In [1]:
```python
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.feature_extraction.text import CountVectorizer
from sklearn.feature_extraction.text import TfidfTransformer
from sklearn import feature_extraction, linear_model, model_selection, preprocessing
from sklearn.metrics import accuracy_score
from sklearn.model_selection import train_test_split
from sklearn.pipeline import Pipeline
```

In [3]:
```python
fake = pd.read_csv("Fake.csv")
true = pd.read_csv("True.csv")
```

In [4]:
```python
fake.shape
```

Out[4]:
```
(23481, 4)
```

In [5]:
```python
true.shape
```

Out[5]:
```
(21417, 4)
```

Data cleaning and preparation

In [6]:
```python
# Add flag to track fake and real
fake['target'] = 'fake'
true['target'] = 'true'
```

In [7]:
```python
# Concatenate dataframes
data = pd.concat([fake, true]).reset_index(drop = True)
data.shape
```

Out[7]:
```
(44898, 5)
```

In [8]:
```python
# Shuffle the data
from sklearn.utils import shuffle
data = shuffle(data)
data = data.reset_index(drop=True)
```

In [9]:
```python
# Check the data
data.head()
```

Out[9]:

	title	text	subject	date	target
0	Pathetic: Trump Actually Needs A General To B...	With Reince Priebus out as White House chief o...	News	August 5, 2017	fake
1	SHOCKER! IS MITT ROMNEY Being Considered For A...		politics	Nov 17, 2016	fake
2	Bridgewater executive McCormick declines Defen...	NEW YORK (Reuters) - David McCormick has decli...	politicsNews	January 10, 2017	true
3	Week of clashes in eastern Ethiopia kill 50, d...	ADDIS ABABA (Reuters) - Clashes along the bord...	worldnews	September 17, 2017	true
4	U.S. Senate number two Democrat calls Trump's ...	WASHINGTON (Reuters) - The U.S. Senate's numbe...	politicsNews	May 15, 2017	true

In [10]:
```python
# Removing the date (we won't use it for the analysis)
data.drop(["date"],axis=1,inplace=True)
data.head()
```

Out[10]:

	title	text	subject	target
0	Pathetic: Trump Actually Needs A General To B...	With Reince Priebus out as White House chief o...	News	fake
1	SHOCKER! IS MITT ROMNEY Being Considered For A...		politics	fake
2	Bridgewater executive McCormick declines Defen...	NEW YORK (Reuters) - David McCormick has decli...	politicsNews	true
3	Week of clashes in eastern Ethiopia kill 50, d...	ADDIS ABABA (Reuters) - Clashes along the bord...	worldnews	true
4	U.S. Senate number two Democrat calls Trump's ...	WASHINGTON (Reuters) - The U.S. Senate's numbe...	politicsNews	true

In [11]:
```python
# Removing the title (we will only use the text)
data.drop(["title"],axis=1,inplace=True)
data.head()
```

Out[11]:

	text	subject	target
0	With Reince Priebus out as White House chief o...	News	fake
1		politics	fake
2	NEW YORK (Reuters) - David McCormick has decli...	politicsNews	true
3	ADDIS ABABA (Reuters) - Clashes along the bord...	worldnews	true
4	WASHINGTON (Reuters) - The U.S. Senate's numbe...	politicsNews	true

In [12]:
```python
# Convert to lowercase
data['text'] = data['text'].apply(lambda x: x.lower())
data.head()
```

Out[12]:

	text	subject	target
0	with reince priebus out as white house chief o...	News	fake
1		politics	fake
2	new york (reuters) - david mccormick has decli...	politicsNews	true
3	addis ababa (reuters) - clashes along the bord...	worldnews	true
4	washington (reuters) - the u.s. senate's numbe...	politicsNews	true

In [13]:
```python
# Remove punctuation

import string

def punctuation_removal(text):
    all_list = [char for char in text if char not in string.punctuation]
    clean_str = ''.join(all_list)
    return clean_str

data['text'] = data['text'].apply(punctuation_removal)
```

In [14]:
```python
data.head()
```

Out[14]:

	text	subject	target
0	with reince priebus out as white house chief o...	News	fake
1		politics	fake
2	new york reuters david mccormick has declined...	politicsNews	true
3	addis ababa reuters clashes along the border ...	worldnews	true
4	washington reuters the us senate's number two...	politicsNews	true

In [18]:
```python
# Removing stopwords
import nltk
nltk.download('stopwords')
from nltk.corpus import stopwords
stop = stopwords.words('english')

data['text'] = data['text'].apply(lambda x: ' '.join([word for word in x.split() if word not in (stop)]))
```

```
[nltk_data] Downloading package stopwords to
[nltk_data]     C:\Users\Guru\AppData\Roaming\nltk_data...
[nltk_data]   Package stopwords is already up-to-date!
```

In [17]:
```python
data.head()
```

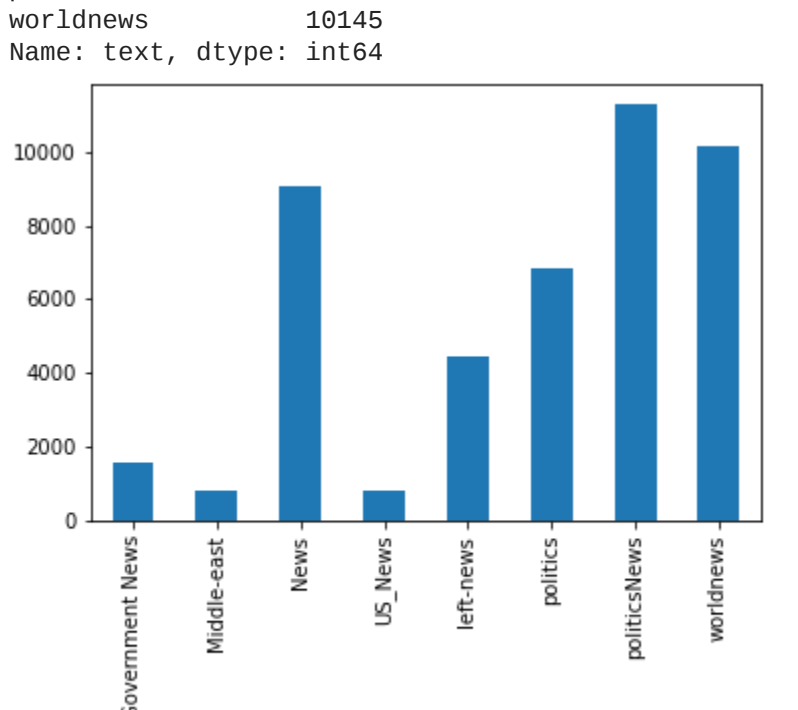
Out[17]:

	text	subject	target
0	reince priebus white house chief staff general...	News	fake
1		politics	fake
2	new york reuters david mccormick declined offe...	politicsNews	true
3	addis ababa reuters clashes along border ethio...	worldnews	true
4	washington reuters us senate's number two demo...	politicsNews	true

Basic data exploration

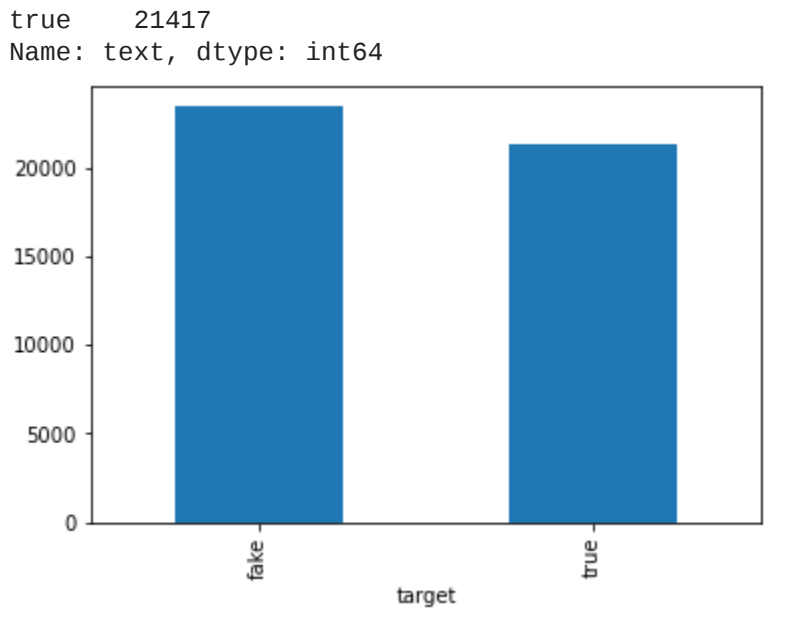
In [19]:
```python
# How many articles per subject?
print(data.groupby(['subject'])['text'].count())
data.groupby(['subject'])['text'].count().plot(kind="bar")
plt.show()
```

```
subject
Government News     1570
Middle-east          778
News                9050
US_News              783
left-news           4459
politics            6841
politicsNews       11272
worldnews          10145
Name: text, dtype: int64
```



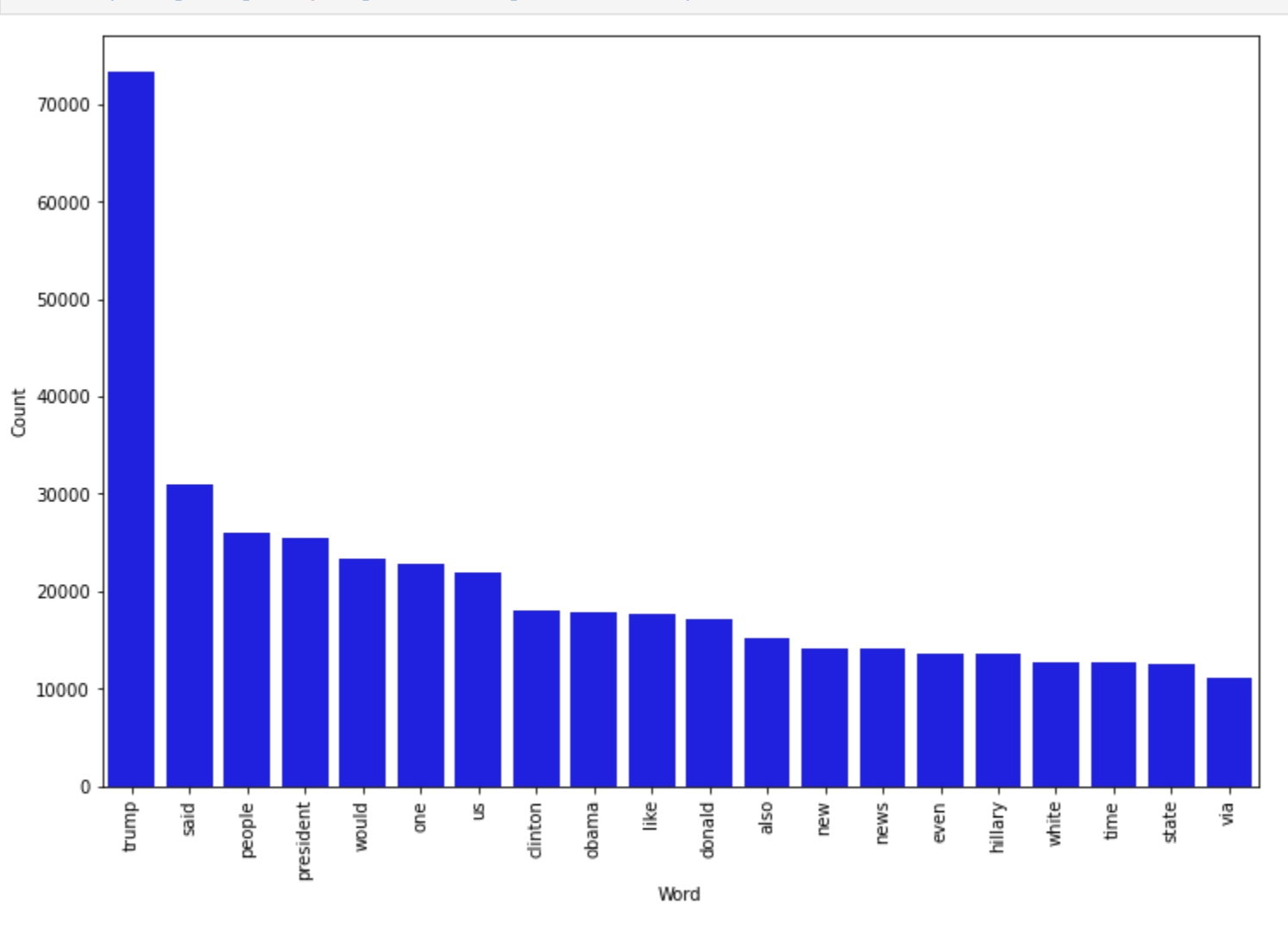
In [20]:
```python
# How many fake and real articles?
print(data.groupby(['target'])['text'].count())
data.groupby(['target'])['text'].count().plot(kind="bar")
plt.show()
```

```
target
fake    23481
true    21417
Name: text, dtype: int64
```

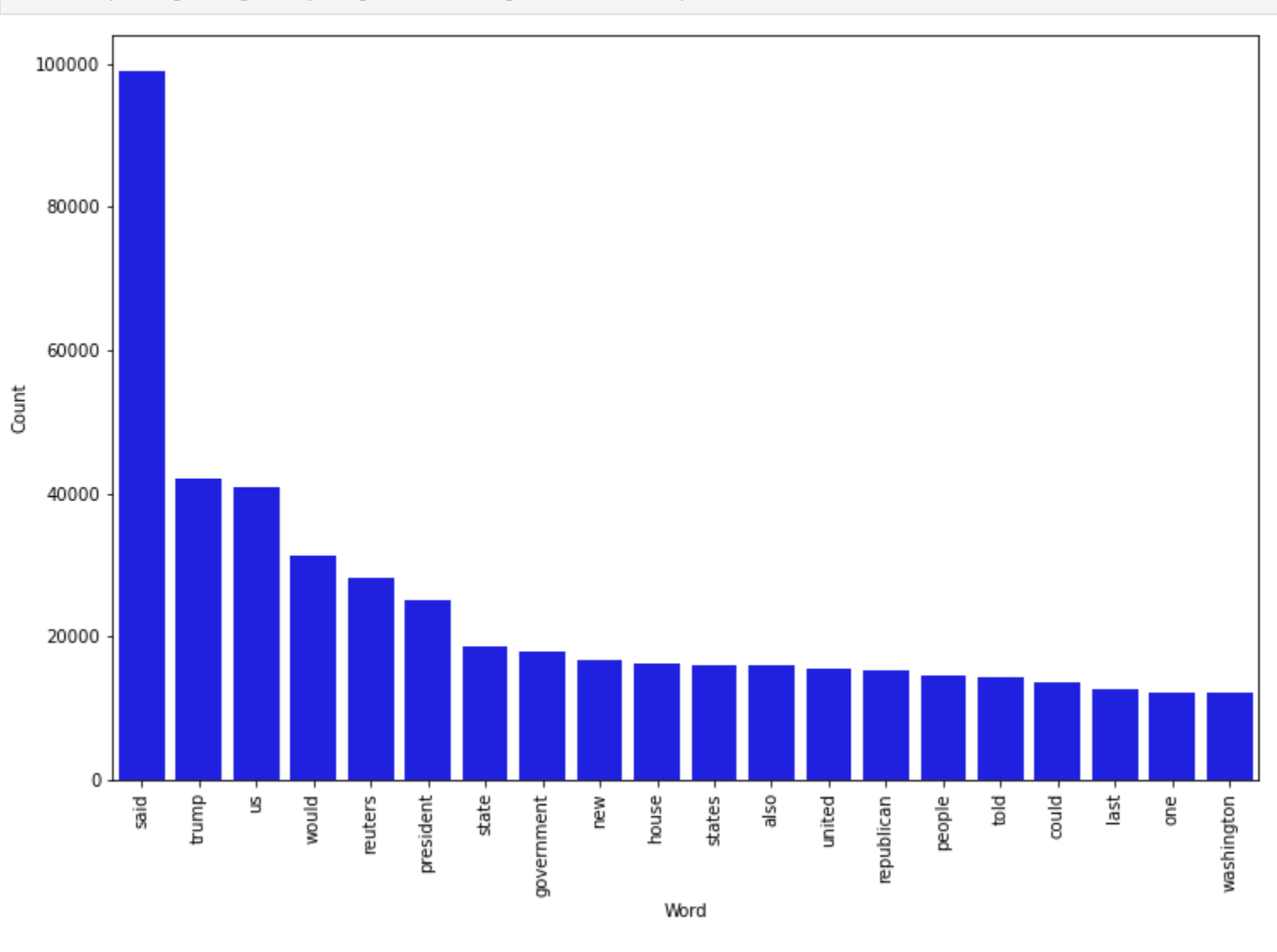


In [23]:
```python
# Most frequent words counter (Code adapted from https://www.kaggle.com/rodolfoluna/fake-news-detector)
from nltk import tokenize

token_space = tokenize.WhitespaceTokenizer()

def counter(text, column_text, quantity):
    all_words = ' '.join([text for text in text[column_text]])
    token_phrase = token_space.tokenize(all_words)
    frequency = nltk.FreqDist(token_phrase)
    df_frequency = pd.DataFrame({"Word": list(frequency.keys()),
                                   "Frequency": list(frequency.values())})
    df_frequency = df_frequency.nlargest(columns = "Frequency", n = quantity)
    plt.figure(figsize=(12,8))
    ax = sns.barplot(data = df_frequency, x = "Word", y = "Frequency", color = 'blue')
    ax.set(ylabel = "Count")
    plt.xticks(rotation='vertical')
    plt.show()
```

In [45]:
```python
# Most frequent words in fake news
counter(data[data["target"] == "fake"], "text", 20)
```



In [25]:
```python
# Most frequent words in real news
counter(data[data["target"] == "true"], "text", 20)
```



Modeling

In [26]:
```python
# Function to plot the confusion matrix (code from https://scikit-learn.org/stable/auto_examples/model_selection/plot_confusion_matrix.html)
from sklearn import metrics
import itertools

def plot_confusion_matrix(cm, classes,
                          normalize=False,
                          title='Confusion matrix',
                          cmap=plt.cm.Blues):

    plt.imshow(cm, interpolation='nearest', cmap=cmap)
    plt.title(title)
    plt.colorbar()
    tick_marks = np.arange(len(classes))
    plt.xticks(tick_marks, classes, rotation=45)
    plt.yticks(tick_marks, classes)

    if normalize:
        cm = cm.astype('float') / cm.sum(axis=1)[:, np.newaxis]
        print("Normalized confusion matrix")
    else:
        print('Confusion matrix, without normalization')

    thresh = cm.max() / 2.
    for i, j in itertools.product(range(cm.shape[0]), range(cm.shape[1])):
        plt.text(j, i, cm[i, j],
                 horizontalalignment="center",
                 color="white" if cm[i, j] > thresh else "black")

    plt.tight_layout()
    plt.ylabel('True label')
    plt.xlabel('Predicted label')
```

In [27]:
```python
#Peparing the data
# Split the data
X_train,X_test,y_train,y_test = train_test_split(data['text'], data.target, test_size=0.2, random_state=42)
```

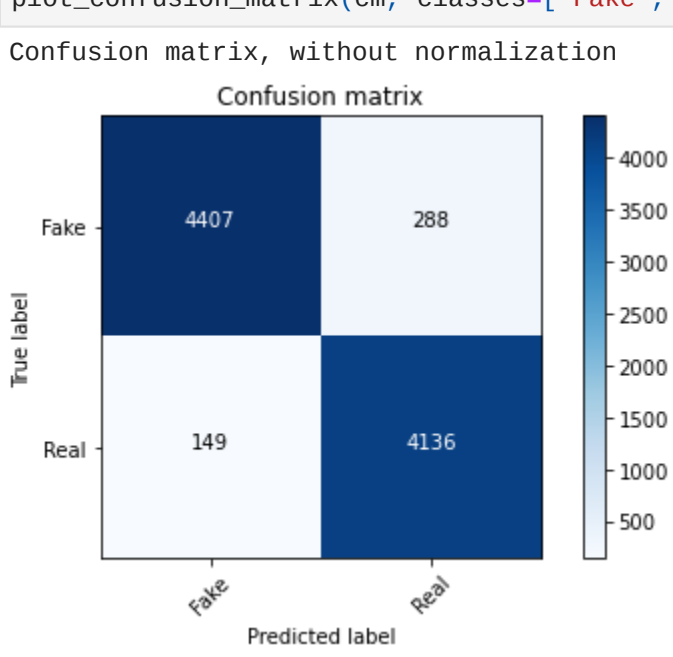
Naive Bayes

In [28]:
```python
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: 95.13%
```

In [29]:
```python
cm = metrics.confusion_matrix(y_test, prediction)
plot_confusion_matrix(cm, classes=['Fake', 'Real'])
```

```
Confusion matrix, without normalization
```



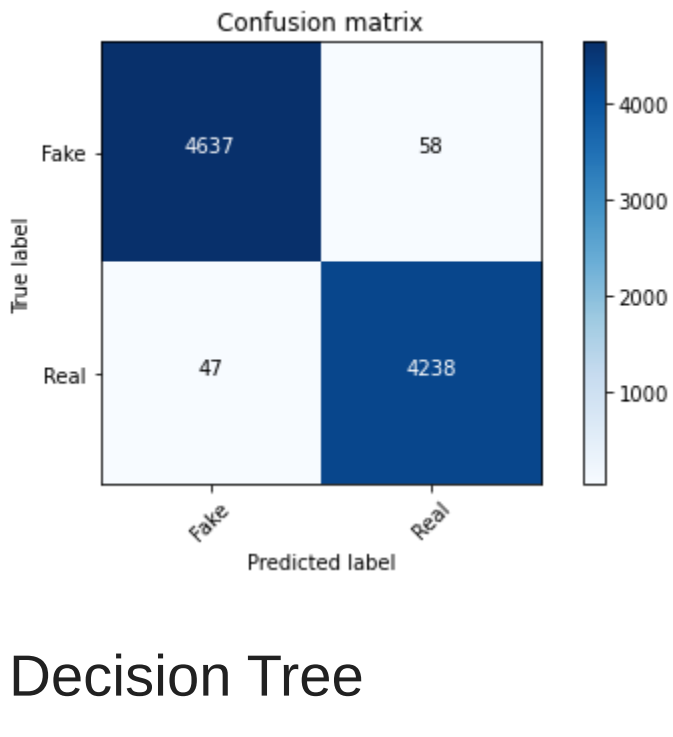
Logistic regression

In [30]:
```python
# Vectorizing and applying TF-IDF
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: 98.83%
```

In [31]:
```python
cm = metrics.confusion_matrix(y_test, prediction)
plot_confusion_matrix(cm, classes=['Fake', 'Real'])
```

```
Confusion matrix, without normalization
```



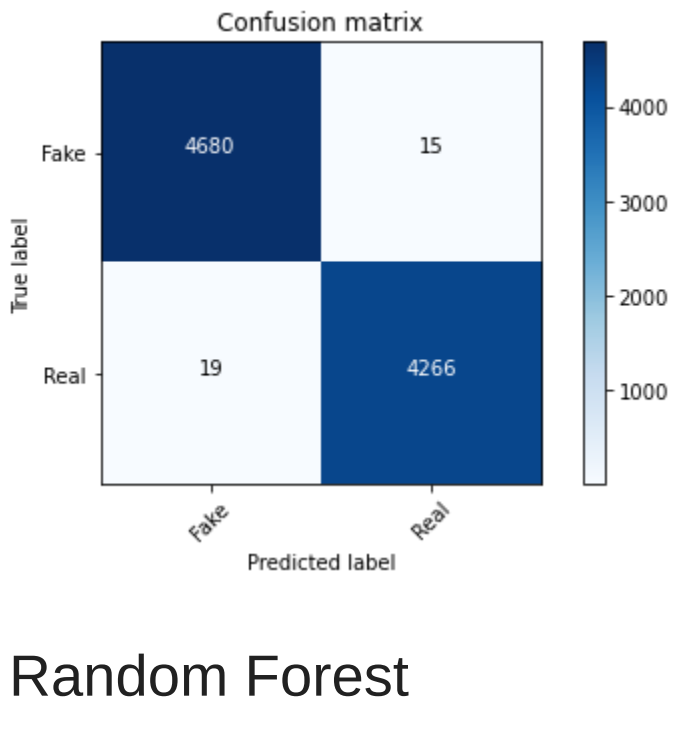
Decision Tree

In [37]:
```python
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: 99.62%
```

In [33]:
```python
cm = metrics.confusion_matrix(y_test, prediction)
plot_confusion_matrix(cm, classes=['Fake', 'Real'])
```

```
Confusion matrix, without normalization
```



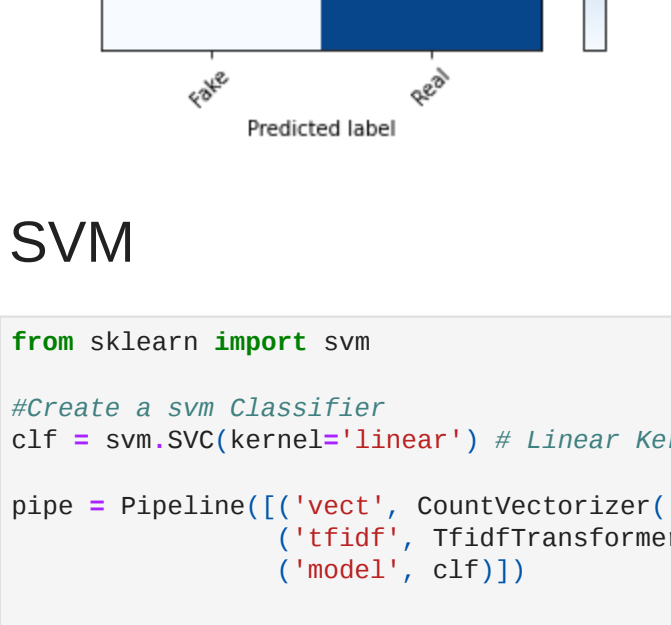
Random Forest

In [36]:
```python
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: 99.01%
```

In [38]:
```python
cm = metrics.confusion_matrix(y_test, prediction)
plot_confusion_matrix(cm, classes=['Fake', 'Real'])
```

```
Confusion matrix, without normalization
```



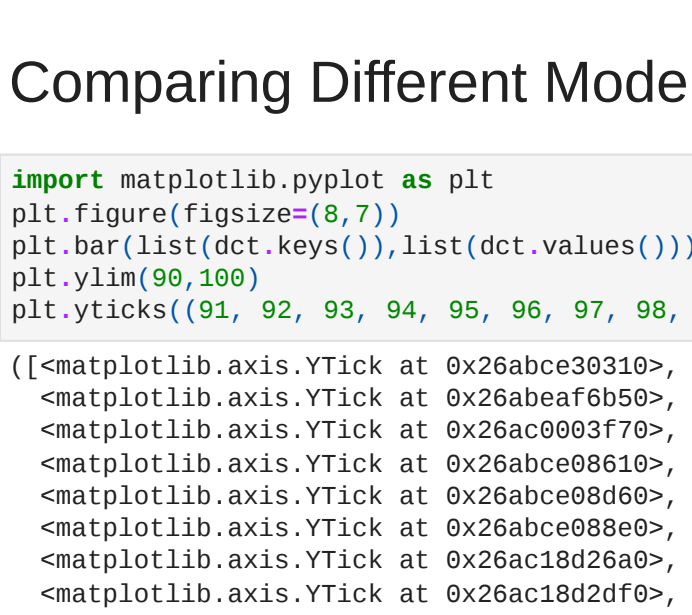
SVM

In [39]:
```python
from sklearn import svm

#Create a svm Classifier
clf = svm.SVC(kernel='linear') # Linear Kernel

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

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

