NITTE MEENAKSHI INSTITUTE OF TECHNOLOGY

(AN AUTONOMOUS INSTITUTION, AFFILIATED TO VISVESVARAYA TECHNOLOGICAL UNIVERSITY, BELGAUM, APPROVED BY AICTE & GOVT.OF KARNATAKA)

LA Report

on

Fake news Detection

Submitted in partial fulfilment of the requirement for the Learning Assessment of Introduction to Machine Learning 18CSE751 Course of 7th Semester

Bachelor of Engineering

in

Computer Science and Engineering

Submitted by:

K Viddya	1NT18CS065
Krithika G Devadiga	1NT18CS080
Jasmine	1NT18CS004

Submitted to

Dr. Vani Vasudevan

Assistant Professor, Dept. of CS&E

NMIT

Department of Computer Science and Engineering

(Accredited by NBA Tier-1)

2021-2022

ABSTRACT

The proliferation of misleading information in everyday access media outlets such as social media feeds, news blogs and online newspapers have made it challenging to identify trustworthy news sources. The spread of fake news and misinformation is causing serious problems to society, partly due to the fact that more and more people only read headlines or highlights of news assuming that everything is reliable, instead of carefully analyzing whether it can contain distorted or false information. Specifically, the headline of a correctly designed news item must correspond to a summary of the main information of that news item. This increases the need for a computational tool that is able to provide insight to the reliability of the content. In this project, we have used natural language processing techniques and machine learning algorithms to classify whether the news is fake or not. Using a benchmark dataset and machine learning models like SVM, Naive-Bayes and Logistic Regression, we are predicting the possibility of the news to be true or false.

CONTENTS

INTRODUCTION

What is machine learning?

One of the thriving areas of artificial intelligence is machine learning. Machine learning had existed long before I was born, so why has it gained such traction after nearly 50 years? We now have access to an enormous amount of data, whether as images, text, audio, or video—this data power machine learning.

There are two schools of thought when it comes to Learning,

learning by example, humans learn everything by example.
Humans learn due to innate capabilities that yet need figuring.
The first thought was adopted to train machines using past data /examples/ samples, and that's the basis of the current state of the art for machine learning.

Machine learning basics:

Machine learning uses the same idea as human Learning to teach machines to learn from past data and perform tasks at a much faster pace than humans. Let's use an example to illustrate how it happens. Assume you want to discover what kind of song a person likes or dislikes based on several parameters such as voice gender, genre, tempo, mood, and personal preference. Let's say this person likes rapid, soaring music and dislikes light, calm music. Feeding the machine various samples of his/her likes and dislikes, the machine can determine whether or not the next song chosen is one of his/her preferences.

As the machine is trained with examples for his likes and dislikes based on tempo and intensity, now if we ask if he likes “x” song? The machine here uses an algorithm (K-nearest neighbor) to decide that “x” is a song that this person "likes"(as its nearest neighbors lie more in the like group).

That was something that everyone could see while listening to their favorite music on Spotify, Gaana, or any other music streaming service. It uses machine learning to recommend a song based on the player’s song selection.

Ways you can make machines learn:

Supervised Learning: In my previous example, I could train my machine with samples in which I've already stated that the person "likes" and "dislikes" certain songs, i.e., I've noted a decision with each instance. Supervised Learning is when the decision/conclusion associated with the set of parameters is already known. In supervised Learning, these decisions are usually called labels. We fed the machine with labeled data.

Unsupervised Learning: You train your machine with different parameters here, but there is no decision associated with it. It does grouping using various techniques(algorithms) to assist humans in making decisions regarding multiple groups. In unsupervised learning, we fed the machine with unlabeled data.

Reinforcement learning: This is reward-based Learning that works on feedback. We give positive feedback for each correct action performed by the machine, while negative feedback for each erroneous action. Here again, we make use of Unlabeled data.

MOTIVATION

Fake news denotes a type of yellow press which intentionally presents misinformation or hoaxes spreading through both traditional print news media and recent online social media. Fake news has existed for a long time. In recent years, due to the booming developments of online social networks, fake news for various commercial and political purposes has been appearing in large numbers and widespread in the online world. With deceptive words, online social network users can get infected by this online fake news easily, which has brought about tremendous effects on the offline society already. During this time where everyone is mostly dependent on the news they see on social media and even news channels, it is necessary to make sure that the news that we are consuming is true.

PROBLEM DOMAIN

Fake news has significant differences compared with traditional suspicious information, like spam in various aspects:

· Impact on society: spams usually exist in personal emails or specific review websites and merely have a local impact on a small number of audiences, while the impact fake news in online social networks can be tremendous due to the massive user numbers globally, which is further boosted by the extensive information sharing and propagation among these users

· Audiences' initiative: instead of receiving spam emails passively, users in online social networks may seek for, receive and share news information actively with no sense about its correctness.

