AIM

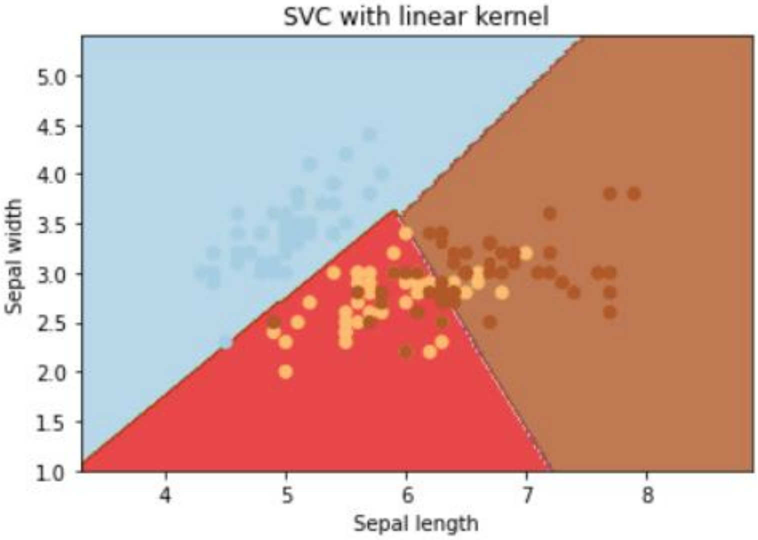
Program to implement text classification using Support vector machine.

Programming code:

Dataset used: iris.csv

```
import numpy as np
import matplotlib.pyplot as plt
from sklearn import svm, datasets

# import some data to play with
iris = datasets.load_iris()
X = iris.data[:, :2] # we only take the first two features. We coul
d
 # avoid this ugly slicing by using a two-dim dataset
y = iris.target

# we create an instance of SVM and fit out data. We do not scale ou
r
# data since we want to plot the support vectors
C = 1.0 # SVM regularization parameter

svc = svm.SVC(kernel='linear', C=1,gamma='auto').fit(X, y)

# create a mesh to plot in
#x_min, x_max = X[:, 0].min() - 1, X[:, 0].max() + 1
#y_min, y_))max = X[:, 1].min() - 1, X[:, 1].max() + 1
#h = (x_max / x_min)/100
#xx, yy = np.meshgrid(np.arange(x_min, x_max, h),
#np.arange(y_min, y_max, h


plt.subplot(1, 1, 1)
Z = svc.predict(np.c_ravel[xx.(), yy.ravel()])
Z = Z.reshape(xx.shape)
plt.contourf(xx, yy, Z, cmap=plt.cm.Paired, alpha=0.8)

plt.scatter(X[:, 0], X[:, 1], c=y, cmap=plt.cm.Paired)
plt.xlabel('Sepal length')
plt.ylabel('Sepal width')
plt.xlim(xx.min(), xx.max())
plt.title('SVC with linear kernel')
plt.show()
```

OUTPUT:



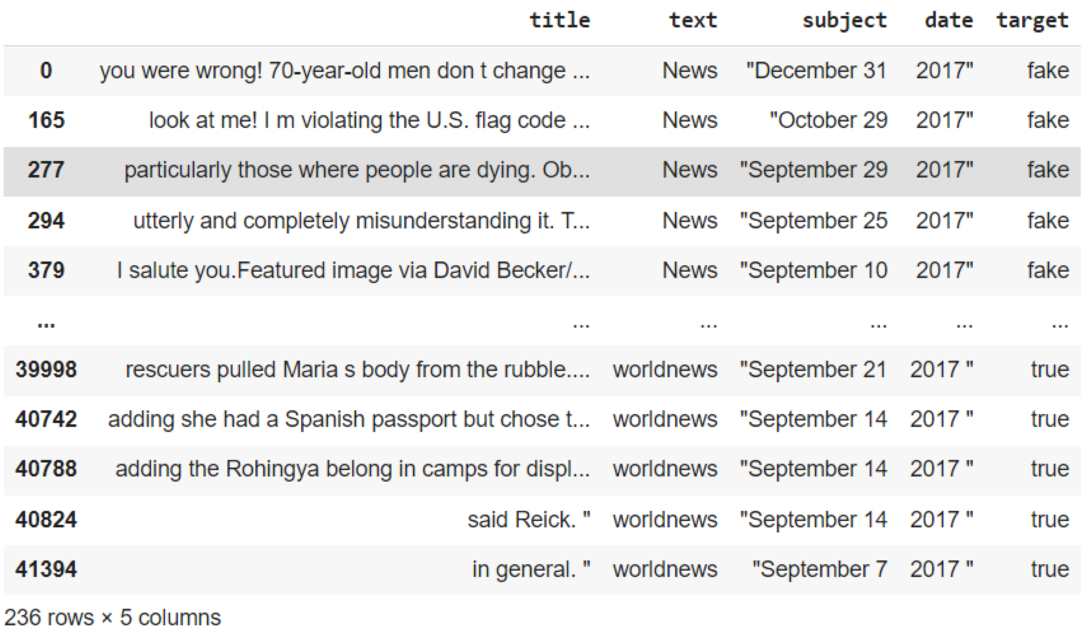
Programming code:

Dataset used: True.csv, Fake.csv

```
 #Importing Libraries
import pandas as pd
import numpy as np
from sklearn.model_selection import train_test_split
from sklearn.pipeline import Pipeline
from sklearn.feature_extraction.text import CountVectorizer
from sklearn.feature_extraction.text import TfidfTransformer
from sklearn.metrics import accuracy_score, confusion_matrix,classi
fication_report

from sklearn.svm import LinearSVC

import csv
true = pd.read_csv("True.csv")
fake = pd.read_csv("Fake.csv")
```

```
 fake['target'] = 'fake'
true['target'] = 'true'
#News dataset
news = pd.concat([fake, true]).reset_index(drop = True)
news.head()
news.dropna()
```

OUTPUT:



	title	text	subject	date	target
0	you were wrong! 70-year-old men don t change ...	News	"December 31	2017"	fake
165	look at me! I m violating the U.S. flag code ...	News	"October 29	2017"	fake
277	particularly those where people are dying. Ob...	News	"September 29	2017"	fake
294	utterly and completely misunderstanding it. T...	News	"September 25	2017"	fake
379	I salute you.Featured image via David Becker/...	News	"September 10	2017"	fake
...	...	...	...	...	...
39998	rescuers pulled Maria s body from the rubble....	worldnews	"September 21	2017 "	true
40742	adding she had a Spanish passport but chose t...	worldnews	"September 14	2017 "	true
40788	adding the Rohingya belong in camps for displ...	worldnews	"September 14	2017 "	true
40824	said Reick. "	worldnews	"September 14	2017 "	true
41394	in general. "	worldnews	"September 7	2017 "	true

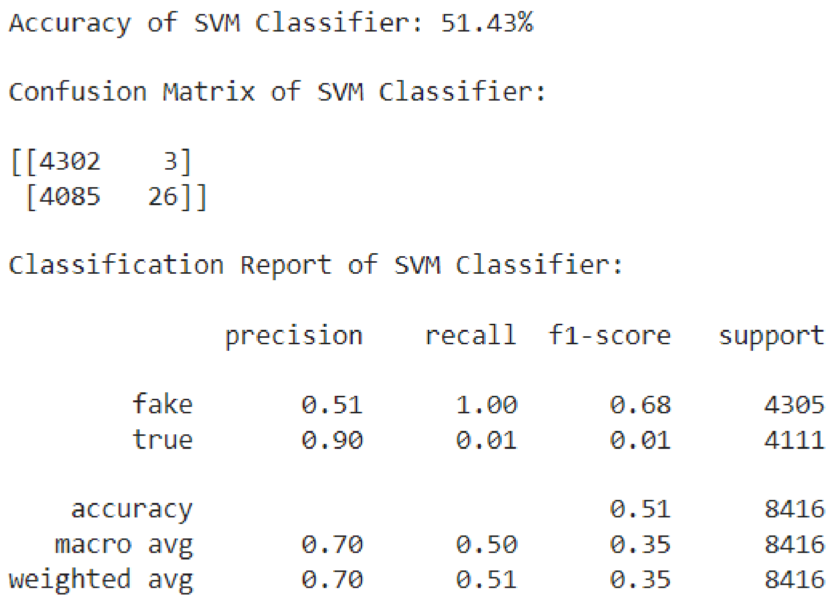
236 rows × 5 columns

Programming code:

```
#Train-test split
x_train,x_test,y_train,y_test = train_test_split(news['text'], news
.target, test_size=0.2, random_state=1)


#Term frequency(TF)=count(word)/total(words)6+   0ZXCVBNM,./
#TF-
IDF: we can even reduce the weightage of more common words like (th
e, is, an etc.) which occurs in all document.
#This is called as TF-
IDF i.e Term Frequency times inverse document frequency.
#count vectorizer : involves counting the number of occurrences eac
h word appears in a document
```

```
pipe2 = Pipeline([('vect', CountVectorizer()), ('tfidf', TfidfTrans
former()), ('model', LinearSVC())])


model_svc = pipe2.fit(x_train.astype('U'), y_train.astype('U'))
svc_pred = model_svc.predict(x_test.astype('U'))

print("Accuracy of SVM Classifier: {}%".format(round(accuracy_score
(y_test, svc_pred)*100,2)))
print("\nConfusion Matrix of SVM Classifier:\n")
print(confusion_matrix(y_test, svc_pred))
print("\nClassification Report of SVM Classifier:\n")
print(classification_report(y_test, svc_pred))
```

OUTPUT:



```
Accuracy of SVM Classifier: 51.43%

Confusion Matrix of SVM Classifier:

[[4302    3]
 [4085   26]]

Classification Report of SVM Classifier:

              precision    recall  f1-score   support

        fake       0.51      1.00      0.68      4305
        true       0.90      0.01      0.01      4111

    accuracy                           0.51      8416
   macro avg       0.70      0.50      0.35      8416
weighted avg       0.70      0.51      0.35      8416
```