```
accuracy: 99.57%
```

In [40]:
```python
cm = metrics.confusion_matrix(y_test, prediction)
plot_confusion_matrix(cm, classes=['Fake', 'Real'])
```

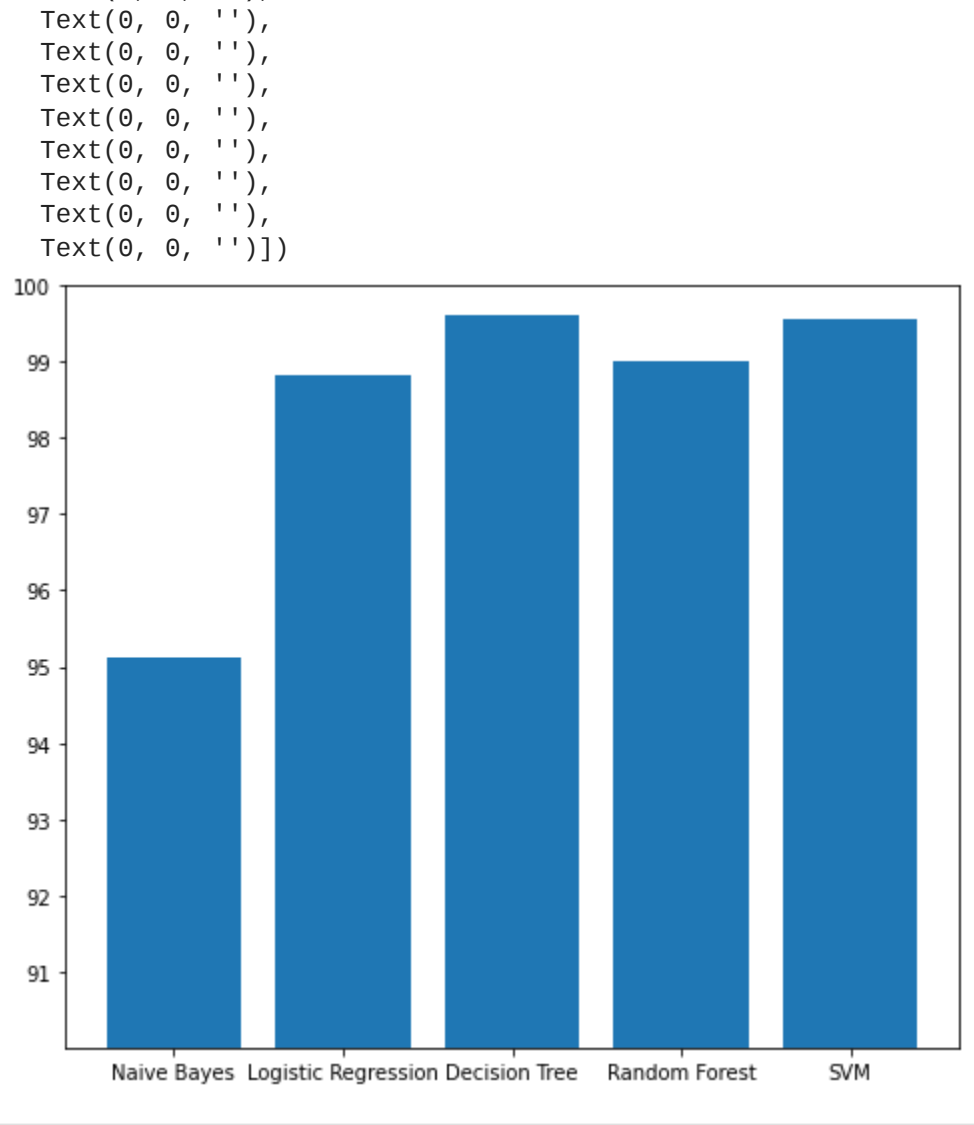
```
Confusion matrix, without normalization
```



Comparing Different Models

In [41]:
```python
import matplotlib.pyplot as plt
plt.figure(figsize=(8,7))
plt.bar(list(dct.keys()),list(dct.values()))
plt.ylim(90,100)
plt.yticks((91, 92, 93, 94, 95, 96, 97, 98, 99, 100))
```

Out[41]:
```
([<matplotlib.axis.YTick at 0x26abce30310>,
  <matplotlib.axis.YTick at 0x26abeaf6b50>,
  <matplotlib.axis.YTick at 0x26ac0003f70>,
  <matplotlib.axis.YTick at 0x26abce08610>,
  <matplotlib.axis.YTick at 0x26abce08d60>,
  <matplotlib.axis.YTick at 0x26abce088e0>,
  <matplotlib.axis.YTick at 0x26ac18d26a0>,
  <matplotlib.axis.YTick at 0x26ac18d2df0>,
  <matplotlib.axis.YTick at 0x26ad551f580>,
  <matplotlib.axis.YTick at 0x26ad551fd00>],
 [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, '')])
```



In [ ]: