

```
In [1]: import warnings
warnings.simplefilter('ignore')

import numpy as np
import pandas as pd
import matplotlib.pyplot as plt

from sklearn.model_selection import train_test_split
from sklearn.feature_extraction.text import TfidfVectorizer

from sklearn.metrics import accuracy_score
from sklearn.metrics import precision_score
from sklearn.metrics import recall_score
from sklearn.metrics import f1_score

from sklearn.linear_model import LogisticRegression
from sklearn.tree import DecisionTreeClassifier
from sklearn.svm import SVC
from sklearn.neighbors import KNeighborsClassifier
from sklearn.ensemble import RandomForestClassifier

from sklearn.ensemble import StackingClassifier
```

```
In [2]: df = pd.read_csv('C:\\Users\\saswa\\OneDrive\\Desktop\\Pinaki_Spam_Email_Detection\\Spam Email Detection-spam.csv')
df
```

Out[2]:

	v1	v2	Unnamed: 2	Unnamed: 3	Unnamed: 4
0	ham	Go until jurong point, crazy.. Available only ...	NaN	NaN	NaN
1	ham	Ok lar... Joking wif u oni...	NaN	NaN	NaN
2	spam	Free entry in 2 a wkly comp to win FA Cup fina...	NaN	NaN	NaN
3	ham	U dun say so early hor... U c already then say...	NaN	NaN	NaN
4	ham	Nah I don't think he goes to usf, he lives aro...	NaN	NaN	NaN
...	...	...	...	...	...
5567	spam	This is the 2nd time we have tried 2 contact u...	NaN	NaN	NaN
5568	ham	Will  b going to esplanade fr home?	NaN	NaN	NaN
5569	ham	Pity, * was in mood for that. So...any other s...	NaN	NaN	NaN
5570	ham	The guy did some bitching but I acted like i'd...	NaN	NaN	NaN
5571	ham	Rofl. Its true to its name	NaN	NaN	NaN

5572 rows × 5 columns

```
In [3]: data = df.drop(labels=['Unnamed: 2', 'Unnamed: 3', 'Unnamed: 4'], axis=1)
data.isnull().sum()
```

```
Out[3]: v1    0
v2    0
dtype: int64
```

```
In [4]: data.dtypes
```

```
Out[4]: v1    object
v2    object
dtype: object
```

```
In [5]: data.loc[data['v1'] == 'spam', 'v1',] = 0
data.loc[data['v1'] == 'ham', 'v1',] = 1
data['v1'].value_counts()
```

```
Out[5]: 1    4825
        0     747
        Name: v1, dtype: int64
```

```
In [6]: X = data['v2']

        Y = data['v1']
```

```
In [7]: X
```

```
Out[7]: 0      Go until jurong point, crazy.. Available only ...
        1              Ok lar... Joking wif u oni...
        2      Free entry in 2 a wkly comp to win FA Cup fina...
        3      U dun say so early hor... U c already then say...
        4      Nah I don't think he goes to usf, he lives aro...

        ...
        5567     This is the 2nd time we have tried 2 contact u...
        5568           Will _ b going to esplanade fr home?
        5569     Pity, * was in mood for that. So...any other s...
        5570     The guy did some bitching but I acted like i'd...
        5571           Rofl. Its true to its name
        Name: v2, Length: 5572, dtype: object
```

```
In [8]: Y
```

```
Out[8]: 0      1
        1      1
        2      0
        3      1
        4      1
        ..
        5567    0
        5568    1
        5569    1
        5570    1
        5571    1
        Name: v1, Length: 5572, dtype: object
```

```
In [9]: X_train, X_test, Y_train, Y_test = train_test_split(X, Y, test_size=0.2, random_state=3)
```

```
In [10]: print(X.shape)
         print(X_train.shape)
         print(X_test.shape)
```

```
(5572,)
(4457,)
(1115,)
```

```
In [11]: feature_extraction = TfidfVectorizer(min_df = 1, stop_words='english')
```

```
In [12]: X_train_features = feature_extraction.fit_transform(X_train)
         X_test_features = feature_extraction.transform(X_test)
```

```
In [13]: Y_train = Y_train.astype('int')
         Y_test = Y_test.astype('int')
```

```
In [14]: lr = LogisticRegression()
         lr.fit(X_train_features, Y_train)

         lr_train = lr.predict(X_train_features)
         lr_test = lr.predict(X_test_features)

         lr_train_acc = accuracy_score(Y_train, lr_train)
```

```
lr_test_acc = accuracy_score(Y_test, lr_test)

lr_precision = precision_score(Y_test, lr_test)
lr_recall = recall_score(Y_test, lr_test)
lr_f1 = f1_score(Y_test, lr_test)

print("Logistic Regression:\n")
print("Training Data Accuracy:", lr_train_acc)
print("Testing Data Accuracy :", lr_test_acc)

print("Precision          :", lr_precision)
print("Recall              :", lr_recall)
print("F1 Score            :", lr_f1)
```

Logistic Regression:

```
Training Data Accuracy: 0.9661207089970832
Testing Data Accuracy : 0.9623318385650225
Precision              : 0.959
Recall                 : 0.9989583333333333
F1 Score               : 0.9785714285714285
```

```
In [15]: dtrees = DecisionTreeClassifier()
dtrees.fit(X_train_features, Y_train)

dt_train = dtrees.predict(X_train_features)
dt_test = dtrees.predict(X_test_features)

dt_train_acc = accuracy_score(Y_train, dt_train)
dt_test_acc = accuracy_score(Y_test, dt_test)

dt_precision = precision_score(Y_test, dt_test)
dt_recall = recall_score(Y_test, dt_test)
dt_f1 = f1_score(Y_test, dt_test)
```

```
print("Decision Tress:\n")
print("Training Data Accuracy:", dt_train_acc)
print("Testing Data Accuracy :", dt_test_acc)

print("Precision          :", dt_precision)
print("Recall              :", dt_recall)
print("F1 Score            :", dt_f1)
```

Decision Tress:

```
Training Data Accuracy: 1.0
Testing Data Accuracy : 0.9704035874439462
Precision              : 0.9753846153846154
Recall                 : 0.990625
F1 Score               : 0.9829457364341085
```

```
In [16]: knn = KNeighborsClassifier()
knn.fit(X_train_features, Y_train)

knn_train = knn.predict(X_train_features)
knn_test  = knn.predict(X_test_features)

knn_train_acc = accuracy_score(Y_train, knn_train)
knn_test_acc  = accuracy_score(Y_test, knn_test)

knn_precision = precision_score(Y_test, knn_test)
knn_recall    = recall_score(Y_test, knn_test)
knn_f1        = f1_score(Y_test, knn_test)

print("K Nearest Neighbors:\n")
print("Training Data Accuracy:", knn_train_acc)
print("Testing Data Accuracy :", knn_test_acc)

print("Precision          :", knn_precision)
print("Recall              :", knn_recall)
print("F1 Score            :", knn_f1)
```

K Nearest Neighbors:

Training Data Accuracy: 0.9201256450527261  
Testing Data Accuracy : 0.905829596412556  
Precision : 0.9014084507042254  
Recall : 1.0  
F1 Score : 0.9481481481481481

```
In [17]: rf = RandomForestClassifier()
rf.fit(X_train_features, Y_train)

rf_train = rf.predict(X_train_features)
rf_test = rf.predict(X_test_features)

rf_train_acc = accuracy_score(Y_train, rf_train)
rf_test_acc = accuracy_score(Y_test, rf_test)

rf_precision = precision_score(Y_test, rf_test)
rf_recall = recall_score(Y_test, rf_test)
rf_f1 = f1_score(Y_test, rf_test)

print("Random Forest:\n")
print("Training Data Accuracy:", rf_train_acc)
print("Testing Data Accuracy :", rf_test_acc)

print("Precision      :", rf_precision)
print("Recall          :", rf_recall)
print("F1 Score        :", rf_f1)
```

Random Forest:

Training Data Accuracy: 1.0  
Testing Data Accuracy : 0.9829596412556054  
Precision : 0.9815762538382804  
Recall : 0.9989583333333333  
F1 Score : 0.9901910170366547

```
In [18]: estimators = [ ('lr', lr), ('dtree', dtrees), ('knn', knn), ('rf', rf) ]
stack = StackingClassifier(estimators, final_estimator = SVC(kernel='linear'))
stack.fit(X_train_features, Y_train)

stack_train = stack.predict(X_train_features)
stack_test = stack.predict(X_test_features)

stack_train_acc = accuracy_score(Y_train, stack_train)
stack_test_acc = accuracy_score(Y_test, stack_test)

stack_precision = precision_score(Y_test, stack_test)
stack_recall = recall_score(Y_test, stack_test)
stack_f1 = f1_score(Y_test, stack_test)

print("Stacking Classifier:\n")
print("Training Data Accuracy:", stack_train_acc)
print("Testing Data Accuracy :", stack_test_acc)

print("Precision          :", stack_precision)
print("Recall              :", stack_recall)
print("F1 Score            :", stack_f1)
```

Stacking Classifier:

```
Training Data Accuracy: 0.9995512676688355
Testing Data Accuracy : 0.9856502242152466
Precision              : 0.98659793814433
Recall                 : 0.996875
F1 Score               : 0.9917098445595856
```

```
In [19]: train_acc_list = {"LR":lr_train_acc,
                           "DT":dt_train_acc,
                           "KNN":knn_train_acc,
                           "RF":rf_train_acc,
                           "STACK":stack_train_acc}

test_acc_list = {"LR":lr_test_acc,
```



```
        "DT":dt_test_acc,
        "KNN":knn_test_acc,
        "RF":rf_test_acc,
        "STACK":stack_test_acc}

precision_list = {"LR":lr_precision,
                  "DT":dt_precision,
                  "KNN":knn_precision,
                  "RF":rf_precision,
                  "STACK":stack_precision}

recall_list = {"LR":lr_recall,
                "DT":dt_recall,
                "KNN":knn_recall,
                "RF":rf_recall,
                "STACK":stack_recall}

f1_list = {"LR":lr_f1,
            "DT":dt_f1,
            "KNN":knn_f1,
            "RF":rf_f1,
            "STACK":stack_f1}
```

```
In [20]: a1 = pd.DataFrame.from_dict(train_acc_list, orient = 'index', columns = ["Traning Accuracy"])
a2 = pd.DataFrame.from_dict(test_acc_list, orient = 'index', columns = ["Testing Accuracy"])
a3 = pd.DataFrame.from_dict(precision_list, orient = 'index', columns = ["Precision Score"])
a4 = pd.DataFrame.from_dict(recall_list, orient = 'index', columns = ["Recall Score"])
a5 = pd.DataFrame.from_dict(f1_list, orient = 'index', columns = ["F1 Score"])

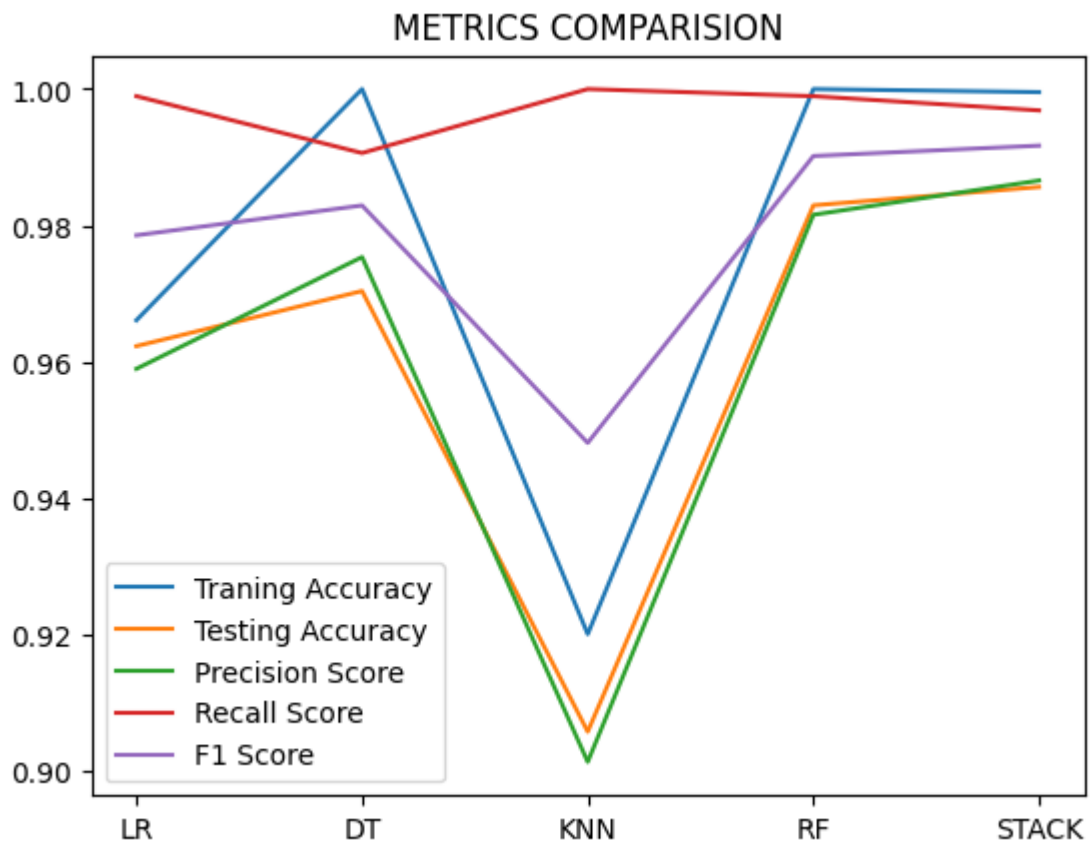
org = pd.concat([a1, a2, a3, a4, a5], axis = 1)
org
```

Out[20]:

	Traning Accuracy	Testing Accuracy	Precision Score	Recall Score	F1 Score
<b>LR</b>	0.966121	0.962332	0.959000	0.998958	0.978571
<b>DT</b>	1.000000	0.970404	0.975385	0.990625	0.982946
<b>KNN</b>	0.920126	0.905830	0.901408	1.000000	0.948148
<b>RF</b>	1.000000	0.982960	0.981576	0.998958	0.990191
<b>STACK</b>	0.999551	0.985650	0.986598	0.996875	0.991710

In [21]:

```
alg = ['LR', 'DT', 'KNN', 'RF', 'STACK']
plt.plot(alg, a1)
plt.plot(alg, a2)
plt.plot(alg, a3)
plt.plot(alg, a4)
plt.plot(alg, a5)
legend = ['Traning Accuracy', 'Testing Accuracy', 'Precision Score', 'Recall Score', 'F1 Score']
plt.title("METRICS COMPARISION")
plt.legend(legend)
plt.show()
```



In [ ]: