AI-POWERED SPAM CLASSIFIER

TEAM MEMBER

B.Chellammal @ selvi - 950921104008

Project Title: AI-POWERED SPAM CLASSIFIER

Objectives

Spam classification is a challenging task, as spammers are constantly evolving their techniques. AI-powered spam classifiers can be used to accurately distinguish between spam and non-spam messages in emails or text messages. The goal of this project is to build an AI-powered spam classifier that can reduce the number of false positives and false negatives while achieving a high level of accuracy.

Phase 1: Data Preprocessing and Feature Engineering

Modules

The following modules are required to build an AI-powered spam classifier:

1.Data Source:

A good data source for house price prediction using machine learning should be Accurate, Complete, Covering the geographic area of interest, Accessible.

Dataset Link: https://www.kaggle.com/datasets/uciml/sms-spam-collection-dataset

V1	V2	Unnamed: 2	Unnamed: 3	Unnamed: 4	
0	ham	Go until jurong point, crazy.. Available only ...	NaN	NaN	NaN
1	ham	Ok lar... Joking wif u oni...	NaN	NaN	NaN
2	spam	Free entry in 2 a wkly comp to win FA Cup fina...	NaN	NaN	NaN

V1	V2	Unnamed: 2	Unnamed: 3	Unnamed: 4	
3	ham	U dun say so early hor... U c already then say...	NaN	NaN	NaN
4	ham	Nah I don't think he goes to usf, he lives aro...	NaN	NaN	NaN

2.Data Preprocessing:

This module collects a dataset of labeled spam and non-spam messages. The data is then cleaned and pre processed to remove irrelevant characters and normalize the text.

Tokenization:

This step involves splitting the text into individual words or tokens. This can be done using a variety of different tokenization techniques, such as regular expressions or rule-based tokenizers.

Stop word removal:

Stop words are common words that do not add much meaning to the text, such as articles, prepositions, and pronouns. Stop words are typically removed from the text before training the machine learning model.

Lowercasing:

This step involves converting all of the words in the text to lowercase. This is important because many machine learning algorithms are case-insensitive.

PYTHON PROGRAM

# import Libraries

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

```python
# Reading the dataset
df = pd.read_csv('/kaggle/input/sms-spam-collection-dataset/spam.csv', encoding='
SO-8859-1')
df.head()

# Separating X and y

X = df['v2']
y = df['v1']
display(X, y)

# Encoding the Labels

from sklearn.preprocessing
import LabelEncoder
le = LabelEncoder()
y = le.fit_transform(y)
display(y)
```

OUTPUT:

```
0       Go until jurong point, crazy.. Available only ...
1                           Ok lar... Joking wif u oni...
2       Free entry in 2 a wkly comp to win FA Cup fina...
3       U dun say so early hor... U c already then say...
4       Nah I don't think he goes to usf, he lives aro...
                              ...

5567    This is the 2nd time we have tried 2 contact u...
5568                Will Ì_ b going to esplanade fr home?
5569    Pity, * was in mood for that. So...any other s...
5570    The guy did some bitching but I acted like i'd...
5571                           Rofl. Its true to its name

Name: v2, Length: 5572, dtype: object
0        ham
1        ham
2       spam
3        ham
4        ham
        ...

5567    spam
5568     ham
5569     ham
5570     ham
5571     ham
Name: v1, Length: 5572, dtype: object
```

3.Feature extraction:

This module extracts features from the pre processed data. Features are the characteristics of the data that the machine learning model will use to learn to distinguish between spam and non-spam messages.

PROGRAM

```
# Create a CountVectorizer for text feature extraction

featurizer = CountVectorizer(decode_error='ignore')
x_train = featurizer.fit_transform(df_train)
x_test = featurizer.transform(df_test)

#Check the result of feature extraction
x_train
```

OUTPUT:

```
<4457x7735 sparse matrix of type '<class 'numpy.int64'>'
        with 58978 stored elements in Compressed Sparse Row format>
```

Model Selection:

Naive Bayes Model

```
# Create the Multinomial Naive Bayes model
model = MultinomialNB()
model.fit(x_train, y_train)
```

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)
```

<u>OUTPUT</u>:

```
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}
```

<u>Model evaluation</u>:

This module evaluates the performance of the trained model on a held-out test set of labeled messages. This gives an idea of how well the model will generalize to new data.

Reducing false positives and false negatives

There are a variety of techniques that can be used to reduce false positives and false negatives, such as:

- Using a balanced dataset: A balanced dataset has an equal number of spam and non-spam messages. This helps to prevent the model from becoming biased towards one class or the other.
- Using a validation set: A validation set is a subset of the training data that is used to tune the parameters of the model. This helps to prevent overfitting, which can lead to poor performance on new data.
- Using ensemble methods: Ensemble methods combine the predictions of multiple models to produce a more accurate prediction. This can help to reduce both false positives and false negatives.

Achieving a high level of accuracy

To achieve a high level of accuracy, it is important to use a large and diverse dataset, a variety of features, and a well-tuned machine learning model. It is also important to monitor the performance of the model over time and update it with new data and features as needed.

Conclusion:

By following the steps outlined in this abstract and using the modules described above, it is possible to build an AI-powered spam classifier that can accurately distinguish between spam and non-spam messages in emails or text messages while reducing the number of false positives and false negatives.