In [6]:

```
import pandas as pd
import numpy as np
```

In [11]:

```
data = pd.read_csv('Documents/Mini Project/Existing System/spam_ham_dataset.csv')
data.head()
```

Out[11]:

| | text | label |
|---|---|---|
| 0 | Subject: enron methanol ; meter # : 988291\r\n... | ham |
| 1 | Subject: hpl nom for january 9 , 2001\r\n( see... | ham |
| 2 | Subject: neon retreat\r\nho ho ho , we ' re ar... | ham |
| 3 | Subject: photoshop , windows , office . cheap ... | spam |
| 4 | Subject: re : indian springs\r\nthis deal is t... | ham |

In [12]:

```
len(data)
```

Out[12]:

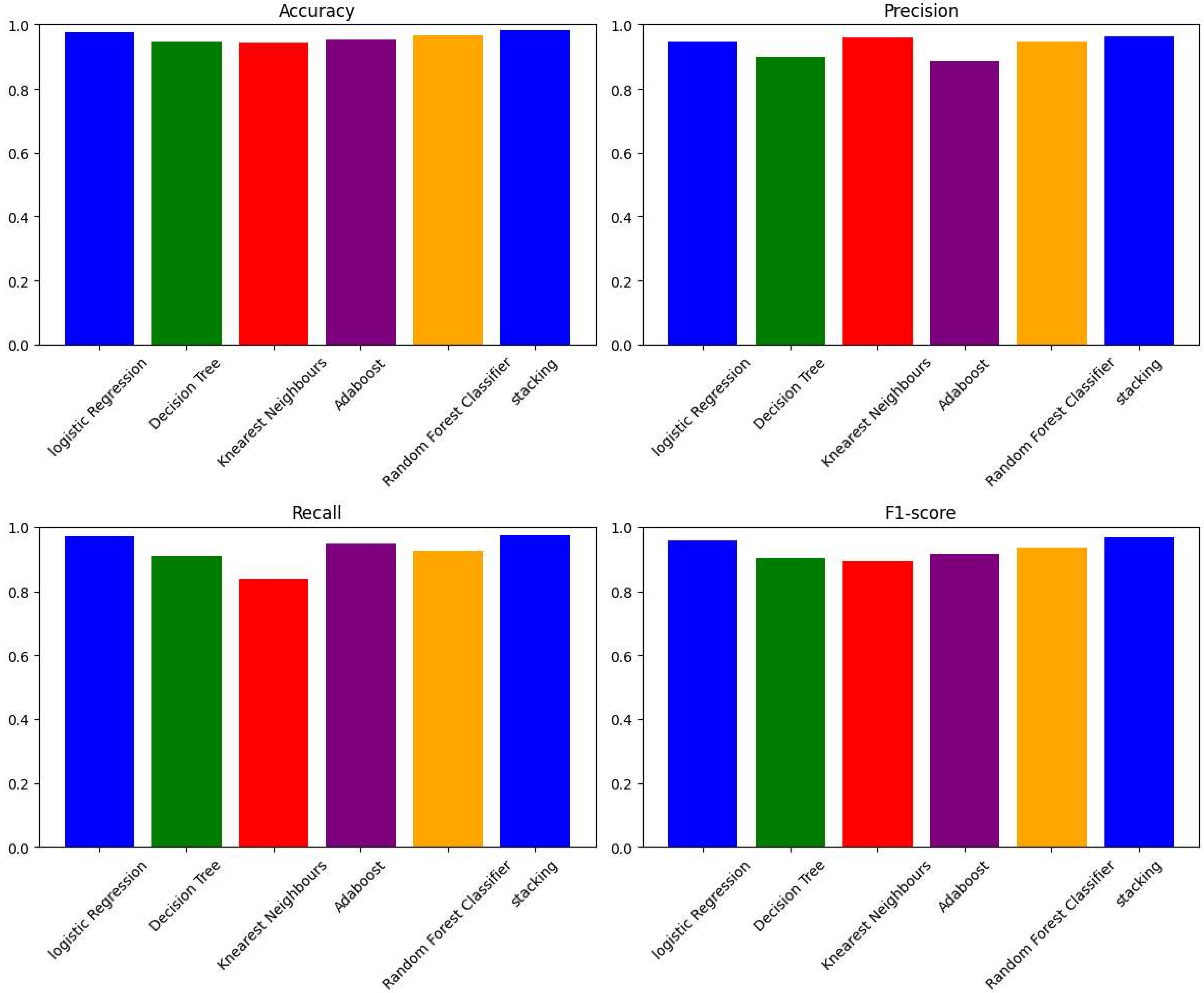
```
5171
```

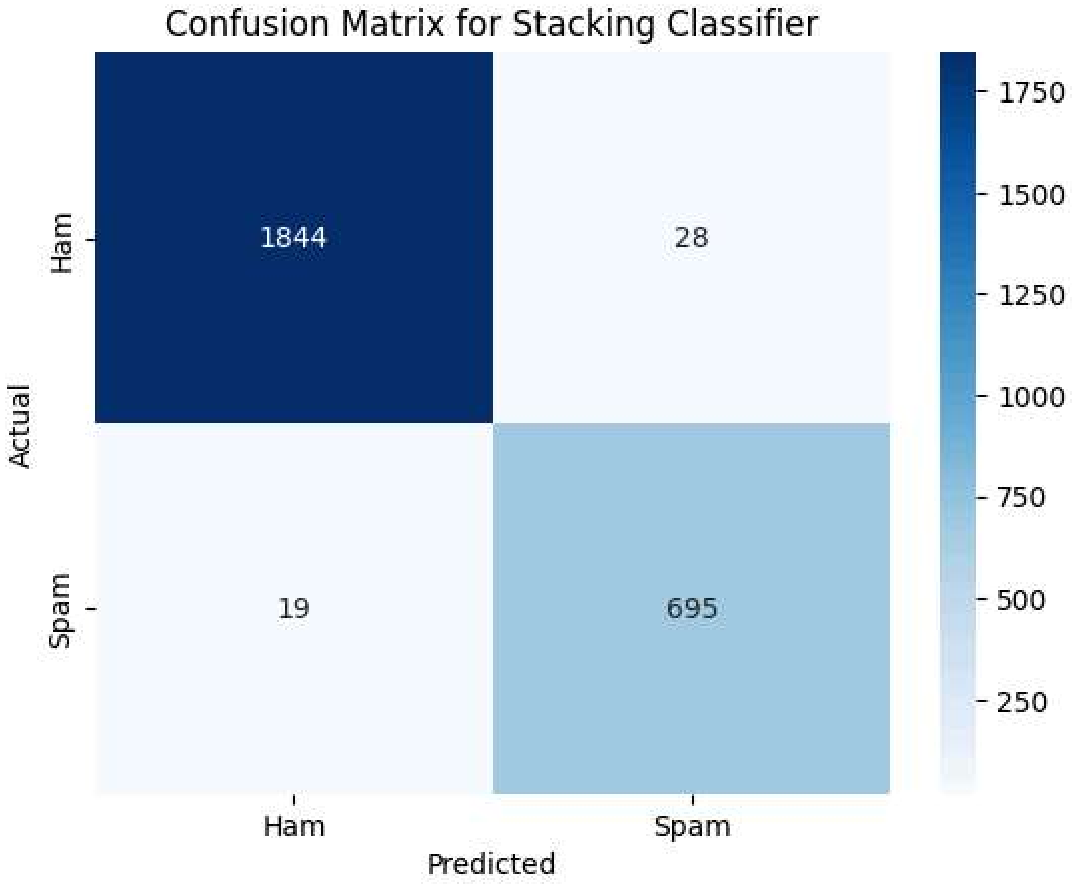
In [13]:

```
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.model_selection import train_test_split
from sklearn.feature_extraction.text import TfidfVectorizer
from sklearn.linear_model import LogisticRegression
from sklearn.tree import DecisionTreeClassifier
from sklearn.neighbors import KNeighborsClassifier
from sklearn.ensemble import AdaBoostClassifier, RandomForestClassifier
from sklearn.ensemble import StackingClassifier
from sklearn.metrics import accuracy_score, precision_score, recall_score, f1_score, confusion_matrix
import warnings
warnings.filterwarnings("ignore")
data['label'] = data['label'].map({'spam': 1, 'ham': 0})
vectorizer = TfidfVectorizer()
X = vectorizer.fit_transform(data['text'])
y = data['label']
print(y)
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.5, random_state=42)
base_models = [('logistic Regression', LogisticRegression()), ('Decision Tree', DecisionTreeClassifier()), ('Knearest Neighbours', KNeighborsClassifier()), ('Adaboost', AdaBoostClassifier()),
stacking_model = StackingClassifier(estimators=base_models, final_estimator=LogisticRegression())
results = {}
for name, model in base_models:
    model.fit(X_train, y_train)
    y_pred_base = model.predict(X_test)
    acc = accuracy_score(y_test, y_pred_base)
    prec = precision_score(y_test, y_pred_base)
    rec = recall_score(y_test, y_pred_base)
    f1 = f1_score(y_test, y_pred_base)
    results[name] = [acc, prec, rec, f1]
    print(f"{name}\nAccuracy: {acc}\nPrecision: {prec}\nRecall: {rec}\nF1-score: {f1}\n\n")
stacking_model.fit(X_train, y_train)
y_pred = stacking_model.predict(X_test)
accuracy = accuracy_score(y_test, y_pred)
precision = precision_score(y_test, y_pred)
recall = recall_score(y_test, y_pred)
f1 = f1_score(y_test, y_pred)
```

```
results['stacking'] = [accuracy, precision, recall, f1]
print(f"Stacking Classifier: \nAccuracy: {accuracy}\nPrecision: {precision}\nRecall: {recall}\nF1-score: {f1}\n\n")
metrics = ['Accuracy', 'Precision', 'Recall', 'F1-score']
fig, axes = plt.subplots(2, 2, figsize=(12, 10))
axes = axes.flatten()
for i, metric in enumerate(metrics):
    values = [results[model][i] for model in results]
    models = list(results.keys())
    axes[i].bar(models, values, color=['blue', 'green', 'red', 'purple', 'orange'])
    axes[i].set_title(metric)
    axes[i].set_ylim([0, 1])
    axes[i].set_xticklabels(models, rotation=45)
plt.tight_layout()
plt.show()
cm = confusion_matrix(y_test, y_pred)
