# PROJECT NAME

# FAKE NEWS PROJECT

Submitted by:

Aarti Ambhore

# ACKNOWLEDGMENT

This includes mentioning of all the references, research papers, data sources, professionals and other resources that helped you and guided you in completion of the project.

I took help from google, Kaggle, GitHub and my previous project.

# INTRODUCTION

## Problem Statement:

Fake news has become one of the biggest problems of our age. It has serious impact on our online as well as offline discourse. One can even go as far as saying that, to date, fake news poses a clear and present danger to western democracy and stability of the society.

## What is fake news?

Fake news's simple meaning is to incorporate information that leads people to the wrong path. Nowadays fake news spreading like water and people share this information without verifying it. This is often done to further or impose certain ideas and is often achieved with political agendas.

For media outlets, the ability to attract viewers to their websites is necessary to generate online advertising revenue. So, it is necessary to detect fake news.

## Analytical Modelling:

- Naïve Bayes Model
- Logistic Regression Model
- Decision Tree Classifier Model
- Random Forest Model

## Data Source (Given):

You can find many datasets for fake news detection on Kaggle or many other sites. I download these datasets from Kaggle. There are two datasets one for fake news and one for true news. In true news, there is 21417 news, and in fake news, there is 23481 news. You have to insert one label column zero for fake news and one for true news. We are combined both datasets using pandas built-in function.

The csv file then read as DataFrame (df).

Data processing was done and plotted the graph for the data and compared that which model is best fit and having high accuracy as compare to others.

# Evaluation of the Models:

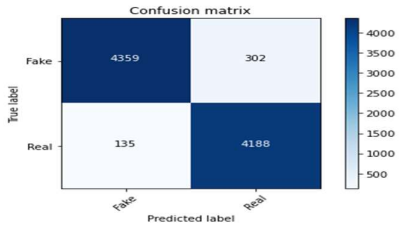
## 1. Naïve Bayes Model:

In [46]:
```
dct = dict()

from sklearn.naive_bayes import MultinomialNB

NB_classifier = MultinomialNB()
pipe = Pipeline([('vect', CountVectorizer()),
                 ('tfidf', TfidfTransformer()),
                 ('model', NB_classifier)])

model = pipe.fit(X_train, y_train)
prediction = model.predict(X_test)
print("accuracy: {}%".format(round(accuracy_score(y_test, prediction)*100,2)))

dct['Naive Bayes'] = round(accuracy_score(y_test, prediction)*100,2)
```

```
accuracy: 95.14%
```

In [47]:
```
cm = metrics.confusion_matrix(y_test, prediction)
plot_confusion_matrix(cm, classes=['Fake', 'Real'])
```

```
Confusion matrix, without normalization
```



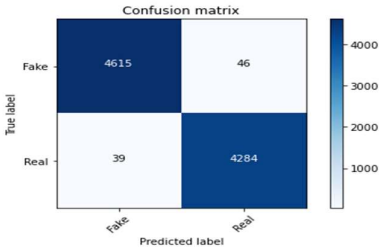

## 2. Logistic Regression Model:

In [48]:
```
# Logistic regression
from sklearn.linear_model import LogisticRegression

pipe = Pipeline([('vect', CountVectorizer()),
                 ('tfidf', TfidfTransformer()),
                 ('model', LogisticRegression())])

# Fitting the model
model = pipe.fit(X_train, y_train)

# Accuracy
prediction = model.predict(X_test)
print("accuracy: {}%".format(round(accuracy_score(y_test, prediction)*100,2)))
dct['Logistic Regression'] = round(accuracy_score(y_test, prediction)*100,2)
```

```
accuracy: 99.05%
```

In [49]:
```
cm = metrics.confusion_matrix(y_test, prediction)
plot_confusion_matrix(cm, classes=['Fake', 'Real'])
```

```
Confusion matrix, without normalization
```



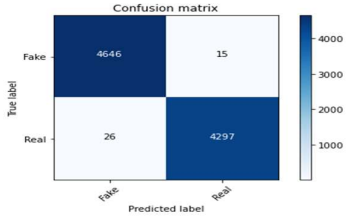
## 3. Decision Tree Classifier Model:

In [50]:
```
from sklearn.tree import DecisionTreeClassifier

# Vectorizing and applying TF-IDF
pipe = Pipeline([('vect', CountVectorizer()),
                 ('tfidf', TfidfTransformer()),
                 ('model', DecisionTreeClassifier(criterion= 'entropy',
                                                  max_depth = 20,
                                                  splitter='best',
                                                  random_state=42))])
# Fitting the model
model = pipe.fit(X_train, y_train)

# Accuracy
prediction = model.predict(X_test)
print("accuracy: {}%".format(round(accuracy_score(y_test, prediction)*100,2)))
dct['Decision Tree'] = round(accuracy_score(y_test, prediction)*100,2)
```

```
accuracy: 99.54%
```

In [51]:
```
cm = metrics.confusion_matrix(y_test, prediction)
plot_confusion_matrix(cm, classes=['Fake', 'Real'])
```

```
Confusion matrix, without normalization
```



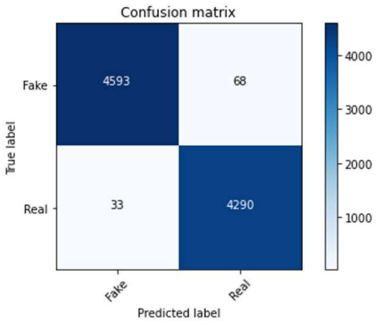

## 4. Random Forest:

In [52]:
```
from sklearn.ensemble import RandomForestClassifier

pipe = Pipeline([('vect', CountVectorizer()),
                 ('tfidf', TfidfTransformer()),
                 ('model', RandomForestClassifier(n_estimators=50, criterion="entropy"))])

model = pipe.fit(X_train, y_train)
prediction = model.predict(X_test)
print("accuracy: {}%".format(round(accuracy_score(y_test, prediction)*100,2)))
dct['Random Forest'] = round(accuracy_score(y_test, prediction)*100,2)
```

```
accuracy: 98.88%
```

In [53]:
```
cm = metrics.confusion_matrix(y_test, prediction)
plot_confusion_matrix(cm, classes=['Fake', 'Real'])
```

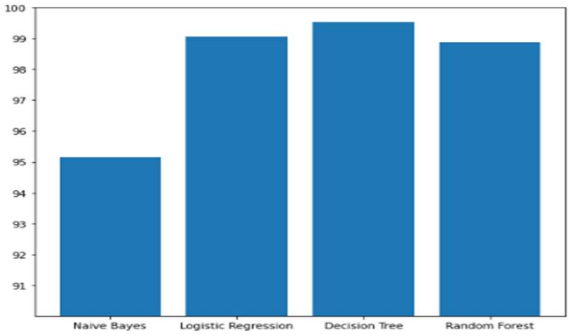
```
Confusion matrix, without normalization
```



### Comparison:

```
In [54]:
1 import matplotlib.pyplot as plt
2 plt.figure(figsize=(8,7))
3 plt.bar(list(dct.keys()),list(dct.values()))
4 plt.ylim(90,100)
5 plt.yticks((91, 92, 93, 94, 95, 96, 97, 98, 99, 100))
```

```
Out[54]: ([<matplotlib.axis.YTick at 0x1eaad66b1c0>,
  <matplotlib.axis.YTick at 0x1eaacf2cfd0>,
  <matplotlib.axis.YTick at 0x1ea83011610>,
  <matplotlib.axis.YTick at 0x1eaad744bb0>,
  <matplotlib.axis.YTick at 0x1eaad744d30>,
  <matplotlib.axis.YTick at 0x1eaad664a90>,
  <matplotlib.axis.YTick at 0x1ea82f4af10>,
  <matplotlib.axis.YTick at 0x1ea82f4a220>,
  <matplotlib.axis.YTick at 0x1eaad656af0>,
  <matplotlib.axis.YTick at 0x1ea82fdf460>],
 [Text(0, 0, ''),
  Text(0, 0, ''),
  Text(0, 0, ''),
  Text(0, 0, ''),
  Text(0, 0, ''),
  Text(0, 0, ''),
  Text(0, 0, ''),
  Text(0, 0, ''),
  Text(0, 0, ''),
  Text(0, 0, '')])
```

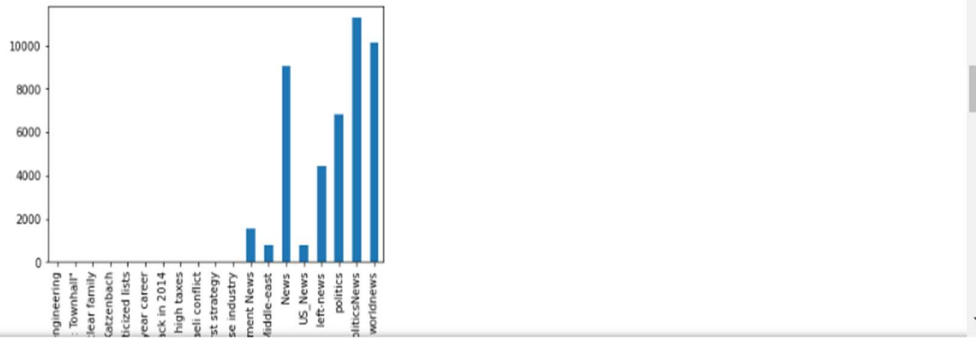



# Count Plots

### News subject:

```
In [36]:
1 # How many articles per subject?
2 print(data.groupby(['subject'])['text'].count())
3 data.groupby(['subject'])['text'].count().plot(kind="bar")
4 plt.show()
```

```
10145
Name: text, dtype: int64
```




### Observation:

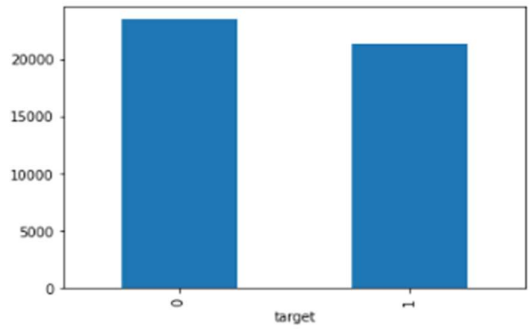
The given count plot is for the news subject, and from this plot we can say that there are more articles published on the political subjects.

The exact number of the articles on politics is 11272.

## Fake and Real Articles:

```
In [37]:
1 # How many fake and real articles?
2 print(data.groupby(['target'])['text'].count())
3 data.groupby(['target'])['text'].count().plot(kind="bar")
4 plt.show()
```

```
target
0    23502
1    21417
Name: text, dtype: int64
```




**Observation:**

So as per the instruction, I have labelled the Fake news columns as “0” and real news is labelled as “1”.

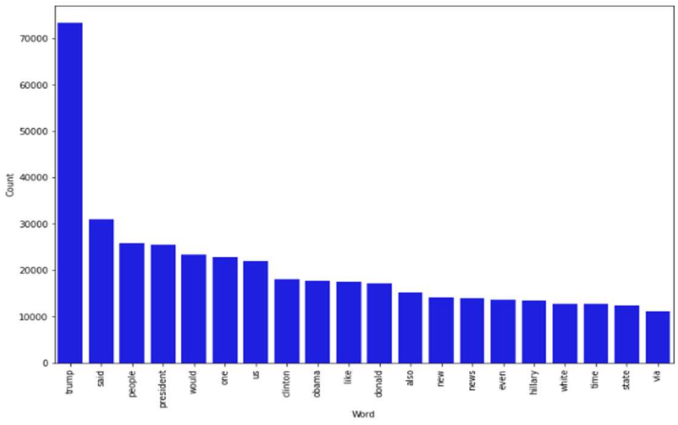
So, from the graph we can say that there are 23502 counts for fake news,
And 21417 counts for the real news.

We can say that the fake news number is higher than the real news.

# Bar Graph:

## Most frequent words in Fake News:

```
In [39]:
# Most frequent words in fake news
counter(data[data["target"] == 0], "text", 20)
```

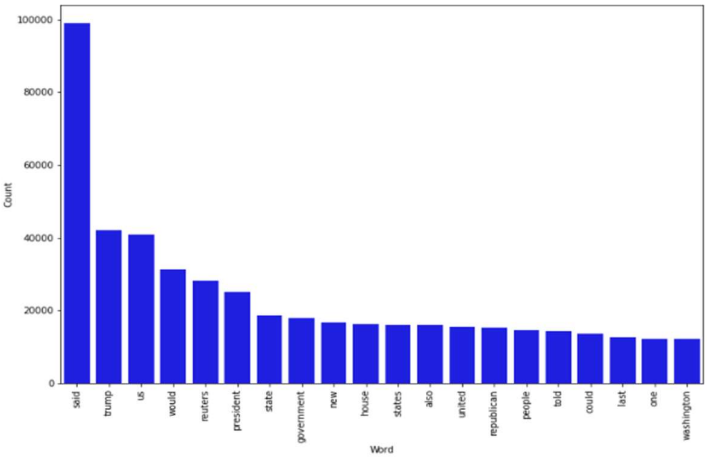



**Observation:**

So, the above graph is showing the most frequent words in fake news, we can say that “trump” word has been used frequently than the other words shown in the graph.

## Most frequent words in Real news:

```
In [40]:
# Most frequent words in real news
counter(data[data["target"] == 1], "text", 20)
```




**Observation:**

Form the above graph we can say that “said” has been used most frequently than the other words.

# Result:

From the count plots and bar graphs we can interpret the following results,

1. Most of the articles are on Politics news.
2. As compare to the real news the number of fake news is high.
3. Most frequent word used in the fake news is “trump”.
4. And most frequent word used in real news is “said”.

# Key findings and Conclusion:

The model that we developed is not particularly belongs to one media and in our dataset all the data consists of news reports from various digital media that means our model understanding could be applied to any digital media to know what is fraud and what is real.

# Outcomes:

Here we have used four models to represent our data, out of which Decision Tree Classifier is the best fit model and having 99.54% accuracy.

# Limitations and future scope:

Our model’s future work is to develop a dynamic model so that our users can download our app and can easily detect any news and any fake URL’s and we are also thinking to develop a model so that it can detect any fake profiles present in any media such as Facebook, Instagram, Stack Overflow and also any fraud reviews for duplicate products. Not only these but there are also many unsettled controversies and topics about celebrity’s fraud news and the news articles about the world is always a concern to each and everyone. By considering above matters professionals have to share out with them. For example, recently many frauds news articles about covid vaccine created an immense effect on people. My opinion people must research about news if any new news is in front of us. Now-a-days many news are being forwarded to WhatsApp application about funding and it is especially important to point out the major sources of that news and have knowledge about them and share to people so that everyone can able to understand about a particular news. I think everything is there in people’s hand, if we SCRS Conference Proceedings on Intelligent Systems (2021) 67 are concern about any social cause then there will be no spread of Fake news. Proposed BERT outperforms LR and KNN models and accuracy may be improved by using machine learning ensemble methods.

**THE END**