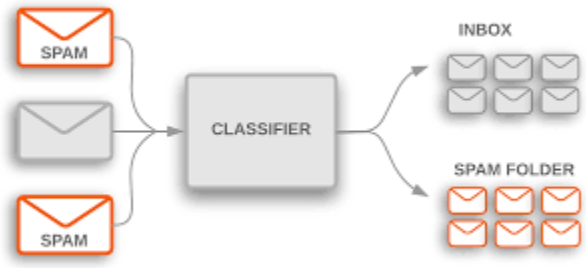
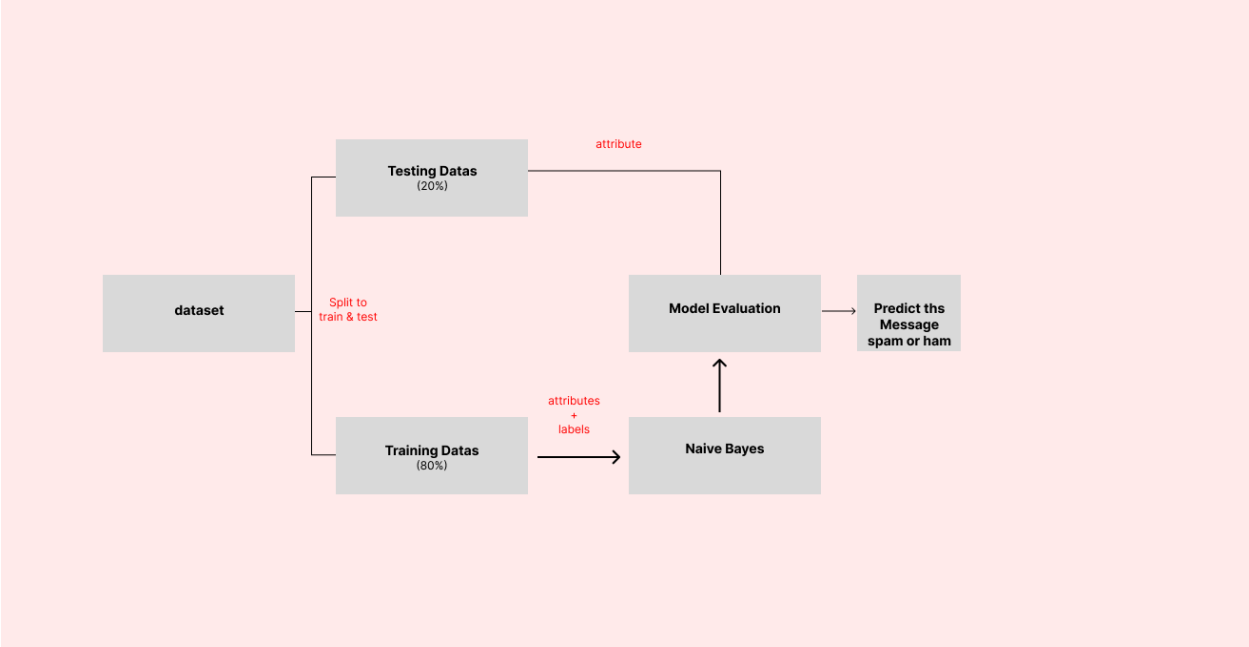
Unmasking Spam Message Detection

Diagram:



Flow char:



Code:

```
# import modules

# processing the messages
import nltk
# read the data set
import pandas as pd
from collections import defaultdict
from nltk.corpus import stopwords
from nltk.stem import PorterStemmer
from sklearn.model_selection import train_test_split
from sklearn.metrics import accuracy_score
```

[1] ✓ 3.7s Python

```
# Download necessary NLTK resources
nltk.download('punkt')
nltk.download('stopwords')
```

[2] ✓ 0.3s Python

```
[nltk_data] Downloading package punkt to C:\Users\Arun
[nltk_data]     Venkat\AppData\Roaming\nltk_data...
[nltk_data]   Package punkt is already up-to-date!
[nltk_data] Downloading package stopwords to C:\Users\Arun
[nltk_data]     Venkat\AppData\Roaming\nltk_data...
[nltk_data]   Package stopwords is already up-to-date!

True
```

```
class NaiveBayesClassifier:
    def __init__(self):
        self.word_freq = defaultdict(lambda: defaultdict(int))
        self.class_freq = defaultdict(int)
        self.vocab = set()
        self.num_messages = 0
        self.num_classes = 0
```

```
    def tokenize_text(self, text):
        return nltk.word_tokenize(text)

    def preprocess_text(self, text):
        stop_words = set(stopwords.words('english'))
        stemmer = PorterStemmer()
        words = self.tokenize_text(text.lower())
        words = [stemmer.stem(word) for word in words if word.isalpha() and word not in stop_words]
        return words

    def train(self, X_train, y_train):
        self.num_messages = len(X_train)
        self.num_classes = len(set(y_train))
        for message, label in zip(X_train, y_train):
            words = self.preprocess_text(message)
            self.class_freq[label] += 1
            for word in words:
                self.word_freq[word][label] += 1
                self.vocab.add(word)

    def predict(self, X_test):
        y_pred = []
        for message in X_test:
            words = self.preprocess_text(message)
            scores = {label: 0 for label in self.class_freq}
            for word in words:
                if word in self.vocab:
                    for label in self.class_freq:
                        scores[label] += self.word_freq[word][label]
            predicted_label = max(scores, key=scores.get)
            y_pred.append(predicted_label)
        return y_pred
```

[3] ✓ 0.0s Python

Output:

```
if __name__ == "__main__":
    # Load the dataset
    data = pd.read_table('unmasked_data', sep='\t', names=['label', 'message'])
    # Split the dataset into training and testing sets
    X_train, X_test, y_train, y_test = train_test_split(data['message'], data['label'], random_state=42)
    # Train the Naive Bayes classifier
    classifier = NaiveBayesClassifier()
    classifier.train(X_train, y_train)
    # Make predictions on the test set
    y_pred = classifier.predict(X_test)
    # Calculate and print the accuracy
    accuracy = accuracy_score(y_test, y_pred)
    print(f"Accuracy: {accuracy:.2f}")
```

[10] ✓ 6.3s Python

```
Accuracy: 0.93
```

```
y_pred
```

[9] ✓ 0.0s Python

```
['ham',
 'ham',
 'ham',
 'ham',
 'ham',
 'ham',
 'ham',
 'ham',
 'ham',
 'ham',
 'ham',
 'ham',
 'ham',
 'ham',
 'ham',
 'spam',
 'ham',
 'ham',
```