sns.heatmap(cm, annot=True, fmt='d', cmap='Blues', xticklabels=['Ham', 'Spam'], yticklabels=['Ham', 'Spam'])
plt.xlabel('Predicted')
plt.ylabel('Actual')
plt.title('Confusion Matrix for Stacking Classifier')
plt.show()
print("This is Existing System, used Stacking Classfier with final_estimator: LogisticRegression()\n")
```

```
0       0
1       0
2       0
3       1
4       0
       ..
5166    0
5167    0
5168    0
5169    0
5170    1
Name: label, Length: 5171, dtype: int64
logistic Regression
Accuracy: 0.9767981438515081
Precision: 0.947945205479452
Recall: 0.969187675070028
F1-score: 0.9584487534626038


Decision Tree
Accuracy: 0.9470224284609435
Precision: 0.9001386962552012
Recall: 0.9089635854341737
F1-score: 0.9045296167247386


Knearest Neighbours
Accuracy: 0.9450889404485692
Precision: 0.9583333333333334
Recall: 0.8375350140056023
F1-score: 0.8938714499252616


Adaboost
Accuracy: 0.9520494972931168
Precision: 0.8861256544502618
Recall: 0.9481792717086834
F1-score: 0.9161028416779432


Random Forest Classifier
Accuracy: 0.9651972157772621
Precision: 0.9469914040114613
Recall: 0.9257703081232493
F1-score: 0.9362606232294618


Stacking Classifier:
Accuracy: 0.9818252126836814
Precision: 0.9612724757952974
Recall: 0.9733893557422969
F1-score: 0.9672929714683368
```


Accuracy
Precision
Recall
F1-score
1.0
0.8
0.6
0.4
0.2
0.0
logistic Regression
Decision Tree
Knearest Neighbours
Adaboost
Random Forest Classifier
stacking



```
This is Existing System, used Stacking Classfier with final_estimator: LogisticRegression()
```

In [ ]: