Email Spam

About Dataset

The SMS Spam Collection is a set of SMS tagged messages that have been collected for SMS Spam research. It contains one set of SMS messages in English of 5,574 messages, tagged acording being ham (legitimate) or spam.

Objective

The objective of this project is to develop a machine learning model capable of accurately classifying SMS messages as either "spam" or "ham" (legitimate). By leveraging state-of-the-art natural language processing techniques and machine learning algorithms, our aim is to create a robust and effective SMS spam detection system that enhances user communication experiences by filtering out unwanted and potentially harmful messages.

The project will involve data preprocessing, feature extraction, model training, and rigorous evaluation to achieve a high level of accuracy in identifying spam messages while minimizing false positives and false negatives.

Approach

1. Load the data and load all the libraries
2. Data Preparation and Data transformation
   1. Convert all text into LowerCase
   2. Remove all special characters
   3. Remove stop words
   4. Lemmatization and Stemming
3. Vectorization
   1. TFIDF Vectorizer
4. Machine Learning and also Deep Learning

Load the Data and The Libraries

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
from sklearn.preprocessing import LabelEncoder
from sklearn.model_selection import train_test_split
from sklearn.naive_bayes import GaussianNB
from sklearn.metrics import accuracy_score,classification_report,confusion_matrix
import seaborn as sns
```

```
df=pd.read_csv('/content/spam.csv',encoding='latin-1')
df.head()
```

	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

Next steps: Generate code with df View recommended plots New interactive sheet

```
df.shape
```

```
(5572, 5)
```

EDA

```
df.info()
```

```
<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
```

```
df['Unnamed: 2'].value_counts()
```

	Unnamed: 2
bt not his girlfrnd... G o o d n i g h t . . .@"	3
PO Box 5249	2
this wont even start........ Datz confidence.."	2
GN	2
don't miss ur best life for anything... Gud nyt..."	2
but dont try to prove it..\" .Gud noon...."	2
Gud night...."	1
like you are the KING\"...! OR \"Walk like you Dont care	1
HAD A COOL NYTHO	1
PO Box 1146 MK45 2WT (2/3)"	1
\"It is d wonderful fruit that a tree gives when it is being hurt by a stone.. Good night......"	1
we made you hold all the weed\""	1
its a miracle to Love a person who can't Love anyone except U...\" Gud nyt..."	1
hopeSo hunny. i amnow feelin ill & ithink i may have tonsolitusaswell! damn iam layin in bedreal bored. lotsof luv me xxxx\""	1
that's the tiny street where the parking lot is"	1
PROBPOP IN & CU SATTHEN HUNNY 4BREKKIE! LOVE JEN XXX. PSXTRA LRG PORTIONS 4 ME PLEASE \""	1
SHE SHUDVETOLD U. DID URGRAN KNOW?NEWAY	1
GOD said	1
always give response 2 who cares 4 U\"... Gud night..swt dreams..take care"	1
HOPE UR OK... WILL GIVE U A BUZ WEDLUNCH. GO OUTSOMEWHERE 4 ADRINK IN TOWN..CUD GO 2WATERSHD 4 A BIT? PPL FROMWRK WILL BTHERE. LOVE PETEXXX.\""	1
b'coz nobody will fight for u. Only u &amp; u have to fight for ur self &amp; win the battle. -VIVEKANAND- G 9t.. SD.."	1
DEVIOUSBITCH.ANYWAY	1
but watever u shared should be true\"...."	1
Dont Come Near My Body..!! Bcoz My Hands May Not Come 2 Wipe Ur Tears Off That Time..!Gud ni8"	1
but dont try to prove\" ..... Gud mrng..."	1
the toughest is acting Happy with all unspoken pain inside..\""	1
HOWU DOIN? FOUNDURSELF A JOBYET SAUSAGE?LOVE JEN XXX\""	1
wanted to say hi. HI!!!\" Stop? Send STOP to 62468"	1
.;-):-D"	1
just been in bedbut mite go 2 thepub l8tr if uwana mt up?loads a luv Jenxxx.\""	1
I'll come up"	1
just as a shop has to give a guarantee on what they sell. B. G."	1
But at d end my love compromised me for everything:-(\".. Gud mornin:-)"	1
smoke hella weed\""	1
Well there's still a bit left if you guys want to tonight	1
\" not \"what i need to do.\""	1
JUST GOT PAYED2DAY & I HAVBEEN GIVEN Aå£50 PAY RISE 4MY WORK & HAVEBEEN MADE PRESCHOOLCO-ORDINATOR 2I AM FEELINGOOD LUV\""	1
justthought iåÕd sayhey! how u doin?nearly the endof me wk offdam nevamind!We will have 2Hook up sn if uwant m8? loveJen x.\""	1
JUST REALLYNEED 2DOCD.PLEASE DONTPLEASE DONTIGNORE MYCALLS	1
u hav2hear it!c u sn xxxx\""	1
I don't mind	1
the person is definitely special for u..... But if the person is so special	1
ENJOYIN INDIANS AT THE MO..yeP. SaLL gOoD HehE ;> hows bout u shexy? Pete Xx\""	1

```
df['Unnamed: 3'].value_counts()
```

Unnamed: 3	count
MK17 92H. 450Ppw 16"	2
GE	2
why to miss them	1
U NO THECD ISV.IMPORTANT TOME 4 2MORO\""	1
i wil tolerat.bcs ur my someone..... But	1
ILLSPEAK 2 U2MORO WEN IM NOT ASLEEP...\""	1
whoever is the KING\"!... Gud nyt"	1
TX 4 FONIN HON	1
\"OH No! COMPETITION\". Who knew	1
IåÕL CALL U\""	1

dtype: int64

Delete unwanted columns

```
df.drop(['Unnamed: 2','Unnamed: 3','Unnamed: 4'],axis=1,inplace=True)
```

Checking for null values

```
df.isnull().sum()
```

	0
v1	0
v2	0

```
df.describe()
```

	v1	v2
count	5572	5572
unique	2	5169
top	ham	Sorry, I'll call later
freq	4825	30

```
df['v1'].value_counts()
```

v1	count
ham	4825
spam	747

1. Convert all the text into Lowercase

2. Remove all the special characters

3. Remove all the stop words

4. Lemmatization and Stemming

```
import nltk
from nltk.tokenize import RegexpTokenizer
from nltk.stem import WordNetLemmatizer,PorterStemmer
from nltk.corpus import stopwords
import re


nltk.download('stopwords')
lemmatizer = WordNetLemmatizer()
stemmer = PorterStemmer()
nltk.download('wordnet')
def preprocess(sentence):
    sentence=str(sentence)
    sentence = sentence.lower()
    sentence=sentence.replace('{html}',"")
    cleanr = re.compile('<.*?>')
    cleantext = re.sub(cleanr, '', sentence)
    rem_url=re.sub(r'http\S+', '',cleantext)
    rem_num = re.sub('[0-9]+', '', rem_url)
    tokenizer = RegexpTokenizer(r'\w+')
    tokens = tokenizer.tokenize(rem_num)
    filtered_words = [w for w in tokens if len(w) > 2 if not w in stopwords.words('english')]
    stem_words=[stemmer.stem(w) for w in filtered_words]
    lemma_words=[lemmatizer.lemmatize(w) for w in stem_words]
    return " ".join(filtered_words)
```

```
[nltk_data] Downloading package stopwords to /root/nltk_data...
[nltk_data]   Unzipping corpora/stopwords.zip.
[nltk_data] Downloading package wordnet to /root/nltk_data...
```

```
df['v2']=df['v2'].map(lambda s:preprocess(s))
```

```
df.head()
```

	v1	v2
0	ham	jurong point crazy available bugis great world...
1	ham	lar joking wif oni
2	spam	free entry wkly comp win cup final tkts may te...
3	ham	dun say early hor already say
4	ham	nah think goes usf lives around though

Next steps: Generate code with df View recommended plots New interactive sheet

Vectorization -TFIDF vector

```
from sklearn.feature_extraction.text import TfidfVectorizer
```

```
vect=TfidfVectorizer()
```

```
x=vect.fit_transform(df['v2'])
x.shape
```

```
(5572, 7386)
```

Data Encoding and Data Splitting

```
le=LabelEncoder()
```

```
y=le.fit_transform(df['v1'])
```

```
x_train, x_test, y_train, y_test = train_test_split(x,y, test_size = 0.2, random_state = 123)
```

```
print(x_train.shape)
print(x_test.shape)
print(y_train.shape)
print(y_test.shape)
```

```
(4457, 7386)
(1115, 7386)
(4457,)
(1115,)
```

```
x.data
```

```
array([0.36750082, 0.28744258, 0.28460409, ..., 0.48395639, 0.53118971,
       0.69543059])
```

Random Forest

```
from sklearn.model_selection import RandomizedSearchCV
from sklearn.ensemble import RandomForestClassifier
```

```
random_grid = {'criterion': ['gini', 'entropy', 'log_loss'],
               'max_depth': [10, 20, 30, 40, 50, 60, 70, 80, 90, 100, 110],

               'min_samples_leaf': [1, 2, 4],
               'min_samples_split': [2, 5, 10],
               'n_estimators': [130, 180, 230]}
```

```
rf=RandomForestClassifier()
clf=RandomizedSearchCV(estimator=rf ,param_distributions=random_grid,verbose=2,random_state=142)
```

```
search=clf.fit(x_train,y_train)
```

```
Fitting 5 folds for each of 10 candidates, totalling 50 fits
[CV] END criterion=gini, max_depth=50, min_samples_leaf=2, min_samples_split=2, n_estimators=130; total time=   2.3s
[CV] END criterion=gini, max_depth=50, min_samples_leaf=2, min_samples_split=2, n_estimators=130; total time=   2.3s
[CV] END criterion=gini, max_depth=50, min_samples_leaf=2, min_samples_split=2, n_estimators=130; total time=   2.2s
[CV] END criterion=gini, max_depth=50, min_samples_leaf=2, min_samples_split=2, n_estimators=130; total time=   2.2s
[CV] END criterion=gini, max_depth=50, min_samples_leaf=2, min_samples_split=2, n_estimators=130; total time=   3.4s
[CV] END criterion=entropy, max_depth=10, min_samples_leaf=1, min_samples_split=5, n_estimators=180; total time=   1.1s
[CV] END criterion=entropy, max_depth=10, min_samples_leaf=1, min_samples_split=5, n_estimators=180; total time=   0.9s
[CV] END criterion=entropy, max_depth=10, min_samples_leaf=1, min_samples_split=5, n_estimators=180; total time=   1.0s
[CV] END criterion=entropy, max_depth=10, min_samples_leaf=1, min_samples_split=5, n_estimators=180; total time=   1.5s
[CV] END criterion=entropy, max_depth=10, min_samples_leaf=1, min_samples_split=5, n_estimators=180; total time=   2.0s
[CV] END criterion=gini, max_depth=110, min_samples_leaf=2, min_samples_split=5, n_estimators=230; total time=   7.6s
[CV] END criterion=gini, max_depth=110, min_samples_leaf=2, min_samples_split=5, n_estimators=230; total time=   6.4s
[CV] END criterion=gini, max_depth=110, min_samples_leaf=2, min_samples_split=5, n_estimators=230; total time=   7.6s
[CV] END criterion=gini, max_depth=110, min_samples_leaf=2, min_samples_split=5, n_estimators=230; total time=   6.4s
[CV] END criterion=gini, max_depth=110, min_samples_leaf=2, min_samples_split=5, n_estimators=230; total time=   7.7s
[CV] END criterion=entropy, max_depth=70, min_samples_leaf=2, min_samples_split=10, n_estimators=230; total time=   4.8s
[CV] END criterion=entropy, max_depth=70, min_samples_leaf=2, min_samples_split=10, n_estimators=230; total time=   5.5s
[CV] END criterion=entropy, max_depth=70, min_samples_leaf=2, min_samples_split=10, n_estimators=230; total time=   5.5s
[CV] END criterion=entropy, max_depth=70, min_samples_leaf=2, min_samples_split=10, n_estimators=230; total time=   5.0s
[CV] END criterion=entropy, max_depth=70, min_samples_leaf=2, min_samples_split=10, n_estimators=230; total time=   6.0s
[CV] END criterion=gini, max_depth=50, min_samples_leaf=1, min_samples_split=2, n_estimators=130; total time=   2.6s
[CV] END criterion=gini, max_depth=50, min_samples_leaf=1, min_samples_split=2, n_estimators=130; total time=   2.6s
[CV] END criterion=gini, max_depth=50, min_samples_leaf=1, min_samples_split=2, n_estimators=130; total time=   2.6s
[CV] END criterion=gini, max_depth=50, min_samples_leaf=1, min_samples_split=2, n_estimators=130; total time=   3.9s
[CV] END criterion=gini, max_depth=50, min_samples_leaf=1, min_samples_split=2, n_estimators=130; total time=   2.7s
[CV] END criterion=gini, max_depth=30, min_samples_leaf=4, min_samples_split=5, n_estimators=230; total time=   2.5s
[CV] END criterion=gini, max_depth=30, min_samples_leaf=4, min_samples_split=5, n_estimators=230; total time=   2.5s
```

```
[CV] END criterion=gini, max_depth=30, min_samples_leaf=4, min_samples_split=5, n_estimators=230; total time=   2.7s
[CV] END criterion=gini, max_depth=30, min_samples_leaf=4, min_samples_split=5, n_estimators=230; total time=   3.5s
[CV] END criterion=gini, max_depth=30, min_samples_leaf=4, min_samples_split=5, n_estimators=230; total time=   2.5s
[CV] END criterion=log_loss, max_depth=20, min_samples_leaf=4, min_samples_split=10, n_estimators=230; total time=   1.8s
[CV] END criterion=log_loss, max_depth=20, min_samples_leaf=4, min_samples_split=10, n_estimators=230; total time=   1.8s
[CV] END criterion=log_loss, max_depth=20, min_samples_leaf=4, min_samples_split=10, n_estimators=230; total time=   1.8s
[CV] END criterion=log_loss, max_depth=20, min_samples_leaf=4, min_samples_split=10, n_estimators=230; total time=   2.2s
[CV] END criterion=log_loss, max_depth=20, min_samples_leaf=4, min_samples_split=10, n_estimators=230; total time=   2.8s
[CV] END criterion=gini, max_depth=80, min_samples_leaf=1, min_samples_split=2, n_estimators=130; total time=   3.8s
[CV] END criterion=gini, max_depth=80, min_samples_leaf=1, min_samples_split=2, n_estimators=130; total time=   3.8s
[CV] END criterion=gini, max_depth=80, min_samples_leaf=1, min_samples_split=2, n_estimators=130; total time=   4.6s
[CV] END criterion=gini, max_depth=80, min_samples_leaf=1, min_samples_split=2, n_estimators=130; total time=   4.1s
[CV] END criterion=gini, max_depth=80, min_samples_leaf=1, min_samples_split=2, n_estimators=130; total time=   3.8s
[CV] END criterion=entropy, max_depth=70, min_samples_leaf=2, min_samples_split=2, n_estimators=180; total time=   4.3s
[CV] END criterion=entropy, max_depth=70, min_samples_leaf=2, min_samples_split=2, n_estimators=180; total time=   4.6s
[CV] END criterion=entropy, max_depth=70, min_samples_leaf=2, min_samples_split=2, n_estimators=180; total time=   3.8s
[CV] END criterion=entropy, max_depth=70, min_samples_leaf=2, min_samples_split=2, n_estimators=180; total time=   3.9s
[CV] END criterion=entropy, max_depth=70, min_samples_leaf=2, min_samples_split=2, n_estimators=180; total time=   5.1s
[CV] END criterion=log_loss, max_depth=90, min_samples_leaf=4, min_samples_split=5, n_estimators=180; total time=   3.7s
[CV] END criterion=log_loss, max_depth=90, min_samples_leaf=4, min_samples_split=5, n_estimators=180; total time=   3.8s
[CV] END criterion=log_loss, max_depth=90, min_samples_leaf=4, min_samples_split=5, n_estimators=180; total time=   4.9s
[CV] END criterion=log_loss, max_depth=90, min_samples_leaf=4, min_samples_split=5, n_estimators=180; total time=   3.7s
[CV] END criterion=log_loss, max_depth=90, min_samples_leaf=4, min_samples_split=5, n_estimators=180; total time=   3.8s
```

```
search.best_params_
```

```
{'n_estimators': 130,
 'min_samples_split': 2,
 'min_samples_leaf': 1,
 'max_depth': 80,
 'criterion': 'gini'}
```

```
search.best_score_
```

```
0.9744200852571556
```

```
rf = RandomForestClassifier(n_estimators=130,
 min_samples_split=10,
 min_samples_leaf = 1,
 max_depth= None,
criterion= 'gini')
```

```
rf.fit(x_train.toarray(),y_train)
rf_preds_train = rf.predict(x_train.toarray())
rf_preds_test = rf.predict(x_test.toarray())
```

```
print('Accuracy score for train data : ', round(accuracy_score(y_train, rf_preds_train),2))
print('Accuracy score for test data : ', round(accuracy_score(y_test, rf_preds_test),2))
print('Confusion matrix of the model is: \n', confusion_matrix(y_test, rf_preds_test))
print(f'Classification report of the model is: \n', classification_report(y_test, rf_preds_test))
```

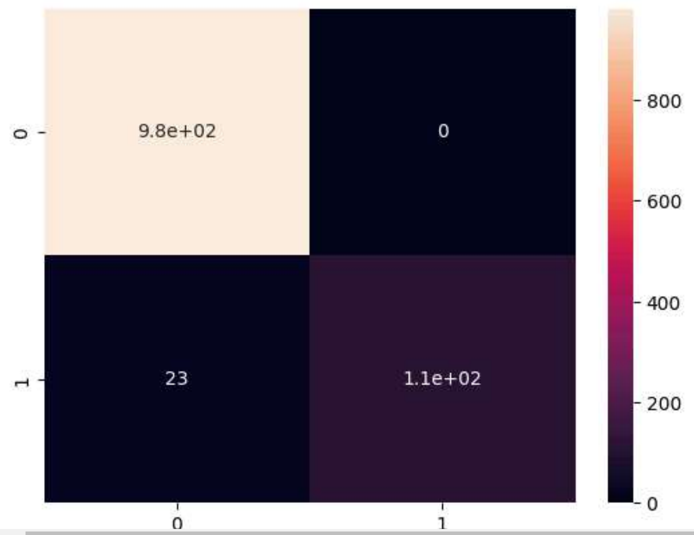
```
Accuracy score for train data :  1.0
Accuracy score for test data :  0.98
Confusion matrix of the model is:
 [[982   0]
 [ 23 110]]
Classification report of the model is:
               precision    recall  f1-score   support

           0       0.98      1.00      0.99       982
           1       1.00      0.83      0.91       133

    accuracy                           0.98      1115
   macro avg       0.99      0.91      0.95      1115
weighted avg       0.98      0.98      0.98      1115
```

```
sns.heatmap(confusion_matrix(y_test, rf_preds_test), annot=True)
```

```
<Axes: >
```



MultinomialNB

```
nb=GaussianNB()
nb.fit(x_train.toarray(),y_train)
nb_preds_train=nb.predict(x_train.toarray())
nb_preds_test=nb.predict(x_test.toarray())
```

```
print('Accuracy score of the model is: ', round(accuracy_score(y_train, nb_preds_train),2))
print('Accuracy score of the model is: ', round(accuracy_score(y_test, nb_preds_test),2))
print('Confusion matrix of the model is: ', confusion_matrix(y_test, nb_preds_test))
print('Classification report of the model is: ', classification_report(y_test, nb_preds_test))
```

```
Accuracy score of the model is:  0.93
Accuracy score of the model is:  0.88
```

Support Vector Machine

```
from sklearn.svm import SVC

svc = SVC()
```

```
from scipy.stats import reciprocal, randint
param_dist = {
    'C': reciprocal(0.1, 10),  # Regularization parameter
    'kernel': ['linear', 'poly', 'rbf', 'sigmoid'],  # Kernel type
    'gamma': ['scale', 'auto'] + list(reciprocal(0.01, 0.1).rvs(size=3)),  # Kernel coefficient for 'poly', 'rbf', 'sigmoid'
    'degree': randint(2, 5),  # Degree of the polynomial kernel function
    'coef0': reciprocal(0.1, 10)  # Independent term in kernel function
}
```

```
random_search = RandomizedSearchCV(svc, param_distributions=param_dist, n_iter=100, cv=5, scoring='accuracy', n_jobs=-1, verbose=2)
```

```
search1 = random_search.fit(x_train,y_train)
```

```
Fitting 5 folds for each of 100 candidates, totalling 500 fits
---------------------------------------------------------------------------
KeyboardInterrupt                         Traceback (most recent call last)
<ipython-input-41-457d1100ddd3> in <cell line: 1>()
----> 1 search1 = random_search.fit(x_train,y_train)

        7 frames
/usr/local/lib/python3.10/dist-packages/joblib/parallel.py in _retrieve(self)
   1760                 (self._jobs[0].get_status(
   1761                     timeout=self.timeout) == TASK_PENDING)):
-> 1762                 time.sleep(0.01)
   1763                 continue
   1764

KeyboardInterrupt:
```

```
search1.best_params_
```

```
svc = SVC(C= 0.3321408221627493,
 coef0=6.852383815557032,
 degree= 2,
 gamma= 'scale',
 kernel= 'linear')
```

```
svc.fit(x_train.toarray(),y_train)
svc_preds_train = svc.predict(x_train.toarray())
svc_preds_test = svc.predict(x_test.toarray())
```

```
print('Accuracy score for train data : ', round(accuracy_score(y_train, svc_preds_train),2))
print('Accuracy score for test data : ', round(accuracy_score(y_test, svc_preds_test),2))
```

Creating App using Gradio

```
!pip install gradio
```

```
    Downloading fastapi-0.115.5-py3-none-any.whl (94 kB)
       ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 94.9/94.9 kB 8.1 MB/s eta 0:00:00
    Downloading MarkupSafe-2.1.5-cp310-cp310-manylinux_2_17_x86_64.manylinux2014_x86_64.whl (25 kB)
    Downloading ruff-0.7.3-py3-none-manylinux_2_17_x86_64.manylinux2014_x86_64.whl (11.0 MB)
       ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 11.0/11.0 MB 103.4 MB/s eta 0:00:00
    Downloading safehttpx-0.1.1-py3-none-any.whl (8.4 kB)
    Downloading semantic_version-2.10.0-py2.py3-none-any.whl (15 kB)
    Downloading starlette-0.41.2-py3-none-any.whl (73 kB)
       ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 73.3/73.3 kB 6.0 MB/s eta 0:00:00
    Downloading uvicorn-0.32.0-py3-none-any.whl (63 kB)
       ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 63.7/63.7 kB 5.2 MB/s eta 0:00:00
    Downloading ffmpy-0.4.0-py3-none-any.whl (5.8 kB)
    Downloading pydub-0.25.1-py2.py3-none-any.whl (32 kB)
    Downloading websockets-12.0-cp310-cp310-manylinux_2_5_x86_64.manylinux1_x86_64.manylinux_2_17_x86_64.manylinux2014_x86_64.whl (130 kB
       ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 130.2/130.2 kB 10.8 MB/s eta 0:00:00
    Installing collected packages: pydub, websockets, uvicorn, tomlkit, semantic-version, ruff, python-multipart, markupsafe, ffmpy, aiof
      Attempting uninstall: markupsafe
        Found existing installation: MarkupSafe 3.0.2
        Uninstalling MarkupSafe-3.0.2:
          Successfully uninstalled MarkupSafe-3.0.2
    Successfully installed aiofiles-23.2.1 fastapi-0.115.5 ffmpy-0.4.0 gradio-5.5.0 gradio-client-1.4.2 markupsafe-2.1.5 pydub-0.25.1 pyt
```

```python
import gradio as gr
```

```python
def transform(input_text):
```