## BUILDING A SMARTER AI-POWERED SPAM CLASSIFIER

### Phase-1 Document submission

Project:**Building a smarter AI-powered spam classifier**

**Problem Definition:**

The problem is to build an AI-powered spam classifier that can accurately distinguish between spam non-spam messages in emails or text messages. The goal is to reduce the number of false positives (classifying legitimate messages as spam) and false negatives (missing actual spam messages) while achieving a high level of accuracy.

***BUILDING A SMARTER AI-POWERED SPAM CLASSIFIER INVOLVES A SEVERAL STEPS :***

Building a smarter AI-powered spam classifier involves several steps, from data preprocessing to model deployment. Below is a comprehensive guide to building such a classifier using Python and popular machine learning libraries like scikit-learn:

***1. Data Collection and Preprocessing:***

- Collect a dataset of spam and non-spam (ham) messages. This dataset should be well-labeled.

- Preprocess the data by removing HTML tags, special characters, and unnecessary whitespace. You can also perform text normalization, tokenization, and stemming or lemmatization.

***2. Feature Extraction:***

- Use TF-IDF (Term Frequency-Inverse Document Frequency) or word embeddings (e.g., Word2Vec, GloVe) to convert text data into numerical features.

- You can also consider character-level features or other advanced text representations.

***3. Data Splitting:***

- Split your dataset into training, validation, and test sets. A common split is 70% for training, 15% for validation, and 15% for testing.

***4. Model Selection:***

- Choose a machine learning algorithm or model. Common choices include:

- Naive Bayes

- Support Vector Machines (SVM)

- Random Forest

- Gradient Boosting

- Neural Networks (e.g., deep learning models like LSTM or BERT)

***5. Model Training:***

- Train your chosen model on the training data.

- Experiment with different hyperparameters to optimize the model's performance.

- Use techniques like k-fold cross-validation to assess the model's robustness.

***6. Model Evaluation:***

- Evaluate the model on the validation set using appropriate metrics like accuracy, precision, recall, F1-score, and ROC-AUC.

***7. Hyperparameter Tuning:***

- Fine-tune the model's hyperparameters using methods like grid search or random search.

***8. Model Ensemble (Optional):***

- Combine multiple models (e.g., ensembling Decision Trees or using stacking techniques) to improve classification accuracy.

***9. Handling Imbalanced Data:***

- If your dataset is imbalanced (more hams than spams or vice versa), consider oversampling, undersampling, or using Synthetic Minority Over-sampling Technique (SMOTE) to balance the classes.

***10. Feature Engineering (Optional):***

- Experiment with adding custom features or engineered features that might help the model better distinguish spam from ham.

***11. Model Interpretability (Optional):***

- Use techniques like SHAP values or LIME to understand and interpret the model's predictions, which can be important for trust and explainability.

***12. Model Deployment:***

- Deploy the trained model into a production environment where it can classify incoming messages in real-time.

***13. Continuous Monitoring and Updating:***

- Continuously monitor the model's performance in production.

- Retrain the model periodically with new data to adapt to evolving spam tactics.

***14. Privacy and Compliance:***

- Ensure that your spam classifier complies with privacy regulations and respects user privacy.

***15. User Feedback:***

- Implement mechanisms for users to report false positives/negatives and provide feedback for model improvement.

***16. Scalability:***

- Design your solution to scale efficiently to handle a large volume of messages in real-time.

## PROGRAM:

**INPUT:**

import numpy as np

import pandas as pd

import osfor dirname, \_, filenames **in** os.walk('/kaggle/input'):

for filename **in** filenames:

print(os.path.join(dirname, filename))

**OUTPUT:**

/kaggle/input/sms-spam-collection-dataset/spam.csv

**INPUT:**

import pandas as pd

df = pd.read\_csv("/kaggle/input/sms-spam-collection-dataset/spam.csv", sep='**\t**', encoding='ISO-8859-1')

df[['label', 'message']] = df['v1,v2,,,'].str.split(',', n=1, expand=True)

df.drop('v1,v2,,,', axis=1, inplace=True)

df.head(10)

