DETECTION OF FAKE NEWS USING MACHINE LEARNING

IMPORTING THE LIBRARIES

In [1]:

```
import pandas as pd
import numpy as np
import re
import string
import nltk
nltk.download('punkt')
from nltk.tokenize import word_tokenize
nltk.download('wordnet')
from nltk.corpus import wordnet as wn
from nltk.stem.wordnet import WordNetLemmatizer
from nltk.stem import WordNetLemmatizer
nltk.download('stopwords')
from sklearn.model_selection import train_test_split
from sklearn.feature_extraction.text import TfidfVectorizer
from sklearn.model_selection import GridSearchCV
from sklearn.neighbors import KNeighborsClassifier
from sklearn.linear_model import LogisticRegression
from sklearn.tree import DecisionTreeClassifier
from sklearn.ensemble import AdaBoostClassifier
from sklearn.ensemble import GradientBoostingClassifier
from sklearn.ensemble import RandomForestClassifier
from sklearn.svm import SVC
from sklearn.ensemble import VotingClassifier
from sklearn.pipeline import Pipeline
from mlxtend.classifier import StackingClassifier
from sklearn.metrics import accuracy_score
from sklearn.metrics import confusion_matrix
from sklearn.model_selection import cross_val_score
import matplotlib.pyplot as plt
import itertools
from sklearn.metrics import classification_report
```

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

IMPORTING THE DATASET

In [2]:

```
df_fake=pd.read_csv("Fake.csv")
df_true=pd.read_csv("True.csv")
```

In [3]:

```
df_fake.tail(10)
```

Out[3]:

	title	text	subject	date
23471	Seven Iranians freed in the prisoner swap have...	21st Century Wire says This week, the historic...	Middle-east	January 20, 2016
23472	#Hashtag Hell & The Fake Left	By Dady Chery and Gilbert MercierAll writers ...	Middle-east	January 19, 2016
23473	Astroturfing: Journalist Reveals Brainwashing ...	Vic Bishop Waking TimesOur reality is carefull...	Middle-east	January 19, 2016
23474	The New American Century: An Era of Fraud	Paul Craig RobertsIn the last years of the 20t...	Middle-east	January 19, 2016
23475	Hillary Clinton: ‘Israel First’ (and no peace ...	Robert Fantina CounterpunchAlthough the United...	Middle-east	January 18, 2016
23476	McPain: John McCain Furious That Iran Treated ...	21st Century Wire says As 21WIRE reported earl...	Middle-east	January 16, 2016
23477	JUSTICE? Yahoo Settles E-mail Privacy Class-ac...	21st Century Wire says It s a familiar theme. ...	Middle-east	January 16, 2016
23478	Sunnistan: US and Allied ‘Safe Zone’ Plan to T...	Patrick Henningsen 21st Century WireRemember ...	Middle-east	January 15, 2016
23479	How to Blow $700 Million: Al Jazeera America F...	21st Century Wire says Al Jazeera America will...	Middle-east	January 14, 2016
23480	10 U.S. Navy Sailors Held by Iranian Military ...	21st Century Wire says As 21WIRE predicted in ...	Middle-east	January 12, 2016

In [4]:

```
df_fake.shape
```

Out[4]:

```
(23481, 4)
```

In [5]:

```
df_true.tail(10)
```

Out[5]:

	title	text	subject	date
21407	Mata Pires, owner of embattled Brazil builder ...	SAO PAULO (Reuters) - Cesar Mata Pires, the ow...	worldnews	August 22, 2017
21408	U.S., North Korea clash at U.N. forum over nuc...	GENEVA (Reuters) - North Korea and the United ...	worldnews	August 22, 2017
21409	U.S., North Korea clash at U.N. arms forum on ...	GENEVA (Reuters) - North Korea and the United ...	worldnews	August 22, 2017
21410	Headless torso could belong to submarine journ...	COPENHAGEN (Reuters) - Danish police said on T...	worldnews	August 22, 2017
21411	North Korea shipments to Syria chemical arms a...	UNITED NATIONS (Reuters) - Two North Korean sh...	worldnews	August 21, 2017
21412	'Fully committed' NATO backs new U.S. approach...	BRUSSELS (Reuters) - NATO allies on Tuesday we...	worldnews	August 22, 2017
21413	LexisNexis withdrew two products from Chinese ...	LONDON (Reuters) - LexisNexis, a provider of l...	worldnews	August 22, 2017
21414	Minsk cultural hub becomes haven from authorities	MINSK (Reuters) - In the shadow of disused Sov...	worldnews	August 22, 2017
21415	Vatican upbeat on possibility of Pope Francis ...	MOSCOW (Reuters) - Vatican Secretary of State ...	worldnews	August 22, 2017
21416	Indonesia to buy $1.14 billion worth of Russia...	JAKARTA (Reuters) - Indonesia will buy 11 Sukh...	worldnews	August 22, 2017

In [6]:

```
df_true.shape
```

Out[6]:

```
(21417, 4)
```

In [7]:

```
df_fake["class"]=0
df_true["class"]=1
```

In [8]:

```
df_fake_manual_testing=df_fake.tail(10)
df_fake.drop([23470,23480],axis=0,inplace=True)
df_true_manual_testing=df_true.tail(10)
df_true.drop([21406,21416],axis=0,inplace=True)
df_manual_testing=pd.concat([df_fake_manual_testing,df_true_manual_testing],axis=0)
df_manual_testing.to_csv("manual_testing.csv")
```

In [9]:

```
df_merge=pd.concat([df_fake,df_true],axis=0)
```

In [10]:

```
df=df_merge.drop(["subject","date"],axis=1)
df = df.sample(frac = 1)
df.head()
```

Out[10]:

	title	text	class
10231	AUDIT: Obama's IRS 'Misled' Americans to Get T...	Soooo the IRS lied to Americans to prod them...	0
13470	Kremlin: U.S. sanctions aimed at turning busin...	MOSCOW (Reuters) - The Kremlin said on Thursda...	1
22875	SYRIA: British and American Presence Directly ...	US paratrooper on security duty during a miss...	0
2240	Watch NBC's Andrea Mitchell Get BULLIED Out O...	If one thing has become abundantly clear, it s...	0
17190	FAMILY THREATENED AT GUNPOINT FOR DISPLAYING C...	Nothing says tolerance like putting a loaded g...	0

In [11]:

```
df.isnull().sum()
```

Out[11]:

```
title    0
text     0
class    0
dtype: int64
```

DATA PREPROCCESING

In [12]:

```
def conversion(title):
    title = title.lower()
    title = re.sub('\[.*?\]', '', title)
    title = re.sub("\\W"," ",title)
    title = re.sub('https?://\S+|www\.\S+', '', title)
    title = re.sub('<.*?>+', '', title)
    title = re.sub('[%s]' % re.escape(string.punctuation), '', title)
    title = re.sub('\n', '', title)
    title = re.sub('\w*\d\w*', '', title)
    return title
```

In [13]:

```
df["title"] = df["title"].apply(conversion)
```

In [14]:

```
def tokenization(title):
    title = word_tokenize(title)
    return title
```

In [15]:

```
df["title"] = df["title"].apply(tokenization)
```

In [16]:

```
df.head()
```

Out[16]:

	title	text	class
10231	[audit, obama, s, irs, misled, americans, to, ...	Soooo the IRS lied to Americans to prod them...	0
13470	[kremlin, u, s, sanctions, aimed, at, turning,...	MOSCOW (Reuters) - The Kremlin said on Thursda...	1
22875	[syria, british, and, american, presence, dire...	US paratrooper on security duty during a miss...	0
2240	[watch, nbc, s, andrea, mitchell, get, bullied...	If one thing has become abundantly clear, it s...	0
17190	[family, threatened, at, gunpoint, for, displa...	Nothing says tolerance like putting a loaded g...	0

In [17]:

```
lmtzr=WordNetLemmatizer()
def lemmetization(title):
    title = ' '.join([lmtzr.lemmatize(w,wn.NOUN) for w in title])
    return title
```

In [18]:

```
df["title"] = df["title"].apply(lemmetization)
```

In [19]:

```
df.head()
```

Out[19]:

	title	text	class
10231	audit obama s irs misled american to get them ...	Soooo the IRS lied to Americans to prod them...	0
13470	kremlin u s sanction aimed at turning business...	MOSCOW (Reuters) - The Kremlin said on Thursda...	1
22875	syria british and american presence directly e...	US paratrooper on security duty during a miss...	0
2240	watch nbc s andrea mitchell get bullied out of...	If one thing has become abundantly clear, it s...	0
17190	family threatened at gunpoint for displaying c...	Nothing says tolerance like putting a loaded g...	0

In [20]:

```
from nltk.corpus import stopwords
stop = stopwords.words('english')
df["title"] = df["title"].apply(lambda x: ' '.join([word for word in x.split() if word not in (stop)]))
```

In [21]:

```
df.head()
```

Out[21]:

	title	text	class
10231	audit obama irs misled american get sign obama...	Soooo the IRS lied to Americans to prod them...	0
13470	kremlin u sanction aimed turning business elit...	MOSCOW (Reuters) - The Kremlin said on Thursda...	1
22875	syria british american presence directly escal...	US paratrooper on security duty during a miss...	0
2240	watch nbc andrea mitchell get bullied state de...	If one thing has become abundantly clear, it s...	0
17190	family threatened gunpoint displaying confeder...	Nothing says tolerance like putting a loaded g...	0

SPLITTING DATA INTO TRAINING AND TESTING DATA

In [22]:

```
x = df.iloc[0:5000,0]
y = df.iloc[0:5000,-1]
x_train, x_test, y_train, y_test = train_test_split(x, y, test_size=0.2)
```

VECTORIZATION

In [23]:

```
vectorization = TfidfVectorizer()
xv_train = vectorization.fit_transform(x_train)
xv_test = vectorization.transform(x_test)
```

In [24]:

```
xv = vectorization.fit_transform(x)
```

CODE TO PLOT CONFUSION MATRIX

In [25]:

```
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')
```

KNN

In [26]:

```
grid_params = { 'n_neighbors' : list(range(1,65,2)),
               'weights' : ['uniform','distance'],
               'metric' : ['minkowski','euclidean','manhattan']}
gs = GridSearchCV(KNeighborsClassifier(), grid_params, verbose = 1, cv=3, n_jobs = -1)
g_res = gs.fit(xv_train, y_train)
g_res.best_score_
g_res.best_params_
```

```
Fitting 3 folds for each of 192 candidates, totalling 576 fits
```

Out[26]:

```
{'metric': 'minkowski', 'n_neighbors': 25, 'weights': 'distance'}
```

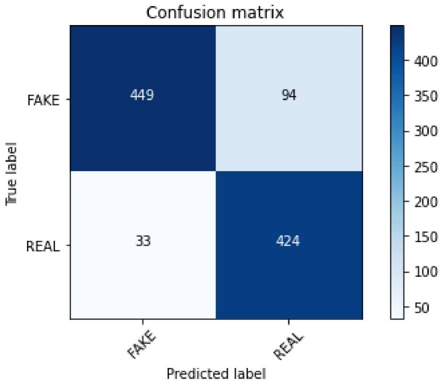
In [27]:

```
knn=KNeighborsClassifier(n_neighbors=25)
knn.fit(xv_train, y_train)
pred_train = knn.predict(xv_train)
pred_test = knn.predict(xv_test)
print("Bias is : ",1-accuracy_score(pred_train,y_train))
print("Variance is: ",1-accuracy_score(pred_test,y_test))
print("Accuracy is: ",accuracy_score(pred_test,y_test))
print("Cross Validation result is: ",cross_val_score(knn, xv, y, cv=10, scoring ='accuracy').mean())
cm=confusion_matrix(y_test,pred_test)
plot_confusion_matrix(cm, classes=['FAKE', 'REAL'])
print(classification_report(y_test, pred_test))
```

```
Bias is :  0.11350000000000005
Variance is:  0.127
Accuracy is:  0.873
Cross Validation result is:  0.8726
Confusion matrix, without normalization
              precision    recall  f1-score   support

           0       0.93      0.83      0.88       543
           1       0.82      0.93      0.87       457

    accuracy                           0.87      1000
   macro avg       0.88      0.88      0.87      1000
weighted avg       0.88      0.87      0.87      1000
```



LOGISTIC REGRESSION

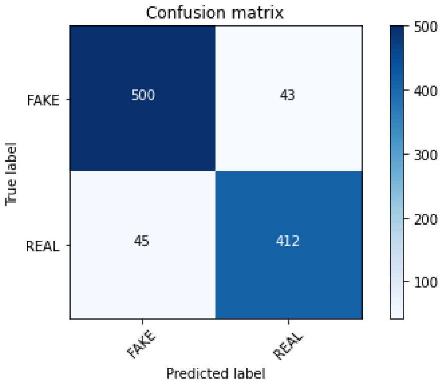
In [28]:

```
LR = LogisticRegression()
LR.fit(xv_train,y_train)
pred_train = LR.predict(xv_train)
pred_test = LR.predict(xv_test)
print("Bias is : ",1-accuracy_score(pred_train,y_train))
print("Variance is: ",1-accuracy_score(pred_test,y_test))
print("Accuracy is: ",accuracy_score(pred_test,y_test))
print("Cross Validation result is: ",cross_val_score(LR, xv, y, cv=10, scoring ='accuracy').mean())
cm=confusion_matrix(y_test,pred_test)
plot_confusion_matrix(cm, classes=['FAKE', 'REAL'])
print(classification_report(y_test, pred_test))
```

```
Bias is :  0.032749999999999946
Variance is:  0.08799999999999997
Accuracy is:  0.912
Cross Validation result is:  0.9152000000000001
Confusion matrix, without normalization
              precision    recall  f1-score   support

           0       0.92      0.92      0.92       543
           1       0.91      0.90      0.90       457

    accuracy                           0.91      1000
   macro avg       0.91      0.91      0.91      1000
weighted avg       0.91      0.91      0.91      1000
```



DECISION TREE

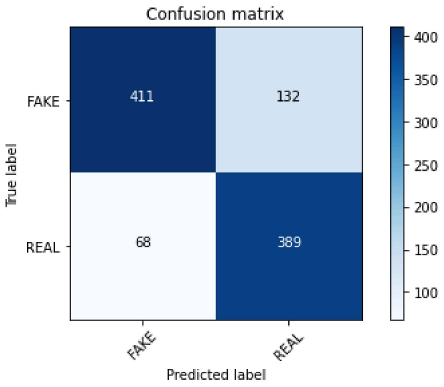
In [29]:

```
DT = DecisionTreeClassifier()
DT.fit(xv_train, y_train)
pred_train = DT.predict(xv_train)
pred_test = DT.predict(xv_test)
print("Bias is : ",1-accuracy_score(pred_train,y_train))
print("Variance is: ",1-accuracy_score(pred_test,y_test))
print("Accuracy is: ",accuracy_score(pred_test,y_test))
print("Cross Validation result is: ",cross_val_score(DT, xv, y, cv=10, scoring ='accuracy').mean())
cm=confusion_matrix(y_test,pred_test)
plot_confusion_matrix(cm, classes=['FAKE', 'REAL'])
print(classification_report(y_test, pred_test))
```

```
Bias is :  0.0
Variance is:  0.19999999999999996
Accuracy is:  0.8
Cross Validation result is:  0.8472
Confusion matrix, without normalization
              precision    recall  f1-score   support

           0       0.86      0.76      0.80       543
           1       0.75      0.85      0.80       457

    accuracy                           0.80      1000
   macro avg       0.80      0.80      0.80      1000
weighted avg       0.81      0.80      0.80      1000
```



SUPPORT VECTOR CLASSIFIER

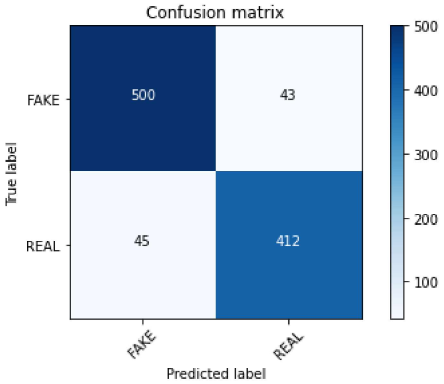
In [34]:

```
svc = SVC()
svc.fit(xv_train,y_train)
pred_train = svc.predict(xv_train)
pred_test = svc.predict(xv_test)
print("Bias is : ",1-accuracy_score(pred_train,y_train))
print("Variance is: ",1-accuracy_score(pred_test,y_test))
print("Accuracy is: ",accuracy_score(pred_test,y_test))
print("Cross Validation result is: ",cross_val_score(svc, xv, y, cv=10, scoring ='accuracy').mean())
cm=confusion_matrix(y_test,pred_test)
plot_confusion_matrix(cm, classes=['FAKE', 'REAL'])
print(classification_report(y_test, pred_test))
```

```
Bias is :  0.032749999999999946
Variance is:  0.08799999999999997
Accuracy is:  0.912
Cross Validation result is:  0.9152000000000001
Confusion matrix, without normalization
              precision    recall  f1-score   support

           0       0.92      0.92      0.92       543
           1       0.91      0.90      0.90       457

    accuracy                           0.91      1000
   macro avg       0.91      0.91      0.91      1000
weighted avg       0.91      0.91      0.91      1000
```



ADA BOOST CLASSIFIER (INBUILT ENSEMBLING)

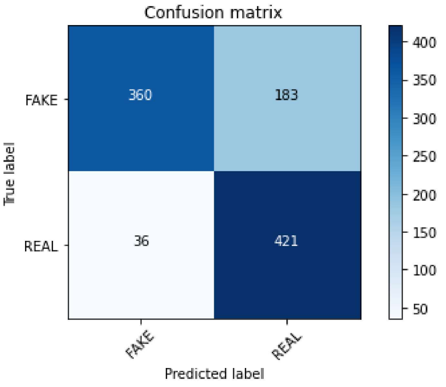
In [35]:

```
ada = AdaBoostClassifier()
ada.fit(xv_train, y_train)
pred_train = ada.predict(xv_train)
pred_test = ada.predict(xv_test)
print("Bias is : ",1-accuracy_score(pred_train,y_train))
print("Variance is: ",1-accuracy_score(pred_test,y_test))
print("Accuracy is: ",accuracy_score(pred_test,y_test))
print("Cross Validation result is: ",cross_val_score(ada, xv, y, cv=10, scoring ='accuracy').mean())
cm=confusion_matrix(y_test,pred_test)
plot_confusion_matrix(cm, classes=['FAKE', 'REAL'])
print(classification_report(y_test, pred_test))
```

```
Bias is :  0.16825
Variance is:  0.21899999999999997
Accuracy is:  0.781
Cross Validation result is:  0.8106
Confusion matrix, without normalization
              precision    recall  f1-score   support

           0       0.91      0.66      0.77       543
           1       0.70      0.92      0.79       457

    accuracy                           0.78      1000
   macro avg       0.80      0.79      0.78      1000
weighted avg       0.81      0.78      0.78      1000
```



GRADIENT BOOSTING CLASSIFIER (IN BUILT ENSEMBLING)

