Configuration Manual

MSc Research Project

MSc in Cybersecurity

Shiva Prasad Bonu

Student ID: x20169850

School of Computing

National College of Ireland

Supervisor: Imran khan

National College of Ireland

MSc Project Submission Sheet

School of Computing

Student Name:	BONU SHIVA PRASAD
Student ID:	X20169850
Programme:	MSc in Cybersecurity **Year:** 2020-2021
Module:	RESEARCH PROJECT
Lecturer:	IMRAN KHAN
Submission Due Date:	……31-01-2022…….……
Project Title:	IMPROVING THE DETECTION OF EMAIL SPAM FILTER USING LGS-COUNT MODEL
Word Count:	**920** **Page Count:** ……36……..…….……

I hereby certify that the information contained in this (my submission) is information pertaining to research I conducted for this project. All information other than my own contribution will be fully referenced and listed in the relevant bibliography section at the rear of the project.
ALL internet material must be referenced in the bibliography section. Students are required to use the Referencing Standard specified in the report template. To use other author's written or electronic work is illegal (plagiarism) and may result in disciplinary action.

Signature: ……BONU SHIVA PRASAD……

Date: ……31-01-2022……

PLEASE READ THE FOLLOWING INSTRUCTIONS AND CHECKLIST

Attach a completed copy of this sheet to each project (including multiple copies)	□
Attach a Moodle submission receipt of the online project submission, to each project (including multiple copies).	□
You must ensure that you retain a HARD COPY of the project, both for your own reference and in case a project is lost or mislaid. It is not sufficient to keep a copy on computer.	□

Assignments that are submitted to the Programme Coordinator Office must be placed into the assignment box located outside the office.

Office Use Only	
Signature:	
Date:	
Penalty Applied (if applicable):	

CONFIGURATION MANUAL

1. Introduction

This configuration manual gives a detailed information regarding all necessary hardware and software setup required to build the entire system from scratch. The configuration manual will help in replicating the research done by using a more practical way. We are going to evaluate the complete working of the system with its user interface.

This document provides the steps to replicate the implementation of the research "**Spam Email Detection using Machine Learning**". The main aim of the research is to detect Spam emails for the dataset using two different Test processing approach : Count Vectorization & TF-IDF. For building this model, involves implementing various machine learning algorithms like Naive Bayes, Logistic Regression, AdaBoost & Random Forest Classifiers. This manual is organized in sections and the process is explained from data collection to evaluation of the results. The following are structured as the environment for conducting this research has been provided in the section 2, then the preparation of data and procedure for handling them is provided in section 3. Finally, the implementation and evaluation of the research is provided in section 4.

The Configuration Manual will be divided into six sections excluding this introduction part.

- Environmental Setup
- Libraries Required
- Dataset
- User Interface
- Implementation
- Code Repository

2. Environmental Setup

2.1 Hardware Requirements

- 8GB RAM.
- 250 GB HDD.
- 2.2 GHz Intel. Core i5

2.2 Software Requirements

- Windows 10
- Python 3.6.3

2.3 Programming Prerequisites

- Python(Version 3.6.3)
- Visual Studio Code IDE

3. Libraries Required

All the libraries required for building this research project are mentioned in table 1 along with their usage:

pandas	It offers data structures and operations for manipulating numerical tables and time series
numpy	It is used for working with arrays. It also has functions for working in domain of linear algebra, Fourier transform, and matrices
sklearn	Used for implementing various machine learning algorithms : Gaussian Naiive Bayes, Support vector classifier, LogisticRegression
matplotlib	It is used for plotting various graphical visualisations
RE	Package to perform Regular Expression operations
sklearn.metrics	For generating various performance metrics: Accuracy, ROC, Confusion matrix
Os	To perform operating system level tasks for file operations
sklearn.ensemble	Implementing RandomForestClassifier, AdaBoostClassifier,
yellowbrick.classifier	For generating intuitive ClassificationReport
sklearn.feature_extraction.text	Importing TfidfVectorizer & CountVectorizer For text feature extraction
Tkinter	For graphical user interface design

4. Data Set Details

CSDMC2010 SPAM corpus dataset

The proposed dataset has been taken from CSDMC2010 SPAM corpus, through Kaggle repository. This dataset contains spam and ham data folders with a spam count of 2332 files and ham count of 1083 mail files. Rather than making use of complicated hybrid models, our proposed approach utilizes simple working ML algorithms along with the concept of TFI-DF and CV. The dataset which is in the HTML format is further converted into plain text format using the text-processing technique

Link to dataset :

https://www.kaggle.com/c/anomaly-detection-challenges-2015-challenge-4/data

Below is the count model of dataset containing spam and ham files:

Figure 5: Dataset of spam and ham files

5. Implementation & Experiment:

Following are the library imports:

```
import re, os, math, string, json
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import random
import re
import nltk
#nltk.download('stopwords')
#nltk.download('punkt')
from sklearn.model_selection import train_test_split
from sklearn.feature_extraction.text import TfidfVectorizer, CountVectorizer
from sklearn.linear_model import LogisticRegression
from sklearn.naive_bayes import MultinomialNB
from sklearn.ensemble import AdaBoostClassifier,RandomForestClassifier
from sklearn.metrics import confusion_matrix, classification_report
import seaborn as sns
from sklearn.metrics import confusion_matrix
from sklearn.metrics import confusion_matrix,accuracy_score,roc_auc_score,classification,roc_curve
from sklearn.metrics import classification_report
from yellowbrick.classifier import ClassificationReport
from sklearn import preprocessing
#GUI
from PIL import ImageTk,Image
import tkinter as tk
from tkinter import *
from tkinter import filedialog
```

Directory Loading:

```
# Load all of the emails from the "ham" directory
print("- Loading Ham -")

for each in os.listdir('ham'):
    try:
        with open('ham/' + each, 'r') as f:
            corpus.append(f.read())
            labels.append("ham")
    except:
        os.remove('ham/' + each)
        print(each)

# Load all of the emails from the "spam" directory
print("- Loading Spam -")

for each in os.listdir('spam'):
    try:
        with open('spam/'+each, 'r') as f:
            corpus.append(f.read())
            labels.append("spam")
    except:
        os.remove('spam/' + each)
        print(each)
```

Data Pre-processing:

Following are the data preprocessing techniques implemented:

- Stop word removal
- Punctuation detection
- Tokenization
- Stemming – using Porter Stemmer

```
#Tokanization
def tokenizer(text):
    punctuations = list(string.punctuation)
    stopwords = nltk.corpus.stopwords.words('english')
    stemmer = nltk.stem.PorterStemmer()

    tokens = nltk.word_tokenize(text.lower())
    tokens = [i.strip(''.join(punctuations))
              for i in tokens
              if i not in punctuations]
    tokens = [stemmer.stem(i)
              for i in tokens]
    return [w for w in tokens if w not in stopwords and w != ""]
```

Text Feature / Vectorizing the data:

- CountVectorizer

```
# Vectorize the  inputs
print("- Training Count Vectorizer -")
cVec = CountVectorizer(tokenizer=tokenizer)
count_X = cVec.fit_transform(corpus)
```

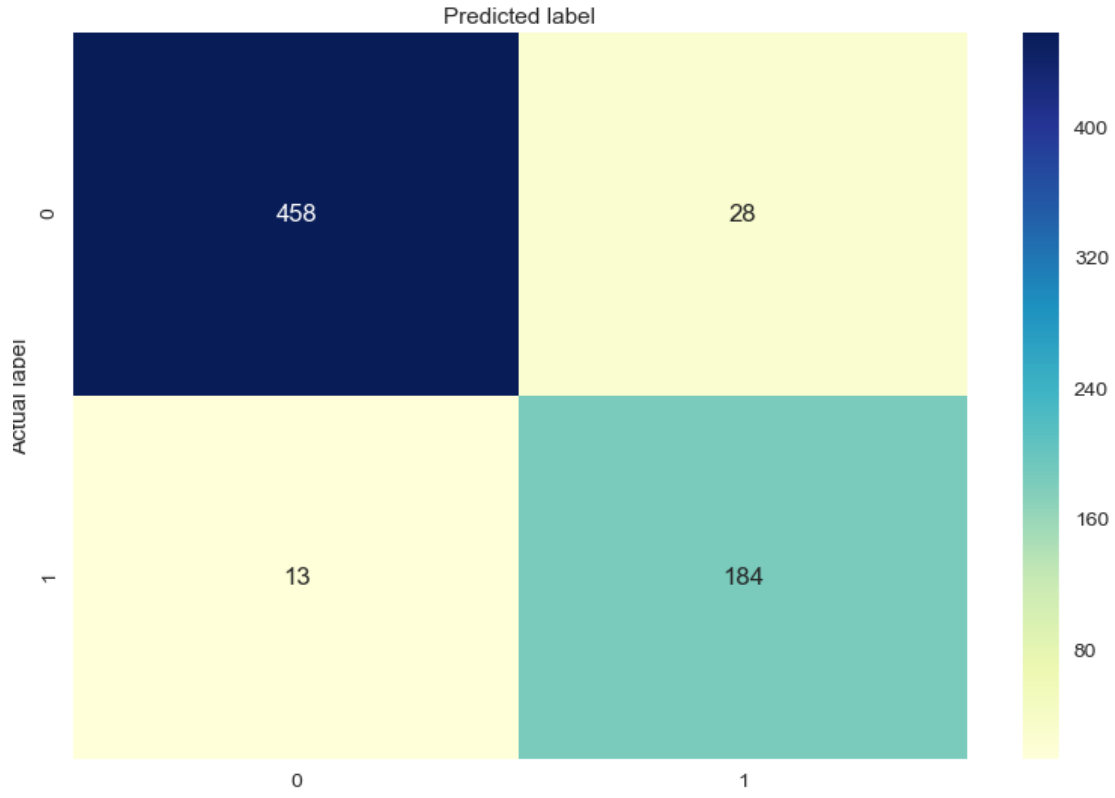
- TF-IDF

```
print("- Training TF-IDF Vectorizer -")
tVec = TfidfVectorizer(tokenizer=tokenizer)
tfidf_X = tVec.fit_transform(corpus)

print("\n### Vectorizing Complete ###\n")
```

Model Implementation:

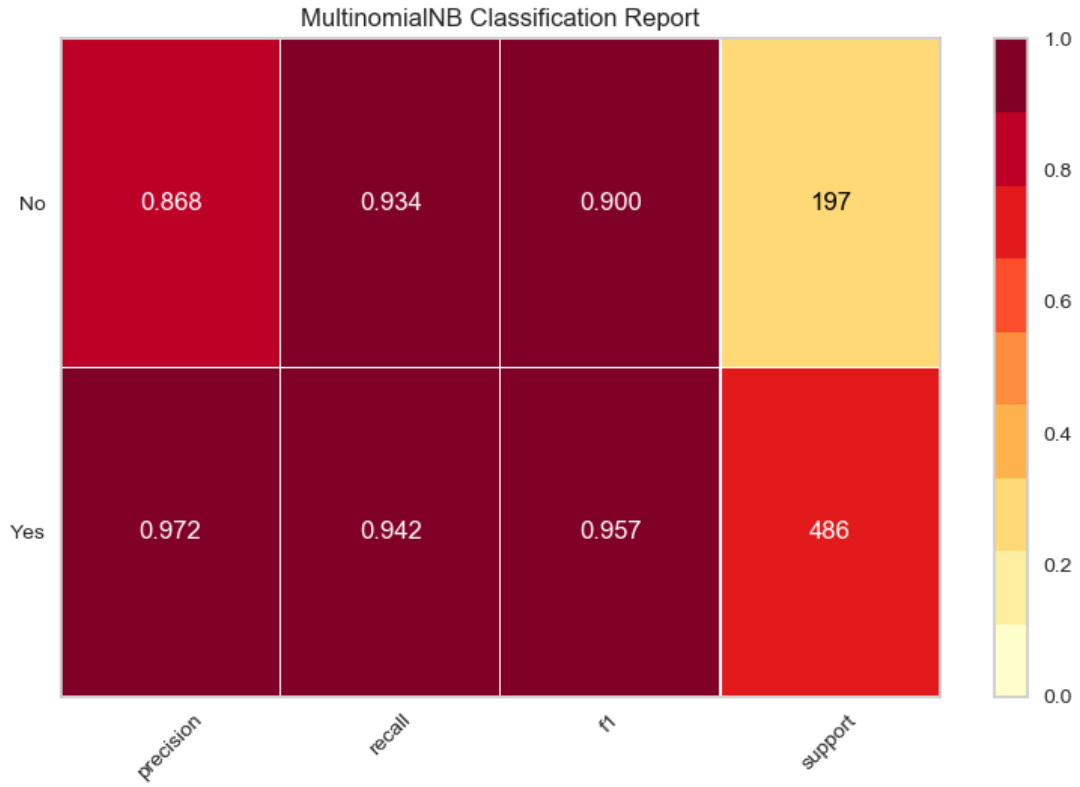
- Experiment 1 – Multinomial Naiive Bayes on CountVectorizer

```
clf = MultinomialNB()
clf.fit(X_train,y_train)
print("MultinomialNB :train set")
y_pred = clf.predict(X_train)
pred=clf.predict_proba(X_test)
print("MultinomialNB using Count Vectorizer :Confusion Matrix: ", confusion_matrix(y_train, y_pr
print ("MultinomialNB using Count Vectorizer :Accuracy : ", accuracy_score(y_train,y_pred)*100)
print("MultinomialNB :Test set")
y_pred = clf.predict(X_test)
print("MultinomialNB using Count Vectorizer :Confusion Matrix: ", confusion_matrix(y_test, y_pre
print ("MultinomialNB using Count Vectorizer :Accuracy : ", accuracy_score(y_test,y_pred)*100)
acc1 =accuracy_score(y_test,y_pred)*100

#confusion Matrix
matrix =confusion_matrix(y_test, y_pred)
class_names=[0,1]
fig, ax = plt.subplots()
tick_marks = np.arange(len(class_names))
plt.xticks(tick_marks, class_names)
plt.yticks(tick_marks, class_names)
sns.heatmap(pd.DataFrame(matrix), annot=True, cmap="YlGnBu" ,fmt='g')
ax.xaxis.set_label_position("top")
plt.tight_layout()
plt.title('Confusion matrix', y=1.1)
plt.ylabel('Actual label')
plt.xlabel('Predicted label')
```



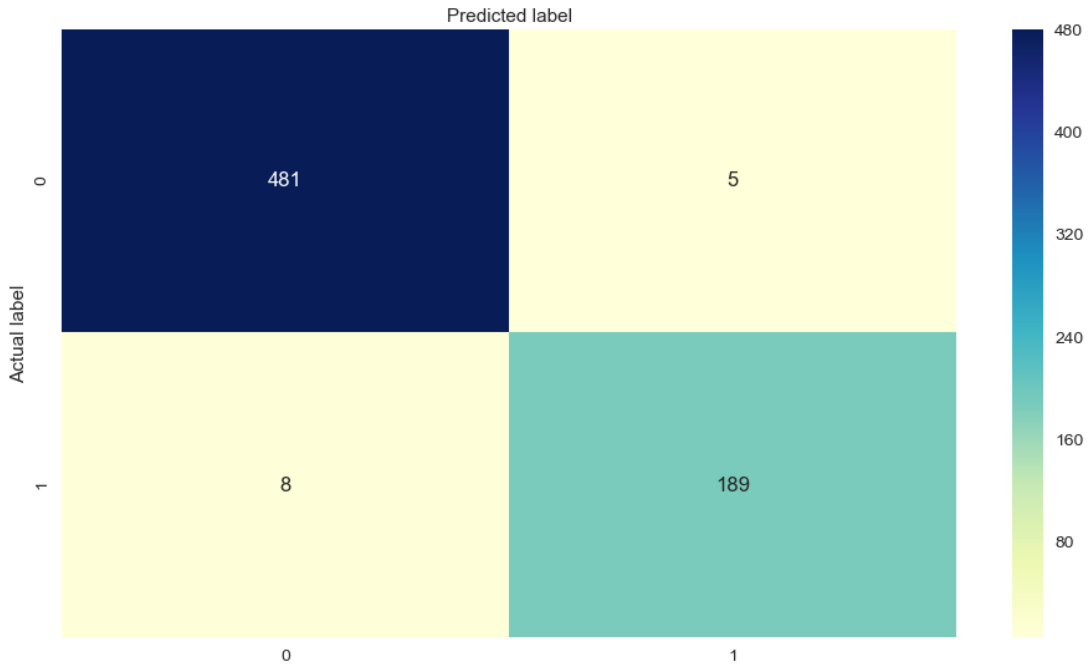
Classification Report:

	precision	recall	f1-score	support
Yes	0.97	0.94	0.96	486
No	0.87	0.93	0.90	197
accuracy			0.94	683
macro avg	0.92	0.94	0.93	683
weighted avg	0.94	0.94	0.94	683



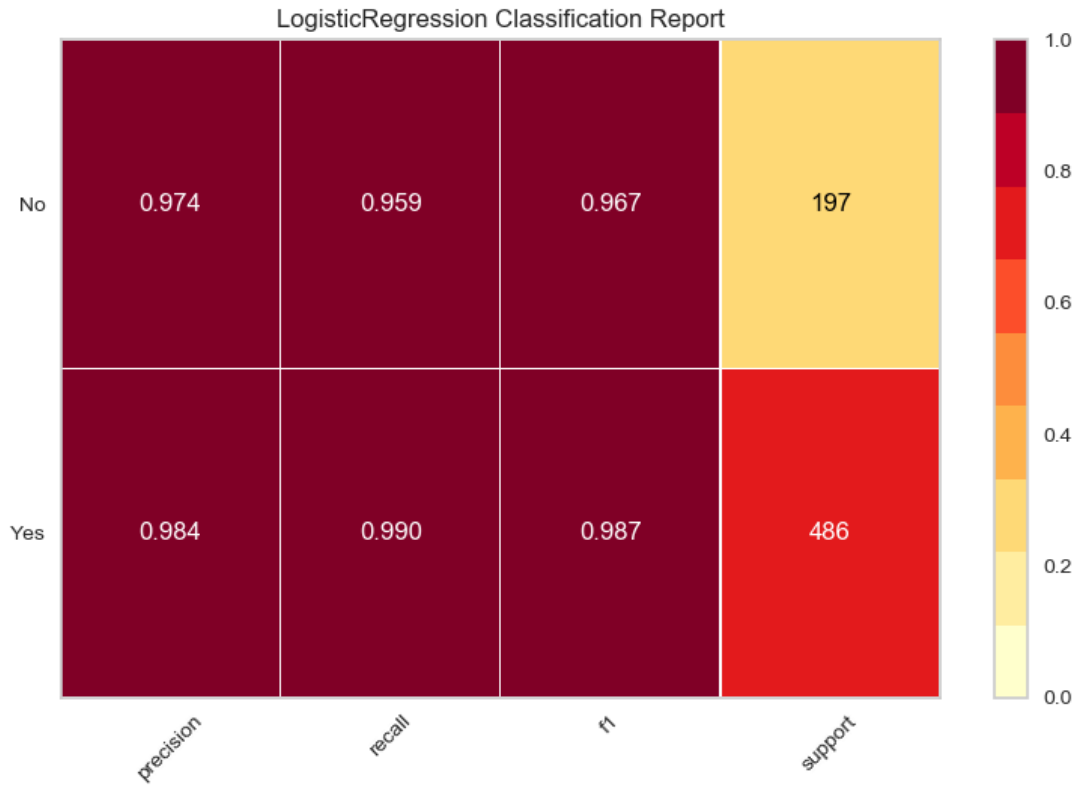
Experiment 2 – LogisticRegression on CountVectorizer

```
def LR_Count_Vec(X_train,y_train,X_test,y_test):
  global acc2
  clf = LogisticRegression()
  clf.fit(X_train,y_train)
  print("LogisticRegression :train set")
  y_pred = clf.predict(X_train)
  pred=clf.predict_proba(X_test)
  print("LogisticRegression using Count Vectorizer :Confusion Matrix: ", confusion_matrix(y_train, y_pred))
  print ("LogisticRegression using Count Vectorizer :Accuracy : ", accuracy_score(y_train,y_pred)*100)
  print("LogisticRegression :Test set")
  y_pred = clf.predict(X_test)
  print("LogisticRegression using Count Vectorizer :Confusion Matrix: ", confusion_matrix(y_test, y_pred))
  print ("LogisticRegression using Count Vectorizer :Accuracy : ", accuracy_score(y_test,y_pred)*100)
  acc2=accuracy_score(y_test,y_pred)*100

  #confusion Matrix
  matrix =confusion_matrix(y_test, y_pred)
  class_names=[0,1]
  fig, ax = plt.subplots()
  tick_marks = np.arange(len(class_names))
  plt.xticks(tick_marks, class_names)
  plt.yticks(tick_marks, class_names)
  sns.heatmap(pd.DataFrame(matrix), annot=True, cmap="YlGnBu" ,fmt='g')
  ax.xaxis.set_label_position("top")
  plt.tight_layout()
  plt.title('Confusion matrix', y=1.1)
  plt.ylabel('Actual label')
```


Predicted label
Actual label
0
1
481
5
8
189
0
1
480
400
320
240
160
80

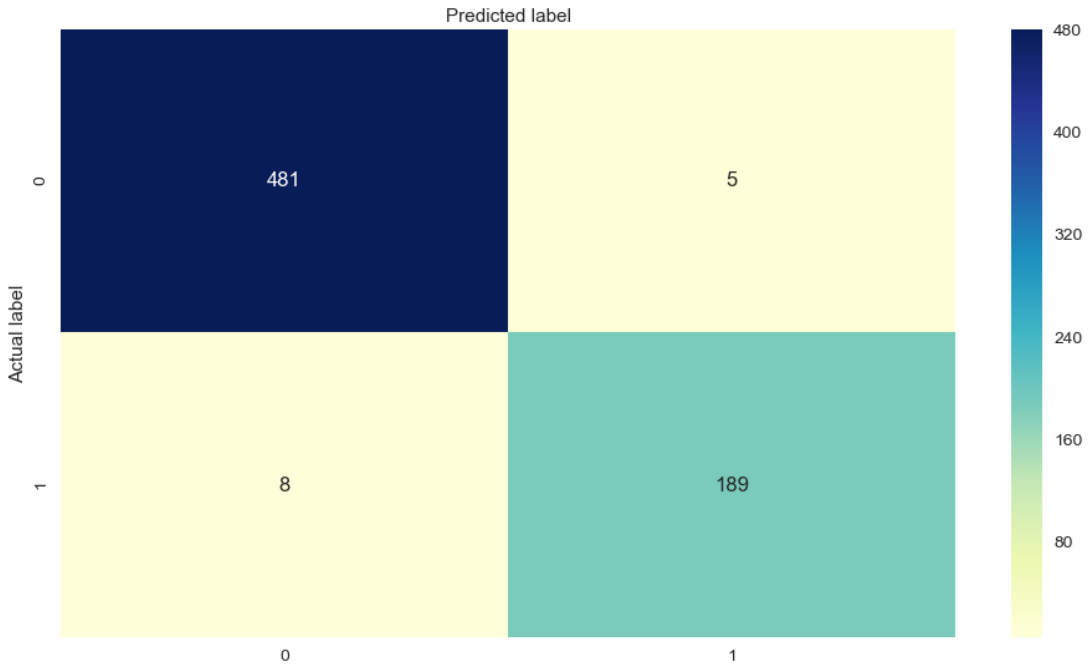
Classification Report:

	precision	recall	f1-score	support
Yes	0.98	0.99	0.99	486
No	0.97	0.96	0.97	197
accuracy			0.98	683
macro avg	0.98	0.97	0.98	683
weighted avg	0.98	0.98	0.98	683



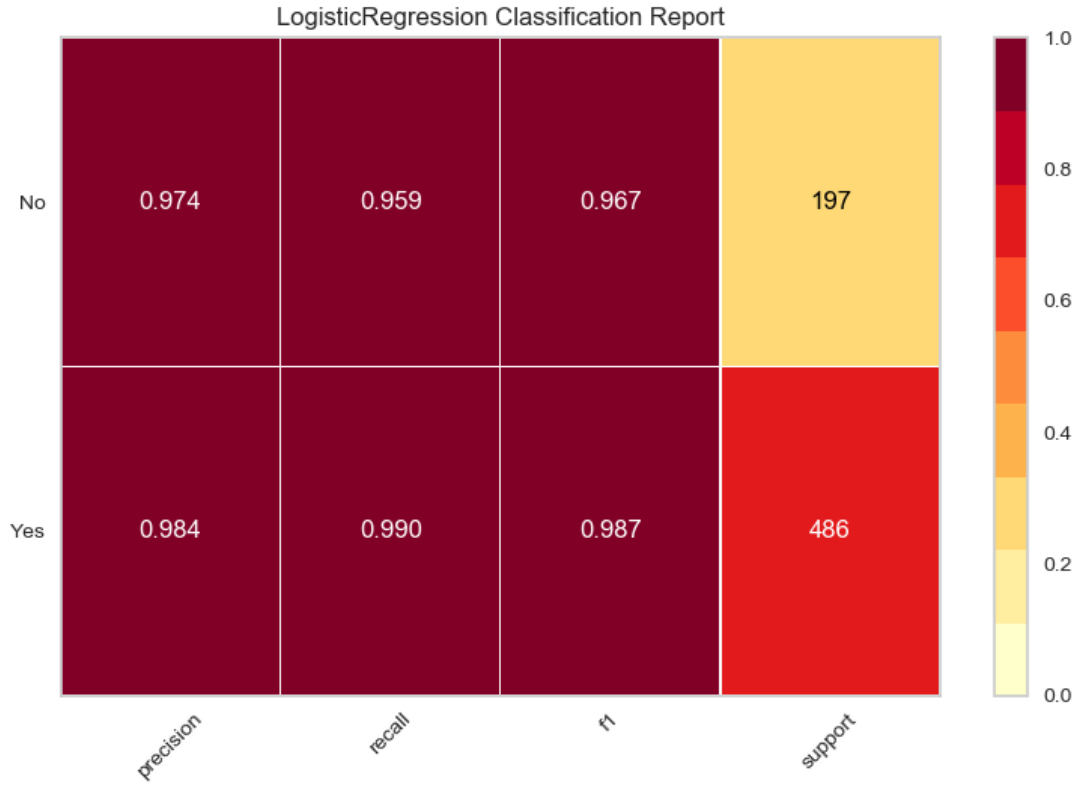
Experiment 2 – LogisticRegression on CountVectorizer

```
def LR_Count_Vec(X_train,y_train,X_test,y_test):
  global acc2
  clf = LogisticRegression()
  clf.fit(X_train,y_train)
  print("LogisticRegression :train set")
  y_pred = clf.predict(X_train)
  pred=clf.predict_proba(X_test)
  print("LogisticRegression using Count Vectorizer :Confusion Matrix: ", confusion_matrix(y_train, y_pred))
  print ("LogisticRegression using Count Vectorizer :Accuracy : ", accuracy_score(y_train,y_pred)*100)
  print("LogisticRegression :Test set")
  y_pred = clf.predict(X_test)
  print("LogisticRegression using Count Vectorizer :Confusion Matrix: ", confusion_matrix(y_test, y_pred))
  print ("LogisticRegression using Count Vectorizer :Accuracy : ", accuracy_score(y_test,y_pred)*100)
  acc2=accuracy_score(y_test,y_pred)*100

  #confusion Matrix
  matrix =confusion_matrix(y_test, y_pred)
  class_names=[0,1]
  fig, ax = plt.subplots()
  tick_marks = np.arange(len(class_names))
  plt.xticks(tick_marks, class_names)
  plt.yticks(tick_marks, class_names)
  sns.heatmap(pd.DataFrame(matrix), annot=True, cmap="YlGnBu" ,fmt='g')
  ax.xaxis.set_label_position("top")
  plt.tight_layout()
  plt.title('Confusion matrix', y=1.1)
  plt.ylabel('Actual label')
```



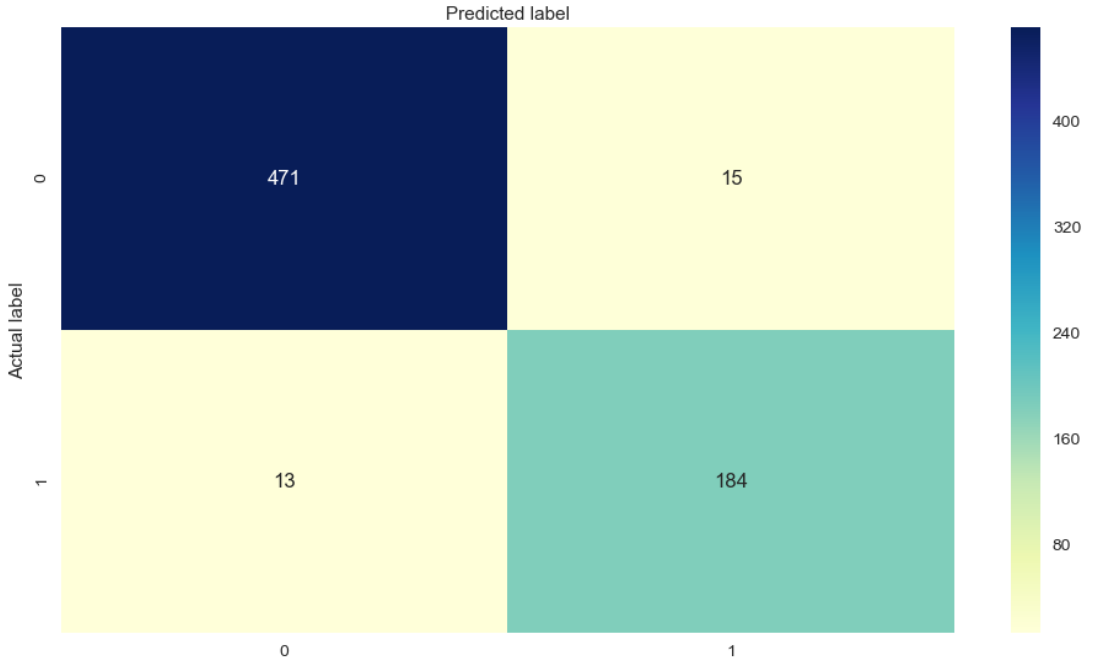
Classification Report:

	precision	recall	f1-score	support
Yes	0.98	0.99	0.99	486
No	0.97	0.96	0.97	197
accuracy			0.98	683
macro avg	0.98	0.97	0.98	683
weighted avg	0.98	0.98	0.98	683



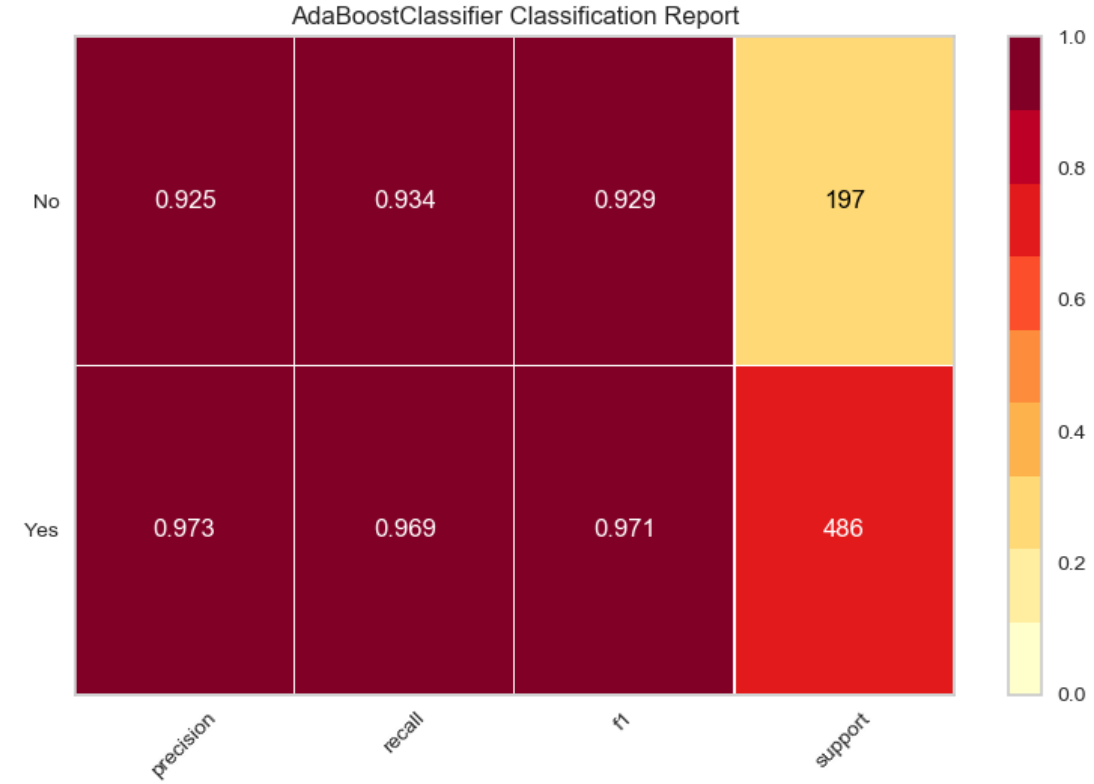
Experiment 3 – Adaboost on CountVectorizer

```
def Ada_Count_Vec(X_train,y_train,X_test,y_test):
  global acc3
  clf = AdaBoostClassifier()
  clf.fit(X_train,y_train)
  print("AdaBoost :train set")
  y_pred = clf.predict(X_train)
  pred=clf.predict_proba(X_test)
  print("AdaBoost using Count Vectorizer :Confusion Matrix: ", confusion_matrix(y_train, y_pred))
  print ("AdaBoost using Count Vectorizer :Accuracy : ", accuracy_score(y_train,y_pred)*100)
  print("AdaBoost :Test set")
  y_pred = clf.predict(X_test)
  print("AdaBoost using Count Vectorizer :Confusion Matrix: ", confusion_matrix(y_test, y_pred))
  print ("AdaBoost using Count Vectorizer :Accuracy : ", accuracy_score(y_test,y_pred)*100)
  acc3=accuracy_score(y_test,y_pred)*100

  #confusion Matrix
  matrix =confusion_matrix(y_test, y_pred)
  class_names=[0,1]
  fig, ax = plt.subplots()
  tick_marks = np.arange(len(class_names))
  plt.xticks(tick_marks, class_names)
  plt.yticks(tick_marks, class_names)
  sns.heatmap(pd.DataFrame(matrix), annot=True, cmap="YlGnBu" ,fmt='g')
  ax.xaxis.set_label_position("top")
  plt.tight_layout()
  plt.title('Confusion matrix', y=1.1)
  plt.ylabel('Actual label')
```



Classification report:.

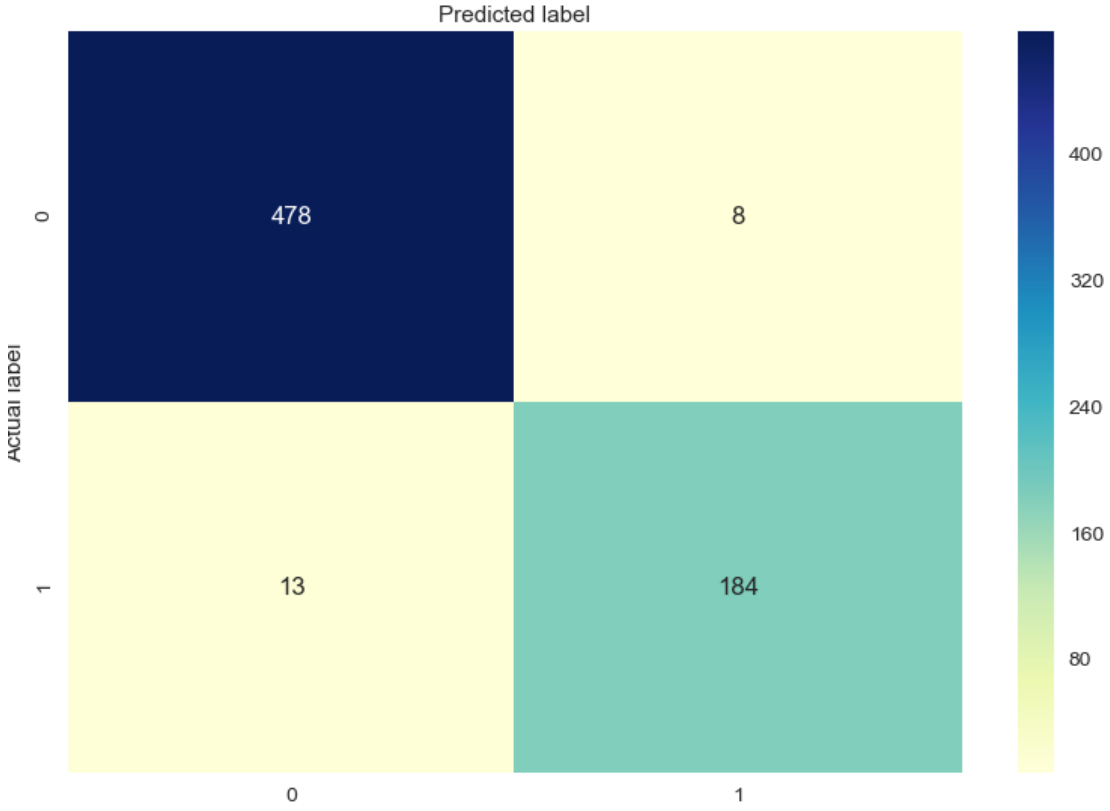
	precision	recall	f1-score	support
Yes	0.97	0.97	0.97	486
No	0.92	0.93	0.93	197
accuracy			0.96	683
macro avg	0.95	0.95	0.95	683
weighted avg	0.96	0.96	0.96	683

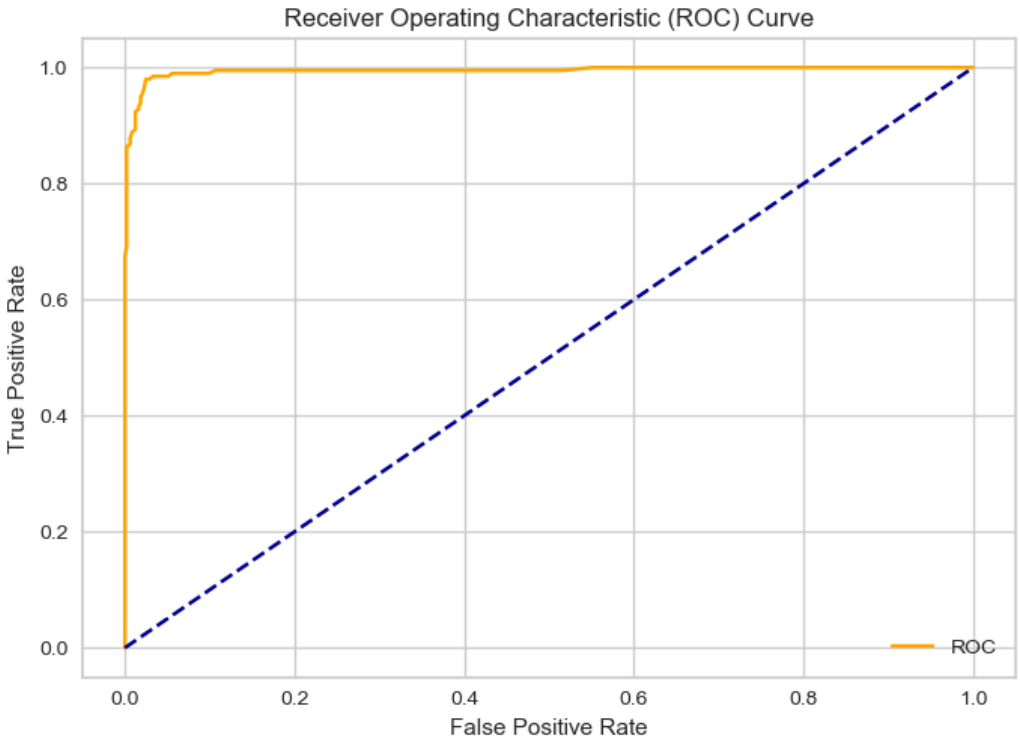


Experiment 4 – RandomForest on CountVectorizer

```
def RF_Count_Vec(X_train,y_train,X_test,y_test):
  global acc4
  clf = RandomForestClassifier()
  clf.fit(X_train,y_train)
  print("Random Forest :train set")
  y_pred = clf.predict(X_train)
  pred=clf.predict_proba(X_test)
  print("Random Forest using Count Vectorizer :Confusion Matrix: ", confusion_matrix(y_train, y_pred))
  print ("Random Forest using Count Vectorizer :Accuracy : ", accuracy_score(y_train,y_pred)*100)
  print("Random Forest :Test set")
  y_pred = clf.predict(X_test)
  print("Random Forest using Count Vectorizer :Confusion Matrix: ", confusion_matrix(y_test, y_pred))
  print ("Random Forest using Count Vectorizer :Accuracy : ", accuracy_score(y_test,y_pred)*100)
  acc4=accuracy_score(y_test,y_pred)*100

  #confusion Matrix
  matrix =confusion_matrix(y_test, y_pred)
  class_names=[0,1]
  fig, ax = plt.subplots()
  tick_marks = np.arange(len(class_names))
  plt.xticks(tick_marks, class_names)
  plt.yticks(tick_marks, class_names)
  sns.heatmap(pd.DataFrame(matrix), annot=True, cmap="YlGnBu" ,fmt='g')
  ax.xaxis.set_label_position("top")
  plt.tight_layout()
  plt.title('Confusion matrix', y=1.1)
  plt.ylabel('Actual label')
```

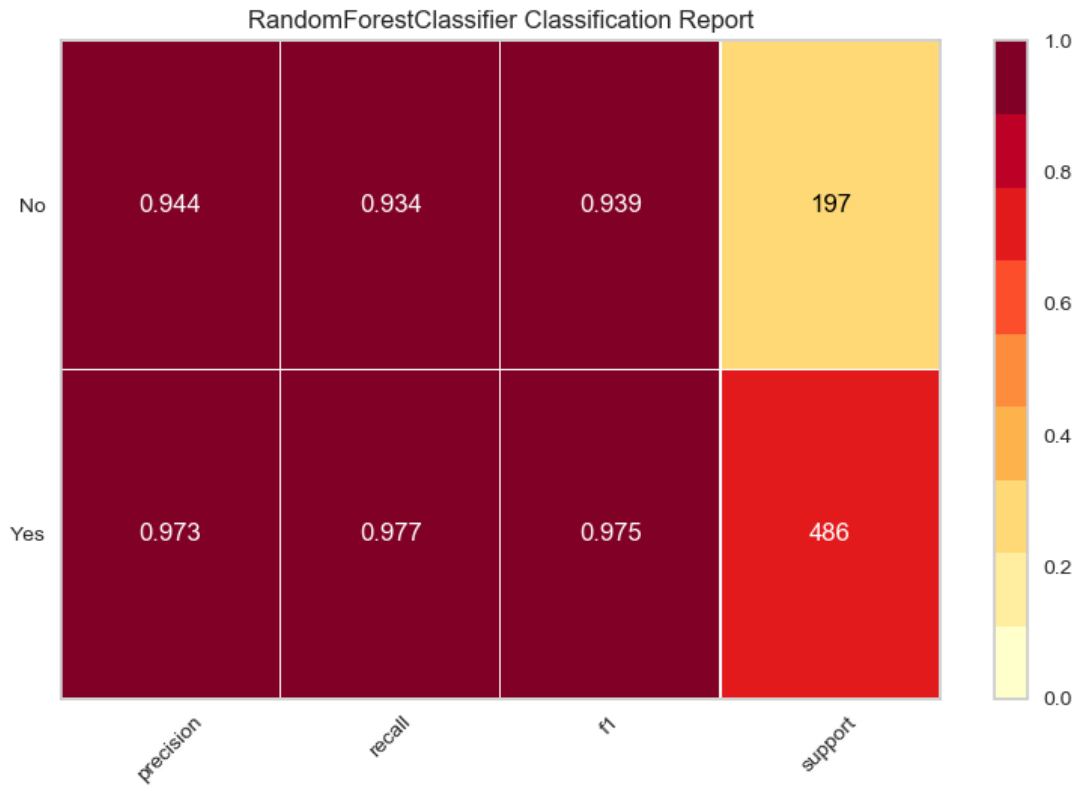
Random Forest using Count Vectorizer :Accuracy : 96.92532942898976



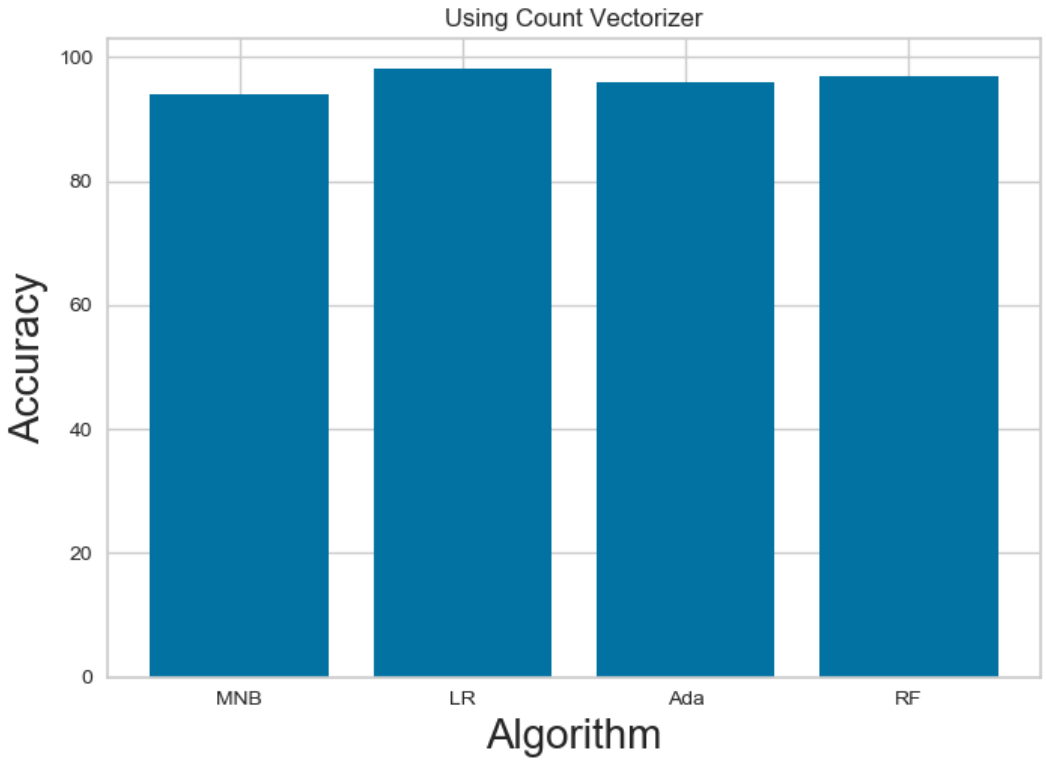

Receiver Operating Characteristic (ROC) Curve
True Positive Rate
False Positive Rate
0.0
0.2
0.4
0.6
0.8
1.0
ROC

Classification report

	precision	recall	f1-score	support
Yes	0.97	0.98	0.98	486
No	0.96	0.93	0.95	197
accuracy			0.97	683
macro avg	0.97	0.96	0.96	683
weighted avg	0.97	0.97	0.97	683



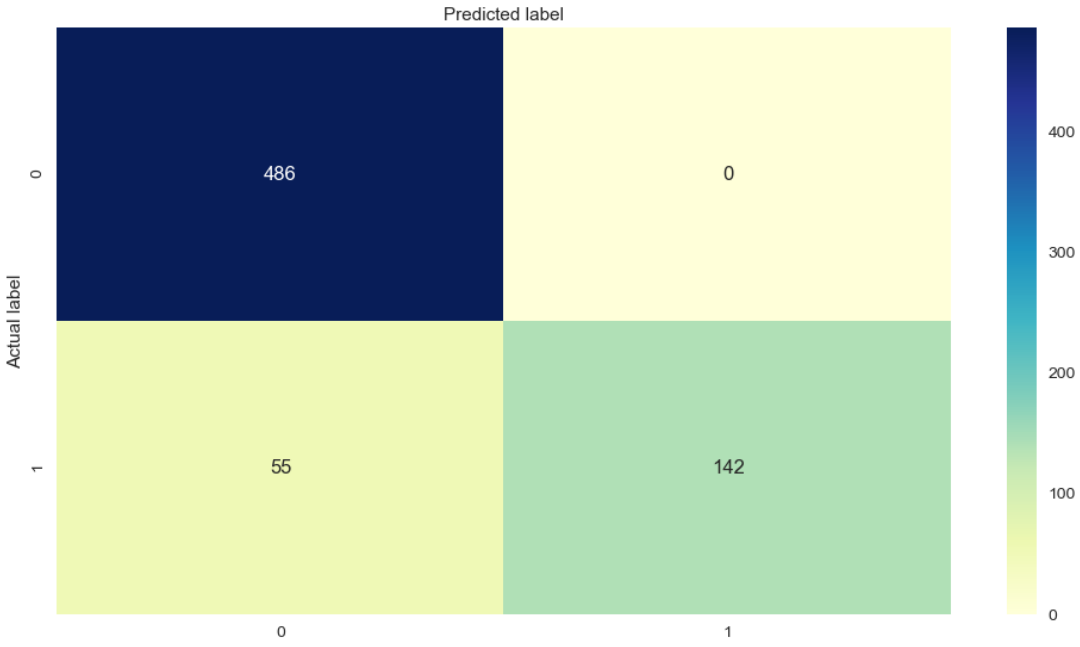
Comparative Analysis of Various Algorithms using CountVectorizer



- Experiment 1 – Multinomial Naïve Bayes on TF-IDF

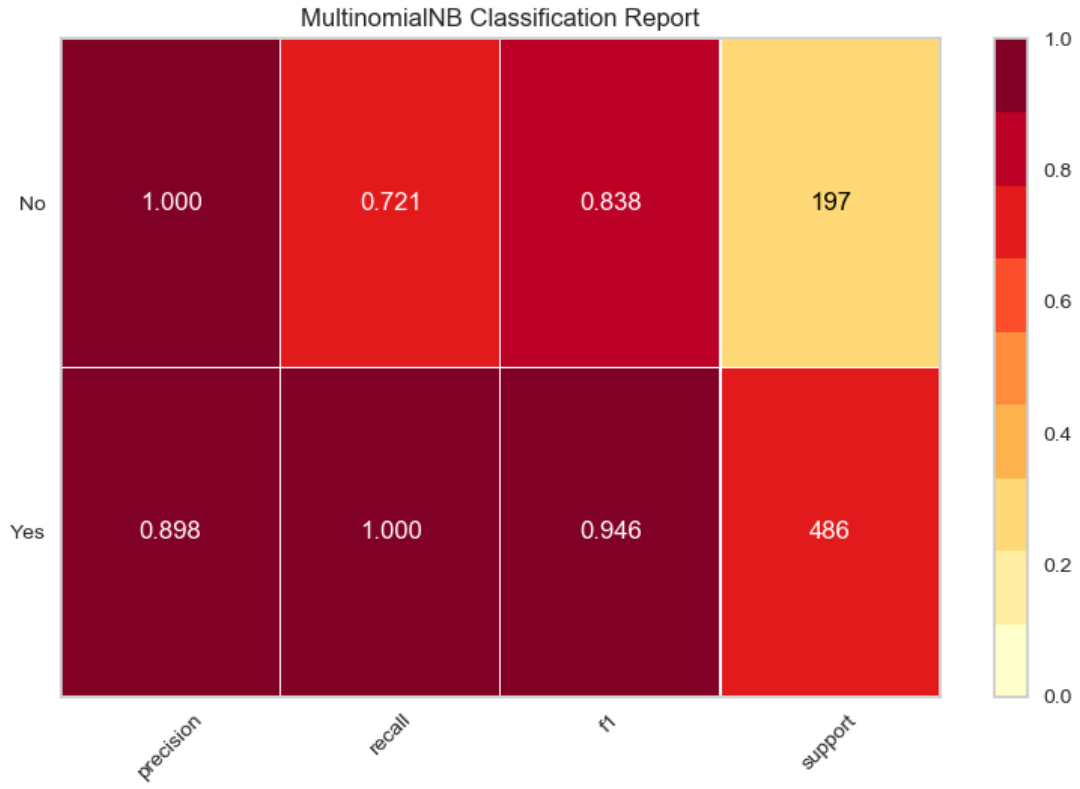
```
#MultinomialNB
def MNaiveBayes_TFID_Vec(X_train,y_train,X_test,y_test):
  global acc1
  clf = MultinomialNB()
  clf.fit(X_train,y_train)
  print("MultinomialNB :train set")
  y_pred = clf.predict(X_train)
  pred=clf.predict_proba(X_test)
  print("MultinomialNB using TFIDF Vectorizer :Confusion Matrix: ", confusion_matrix(y_train, y_pred))
  print ("MultinomialNB using TFIDF Vectorizer :Accuracy : ", accuracy_score(y_train,y_pred)*100)
  print("MultinomialNB :Test set")
  y_pred = clf.predict(X_test)
  print("MultinomialNB using TFIDF Vectorizer :Confusion Matrix: ", confusion_matrix(y_test, y_pred))
  print ("MultinomialNB using TFIDF Vectorizer :Accuracy : ", accuracy_score(y_test,y_pred)*100)
  acc1 =accuracy_score(y_test,y_pred)*100

  #confusion Matrix
  matrix =confusion_matrix(y_test, y_pred)
  class_names=[0,1]
  fig, ax = plt.subplots()
  tick_marks = np.arange(len(class_names))
  plt.xticks(tick_marks, class_names)
  plt.yticks(tick_marks, class_names)
  sns.heatmap(pd.DataFrame(matrix), annot=True, cmap="YlGnBu" ,fmt='g')
  ax.xaxis.set_label_position("top")
  plt.tight_layout()
  plt.title('Confusion matrix', y=1.1)
```

MultinomialNB using TFIDF Vectorizer :Accuracy : 91.94729136163983



Classification Report:

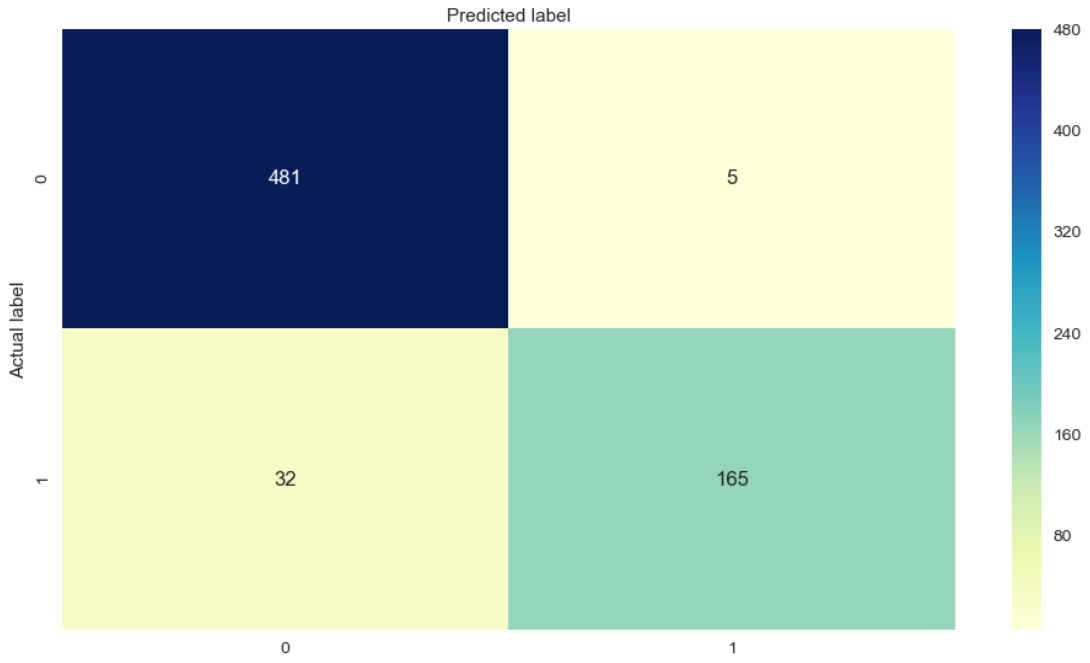
	precision	recall	f1-score	support
Yes	0.90	1.00	0.95	486
No	1.00	0.72	0.84	197
accuracy			0.92	683
macro avg	0.95	0.86	0.89	683
weighted avg	0.93	0.92	0.92	683



Experiment 2 – LogisticRegression on TF-IDF Vectorizer

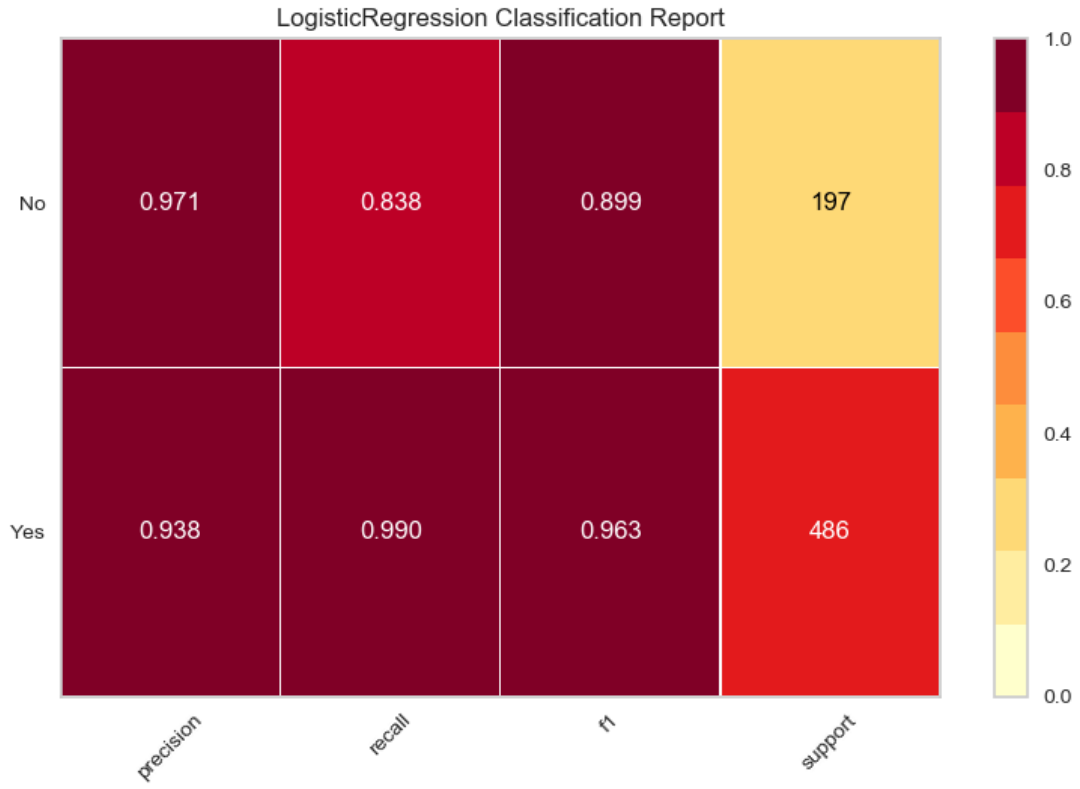
```python
#Logistic Regression
def LR_TFID_Vec(X_train,y_train,X_test,y_test):
  global acc2
  clf = LogisticRegression()
  clf.fit(X_train,y_train)
  print("LogisticRegression :train set")
  y_pred = clf.predict(X_train)
  pred=clf.predict_proba(X_test)
  print("LogisticRegression using TFIDF Vectorizer :Confusion Matrix: ", confusion_matrix(y_train, y_pred))
  print ("LogisticRegression using TFIDF Vectorizer :Accuracy : ", accuracy_score(y_train,y_pred)*100)
  print("LogisticRegression :Test set")
  y_pred = clf.predict(X_test)
  print("LogisticRegression using TFIDF Vectorizer :Confusion Matrix: ", confusion_matrix(y_test, y_pred))
  print ("LogisticRegression using TFIDF Vectorizer :Accuracy : ", accuracy_score(y_test,y_pred)*100)
  acc2=accuracy_score(y_test,y_pred)*100

  #confusion Matrix
  matrix =confusion_matrix(y_test, y_pred)
  class_names=[0,1]
  fig, ax = plt.subplots()
  tick_marks = np.arange(len(class_names))
  plt.xticks(tick_marks, class_names)
  plt.yticks(tick_marks, class_names)
  sns.heatmap(pd.DataFrame(matrix), annot=True, cmap="YlGnBu" ,fmt='g')
  ax.xaxis.set_label_position("top")
  plt.tight_layout()
```

LogisticRegression using TFIDF Vectorizer :Accuracy : 94.5827232796486



Classification Report:

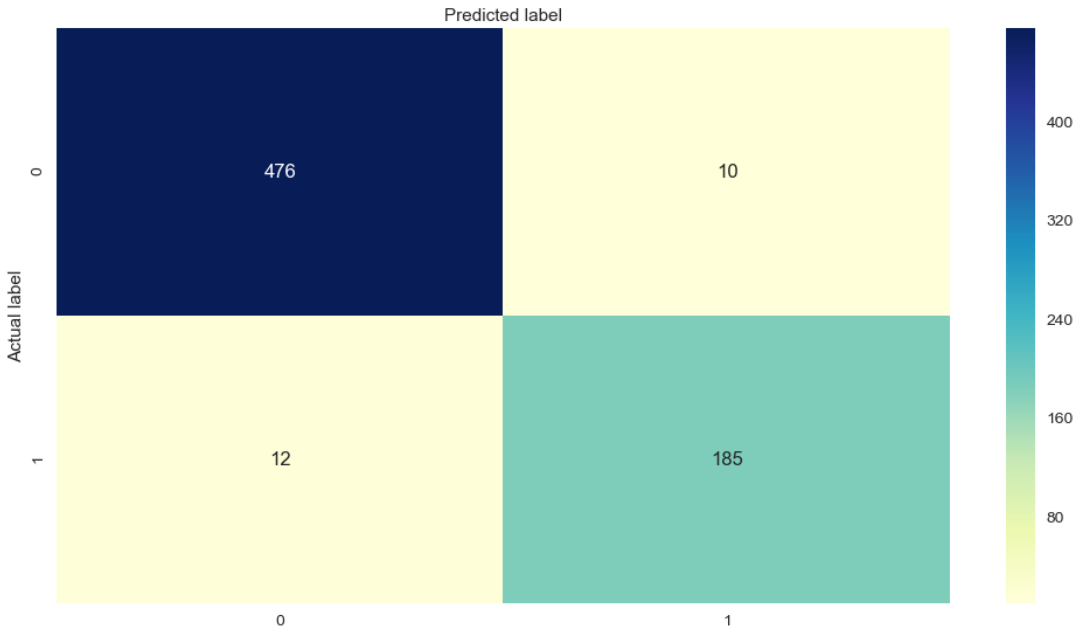
	precision	recall	f1-score	support
Yes	0.94	0.99	0.96	486
No	0.97	0.84	0.90	197
accuracy			0.95	683
macro avg	0.95	0.91	0.93	683
weighted avg	0.95	0.95	0.94	683



Experiment 3 – Adaboost on CountVectorizer

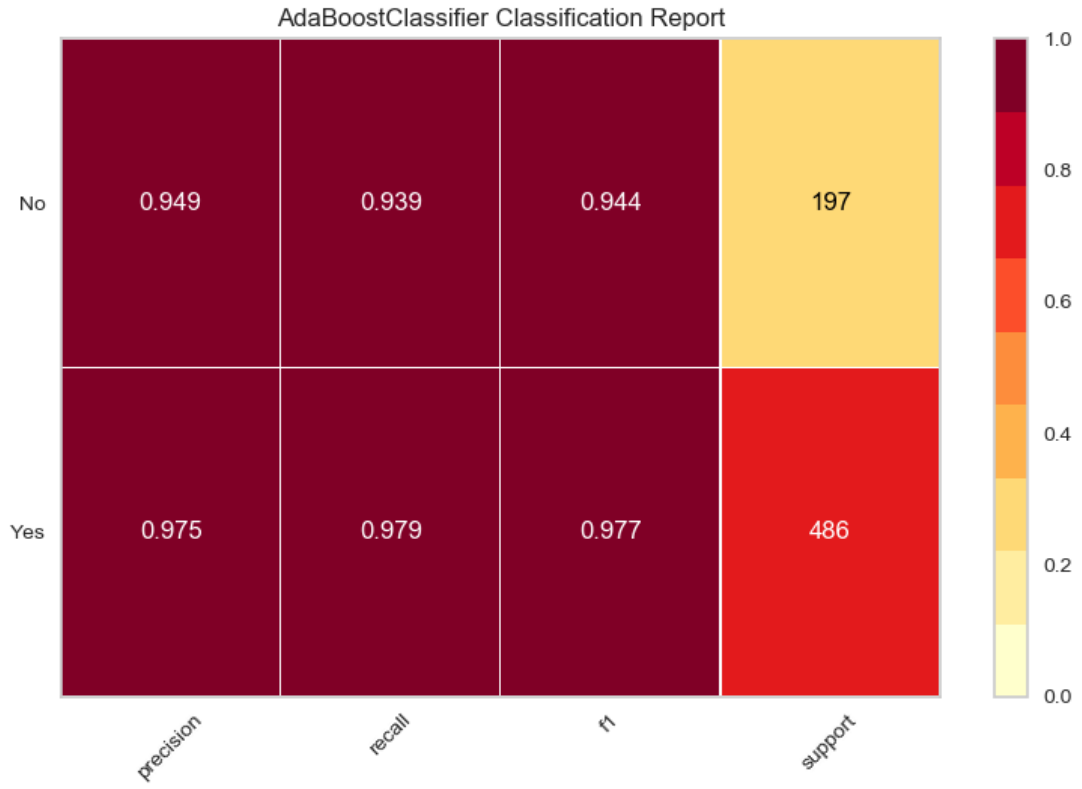
```
#AdaBoost Classifier
def Ada_TFID_Vec(X_train,y_train,X_test,y_test):
  global acc3
  clf = AdaBoostClassifier()
  clf.fit(X_train,y_train)
  print("AdaBoost :train set")
  y_pred = clf.predict(X_train)
  pred=clf.predict_proba(X_test)
  print("AdaBoost using TFIDF Vectorizer :Confusion Matrix: ", confusion_matrix(y_train, y_pred))
  print ("AdaBoost using TFIDF Vectorizer :Accuracy : ", accuracy_score(y_train,y_pred)*100)
  print("AdaBoost :Test set")
  y_pred = clf.predict(X_test)
  print("AdaBoost using TFIDF Vectorizer :Confusion Matrix: ", confusion_matrix(y_test, y_pred))
  print ("AdaBoost using TFIDF Vectorizer :Accuracy : ", accuracy_score(y_test,y_pred)*100)
  acc3=accuracy_score(y_test,y_pred)*100

  #confusion Matrix
  matrix =confusion_matrix(y_test, y_pred)
  class_names=[0,1]
  fig, ax = plt.subplots()
  tick_marks = np.arange(len(class_names))
  plt.xticks(tick_marks, class_names)
  plt.yticks(tick_marks, class_names)
  sns.heatmap(pd.DataFrame(matrix), annot=True, cmap="YlGnBu" ,fmt='g')
  ax.xaxis.set_label_position("top")
  plt.tight_layout()
  plt.title('Confusion matrix', y=1.1)
```

AdaBoost using TFIDF Vectorizer :Accuracy : 96.77891654465594



Classification Report

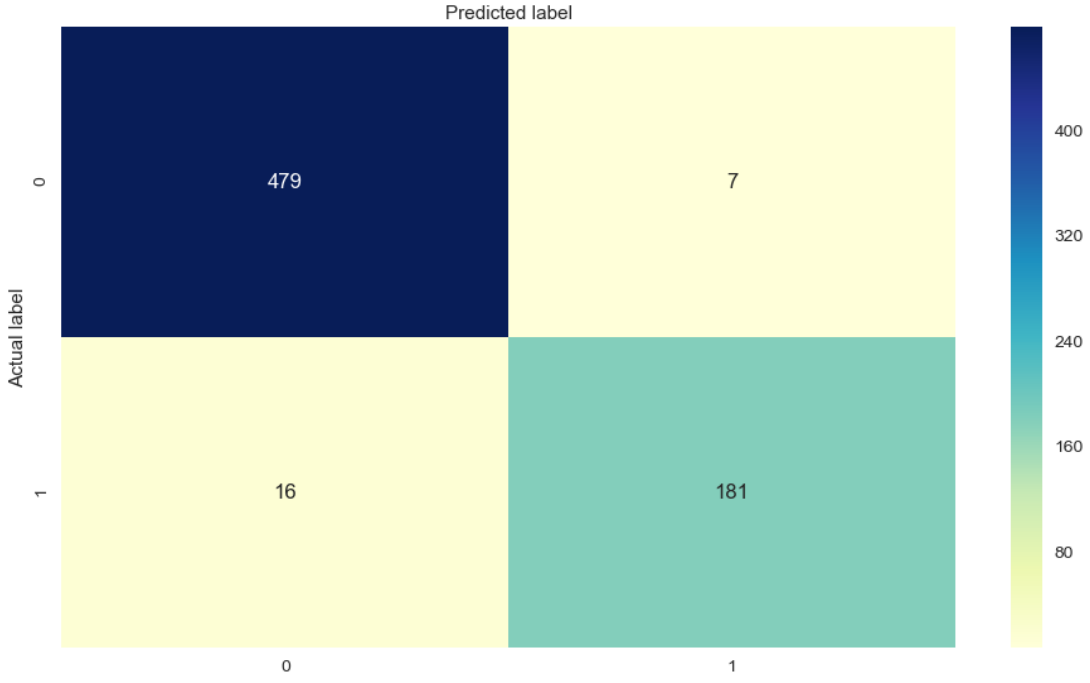
	precision	recall	f1-score	support
Yes	0.98	0.98	0.98	486
No	0.95	0.94	0.94	197
accuracy			0.97	683
macro avg	0.96	0.96	0.96	683
weighted avg	0.97	0.97	0.97	683



Experiment 4 – RandomForest on TF-IDF Vectorizer

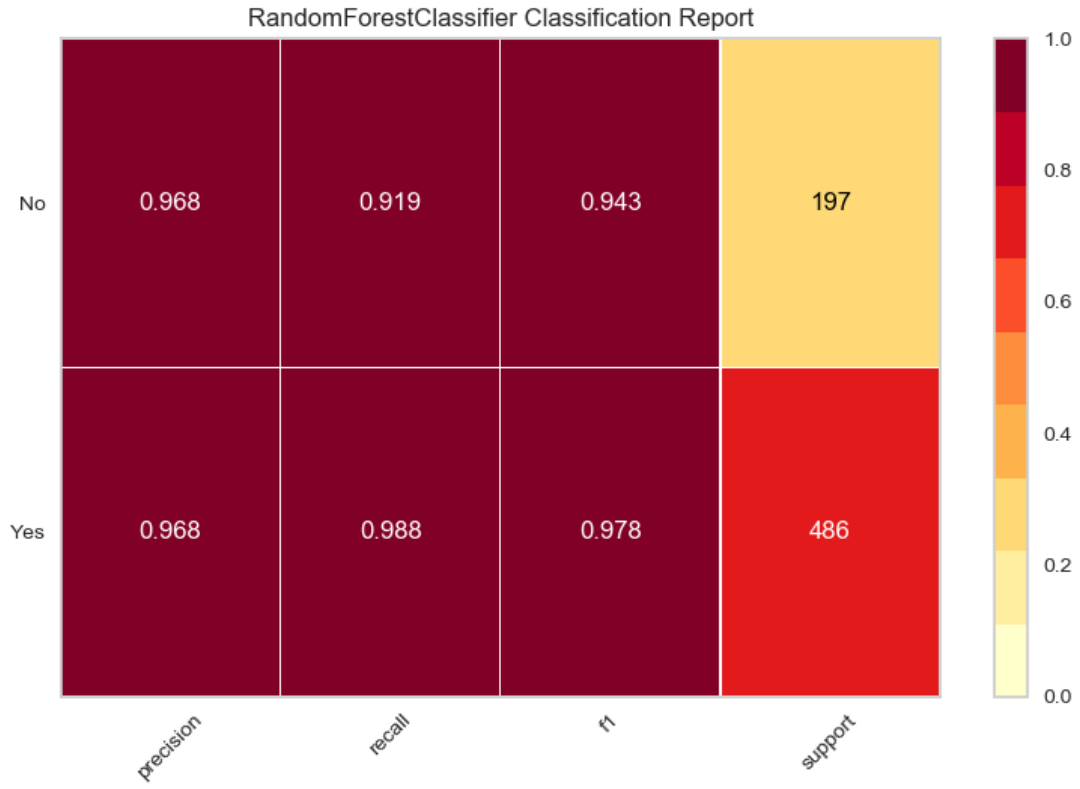
```
#RandomForestClassifier
def RF_TFID_Vec(X_train,y_train,X_test,y_test):
  global acc4
  clf = RandomForestClassifier()
  clf.fit(X_train,y_train)
  print("Random Forest :train set")
  y_pred = clf.predict(X_train)
  pred=clf.predict_proba(X_test)
  print("Random Forest using TFIDF Vectorizer :Confusion Matrix: ", confusion_matrix(y_train, y_pred))
  print ("Random Forest using TFIDF Vectorizer :Accuracy : ", accuracy_score(y_train,y_pred)*100)
  print("Random Forest :Test set")
  y_pred = clf.predict(X_test)
  print("Random Forest using TFIDF Vectorizer :Confusion Matrix: ", confusion_matrix(y_test, y_pred))
  print ("Random Forest using TFIDF Vectorizer :Accuracy : ", accuracy_score(y_test,y_pred)*100)
  acc4=accuracy_score(y_test,y_pred)*100

  #confusion Matrix
  matrix =confusion_matrix(y_test, y_pred)
  class_names=[0,1]
  fig, ax = plt.subplots()
  tick_marks = np.arange(len(class_names))
  plt.xticks(tick_marks, class_names)
  plt.yticks(tick_marks, class_names)
  sns.heatmap(pd.DataFrame(matrix), annot=True, cmap="YlGnBu" ,fmt='g')
  ax.xaxis.set_label_position("top")
  plt.tight_layout()
  plt.title('Confusion matrix', y=1.1)
```

Random Forest using TFIDF Vectorizer :Accuracy : 96.63250366032212

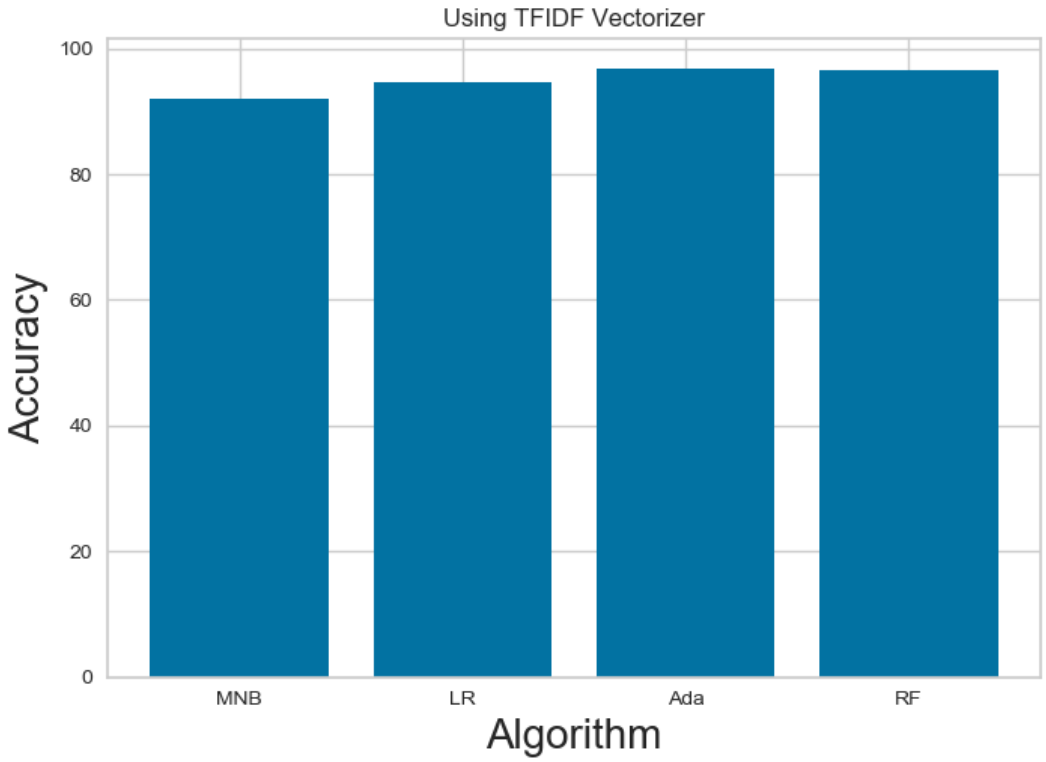


Classification Report:

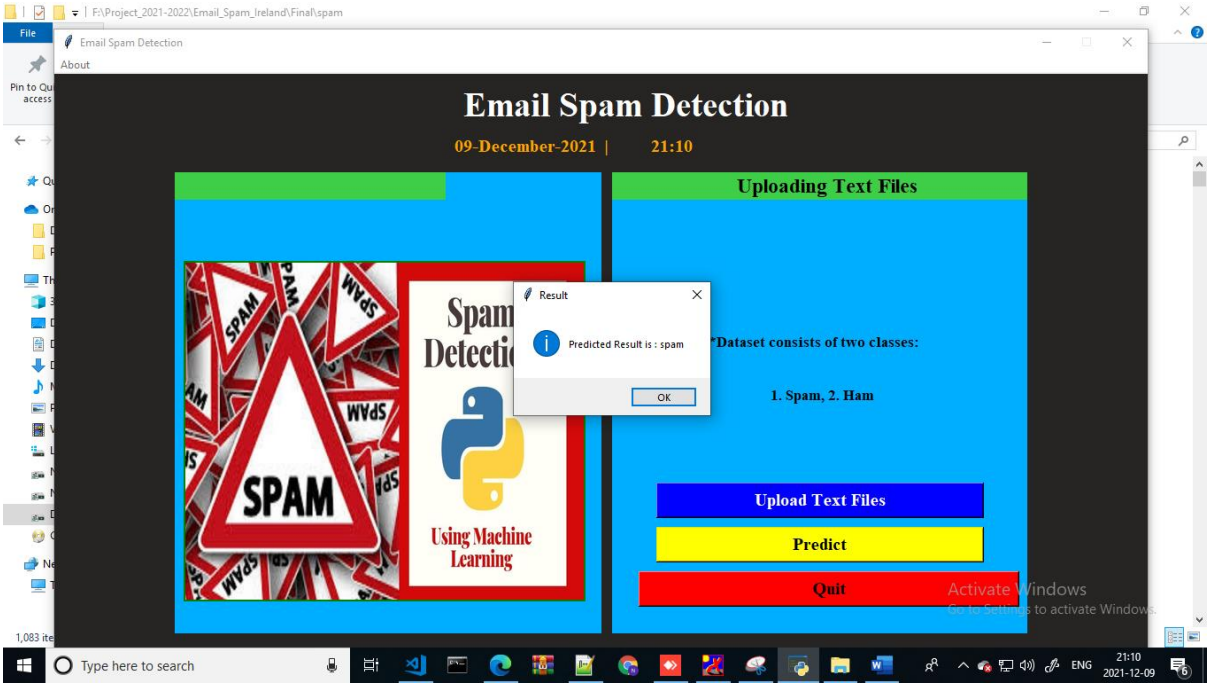
	precision	recall	f1-score	support
Yes	0.97	0.99	0.98	486
No	0.96	0.92	0.94	197
accuracy			0.97	683
macro avg	0.97	0.95	0.96	683
weighted avg	0.97	0.97	0.97	683



Comparative Analysis of Various Algorithms using TF-IDF Vectorizer



GUI OUTPUT:


Email Spam Detection
About
Email Spam Detection
09-December-2021 | 21:10
Uploading Text Files
Result
Predicted Result is : spam
OK
Dataset consists of two classes:
1. Spam, 2. Ham
Upload Text Files
Predict
Quit
Using Machine Learning
Activate Windows
Type here to search