INFYTECHNISM

www.Infytechnism.org

Fake News Detection Using Machine Learning Ensemble Methods

Project by,

Soujanya Syamal

Institute of Engineering & Management

www.soujanyasyamal.com

soujanyasyamal@gmail.com

August 17, 2021

Contents-

Abstract

The rise of false information in everyday access media venues like social media feeds, news blogs, and online newspapers has made it difficult to identify reliable news sources, necessitating the development of computer algorithms that can assess the authenticity of online content. We focus on the automatic detection of false information in internet news in this research. We make a two-fold contribution. First, we present two new datasets for the purpose of detecting fake news, each of which covers seven different news categories. We give many explanatory analyses on the identification of linguistic discrepancies in false news content, as well as a detailed description of the collecting, annotation, and validation procedure. Second, we run a series of learning experiments in order to develop reliable fake news detectors. Furthermore, we presented comparisons of the automatic and manual detection of fake news.

Introduction-

As an increasing amount of our lives is spent interacting online through social media platforms, more and more people tend to hunt out and consume news from social media instead of traditional news organizations. The explanations for this alteration in consumption behavior are inherent within the nature of those social media platforms: (i) it's often more timely and fewer expensive to consume news on social media compared with traditional journalism, like newspapers or television; and (ii) it's easier to further share and discuss the news with friends or other readers on social media. For instance, 62 percent of adults get news on social media in 2016, while in 2012; only 49 percent reported seeing news on social media. It had been also found that social media now outperforms television because the major news source. Despite the benefits provided by social media, the standard of stories on social media is less than traditional news organizations. However, because it's inexpensive to supply news online and far faster and easier to propagate through social media, large volumes of faux news, i.e., those news articles with intentionally false information, are produced online for a spread of purposes, like financial and political gain. It had been estimated that over 1 million tweets are associated with fake news. “Fake news" was even named the word of the year by the Macquarie dictionary in 2016. The extensive spread of faux news can have a significant negative impact on individuals and society. First, fake news can shatter the authenticity equilibrium of the news ecosystem for instance; it's evident that the most popular fake news was even more outspread on

Facebook than the most accepted genuine mainstream news in 2016. Second, fake news intentionally persuades consumers to simply accept biased or false beliefs. Fake news is typically manipulated by propagandists to convey political messages or influence. Third, fake news changes the way people interpret and answer real news, for instance, some fake news was just created to trigger people's distrust and make them confused; impeding their abilities to differentiate what is true from what's not. To assist mitigate the negative effects caused by fake news (both to profit the general public and therefore the news ecosystem). It's crucial that we build up methods to automatically detect fake news broadcast on social media. Internet and social media have made the access to the news information much easier and comfortable. Often Internet users can pursue the events of their concern in online form, and increased number of the mobile devices makes this process even easier. But with great possibilities come great challenges. Mass media have an enormous influence on the society, and because it often happens, there's someone who wants to require advantage of this fact. Sometimes to realize some goals mass-media may manipulate the knowledge in several ways. This result in producing of the news articles that isn't completely true or maybe completely false. There even exist many websites that produce fake news almost exclusively. They intentionally publish hoaxes, half-truths, propaganda and disinformation asserting to be real news – often using social media to drive web traffic and magnify their effect. The most goals of faux news websites are to affect the general public opinion on certain matters mostly political. Thus, fake news may be a global issue also as a worldwide challenge. Many scientists believe that fake news issue could also be addressed by means of machine learning and AI. There's a reason for that: recently AI algorithms have begun to work far better on many classification problems because hardware is cheaper and larger datasets are Institute of Engineering & Management | 8 available. There are several influential articles about automatic deception detection. They collect the info by means of asking people to directly provide true or false information on several topics – abortion, execution and friendship. The accuracy of the detection achieved by the system is around 70%. This text describes an easy fake news detection method supported one among the synthetic intelligence algorithms – naïve Bayes classifier, Random Forest and Logistic Regression. The goal of the research is to look at how these particular methods work for this particular problem given a manually labelled news dataset and to support the thought of using AI for fake news detection. The difference between these article and articles on the similar topics is that during this paper Logistic Regression was specifically used for fake news detection; also, the developed

system was tested on a comparatively new data set, which gave a chance to gauge its performance on a recent data.

Literature Review-

Mykhailo Granik et. al. in their paper shows a simple approach for fake news detection using naive Bayes classifier. This approach was implemented as a software system and tested against a data set of Facebook news posts. They were collected from three large Facebook pages each from the right and from the left, as well as three large mainstream political news pages (Politico, CNN, ABC News). They achieved classification accuracy of approximately 74%. Classification accuracy for fake news is slightly worse. This may be caused by the skewness of the dataset: only 4.9% of it is fake news. Himank Gupta et. al. [1] gave a framework based on different machine learning approach that deals with various problems including accuracy shortage, time lag (BotMaker) and high processing time to handle thousands of tweets in 1 sec. Firstly, they have collected 400,000 tweets from HSpam14 dataset. Then they further characterize the 150,000 spam tweets and 250,000 non- spam tweets. They also derived some lightweight features along with the Top-30 words that are providing highest information gain from Bag-of-Words model. 4. They were able to achieve an accuracy of 91.65% and surpassed the existing solution by approximately18%.

Marco L. Della Vedova et. al.[2] first proposed a novel ML fake news detection method which, by combining news content and social context features, outperforms existing methods in the literature, increasing its accuracy up to 78.8%. Second, they implemented their method within a Facebook Messenger Chabot and validate it with a real-world application, obtaining a fake news detection accuracy of 81.7%. Their goal was to classify a news item as reliable or fake; they first described the datasets they used for their test, then presented the content-based approach they implemented and the method they proposed to combine it with a social-based approach available in the literature. The resulting dataset is composed of 15,500 posts, coming from 32 pages (14 conspiracy pages, 18 scientific pages), with more than 2, 300, 00 likes by 900,000+ users. 8,923 (57.6%) posts are hoaxes and 6,577 (42.4%) are non-hoaxes.

Cody Buntain et. al. [3] develops a method for automating fake news detection on Twitter by learning to predict accuracy assessments in two credibility-focused Twitter datasets: CREDBANK, a crowd sourced dataset of accuracy assessments for events in Twitter, and PHEME, a dataset of potential rumours in Twitter and journalistic assessments of their accuracies. They apply this method to Twitter content sourced from BuzzFeed‟s fake news dataset. A feature analysis identifies features that are most predictive for crowd sourced and journalistic accuracy assessments, results of which are consistent with prior work. They rely on identifying highly retweeted threads of conversation and use the features of these threads to classify stories, limiting this work‘s applicability only to the set of popular tweets. Since the majority of tweets are rarely retweeted, this method therefore is only usable on a minority of Twitter conversation threads.

In his paper, Shivam B. Parikh et. al. [4] aims to present an insight of characterization of news story in the modern diaspora combined with the differential content types of news story and its impact on readers. Subsequently, we dive into existing fake news detection approaches that are heavily based on textbased analysis, and also describe popular fake news datasets. We conclude the paper by identifying 4 key open research challenges that can guide future research. It is a theoretical Approach which gives Illustrations of fake news detection by analysing the psychological factors.

Scikit-learn [5] is a Python module integrating a wide range of state-of-theart machine learning algorithms for medium-scale supervised and unsupervised problems. This package focuses on bringing machine learning to non-specialists using a general-purpose high-level language. Emphasis is put on ease of use, performance, documentation, and API consistency. It has minimal dependencies and is distributed under the simplified BSD license, encouraging its use in both academic and commercial settings.

Automatic fake news [6] detection is a challenging problem in deception detection, and it has tremendous real-world political and social impacts. However, statistical approaches to combating fake news has been dramatically limited by the lack of labeled benchmark datasets. In this paper, we present liar: a new, publicly available dataset for fake news detection. We collected a decadelong, 12.8K manually labeled short statements in various contexts, 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. Empirically, we investigate automatic fake news detection based on surface-level linguistic patterns. We have designed a novel, hybrid convolutional neural network to integrate meta-data with text. We show that this hybrid approach can improve a text-only deep learning model.

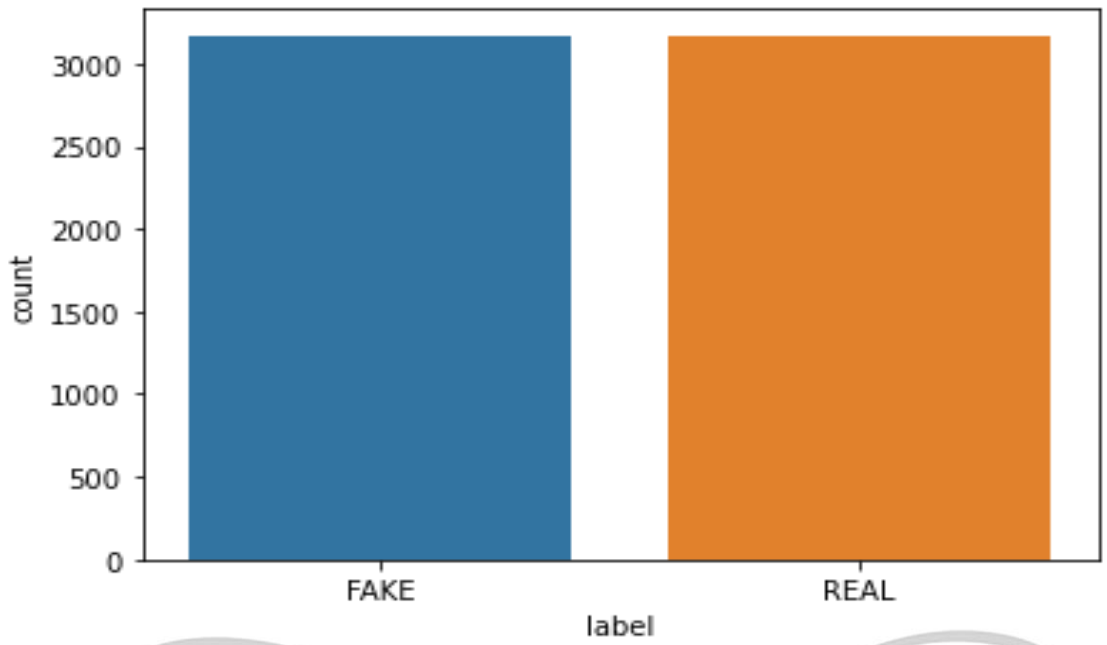
The problem of high-quality automatic natural language processing [12] is one of the most important problems in computational linguistics. Automatic natural language processing is used in information retrieval, in tasks of text generation and text recognition, in machine translation, in sentiment analysis and so on. All of these areas require specialized linguistic and mathematical models to represent the morphology, syntax, and semantics of text in a form that is convenient for automatic processing. The article describes a developed system that implements specific linguistic tasks related to the processing of Ukrainian language, that is text preprocessing, morphological and lexical analyzes of text. In order to create such a system, an analysis of the available natural language textprocessing tools was carried out and the possibility of using them for text processing of Ukrainian language was examined. Also, the most appropriate text processing tools in Ukrainian language were selected. The basic stages of text preprocessing were considered in detail and algorithms of their program implementation were given. In addition, the results of the developed system were demonstrated.

Data Set-

A data set is an ordered collection of data. While handling the data, the data set can be a bunch of tables, schema and other objects. The data are essentially organized to a certain model that helps to process the needed information. The set of data is any permanently saved collection of information that usually contains either case-level, gathered data, or statistical guidance level data.

The dataset we have used in this project it is well distributed dataset and it has total of 7796 attributes and there has 4 columns. The total number of Data 7796. This Data set contains some real data and some fake data However, the number of real data and fake data are same. If we are train after that, the possibility of getting basis is less.

Fake data=Real data is like that-



Example-

	tittle	Text	label
8476	You Can Smell Hillaryâ€™s Fear	Daniel	FAKE

Preprocessing of Dataset

Social media data is highly unstructured – majority of them are informal communication with typos, slangs and bad-grammar etc. Quest for increased performance and reliability has made it imperative to develop techniques for utilization of resources to make informed decisions. To achieve better insights, it is necessary to clean the data before it can be used for predictive modelling. For this purpose, basic preprocessing was done on the News training data.

- **Stopwords remove** - To remove stop words from a sentence, it can be divided your text into words and then remove the word if it exits in

the list of stop words provided by NLTK. In the script above, we first import the stopwords collection from the nltk. corpus module. Next, we import the word tokenize () method from the nltk.

- **Tokenization -** Tokenization is the process of turning a meaningful piece of data, such as an account number, into a random string of characters called a token that has no meaningful value if breached. Tokens serve as reference to the original data but cannot be used to guess those values.

- **Vectorization –** We done Vectorization by TFID.

  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.

  The TfidfVectorizer converts a collection of raw documents into a matrix of TF-IDF features.

  It computes ―relative frequency‖ that a word appears in a document compared to its frequency across all documents TF-IDF weight represents the relative importance of a term in the document and entire corpus [17]. TF stands for Term Frequency: It calculates how frequently a term appears in a document. Since, every document size varies, a term may appear more in a long sized document that a short one. Thus, the length of the document often divides Term frequency.

Now let's take a look at the simple formula behind the TF-IDF statistical measure.

First let's define some notations:

- N is the number of documents we have in our dataset.
- d is a given document from our dataset.
- D is the collection of all documents.
- w is a given word in a document.

First step is to calculate the term frequency, our first measure if the score.

$$Tf(w,d) = \log(1 + f(w,d))$$

Here f(w,d) is the frequency of word w in document d.

Second step is to calculate the inverse term frequency.

$$idf(w, D) = \log\left(\frac{N}{f(w, D)}\right)$$

With N documents in the dataset and f(w, D) the frequency of word w in the whole dataset, this number will be lower with more appearances of the word in the whole dataset.

Final step is to compute the TF-IDF score by the following formula:

$$Tfidf(w,d,D) = tf(w,d) * idf(w,d)$$

- Padding - The standard way to add padding to a string in Python is using the str. rjust() function. It takes the width and padding to be used. If no padding is specified, the default padding of ASCII space is used.

Classifiers

A classifier is an algorithm that automatically orders or categorizes data into one or more of a set of "classes." One of the most common examples is an email classifier that scans emails to filter them by class label: Spam or Not Spam. Machine Learning Algorithm are helpful to automate tasks that previously had to be done manually. They can save huge amounts of time and money and make businesses more efficient.

1. We have used two different type of Classifier- i. Base Classifier
   ii. Ensemble Classifier

i.Base classifiers

This term is used to indicate the base component of a multiple classifier system. In other words, a multiple classifier system is made up by a set of base classifiers. Some authors use this term only when the multiple classifier system is designed using a single classification model and multiple versions of this base classifier are generated to build the multiple classifier system. We have used five types of base classifiers –

Passive Aggressive Classifier

About-

The Passive-Aggressive algorithms are a family of Machine learning algorithms. They are generally used for large-scale learning. It is one of the few 'onlinelearning algorithms '. In online machine learning algorithms, the input data comes in sequential order and the machine learning model is updated

step-bystep, as opposed to batch learning, where the entire training dataset is used at once. This is very useful in situations where there is a huge amount of data and it is computationally infeasible to train the entire dataset because of the sheer size of the data. A very good example of this would be to detect fake news on a social media website like Twitter, where new data is being added every second.

To dynamically read data from Twitter continuously, the data would be huge, and using an online-learning algorithm would be ideal. Passive-Aggressive algorithms are somewhat similar to a Perceptron model, in the sense that they do not require a learning rate. However, they do include a regularization parameter.

Working-

Passive-Aggressive algorithms are called so because -Passive- If the prediction is correct, keep the model and do not make any changes. i.e., the data in the example is not enough to cause any changes in the model. Aggressive- If the prediction is incorrect, make changes to the model. i.e., some change to the model may correct it.

Decision Tree Classifier

About-

Decision Tree is a Supervised learning technique that can be used for both classification and Regression problems, but mostly it is preferred for solving Classification problems. It is a tree-structured classifier, where internal nodes represent the features of a dataset, branches represent the decision rules and each leaf node represents the outcome. In a Decision tree, there are two nodes, which are the Decision Node and Leaf Node. Decision nodes are used to make any decision and have multiple branches, whereas Leaf nodes are the output of those decisions and do not contain any further branches. The decisions or the test are performed on the basis of features of the given dataset. It is a graphical representation for getting all the possible solutions to a problem/decision based on given conditions. It is called a decision tree

because, similar to a tree, it starts with the root node, which expands on further branches and constructs a tree-like structure. In order to build a tree, we use the CART algorithm, which stands for Classification and Regression Tree algorithm. A decision tree simply asks a question, and based on the answer (Yes/No), it further split the tree into subtrees. Decision Trees usually mimic human thinking ability while making a decision, so it is easy to understand. The logic behind the decision tree can be easily understood because it shows a tree-like structure.

Working- In a decision tree, for predicting the class of the given dataset, the algorithm starts from the root node of the tree. This algorithm compares the values of root attribute with the record attribute and based on the comparison, follows the branch and jumps to the next node. For the next node, the algorithm again compares the attribute value with the other sub-nodes and move further. It continues the process until it reaches the leaf node of the tree. The complete process can be better understood using the below algorithm:

- Step-1: Begin the tree with the root node, says S, which contains the complete dataset.
- Step-2: Find the best attribute in the dataset using Attribute Selection Measure (ASM).
- Step-3: Divide the S into subsets that contains possible values for the best attributes.
- Step-4: Generate the decision tree node, which contains the best attribute.
- Step-5: Recursively make new decision trees using the subsets of the dataset created in step -3. Continue this process until a stage is reached where you cannot further classify the nodes and called the final node as a leaf node.

Example: Suppose there is a candidate who has a job offer and wants to decide whether he should accept the offer or Not. So, to solve this problem, the decision tree starts with the root node (Salary attribute by ASM). The root node splits further into the next decision node (distance from the office) and one leaf node based on the corresponding labels. The next decision node further gets split into one decision node (Cab facility) and one leaf node. Finally, the decision node splits into two leaf nodes (Accepted offers and Declined offer).

Summarized, for each descriptive feature, we sum up the resulting entropies for splitting the dataset along the feature values and additionally weight the feature value entropies by their occurrence probability.

Random Forest Classifier

About-

Random forest is a flexible, easy to use machine learning algorithm that produces, even without hyper-parameter tuning, a great result most of the time. It is also one of the most used algorithms, because of its simplicity and diversity (it can be used for both classification and regression tasks).It is a meta estimator that fits a number of decision tree classifiers on various sub-samples of the dataset and uses averaging to improve the predictive accuracy and control over-fitting.

Working-Random Forest is a supervised learning algorithm. The "forest" it builds, is an ensemble of decision trees, usually trained with the “bagging” method. The general idea of the bagging method is that a combination of learning models increases the overall result**.** Random forest builds multiple decision trees and merges them together to get a more accurate and stable prediction.

One big advantage of random forest is that it can be used for both classification and regression problems, which form the majority of current machine learning systems. Random forest has nearly the same hyperparameters as a decision tree or a bagging classifier. Fortunately, there's no need to combine a decision tree with a bagging classifier because you can

easily use the classifier-class of random forest. With random forest, you can also deal with regression tasks by using the algorithm's regressor. Random forest adds additional randomness to the model, while growing the trees. Instead of searching for the most important feature while splitting a node, it searches for the best feature among a random subset of features. This results in a wide diversity that generally results in a better model. Therefore, in random forest, only a random subset of the features is taken into consideration by the algorithm for splitting a node. You can even make trees more random by additionally using random thresholds for each feature rather than searching for the best possible thresholds (like a normal decision tree does).

Formula The Gini Impurity of a node is the probability that a randomly chosen sample in a node would be incorrectly labelled if it was labelled by the distribution of samples in the node.

SVC (Support Vector Classifier)

ABOUT-

In machine learning, support-vector classifier (SVC) are supervised learning models with associated learning algorithms that analyse data for classification and regression analysis. Given a set of training examples, each marked as belonging to one of two categories, a SVC training algorithm builds a model that assigns new examples to one category or the other, making it a nonprobabilistic binary linear classifier (although methods such as Platt scaling exist to use SVC in a probabilistic classification setting). SVC maps training examples to points in space so as to maximise the width of the gap between the two categories. New examples are then mapped into that same space and predicted to belong to a category based on which side of the gap they fall.

In addition to performing linear classification, SVCs can efficiently perform a nonlinear classification using what is called the kernel trick, implicitly mapping their inputs into high-dimensional feature spaces.

When data are unlabeled, supervised learning is not possible, and an unsupervised learning approach is required, which attempts to find natural clustering of the data to groups, and then map new data to these formed groups. .

WORKING-The objective of a SVC (Support Vector Classifier) is to fit to the data provided, returning a "best fit" hyperplane that divides, or categorizes data. From there, after getting the hyperplane, we can then feed some features to our classifier to see what the "predicted" class is. This makes this specific algorithm rather suitable for our uses, though you can use this for many situations.

Multinomial Naive Bayes classifier

About-There are thousands of software or tools for the analysis of numerical data but there are very few for texts. Multinomial Naive Bayes is one of the most popular supervised learning classifications that is used for the analysis of the categorical text data. Text data classification is gaining popularity because there is an enormous amount of information available in email, documents, websites, etc. that needs to be analysed. Knowing the context around a certain type of text helps in finding the perception of a software or product to users who are going to use it.

Working-Multinomial Naive Bayes is a powerful algorithm that is used for text data analysis and with problems with multiple classes. To understand Naive Bayes theorem's working, it is important to understand the Bayes theorem concept first as it is based on the latter.

Bayes theorem, formulated by Thomas Bayes, calculates the probability of an event occurring based on the prior knowledge of conditions related to an event. It is based on the following formula:

$P(A|B) = P(A) * P(B|A)/P(B)$

Where we are calculating the probability of class A when predictor B is already provided.

P(B) = prior probability of B

P(A) = prior probability of class A

P(B|A) = occurrence of predictor B given class A probability

This formula helps in calculating the probability of the tags in the text.

ii. Ensemble Classifier

About-

Ensemble learning helps improve machine learning results by combining several models. This approach allows the production of better predictive performance compared to a single model. Basic idea is to learn a set of classifiers and to allow them to vote.

Advantage: Improvement in predictive accuracy.

Disadvantage**:** It is difficult to understand an ensemble of classifiers.

Working**-** Ensembles overcome the problems of the Statistical Problem which arises when the hypothesis space is too large for the amount of available data. Hence, there are many hypotheses with the same accuracy on the data and the learning algorithm chooses only one of them! There is a risk that the accuracy of the chosen hypothesis is low on unseen data.

The Computational Problem arises when the learning algorithm cannot guarantees finding the best hypothesis. The Representational Problem arises when the hypothesis space does not contain any good approximation of the target classes.

In this project we have used Voting Ensemble Classifier

Voting Ensemble Classifier

A voting ensemble (or a "majority voting ensemble ") is an ensemble machine learning model that combines the predictions from multiple other models. It is a technique that may be used to improve model performance, ideally achieving better performance than any single model used in the ensemble. There are two approaches to the majority vote prediction for classification; they are hard voting and soft voting. Hard voting involves summing the predictions for each class label and predicting the class label with the most votes. Soft voting involves summing the predicted probabilities (or probability-like scores) for each class label and predicting the class label with the largest probability.

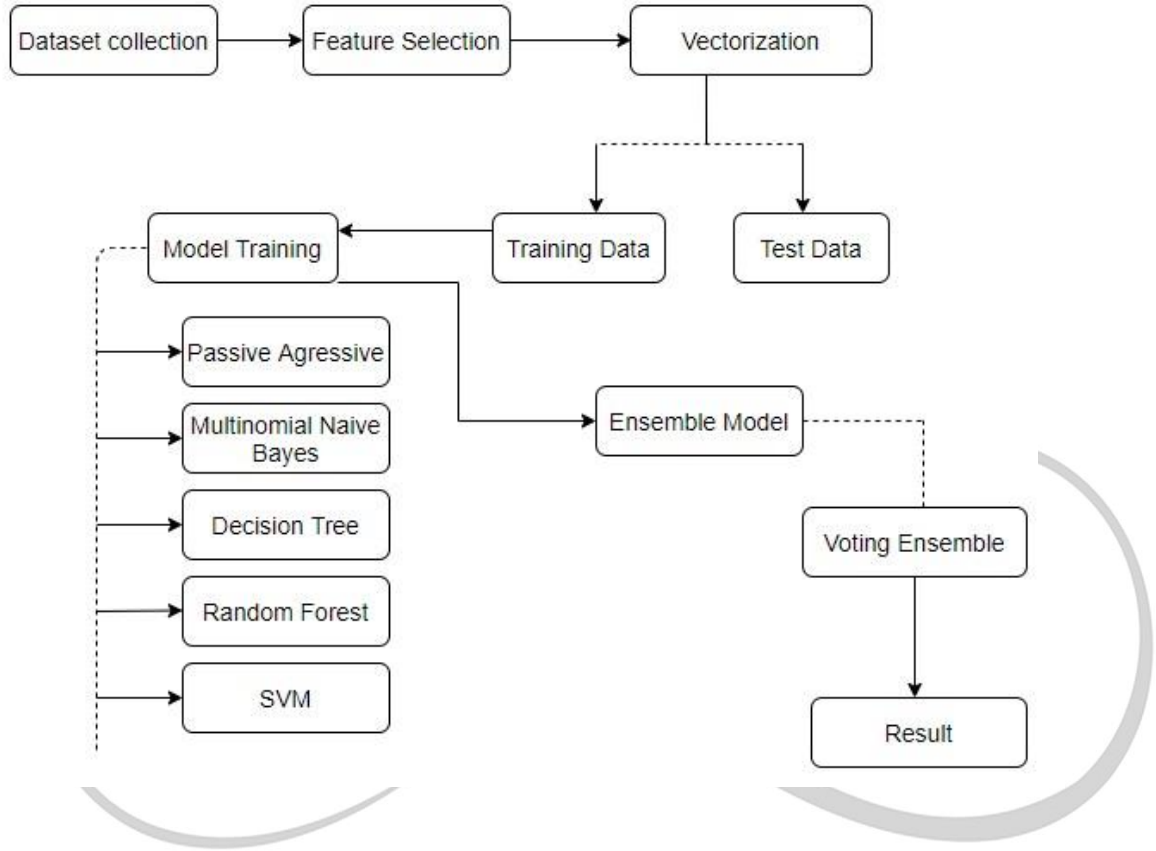
- **Hard Voting.** Predict the class with the largest sum of votes from models.
- **Soft Voting.** Predict the class with the largest summed probability from models.

A voting ensemble may be considered a meta-model, a model of models. As a meta-model, it could be used with any collection of existing trained machine learning models and the existing models do not need to be aware that they are being used in the ensemble. This means we could explore using a voting ensemble on any set or subset of fit models for your predictive modeling task. A voting ensemble is appropriate when you have two or more models that perform well on a predictive modeling task. The models used in the ensemble must mostly agree with their predictions.

Working- A voting ensemble works by combining the predictions from multiple models. It can be used for classification or regression. In the case of regression, this involves calculating the average of the predictions from the models. In the case of classification, the predictions for each label are summed and the label with the majority vote is predicted.

- **Regression Voting Ensemble**: Predictions are the average of contributing models.
- **Classification Voting Ensemble**: Predictions are the majority vote of contributing models.

Proposed Diagram:



Results

Implementation was done using the above algorithms with Vector features- Count Vectors and Tf-Idf vectorize Word level and Engram-level. Accuracy was noted for all models. We used K-fold cross validation technique to improve the effectiveness of the models.

Dataset split using K-fold cross validation This cross-validation technique was used for splitting the dataset randomly into k-folds. (k-1) folds were used for building the model while kth fold was used to check the effectiveness of the model. This was repeated until each of the k-folds served as the test set. I used 3-fold cross validation for this experiment where 70% of the data is used for training the model and remaining 30% for testing.

Classifier	Accuracy
Decision Tree Classifier ()	82.11%
RandomForestClassifier ()	90.48%
SVC ()	92.48%
Passive Aggressive Classifier	92.64%
MultinomialNB ()	82.11%
Voting Ensembling Classifier	92.37%

Table1: Accuracy table

We have made a table for classifiers and accuracy for better understanding.

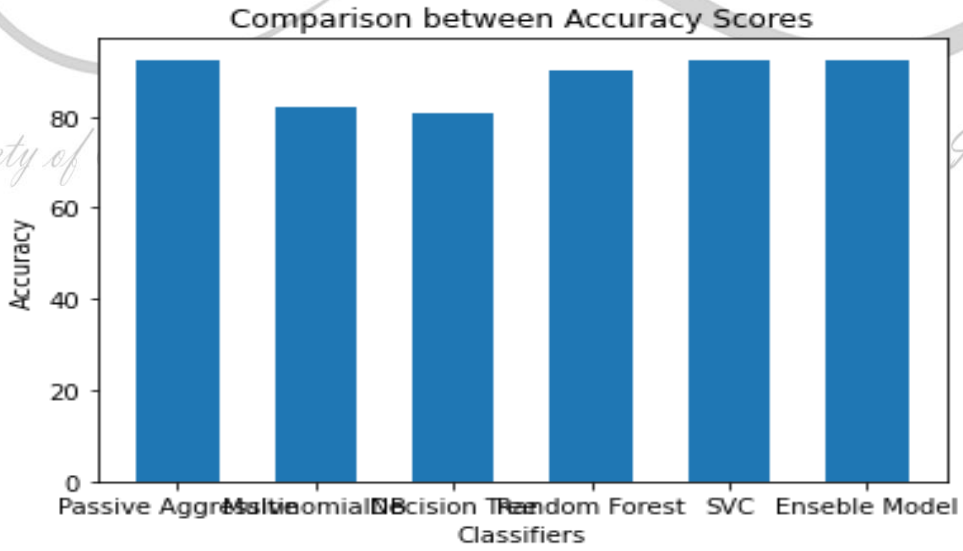


Fig 1: Comparison of the accuracy between classifiers

After that we have created this comparison graph. By this we can clearly see the differences between the accuracy scores.

Conclusion

The fake news challenge is perilous and is spreading rapidly like a wildfire as it becomes easier for information to reach the mass in various flavors. Reports have shown that, just like in the last US presidential elections, fake news can have a huge impact in politics and thereafter on the people like a domino effect. With the help of artificial intelligence, we can control and limit the spread of such misinformation more quickly and efficiently as compared to manual efforts. The work in this project proposes a stacked model which fine tunes the informational insight gained from the data at each step and then tries to make a prediction. Now a days during Covid 19 Pandemic, Many fake news regarding vaccination was running across the social platforms, by this method we can stop this and can spread positive and true news and can motivate people for mass vaccination.

Although many attempts have been made to solve the problem of fake news, any significant success is yet to be seen. With huge amounts of data collected from social media websites like Facebook, Twitter, etc., the best models improve with huge amounts of data collected from social media websites like Facebook, Twitter, etc., the best models improve.

The limitations that come packaged with this problem is that the data is erratic and this means that any type of prediction model can have anomalies and can make mistakes. For future improvements, concepts like POS tagging, word2vec and topic modelling can be utilized. These will give for future improvements, concepts like POS tagging, word2vec and topic modelling can be utilized. These will give.

FUTURE PROSPECT-

There are lots of future prospect in this project. Now a days during Covid 19 Pandemic, Many fake news regarding vaccination was running across the social platforms, by this method we can stop this and can spread positive and true news and can motivate people for mass vaccination.

Some Future prospects are-

1. Integrate this feature directly with social media to stop fake news spreading immediately. It can Detect fake news and will delete that post immediately after posting and warn the user and also remember the user as a fake news spreader.
2. We can make a front end and embed a search engine to develop a software, which can be used to detect fake news.
3. We can use as a Browser extension and detect fake news

CODE-

```
import numpy as np
import pandas as pd
import itertools
import array
import matplotlib
from matplotlib import pyplot as plt
from sklearn.model_selection import train_test_split
from sklearn.feature_extraction.text import TfidfVectorizer
from sklearn.linear_model import PassiveAggressiveClassifier
from sklearn.naive_bayes import MultinomialNB
from sklearn.ensemble import RandomForestClassifier
from sklearn.svm import SVC
from sklearn.tree import DecisionTreeClassifier
from sklearn.ensemble import VotingClassifier
from sklearn.metrics import accuracy_score, confusion_matrix
from sklearn.metrics import classification_report


# In[21]:
#Read the data
```

```
df=pd.read_csv('E:\\Major Project\\Dataset\\news.csv')
#Get shape and head
df.shape
df.head()

# In[22]:
df.shape
import seaborn as sns
sns.countplot(df['label'])
plt.show()

# In[23]:
#Get the labels
labels=df.label
labels.head()

# In[24]:
#Split the dataset
x_train,x_test,y_train,y_test=train_test_split(df['text'], labels, test_size=0.3, random_state=7)

# In[25]:
#Initialize a TfidfVectorizer
tfidf_vectorizer=TfidfVectorizer(stop_words='english', max_df=0.7)
#Fit and transform train set, transform test set
tfidf_train=tfidf_vectorizer.fit_transform(x_train)
tfidf_test=tfidf_vectorizer.transform(x_test)

# In[26]:
tfidf_train.get_shape()

# In[27]:
```

```
tfidf_test.get_shape()


# In[28]:

acc=[]


# In[29]:

#Initialize a PassiveAggressiveClassifier

pac=PassiveAggressiveClassifier(max_iter=50)

pac.fit(tfidf_train,y_train)

#Predict on the test set and calculate accuracy

y_pred=pac.predict(tfidf_test)

print(y_pred)

score=accuracy_score(y_test,y_pred)

print(f'Accuracy: {round(score*100,2)}%')

acc.append(round(score*100,2))


# In[30]:

mul=MultinomialNB()

mul.fit(tfidf_train,y_train)

#Predict on the test set and calculate accuracy

y_pred1=mul.predict(tfidf_test)

print(y_pred1)

score=accuracy_score(y_test,y_pred1)

print(f'Accuracy: {round(score*100,2)}%')

acc.append(round(score*100,2))


# In[31]:

dec=DecisionTreeClassifier()

dec.fit(tfidf_train,y_train)

#Predict on the test set and calculate accuracy
```

```
y_pred2=dec.predict(tfidf_test)
print(y_pred2)
score=accuracy_score(y_test,y_pred2)
print(f'Accuracy: {round(score*100,2)}%')
acc.append(round(score*100,2))

# In[32]:
ran=RandomForestClassifier()
ran.fit(tfidf_train,y_train)
#Predict on the test set and calculate accuracy
y_pred3=ran.predict(tfidf_test)
print(y_pred3)
score=accuracy_score(y_test,y_pred3)
print(f'Accuracy: {round(score*100,2)}%')
acc.append(round(score*100,2))

# In[33]:
svc=SVC()
svc.fit(tfidf_train,y_train)
#Predict on the test set and calculate accuracy
y_pred4=svc.predict(tfidf_test)
print(y_pred4)
score=accuracy_score(y_test,y_pred4)
print(f'Accuracy: {round(score*100,2)}%')
acc.append(round(score*100,2))

# In[34]:
#Build confusion matrix
confusion_matrix(y_test,y_pred, labels=['FAKE','REAL'])
```

```
# In[35]:
print(classification_report(y_test, y_pred))



# In[36]:
len=1901
i=0
y_final=[]
while i < len:
    a=[]
    a.append(y_pred[i])
    a.append(y_pred1[i])
    a.append(y_pred2[i])
    a.append(y_pred3[i])
    a.append(y_pred4[i])
    Rcount=a.count('REAL')
    Fcount=a.count('FAKE')
    if Rcount>Fcount:
        y_final.append('REAL')
    else:
        y_final.append('FAKE')
    i=i+1
print(y_final)

# In[37]:
score=accuracy_score(y_test,y_final)
print(score)

# In[50]:
a=["Passive Aggressive","MultinomialNB","Decision Tree","Random Forest","SVC","Enseble Model"]
b=[92.37,82.11,80.75,90.01,92.48,92.37]
```

```
plt.bar(a,b,0.6)
plt.xlabel("Classifiers")
plt.ylabel("Accuracy")
plt.title("Comparison between Accuracy Scores")
plt.show()
```

References:

1. H. Gupta, M. S. Jamal, S. Madisetty and M. S. Desarkar, "A framework for real-time spam detection in Twitter," 2018 10th International Conference on Communication Systems & Networks (COMSNETS), Bengaluru, 2018, pp. 380-383.

2. M. L. Della Vedova, E. Tacchini, S. Moret, G. Ballarin, M. DiPierro and L. de Alfaro, "Automatic Online Fake News Detection Combining Content and Social Signals," 2018 22nd Conference of Open Innovations Association (FRUCT), Jyvaskyla, 2018, pp. 272279.

3. C. Buntain and J. Golbeck, "Automatically Identifying Fake News in Popular Twitter Threads," 2017 IEEE International Conference on Smart Cloud (SmartCloud), New York, NY, 2017, pp. 208-215.

4. S. B. Parikh and P. K. Atrey, "Media-Rich Fake News Detection: A Survey," 2018 IEEE Conference on Multimedia Information Processing and Retrieval (MIPR), Miami, FL, 2018, pp. 436-441

5. Scikit-Learn- Machine Learning In Python

6. Dataset- Fake News detection William Yang Wang. " liar, liar pants on _re": A new benchmark dataset for fake news detection. arXiv preprint arXiv:1705.00648, 2017.

7. Shankar M. Patil, Dr. Praveen Kumar, —Data mining model for effective data analysis of higher education students using MapReduce‖ IJERMT, April 2017 (Volume-6, Issue-4).

8. Aayush Ranjan, — Fake News Detection Using Machine Learning‖, Department Of Computer Science & Engineering Delhi

Technological University, July 2018.

9. Patil S.M., Malik A.K. (2019) Correlation Based Real-Time Data Analysis of Graduate Students Behaviour. In: Santosh K., Hegadi R. (eds) Recent Trends in Image Processing and Pattern Recognition.
RTIP2R 2018. Communications in Computer and Information Science, vol 1037. Springer, Singapore.

10. Badreesh Shetty, "Natural Language Processing (NLP) for machine learning‖ at towards data science, Medium.

11. Ultimate guide to deal with Text Data (using Python) – for Data Scientists and Engineers by Shubham Jain, February 27, 2018.

12. NLTK 3.5b1 documentation, Nltk generate n gram [21] Ultimate.

ORIGINAL PYTHON FILE AND CODE with OUTPUTS ARE AS FOLLOWS-

In [20]:
```
import numpy as np
import pandas as pd
import itertools
import array
import matplotlib
from matplotlib import pyplot as plt
from sklearn.model_selection import train_test_split
from sklearn.feature_extraction.text import TfidfVectorizer

from sklearn.linear_model import PassiveAggressiveClassifier
from sklearn.naive_bayes import MultinomialNB
from sklearn.ensemble import RandomForestClassifier
from sklearn.svm import SVC
from sklearn.tree import DecisionTreeClassifier
from sklearn.ensemble import VotingClassifier

from sklearn.metrics import accuracy_score, confusion_matrix
from sklearn.metrics import classification_report
```

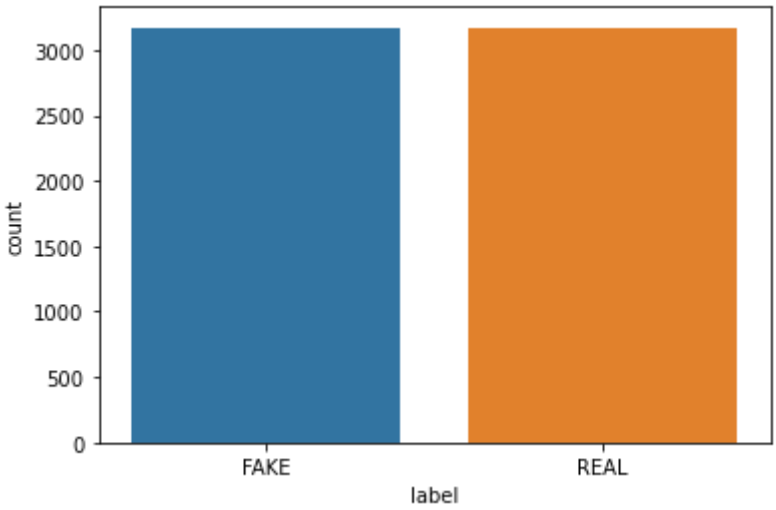
In [21]:
```
#Read the data
df=pd.read_csv('E:\\Major Project\\Dataset\\news.csv')
#Get shape and head
df.shape
df.head()
```

Out[21]:

	Unnamed: 0	title	text	label
0	8476	You Can Smell Hillary's Fear	Daniel Greenfield, a Shillman Journalism Fello...	FAKE
1	10294	Watch The Exact Moment Paul Ryan Committed Pol...	Google Pinterest Digg Linkedin Reddit Stumbleu...	FAKE
2	3608	Kerry to go to Paris in gesture of sympathy	U.S. Secretary of State John F. Kerry said Mon...	REAL
3	10142	Bernie supporters on Twitter erupt in anger ag...	— Kaydee King (@KaydeeKing) November 9, 2016 T...	FAKE
4	875	The Battle of New York: Why This Primary Matters	It's primary day in New York and front-runners...	REAL

In [22]:
```
df.shape
import seaborn as sns
sns.countplot(df['label'])
plt.show()
```

```
C:\ProgramData\Anaconda3\lib\site-packages\seaborn\_decorators.py:36: FutureWarning:
Pass the following variable as a keyword arg: x. From version 0.12, the only valid p
ositional argument will be `data`, and passing other arguments without an explicit k
eyword will result in an error or misinterpretation.
  warnings.warn(
```



In [23]:
```
#Get the labels
labels=df.label
labels.head()
```

Out[23]:
```
0    FAKE
1    FAKE
2    REAL
3    FAKE
4    REAL
Name: label, dtype: object
```

In [24]:
```
#Split the dataset
x_train,x_test,y_train,y_test=train_test_split(df['text'], labels, test_size=0.3, ra
```

In [25]:
```
#Initialize a TfidfVectorizer
tfidf_vectorizer=TfidfVectorizer(stop_words='english', max_df=0.7)

#Fit and transform train set, transform test set
tfidf_train=tfidf_vectorizer.fit_transform(x_train)
tfidf_test=tfidf_vectorizer.transform(x_test)
```

In [26]:
```
tfidf_train.get_shape()
```

Out[26]:
```
(4434, 57794)
```

In [27]:
```
tfidf_test.get_shape()
```

Out[27]:
```
(1901, 57794)
```

In [28]:
```
acc=[]
```

In [29]:
```
#Initialize a PassiveAggressiveClassifier

pac=PassiveAggressiveClassifier(max_iter=50)
pac.fit(tfidf_train,y_train)

#Predict on the test set and calculate accuracy

y_pred=pac.predict(tfidf_test)
```

```
print(y_pred)
score=accuracy_score(y_test,y_pred)
print(f'Accuracy: {round(score*100,2)}%')
acc.append(round(score*100,2))
```

```
['REAL' 'FAKE' 'REAL' ... 'FAKE' 'FAKE' 'REAL']
Accuracy: 92.37%
```

In [30]:

```
mul=MultinomialNB()
mul.fit(tfidf_train,y_train)

#Predict on the test set and calculate accuracy

y_pred1=mul.predict(tfidf_test)
print(y_pred1)
score=accuracy_score(y_test,y_pred1)
print(f'Accuracy: {round(score*100,2)}%')
acc.append(round(score*100,2))
```

```
['REAL' 'FAKE' 'REAL' ... 'FAKE' 'REAL' 'REAL']
Accuracy: 82.11%
```

In [31]:

```
dec=DecisionTreeClassifier()
dec.fit(tfidf_train,y_train)

#Predict on the test set and calculate accuracy

y_pred2=dec.predict(tfidf_test)
print(y_pred2)
score=accuracy_score(y_test,y_pred2)
print(f'Accuracy: {round(score*100,2)}%')
acc.append(round(score*100,2))
```

```
['REAL' 'FAKE' 'REAL' ... 'FAKE' 'FAKE' 'REAL']
Accuracy: 80.75%
```

In [32]:

```
ran=RandomForestClassifier()
ran.fit(tfidf_train,y_train)

#Predict on the test set and calculate accuracy

y_pred3=ran.predict(tfidf_test)
print(y_pred3)
score=accuracy_score(y_test,y_pred3)
print(f'Accuracy: {round(score*100,2)}%')
acc.append(round(score*100,2))
```

```
['REAL' 'FAKE' 'REAL' ... 'FAKE' 'FAKE' 'REAL']
Accuracy: 90.01%
```

In [33]:

```
svc=SVC()
svc.fit(tfidf_train,y_train)

#Predict on the test set and calculate accuracy

y_pred4=svc.predict(tfidf_test)
print(y_pred4)
score=accuracy_score(y_test,y_pred4)
print(f'Accuracy: {round(score*100,2)}%')
acc.append(round(score*100,2))
```

```
['REAL' 'FAKE' 'REAL' ... 'FAKE' 'FAKE' 'REAL']
Accuracy: 92.48%
```

In [34]:
```
#Build confusion matrix

confusion_matrix(y_test,y_pred, labels=['FAKE','REAL'])
```

Out[34]:
```
array([[900,  74],
       [ 71, 856]], dtype=int64)
```

In [35]:
```
print(classification_report(y_test, y_pred))
```

```
              precision    recall  f1-score   support

        FAKE       0.93      0.92      0.93       974
        REAL       0.92      0.92      0.92       927

    accuracy                           0.92      1901
   macro avg       0.92      0.92      0.92      1901
weighted avg       0.92      0.92      0.92      1901
```

In [36]:
```
len=1901
i=0
y_final=[]
while i < len:
    a=[]
    a.append(y_pred[i])
    a.append(y_pred1[i])
    a.append(y_pred2[i])
    a.append(y_pred3[i])
    a.append(y_pred4[i])
    Rcount=a.count('REAL')
    Fcount=a.count('FAKE')
    if Rcount>Fcount:
        y_final.append('REAL')
    else:
        y_final.append('FAKE')
    i=i+1
print(y_final)
```

```
['REAL', 'FAKE', 'REAL', 'REAL', 'REAL', 'REAL', 'REAL', 'REAL', 'REAL', 'FAKE', 'FA
KE', 'REAL', 'REAL', 'REAL', 'REAL', 'FAKE', 'REAL', 'REAL', 'REAL', 'FAKE', 'FAKE',
'REAL', 'REAL', 'FAKE', 'REAL', 'REAL', 'REAL', 'FAKE', 'FAKE', 'REAL', 'REAL', 'REA
L', 'REAL', 'REAL', 'REAL', 'FAKE', 'REAL', 'REAL', 'REAL', 'FAKE', 'REAL', 'REAL',
'REAL', 'REAL', 'FAKE', 'FAKE', 'REAL', 'REAL', 'FAKE', 'FAKE', 'REAL', 'REAL', 'REA
L', 'REAL', 'FAKE', 'REAL', 'FAKE', 'REAL', 'FAKE', 'FAKE', 'FAKE', 'REAL', 'FAKE',
'REAL', 'FAKE', 'REAL', 'FAKE', 'REAL', 'FAKE', 'REAL', 'REAL', 'REAL', 'REAL', 'REA
L', 'FAKE', 'FAKE', 'FAKE', 'FAKE', 'REAL', 'FAKE', 'FAKE', 'FAKE', 'REAL', 'FAKE',
'REAL', 'FAKE', 'REAL', 'FAKE', 'FAKE', 'FAKE', 'FAKE', 'REAL', 'REAL', 'FAKE', 'REA
L', 'FAKE', 'FAKE', 'FAKE', 'REAL', 'FAKE', 'REAL', 'FAKE', 'REAL', 'FAKE', 'REAL',
'REAL', 'REAL', 'REAL', 'REAL', 'REAL', 'FAKE', 'REAL', 'REAL', 'FAKE', 'FAKE', 'REA
L', 'REAL', 'REAL', 'FAKE', 'FAKE', 'FAKE', 'FAKE', 'REAL', 'REAL', 'FAKE', 'REAL',
'REAL', 'REAL', 'REAL', 'REAL', 'FAKE', 'REAL', 'REAL', 'REAL', 'REAL', 'REAL', 'REA
L', 'FAKE', 'FAKE', 'REAL', 'REAL', 'REAL', 'REAL', 'FAKE', 'REAL', 'FAKE', 'FAKE',
'REAL', 'FAKE', 'FAKE', 'REAL', 'FAKE', 'REAL', 'REAL', 'FAKE', 'REAL', 'FAKE', 'REA
L', 'REAL', 'REAL', 'REAL', 'REAL', 'REAL', 'FAKE', 'REAL', 'REAL', 'FAKE', 'FAKE',
'REAL', 'REAL', 'REAL', 'FAKE', 'REAL', 'FAKE', 'REAL', 'FAKE', 'FAKE', 'FAKE', 'FAK
E', 'FAKE', 'REAL', 'FAKE', 'FAKE', 'REAL', 'FAKE', 'REAL', 'FAKE', 'FAKE', 'REAL',
'REAL', 'REAL', 'REAL', 'REAL', 'REAL', 'REAL', 'FAKE', 'REAL', 'REAL', 'FAKE', 'FAK
E', 'FAKE', 'FAKE', 'FAKE', 'REAL', 'FAKE', 'FAKE', 'FAKE', 'FAKE', 'REAL', 'REAL',
'REAL', 'REAL', 'FAKE', 'FAKE', 'FAKE', 'FAKE', 'REAL', 'REAL', 'REAL', 'FAKE', 'FAK
E', 'FAKE', 'FAKE', 'FAKE', 'FAKE', 'REAL', 'REAL', 'REAL', 'REAL', 'FAKE', 'FAKE',
'FAKE', 'FAKE', 'FAKE', 'REAL', 'REAL', 'FAKE', 'FAKE', 'FAKE', 'FAKE', 'REAL', 'REA
L', 'REAL', 'FAKE', 'FAKE', 'FAKE', 'REAL', 'FAKE', 'REAL', 'REAL', 'REAL', 'REAL',
'REAL', 'REAL', 'FAKE', 'FAKE', 'FAKE', 'REAL', 'FAKE', 'FAKE', 'REAL', 'FAKE', 'FAK
E', 'REAL', 'REAL', 'FAKE', 'FAKE', 'FAKE', 'FAKE', 'FAKE', 'REAL', 'FAKE', 'REAL',
```

'FAKE', 'FAKE', 'REAL', 'FAKE', 'FAKE', 'REAL', 'FAKE', 'FAKE', 'REAL', 'REAL', 'FAK
E', 'REAL', 'REAL', 'FAKE', 'FAKE', 'REAL', 'REAL', 'REAL', 'REAL', 'FAKE', 'REAL',
'REAL', 'REAL', 'REAL', 'REAL', 'REAL', 'REAL', 'FAKE', 'REAL', 'FAKE', 'FAKE', 'FAK
E', 'REAL', 'FAKE', 'FAKE', 'REAL', 'FAKE', 'FAKE', 'FAKE', 'FAKE', 'REAL', 'FAKE',
'FAKE', 'FAKE', 'REAL', 'REAL', 'FAKE', 'FAKE', 'FAKE', 'REAL', 'FAKE', 'FAKE', 'FAK
E', 'REAL', 'REAL', 'FAKE', 'FAKE', 'REAL', 'REAL', 'REAL', 'REAL', 'REAL', 'REAL',
'REAL', 'FAKE', 'FAKE', 'FAKE', 'REAL', 'REAL', 'FAKE', 'FAKE', 'REAL', 'REAL', 'REA
L', 'FAKE', 'REAL', 'REAL', 'FAKE', 'REAL', 'FAKE', 'REAL', 'REAL', 'REAL', 'REAL',
'REAL', 'REAL', 'FAKE', 'REAL', 'FAKE', 'REAL', 'REAL', 'REAL', 'FAKE', 'FAKE', 'FAK
E', 'FAKE', 'REAL', 'REAL', 'REAL', 'REAL', 'FAKE', 'FAKE', 'REAL', 'FAKE', 'REAL',
'FAKE', 'FAKE', 'FAKE', 'REAL', 'FAKE', 'FAKE', 'REAL', 'REAL', 'FAKE', 'REAL', 'REA
L', 'FAKE', 'FAKE', 'FAKE', 'FAKE', 'FAKE', 'FAKE', 'FAKE', 'REAL', 'REAL', 'REAL',
'FAKE', 'FAKE', 'FAKE', 'FAKE', 'REAL', 'REAL', 'FAKE', 'FAKE', 'REAL', 'REAL', 'FAK
E', 'FAKE', 'FAKE', 'REAL', 'FAKE', 'FAKE', 'FAKE', 'FAKE', 'FAKE', 'FAKE', 'REAL',
'REAL', 'REAL', 'REAL', 'REAL', 'FAKE', 'REAL', 'FAKE', 'REAL', 'FAKE', 'FAKE', 'FAK
E', 'FAKE', 'FAKE', 'FAKE', 'REAL', 'REAL', 'FAKE', 'REAL', 'FAKE', 'FAKE', 'FAKE',
'REAL', 'REAL', 'REAL', 'FAKE', 'FAKE', 'REAL', 'FAKE', 'FAKE', 'FAKE', 'REAL', 'REA
L', 'FAKE', 'REAL', 'FAKE', 'REAL', 'REAL', 'FAKE', 'FAKE', 'FAKE', 'REAL', 'FAKE',
'FAKE', 'REAL', 'FAKE', 'FAKE', 'REAL', 'REAL', 'REAL', 'REAL', 'REAL', 'REAL', 'REA
L', 'REAL', 'FAKE', 'REAL', 'FAKE', 'FAKE', 'FAKE', 'FAKE', 'REAL', 'REAL', 'REAL',
'REAL', 'REAL', 'REAL', 'FAKE', 'REAL', 'FAKE', 'REAL', 'REAL', 'FAKE', 'FAKE', 'REA
L', 'REAL', 'FAKE', 'REAL', 'REAL', 'REAL', 'REAL', 'REAL', 'REAL', 'FAKE', 'REAL',
'FAKE', 'FAKE', 'FAKE', 'FAKE', 'FAKE', 'FAKE', 'REAL', 'FAKE', 'FAKE', 'REAL', 'FAK
E', 'FAKE', 'FAKE', 'FAKE', 'FAKE', 'REAL', 'FAKE', 'FAKE', 'FAKE', 'REAL', 'FAKE',
'FAKE', 'FAKE', 'REAL', 'REAL', 'FAKE', 'FAKE', 'REAL', 'REAL', 'FAKE', 'REAL', 'REA
L', 'FAKE', 'REAL', 'FAKE', 'REAL', 'FAKE', 'REAL', 'REAL', 'REAL', 'FAKE', 'REAL',
'FAKE', 'FAKE', 'REAL', 'REAL', 'FAKE', 'REAL', 'REAL', 'REAL', 'REAL', 'REAL', 'FAK
E', 'REAL', 'FAKE', 'REAL', 'FAKE', 'FAKE', 'FAKE', 'REAL', 'FAKE', 'REAL', 'REAL',
'REAL', 'REAL', 'REAL', 'REAL', 'REAL', 'REAL', 'FAKE', 'REAL', 'REAL', 'REAL', 'REA
L', 'REAL', 'REAL', 'FAKE', 'FAKE', 'FAKE', 'REAL', 'FAKE', 'REAL', 'FAKE', 'REAL',
'REAL', 'FAKE', 'REAL', 'REAL', 'REAL', 'REAL', 'REAL', 'REAL', 'REAL', 'FAKE', 'FAK
E', 'REAL', 'REAL', 'REAL', 'REAL', 'FAKE', 'FAKE', 'FAKE', 'FAKE', 'REAL', 'FAKE',
'REAL', 'FAKE', 'FAKE', 'FAKE', 'REAL', 'FAKE', 'FAKE', 'REAL', 'FAKE', 'REAL', 'REA
L', 'FAKE', 'REAL', 'FAKE', 'REAL', 'REAL', 'REAL', 'REAL', 'REAL', 'REAL', 'FAKE',
'REAL', 'REAL', 'FAKE', 'REAL', 'REAL', 'FAKE', 'REAL', 'REAL', 'REAL', 'FAKE', 'REA
L', 'REAL', 'REAL', 'FAKE', 'REAL', 'REAL', 'REAL', 'FAKE', 'REAL', 'REAL', 'REAL',
'FAKE', 'REAL', 'REAL', 'REAL', 'REAL', 'REAL', 'FAKE', 'REAL', 'REAL', 'REAL', 'REA
L', 'FAKE', 'REAL', 'REAL', 'FAKE', 'REAL', 'REAL', 'REAL', 'REAL', 'FAKE', 'REAL',
'REAL', 'REAL', 'FAKE', 'REAL', 'FAKE', 'FAKE', 'REAL', 'REAL', 'REAL', 'REAL', 'REA
L', 'FAKE', 'REAL', 'FAKE', 'REAL', 'REAL', 'REAL', 'FAKE', 'FAKE', 'REAL', 'REAL',
'REAL', 'FAKE', 'FAKE', 'FAKE', 'FAKE', 'FAKE', 'FAKE', 'FAKE', 'FAKE', 'REAL', 'FAK
E', 'FAKE', 'REAL', 'REAL', 'FAKE', 'REAL', 'FAKE', 'FAKE', 'REAL', 'REAL', 'FAKE',
'REAL', 'FAKE', 'FAKE', 'FAKE', 'REAL', 'FAKE', 'FAKE', 'REAL', 'FAKE', 'FAKE', 'FAK
E', 'REAL', 'REAL', 'REAL', 'FAKE', 'REAL', 'REAL', 'FAKE', 'FAKE', 'FAKE', 'FAKE',
'FAKE', 'REAL', 'REAL', 'FAKE', 'FAKE', 'FAKE', 'FAKE', 'REAL', 'REAL', 'FAKE', 'REA
L', 'REAL', 'REAL', 'REAL', 'FAKE', 'FAKE', 'REAL', 'REAL', 'REAL', 'FAKE', 'FAKE',
'FAKE', 'REAL', 'REAL', 'FAKE', 'FAKE', 'FAKE', 'FAKE', 'REAL', 'FAKE', 'REAL', 'FAK
E', 'FAKE', 'FAKE', 'REAL', 'REAL', 'FAKE', 'FAKE', 'FAKE', 'REAL', 'FAKE', 'FAKE',
'REAL', 'REAL', 'REAL', 'REAL', 'REAL', 'FAKE', 'FAKE', 'FAKE', 'FAKE', 'FAKE', 'REA
L', 'FAKE', 'REAL', 'REAL', 'REAL', 'REAL', 'REAL', 'REAL', 'FAKE', 'REAL', 'FAKE',
'FAKE', 'REAL', 'FAKE', 'FAKE', 'REAL', 'REAL', 'REAL', 'REAL', 'REAL', 'FAKE', 'FAK
E', 'REAL', 'FAKE', 'REAL', 'FAKE', 'REAL', 'FAKE', 'FAKE', 'REAL', 'REAL', 'REAL',
'REAL', 'REAL', 'FAKE', 'REAL', 'FAKE', 'REAL', 'FAKE', 'REAL', 'REAL', 'FAKE', 'REA
L', 'FAKE', 'REAL', 'FAKE', 'FAKE', 'FAKE', 'FAKE', 'FAKE', 'REAL', 'FAKE', 'FAKE',
'REAL', 'FAKE', 'FAKE', 'FAKE', 'FAKE', 'REAL', 'FAKE', 'REAL', 'FAKE', 'REAL', 'FAK
E', 'REAL', 'REAL', 'FAKE', 'REAL', 'FAKE', 'FAKE', 'FAKE', 'FAKE', 'REAL', 'FAKE',
'FAKE', 'FAKE', 'REAL', 'REAL', 'REAL', 'FAKE', 'REAL', 'FAKE', 'REAL', 'REAL', 'REA
L', 'FAKE', 'REAL', 'FAKE', 'REAL', 'REAL', 'FAKE', 'REAL', 'REAL', 'FAKE', 'REAL',
'REAL', 'REAL', 'FAKE', 'FAKE', 'REAL', 'REAL', 'FAKE', 'REAL', 'FAKE', 'FAKE', 'FAK
E', 'FAKE', 'REAL', 'REAL', 'FAKE', 'REAL', 'REAL', 'FAKE', 'FAKE', 'FAKE', 'FAKE',
'FAKE', 'FAKE', 'REAL', 'REAL', 'FAKE', 'FAKE', 'REAL', 'REAL', 'FAKE', 'REAL', 'FAK
E', 'FAKE', 'REAL', 'FAKE', 'REAL', 'REAL', 'REAL', 'FAKE', 'REAL', 'REAL', 'FAKE',
'REAL', 'FAKE', 'FAKE', 'FAKE', 'REAL', 'FAKE', 'FAKE', 'FAKE', 'REAL', 'REAL', 'REA
L', 'REAL', 'REAL', 'REAL', 'REAL', 'REAL', 'FAKE', 'REAL', 'REAL', 'REAL', 'REAL',
'FAKE', 'REAL', 'FAKE', 'FAKE', 'REAL', 'REAL', 'FAKE', 'FAKE', 'FAKE', 'REAL', 'FAK
E', 'REAL', 'REAL', 'FAKE', 'FAKE', 'FAKE', 'FAKE', 'FAKE', 'REAL', 'REAL', 'FAKE',
'REAL', 'FAKE', 'REAL', 'FAKE', 'FAKE', 'FAKE', 'REAL', 'FAKE', 'REAL', 'FAKE', 'FAK
E', 'FAKE', 'REAL', 'FAKE', 'FAKE', 'REAL', 'REAL', 'REAL', 'REAL', 'REAL', 'FAKE',
'REAL', 'FAKE', 'FAKE', 'REAL', 'REAL', 'REAL', 'FAKE', 'REAL', 'REAL', 'FAKE', 'FAK

```
E', 'FAKE', 'FAKE', 'REAL', 'FAKE', 'REAL', 'FAKE', 'FAKE', 'REAL', 'FAKE', 'REAL',
'FAKE', 'FAKE', 'FAKE', 'REAL', 'FAKE', 'FAKE', 'REAL', 'FAKE', 'REAL', 'REAL', 'FAK
E', 'REAL', 'REAL', 'REAL', 'FAKE', 'REAL', 'REAL', 'FAKE', 'FAKE', 'REAL', 'REAL',
'REAL', 'REAL', 'REAL', 'REAL', 'REAL', 'REAL', 'FAKE', 'REAL', 'FAKE', 'FAKE', 'REA
L', 'FAKE', 'REAL', 'FAKE', 'FAKE', 'REAL', 'REAL', 'FAKE', 'REAL', 'FAKE', 'REAL',
'REAL', 'REAL', 'FAKE', 'REAL', 'REAL', 'FAKE', 'FAKE', 'FAKE', 'FAKE', 'REAL', 'FAK
E', 'REAL', 'FAKE', 'FAKE', 'FAKE', 'FAKE', 'FAKE', 'FAKE', 'FAKE', 'REAL', 'FAKE',
'REAL', 'FAKE', 'FAKE', 'FAKE', 'FAKE', 'FAKE', 'REAL', 'REAL', 'FAKE', 'REAL', 'REA
L', 'FAKE', 'REAL', 'FAKE', 'FAKE', 'FAKE', 'REAL', 'REAL', 'FAKE', 'FAKE', 'FAKE',
'REAL', 'FAKE', 'REAL', 'FAKE', 'REAL', 'REAL', 'REAL', 'FAKE', 'REAL', 'REAL', 'REA
L', 'FAKE', 'FAKE', 'REAL', 'FAKE', 'FAKE', 'FAKE', 'FAKE', 'REAL', 'REAL', 'FAKE',
'FAKE', 'REAL', 'REAL', 'FAKE', 'REAL', 'REAL', 'REAL', 'FAKE', 'FAKE', 'REAL', 'FAK
E', 'FAKE', 'FAKE', 'REAL', 'REAL', 'FAKE', 'FAKE', 'REAL', 'FAKE', 'REAL', 'FAKE',
'REAL', 'REAL', 'FAKE', 'FAKE', 'FAKE', 'FAKE', 'REAL', 'FAKE', 'REAL', 'REAL', 'FAK
E', 'FAKE', 'FAKE', 'REAL', 'REAL', 'FAKE', 'FAKE', 'FAKE', 'REAL', 'REAL', 'FAKE',
'FAKE', 'REAL', 'REAL', 'REAL', 'REAL', 'REAL', 'FAKE', 'REAL', 'FAKE', 'FAKE', 'FAK
E', 'FAKE', 'REAL', 'FAKE', 'REAL', 'FAKE', 'FAKE', 'FAKE', 'REAL', 'FAKE', 'FAKE',
'FAKE', 'REAL', 'REAL', 'FAKE', 'REAL', 'FAKE', 'FAKE', 'REAL', 'FAKE', 'FAKE', 'REA
L', 'FAKE', 'FAKE', 'FAKE', 'REAL', 'FAKE', 'FAKE', 'FAKE', 'REAL', 'REAL', 'FAKE',
'FAKE', 'FAKE', 'REAL', 'REAL', 'REAL', 'REAL', 'REAL', 'FAKE', 'FAKE', 'REAL', 'FAK
E', 'FAKE', 'FAKE', 'FAKE', 'FAKE', 'REAL', 'FAKE', 'FAKE', 'REAL', 'FAKE', 'FAKE',
'FAKE', 'REAL', 'REAL', 'REAL', 'REAL', 'REAL', 'FAKE', 'FAKE', 'FAKE', 'FAKE', 'FAK
E', 'FAKE', 'FAKE', 'REAL', 'REAL', 'REAL', 'FAKE', 'REAL', 'REAL', 'FAKE', 'FAKE',
'FAKE', 'FAKE', 'REAL', 'REAL', 'FAKE', 'FAKE', 'REAL', 'FAKE', 'FAKE', 'REAL', 'FAK
E', 'FAKE', 'FAKE', 'FAKE', 'REAL', 'REAL', 'FAKE', 'REAL', 'REAL', 'FAKE', 'REAL',
'FAKE', 'FAKE', 'REAL', 'REAL', 'REAL', 'FAKE', 'REAL', 'FAKE', 'FAKE', 'FAKE', 'FAK
E', 'REAL', 'REAL', 'FAKE', 'FAKE', 'FAKE', 'FAKE', 'FAKE', 'REAL', 'REAL', 'FAKE',
'FAKE', 'REAL', 'FAKE', 'FAKE', 'REAL', 'FAKE', 'REAL', 'FAKE', 'REAL', 'FAKE', 'REA
L', 'FAKE', 'FAKE', 'REAL', 'REAL', 'REAL', 'REAL', 'REAL', 'REAL', 'REAL', 'REAL',
'FAKE', 'REAL', 'REAL', 'REAL', 'FAKE', 'FAKE', 'FAKE', 'REAL', 'REAL', 'REAL', 'FAK
E', 'FAKE', 'FAKE', 'REAL', 'REAL', 'REAL', 'FAKE', 'REAL', 'FAKE', 'REAL', 'FAKE',
'REAL', 'FAKE', 'FAKE', 'REAL', 'FAKE', 'REAL', 'REAL', 'REAL', 'FAKE', 'REAL', 'FAK
E', 'FAKE', 'REAL', 'FAKE', 'FAKE', 'FAKE', 'FAKE', 'REAL', 'FAKE', 'REAL', 'REAL',
'FAKE', 'REAL', 'FAKE', 'REAL', 'REAL', 'FAKE', 'FAKE', 'REAL', 'FAKE', 'FAKE', 'REA
L', 'REAL', 'REAL', 'FAKE', 'FAKE', 'REAL', 'REAL', 'FAKE', 'FAKE', 'FAKE', 'FAKE',
'REAL', 'FAKE', 'REAL', 'REAL', 'FAKE', 'FAKE', 'REAL', 'REAL', 'FAKE', 'REAL', 'REA
L', 'REAL', 'FAKE', 'FAKE', 'FAKE', 'REAL', 'REAL', 'REAL', 'REAL', 'REAL', 'FAKE',
'FAKE', 'REAL', 'FAKE', 'REAL', 'FAKE', 'REAL', 'REAL', 'REAL', 'FAKE', 'REAL', 'FAK
E', 'FAKE', 'FAKE', 'REAL', 'FAKE', 'FAKE', 'REAL', 'REAL', 'FAKE', 'FAKE', 'REAL',
'REAL', 'FAKE', 'FAKE', 'FAKE', 'FAKE', 'REAL', 'REAL', 'REAL', 'FAKE', 'REAL', 'REA
L', 'FAKE', 'REAL', 'FAKE', 'FAKE', 'FAKE', 'FAKE', 'REAL', 'REAL', 'FAKE', 'FAKE',
'FAKE', 'FAKE', 'FAKE', 'FAKE', 'FAKE', 'FAKE', 'FAKE', 'REAL', 'REAL', 'FAKE', 'REA
L', 'REAL', 'FAKE', 'REAL', 'FAKE', 'REAL', 'REAL', 'FAKE', 'REAL', 'REAL', 'REAL',
'FAKE', 'FAKE', 'FAKE', 'FAKE', 'FAKE', 'REAL', 'REAL', 'FAKE', 'FAKE', 'FAKE', 'REA
L', 'FAKE', 'REAL', 'REAL', 'REAL', 'REAL', 'FAKE', 'FAKE', 'REAL', 'REAL', 'FAKE',
'REAL', 'FAKE', 'REAL', 'FAKE', 'FAKE', 'FAKE', 'FAKE', 'REAL', 'REAL', 'REAL', 'FAK
E', 'REAL', 'FAKE', 'REAL', 'REAL', 'REAL', 'FAKE', 'FAKE', 'FAKE', 'REAL', 'REAL',
'REAL', 'REAL', 'REAL', 'REAL', 'REAL', 'FAKE', 'FAKE', 'FAKE', 'REAL', 'REAL', 'FAK
E', 'REAL', 'REAL', 'REAL', 'FAKE', 'FAKE', 'REAL', 'REAL', 'FAKE', 'FAKE', 'REAL',
'REAL', 'REAL', 'REAL', 'FAKE', 'FAKE', 'FAKE', 'FAKE', 'REAL', 'FAKE', 'REAL', 'REA
L', 'FAKE', 'FAKE', 'FAKE', 'FAKE', 'FAKE', 'REAL', 'REAL', 'REAL', 'REAL', 'FAKE',
'FAKE', 'REAL', 'FAKE', 'FAKE', 'FAKE', 'FAKE', 'FAKE', 'FAKE', 'REAL', 'FAKE', 'FAK
E', 'REAL', 'REAL', 'FAKE', 'FAKE', 'REAL', 'REAL', 'FAKE', 'FAKE', 'FAKE', 'REAL',
'REAL', 'REAL', 'FAKE', 'FAKE', 'FAKE', 'FAKE', 'REAL', 'FAKE', 'FAKE', 'FAKE', 'FAK
E', 'FAKE', 'FAKE', 'FAKE', 'REAL', 'FAKE', 'FAKE', 'REAL', 'FAKE', 'REAL', 'FAKE',
'FAKE', 'FAKE', 'REAL', 'REAL', 'REAL', 'FAKE', 'REAL', 'FAKE', 'FAKE', 'REAL', 'REA
L', 'REAL', 'REAL', 'REAL', 'FAKE', 'REAL', 'REAL', 'REAL', 'FAKE', 'FAKE', 'FAKE',
'FAKE', 'FAKE', 'REAL', 'FAKE', 'FAKE', 'REAL', 'FAKE', 'REAL', 'FAKE', 'REAL', 'FAK
E', 'FAKE', 'REAL', 'FAKE', 'REAL', 'FAKE', 'FAKE', 'FAKE', 'FAKE', 'REAL', 'FAKE',
'FAKE', 'REAL', 'REAL', 'FAKE', 'REAL', 'FAKE', 'FAKE', 'FAKE', 'REAL', 'REAL', 'REA
L', 'REAL', 'FAKE', 'FAKE', 'FAKE', 'FAKE', 'FAKE', 'FAKE', 'FAKE', 'REAL', 'FAKE',
'FAKE', 'REAL', 'FAKE', 'REAL', 'REAL', 'FAKE', 'FAKE', 'FAKE', 'REAL', 'FAKE', 'FAK
E', 'REAL', 'FAKE', 'FAKE', 'REAL', 'FAKE', 'FAKE', 'REAL', 'REAL', 'FAKE', 'REAL',
'FAKE', 'FAKE', 'REAL', 'FAKE', 'REAL', 'FAKE', 'FAKE', 'FAKE', 'FAKE', 'FAKE', 'REA
L', 'REAL', 'REAL', 'REAL', 'FAKE', 'REAL', 'REAL', 'FAKE', 'REAL', 'FAKE', 'FAKE',
'REAL', 'FAKE', 'FAKE', 'FAKE', 'FAKE', 'FAKE', 'REAL', 'FAKE', 'REAL', 'REAL', 'REA
L', 'FAKE', 'REAL', 'REAL', 'FAKE', 'FAKE', 'REAL', 'REAL', 'REAL', 'FAKE', 'REAL',
'FAKE', 'FAKE', 'FAKE', 'FAKE', 'REAL', 'FAKE', 'FAKE', 'REAL', 'REAL', 'FAKE', 'REA
L', 'FAKE', 'FAKE', 'REAL', 'FAKE', 'REAL', 'REAL', 'FAKE', 'REAL', 'FAKE', 'FAKE',
```

```
'FAKE', 'REAL', 'REAL', 'FAKE', 'REAL', 'FAKE', 'REAL', 'REAL', 'FAKE', 'FAKE', 'REA
L', 'FAKE', 'FAKE', 'FAKE', 'REAL', 'FAKE', 'REAL', 'FAKE', 'FAKE', 'REAL', 'FAKE',
'FAKE', 'REAL', 'FAKE', 'REAL', 'REAL', 'REAL', 'REAL', 'REAL', 'REAL', 'FAKE', 'REA
L', 'FAKE', 'FAKE', 'FAKE', 'FAKE', 'FAKE', 'FAKE', 'REAL', 'FAKE', 'REAL', 'REAL',
'REAL', 'REAL', 'REAL', 'REAL', 'FAKE', 'REAL', 'REAL', 'FAKE', 'REAL', 'REAL', 'FAK
E', 'REAL', 'FAKE', 'FAKE', 'FAKE', 'REAL', 'REAL', 'REAL', 'REAL', 'FAKE', 'FAKE',
'REAL', 'FAKE', 'FAKE', 'FAKE', 'FAKE', 'FAKE', 'REAL', 'REAL', 'FAKE', 'REAL', 'REA
L', 'FAKE', 'REAL', 'REAL', 'REAL', 'FAKE', 'FAKE', 'FAKE', 'FAKE', 'REAL', 'REAL',
'REAL', 'REAL', 'FAKE', 'FAKE', 'REAL', 'REAL', 'REAL', 'FAKE', 'REAL', 'REAL', 'REA
L', 'REAL', 'REAL', 'REAL', 'REAL', 'REAL', 'FAKE', 'REAL', 'REAL', 'REAL', 'FAKE',
'REAL', 'FAKE', 'FAKE', 'REAL', 'REAL', 'FAKE', 'REAL', 'FAKE', 'FAKE', 'REAL', 'REA
L', 'REAL', 'REAL', 'REAL', 'FAKE', 'REAL', 'FAKE', 'FAKE', 'REAL', 'REAL', 'REAL',
'REAL', 'REAL', 'FAKE', 'FAKE', 'REAL', 'REAL', 'FAKE', 'FAKE', 'FAKE', 'REAL', 'FAK
E', 'REAL', 'REAL', 'FAKE', 'FAKE', 'FAKE', 'FAKE', 'REAL', 'FAKE', 'FAKE', 'FAKE',
'REAL', 'REAL', 'REAL', 'FAKE', 'REAL', 'FAKE', 'FAKE', 'REAL', 'FAKE', 'FAKE', 'FAK
E', 'FAKE', 'REAL', 'REAL', 'FAKE', 'FAKE', 'REAL', 'FAKE', 'FAKE', 'FAKE', 'REAL',
'REAL', 'REAL', 'FAKE', 'REAL', 'FAKE', 'FAKE', 'FAKE', 'FAKE', 'FAKE', 'FAKE', 'REA
L']
```