· Identification difficulty: via comparisons with abundant regular messages (in emails or review websites), spams are usually easier to be distinguished; meanwhile, identifying fake news with erroneous information is incredibly challenging, since it requires both tedious evidence-collecting and careful fact checking due to the lack of other comparative news articles available.

Fake news paves the way for deceiving others and promoting ideologies. These people who produce the wrong information benefit by earning money with the number of interactions on their publications. Spreading disinformation holds various intentions, in particular, to gain favor in political elections, for business and products, done out of spite or revenge. Humans can be gullible and fake news is challenging to differentiate from the normal news. Most are easily influenced, especially by the sharing of friends and family due to relations and trust. Detection of fake news online is important in today's society as fresh news content is rapidly being produced as a result of the abundance of available technology. Due to this false information is reaching a large number of people.

AIM AND OBJECTIVES

- The primary objective of the project is to classify whether the input news is true or false.
- The project also aims to give the probability of the news to be true.
- In this project we have trained and tested 3 different classification models. They are Naive Bayes classifier, Logistic regression and Support vector machine respectively.
- The Confusion matrix is evaluated from all the 3 classification models.
- The classifier with the highest accuracy is selected and is validated.

DATA SOURCE AND DATA QUALITY

The data source used for this project is the 'LIAR' dataset which contains 3 files with .tsv format for testing, training and validation. It is 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.A decade-long of 12.8K manually labeled short statements were collected in various contexts from POLITIFACT.COM's API, which provides detailed analysis report and links to source documents for each case. This dataset can be used for fact-checking research as well. Notably, this new dataset is an order of magnitude larger than previously largest public fake news datasets of similar type. The original dataset, retrieved from Kaggle.com, contained 13 variables/columns for training, testing and validation sets. To make things simple we have chosen only 2 variables from this original dataset for this classification. The columns used are:

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

The newly created dataset has only 2 classes as compared to 6 from the original classes. Below is the 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

As a data quality check, the dataset was passed through a data integrity check function, where all the three files (train, test, validation) were checked for missing values. It was incurred that the dataset had no null values. The dataset is claimed to be created by manually labeling short statements that were collected in various contexts from POLITIFACT.COM's API, which provides detailed analysis reports and links to source documents for each case, to prove its authenticity. This means that this dataset is best suited for fake news detection and fact checking.

DATA PRE-PROCESSING

Data preprocessing is a data mining technique which is used to transform the raw data in a useful and efficient format.

The major preprocessing techniques used are:

- Ngram-N-gram can be defined as the contiguous sequence of n items from a given sample of text or speech. The items can be letters, words, or base pairs according to the application. The N-grams typically are collected from a text or speech corpus. Unigram means taking only one word at a time, bigram means taking two words at a time and trigram means taking three words at a time.
- CountVectorizer- It is used to convert a collection of text documents to a matrix of token counts.Countvectorizer makes it easy for text data to be used directly in machine learning and deep learning models such as text classification. Countvectorizer gives the number of frequency with respect to the index of vocabulary . For text input:

  ```
  text = [‘Hello my name is james' , ’this is my python notebook’]
  ```

  Countvectorizer will be:

	hello	is	james	my	name	notebook	python	this
0	1	1	1	1	1	0	0	0
1	0	1	0	1	0	1	1	1

- TF-IDF Vectorizer-TF-IDF is an abbreviation for Term Frequency Inverse Document Frequency.It is used to transform text into a meaningful representation of numbers which is used to fit machine algorithms for prediction. *tf-idf* considers overall documents of weight of words.The meaning increases proportionally to the number of times in the text a word appears but is compensated by the word frequency in the corpus (data-set).

MACHINE LEARNING MODELS

Naive Bayes classifier

Naive Bayes classifiers are linear classifiers based on Bayes' theorems. The model generated is probabilistic. It estimates conditional probability, which is the probability that something will happen, given that something else has already occurred. For example, the given mail is likely spam given the appearance of the words such as 'prize'.

It is called naive due to the assumption that the features in the dataset are mutually independent. In the real-world, the independence assumption is often violated, but naive Bayes classifiers still tend to perform very well.

The idea is to factor all available evidence in the form of predictors into the naive Bayes rule to obtain a more accurate probability for class prediction. Being relatively robust, easy to implement, fast, and accurate, naive Bayes classifiers are used in many different fields.

- Event: Outcome of an experiment.
- Experiment: Process performed to understand possible outcomes.
- Sample Space: Set of all outcomes of an experiment.
- Probability: Chance of a particular event taking place.
- Joint Probability: It is the probability of multiple events occurring together.
  From a pack of cards, what is the probability that the card is a red king?
- Conditional Probability: It is invalid if the joint probability is zero
  From a pack of cards, what is the probability that the card is a king given that it is red?

Support Vector Machines

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. SVM chooses the extreme points/vectors that help in creating the hyperplane. Hyperplane is the best decision boundary that helps to classify the data points and the extreme cases are called support vectors, and hence the algorithm is termed as Support Vector Machine. The dimensions of the hyperplane depend on the features present in the dataset, which means if there are 2 features, then the hyperplane will be a straight line. And if there are 3 features, then the 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.

SVM can be of two types - Linear and Nonlinear. 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 the classifier used is called Linear SVM classifier. 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 the classifier used is called Non-linear SVM classifier. We have used Linear SVM for our project.

For linearly separable data, the best hyperplane is chosen such that it represents the largest separation or margin between the two classes. So we choose the hyperplane whose distance from it to the nearest data point on each side is maximized.The SVM algorithm has the characteristics to ignore the outlier and finds the best hyperplane that maximizes the margin. SVM is robust to outliers.So in this type of data points what SVM does is it finds maximum margin as done with previous data sets along with that it adds a penalty each time a point crosses the margin. So the margins in these types of cases are called soft margins.

Linear Regression

Linear Regression is a machine learning algorithm based on supervised learning. It performs a regression task. Regression models a target prediction value based on independent variables. It is mostly used for finding out the relationship between variables and forecasting. Linear regression performs the task to predict a dependent variable value (y) based on a given independent variable (x). The dependent variable is also called the response variable. Independent variables are also called explanatory or predictor variables. So, this regression technique finds out a linear relationship between x (input) and y(output).When working with linear regression, our main goal is to find the best fit line that means the error between predicted values and actual values should be minimized. The best fit line will have the least error.

Hypothesis function for Linear Regression :

$$y = \theta_1 + \theta_2.x$$

While training the model we are given :
x: input training data (univariate – one input variable(parameter))
y: labels to data (supervised learning)

When training the model – it fits the best line to predict the value of y for a given value of x. The model gets the best regression fit line by finding the best θ1 and θ2 values.
θ1: intercept
θ2: coefficient of x

Once we find the best θ1 and θ2 values, we get the best fit line.The different values for weights or the coefficient of lines gives a different line of regression, so to calculate this we use the cost function. For Linear Regression, we use the **Mean Squared Error (MSE)** cost function, which is the average of squared error occurred between the predicted values and actual values.

The distance between the actual value and predicted values is called residual. If the observed points are far from the regression line, then the residual will be high, and so the cost function will be high. If the scatter points are close to the regression line, then the residual will be small and hence the cost function.

GridSearchCV (for hypermater tuning)

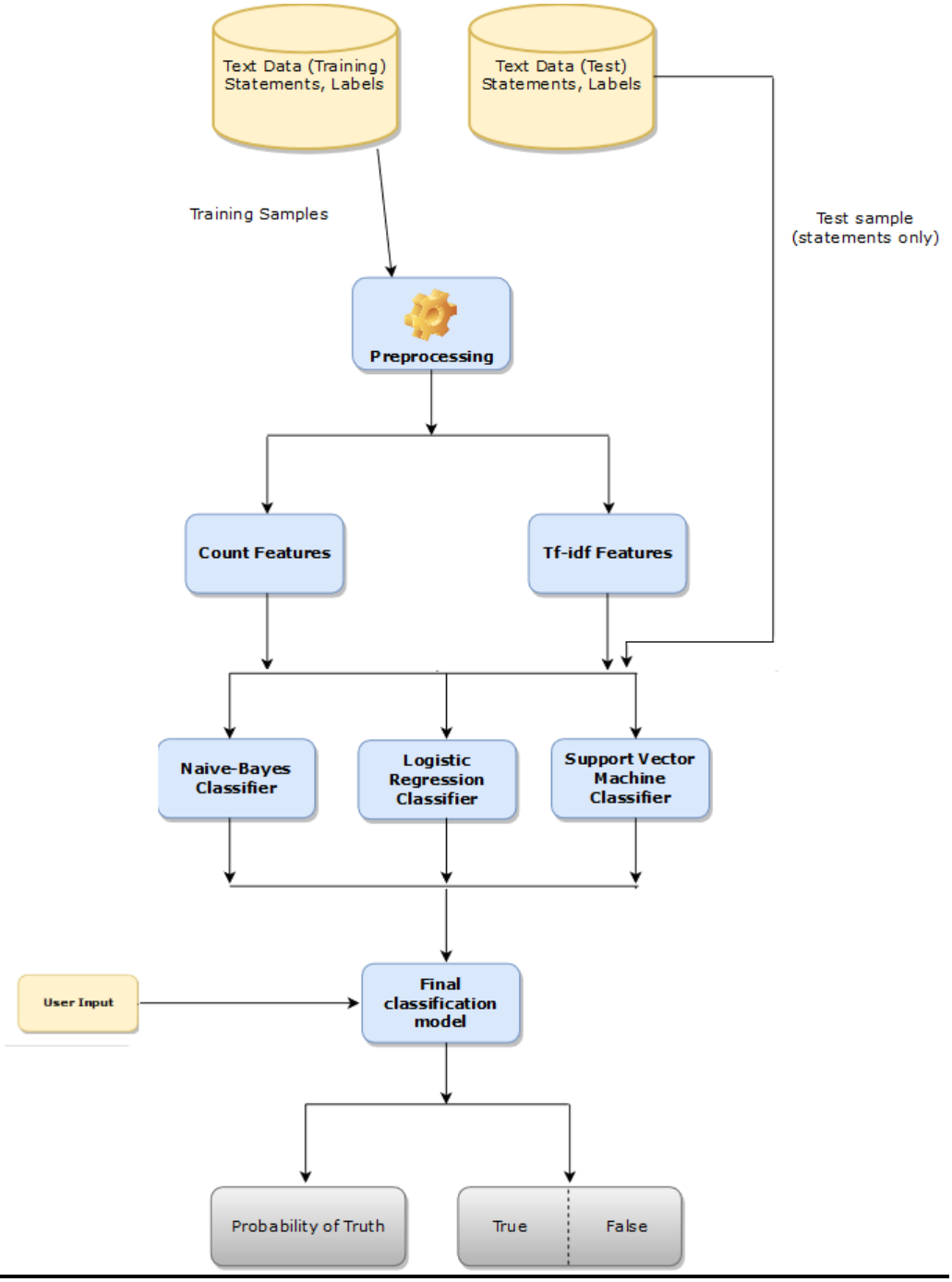
This entire process of finding different values for a hyperparameter for a machine learning model and then objectively selecting the best performance values is called Hyperparameter Tuning. Result of this is a set of values which we can use to configure our model.

To get the best set of hyperparameters we can use Grid Search. Grid Search passes all combinations of hyperparameters one by one into the model and checks the result. Finally it gives us the set of hyperparameters which gives the best result after passing in the model. We

make a dictionary called parameters for the parameters of the model which we want to pass to through GridSearchCV to get the best parameters.

Important parameters are:

- estimator: In this we have to pass the models or functions on which we want to use GridSearchCV
- param_grid: Dictionary or list of parameters of models or functions in which GridSearchCV has to select the best.
- n_jobs : This signifies the number of jobs to be run in parallel, -1 signifies the use of all processors.



RESULTS AND DISCUSSION

Confusion matrix

A confusion matrix is a table that is often used to describe the performance of a classification model (or "classifier") on a set of test data for which the true values are known. The confusion matrix itself is relatively simple to understand, but the related terminology can be confusing.

What can we learn from this matrix?

- There are two possible predicted classes: "yes" and "no". If we were predicting whether the news is true or false, for example, "true" would mean that the news is true, and "false" would mean that the news is false.
- For example, if the classifier made a total of 165 predictions (e.g., 165 news headlines).
- Out of those 165 cases, the classifier predicted "true" 110 times, and "false" 55 times.
- In reality, 105 news headlines are true and 60 news headlines are false.

- **true positives (TP):** These are cases in which we predicted true and they are true.
- **true negatives (TN):** We predicted false, and they are false news.
- **false positives (FP):** We predicted true, but they are actually false news.
- **false negatives (FN):** We predicted false, but they actually is true news.

```
[26] build_confusion_matrix(nb_pipeline_ngram)
     build_confusion_matrix(logR_pipeline_ngram)
     build_confusion_matrix(svm_pipeline_ngram)

     Total statements classified: 10240
     Score: 0.7224053159841455
     score length 5
     Confusion matrix:
     [[ 758 3730]
      [ 390 5362]]
     Total statements classified: 10240
     Score: 0.7044355553757985
     score length 5
     Confusion matrix:
     [[1580 2908]
      [1043 4709]]
     Total statements classified: 10240
     Score: 0.6790920142902143
     score length 5
     Confusion matrix:
     [[2016 2472]
      [1524 4228]]
     (None, None, None, None, None)
```

```
[22] build_confusion_matrix(svm_pipeline)
```

```
Total statements classified: 10240
Score: 0.6104687487924283
score length 5
Confusion matrix:
[[2260 2228]
 [2246 3506]]
/usr/local/lib/python3.7/dist-packages/sklearn/svm/_ba
  ConvergenceWarning,
(None, None, None, None, None)
```

```
[21] build_confusion_matrix(logR_pipeline)
```

```
Total statements classified: 10240
Score: 0.6467945943441228
score length 5
Confusion matrix:
[[2252 2236]
 [1934 3818]]
(None, None, None, None, None)
```

```
[20] build_confusion_matrix(nb_pipeline)
```

```
Total statements classified: 10240
Score: 0.66961153965076
score length 5
Confusion matrix:
[[2118 2370]
 [1664 4088]]
(None, None, None, None, None)
```

```
[20] build_confusion_matrix(nb_pipeline)

     Total statements classified: 10240
     Score: 0.66961153965076
     score length 5
     Confusion matrix:
     [[2118 2370]
      [1664 4088]]
     (None, None, None, None, None)
```

From the confusion matrix, we can see that random forest and logistic regression are best performing in terms of precision and recall (take a look into false positive and true negative counts which appear to be low compared to rest of the models)

PR curve

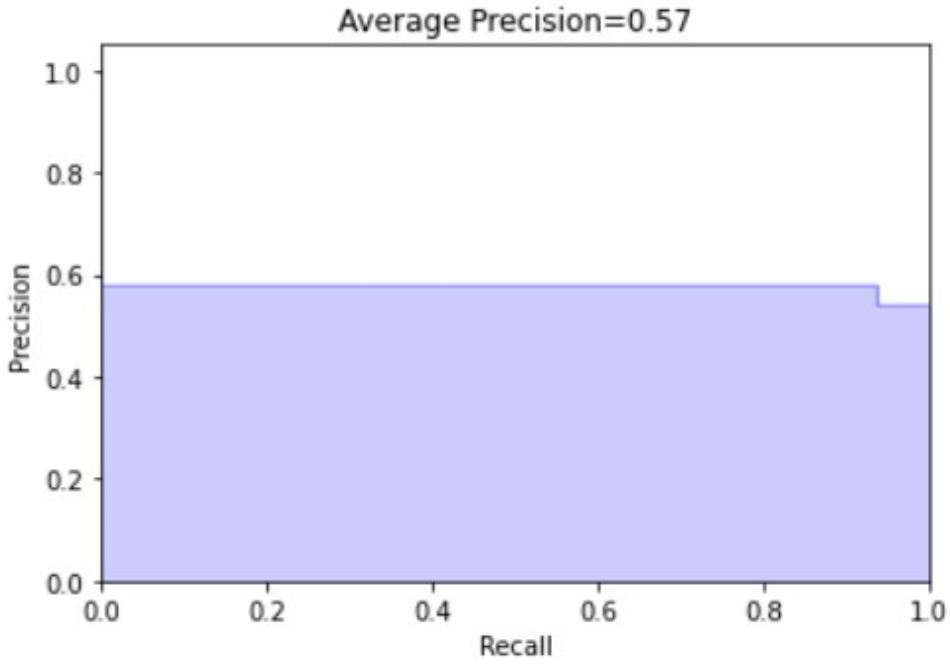
A precision-recall curve (or PR Curve) is a plot of the precision (y-axis) and the recall (x-axis) for different probability thresholds. The precision-recall curve is constructed by calculating and plotting the precision against the recall for a single classifier at a variety of thresholds. These curves give the shape we would expect — at thresholds with low recall, the precision is correspondingly high, and at very high recall, the precision begins to drop.

A PR curve is simply a graph with Precision values on the y-axis and Recall values on the x-axis. In other words, the PR curve contains TP/(TP+FN) on the y-axis and TP/(TP+FP) on the x-axis.

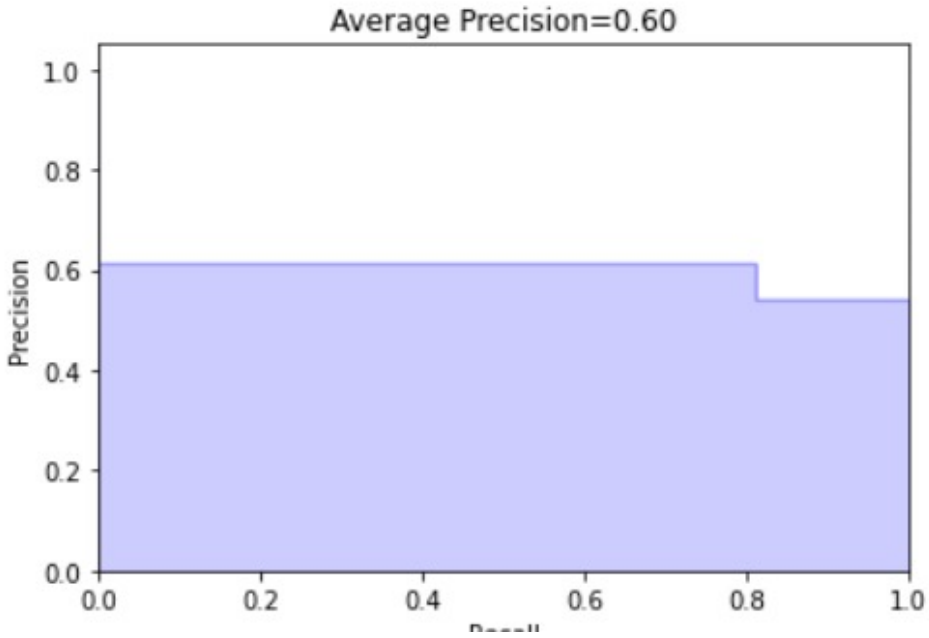
- It is important to note that Precision is also called the Positive Predictive Value (PPV).
- Recall is also called Sensitivity, Hit Rate or True Positive Rate (TPR).

It is desired that the algorithm should have both high precision, and high recall. However, most machine learning algorithms often involve a trade-off between the two. A good PR curve has greater AUC (area under curve)

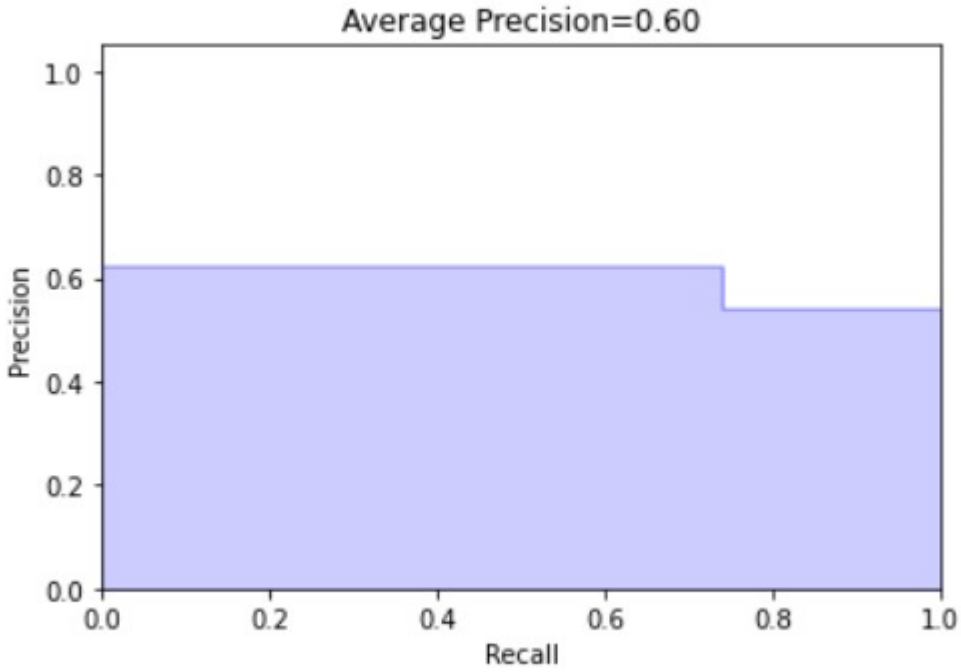
```
[29] plot_PR_curve(predicted_nb_ngram)
```




```
[28] plot_PR_curve(predicted_LogR_ngram)
```



```
[27] plot_PR_curve(predicted_svm_ngram)
```



The above graphs indicates that logistic regression classifier is most efficient.

Classification report

A Classification report is used to measure the quality of predictions from a classification algorithm. How many predictions are True and how many are False. More specifically, True Positives, False Positives, True negatives and False Negatives are used to predict the metrics of a classification.

Precision- Precision is one indicator of a machine learning model's performance – the quality of a positive prediction made by the model. Precision refers to the number of true positives divided by the total number of positive predictions (i.e., the number of true positives plus the number of false positives).

Recall -Recall is how many of the true positives were recalled (found), i.e. how many of the correct hits were also found. Precision (your formula is incorrect) is how many of the returned hits were true positive i.e. how many of the found were correct hits.

F1 score - F1 Score is the weighted average of Precision and Recall. Therefore, this score takes both false positives and false negatives into account. Intuitively it is not as easy to understand as accuracy, but F1 is usually more useful than accuracy, especially if you have an uneven class

distribution.

Support - Support is the number of actual occurrences of the class in the specified dataset. Imbalanced support in the training data may indicate structural weaknesses in the reported scores of the classifier and could indicate the need for stratified sampling or rebalancing.

```
[30] print(classification_report(test_news['Label'], predicted_nb_ngram))
     print(classification_report(test_news['Label'], predicted_LogR_ngram))
     print(classification_report(test_news['Label'], predicted_svm_ngram))
```

```
                precision    recall  f1-score   support

         False       0.72      0.19      0.30      1169
          True       0.58      0.94      0.71      1382

      accuracy                           0.59      2551
     macro avg       0.65      0.56      0.51      2551
  weighted avg       0.64      0.59      0.52      2551

                precision    recall  f1-score   support

         False       0.64      0.39      0.49      1169
          True       0.61      0.81      0.70      1382

      accuracy                           0.62      2551
     macro avg       0.62      0.60      0.59      2551
  weighted avg       0.62      0.62      0.60      2551

                precision    recall  f1-score   support

         False       0.61      0.47      0.53      1169
          True       0.62      0.74      0.68      1382

      accuracy                           0.62      2551
     macro avg       0.61      0.61      0.60      2551
  weighted avg       0.62      0.62      0.61      2551
```

CONCLUSION AND FUTURE DIRECTIONS

On plotting the PR-curve for the chosen 3 algorithms, we observed that the Naive Bayes algorithm gives the maximum average precision. We then evaluated these three models with the help of confusion matrices as well; it was observed that the Naive Bayes algorithm had the least number of misclassifications. This shows that Naive Bayes is the best suited algorithm for fake new detection. Later, since n-grams help in developing better features for the ML algorithms, we introduced it as well and evaluated the models in a similar fashion. Introduction of n-grams reduced the overall misclassifications significantly but this time, SVM and Linear Regression algorithms yield better average prediction values.

In future we would like to add the remaining variables/columns present in the dataset to add some more complexity and enhance the features of the model. We would also like to implement more models like random forest, gradient descent, etc, to have a better idea on how accurately these models provide results by comparing them and choosing the best suited option ultimately.

LESSON LEARNT

This project helped us to gain an in-depth knowledge on the different machine learning algorithms that we used for implementation. We came across various data preprocessing methods as well which were new to us, such as count vectoriser, tfidf vectorizer and N-grams. We learnt their uses and significance corresponding to text preprocessing for fake news detection. In Order to evaluate and compare the chosen machine learning models, we used methods like confusion matrix and PR-curves. This helped us to enhance our knowledge on their significance in machine learning and how they can be used efficiently to our advantage.

REFERENCES

https://towardsdatascience.com/identifying-fake-news-the-liar-dataset-713eca8af6ac

https://sites.cs.ucsb.edu/~william/papers/acl2017.pdf “Liar, Liar Pants on Fire”: A New Benchmark Dataset for Fake News Detection”

A. Jain and A. Kasbe, "Fake News Detection," 2018 IEEE International Students' Conference on Electrical, Electronics and Computer Science (SCEECS), 2018, pp. 1-5, doi: 10.1109/SCEECS.2018.8546944.

https://data-flair.training/blogs/advanced-python-project-detecting-fake-news/

https://www.hindawi.com/journals/complexity/2020/8885861

APPENDIX

The dataset used in the project is the liar dataset which is available at kaggle website. It is available at https://www.kaggle.com/hendrixwilsonj/liar-data-analysis.

PYTHON CODE IMPLEMENTED

```
import pandas as pd

import csv

import numpy as np

import nltk

from sklearn.feature_extraction.text import CountVectorizer

from sklearn.feature_extraction.text import TfidfTransformer

from sklearn.feature_extraction.text import TfidfVectorizer

from sklearn.pipeline import Pipeline

import pickle

from sklearn.naive_bayes import MultinomialNB

from sklearn.linear_model import  LogisticRegression

from sklearn import svm

from sklearn.model_selection import KFold

from     sklearn.metrics     import     confusion_matrix,     f1_score,
classification_report

from sklearn.model_selection import GridSearchCV

import matplotlib.pyplot as plt

from sklearn.metrics import precision_recall_curve

from sklearn.metrics import average_precision_score

import seaborn as sb

test_filename = 'test.csv'

train_filename = 'train.csv'
```

```
valid_filename = 'valid.csv'


train_news = pd.read_csv(train_filename)
test_news = pd.read_csv(test_filename)
valid_news = pd.read_csv(valid_filename)
def data_obs():
    print("training dataset size:")
    print(train_news.shape)
    print(train_news.head(10))


    print("test dataset size:")
    print(test_news.shape)
    print(test_news.head(10))


    print("validity dataset size:")
    print(valid_news.shape)
    print(valid_news.head(10))


data_obs()
def create_distribution(dataFile):


    return sb.countplot(x='Label', data=dataFile, palette='hls')
create_distribution(train_news)
create_distribution(test_news)
create_distribution(valid_news)
```

```
countV = CountVectorizer()

train_count = countV.fit_transform(train_news['Statement'].values)


print(countV)

print(train_count)

tfidfV = TfidfTransformer()

train_tfidf = tfidfV.fit_transform(train_count)


tfidf_ngram                                                              =
TfidfVectorizer(stop_words='english',ngram_range=(1,4),use_idf=True,smo
oth_idf=True)

nb_pipeline = Pipeline([

        ('NBCV',countV),

        ('nb_clf',MultinomialNB())])


nb_pipeline.fit(train_news['Statement'],train_news['Label'])

predicted_nb = nb_pipeline.predict(test_news['Statement'])

np.mean(predicted_nb == test_news['Label'])

logR_pipeline = Pipeline([

        ('LogRCV',countV),

        ('LogR_clf',LogisticRegression(max_iter=250))

        ])


logR_pipeline.fit(train_news['Statement'],train_news['Label'])

predicted_LogR = logR_pipeline.predict(test_news['Statement'])

np.mean(predicted_LogR == test_news['Label'])
```

```
svm_pipeline = Pipeline([
        ('svmCV',countV),
        ('svm_clf',svm.LinearSVC( max_iter=1000))
        ])

svm_pipeline.fit(train_news['Statement'],train_news['Label'])
predicted_svm = svm_pipeline.predict(test_news['Statement'])
np.mean(predicted_svm == test_news['Label'])
def build_confusion_matrix(classifier):

    k_fold = KFold(n_splits=5)
    scores = []
    confusion = np.array([[0,0],[0,0]])

    for train_ind, test_ind in k_fold.split(train_news):
        train_text = train_news.iloc[train_ind]['Statement']
        train_y = train_news.iloc[train_ind]['Label']

        test_text =train_news.iloc[test_ind]['Statement']
        test_y =train_news.iloc[test_ind]['Label']

        classifier.fit(train_text,train_y)
        predictions = classifier.predict(test_text)

        confusion += confusion_matrix(test_y,predictions)
```

```
        score = f1_score(test_y,predictions)
        scores.append(score)

    return (print('Total statements classified:', len(train_news)),
    print('Score:', sum(scores)/len(scores)),
    print('score length', len(scores)),
    print('Confusion matrix:'),
    print(confusion))
def plot_PR_curve(classifier):

    precision, recall, thresholds = precision_recall_curve(test_news['Label'], classifier)
    average_precision = average_precision_score(test_news['Label'], classifier)

    plt.step(recall, precision, color='b', alpha=0.2,
             where='post')
    plt.fill_between(recall, precision, step='post', alpha=0.2,
                     color='b')

    plt.xlabel('Recall')
    plt.ylabel('Precision')
    plt.ylim([0.0, 1.05])
    plt.xlim([0.0, 1.0])
    plt.title('Average Precision={0:0.2f}'.format(average_precision))
plot_PR_curve(predicted_LogR)
```

```
plot_PR_curve(predicted_nb)

plot_PR_curve(predicted_svm)

build_confusion_matrix(logR_pipeline)

build_confusion_matrix(nb_pipeline)

build_confusion_matrix(svm_pipeline)

nb_pipeline_ngram = Pipeline([

        ('nb_tfidf',tfidf_ngram),

        ('nb_clf',MultinomialNB())])


nb_pipeline_ngram.fit(train_news['Statement'],train_news['Label'])

predicted_nb_ngram = nb_pipeline_ngram.predict(test_news['Statement'])

np.mean(predicted_nb_ngram == test_news['Label'])

logR_pipeline_ngram = Pipeline([

        ('LogR_tfidf',tfidf_ngram),

        ('LogR_clf',LogisticRegression(penalty="l2",C=1))

        ])


logR_pipeline_ngram.fit(train_news['Statement'],train_news['Label'])

predicted_LogR_ngram                                                    =
logR_pipeline_ngram.predict(test_news['Statement'])

np.mean(predicted_LogR_ngram == test_news['Label'])

svm_pipeline_ngram = Pipeline([

        ('svm_tfidf',tfidf_ngram),

        ('svm_clf',svm.LinearSVC())

        ])
```

```
svm_pipeline_ngram.fit(train_news['Statement'],train_news['Label'])

predicted_svm_ngram = svm_pipeline_ngram.predict(test_news['Statement'])

np.mean(predicted_svm_ngram == test_news['Label'])

build_confusion_matrix(nb_pipeline_ngram)

build_confusion_matrix(logR_pipeline_ngram)

build_confusion_matrix(svm_pipeline_ngram)

plot_PR_curve(predicted_svm_ngram)

plot_PR_curve(predicted_LogR_ngram)

plot_PR_curve(predicted_nb_ngram)

print(classification_report(test_news['Label'], predicted_nb_ngram))

print(classification_report(test_news['Label'], predicted_LogR_ngram))

print(classification_report(test_news['Label'], predicted_svm_ngram))

parameters = {'LogR_tfidf__ngram_range': [(1, 1), (1,
2),(1,3),(1,4),(1,5)],

              'LogR_tfidf__use_idf': (True, False),

              'LogR_tfidf__smooth_idf': (True, False)

}


gs_clf = GridSearchCV(logR_pipeline_ngram, parameters, n_jobs=-1)

gs_clf                                                                  =
gs_clf.fit(train_news['Statement'][:10000],train_news['Label'][:10000])

gs_clf.best_score_

gs_clf.best_params_

gs_clf.cv_results_

logR_pipeline_final = Pipeline([
```

```
('LogR_tfidf',TfidfVectorizer(stop_words='english',ngram_range=(1,4),us
e_idf=True,smooth_idf=False)),
        ('LogR_clf',LogisticRegression(penalty="l2",C=1))
        ])

logR_pipeline_final.fit(train_news['Statement'],train_news['Label'])
predicted_LogR_final                                                =
logR_pipeline_final.predict(test_news['Statement'])
np.mean(predicted_LogR_final == test_news['Label'])
print(classification_report(test_news['Label'], predicted_LogR_final))
model_file = 'final_model.sav'
pickle.dump(logR_pipeline_ngram,open(model_file,'wb'))
var = input("Please enter the news text you want to verify: ")
print("You entered: " + str(var))
def detecting_fake_news(var):
#retrieving the best model for prediction call
    load_model = pickle.load(open('final_model.sav', 'rb'))
    prediction = load_model.predict([var])
    prob = load_model.predict_proba([var])

    return (print("The given statement is ",prediction[0]),
        print("The truth probability score is ",prob[0][1]))
if __name__ == '__main__':
    detecting_fake_news(var)
```