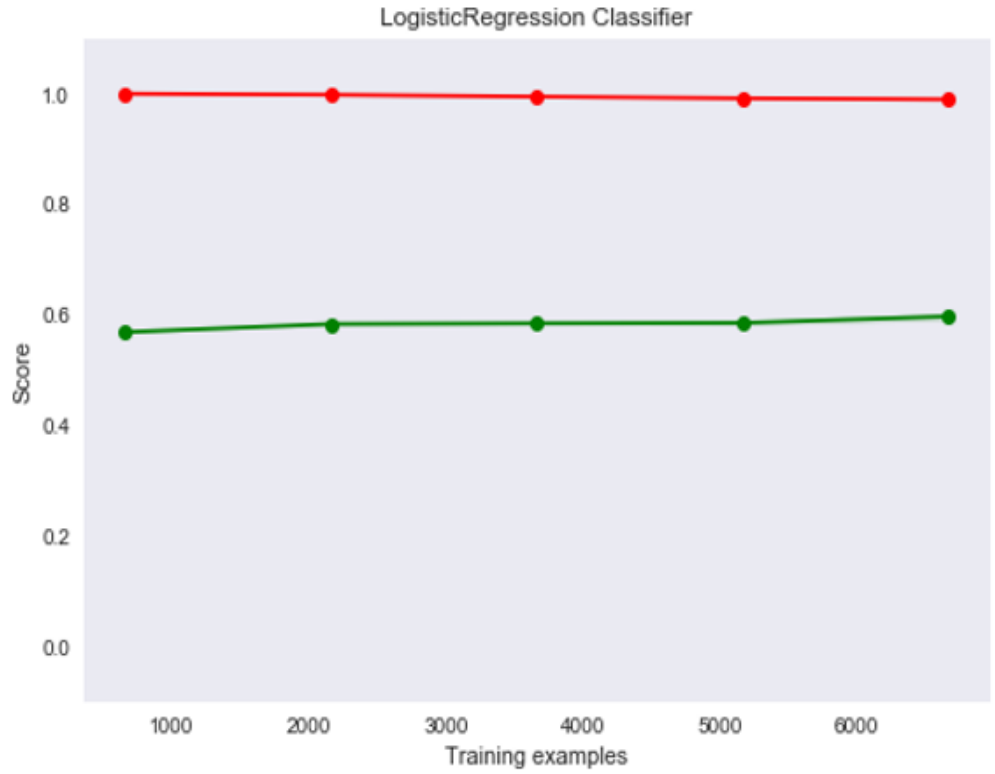
- This is how we obtained the curve and the result:

- #codesnap

Note:

Red-False

Green-True



CODE:

Our code is in 4 files:

1. Data Pre-processing – Dataprep.py
2. Feature Selection – FeatureSelection.py
3. Data Modeling – Classifier.py
4. Prediction (Final file) – prediction.py

CODE: (CONTINUED)

- Dataprep.py

This file contains all the pre processing functions needed to process all input documents and texts. First we read the train, test and validation data files then performed some pre processing like tokenizing, stemming etc. There are some exploratory data analysis is performed like response variable distribution and data quality checks like null or missing values etc.

- FeatureSelection.py

In this file we have performed feature extraction and selection methods from sci-kit learn python libraries. For feature selection, we have used methods like simple bag-of-words and n-grams and then term frequency like tf-tdf weighting. We have also used word2vec and POS tagging to extract the features.

CODE: (CONTINUED)

- Classifier.py

Build all the classifiers for predicting the fake news. The extracted features are fed into different classifiers. We have used Naive-bayes, Logistic Regression, Linear SVM, Stochastic gradient decent & Random forest classifiers from sklearn. Once fitting the model, we compared the f1 score & checked the confusion matrix. After fitting all the classifiers, 2 best performing models were selected as candidate models for fake news classification. We have performed parameter tuning by implementing GridSearchCV methods on these candidate models & chosen best performing parameters for these classifier. Finally selected model was used for fake news detection with the probability of truth. We have also extracted the top 50 features from our term-frequency tfidfvectorizer to see what words are most and important in each of the classes. We have used Precision-Recall & learning curves to see how training & test set performs when we increase the amount of data in our classifiers.

- Prediction.py

Our finally selected and best performing classifier was Logistic Regression which was then saved on disk with name final model.sav. Once you close this repository, this model will be copied to user's machine and will be used by prediction.py file to classify the fake news. It takes an news article as input from user then model is used for final classification output that is shown to user along with probability of truth.

OUTPUT: