```
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.feature_extraction.text import TfidfVectorizer
from sklearn.model_selection import train_test_split
from sklearn.naive_bayes import MultinomialNB
from sklearn.tree import DecisionTreeClassifier
from sklearn.linear_model import LogisticRegression
from sklearn.svm import SVC
from sklearn.metrics import accuracy_score, confusion_matrix, classification_report, f1_score
```

```
file_path = r'C:\Users\hp\Downloads\archive\email_classification.csv'
df=pd.read_csv(file_path)
```

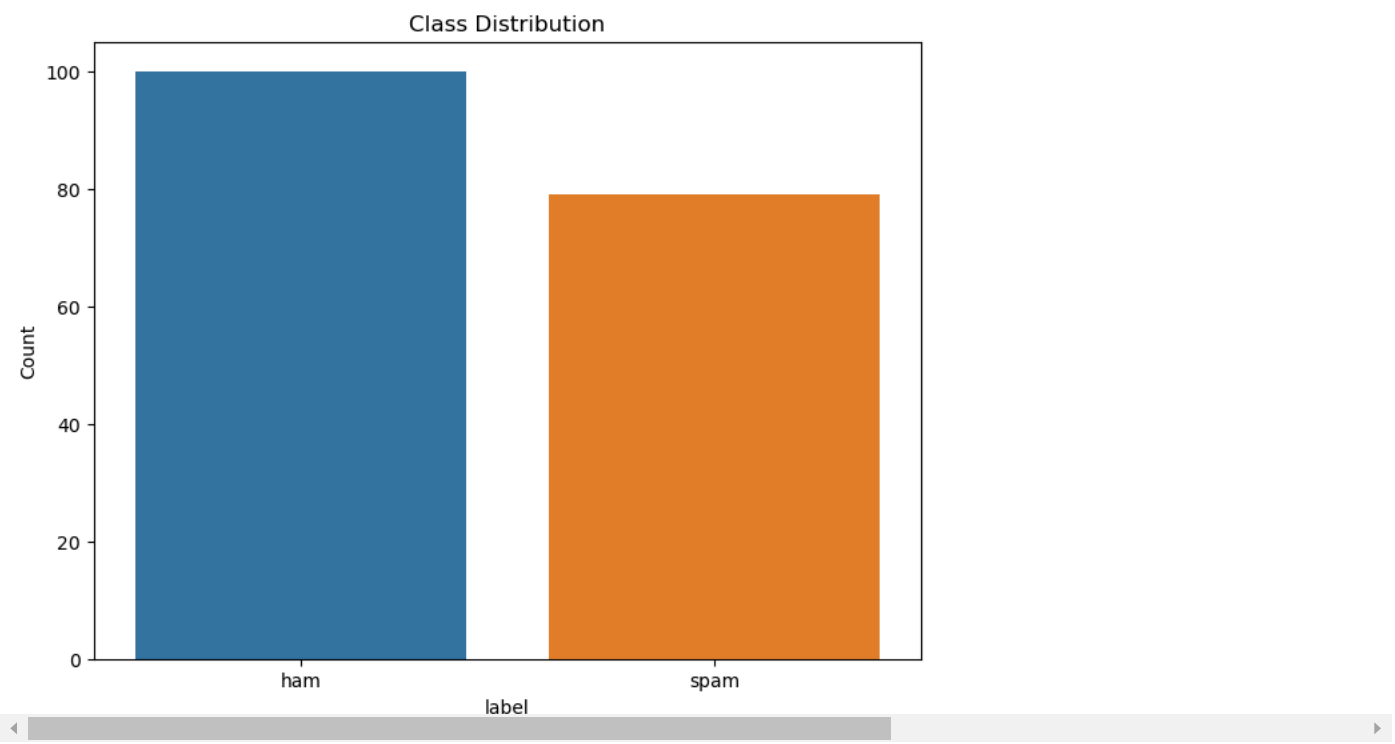
```
df.head()
```

	email	label
0	Upgrade to our premium plan for exclusive acce...	ham
1	Happy holidays from our team! Wishing you joy ...	ham
2	We're hiring! Check out our career opportuniti...	ham
3	Your Amazon account has been locked. Click her...	spam
4	Your opinion matters! Take our survey and help...	ham

```
df.shape
```

```
(179, 2)
```

```
plt.figure(figsize=(8, 6))
sns.countplot(x='label', data=df)
plt.title('Class Distribution')
plt.xlabel('label')
plt.ylabel('Count')
plt.show()
```




```
df.isna().sum()
```

```
email    0
label    0
```

```
dtype: int64
```

```
df['email'] = df['email'].str.lower().str.replace('[^\w\s]', '')
```

```
df.head()
```

	email	label
0	upgrade to our premium plan for exclusive acce...	ham
1	happy holidays from our team! wishing you joy ...	ham
2	we're hiring! check out our career opportuniti...	ham
3	your amazon account has been locked. click her...	spam
4	your opinion matters! take our survey and help...	ham

```
df["label"].replace({"ham": 0, "spam": 1}, inplace=True)
```

```
df.head()
```

	email	label
0	upgrade to our premium plan for exclusive acce...	0
1	happy holidays from our team! wishing you joy ...	0
2	we're hiring! check out our career opportuniti...	0
3	your amazon account has been locked. click her...	1
4	your opinion matters! take our survey and help...	0

```
X=df["email"]
y=df['label']
```

```
X.shape
```

```
(179,)
```

```
X.head()
```

```
0    upgrade to our premium plan for exclusive acce...
1    happy holidays from our team! wishing you joy ...
2    we're hiring! check out our career opportuniti...
3    your amazon account has been locked. click her...
4    your opinion matters! take our survey and help...
Name: email, dtype: object
```

```
X.isnull().sum()
```

```
0
```

```
y.shape
```

```
(179,)
```

```
y.head()
```

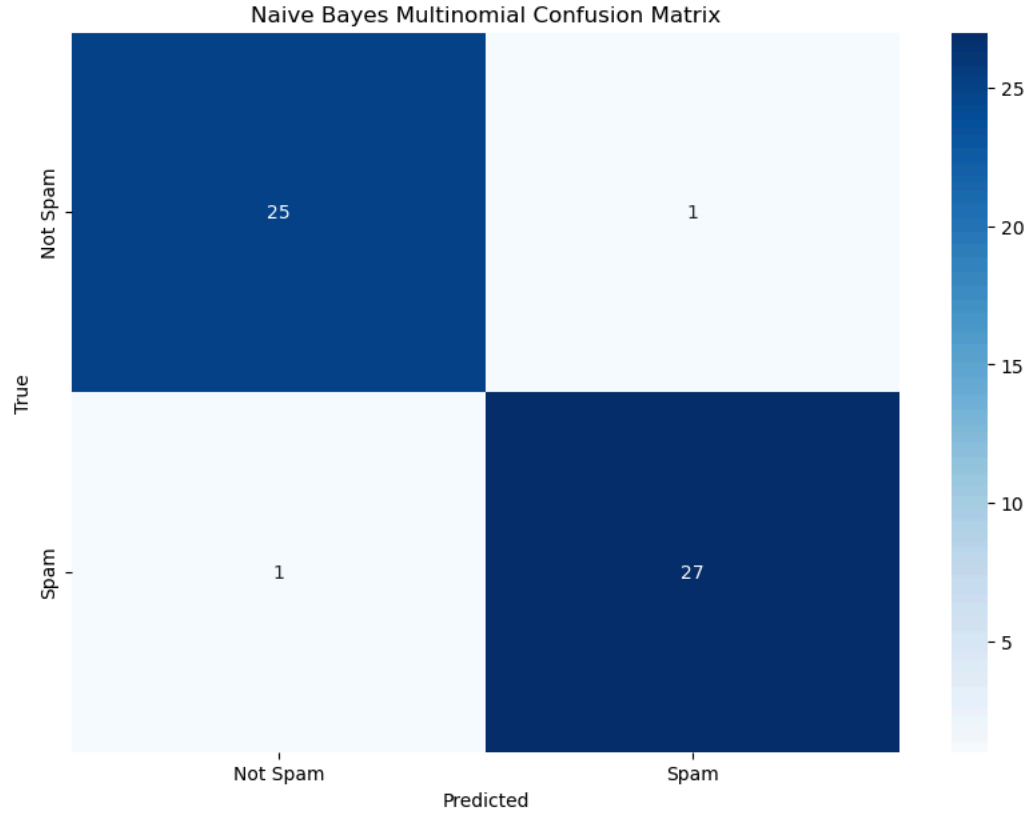
```
0    0
1    0
2    0
3    1
4    0
Name: label, dtype: int64
```

```
tf=TfidfVectorizer(stop_words="english")
X=tf.fit_transform(x)
```

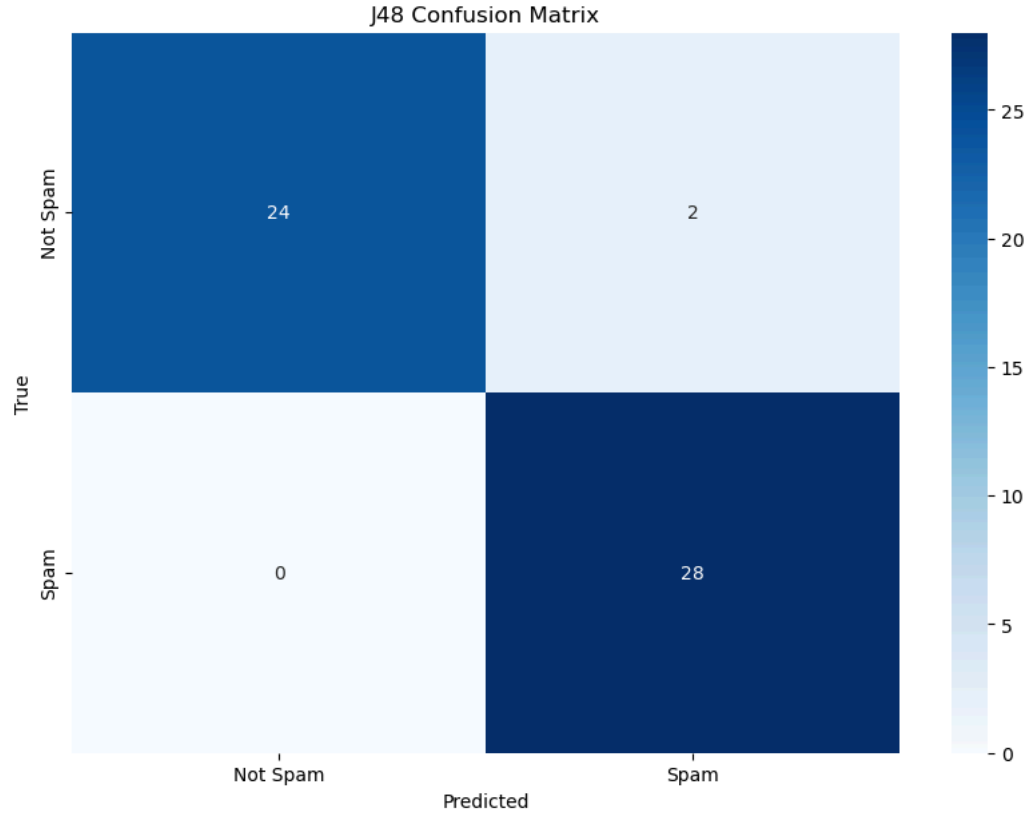
```
x_train,x_test,y_train,y_test=train_test_split(X,y,test_size=0.3,random_state=42)
```

```
nbm_model = MultinomialNB()
nbm_model.fit(x_train, y_train)
y_pred_nbm = nbm_model.predict(x_test)


j48_model = DecisionTreeClassifier()
j48_model.fit(x_train, y_train)
y_pred_j48 = j48_model.predict(x_test)


lr_model = LogisticRegression()
lr_model.fit(x_train, y_train)
y_pred_lr = lr_model.predict(x_test)


svm_model = SVC()
svm_model.fit(x_train, y_train)
y_pred_svm = svm_model.predict(x_test)


def plot_confusion_matrix(cm, title='Confusion Matrix'):
    plt.figure(figsize=(10, 7))
    sns.heatmap(cm, annot=True, fmt='d', cmap='Blues', xticklabels=['Not Spam', 'Spam'], yticklabels=['Not Spam', 'Spam'])
    plt.title(title)
    plt.xlabel('Predicted')
    plt.ylabel('True')
    plt.show()

cm_nbm = confusion_matrix(y_test, y_pred_nbm)
nbm_accuracy = accuracy_score(y_test, y_pred_nbm)
nbm_f1 = f1_score(y_test, y_pred_nbm, average='weighted')
print("Naive Bayes Multinomial Accuracy:", nbm_accuracy)
print("Naive Bayes Multinomial F1 Score:", nbm_f1)
plot_confusion_matrix(cm_nbm, title='Naive Bayes Multinomial Confusion Matrix')

cm_j48 = confusion_matrix(y_test, y_pred_j48)
j48_accuracy = accuracy_score(y_test, y_pred_j48)
j48_f1 = f1_score(y_test, y_pred_j48, average='weighted')
print("J48 Accuracy:", j48_accuracy)
print("J48 F1 Score:", j48_f1)
plot_confusion_matrix(cm_j48, title='J48 Confusion Matrix')

cm_lr = confusion_matrix(y_test, y_pred_lr)
lr_accuracy = accuracy_score(y_test, y_pred_lr)
lr_f1 = f1_score(y_test, y_pred_lr, average='weighted')
print("Logistic Regression Accuracy:", lr_accuracy)
print("Logistic Regression F1 Score:", lr_f1)
plot_confusion_matrix(cm_lr, title='Logistic Regression Confusion Matrix')

cm_svm = confusion_matrix(y_test, y_pred_svm)
svm_accuracy = accuracy_score(y_test, y_pred_svm)
svm_f1 = f1_score(y_test, y_pred_svm, average='weighted')
print("SVM Accuracy:", svm_accuracy)
print("SVM F1 Score:", svm_f1)
plot_confusion_matrix(cm_svm, title='SVM Confusion Matrix')

models = {
    "Naive Bayes Multinomial": (nbm_accuracy, nbm_f1),
    "J48": (j48_accuracy, j48_f1),
    "Logistic Regression": (lr_accuracy, lr_f1),
    "SVM": (svm_accuracy, svm_f1)
}

best_model_accuracy = max(models, key=lambda k: models[k][0])
best_model_f1 = max(models, key=lambda k: models[k][1])

print(f"The best model based on accuracy is: {best_model_accuracy} with Accuracy: {models[best_model_accuracy][0]} and F1 Score: {models[be
print(f"The best model based on F1 score is: {best_model_f1} with Accuracy: {models[best_model_f1][0]} and F1 Score: {models[best_model_f1]
```

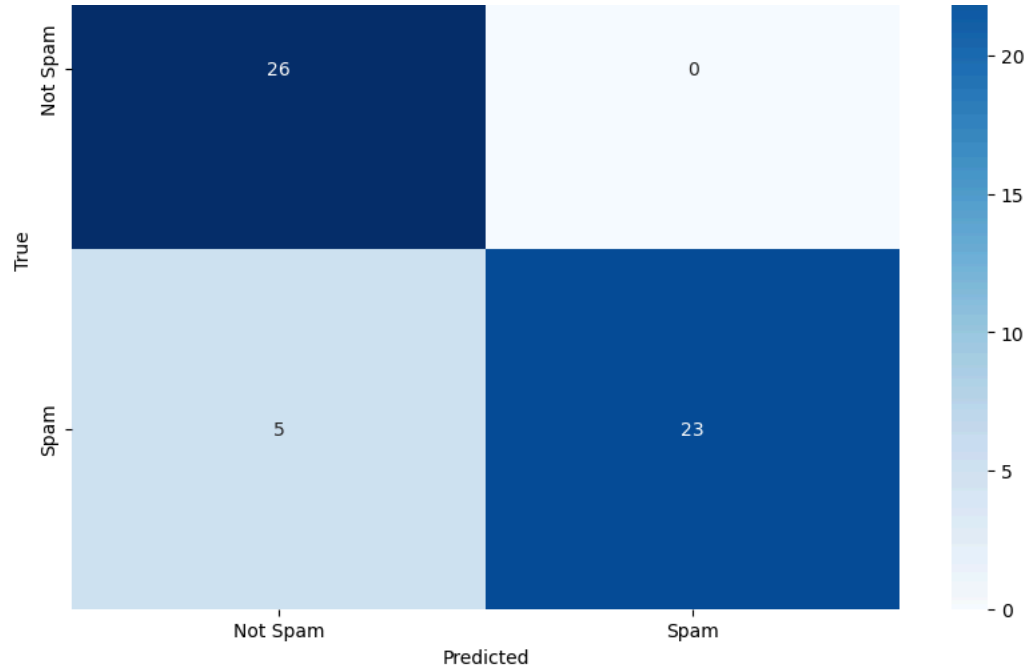
```
Naive Bayes Multinomial Accuracy: 0.9629629629629629
Naive Bayes Multinomial F1 Score: 0.9629629629629629
```



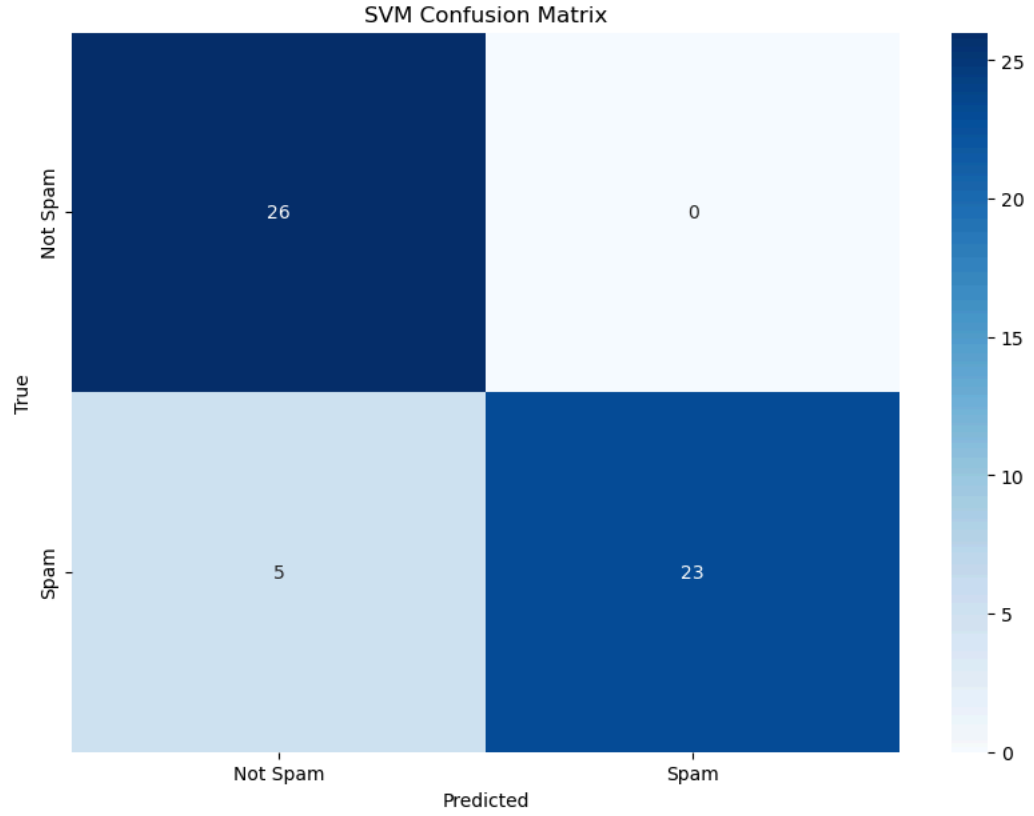

```
J48 Accuracy: 0.9629629629629629
J48 F1 Score: 0.9628607918263091
```




```
Logistic Regression Accuracy: 0.9074074074074074
Logistic Regression F1 Score: 0.9069296334518213
```




```
SVM Accuracy: 0.9074074074074074
SVM F1 Score: 0.9069296334518213
```




```
The best model based on accuracy is: Naive Bayes Multinomial with Accuracy: 0.9629629629629629 and F1 Score: 0.9629629629629629
```