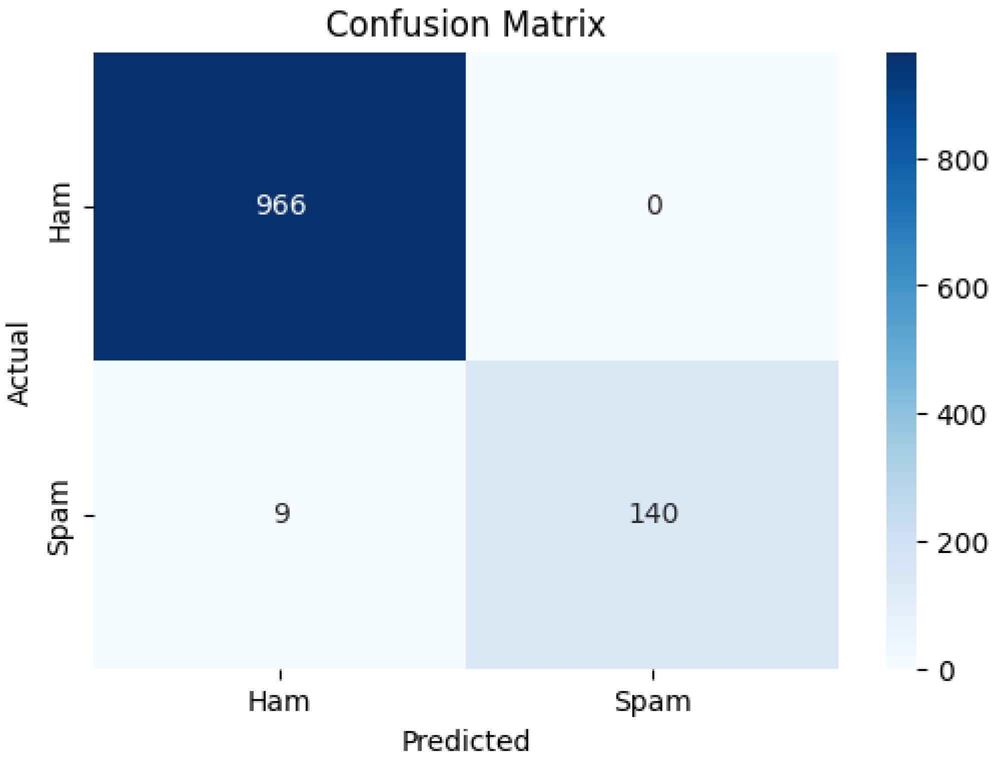
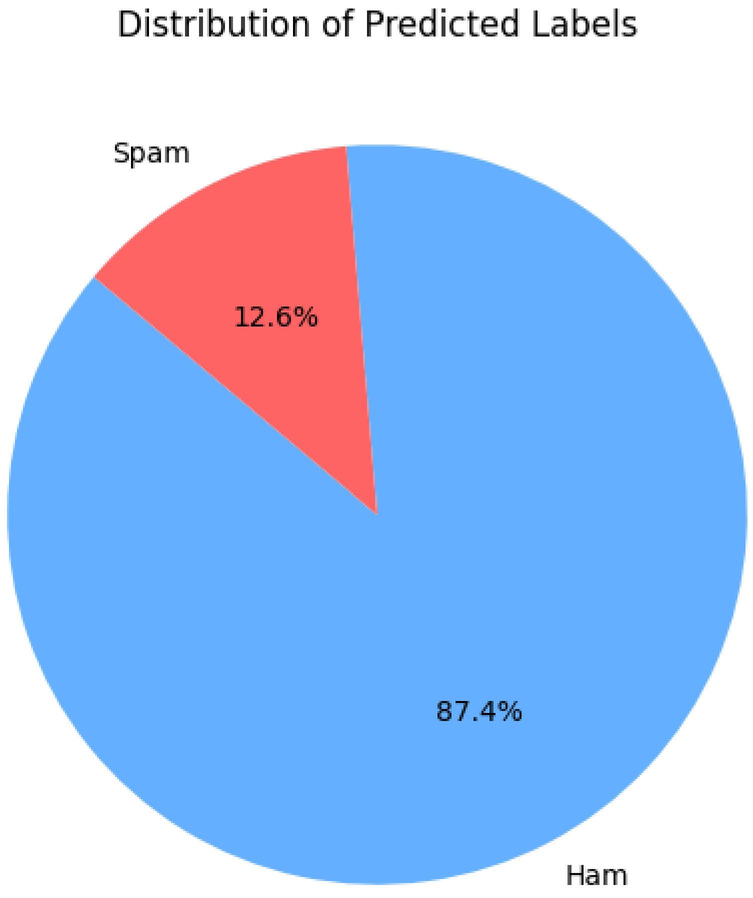
```
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.feature_extraction.text import CountVectorizer
from sklearn.naive_bayes import MultinomialNB
from sklearn.metrics import accuracy_score, confusion_matrix, classification_report
import matplotlib.pyplot as plt
import seaborn as sns

# Load the dataset
data = pd.read_csv('/content/spam.csv')

# Adjust column names based on the actual dataset
data = data[['Category', 'Message']]  # Update these names as per your dataset
data.columns = ['label', 'text']

# Encode labels: ham -> 0, spam -> 1
data['label'] = data['label'].map({'ham': 0, 'spam': 1})

# Split the dataset into training and testing sets
X_train, X_test, y_train, y_test = train_test_split(data['text'], data['label'], test_size=0.2, random_state=42)

# Convert text data to numerical data
vectorizer = CountVectorizer()
X_train_vec = vectorizer.fit_transform(X_train)
X_test_vec = vectorizer.transform(X_test)

# Initialize and train the classifier
clf = MultinomialNB()
clf.fit(X_train_vec, y_train)

# Make predictions on the test set
y_pred = clf.predict(X_test_vec)

# Calculate accuracy
accuracy = accuracy_score(y_test, y_pred)
print(f'Accuracy: {accuracy * 100:.2f}%')

# Generate confusion matrix
cm = confusion_matrix(y_test, y_pred)

# Plot confusion matrix
plt.figure(figsize=(6, 4))
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')
plt.show()

# Print classification report
print(classification_report(y_test, y_pred, target_names=['Ham', 'Spam']))

# Count the occurrences of each predicted label
pred_counts = pd.Series(y_pred).value_counts()

# Plot the pie chart
plt.figure(figsize=(6, 6))
plt.pie(pred_counts, labels=['Ham', 'Spam'], autopct='%0.1f%%', colors=['#66b3ff', '#ff6666'], startangle=140)
plt.title('Distribution of Predicted Labels')
plt.show()

new_email = ["HEllo Good Morning sir ree entry in  a wkly comp to "]

# Convert the new email text to numerical data
new_email_vec = vectorizer.transform(new_email)

# Make a prediction
prediction = clf.predict(new_email_vec)

# Interpret the prediction
label = 'Spam' if prediction[0] == 1 else 'Ham'
print(f'The email is classified as: {label}')

```

```
Accuracy: 99.19%
```



```
              precision    recall  f1-score   support

         Ham       0.99      1.00      1.00       966
        Spam       1.00      0.94      0.97       149

    accuracy                           0.99      1115
   macro avg       1.00      0.97      0.98      1115
weighted avg       0.99      0.99      0.99      1115
```



```
The email is classified as: Spam
```