```
{
 "cells": [
  {
   "cell_type": "markdown",
   "metadata": {
    "id": "fNn9aA-3Onp2"
   },
   "source": [
    "# Identify the IMDB Movies review recieved as either **positive or negative**\n"
   ]
  },
  {
   "cell_type": "code",
   "execution_count": null,
   "metadata": {
    "id": "Yi5YxCCaNR0Y"
   },
   "outputs": [],
   "source": [
    "#N Ravinder Reddy\n",
    "#importing the libraries\n",
    "import numpy as np # Numerical Computations\n",
    "import pandas as pd # Data Manipulations"
   ]
  },
  {
   "cell_type": "code",
   "execution_count": null,
   "metadata": {
    "id": "l0W6BhJvNlOA"
   },
   "
```

```
 "outputs": [],
 "source": [
  "# reading the imdb movies review store into df\n",
  "df=pd.read_csv('/content/IMDB_Dataset.csv',)"
 ]
},
{
 "cell_type": "code",
 "execution_count": null,
 "metadata": {
  "colab": {
   "base_uri": "https://localhost:8080/",
   "height": 206
  },
  "id": "8wJxVfipQYPm",
  "outputId": "5e695a44-bcdb-4448-8142-4dc04a6895b7"
 },
 "outputs": [
  {
   "data": {
    "text/html": [
     "\n",
     "  <div id=\"df-46ae03fc-02e6-4d17-b8e6-7fc88348ca96\" class=\"colab-df-container\">\n",
     "    <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>review</th>\n",
"      <th>sentiment</th>\n",
"    </tr>\n",
"  </thead>\n",
"  <tbody>\n",
"    <tr>\n",
"      <th>0</th>\n",
"      <td>One of the other reviewers has mentioned that ...</td>\n",
"      <td>positive</td>\n",
"    </tr>\n",
"    <tr>\n",
"      <th>1</th>\n",
"      <td>A wonderful little production. &amp;lt;br /&amp;gt;&amp;lt;br /&amp;gt;The...</td>\n",
"      <td>positive</td>\n",
"    </tr>\n",
"    <tr>\n",
"      <th>2</th>\n",
"      <td>I thought this was a wonderful way to spend ti...</td>\n",
"      <td>positive</td>\n",
"    </tr>\n",
"    <tr>\n",
```

```
"     <th>3</th>\n",
"     <td>Basically there's a family where a little boy ...</td>\n",
"     <td>negative</td>\n",
"   </tr>\n",
"   <tr>\n",
"     <th>4</th>\n",
"     <td>Petter Mattei's \"Love in the Time of Money\" is...</td>\n",
"     <td>positive</td>\n",
"   </tr>\n",
"  </tbody>\n",
"</table>\n",
"</div>\n",
"   <div class=\"colab-df-buttons\">\n",
"\n",
"  <div class=\"colab-df-container\">\n",
"   <button class=\"colab-df-convert\" onclick=\"convertToInteractive('df-46ae03fc-02e6-4d17-b8e6-7fc88348ca96')\"\n",
"           title=\"Convert this dataframe to an interactive table.\"\n",
"           style=\"display:none;\">\n",
"\n",
"  <svg xmlns=\"http://www.w3.org/2000/svg\" height=\"24px\" viewBox=\"0 -960 960 960\">\n",
"    <path d=\"M120-120v-720h720v720H120Zm60-500h600v-160H180v160Zm220 220h160v-160H400v160Zm0 220h160v-160H400v160ZM180-400h160v-160H180v160Zm440 0h160v-160H620v160ZM180-180h160v-160H180v160Zm440 0h160v-160H620v160Z\"/>\n",
"  </svg>\n",
"   </button>\n",
"\n",
"  <style>\n",
"   .colab-df-container {\n",
"     display:flex;\n",
"     gap: 12px;\n",
```

```
"    }\n",
"\n",
"    .colab-df-convert {\n",
"      background-color: #E8F0FE;\n",
"      border: none;\n",
"      border-radius: 50%;\n",
"      cursor: pointer;\n",
"      display: none;\n",
"      fill: #1967D2;\n",
"      height: 32px;\n",
"      padding: 0 0 0 0;\n",
"      width: 32px;\n",
"    }\n",
"\n",
"    .colab-df-convert:hover {\n",
"      background-color: #E2EBFA;\n",
"      box-shadow: 0px 1px 2px rgba(60, 64, 67, 0.3), 0px 1px 3px 1px rgba(60, 64, 67, 0.15);\n",
"      fill: #174EA6;\n",
"    }\n",
"\n",
"    .colab-df-buttons div {\n",
"      margin-bottom: 4px;\n",
"    }\n",
"\n",
"    [theme=dark] .colab-df-convert {\n",
"      background-color: #3B4455;\n",
"      fill: #D2E3FC;\n",
"    }\n",
"\n",
"    [theme=dark] .colab-df-convert:hover {\n",
"      background-color: #434B5C;\n",
```

```
"      box-shadow: 0px 1px 3px 1px rgba(0, 0, 0, 0.15);\n",
"      filter: drop-shadow(0px 1px 2px rgba(0, 0, 0, 0.3));\n",
"      fill: #FFFFFF;\n",
"    }\n",
"  </style>\n",
"\n",
"    <script>\n",
"      const buttonEl =\n",
"        document.querySelector('#df-46ae03fc-02e6-4d17-b8e6-7fc88348ca96 button.colab-df-convert');\n",
"      buttonEl.style.display =\n",
"        google.colab.kernel.accessAllowed ? 'block' : 'none';\n",
"\n",
"      async function convertToInteractive(key) {\n",
"        const element = document.querySelector('#df-46ae03fc-02e6-4d17-b8e6-7fc88348ca96');\n",
"        const dataTable =\n",
"          await google.colab.kernel.invokeFunction('convertToInteractive',\n",
"                                                   [key], {});\n",
"        if (!dataTable) return;\n",
"\n",
"        const docLinkHtml = 'Like what you see? Visit the ' +\n",
"          '<a target=\"_blank\" href=https://colab.research.google.com/notebooks/data_table.ipynb>data table notebook</a>'\n",
"          + ' to learn more about interactive tables.';\n",
"        element.innerHTML = '';\n",
"        dataTable['output_type'] = 'display_data';\n",
"        await google.colab.output.renderOutput(dataTable, element);\n",
"        const docLink = document.createElement('div');\n",
"        docLink.innerHTML = docLinkHtml;\n",
"        element.appendChild(docLink);\n",
"      }\n",
```

```
"    </script>\n",
"  </div>\n",
"\n",
"\n",
"<div id=\"df-96ddb545-b28f-4acc-bbb4-70db73a20dae\">\n",
"  <button class=\"colab-df-quickchart\" onclick=\"quickchart('df-96ddb545-b28f-4acc-bbb4-70db73a20dae')\"\n",
"          title=\"Suggest charts.\"\n",
"          style=\"display:none;\">\n",
"\n",
"<svg xmlns=\"http://www.w3.org/2000/svg\" height=\"24px\"viewBox=\"0 0 24 24\"\n",
"     width=\"24px\">\n",
"    <g>\n",
"      <path d=\"M19 3H5c-1.1 0-2 .9-2 2v14c0 1.1.9 2 2 2h14c1.1 0 2-.9 2-2V5c0-1.1-.9-2-2-2zM9 17H7v-7h2v7zm4 0h-2V7h2v10zm4 0h-2v-4h2v4z\"/>\n",
"    </g>\n",
"</svg>\n",
"  </button>\n",
"\n",
"<style>\n",
"  .colab-df-quickchart {\n",
"      --bg-color: #E8F0FE;\n",
"      --fill-color: #1967D2;\n",
"      --hover-bg-color: #E2EBFA;\n",
"      --hover-fill-color: #174EA6;\n",
"      --disabled-fill-color: #AAA;\n",
"      --disabled-bg-color: #DDD;\n",
"  }\n",
"\n",
"  [theme=dark] .colab-df-quickchart {\n",
"      --bg-color: #3B4455;\n",
"      --fill-color: #D2E3FC;\n",
```

```
"      --hover-bg-color: #434B5C;\n",
"      --hover-fill-color: #FFFFFF;\n",
"      --disabled-bg-color: #3B4455;\n",
"      --disabled-fill-color: #666;\n",
"  }\n",
"\n",
"  .colab-df-quickchart {\n",
"    background-color: var(--bg-color);\n",
"    border: none;\n",
"    border-radius: 50%;\n",
"    cursor: pointer;\n",
"    display: none;\n",
"    fill: var(--fill-color);\n",
"    height: 32px;\n",
"    padding: 0;\n",
"    width: 32px;\n",
"  }\n",
"\n",
"  .colab-df-quickchart:hover {\n",
"    background-color: var(--hover-bg-color);\n",
"    box-shadow: 0 1px 2px rgba(60, 64, 67, 0.3), 0 1px 3px 1px rgba(60, 64, 67, 0.15);\n",
"    fill: var(--button-hover-fill-color);\n",
"  }\n",
"\n",
"  .colab-df-quickchart-complete:disabled,\n",
"  .colab-df-quickchart-complete:disabled:hover {\n",
"    background-color: var(--disabled-bg-color);\n",
"    fill: var(--disabled-fill-color);\n",
"    box-shadow: none;\n",
"  }\n",
"\n",
```

```
"  .colab-df-spinner {\n",
"    border: 2px solid var(--fill-color);\n",
"    border-color: transparent;\n",
"    border-bottom-color: var(--fill-color);\n",
"    animation:\n",
"      spin 1s steps(1) infinite;\n",
"  }\n",
"\n",
"  @keyframes spin {\n",
"    0% {\n",
"      border-color: transparent;\n",
"      border-bottom-color: var(--fill-color);\n",
"      border-left-color: var(--fill-color);\n",
"    }\n",
"    20% {\n",
"      border-color: transparent;\n",
"      border-left-color: var(--fill-color);\n",
"      border-top-color: var(--fill-color);\n",
"    }\n",
"    30% {\n",
"      border-color: transparent;\n",
"      border-left-color: var(--fill-color);\n",
"      border-top-color: var(--fill-color);\n",
"      border-right-color: var(--fill-color);\n",
"    }\n",
"    40% {\n",
"      border-color: transparent;\n",
"      border-right-color: var(--fill-color);\n",
"      border-top-color: var(--fill-color);\n",
"    }\n",
"    60% {\n",
```

```
"      border-color: transparent;\n",
"      border-right-color: var(--fill-color);\n",
"    }\n",
"    80% {\n",
"      border-color: transparent;\n",