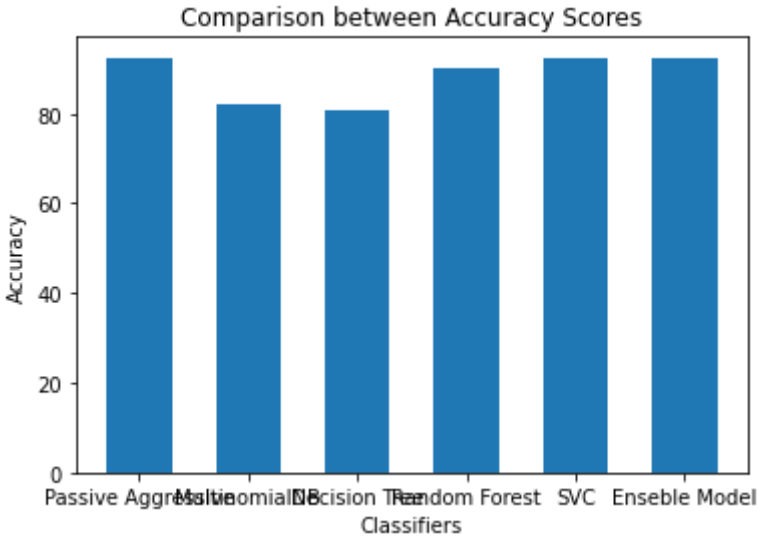
In [37]:

```
score=accuracy_score(y_test,y_final)
print(score)
```

```
0.9237243556023146
```

In [50]:

```
a=["Passive Aggressive","MultinomialNB","Decision Tree","Random Forest","SVC","Enseb
b=[92.37,82.11,80.75,90.01,92.48,92.37]
plt.bar(a,b,0.6)
plt.xlabel("Classifiers")
plt.ylabel("Accuracy")
plt.title("Comparison between Accuracy Scores")
plt.show()
```



In [ ]: