AI-POWERED SPAM CLASSIFIER

TEAM MEMBER

R.ANITHA-950921104001

Project Title: AI-POWERED SPAM CLASSIFIER

Phase 4 : Submission Document

TOPIC: Continue building the AI- Powered Spam Classifier by feature engineering, model training, and evaluation.

Phase 4 : Development Part 2

INTRODUCTION:

★An AI-powered spam classifier is a machine learning model that is trained to identify and classify spam messages. Spam messages are unsolicited messages that are often used to spread malware, phishing attacks, or other malicious content. AI spam classifiers can help to reduce spam by filtering out spam messages before they reach the user's inbox.

★Feature selection is the process of selecting a subset of features from the original set of features that are most relevant to the target variable. This can be done for a variety of reasons, such as to improve the performance of a machine learning model, to reduce the computational cost of training a model, or to improve the interpretability of a model.

★Model evaluation is the process of assessing the performance of a machine learning model on a held-out test set. This is an important step in the machine learning workflow, as it helps to ensure that the model is generalizing well to new data and is not simply overfitting to the training data.

★ Model training is the process of teaching a machine learning model to perform a specific task. In the case of a spam classifier, the task is to predict whether a given message is spam or ham.To train a spam classifier, you will need a labeled dataset of messages. The dataset should contain both spam messages and ham messages. Each message should be labeled with its correct class (spam or ham).

Given Dataset

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

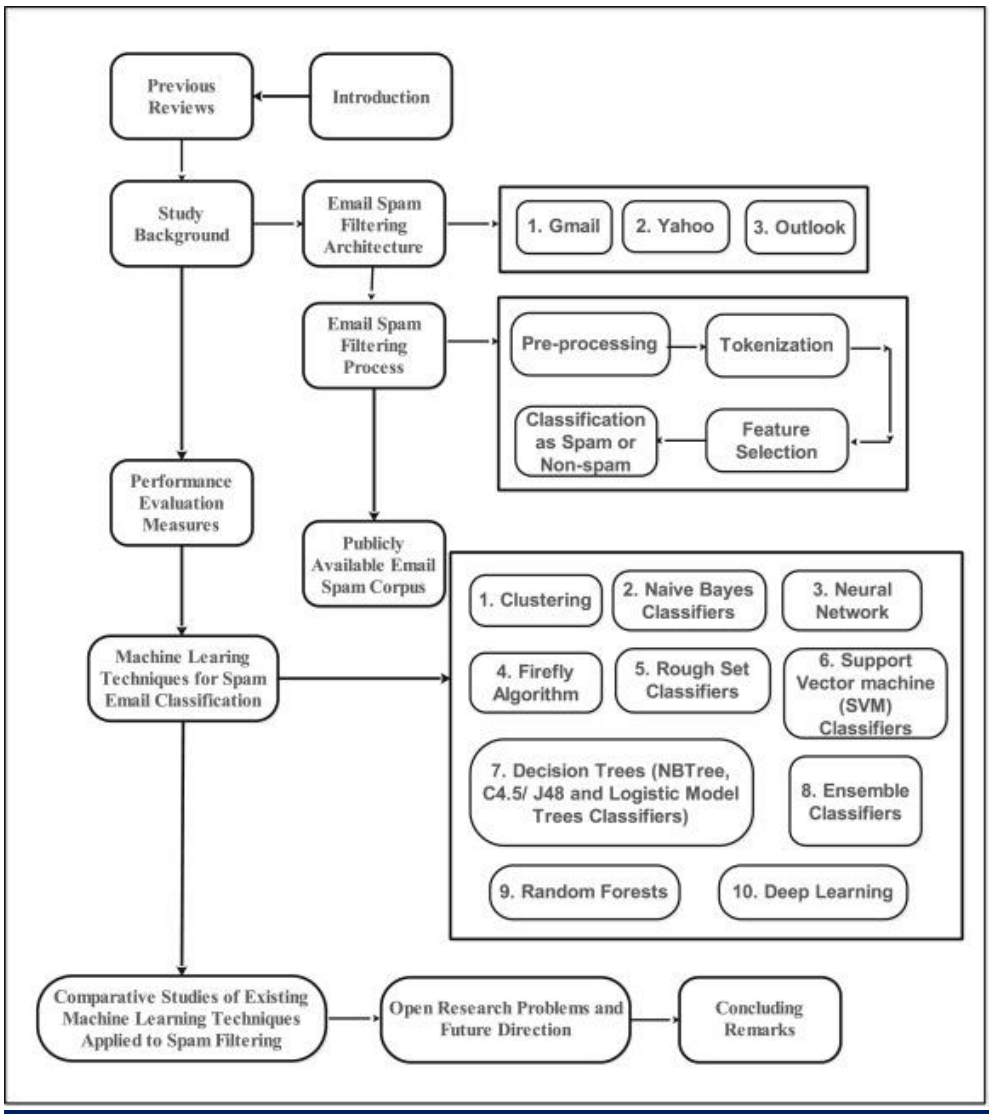
5572 rows × 2 columns

Overview of the process:

The following is an overview of the process of building a spam classifier model by feature selection, model training, and evaluation:

1. **Prepare the data**: This includes cleaning the data, removing outliers, and handling missing values.

2. **Perform feature selection:** This can be done using a variety of methods, such as correlation analysis, information gain, and recursive feature elimination.
3. **Train the model:** There are many different machine learning algorithms that can be used for house price prediction. Some popular choices include linear regression, random forests, and gradient boosting machines.
4. **Evaluate the model:** This can be done by calculating the mean squared error (MSE) or the root mean squared error (RMSE) of the model's predictions on the held-out test set.
5. **Deploy the model:** Once the model has been evaluated and found to be performing well, it can be deployed to production so that it can be used to predict the house prices of new houses.



PROCEDURE:

Feature selection:

Identify the target variable. This is the variable that you want to predict, such as spam classifier.

Explore the data. This will help you to understand the relationships between the different features and the target variable. You can use data visualization and correlation analysis to identify features that are highly correlated with the target variable.

Remove redundant features. If two features are highly correlated with each other, then you can remove one of the features, as they are likely to contain redundant information.

Remove irrelevant features. If a feature is not correlated with the target variable, then you can remove it, as it is unlikely to be useful for prediction.

Feature Selection:

Checking the missing values

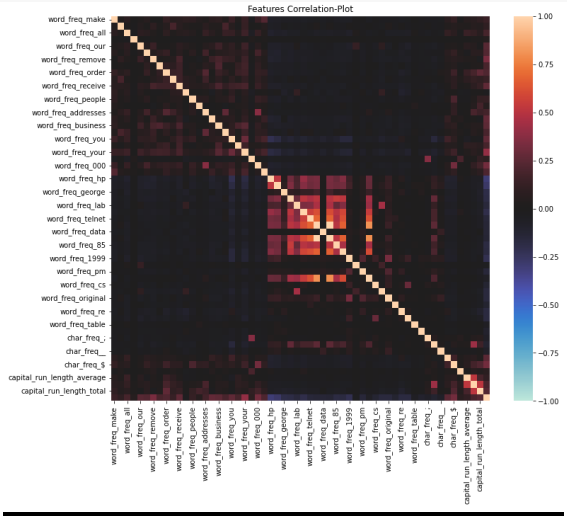
In[1]:
```
#Checking missing values
df.isnull().sum()
```

Out[1]:
```
target    0
text      0
dtype: int64
```

PROGRAM :

Checking the correlation

```
features = df.columns
plt.figure(figsize=[12,10])
plt.title('Features Correlation-Plot')
sns.heatmap(df[features].corr(), vmin=-1, vmax=1, center=0) #,
plt.show()
```



Model training:

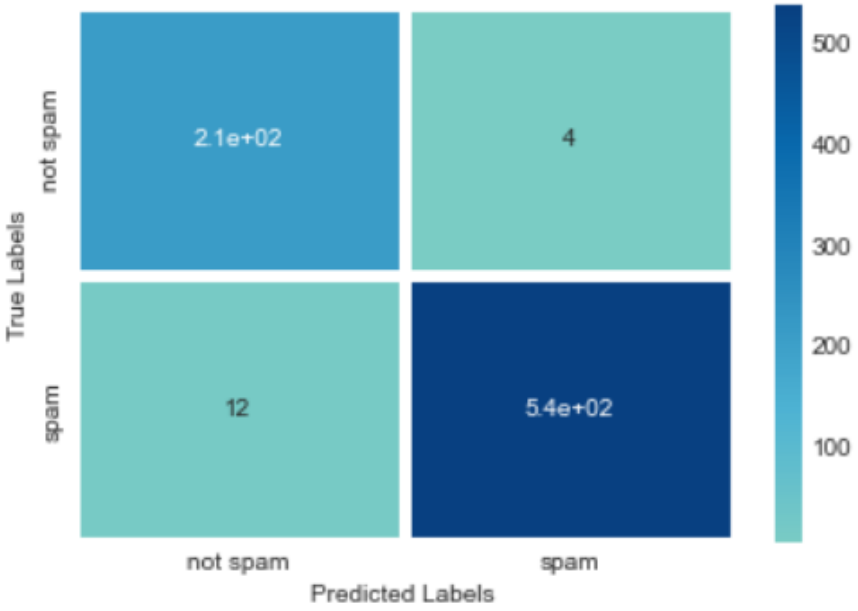
There are many different machine learning models that can be used for AI spam detection. Some of the most common models include:

- **Naive Bayes**: Naive Bayes is a simple but effective model for spam detection. It works by calculating the probability of a given email being spam based on the presence or absence of certain words and phrases.
- **Support vector machines (SVMs):** SVMs are a more complex model than Naive Bayes, but they can be more effective at detecting spam. SVMs work by finding a hyperplane that separates spam emails from non-spam emails.
- **Decision trees:** Decision trees are another type of model that can be used for spam detection. Decision trees work by constructing a tree of rules that can be used to classify emails as spam or non-spam.
- **Random forests:** Random forests are an ensemble model that combines the predictions of multiple decision trees. Random forests are often more effective than individual decision trees at detecting spam.
- **Deep learning models:** Deep learning models are a type of machine learning model that can be used for a variety of tasks, including spam detection. Deep learning models are typically trained on large datasets of labeled data. Once trained, deep learning models can be used to classify new data with a high degree of accuracy.

Naive Bayes Model

In[2]:
```
naive = GaussianNB()
naive.fit(X_train.toarray(), y_train)
preds = naive.predict(X_test.toarray())
print(accuracy_score(preds,y_test))
```

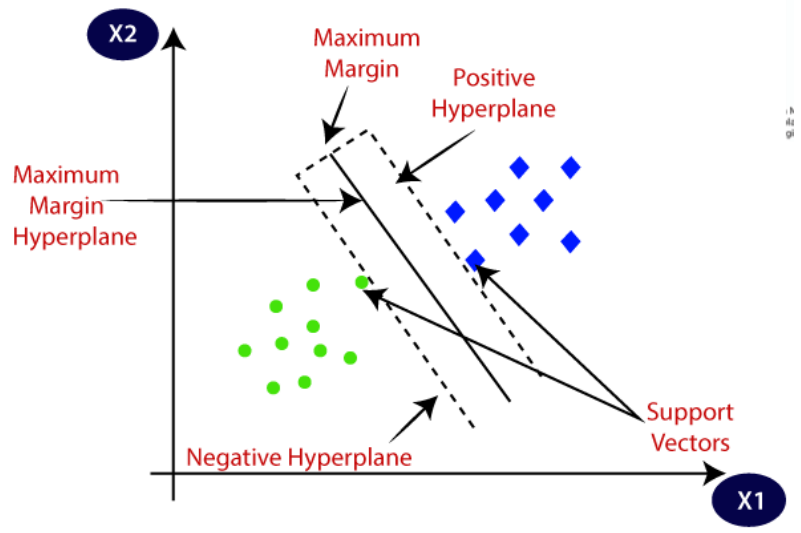
Out[2]: 0.9409326424870467

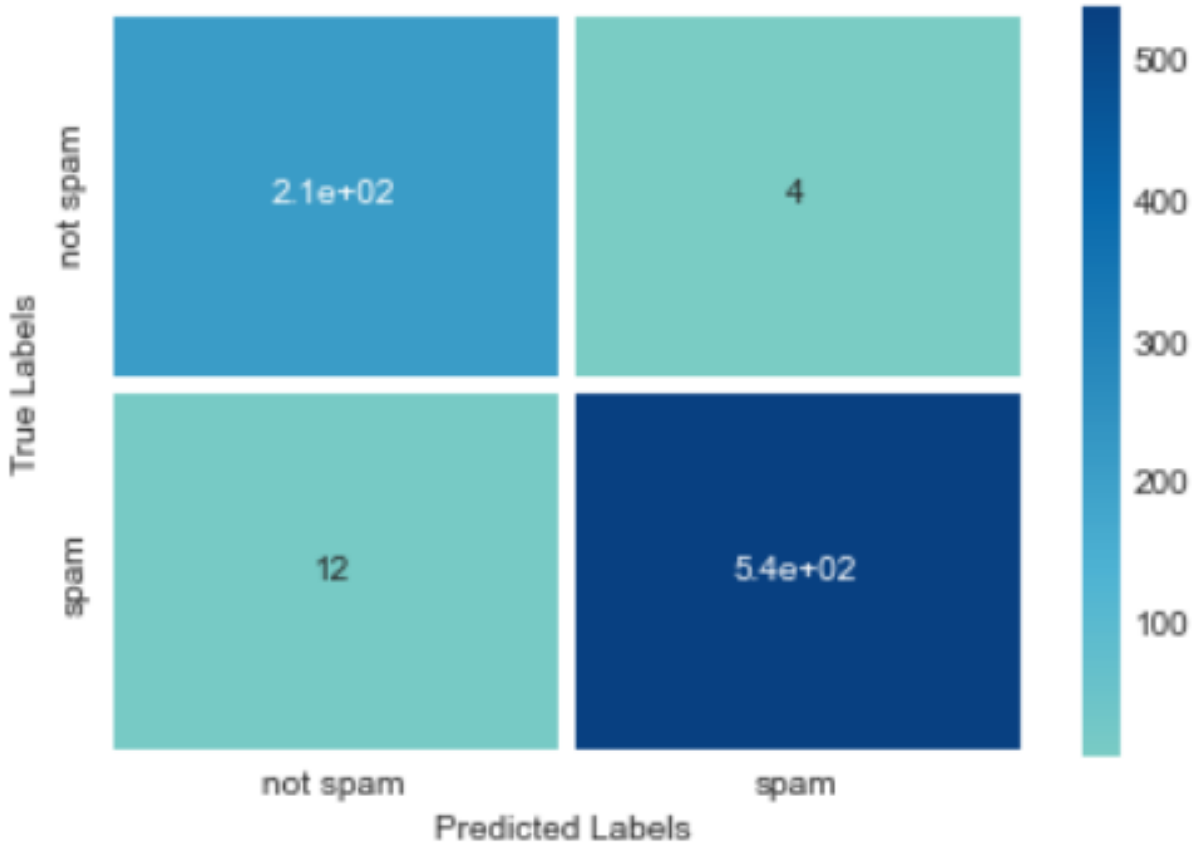


Support vector machines (SVMs)

In[3]:
```
SVM = SVC()
SVM.fit(X_train, y_train)
preds = SVM.predict(X_test)
print(accuracy_score(preds,y_test))
```

Out[3]: 0.9989637305699481





Decision trees:

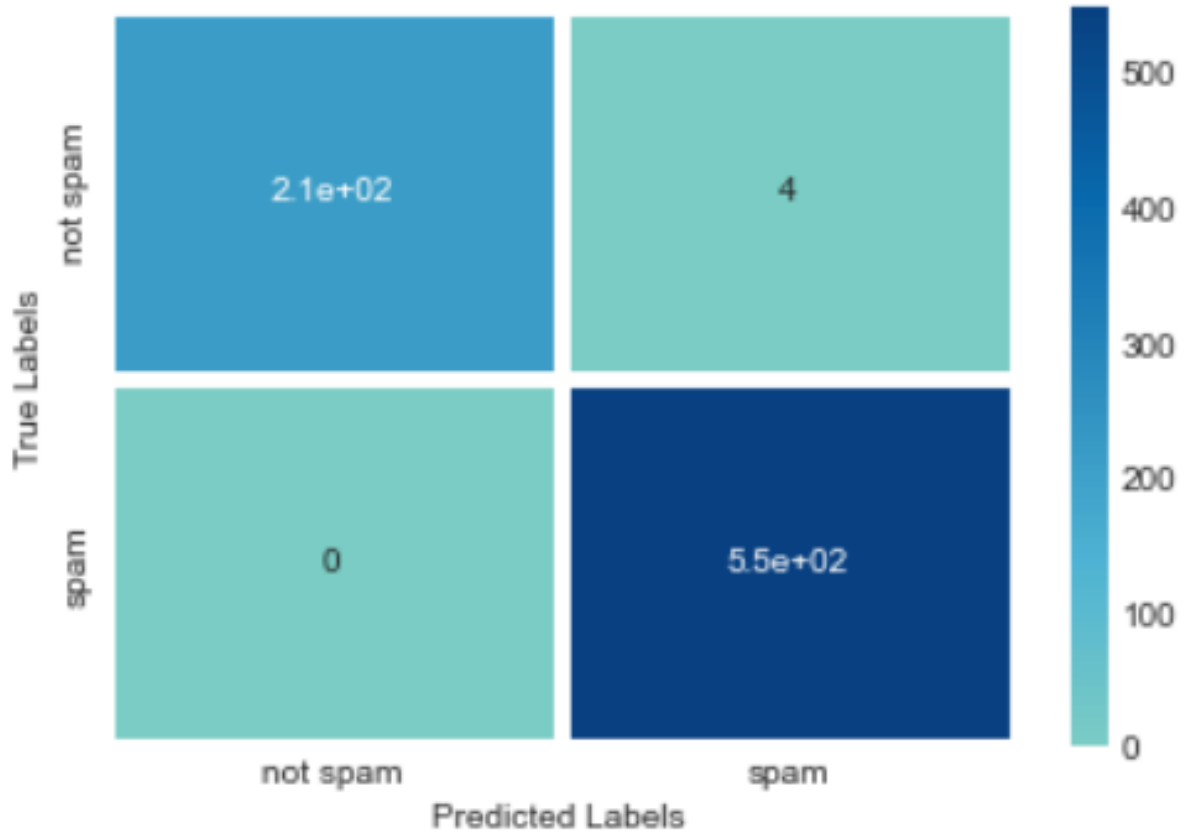
In[4]:
```
tree = DecisionTreeClassifier()
tree.fit(X_train,y_train)
preds = tree.predict(X_test)
print(accuracy_score(preds,y_test))
```

Out[4]: 0.9787564766839378

Random forests:

In[5]:
```
forest = RandomForestClassifier()
forest.fit(X_train,y_train)
preds = forest.predict(X_test)
print(accuracy_score(preds,y_test))
```

Out[5]: 1.0



Model Training

To train an AI spam classifier, you will need a dataset of labeled email messages, where each message is labeled as either "spam" or "ham" (non-spam). Once you have your dataset, you can follow these steps:

1. Preprocess the data. This may involve cleaning the text, removing stop words, and stemming or lemmatizing the words.
2. Extract features from the data. This could involve using a bag-of-words approach, where each feature represents a word in the vocabulary, or a more sophisticated approach, such as using TF-IDF or word embeddings.
3. Choose a machine learning algorithm. Some popular algorithms for spam classification include Naive Bayes, Support Vector Machines, and Random Forests.
4. Train the model. This involves feeding the training data to the algorithm and allowing it to learn the patterns in the data.
5. Evaluate the model. Once the model is trained, you can evaluate its performance on the test set. This will give you an idea of how well the model will generalize to new data.
6. Deploy the model. Once you are satisfied with the model's performance, you can deploy it to production. This may involve integrating the model into an email server or other application.

Here are some additional tips for training an effective AI spam classifier:

- Use a large and diverse dataset. The more data you have, the better your model will be able to learn the patterns in spam emails. Make sure your dataset includes a variety of spam emails, such as phishing emails, advertising emails, and chain letters.
- Handle imbalanced datasets. Spam datasets are often imbalanced, with many more ham emails than spam emails. This can make it difficult for the model to learn the patterns in spam emails. To address this, you can use techniques such as oversampling the spam emails or undersampling the ham emails.
- Use feature selection. Not all features are equally important for spam classification. You can use feature selection techniques to identify the most important features and remove the less important features. This can improve the performance of your model and reduce the training time.
- Use a validation set. A validation set is a small set of data that is used to evaluate the model during training. This helps to prevent overfitting, which is when the model learns the training data too well and does not generalize well to new data.
- Tune the hyperparameters. Hyperparameters are parameters of the machine learning algorithm that affect its performance. You can tune the hyperparameters using a grid search or random search.

Dividing Dataset in to features and target variable:

Machine Learning

```
Create a function to perform classification metrics
def show_metrics(y_true, y_pred, grid_search=None):
    from sklearn.metrics import (classification_report,
                                 confusion_matrix,
                                 ConfusionMatrixDisplay)

    print('-' * 20)
    print(classification_report(y_true, y_pred))
    print(confusion_matrix(y_true, y_pred))

    if grid_search:
        print('-' * 20)
        print(grid_search.best_params_)
```

In [56]:
Linkcode

```
# SVM metrics
best_svm = model_svm.best_estimator_
y_pred_svm = best_svm.predict(X_test)

show_metrics(y_test, y_pred_svm, model_svm)
```

OUTPUT:

```
precision    recall  f1-score   support

           0       0.99      1.00      0.99      1464
           1       0.97      0.93      0.95       208

    accuracy                           0.99      1672
   macro avg       0.98      0.96      0.97      1672
weighted avg       0.99      0.99      0.99      1672

[[1458    6]
 [  14  194]]
--------------------
{'classifier__C': 10.0, 'tfidf__ngram_range': (1, 2), 'tfidf__stop_words': None}
```

Split the data into training and test sets. The training set will be usedto train themodel, and the test set will be used to evaluate the performance of the model.

Program:

```
clas0,clas1=data["Category"].value_counts()
df0=data[data["Category"]==0]
df1=data[data["Category"]==1]
df1=df1.sample(clas0,replace=True)
```

```
data = pd.concat([df0,df1])

labels = data["Category"]
data = data["text"]
labels.value_counts(
```

```
0    4825
1    4825
Name: Category, dtype: int64
```

```
X_train, X_test, y_train, y_test = train_test_split(data, labels, test_size=0.2, random_
state=42)
```

Train the model on the training set. This involves feeding the training data to the model and allowing it to learn the relationships between the features and the target variable.

Evaluate the model on the test set. This involves feeding the test data to the model and measuring how well it predicts the target variable.

MODEL EVALUATION: To evaluate an AI spam classifier, you can use a variety of metrics, including:

- Accuracy: This is the percentage of emails that the classifier correctly predicts as spam or ham.
- Precision: This is the percentage of emails that the classifier predicts as spam that are actually spam.
- Recall: This is the percentage of spam emails that the classifier correctly predicts as spam.
- F1 score: This is a harmonic mean of precision and recall.

These metrics can be calculated using a confusion matrix, which is a table that shows the number of emails that were correctly and incorrectly predicted by the classifier.

Here is an example of a confusion matrix for a spam classifier:

Predicted	Actual
Spam	Spam
Spam	Ham

Ham Spam

Ham Ham

drive_spreadsheetExport to Sheets

Accuracy can be calculated as follows:

Accuracy = (TP + TN) / (TP + FP + FN + TN)

Precision can be calculated as follows:

Precision = TP / (TP + FP)

Recall can be calculated as follows:

Recall = TP / (TP + FN)

F1 score can be calculated as follows:

F1 score = 2 * (Precision * Recall) / (Precision + Recall)

It is important to note that no single metric is perfect for evaluating a spam classifier. For example, a classifier with high accuracy may not have high recall, meaning that it is missing some spam emails. A classifier with high recall may not have high precision, meaning that it is incorrectly classifying some ham emails as spam.

The best approach is to choose a set of metrics that are important to your specific needs. For example, if you are concerned about missing spam emails, you may want to focus on recall. If you are concerned about incorrectly classifying ham emails as spam, you may want to focus on precision.

Once you have chosen a set of metrics, you can use them to compare different spam classifiers. You can also use the metrics to track the performance of your spam classifier over time and to identify areas for improvement.

Here are some additional tips for evaluating an AI spam classifier:

- Use a held-out test set. The test set should be a set of emails that the classifier has not seen before. This will help to ensure that the evaluation results are unbiased.
- Use multiple metrics. As mentioned above, no single metric is perfect for evaluating a spam classifier. Use a set of metrics that are important to your specific needs.

- Compare different classifiers. Once you have evaluated your spam classifier, compare its performance to other spam classifiers. This will help you to identify the best classifier for your needs.
- Track performance over time. Monitor the performance of your spam classifier over time and identify any areas for improvement.

  Perform EDA to understand your data better. This includes checking for missing values, exploring the data's statistics, and visualizing it to identify patterns.

Program:

```
from sklearn.model_selection

import train_test_split features = df1.drop('v1' , axis = 1)

label = df1['v1']

x_train , x_test , y_train , y_test = train_test_split(features , label , test_size =
0.3)

print(f"X train shape : {x_train.shape}\nY train shape : {y_train.shape}\nX test
shape : {x_test.shape}\nY test shape : {y_test.shape}")
```

EDA

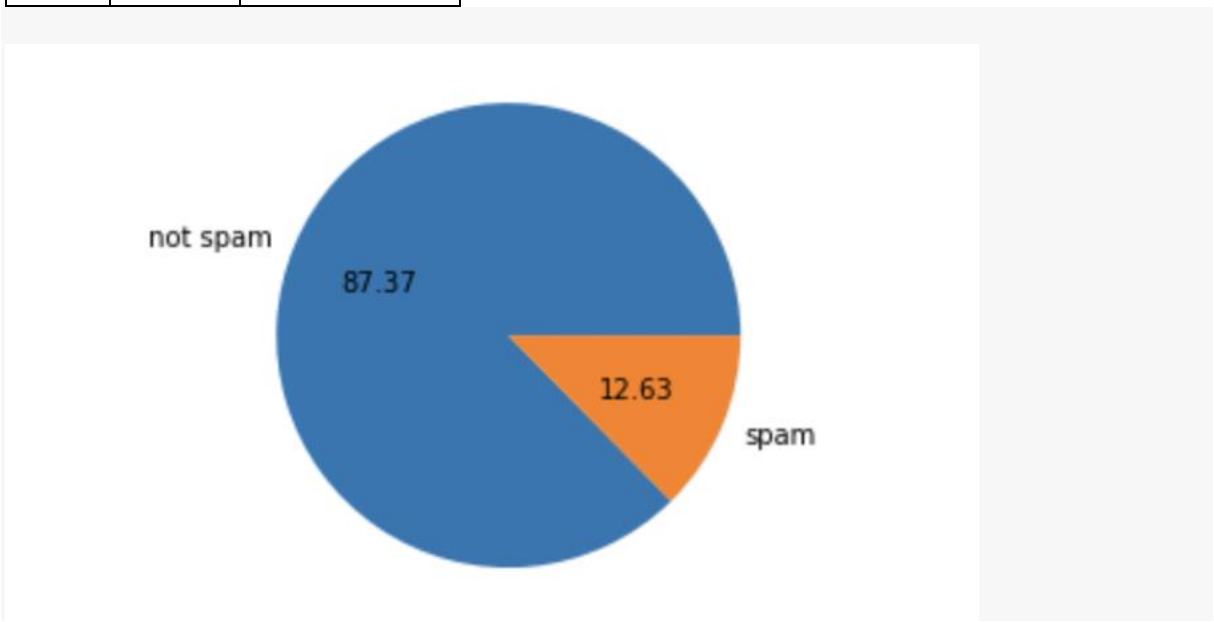
```
messages.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 5572 entries, 0 to 5571
Data columns (total 2 columns):
 #   Column   Non-Null Count  Dtype
---  ------   --------------  -----
 0   label    5572 non-null   object
 1   message  5572 non-null   object
dtypes: object(2)
memory usage: 87.2+ KB
```

There are total 5572 SMS in this dataset with 2 columns label and message.

```
messages.describe()
```

label	message	
count	5572	5572
unique	2	5169
top	ham	Sorry, I'll call later
freq	4825	30



Feature Engineering:

Feature engineering is the process of transforming raw data into features that are more informative and useful for machine learning algorithms. In the context of AI spam classifiers, feature engineering can involve a variety of techniques, such as:

- **Text preprocessing:** This may involve cleaning the text, removing stop words, and stemming or lemmatizing the words.
- **Feature extraction:** This involves extracting features from the text, such as the presence of certain words or phrases, the length of the email, and the number of links.
- **Feature selection:** This involves selecting the most important features for spam classification.

- **Feature engineering:** This involves creating new features from the existing features. For example, you could create a feature that represents the percentage of words in the email that are on a list of known spam words.

  Here are some specific examples of features that can be used for AI spam classification:

- **Presence of certain words or phrases:** Some words and phrases, such as "free money" and "click here," are commonly used in spam emails.
- **Length of the email:** Spam emails are often shorter than ham emails.
- **Number of links:** Spam emails often contain a large number of links.
- **Presence of attachments:** Spam emails often contain attachments, such as malware or phishing links.
- **Sender's email address:** Spam emails often come from email addresses that are known to be spammers.
- **Receiver's email address**: Spam emails are often sent to a large number of recipients, including email addresses that are not known to the sender.

Features to perform

Model Training

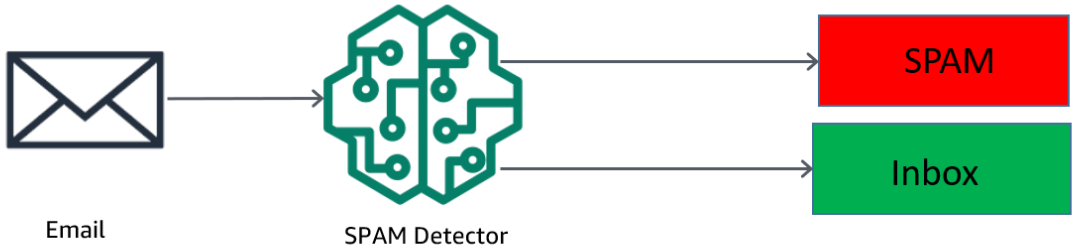
The features that you choose to train your machine learning model will depend on the specific problem that you are trying to solve. However, there are some general principles that you can follow when selecting features:

- **Choose features that are relevant to the problem**. The features should be able to help the model to predict the target variable.
- **Choose features that are informative**. The features should contain enough information to allow the model to make accurate predictions.
- **Choose features that are not correlated**. Correlated features can lead to overfitting, which is when the model learns the training data too well and does not generalize well to new data.

Here are some specific examples of features that can be used for different machine learning problems:

- **Image classification:** Pixels, edges, shapes, colors, textures
- **Natural language processing**: Words, phrases, sentences, part-of-speech tags, dependency trees
- **Recommendation systems:** User ratings, item attributes, user-item interactions
- **Time series forecasting:** Historical data, seasonal patterns, trends

- **Fraud detection:** Transaction amounts, locations, types of transactions, user behaviour.



Conclusion:

In the quest to build an accurate and reliable Spam detection model, we have embarked on a journey that encompasses critical phases, from feature selection to model training and evaluation. Each of these stages plays an indispensable role in crafting a model that can provide meaningful insights and estimates for one of the most significant financial decisions individuals and businesses.
Model training is where the model's predictive power is forged. We have explored a variety of regression techniques, fine-tuning their parameters to learn from historical data patterns. This step allows the model to capture the intricate relationships between features giving it the ability to generalize beyond the training dataset.