"      border-right-color: var(--fill-color);\n",
"      border-bottom-color: var(--fill-color);\n",
"    }\n",
"    90% {\n",
"      border-color: transparent;\n",
"      border-bottom-color: var(--fill-color);\n",
"    }\n",
"  }\n",
"</style>\n",
"\n",
"  <script>\n",
"    async function quickchart(key) {\n",
"      const quickchartButtonEl =\n",
"        document.querySelector('#' + key + ' button');\n",
"      quickchartButtonEl.disabled = true;  // To prevent multiple clicks.\n",
"      quickchartButtonEl.classList.add('colab-df-spinner');\n",
"      try {\n",
"        const charts = await google.colab.kernel.invokeFunction(\n",
"            'suggestCharts', [key], {});\n",
"      } catch (error) {\n",
"        console.error('Error during call to suggestCharts:', error);\n",
"      }\n",
"      quickchartButtonEl.classList.remove('colab-df-spinner');\n",
"      quickchartButtonEl.classList.add('colab-df-quickchart-complete');\n",
"    }\n",
"    (() => {\n",
```

```
    "      let quickchartButtonEl =\n",
    "        document.querySelector('#df-96ddb545-b28f-4acc-bbb4-70db73a20dae button');\n",
    "      quickchartButtonEl.style.display =\n",
    "        google.colab.kernel.accessAllowed ? 'block' : 'none';\n",
    "    })();\n",
    "  </script>\n",
    "</div>\n",
    "    </div>\n",
    "  </div>\n"
   ],
   "text/plain": [
    "                                        review sentiment\n",
    "0  One of the other reviewers has mentioned that ...  positive\n",
    "1  A wonderful little production. <br /><br />The...  positive\n",
    "2  I thought this was a wonderful way to spend ti...  positive\n",
    "3  Basically there's a family where a little boy ...  negative\n",
    "4  Petter Mattei's \"Love in the Time of Money\" is...  positive"
   ]
  },
  "execution_count": 250,
  "metadata": {},
  "output_type": "execute_result"
 }
],
"source": [
 "#top 5 rows\n",
 "df.head()"
]
},
{
 "cell_type": "markdown",
```

```
 "metadata": {
  "id": "nmY7ERVdOnz4"
 },
 "source": [
  "*there are 2 columns with review and sentiment as names"
 ]
},
{
 "cell_type": "code",
 "execution_count": null,
 "metadata": {
  "colab": {
   "base_uri": "https://localhost:8080/"
  },
  "id": "efA_O-d_Rx0A",
  "outputId": "d89928e6-b1b7-4e69-ae4f-8a80c9b0473f"
 },
 "outputs": [
  {
   "data": {
    "text/plain": [
     "(50000, 2)"
    ]
   },
   "execution_count": 251,
   "metadata": {},
   "output_type": "execute_result"
  }
 ],
 "source": [
  "# shape of the dataset\n",
```

```
  "df.shape"
 ]
},
{
 "cell_type": "markdown",
 "metadata": {
  "id": "eX6zrdVYPHNe"
 },
 "source": [
  "*50000 rows and 2 columns"
 ]
},
{
 "cell_type": "code",
 "execution_count": null,
 "metadata": {
  "colab": {
   "base_uri": "https://localhost:8080/"
  },
  "id": "LJpY_ZkwTiHu",
  "outputId": "b64385c9-02f0-43c6-eb16-3b233b1d8708"
 },
 "outputs": [
  {
   "data": {
    "text/plain": [
     "(3000, 2)"
    ]
   },
   "execution_count": 252,
   "metadata": {},
```

```
   "output_type": "execute_result"
  }
 ],
 "source": [
  "df = df[0:3000]  # with 50000 records we are having resource limitations hence taken only 3000 rows\n",
  "df.shape"
 ]
},
{
 "cell_type": "markdown",
 "metadata": {
  "id": "sC-QqKvPVWUJ"
 },
 "source": [
  "*3000 rows  2 columns  "
 ]
},
{
 "cell_type": "code",
 "execution_count": null,
 "metadata": {
  "colab": {
   "base_uri": "https://localhost:8080/"
  },
  "id": "DuOVy9iKRx3f",
  "outputId": "6088f935-1edb-4084-d4db-a9a3b866856a"
 },
 "outputs": [
  {
   "data": {
```

```
    "text/plain": [
     "Index(['review', 'sentiment'], dtype='object')"
    ]
   },
   "execution_count": 253,
   "metadata": {},
   "output_type": "execute_result"
  }
 ],
 "source": [
  "df.columns"
 ]
},
{
 "cell_type": "code",
 "execution_count": null,
 "metadata": {
  "colab": {
   "base_uri": "https://localhost:8080/",
   "height": 123
  },
  "id": "WRFkgf_uVB_Y",
  "outputId": "c88aec49-723e-4838-934c-ef074b86876b"
 },
 "outputs": [
  {
   "data": {
    "application/vnd.google.colaboratory.intrinsic+json": {
     "type": "string"
    },
    "text/plain": [
```

"'A wonderful little production.

The filming technique is very unassuming- very old-time-BBC fashion and gives a comforting, and sometimes discomforting, sense of realism to the entire piece.

The actors are extremely well chosen- Michael Sheen not only \"has got all the polari\" but he has all the voices down pat too! You can truly see the seamless editing guided by the references to Williams\\' diary entries, not only is it well worth the watching but it is a terrificly written and performed piece. A masterful production about one of the great master\\'s of comedy and his life.

The realism really comes home with the little things: the fantasy of the guard which, rather than use the traditional \\'dream\\' techniques remains solid then disappears. It plays on our knowledge and our senses, particularly with the scenes concerning Orton and Halliwell and the sets (particularly of their flat with Halliwell\\'s murals decorating every surface) are terribly well done.'"

]

},

"execution_count": 254,

"metadata": {},

"output_type": "execute_result"

}

],

"source": [

"df.review[1] # here we can see our data is not processed ex. html tags are there, special character, case of words not uniform\n"

]

},

{

"cell_type": "code",

"execution_count": null,

"metadata": {

"colab": {

"base_uri": "https://localhost:8080/"

},

"id": "HEvJTLaIWBys",

"outputId": "d0a43161-5c3d-4819-c741-1af10b358668"

},

"outputs": [

{

```
  "data": {
   "text/plain": [
    "positive    1508\n",
    "negative    1492\n",
    "Name: sentiment, dtype: int64"
   ]
  },
  "execution_count": 255,
  "metadata": {},
  "output_type": "execute_result"
 }
],
"source": [
 "df['sentiment'].value_counts() # Which shows data is approximately balanced\n"
]
},
{
 "cell_type": "code",
 "execution_count": null,
 "metadata": {
  "colab": {
   "base_uri": "https://localhost:8080/"
  },
  "id": "Em5WQNLLWCBT",
  "outputId": "24ae1ea9-3438-497b-a0f6-d35ae2124090"
 },
 "outputs": [
  {
   "data": {
    "text/plain": [
     "review      0\n",
```

```
    "sentiment    0\n",
    "dtype: int64"
   ]
  },
  "execution_count": 256,
  "metadata": {},
  "output_type": "execute_result"
 }
],
"source": [
 "df.isnull().sum()  # Here we can see data does not have null values\n"
]
},
{
"cell_type": "code",
"execution_count": null,
"metadata": {
 "colab": {
  "base_uri": "https://localhost:8080/"
 },
 "id": "PMKwp4jjWCMp",
 "outputId": "6dbb7426-ac0c-4209-b24d-53bc3acba285"
},
"outputs": [
 {
  "data": {
   "text/plain": [
    "0"
   ]
  },
  "execution_count": 257,
```

```
  "metadata": {},
  "output_type": "execute_result"
 }
],
"source": [
 "df.duplicated().sum()  # if there are duplicate records which we can drop\n"
]
},
{
"cell_type": "code",
"execution_count": null,
"metadata": {
 "colab": {
  "base_uri": "https://localhost:8080/"
 },
 "id": "7S0ZBAF9jUCt",
 "outputId": "869e6f75-ac38-423d-a38b-caf3a2577560"
},
"outputs": [
 {
  "data": {
   "text/plain": [
    "3000"
   ]
  },
  "execution_count": 261,
  "metadata": {},
  "output_type": "execute_result"
 }
],
"source": [
```

```
 "len(df)\n",
 "# length of data after duplicates removed"
]
},
{
 "cell_type": "code",
 "execution_count": null,
 "metadata": {
  "colab": {
   "base_uri": "https://localhost:8080/"
  },
  "id": "9OGmrugJVT92",
  "outputId": "a4ed26ea-71ee-4f23-d6bf-78c39e5043ce"
 },
 "outputs": [
  {
   "name": "stderr",
   "output_type": "stream",
   "text": [
    "[nltk_data] Downloading package stopwords to /root/nltk_data...\n",
    "[nltk_data]   Package stopwords is already up-to-date!\n"
   ]
  },
  {
   "data": {
    "text/plain": [
     "True"
    ]
   },
   "execution_count": 262,
   "metadata": {},
```

```
  "output_type": "execute_result"
 }
],
"source": [
 "# data cleaning or preprocessing using libraries\n",
 "import re # regular expression library\n",
 "import nltk # text based library\n",
 "nltk.download('stopwords') # downloaded the stopwords library"
]
},
{
"cell_type": "code",
"execution_count": null,
"metadata": {
 "id": "mqp3TZRSV2cb"
},
"outputs": [],
"source": [
 "# imported the libraries stopwords and stemming\n",
 "from nltk.corpus import stopwords\n",
 "from nltk.stem.porter import PorterStemmer"
]
},
{
"cell_type": "code",
"execution_count": null,
"metadata": {
 "id": "cdP0MBHLPj7C"
},
"outputs": [],
"source": [
```

```
  "ps=PorterStemmer() # initalized ps object to the porter stemmer"
 ]
},
{
 "cell_type": "code",
 "execution_count": null,
 "metadata": {
  "id": "fjmaXjCRSAN4"
 },
 "outputs": [],
 "source": [
  "corpus=[]\n",
  "for i in range(0,len(df)):\n",
  "#  print(df['review'][i])\n",
  " review=re.sub('[^a-zA-Z]',\" \",df['review'][i])\n",
  " #print(review)\n",
  " review=review.lower() # lowering all text\n",
  " #print(review)\n",
  " review=review.split() # splitting (converting each sentence into the list of words)\n",
  " #print(\"before stemming\",review[1])\n",
  " review=[ps.stem(word) for word in review if not word in stopwords.words('english')]\n",
  " # Each data in english taken\n",
  " #print(\"after stemming\",review[1])\n",
  " review=\" \".join(review)\n",
  " #print(review)\n",
  " corpus.append(review) # appending to corpus everytime"
 ]
},
{
 "cell_type": "code",
 "execution_count": null,
```

```
  "metadata": {
   "id": "fGDV6CQoeS9q"
  },
  "outputs": [],
  "source": [
   "# bag of words initalizing\n",
   "from sklearn.feature_extraction.text import CountVectorizer\n",
   "cv=CountVectorizer(max_features=3000) #object cv created and maxfeature to 3k to reduced execution time\n",
   "X=cv.fit_transform(corpus).toarray() # transform cv to X via corpus"
  ]
 },
 {
  "cell_type": "code",
  "execution_count": null,
  "metadata": {
   "colab": {
    "base_uri": "https://localhost:8080/"
   },
   "id": "NDr4-CoBBy2f",
   "outputId": "58eb313d-bfc2-4b3e-8dc4-2a48639bd655"
  },
  "outputs": [
   {
    "data": {
     "text/plain": [
      "(3000, 3000)"
     ]
    },
    "execution_count": 267,
    "metadata": {},
```

```
   "output_type": "execute_result"
  }
 ],
 "source": [
  "X.shape\n"
 ]
},
{
 "cell_type": "code",
 "execution_count": null,
 "metadata": {
  "id": "-funM3T_BzJI"
 },
 "outputs": [],
 "source": [
  "y=pd.get_dummies(df['sentiment']) # converting the categorical output variable into numbers"
 ]
},
{
 "cell_type": "code",
 "execution_count": null,
 "metadata": {
  "colab": {
   "base_uri": "https://localhost:8080/",
   "height": 423
  },
  "id": "TZxtIMQCS5jm",
  "outputId": "1bfde403-f0cd-4e65-ece8-ca458a2c93a1"
 },
 "outputs": [
  {
```

```
"data": {
 "text/html": [
  "\n",
  "  <div id=\"df-5be7a2db-5442-4318-af5d-11ca7576ca55\" class=\"colab-df-container\">\n",
  "    <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>negative</th>\n",
  "      <th>positive</th>\n",
  "    </tr>\n",
  "  </thead>\n",
  "  <tbody>\n",
  "    <tr>\n",
  "      <th>0</th>\n",
  "      <td>0</td>\n",
  "      <td>1</td>\n",
```

```
"   </tr>\n",
"   <tr>\n",
"     <th>1</th>\n",
"     <td>0</td>\n",
"     <td>1</td>\n",
"   </tr>\n",
"   <tr>\n",
"     <th>2</th>\n",
"     <td>0</td>\n",
"     <td>1</td>\n",
"   </tr>\n",
"   <tr>\n",
"     <th>3</th>\n",
"     <td>1</td>\n",
"     <td>0</td>\n",
"   </tr>\n",
"   <tr>\n",
"     <th>4</th>\n",
"     <td>0</td>\n",
"     <td>1</td>\n",
"   </tr>\n",
"   <tr>\n",
"     <th>...</th>\n",
"     <td>...</td>\n",
"     <td>...</td>\n",
"   </tr>\n",
"   <tr>\n",
"     <th>2995</th>\n",
"     <td>0</td>\n",
"     <td>1</td>\n",
"   </tr>\n",
```

```
"    <tr>\n",
"      <th>2996</th>\n",
"      <td>1</td>\n",
"      <td>0</td>\n",
"    </tr>\n",
"    <tr>\n",
"      <th>2997</th>\n",
"      <td>0</td>\n",
"      <td>1</td>\n",
"    </tr>\n",
"    <tr>\n",
"      <th>2998</th>\n",
"      <td>1</td>\n",
"      <td>0</td>\n",
"    </tr>\n",
"    <tr>\n",
"      <th>2999</th>\n",
"      <td>1</td>\n",
"      <td>0</td>\n",
"    </tr>\n",
"  </tbody>\n",
"</table>\n",
"<p>3000 rows × 2 columns</p>\n",
"</div>\n",
"    <div class=\"colab-df-buttons\">\n",
"\n",
"  <div class=\"colab-df-container\">\n",
"    <button class=\"colab-df-convert\" onclick=\"convertToInteractive('df-5be7a2db-5442-4318-af5d-11ca7576ca55')\"\n",
"            title=\"Convert this dataframe to an interactive table.\"\n",
"            style=\"display:none;\">\n",
```

```
    "\n",
    "  <svg xmlns=\"http://www.w3.org/2000/svg\" height=\"24px\" viewBox=\"0 -960 960 960\">\n",
    "    <path d=\"M120-120v-720h720v720H120Zm60-500h600v-160H180v160Zm220 220h160v-160H400v160Zm0 220h160v-160H400v160ZM180-400h160v-160H180v160Zm440 0h160v-160H620v160ZM180-180h160v-160H180v160Zm440 0h160v-160H620v160Z\"/>\n",
    "  </svg>\n",
    "    </button>\n",
    "\n",
    "  <style>\n",
    "    .colab-df-container {\n",
    "      display:flex;\n",
    "      gap: 12px;\n",
    "    }\n",
    "\n",
    "    .colab-df-convert {\n",
    "      background-color: #E8F0FE;\n",
    "      border: none;\n",
    "      border-radius: 50%;\n",
    "      cursor: pointer;\n",
    "      display: none;\n",
    "      fill: #1967D2;\n",
    "      height: 32px;\n",
    "      padding: 0 0 0 0;\n",
    "      width: 32px;\n",
    "    }\n",
    "\n",
    "    .colab-df-convert:hover {\n",
    "      background-color: #E2EBFA;\n",
    "      box-shadow: 0px 1px 2px rgba(60, 64, 67, 0.3), 0px 1px 3px 1px rgba(60, 64, 67, 0.15);\n",
    "      fill: #174EA6;\n",
    "    }\n",
```

```
    "\n",
    "    .colab-df-buttons div {\n",
    "      margin-bottom: 4px;\n",
    "    }\n",
    "\n",
    "    [theme=dark] .colab-df-convert {\n",
    "      background-color: #3B4455;\n",
    "      fill: #D2E3FC;\n",
    "    }\n",
    "\n",
    "    [theme=dark] .colab-df-convert:hover {\n",
    "      background-color: #434B5C;\n",
    "      box-shadow: 0px 1px 3px 1px rgba(0, 0, 0, 0.15);\n",
    "      filter: drop-shadow(0px 1px 2px rgba(0, 0, 0, 0.3));\n",
    "      fill: #FFFFFF;\n",
    "    }\n",
    "  </style>\n",
    "\n",
    "    <script>\n",
    "      const buttonEl =\n",
    "        document.querySelector('#df-5be7a2db-5442-4318-af5d-11ca7576ca55 button.colab-df-
convert');\n",
    "      buttonEl.style.display =\n",
    "        google.colab.kernel.accessAllowed ? 'block' : 'none';\n",
    "\n",
    "      async function convertToInteractive(key) {\n",
    "        const element = document.querySelector('#df-5be7a2db-5442-4318-af5d-
11ca7576ca55');\n",
    "        const dataTable =\n",
    "          await google.colab.kernel.invokeFunction('convertToInteractive',\n",
    "                                                   [key], {});\n",
    "        if (!dataTable) return;\n",
```

```
    "\n",
    "       const docLinkHtml = 'Like what you see? Visit the ' +\n",
    "         '<a target=\"_blank\" href=https://colab.research.google.com/notebooks/data_table.ipynb>data table notebook</a>'\n",
    "         + ' to learn more about interactive tables.';\n",
    "       element.innerHTML = '';\n",
    "       dataTable['output_type'] = 'display_data';\n",
    "       await google.colab.output.renderOutput(dataTable, element);\n",
    "       const docLink = document.createElement('div');\n",
    "       docLink.innerHTML = docLinkHtml;\n",
    "       element.appendChild(docLink);\n",
    "     }\n",
    "   </script>\n",
    " </div>\n",
    "\n",
    "\n",
    "<div id=\"df-913fcf41-9d93-442f-94cf-700c81fd7f9c\">\n",
    "  <button class=\"colab-df-quickchart\" onclick=\"quickchart('df-913fcf41-9d93-442f-94cf-700c81fd7f9c')\"\n",
    "          title=\"Suggest charts.\"\n",
    "          style=\"display:none;\">\n",
    "\n",
    "<svg xmlns=\"http://www.w3.org/2000/svg\" height=\"24px\"viewBox=\"0 0 24 24\"\n",
    "     width=\"24px\">\n",
    "    <g>\n",
    "       <path d=\"M19 3H5c-1.1 0-2 .9-2 2v14c0 1.1.9 2 2 2h14c1.1 0 2-.9 2-2V5c0-1.1-.9-2-2-2zM9 17H7v-7h2v7zm4 0h-2V7h2v10zm4 0h-2v-4h2v4z\"/>\n",
    "   </g>\n",
    "</svg>\n",
    "  </button>\n",
    "\n",
    "<style>\n",
```

```
"  .colab-df-quickchart {\n",
"      --bg-color: #E8F0FE;\n",
"      --fill-color: #1967D2;\n",
"      --hover-bg-color: #E2EBFA;\n",
"      --hover-fill-color: #174EA6;\n",
"      --disabled-fill-color: #AAA;\n",
"      --disabled-bg-color: #DDD;\n",
"  }\n",
"\n",
"  [theme=dark] .colab-df-quickchart {\n",
"      --bg-color: #3B4455;\n",
"      --fill-color: #D2E3FC;\n",
"      --hover-bg-color: #434B5C;\n",
"      --hover-fill-color: #FFFFFF;\n",
"      --disabled-bg-color: #3B4455;\n",
"      --disabled-fill-color: #666;\n",
"  }\n",
"\n",
"  .colab-df-quickchart {\n",
"    background-color: var(--bg-color);\n",
"    border: none;\n",
"    border-radius: 50%;\n",
"    cursor: pointer;\n",
"    display: none;\n",
"    fill: var(--fill-color);\n",
"    height: 32px;\n",
"    padding: 0;\n",
"    width: 32px;\n",
"  }\n",
"\n",
"  .colab-df-quickchart:hover {\n",
```

```
"    background-color: var(--hover-bg-color);\n",
"    box-shadow: 0 1px 2px rgba(60, 64, 67, 0.3), 0 1px 3px 1px rgba(60, 64, 67, 0.15);\n",
"    fill: var(--button-hover-fill-color);\n",
"  }\n",
"\n",
"  .colab-df-quickchart-complete:disabled,\n",
"  .colab-df-quickchart-complete:disabled:hover {\n",
"    background-color: var(--disabled-bg-color);\n",
"    fill: var(--disabled-fill-color);\n",
"    box-shadow: none;\n",
"  }\n",
"\n",
"  .colab-df-spinner {\n",
"    border: 2px solid var(--fill-color);\n",
"    border-color: transparent;\n",
"    border-bottom-color: var(--fill-color);\n",
"    animation:\n",
"      spin 1s steps(1) infinite;\n",
"  }\n",
"\n",
"  @keyframes spin {\n",
"    0% {\n",
"      border-color: transparent;\n",
"      border-bottom-color: var(--fill-color);\n",
"      border-left-color: var(--fill-color);\n",
"    }\n",
"    20% {\n",
"      border-color: transparent;\n",
"      border-left-color: var(--fill-color);\n",
"      border-top-color: var(--fill-color);\n",
"    }\n",
```

```
"    30% {\n",
"      border-color: transparent;\n",
"      border-left-color: var(--fill-color);\n",
"      border-top-color: var(--fill-color);\n",
"      border-right-color: var(--fill-color);\n",
"    }\n",
"    40% {\n",
"      border-color: transparent;\n",
"      border-right-color: var(--fill-color);\n",
"      border-top-color: var(--fill-color);\n",
"    }\n",
"    60% {\n",
"      border-color: transparent;\n",
"      border-right-color: var(--fill-color);\n",
"    }\n",
"    80% {\n",
"      border-color: transparent;\n",
"      border-right-color: var(--fill-color);\n",
"      border-bottom-color: var(--fill-color);\n",
"    }\n",
"    90% {\n",
"      border-color: transparent;\n",
"      border-bottom-color: var(--fill-color);\n",
"    }\n",
"  }\n",
"</style>\n",
"\n",
"  <script>\n",
