SPAM MAIL PREDICTION

SUBMITTED BY

Lavanya B	CB20S202960
Nandhini M	CB20S202961
Parameshwari M	CB20S202962
Abinaya M	CB20S202948
Divya P	CB20S202949

CONTENT:

Email filtering:

one of the primary methods for spam mail detection is email filtering. It involves categorize incoming into spam and non-spam. Machine learning algorithm can be trained to filter out spam mails based on their content and metadata.

Natural Language Processing:

Natural language processing(NLP) is a techinque that enables machine to understand and process human language. It plays a crucial role in spam detection, as it helps in extracting meaningful features from emails such as subject, body, and attachments.

Text Classification:

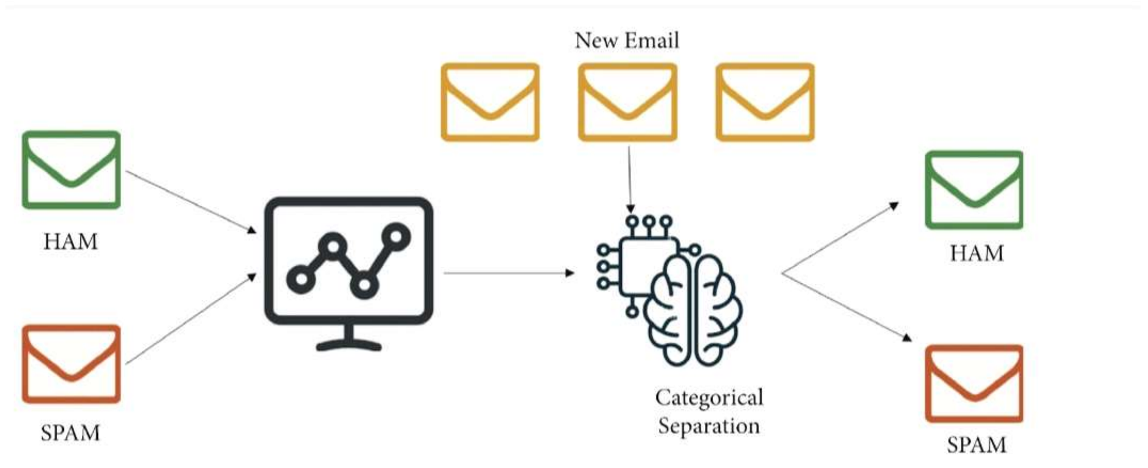
Text classfication is a supervised learning technique used for spam detection. It involves labelling emails as spam or non-spam based on their features, such as the present of certain keyword,tone, or grammer.

Features Engineering:

Features engineering is the process of selecting relevant features from the email to classify it as spam or non-spam. It involves extracting features such as the sender's email address, the presence of certain words or phrases, and the length of the email.

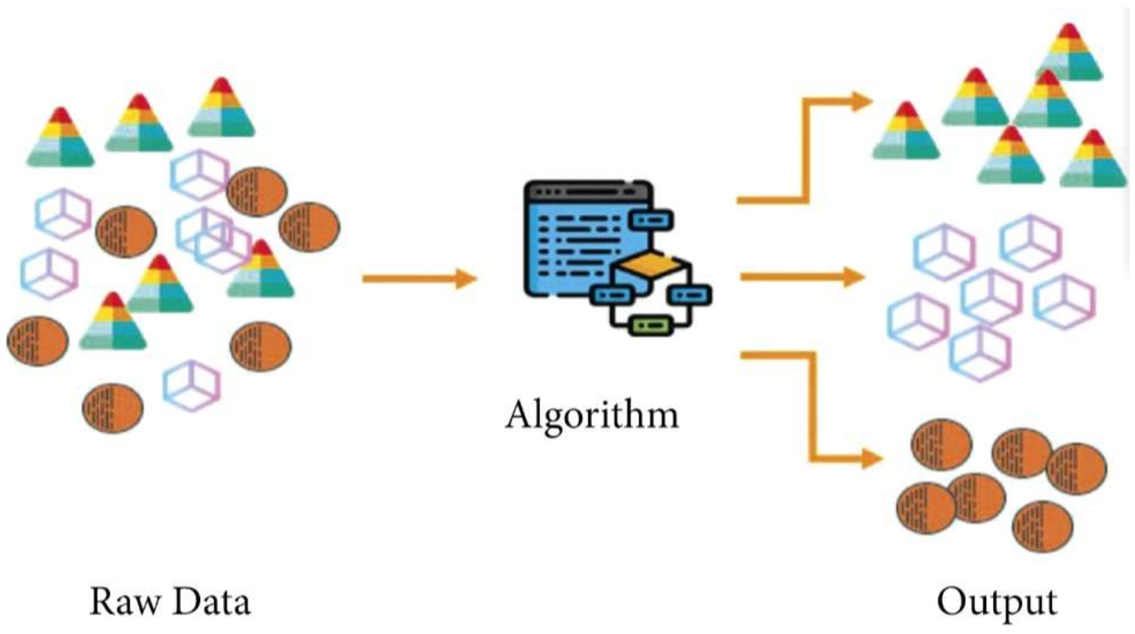
Supervised Learning:

Supervised learning is a technique that involves training the model on lables of new, unlabeled data. It is widely used in spam detection for text classification tasks



Unsupervised Learning:

Unsupervised learning is a technique used to find hidden pattern in the data without the need for labelled data. It can be used for ankomaly rule mining.

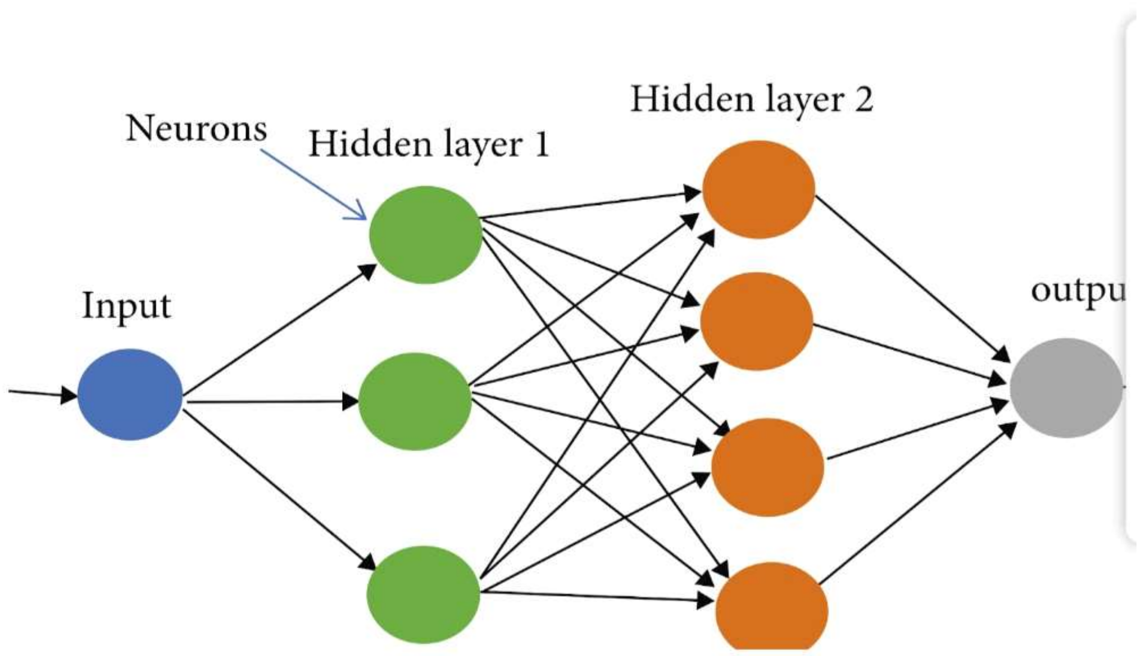


Deep Learning:

Deep learning is a subfield of machine learning that involves training deep neural nerworks with multiple hidden layers to learn complex features from the data. It has shown great promise in spam detection tasks.

Neural Networks:

Neural networks are a type of deep learning model inspired by the human brain. They can be trained to extract meaningful features from emails and classify them as spam or non-spam.



Decision Trees:

Decision trees are a simple yet effective algorithm used for classification tasks. They can be used for features selection, and the results can be easily interpreted.

Random Forest:

Random forest is an ensemble learning technique that combines multiple decisio trees to improve the classificatio performance. It is widely used in spam detection due to its high accuracy and robustness.

Support Vector Machines:

Support Vector Machines(SVM)are a popular machine learning algorithim used for classification tasks. They work by finding the hyperplane that different classes.

Naive Bayes:

Naive bayes is a probailistic algorithm widely used in text classification tasks, including spam detection. It works by calculating the probability of a message being spam given its features.

Conclusion:

with the right tools and technique, it is possible to build highly effective spam mail detection system using machine learning. By leveraging the power of these technique,we can help protect individuals and organization from the growing threat of spam and other email-based attacks.

program:

```
{
 "cells": [
  {
   "cell_type": "markdown",
   "id": "b5346c69",
   "metadata": {},
   "source": [
    "# Importing the Dependencies"
   ]
  },
  {
   "cell_type": "code",
   "execution_count": 1,
 "id": "bc714010",
 "metadata": {},
 "outputs": [],
 "source": [
  "import numpy as np\n",
  "import pandas as pd\n",
  "from sklearn.model_selection import train_test_split\n",
  "from sklearn.feature_extraction.text import TfidfVectorizer\n",
  "from sklearn.linear_model import LogisticRegression\n",
  "from sklearn.metrics import accuracy_score\n"
 ]
},
{
 "cell_type": "markdown",
 "id": "c4f38654",
 "metadata": {},
 "source": [
  "# Data Collection & Pre-Processing"
 ]
},
{
 "cell_type": "code",
 "execution_count": 2,
 "id": "f8d225f2",
   "metadata": {},
   "outputs": [],
   "source": [
    "raw_mail_data = pd.read_csv('mail_data.csv')"
   ]
  },
  {
   "cell_type": "code",
   "execution_count": 3,
   "id": "9492e260",
   "metadata": {},
   "outputs": [
    {
     "data": {
      "text/html": [
       "<div>\n",
       "<style scoped>\n",
       "    .dataframe tbody tr th:only-of-type {\n",
       "        vertical-align: middle;\n",
       "    }\n",
       "\n",
       "    .dataframe tbody tr th {\n",
       "        vertical-align: top;\n",
       "    }\n",
"\n",
"    .dataframe thead th {\n",
"        text-align: right;\n",
"    }\n",
"</style>\n",
"<table border=\"1\" class=\"dataframe\">\n",
"  <thead>\n",
"    <tr style=\"text-align: right;\">\n",
"      <th></th>\n",
"      <th>Category</th>\n",
"      <th>Message</th>\n",
"    </tr>\n",
"  </thead>\n",
"  <tbody>\n",
"    <tr>\n",
"      <th>0</th>\n",
"      <td>ham</td>\n",
"      <td>Go until jurong point, crazy.. Available only ...</td>\n",
"    </tr>\n",
"    <tr>\n",
"      <th>1</th>\n",
"      <td>ham</td>\n",
"      <td>Ok lar... Joking wif u oni...</td>\n",
"    </tr>\n",
"    <tr>\n",
"      <th>2</th>\n",
"      <td>spam</td>\n",
"      <td>Free entry in 2 a wkly comp to win FA Cup fina...</td>\n",
"    </tr>\n",
"    <tr>\n",
"      <th>3</th>\n",
"      <td>ham</td>\n",
"      <td>U dun say so early hor... U c already then say...</td>\n",
"    </tr>\n",
"    <tr>\n",
"      <th>4</th>\n",
"      <td>ham</td>\n",
"      <td>Nah I don't think he goes to usf, he lives aro...</td>\n",
"    </tr>\n",
"    <tr>\n",
"      <th>...</th>\n",
"      <td>...</td>\n",
"      <td>...</td>\n",
"    </tr>\n",
"    <tr>\n",
"      <th>5567</th>\n",
"      <td>spam</td>\n",
"      <td>This is the 2nd time we have tried 2 contact u...</td>\n",
"    </tr>\n",
"    <tr>\n",
"      <th>5568</th>\n",
"      <td>ham</td>\n",
"      <td>Will ü b going to esplanade fr home?</td>\n",
"    </tr>\n",
"    <tr>\n",
"      <th>5569</th>\n",
"      <td>ham</td>\n",
"      <td>Pity, * was in mood for that. So...any other s...</td>\n",
"    </tr>\n",
"    <tr>\n",
"      <th>5570</th>\n",
"      <td>ham</td>\n",
"      <td>The guy did some bitching but I acted like i'd...</td>\n",
"    </tr>\n",
"    <tr>\n",
"      <th>5571</th>\n",
"      <td>ham</td>\n",
"      <td>Rofl. Its true to its name</td>\n",
"    </tr>\n",
"  </tbody>\n",
"</table>\n",
"<p>5572 rows × 2 columns</p>\n",
    "</div>"
   ],
   "text/plain": [
    "    Category                                           Message\n",
    "0        ham  Go until jurong point, crazy.. Available only ...\n",
    "1        ham                      Ok lar... Joking wif u oni...\n",
    "2       spam  Free entry in 2 a wkly comp to win FA Cup fina...\n",
    "3        ham  U dun say so early hor... U c already then say...\n",
    "4        ham  Nah I don't think he goes to usf, he lives aro...\n",
    "...      ...                                               ...\n",
    "5567    spam  This is the 2nd time we have tried 2 contact u...\n",
    "5568     ham               Will ü b going to esplanade fr home?\n",
    "5569     ham  Pity, * was in mood for that. So...any other s...\n",
    "5570     ham  The guy did some bitching but I acted like i'd...\n",
    "5571     ham                         Rofl. Its true to its name\n",
    "\n",
    "[5572 rows x 2 columns]"
   ]
  },
  "execution_count": 3,
  "metadata": {},
  "output_type": "execute_result"
 }
],
 "source": [
  "raw_mail_data"
 ]
},
{
 "cell_type": "markdown",
 "id": "c3621ca9",
 "metadata": {},
 "source": [
  "# replace the null values with a null string"
 ]
},
{
 "cell_type": "code",
 "execution_count": 4,
 "id": "a1949295",
 "metadata": {},
 "outputs": [],
 "source": [
  "mail_data = raw_mail_data.where((pd.notnull(raw_mail_data)),'')"
 ]
},
{
 "cell_type": "code",
"execution_count": 5,
"id": "0a801669",
"metadata": {},
"outputs": [
 {
  "data": {
   "text/html": [
    "<div>\n",
    "<style scoped>\n",
    "    .dataframe tbody tr th:only-of-type {\n",
    "        vertical-align: middle;\n",
    "    }\n",
    "\n",
    "    .dataframe tbody tr th {\n",
    "        vertical-align: top;\n",
    "    }\n",
    "\n",
    "    .dataframe thead th {\n",
    "        text-align: right;\n",
    "    }\n",
    "</style>\n",
    "<table border=\"1\" class=\"dataframe\">\n",
    "  <thead>\n",
    "    <tr style=\"text-align: right;\">\n",
"      <th></th>\n",
"      <th>Category</th>\n",
"      <th>Message</th>\n",
"    </tr>\n",
"  </thead>\n",
"  <tbody>\n",
"    <tr>\n",
"      <th>0</th>\n",
"      <td>ham</td>\n",
"      <td>Go until jurong point, crazy.. Available only ...</td>\n",
"    </tr>\n",
"    <tr>\n",
"      <th>1</th>\n",
"      <td>ham</td>\n",
"      <td>Ok lar... Joking wif u oni...</td>\n",
"    </tr>\n",
"    <tr>\n",
"      <th>2</th>\n",
"      <td>spam</td>\n",
"      <td>Free entry in 2 a wkly comp to win FA Cup fina...</td>\n",
"    </tr>\n",
"    <tr>\n",
"      <th>3</th>\n",
"      <td>ham</td>\n",
 "     <td>U dun say so early hor... U c already then say...</td>\n",
 "    </tr>\n",
 "    <tr>\n",
 "      <th>4</th>\n",
 "      <td>ham</td>\n",
 "      <td>Nah I don't think he goes to usf, he lives aro...</td>\n",
 "    </tr>\n",
 "  </tbody>\n",
 "</table>\n",
 "</div>"
 ],
 "text/plain": [
 "  Category                                    Message\n",
 "0      ham  Go until jurong point, crazy.. Available only ...\n",
 "1      ham                      Ok lar... Joking wif u oni...\n",
 "2     spam  Free entry in 2 a wkly comp to win FA Cup fina...\n",
 "3      ham  U dun say so early hor... U c already then say...\n",
 "4      ham  Nah I don't think he goes to usf, he lives aro..."
 ]
},
"execution_count": 5,
"metadata": {},
"output_type": "execute_result"
}
```

```
 ],
 "source": [
  "mail_data.head()"
 ]
},
{
 "cell_type": "code",
 "execution_count": 6,
 "id": "04cc2412",
 "metadata": {},
 "outputs": [
  {
   "data": {
    "text/plain": [
     "(5572, 2)"
    ]
   },
   "execution_count": 6,
   "metadata": {},
   "output_type": "execute_result"
  }
 ],
 "source": [
  "mail_data.shape"
 ]
},
{
 "cell_type": "markdown",
 "id": "082b8a6a",
 "metadata": {},
 "source": [
  "# Label Encoding"
 ]
},
{
 "cell_type": "code",
 "execution_count": 8,
 "id": "93d1923c",
 "metadata": {},
 "outputs": [],
 "source": [
  "mail_data.loc[mail_data['Category'] == 'spam', 'Category',] = 0\n",
  "mail_data.loc[mail_data['Category'] == 'ham', 'Category',] = 1"
 ]
},
{
 "cell_type": "code",
 "execution_count": 9,
 "id": "d2a0ccc5",
 "metadata": {},
 "outputs": [],
 "source": [
  "X = mail_data['Message']\n",
  "\n",
  "Y = mail_data['Category']"
 ]
},
{
 "cell_type": "code",
 "execution_count": 10,
 "id": "6e759f67",
 "metadata": {},
 "outputs": [
  {
   "name": "stdout",
   "output_type": "stream",
   "text": [
    "0      Go until jurong point, crazy.. Available only ...\n",
    "1                          Ok lar... Joking wif u oni...\n",
    "2      Free entry in 2 a wkly comp to win FA Cup fina...\n",
    "3      U dun say so early hor... U c already then say...\n",
    "4      Nah I don't think he goes to usf, he lives aro...\n",
    "                              ...                        \n",
    "5567    This is the 2nd time we have tried 2 contact u...\n",
    "5568                 Will ü b going to esplanade fr home?\n",
    "5569    Pity, * was in mood for that. So...any other s...\n",
    "5570    The guy did some bitching but I acted like i'd...\n",
    "5571                           Rofl. Its true to its name\n",
    "Name: Message, Length: 5572, dtype: object\n"
   ]
  }
 ],
 "source": [
  "print(X)"
 ]
},
{
 "cell_type": "code",
 "execution_count": 11,
 "id": "5a21e947",
 "metadata": {},
 "outputs": [
  {
   "name": "stdout",
   "output_type": "stream",
   "text": [
    "0      1\n",
    "1      1\n",
    "2      0\n",
    "3      1\n",
    "4      1\n",
    "       ..\n",
    "5567   0\n",
    "5568   1\n",
    "5569   1\n",
    "5570   1\n",
    "5571   1\n",
    "Name: Category, Length: 5572, dtype: object\n"
   ]
  }
 ],
 "source": [
  "print(Y)"
 ]
},
{
 "cell_type": "markdown",
 "id": "d4df24eb",
 "metadata": {},
 "source": [
  "# Splitting the data into training data & test data"
 ]
},
{
 "cell_type": "code",
 "execution_count": 12,
 "id": "49daf595",
 "metadata": {},
 "outputs": [],
 "source": [
  "X_train, X_test, Y_train, Y_test = train_test_split(X, Y, test_size=0.2, random_state=3)"
 ]
},
{
 "cell_type": "code",
 "execution_count": 13,
 "id": "418f4f7a",
 "metadata": {},
 "outputs": [
  {
   "name": "stdout",
   "output_type": "stream",
   "text": [
    "(5572,)\n",
    "(4457,)\n",
    "(1115,)\n"
   ]
  }
 ],
 "source": [
  "print(X.shape)\n",
  "print(X_train.shape)\n",
  "print(X_test.shape)"
 ]
},
{
 "cell_type": "markdown",
 "id": "33778931",
 "metadata": {},
 "source": [
  "# Feature Extraction"
 ]
},
{
 "cell_type": "code",
 "execution_count": 14,
 "id": "a8cb5198",
 "metadata": {},
 "outputs": [],
 "source": [
  "feature_extraction = TfidfVectorizer(min_df = 1, stop_words='english', lowercase='True')"
 ]
},
{
 "cell_type": "code",
 "execution_count": 15,
 "id": "a17efac1",
 "metadata": {},
 "outputs": [],
 "source": [
  "X_train_features = feature_extraction.fit_transform(X_train)\n",
  "X_test_features = feature_extraction.transform(X_test)\n"
 ]
},
{
 "cell_type": "code",
 "execution_count": 16,
 "id": "fe6751ad",
 "metadata": {},
 "outputs": [],
 "source": [
  "Y_train = Y_train.astype('int')\n",
  "Y_test = Y_test.astype('int')"
 ]
},
{
 "cell_type": "code",
 "execution_count": 17,
 "id": "4422491c",
 "metadata": {},
 "outputs": [
  {
   "name": "stdout",
   "output_type": "stream",
   "text": [
    "3075                Don know. I did't msg him recently.\n",
    "1787    Do you know why god created gap between your f...\n",
    "1614                     Thnx dude. u guys out 2nite?\n",
    "4304                                 Yup i'm free...\n",
    "3266    44 7732584351, Do you want a New Nokia 3510i c...\n",
    "                            ...                        \n",
    "789     5 Free Top Polyphonic Tones call 087018728737,...\n",
    "968     What do u want when i come back?.a beautiful n...\n",
    "1667    Guess who spent all last night phasing in and ...\n",
    "3321    Eh sorry leh... I din c ur msg. Not sad alread...\n",
    "1688    Free Top ringtone -sub to weekly ringtone-get ...\n",
    "Name: Message, Length: 4457, dtype: object\n"
   ]
  }
 ],
 "source": [
  "print(X_train)"
 ]
},
{
 "cell_type": "code",
 "execution_count": 18,
 "id": "83f43405",
 "metadata": {},
 "outputs": [
  {
   "name": "stdout",
   "output_type": "stream",
   "text": [
    "  (0, 5413)\t0.6198254967574347\n",
    "  (0, 4456)\t0.4168658090846482\n",
    "  (0, 2224)\t0.413103377943378\n",
    "  (0, 3811)\t0.34780165336891333\n",
    "  (0, 2329)\t0.38783870336935383\n",
    "  (1, 4080)\t0.18880584110891163\n",
"  (1, 3185)\t0.29694482957694585\n",
"  (1, 3325)\t0.31610586766078863\n",
"  (1, 2957)\t0.3398297002864083\n",
"  (1, 2746)\t0.3398297002864083\n",
"  (1, 918)\t0.22871581159877646\n",
"  (1, 1839)\t0.2784903590561455\n",
"  (1, 2758)\t0.3226407885943799\n",
"  (1, 2956)\t0.33036995955537024\n",
"  (1, 1991)\t0.33036995955537024\n",
"  (1, 3046)\t0.2503712792613518\n",
"  (1, 3811)\t0.17419952275504033\n",
"  (2, 407)\t0.509272536051008\n",
"  (2, 3156)\t0.4107239318312698\n",
"  (2, 2404)\t0.45287711070606745\n",
"  (2, 6601)\t0.6056811524587518\n",
"  (3, 2870)\t0.5864269879324768\n",
"  (3, 7414)\t0.8100020912469564\n",
"  (4, 50)\t0.23633754072626942\n",
"  (4, 5497)\t0.15743785051118356\n",
"  :\t:\n",
"  (4454, 4602)\t0.2669765732445391\n",
"  (4454, 3142)\t0.32014451677763156\n",
"  (4455, 2247)\t0.37052851863170466\n",
"  (4455, 2469)\t0.35441545511837946\n",
    "  (4455, 5646)\t0.33545678464631296\n",
    "  (4455, 6810)\t0.29731757715898277\n",
    "  (4455, 6091)\t0.23103841516927642\n",
    "  (4455, 7113)\t0.30536590342067704\n",
    "  (4455, 3872)\t0.3108911491788658\n",
    "  (4455, 4715)\t0.30714144758811196\n",
    "  (4455, 6916)\t0.19636985317119715\n",
    "  (4455, 3922)\t0.31287563163368587\n",
    "  (4455, 4456)\t0.24920025316220423\n",
    "  (4456, 141)\t0.292943737785358\n",
    "  (4456, 647)\t0.30133182431707617\n",
    "  (4456, 6311)\t0.30133182431707617\n",
    "  (4456, 5569)\t0.4619395404299172\n",
    "  (4456, 6028)\t0.21034888000987115\n",
    "  (4456, 7154)\t0.24083218452280053\n",
    "  (4456, 7150)\t0.3677554681447669\n",
    "  (4456, 6249)\t0.17573831794959716\n",
    "  (4456, 6307)\t0.2752760476857975\n",
    "  (4456, 334)\t0.2220077711654938\n",
    "  (4456, 5778)\t0.16243064490100795\n",
    "  (4456, 2870)\t0.31523196273113385\n"
   ]
  }
 ],
 "source": [
  "print(X_train_features)"
 ]
},
{
 "cell_type": "markdown",
 "id": "1e5d5cec",
 "metadata": {},
 "source": [
  "# Training the Model\n"
 ]
},
{
 "cell_type": "markdown",
 "id": "b89f133f",
 "metadata": {},
 "source": [
  "# Logistic Regression"
 ]
},
{
 "cell_type": "code",
 "execution_count": 19,
 "id": "5ad5c522",
 "metadata": {},
 "outputs": [],
 "source": [
  "model = LogisticRegression()"
 ]
},
{
 "cell_type": "code",
 "execution_count": 20,
 "id": "4f2e74c9",
 "metadata": {},
 "outputs": [
  {
   "data": {
    "text/plain": [
     "LogisticRegression()"
    ]
   },
   "execution_count": 20,
   "metadata": {},
   "output_type": "execute_result"
  }
 ],
 "source": [
   "model.fit(X_train_features, Y_train)"
  ]
 },
 {
  "cell_type": "markdown",
  "id": "2bb3e17e",
  "metadata": {},
  "source": [
   "# Evaluating the trained model"
  ]
 },
 {
  "cell_type": "code",
  "execution_count": 21,
  "id": "05eb0089",
  "metadata": {},
  "outputs": [],
  "source": [
   "prediction_on_training_data = model.predict(X_train_features)\n",
   "accuracy_on_training_data = accuracy_score(Y_train, prediction_on_training_data)"
  ]
 },
 {
  "cell_type": "code",
 "execution_count": 22,
 "id": "2302d9c0",
 "metadata": {},
 "outputs": [
  {
   "name": "stdout",
   "output_type": "stream",
   "text": [
    "Accuracy on training data :  0.9670181736594121\n"
   ]
  }
 ],
 "source": [
  "print('Accuracy on training data : ', accuracy_on_training_data)"
 ]
},
{
 "cell_type": "raw",
 "id": "7d791ae5",
 "metadata": {},
 "source": [
  "# prediction on test data"
 ]
},
{
 "cell_type": "code",
 "execution_count": 23,
 "id": "6067f728",
 "metadata": {},
 "outputs": [],
 "source": [
  "prediction_on_test_data = model.predict(X_test_features)\n",
  "accuracy_on_test_data = accuracy_score(Y_test, prediction_on_test_data)"
 ]
},
{
 "cell_type": "code",
 "execution_count": 24,
 "id": "dfaf7f10",
 "metadata": {},
 "outputs": [
  {
   "name": "stdout",
   "output_type": "stream",
   "text": [
    "Accuracy on test data :  0.9659192825112107\n"
   ]
  }
```

```
   ],
   "source": [
    "print('Accuracy on test data : ', accuracy_on_test_data)"
   ]
  },
  {
   "cell_type": "markdown",
   "id": "b4297c9f",
   "metadata": {},
   "source": [
    "# Building a Predictive System"
   ]
  },
  {
   "cell_type": "code",
   "execution_count": 25,
   "id": "2eb28343",
   "metadata": {},
   "outputs": [],
   "source": [
    "input_mail = [\"I've been searching for the right words to thank you for this breather. I
promise i wont take your help for granted and will fulfil my promise. You have been wonderful
and a blessing at all times\"]\n"
   ]
  },
{
 "cell_type": "code",
 "execution_count": 26,
 "id": "b5e1e009",
 "metadata": {},
 "outputs": [],
 "source": [
  "input_data_features = feature_extraction.transform(input_mail)\n"
 ]
},
{
 "cell_type": "raw",
 "id": "29e1b923",
 "metadata": {},
 "source": [
  "# making prediction"
 ]
},
{
 "cell_type": "code",
 "execution_count": 27,
 "id": "46fa38c5",
 "metadata": {},
 "outputs": [
   {
    "name": "stdout",
    "output_type": "stream",
    "text": [
     "[1]\n"
    ]
   }
  ],
  "source": [
   "prediction = model.predict(input_data_features)\n",
   "print(prediction)\n"
  ]
 },
 {
  "cell_type": "code",
  "execution_count": 28,
  "id": "16fd8847",
  "metadata": {},
  "outputs": [
   {
    "name": "stdout",
    "output_type": "stream",
    "text": [
     "Ham mail\n"
    ]
   }
  ],
  "source": [
   "if (prediction[0]==1):\n",
   "  print('Ham mail')\n",
   "\n",
   "else:\n",
   "  print('Spam mail')"
  ]
 },
 {
  "cell_type": "code",
  "execution_count": null,
  "id": "9374d227",
  "metadata": {},
  "outputs": [],
  "source": []
 },
 {
  "cell_type": "code",
  "execution_count": null,
  "id": "4cd31377",
  "metadata": {},
  "outputs": [],
  "source": []
 }
],
"metadata": {
 "kernelspec": {
  "display_name": "Python 3 (ipykernel)",
  "language": "python",
  "name": "python3"
 },
 "language_info": {
  "codemirror_mode": {
   "name": "ipython",
   "version": 3
  },
  "file_extension": ".py",
  "mimetype": "text/x-python",
  "name": "python",
  "nbconvert_exporter": "python",
  "pygments_lexer": "ipython3",
  "version": "3.9.12"
 }
},
"nbformat": 4,
 "nbformat_minor": 5
}
```