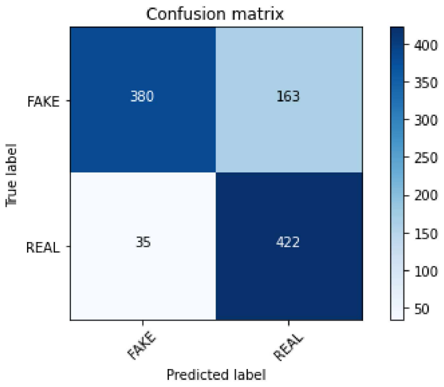
```
In [36]:
```

```
GBC = GradientBoostingClassifier()
GBC.fit(xv_train, y_train)
pred_train = GBC.predict(xv_train)
pred_test = GBC.predict(xv_test)
print("Bias is : ",1-accuracy_score(pred_train,y_train))
print("Variance is: ",1-accuracy_score(pred_test,y_test))
print("Accuracy is: ",accuracy_score(pred_test,y_test))
print("Cross Validation result is: ",cross_val_score(GBC, xv, y, cv=10, scoring ='accuracy').mean())
cm=confusion_matrix(y_test,pred_test)
plot_confusion_matrix(cm, classes=['FAKE', 'REAL'])
print(classification_report(y_test, pred_test))
```

```
Bias is :  0.14200000000000002
Variance is:  0.19799999999999995
Accuracy is:  0.802
Cross Validation result is:  0.8253999999999999
Confusion matrix, without normalization
              precision    recall  f1-score   support

           0       0.92      0.70      0.79       543
           1       0.72      0.92      0.81       457

    accuracy                           0.80      1000
   macro avg       0.82      0.81      0.80      1000
weighted avg       0.83      0.80      0.80      1000
```



RANDOM FOREST CLASSIFIER (INBUILT ENSEMBLING)

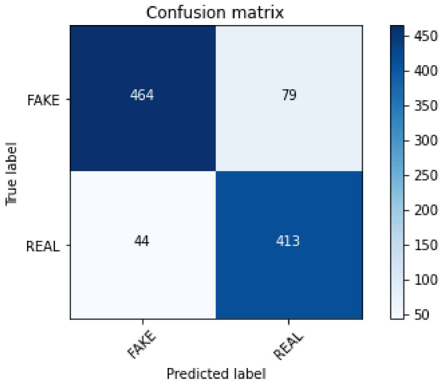
In [37]:

```
RFC = RandomForestClassifier()
RFC.fit(xv_train, y_train)
pred_train = RFC.predict(xv_train)
pred_test = RFC.predict(xv_test)
print("Bias is : ",1-accuracy_score(pred_train,y_train))
print("Variance is: ",1-accuracy_score(pred_test,y_test))
print("Accuracy is: ",accuracy_score(pred_test,y_test))
print("Cross Validation result is: ",cross_val_score(RFC, xv, y, cv=10, scoring ='accuracy').mean())
cm=confusion_matrix(y_test,pred_test)
plot_confusion_matrix(cm, classes=['FAKE', 'REAL'])
print(classification_report(y_test, pred_test))
```

```
Bias is :  0.0
Variance is:  0.123
Accuracy is:  0.877
Cross Validation result is:  0.9002000000000001
Confusion matrix, without normalization
              precision    recall  f1-score   support

           0       0.91      0.85      0.88       543
           1       0.84      0.90      0.87       457

    accuracy                           0.88      1000
   macro avg       0.88      0.88      0.88      1000
weighted avg       0.88      0.88      0.88      1000
```



MAXIMUM VOTING CLASSIFIER (CUSTOM ENSEMBLING)

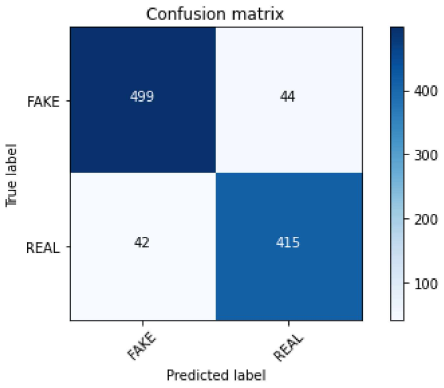
In [38]:

```
knn=KNeighborsClassifier(n_neighbors=25)
knn.fit(xv_train, y_train)
LR = LogisticRegression()
LR.fit(xv_train,y_train)
svc = SVC()
svc.fit(xv_train, y_train)
models = list()
logistic_regression = Pipeline([('m', LogisticRegression())])
models.append(('logistic', logistic_regression))
svc = Pipeline([('m', SVC())])
models.append(('svc', svc))
k_n_n = Pipeline([('m', KNeighborsClassifier(n_neighbors=3))])
models.append(('knn', k_n_n))
maxvoting = VotingClassifier(estimators=models, voting='hard')
maxvoting.fit(xv_train,y_train)
pred_train = maxvoting.predict(xv_train)
pred_test = maxvoting.predict(xv_test)
print("Bias is : ",1-accuracy_score(pred_train,y_train))
print("Variance is: ",1-accuracy_score(pred_test,y_test))
print("Accuracy is: ",accuracy_score(pred_test,y_test))
print("Cross Validation result is: ",cross_val_score(maxvoting, xv, y, cv=10, scoring ='accuracy').mean())
cm=confusion_matrix(y_test,pred_test)
plot_confusion_matrix(cm, classes=['FAKE', 'REAL'])
print(classification_report(y_test, pred_test))
```

```
Bias is :  0.014000000000000012
Variance is:  0.08599999999999997
Accuracy is:  0.914
Cross Validation result is:  0.9188000000000001
Confusion matrix, without normalization
              precision    recall  f1-score   support

           0       0.92      0.92      0.92       543
           1       0.90      0.91      0.91       457

    accuracy                           0.91      1000
   macro avg       0.91      0.91      0.91      1000
weighted avg       0.91      0.91      0.91      1000
```



STACKING (CUSTOM ENSEMBLING)

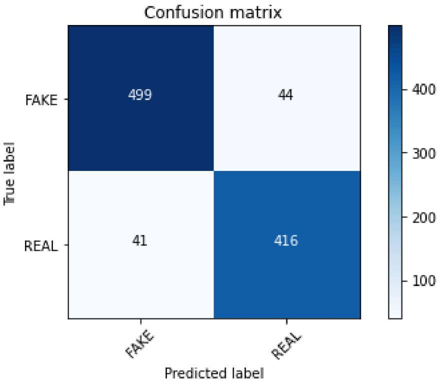
In [39]:

```
base1=SVC()
base2=KNeighborsClassifier(n_neighbors=25)
meta_model=LogisticRegression()
stack=StackingClassifier(classifiers=[base1,base2],meta_classifier=meta_model)
stack.fit(xv_train,y_train)
pred_train = stack.predict(xv_train)
pred_test = stack.predict(xv_test)
print("Bias is : ",1-accuracy_score(pred_train,y_train))
print("Variance is: ",1-accuracy_score(pred_test,y_test))
print("Accuracy is: ",accuracy_score(pred_test,y_test))
print("Cross Validation result is: ",cross_val_score(stack, xv, y, cv=10, scoring ='accuracy').mean())
cm=confusion_matrix(y_test,pred_test)
plot_confusion_matrix(cm, classes=['FAKE', 'REAL'])
print(classification_report(y_test, pred_test))
```

```
Bias is :  0.0010000000000000009
Variance is:  0.08499999999999996
Accuracy is:  0.915
Cross Validation result is:  0.9200000000000002
Confusion matrix, without normalization
              precision    recall  f1-score   support

           0       0.92      0.92      0.92       543
           1       0.90      0.91      0.91       457

    accuracy                           0.92      1000
   macro avg       0.91      0.91      0.91      1000
weighted avg       0.92      0.92      0.92      1000
```


Confusion matrix
FAKE
REAL
True label
FAKE
REAL
Predicted label