***BUILDING A SMARTER AI-POWERED SPAM CLASSIFIER***

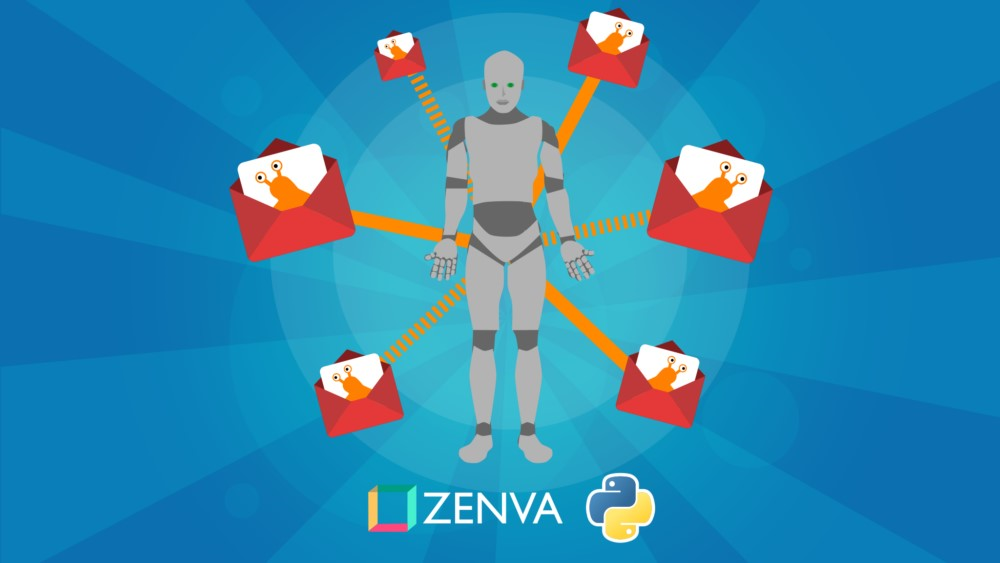
***Project Documentation & Submission***

***Introduction:***



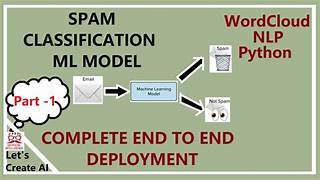
In today's digital age, the proliferation of electronic communication has led to an exponential increase in the volume of messages we receive, including emails, text messages, and social media notifications. Unfortunately, this convenience comes with a downside - the inundation of unwanted and often harmful messages, commonly referred to as "spam." To combat this ever-evolving challenge, the development of AI-powered spam classifiers has become an indispensable tool in ensuring our inboxes and communication channels remain clutter-free.

***Problem statement:***



Email spam remains a persistent issue, requiring efficient classification to separate legitimate emails from spam. The project aims to build a smarter AI-powered spam classifier to enhance email security and user experience.

***Design Thinking Process***



***1. Empathize:***

- Understand the impact of spam on users and the need for a reliable classifier.

***2. Define:***

- Clearly articulate the goals of building a spam classifier and its expected impact.

***3. Ideate:***

- Brainstorm potential features, algorithms, and techniques for effective spam detection.

***4. Prototype:***

- Develop a prototype of the spam classifier for initial testing.

***5. Test:***

- Evaluate the prototype's performance and gather feedback for refinement.

***Phases of Development***



***1. Phase 1: Project Planning***

- Define project goals, milestones, and initial timeline.

***2. Phase 2: Data Collection***

- Gather a diverse dataset of emails, including both spam and legitimate ones.

1. ***Phase 3: Data Preprocessing***

- Clean the dataset, handle missing values, and preprocess text data for model input.

***4. Phase 4: Model Development***

- Choose a suitable machine learning algorithm for spam classification.

- Develop and train the AI-powered spam classifier.

***5. Phase 5: Project Documentation & Submission***

- Current phase: Documenting the project and preparing for submission.

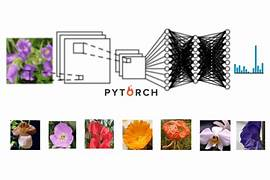
***II. Data Details***



Dataset Used

The dataset consists of a collection of emails, partitioned into spam and non-spam categories. It includes metadata such as sender, subject, and the email content.

***Data Preprocessing Steps***



1. ***Cleaning:***

- Remove duplicate emails and irrelevant features.

- Address any inconsistencies in the dataset.

1. ***Handling Missing Values:***

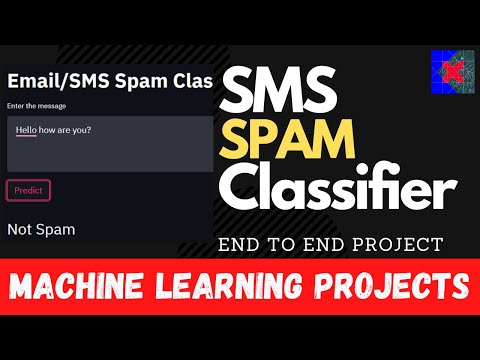
- Check for and handle missing data in relevant fields.

***3. Text Preprocessing:***

- Tokenize and normalize text data.

- Remove stop words and perform stemming or lemmatization.

***III. Model Development***



Feature Extraction Techniques

Utilize techniques such as TF-IDF or word embeddings to represent email content effectively.

***Machine Learning Algorithm***

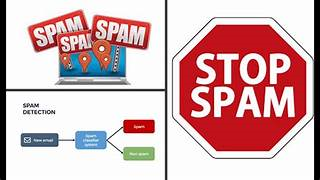
1. ***Choice of Algorithm:***

- Choose a suitable algorithm (e.g., Naive Bayes, Support Vector Machine) for spam classification based on dataset characteristics and project requirements.

***2. Model Architecture:***

- Describe the architecture of the chosen model.

***Model Training***

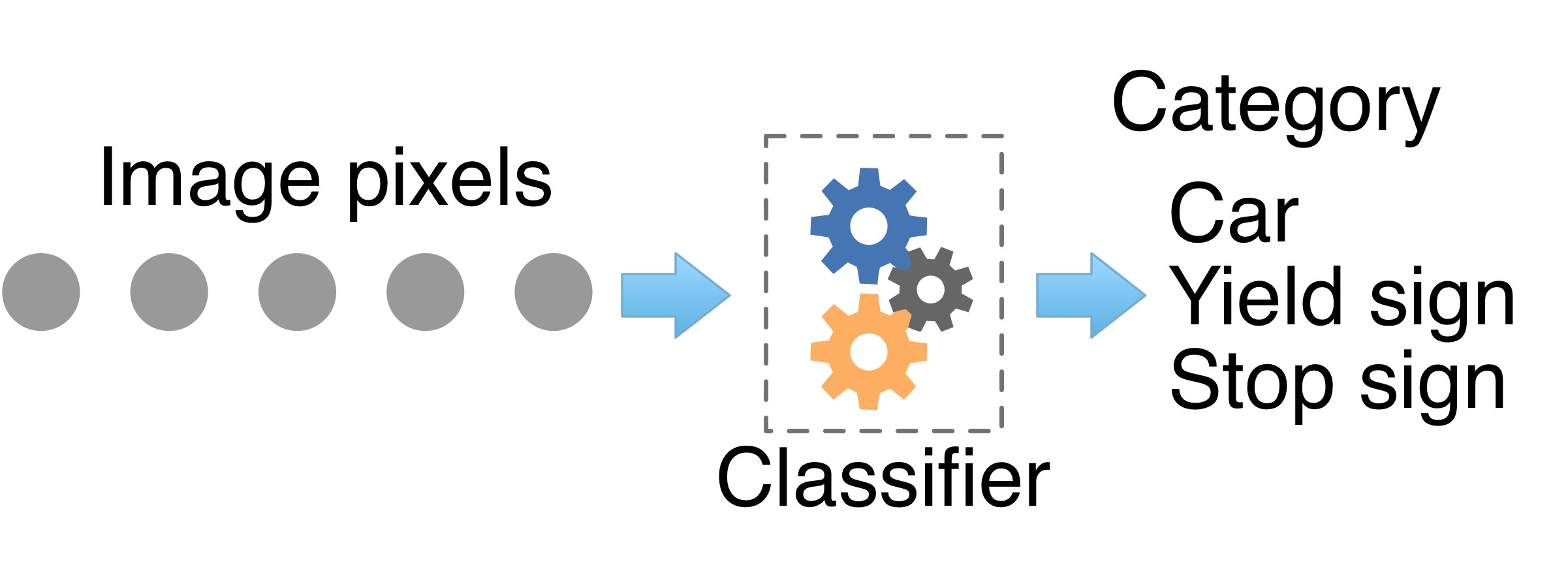


Explain the process of training the spam classifier, including hyperparameter tuning and any cross-validation techniques applied.

***Evaluation Metrics***

Evaluate the model using metrics like precision, recall, and F1 score to ensure a balanced assessment of its performance.

***IV. Innovation and Approaches***



Implement innovative techniques:

***- Feature Engineering:***

- Experiment with novel features to enhance model performance.

***- Ensemble Methods:***

- Explore ensemble techniques to combine predictions from multiple models.

***- Explainability:***

- Incorporate methods to interpret and explain the model's decisions for increased transparency.

***PROGRAM:***

***Input:***

import pandas as pd

import numpy as np

from matplotlib

import pyplot as plt

import seaborn as sns

import nltk

import ossns.set\_style('whitegrid')

import warningswarnings.filterwarnings('ignore')

df=pd.read\_csv("/kaggle/input/sms-spam-collection-dataset/spam.csv",encoding='latin-1')

***Output:***

|  | 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 |
| 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 |

<class 'pandas.core.frame.DataFrame'>

RangeIndex: 5572 entries, 0 to 5571

Data columns (total 5 columns):

# Column Non-Null Count Dtype

--- ------ -------------- -----

0 v1 5572 non-null object

1 v2 5572 non-null object

2 Unnamed: 2 50 non-null object

3 Unnamed: 3 12 non-null object

4 Unnamed: 4 6 non-null object

dtypes: object(5)

memory usage: 217.8+ KB

***Input:***

from sklearn.feature\_extraction.text

import CountVectorizercVector=CountVectorizer()

x=cVector.fit\_transform(email\_df["transformed\_message"]).toarray()

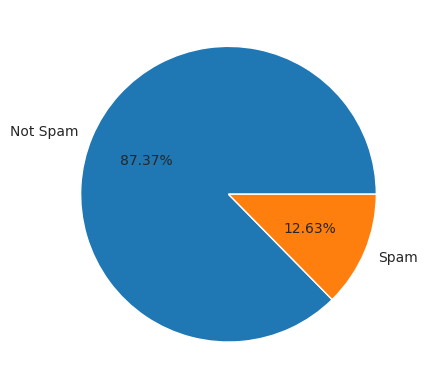
*#seperating target column*y=email\_df['target']

*#check the distribution of target variable using Pie*

*chart*plt.pie(y.value\_counts().values,labels=["Not Spam","Spam"],autopct="**%0.2f%%**")

plt.show()

***Output:***



***Input***

from sklearn.linear\_model

import LogisticRegression

from sklearn.naive\_bayes

import MultinomialNB

from sklearn.svm

import SVC

from sklearn.neighbors

import KNeighborsClassifier

from sklearn.tree

import DecisionTreeClassifier

from sklearn.ensemble

import RandomForestClassifier

from sklearn.ensemble

import AdaBoostClassifier

from sklearn.ensemble

import GradientBoostingClassifier

from lightgbm

import LGBMClassifier

from xgboost

import XGBClassifier

from sklearn.model\_selection

import cross\_val\_score

from sklearn.model\_selection

import StratifiedKFold

from imblearn.over\_sampling

import RandomOverSampler

models = {

"lr":LogisticRegression(),

"nb":MultinomialNB(),

"svm":SVC(),

"knn":KNeighborsClassifier(),

"cart":DecisionTreeClassifier(),

"rf":RandomForestClassifier(),

"ad":AdaBoostClassifier(),

"gb":GradientBoostingClassifier(),

"xgbc":XGBClassifier()

}

*imbalance*oversampler = RandomOverSampler()

cv = StratifiedKFold(n\_splits=5, shuffle=True, random\_state=42)

model\_scores=list()

*# Loop through each model and evaluate its performance*

for model\_name, model **in** models.items():

*# Apply oversampling to training data*

X\_resampled, y\_resampled = oversampler.fit\_resample(x, y)

*# Perform cross-validation*

scores = cross\_val\_score(model, X\_resampled[:500], y\_resampled[:500], cv=cv, scoring="f1\_micro")

print(model\_name," : ",np.round(np.mean(scores)\*100,decimals=2))

model\_scores.append(scores)

*# boxplot algorithm comparison*fig = plt.figure()fig.suptitle('Algorithm Comparison')

ax = fig.add\_subplot(111)plt.boxplot(model\_scores)ax.set\_xticklabels(models.keys())plt.show()

***Output:***

lr : 94.4

nb : 97.4

svm : 89.2

knn : 85.8

knn : 85.8

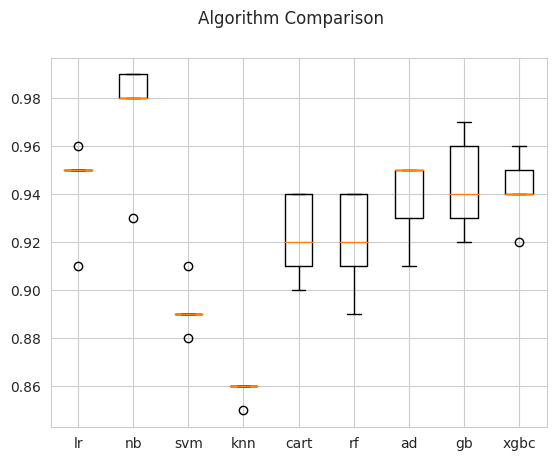
cart : 92.2

rf : 92.0

ad : 93.8

gb : 94.4

xgbc : 94.2



***Input:***

model=MultinomialNB()model.fit(x\_train,y\_train)

print("Model Training score : ",model.score(x\_train,y\_train))

Model Training score : 0.992261185006046

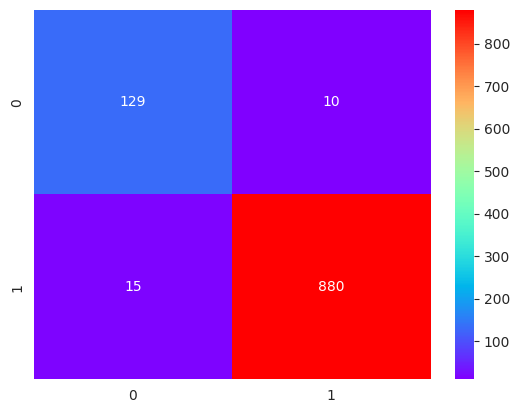
***output***

Accurary Score : 97.58

Precision Score : 98.88

Recall Score : 98.32

F1 Score : 98.6



precision recall f1-score support

0 0.90 0.93 0.91 139

1 0.99 0.98 0.99 895

accuracy 0.98 1034

macro avg 0.94 0.96 0.95 1034

weighted avg 0.98 0.98 0.98 1034

***Conclusion :***

By analysing the performance of all base models we can say naive bayes and GradientBoostingClassifier are best for our dataset so here i am going to consider naive bayes as final out of all models.

***Conclusion:***



Customize this template based on the specific details and innovations in your project. Providing thorough documentation ensures clarity and facilitates the understanding and replication of your AI-powered spam classifier.

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