**OUTPUT:**

|  | label | 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 ar... |
| 5 | spam | "FreeMsg Hey there darling it's been 3 week's ... |
| 6 | ham | Even my brother is not like to speak with me. ... |
| 7 | ham | As per your request 'Melle Melle (Oru Minnamin... |
| 8 | spam | WINNER!! As a valued network customer you have... |
| 9 | spam | Had your mobile 11 months or more? U R entitle... |

***INPUT:***

from sklearn.feature\_extraction.text

import TfidfVectorizer

print('TfidfVectorizer:')

print('modelo SVM:')classifier\_svm = Pipeline([

('tfidf', TfidfVectorizer(sublinear\_tf=True, binary=True, smooth\_idf=False)),

('clf', SVC(kernel='linear', probability=True))])classifier\_svm.fit(X\_train, y\_train)y\_pred\_svm = classifier\_svm.predict(X\_test)evaluetePerformance(y\_test,y\_pred\_svm)

print('modelo MultinomialNB:')classifier\_bayes = Pipeline([

('tfidf', TfidfVectorizer(sublinear\_tf=True, binary=True, smooth\_idf=False)),

('clf', MultinomialNB())])classifier\_bayes.fit(X\_train, y\_train)

print('modelo Random Forest:')classifier\_forest = Pipeline([

('count\_vectorizer', CountVectorizer()),

('clf', RandomForestClassifier(n\_estimators=100, random\_state=42))])

classifier\_forest.fit(X\_train, y\_train)y\_pred\_forest = classifier\_forest.predict(X\_test)evaluetePerformance(y\_test,y\_pred\_forest)

print('modelo SGDClassifier:')classifier\_sgd = Pipeline([

('count\_vectorizer', CountVectorizer()),

('clf', SGDClassifier(loss='hinge', penalty='l2', random\_state=13))])

classifier\_sgd.fit(X\_train, y\_train)y\_pred\_sgd = classifier\_sgd.predict(X\_test)evaluetePerformance(y\_test, y\_pred\_sgd)

**OUTPUT:**

modelo SVM:

Accuracy 0.9787234042553191

Recall 0.8384615384615385

Precision 0.990909090909091

modelo MultinomialNB:

Accuracy 0.9864603481624759

Recall 0.9384615384615385

Precision 0.953125

modelo Random Forest:

Accuracy 0.9661508704061895

Recall 0.7307692307692307

Precision 1.0

modelo SGDClassifier:

Accuracy 0.9748549323017408

Recall 0.8153846153846154

Precision 0.9814814814814815

**INPUT:**

from sklearn.preprocessing import LabelEncoderencoder=LabelEncoder()df['target']=encoder.fit\_transform(df['target'])

df.head()

**OUTPUT:**

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

**INPUT:**

df.isnull().sum()

df.isnull().sum()

**OUTPUT:**

target 0

text 0

Unnamed: 2 5522

Unnamed: 3 5560

Unnamed: 4 5566

dtype: int64

**INPUT:**

df.duplicated().sum()

OUTPUT

403

**INPUT**

df.duplicated().sum()

df.shape

**OUTPUT:**

(5169, 5)

**INPUT:**

df['target'].value\_counts()

**OUTPUT**

target

0 4516

1 653

Name: count, dtype: int64

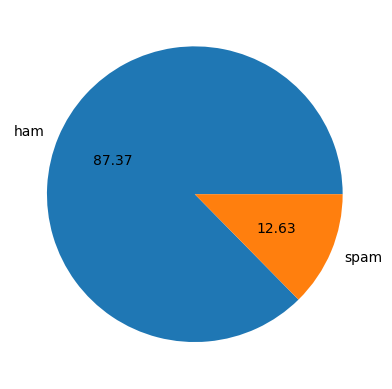
***INPUT***

import matplotlib.pyplot as pltplt.pie(df['target'].value\_counts(),

labels=['ham','spam'],autopct='**%0.2f**')

plt.show()

**OUTPUT**



This is a comprehensive overview of the steps involved in building a smarter AI-powered spam classifier. Depending on your specific needs and constraints, you may need to adjust and fine-tune each step. Additionally, you can experiment with more advanced techniques such as deep learning and natural language processing to further improve classifier performance.

## **\*\*\*THANKING YOU\*\*\*\***