Maulana Azad National Institute Of Technology, Bhopal

Department of Computer Science & Engineering

Detection Of Fake Tweets Related To Covid-19

Classifies tweets/news as fake or real

Mentor
Prof. Pragati Agrawal
Submitted By
Shruti Yadav 171112208

November 11, 2020

Contents

Maulana Azad National Institute Of Technology Bhopal, India, 462003

Department of Computer Science & Engineering

Certificate

This is to certify that the project report on **"Detection Of Fake tweets Related To Covid-19"** prepared by :

Shruti Yadav 171112208

has successfully completed her internship project in partial fulfillment of her degree in Bachelor of Technology in Computer Science and Engineering.

[illegible]

Prof. Pragati Agrawal
(Internship Project Mentor)

Chapter 1

Abstract

Technology has always been affecting the human race in every possible way. Social media platforms are one of the products of technology. Twitter is one of the biggest and one of the most widely used social media platform. The news on twitter and tweets by different people easily become viral and trending.

Fake news and fake tweets have become a matter of great concern in today's world. The rate at which fake tweets travel from one person to another has become a boost to spread of misinformation. Fake news ignites the confusion among people and can cause panic. Amidst this pandemic caused by coronavirus, fake tweets related to Covid'19 has become a real issue with the technology. People come in fear due to the misguidance by these fake tweets/news. For common people, it is difficult to easily distinguish between real and fake tweets. Thus, a model is required which can detect the fake news t

It is noticed that fake tweets follow some pattern which can be studied by visualizing the data sets of real and fake tweets. Machine Learning can come to rescue people from the mushrooming fake tweets by exploiting those patterns. These data sets can be used to train the machine learning classifier models and the trained model then can be used to segregate the real and fake tweets for people.

This project aims to develop a model which can segregate the real news and the fake news.The dataset collected consists of both real and fake tweets. Multiple classification algorithms- multinomial naive bayes, logistic regression, XG boost, passive aggressive classifier, stochastic gradient descent are implemented and evaluated on test data. The Stochastic gradient descent algorithm has shown the best results among the other tested algorithms with 84% accuracy .

Chapter 2

Software Engineering Paradigm

The developed model is based on data driven and procedural software engineering paradigm.

It is a executable code which serves the objective of the project. The model is associated with libraries, modules and various machine learning algorithms.

The project is based upon following three pillars of software engineering:

1. Requirement gathering- All the data required to train the model is collected, and tools or modules required for model implementation are imported.
2. Designing- Algorithms of machine learning related to classification problems and overall structure of the project is designed.
3. Programming- All the data, tools, modules, algorithms, are out together in a code to implement the model and train it on the data.

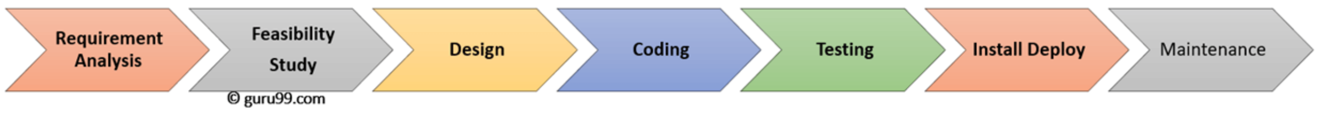

Figure 2.1: SDLC CYCLE

Chapter 3

Architecture

The architecture of the model consists of various layer and activities:

1. Data collection
2. Data pre-processing
3. Data transformation
4. Train/Test split
5. Training the model
6. Testing the model
7. Hyper tuning the parameters
8. Results analysis and visualization

The model will work as follows:
Any new tweet with the hashtags related to Covid-19 will be extracted by the model and that tweet will be passed through the trained model. The model through its intelligence will predict whether the tweet is real or fake. if the tweet is fake, it will report to the authority and evade the fake tweet, and if it is real tweet, then the tweet will be not be evaded. This will prevent the users of the twitter from the fake tweets and misinformation.

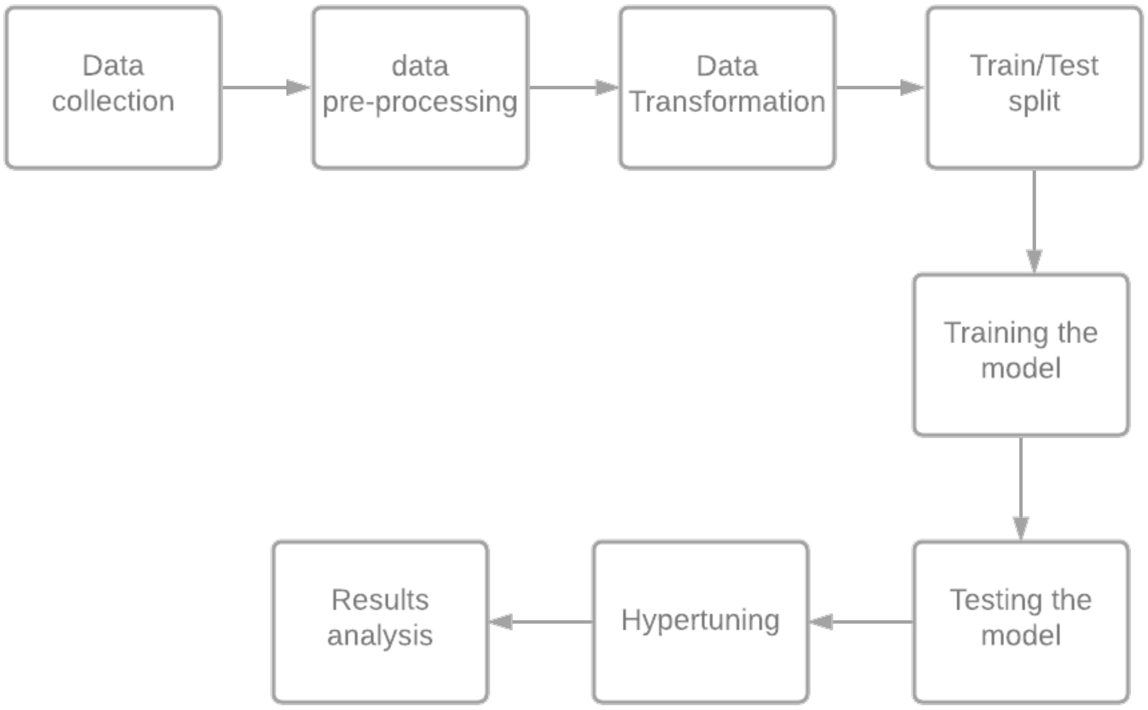

Figure 3.1: Activity diagram

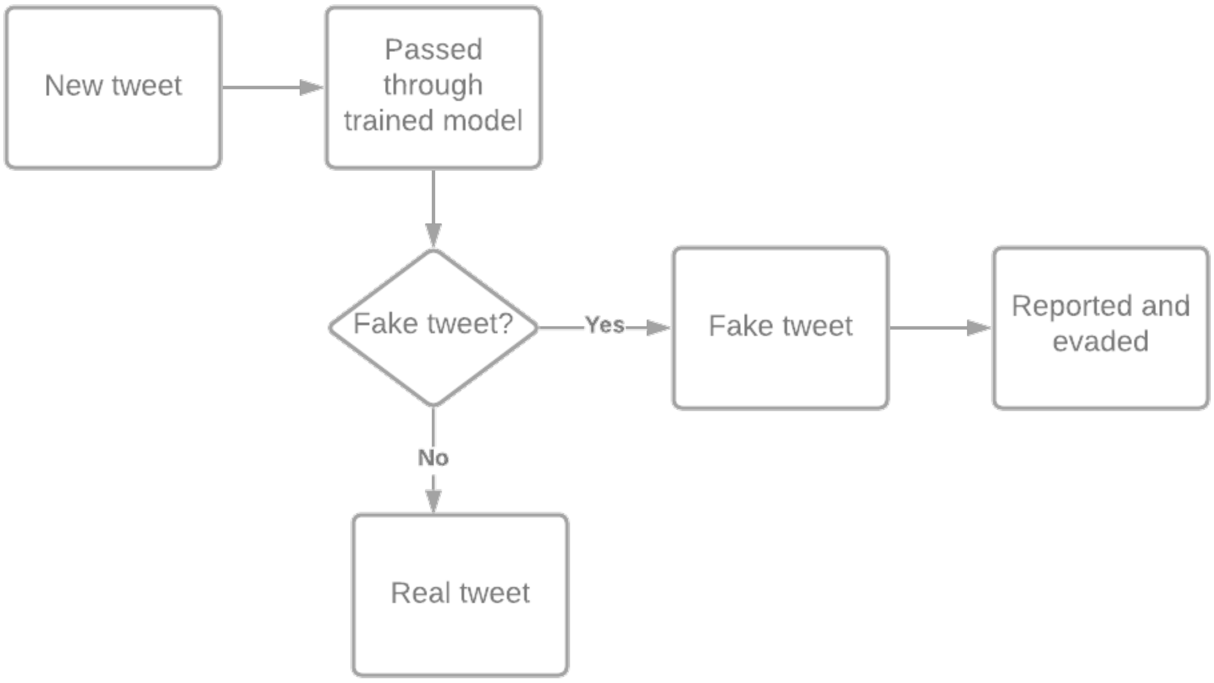

Figure 3.2: Working of the model

Chapter 4

Project Planning and Project Scheduling

The objective of the project is to develop a model which can classify fake and real tweets related to Covid-19. The different tasks performed are:

4.1 Data Collection

Training and test data for the model is collected from data set available on kaggle competition-Real or not?NLP with Disaster Tweets..

The training data set consists both real and fake tweets,along with the binary variable which depicts whether the tweet is fake or not. The testing data is extracted from the twitter api and it consists non-classified tweets.

4.2 Data Preprocessing

The tweets gathered are pre-processed to make them suitable for further modelling and analysis. Following steps are taken to clean the dataset:

1. Removal of contractions: The words like aren't, isn't are called contractions. They do not contribute in anyway to increase the efficiency, hence can be removed easily.
2. Removal of hashtags: The objective of the project is to detect the fake tweets related to covid'19. Therefore, The hashtags related to covid'19 are only useful in data collection and should be removed before furthur processing.
3. Removal of url's and hyperlinks: The url's and hyperlinks should be removed in the pre-processing part.
4. Removal of whitespaces: Removal of whitespaces like tabs and empty strings is the crucial part of text preprocessing.
5. Removal of words with 4 or less letters: Words with few characters are generally not so meaningful and can interfere in proper and efficient text analysis.

6. Removal of characters beyond basic multilingual plane of unicode: These characters are also not required in training or analyzing the text, hence can be removed.
7. Removal of mis-spelt word: Misspelt words can divert our actual results to false results, and therefore should be removed.
8. Removal of emojis: Emojis and emoticons are generally found with every tweet but they are least useful in predicting our goal.

4.3 Data Transformation

Data Transformation is the process of converting data from one form to another to make it more suitable for feeding into the model and further processing. Train-test split (with test-size="0.2") is used to split the training data set into training and test data set to train the model. Count vectorizer is used to transform the documents to document-term matrix or to a matrix of Tf-Idf features.

4.4 Classification Models

Classification models which have been implemented are:

1. stochastic gradient descent
2. passive aggressive classifier
3. XG boost
4. Logistic Regression

The comparative analysis has been performed and the best algorithm is chosen for final model. Stochastic gradient descent gave the best results with the accuracy of 84%.

4.5 Hyper-parameter tuning

In the context of machine learning, hyper-parameters are the parameters of the machine learning algorithm that influence how the model fits with the training data, and hence the test data predictions. For eg, incorrectly defined hyper-parameters will lead to the model over fitting on the training data and improperly predicting the test data when the model is put into development. We care about this because, on the other hand, the optimally defined hyper-parameters will significantly increase the output of the model.[1]

Chapter 5

Tools and Technology Used

Various machine learning classification algorithms have been used in the project

5.1 Gradient Descent Algorithm

Gradient Descent is a popular optimization technique in artificial intelligence and deep learning that can be used with many other learning algorithms, although not all most. A gradient is the slope of a function. In response to some other input variable, it measures the degree of variation of a parameter. Mathematically, the Gradient Descent is a convex function whose output is a partial derivative of a set of its input parameters. The steeper the slope, the gradient becomes higher.

Beginning with the initial value, Gradient Descent is executed iteratively to find out the optimum value of the parameters in order to find the minimum possible value of the cost function..

The word 'stochastic' refers to the mechanism or process associated with a random likelihood. Thus, a few features are collected randomly in Stochastic Gradient Descent rather than the full collection of data in each iteration. The term "batch" is also used in Gradient Descent to denote the total sample size from the dataset used for each iteration to calculate the gradient.The batch is used to be the entire dataset in traditional Gradient Descent optimization, such as Batch Gradient Descent. While it has been very beneficial to use the dataset as a whole to get to the minimum in a much less noisy and less chaotic way, as our datasets grow big, the issue arises.

If you use a standard Gradient Descent optimization process, you will have to use all one million samples to complete each iteration when executing Gradient Descent, assuming that we use a million samples in our dataset, and you need to repeat them for each step until the minimum is achieved. As a consequence, it is very expensive in terms of computational complexity.

In stochastic (or "on-line") gradient descent, the true gradient of $Q(w)$

is approximated by a gradient at a single example:

$w := w - \eta \nabla Q_i(w).w := w - \eta \nabla Q_i(w).$

As the algorithm runs through the testing dataset, for each training case, the aforementioned change is made. Over the training set before the algorithm converges, a number of iterations can also be made. The data can be shuffled to prevent loops with each step if this is accomplished. To allow the algorithm to converge, typical implementations will use an adaptive learning rate.

In a single example, a solution found between calculating the true gradient and the gradient is to measure the gradient at each point against more than one training instance also known as "mini-batch". It can function considerably better than the "true" stochastic gradient descent described, as the programme can be used by vector libraries rather than by separately evaluating each step.[2]

5.2 Passive aggressive classifier

Passive-Aggressive algorithms are commonly used for learning on a wide scale. It is one of the few 'algorithms for online-learning.' The input information arrives in sequential order in online machine learning algorithms and the machine learning model is modified step-by - step, as opposed to batch learning, where the whole testing dataset is used at once.This is very effective in cases where there is a large volume of data and because of the sheer scale of the data, it is computationally infeasible to train the whole dataset. We may simply say that a training example would be given by an online-learning algorithm, update the classifier, and then throw away the example.

Fake news on a social media platform like Twitter, where fresh data is added every second, will be a really clear example of this. In order to constantly read Twitter data dynamically, the data would be massive, and it would be optimal to use an online learning algorithm.

Passive-aggressive algorithms, in the sense that they do not need a learning rate, are somewhat similar to the Perceptron model. They do, however, have a parameter for regularisation.[3]

5.3 Logistic Regression

One of the most popular Machine Learning algorithms, which falls under the Supervised Learning method, is logistic regression. It is used to use a given set of independent variables to predict a categorical dependent variable.

By logistic regression, the output of a categorical dependent variable is calculated. A categorical or discrete attribute must, thus, be the product. It can be either Yes or No, 0 or 1, true or False, etc., but it offers probabilistic values between 0 and 1 instead of giving the same value as 0 and 1,.

With the exception to how they are used, logistic regression is somewhat similar to linear regression. For solving regression problems, linear regression is used, while logistic regression is used for solving classification problems.

Instead of fitting a regression line, we fit a 'S' form logistic function in Logistic Regression, which predicts two maximum values (0 or 1). The logistic function curve shows the

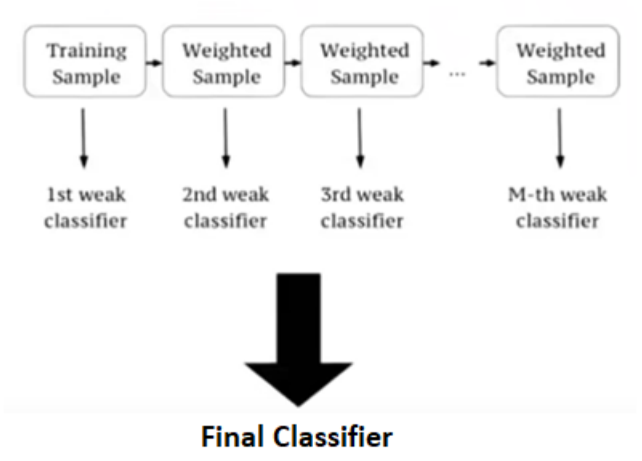

Figure 5.1: XgBoost model

possibility of anything, such as whether the cells are cancerous or not, whether a mouse is fat or not depending on its weight, etc.

Logistic Regression is an effective machine learning algorithm since, using continuous and discrete datasets, it has the potential to provide probabilities and identify new data.

Using various data types, logistic regression can be used to characterise the observations and can quickly evaluate the most powerful variables used for classification.[4]

5.4 Xg Boost

XGBoost is an implementation of decision trees with Gradient Enhanced. In C++, this library was written. It is a kind of software library that was fundamentally developed to boost the efficiency of the model and time. Recently , advanced machine learning has become dominant. In several Kaggle tournaments, XGBoost models largely dominate.

Decision trees are generated in sequential form within this algorithm. In XGBoost, weights play an important part. All the independent variables which are then fed into the decision tree that forecasts outcomes are given weights. The weight of variables that the tree forecasts to be incorrect is raised and the variables are then fed to the second decision tree.To give a powerful and more detailed model, these individual classifiers / predictors then join together. Regression, grouping, rating, and user-defined prediction issues may be solved.(refer to 5.1)[5]

Chapter 6

Modules and description

Various modules in python used in the project implementation are:

6.1 Parfit

Through parfit, users can conduct gird search on model.

The advantages of the parfit module are as follows:

1. Validate: Specifies the validation set on which scoring is too be done flexibly.
2. Score: Scoring metric can be chosen efficiently.
3. Visualize: Score over grid of hyper parameters entered are optimally plotted.
4. Optimize: The best model, corresponding hyper-parameters, and score are returned optimally.
5. Easy to use: All this work can be done easily through single function call.

6.2 numpy

Numpy is a package for general purpose array-processing. It offers a multidimensional array object with high performance and tools for working with these arrays. It is the basic package for Python scientific computing.[6]

6.3 pandas

Pandas is an open-source, BSD-licensed Python library which provides the Python programming language with high-performance, easy-to-use data structures and data analysis tools.[7]

6.4 confusion matrix

The Confusion Matrix is one of the simplest and most intuitive metrics used to determine the accuracy of a classification model in which two or more categories of output can be produced. This is the most popular method used for logistic regression evaluation.[8]

6.5 accuracy score

Accuracy scores refer to the accuracy of which the expected labels in the dataset complement the actual labels. This is similar to the jaccard_score function in multiclass and binary classification.[9]

Chapter 7

Source code

Below is the source code of the stochastic gradient descent algorithm implementation.

```
# -*- coding: utf-8 -*-
"""
Created on Tue Jun 16 11:50:20 2020

@author: shruti yadav
"""

import numpy as np
import pandas as pd
import itertools
from sklearn.model_selection import train_test_split
from sklearn.feature_extraction.text import TfidfVectorizer
from sklearn.linear_model import PassiveAggressiveClassifier
from sklearn.metrics import accuracy_score, confusion_matrix
import emoji
import re
df=pd.read_csv('train_covid.csv',encoding='ISO-8859-1')
#Get shape and head
print(df.shape)
print(df.head())
labels=df.target
print(labels.head())
fake_tweets = df[df.target == 0]
print(fake_tweets.shape)
print(fake_tweets.head(300))
contractions = {
"ain't": "am not",
"aren't": "are not",
"can't": "cannot",
"can't've": "cannot have",
"'cause": "because",
"could've": "could have",
"couldn't": "could not",
"couldn't've": "could not have",
"didn't": "did not",
"doesn't": "does not",
"don't": "do not",
"hadn't": "had not",
"hadn't've": "had not have",
"hasn't": "has not",
"haven't": "have not",
"he'd": "he would",
"he'd've": "he would have",
"he'll": "he will",
"he's": "he is",
"how'd": "how did",
"how'll": "how will",
"how's": "how is",
"i'd": "i would",
"i'll": "i will",
"i'm": "i am",
"i've": "i have",
"isn't": "is not",
"it'd": "it would",
"it'll": "it will",
"it's": "it is",
"let's": "let us",
"ma'am": "madam",
"mayn't": "may not",
"might've": "might have",
"mightn't": "might not",
"must've": "must have",
"mustn't": "must not",
"needn't": "need not",
"oughtn't": "ought not",
"shan't": "shall not",
"sha'n't": "shall not",
"she'd": "she would",
"she'll": "she will",
"she's": "she is",
"should've": "should have",
"shouldn't": "should not",
"that'd": "that would",
"that's": "that is",
"there'd": "there had",
"there's": "there is",
"they'd": "they would",
"they'll": "they will",
"they're": "they are",
"they've": "they have",
"wasn't": "was not",
"we'd": "we would",
"we'll": "we will",
"we're": "we are",
"we've": "we have",
"weren't": "were not",
"what'll": "what will",
"what're": "what are",
"what's": "what is",
"what've": "what have",
"where'd": "where did",
"where's": "where is",
"who'll": "who will",
"who's": "who is",
"won't": "will not",
"wouldn't": "would not",
"you'd": "you would",
"you'll": "you will",
"you're": "you are",
"thx"    : "thanks"
}
def remove_contractions(text):
    return contractions[text.lower()] if text.lower() in
       contractions.keys() else text
df['text']=df['text'].apply(remove_contractions)
#print(df.head())

def clean_dataset(text):
    # Remove hashtag while keeping hashtag text
    text = re.sub(r'#','', text)
    # Remove HTML special entities (e.g. &amp;)
    text = re.sub(r'\&\w*;', '', text)
    # Remove tickers
    text = re.sub(r'\$\w*', '', text)
    # Remove hyperlinks
    text = re.sub(r'https?:\/\/.*\/\w*', '', text)
    # Remove whitespace (including new line characters)
    text = re.sub(r'\s\s+','', text)
    text = re.sub(r'[ ]{2, }',' ',text)
    # Remove URL, RT, mention(@)
    text=  re.sub(r'http(\S)+', '',text)
    text=  re.sub(r'http ...', '',text)
    text=  re.sub(r'(RT|rt)[ ]*@[ ]*[\S]+','',text)
    text=  re.sub(r'RT[ ]?@','',text)
    text = re.sub(r'@[\S]+','',text)
    # Remove words with 4 or fewer letters
    text = re.sub(r'\b\w{1,4}\b', '', text)
    #&, < and >
    text = re.sub(r'&amp;?', 'and',text)
    text = re.sub(r'&lt;','<',text)
    text = re.sub(r'&gt;','>',text)
    # Remove characters beyond Basic Multilingual Plane (BMP) of Unicode:
    text= ''.join(c for c in text if c <= '\uFFFF')
    text = text.strip()
    # Remove misspelling words
    text = ''.join(''.join(s)[:2] for _, s in itertools.groupby(text))
    # Remove emoji
    text = emoji.demojize(text)
    text = text.replace(":"," ")
    text = ' '.join(text.split())
    text = re.sub("([^\x00-\x7F])+"," ",text)
    # Remove Mojibake (also extra spaces)
    text = '
       '.join(re.sub("[^\u4e00-\u9fa5\u0030-\u0039\u0041-\u005a\u0061-\u007a]",
       " ", text).split())
    return text

df['text'] =df['text'].apply(clean_dataset)
print(df.head())
print(df.shape)

x_train,x_test,y_train,y_test=train_test_split(df['text'], labels,
   test_size=0.2, random_state=7)
```

```
#DataFlair - Initialize a TfidfVectorizer
from sklearn.feature_extraction.text import CountVectorizer

vectorizer = CountVectorizer()
vectorizer.fit(x_train)

X_train = vectorizer.transform(x_train)
X_test  = vectorizer.transform(x_test)
from sklearn.naive_bayes import MultinomialNB
clf = MultinomialNB().fit(X_train, y_train)
predicted = clf.predict(X_test)
print(np.mean(predicted == y_test))
from sklearn.linear_model import SGDClassifier
clf = SGDClassifier().fit(X_train, y_train)
predicted = clf.predict(X_test)
print(np.mean(predicted == y_test))


from sklearn.model_selection import GridSearchCV
from sklearn.model_selection import ParameterGrid
import parfit.parfit as pf
from sklearn.metrics import roc_auc_score
param_grid = {
    'alpha': [1e-4, 1e-3, 1e-2, 1e-1, 1e0, 1e1, 1e2, 1e3], # learning rate

    'loss': ['log','hinge'], # logistic regression,
    'penalty': ['l2'],
    'n_jobs': [-1]
}
paramGrid = ParameterGrid(param_grid)

bestModel, bestScore, allModels, allScores = pf.bestFit(SGDClassifier,
   paramGrid,
           X_train, y_train, X_test, y_test,
           metric = roc_auc_score,
           scoreLabel = "AUC")

print(bestModel, bestScore)
clf = SGDClassifier(alpha=0.001, average=False, class_weight=None,
              early_stopping=False, epsilon=0.1, eta0=0.0,
                 fit_intercept=True,
              l1_ratio=0.15, learning_rate='optimal', loss='log',
                 max_iter=1000,
              n_iter_no_change=5, n_jobs=-1, penalty='l2', power_t=0.5,
              random_state=None, shuffle=True, tol=0.001,
              validation_fraction=0.1, verbose=0,
                 warm_start=False).fit(X_train, y_train)
data = {'text':['Last evening, I interacted with various NGO
   representatives on  #AatmanirbharBharat and role of NGOs, a unique
   initiative by MahaNGO Federation.Stressed on need to focus more on
   opportunities for India post #COVID19 pandemic.','After the bill is
   passed into law in Provincial Assembly Sindh - video link conference is
   introduced and is being demonstrated for the first time by Honourable
   Chief Minister and Worthy Speaker Sindh Assembly in picture below.
   #govtsindh         ']}
test_df=pd.DataFrame(data)
test_df['text']=test_df['text'].apply(remove_contractions)
```

```
## Clean Data set
test_df['text'] =test_df['text'].apply(clean_dataset)
X_test  = vectorizer.transform(test_df['text'])
y_pred=clf.predict(X_test)
print(y_pred)
```

Chapter 8

Results analysis and Application

By following the above mentioned methodology, the accuracy obtained for all the trained models are as follows:

1. Logistic regression: 78.06%
2. Passive aggressive classifier: 74%
3. Gradient Descent Algorithm: 84%
4. Xg Boost: 74.08%

The accuracy values obtained show that the gradient descent algorithm outperforms all other models and is the most efficient in detecting fake tweets. Thus, the final model selected for developing the application is gradient descent algorithm model.

The model developed is useful to curb the spread of misinformation through fake tweets in these difficult times of pandemic. The model will only allow the real tweets to be visible to the users and fake tweets will be reported and evaded.

The model can be used to efficiently detect fake tweets and preventing the users from becoming the victim of the misinformation. Fake tweets related to Covid-19 are very prone to be on trending, and thus can be harmful for the society. The application using this model can be used for common good in the society and to stop the chaos caused due to fake information.

Chapter 9

Conclusion

In this project, the objective was to develop an application which can utilize the machine learning model to detect the fake tweets related to Covid-19.

It was observed that fake tweets follow a common pattern and this feature of fake tweets has been exploited in the implementation of the project. The sequence of pattern was studied, analyzed and models which can work on such data were worked upon. Four classification models are used in the project to do the comparative analysis and find the best suited model for our objective.

Dataset was obtained and was used to train those models, Then test data which was extracted from twitter api was used to evaluate the performance of the models. Accuracy measures were analyzed and the best performing model, which is gradient descent, was chosen to be the most effective to serve the fake tweets detection objective.

The application of this developed model can be directly seen from the effect of fake tweets on the minds of people and the chaos it creates in the society. This model can detect and evade the fake tweets with hashtags related to Covid-19 and allow only the real tweets to be seen to the users.

References

[1] Jason Carpenter. *Parfit — quick and powerful hyper-parameter optimization with visualizations*. 2017. URL: https://medium.com/mlreview/parfit-hyper-parameter-optimization-77253e7e175e (page 7).

[2] *ML | Stochastic Gradient Descent (SGD)* (page 9).

[3] *Passive Aggressive Classifiers*. 2020. URL: https://www.geeksforgeeks.org/passive-aggressive-classifiers/ (page 9).

[4] *Logistic Regression in Machine Learning*. URL: https://www.javatpoint.com/logistic-regression-in-machine-learning (page 10).

[5] *ML | XGBoost (eXtreme Gradient Boosting)*. 2019. URL: https://www.geeksforgeeks.org/ml-xgboost-extreme-gradient-boosting/ (page 10).

[6] *Python Numpy*. URL: https://www.geeksforgeeks.org/python-numpy/ (page 11).

[7] *Python Pandas Tutorial - Tutorialspoint*. URL: www.tutorialspoint.com/python_pandas/index.html (page 11).

[8] *What Is a Confusion Matrix?* URL: https://intellipaat.com/blog/confusion-matrix-python/ (page 12).

[9] *sklearn.metrics.accuracy$_s$core*. URL: https://scikit-learn.org/stable/modules/generated/sklearn.metrics.accuracy_score.html (page 12).