PROJECT TITLE: Building a Smarter AI-Powered Spam Classifer.

INNOVATION STEPS:

As a beginner, I built an SMS spam classifier but did a ton of research to know where to start.

1. Load and simplify the dataset.

Our SMS text messages dataset has 5 columns if you read it in pandas: v1 (containing the class labels ham/spam for each text message), v2 (containing the text messages themselves), and three Unnamed columns which have no use.

2. Explore the dataset: Bar Chart.

It's a good idea to carry out some Exploratory Data Analysis (EDA) in a classification problem to visualize, get some information out of, or find any issues with your data before you start working with it.

3. Explore the dataset: Word Clouds.

For my project, I generated word clouds of the most frequently occurring words in my spam messages.First, we'll filter out all the spam messages from our dataset. df_spam is a DataFrame that contains only spam messages.

4.Handle imbalanced datasets.

To handle imbalanced data, you have a variety of options. I got a pretty good f-measure in my project even with unsampled data, but if you want to resample.

5.Split the dataset.

In Machine Learning, we usually split our data into two subsets — train and test. We feed the train set along with the known output values for it (in this case, 0 or 1 corresponding to spam or ham) to our model so that it learns the patterns in our data. Then we use the test set to get the model’s predicted labels on this subset. Let’s see how to split our data.

6.Apply Tf-IDF Vectorizer for feature extraction.

Our Naïve Bayes model requires data to be in either Tf-IDF vectors or word vector count. The latter is achieved using Count Vectorizer, but we’ll obtain the former through using Tf-IDF Vectorizer.

7.Train our Naive Bayes Model.

We fit our Naïve Bayes model, aka MultinomialNB, to our Tf-IDF vector version of x_train, and the true output labels stored in y_train.

8.Check out the accuracy, and f-measure.

It’s time to pass in our Tf-IDF matrix corresponding to x_test, along with the true output labels (y_test), to find out how well our model did!

9.View the confusion matrix and classification report.

Let’s now look at our confusion matrix and f-measure scores to *confirm* if our model is doing OK or not:

10.Heatmap for our Confusion Matrix (Optional).

You can create a heatmap using the seaborn library to visualize your confusion matrix.