"    async function quickchart(key) {\n",
"      const quickchartButtonEl =\n",
"        document.querySelector('#' + key + ' button');\n",
```

```
"     quickchartButtonEl.disabled = true;  // To prevent multiple clicks.\n",
"     quickchartButtonEl.classList.add('colab-df-spinner');\n",
"     try {\n",
"       const charts = await google.colab.kernel.invokeFunction(\n",
"           'suggestCharts', [key], {});\n",
"     } catch (error) {\n",
"       console.error('Error during call to suggestCharts:', error);\n",
"     }\n",
"     quickchartButtonEl.classList.remove('colab-df-spinner');\n",
"     quickchartButtonEl.classList.add('colab-df-quickchart-complete');\n",
"    }\n",
"    (() => {\n",
"     let quickchartButtonEl =\n",
"       document.querySelector('#df-913fcf41-9d93-442f-94cf-700c81fd7f9c button');\n",
"     quickchartButtonEl.style.display =\n",
"       google.colab.kernel.accessAllowed ? 'block' : 'none';\n",
"    })();\n",
"  </script>\n",
"</div>\n",
"    </div>\n",
"  </div>\n"
],
"text/plain": [
"     negative  positive\n",
"0          0       1\n",
"1          0       1\n",
"2          0       1\n",
"3          1       0\n",
"4          0       1\n",
"...      ...      ...\n",
"2995       0       1\n",
```

```
    "2996        1        0\n",
    "2997        0        1\n",
    "2998        1        0\n",
    "2999        1        0\n",
    "\n",
    "[3000 rows x 2 columns]"
   ]
  },
  "execution_count": 269,
  "metadata": {},
  "output_type": "execute_result"
 }
],
"source": [
 "y"
]
},
{
 "cell_type": "code",
 "execution_count": null,
 "metadata": {
  "id": "XeQt4EQoBza3"
 },
 "outputs": [],
 "source": [
  "y=y.iloc[:,1].values"
 ]
},
{
 "cell_type": "code",
 "execution_count": null,
```

```
"metadata": {
 "colab": {
  "base_uri": "https://localhost:8080/"
 },
 "id": "Zw3Ct_XFM83y",
 "outputId": "0bd1cd22-e021-4687-843f-aa7124b67f58"
},
"outputs": [
 {
  "data": {
   "text/plain": [
    "array([1, 1, 1, ..., 1, 0, 0], dtype=uint8)"
   ]
  },
  "execution_count": 271,
  "metadata": {},
  "output_type": "execute_result"
 }
],
"source": [
 "y"
]
},
{
 "cell_type": "markdown",
 "metadata": {
  "id": "43N7Pi-mWNVL"
 },
 "source": [
  "*need only one column of both pos and neg"
 ]
```

```
},
{
 "cell_type": "code",
 "execution_count": null,
 "metadata": {
  "id": "5tFYm0PkBzce"
 },
 "outputs": [],
 "source": [
  "# train and test split\n",
  "from sklearn.model_selection import train_test_split\n",
  "X_train,X_test,y_train,y_test= train_test_split(X,y,test_size=.20,random_state=0)"
 ]
},
{
 "cell_type": "code",
 "execution_count": null,
 "metadata": {
  "id": "agHqcyx9vj1X"
 },
 "outputs": [],
 "source": [
  "# training model using multinomial navie bayes algotithm\n",
  "from sklearn.naive_bayes import MultinomialNB\n",
  "spam_model=MultinomialNB().fit(X_train,y_train)"
 ]
},
{
 "cell_type": "code",
 "execution_count": null,
 "metadata": {
```

```
 "id": "JHWL5NZECPG6"
},
"outputs": [],
"source": [
 "#predict the model with test data\n",
 "y_pred=spam_model.predict(X_test)"
]
},
{
"cell_type": "code",
"execution_count": null,
"metadata": {
 "colab": {
  "base_uri": "https://localhost:8080/"
 },
 "id": "QOzJzOmnCPed",
 "outputId": "54c588d3-7ce7-4afc-8753-dd2ede57d78f"
},
"outputs": [
 {
  "data": {
   "text/plain": [
    "0.8233333333333334"
   ]
  },
  "execution_count": 275,
  "metadata": {},
  "output_type": "execute_result"
 }
],
"source": [
```

```
 "# check the performance\n",
 "from sklearn.metrics import accuracy_score\n",
 "accuracy_score(y_test,y_pred)"
]
},
{
 "cell_type": "code",
 "execution_count": null,
 "metadata": {
  "colab": {
   "base_uri": "https://localhost:8080/"
  },
  "id": "bAbSSn8ZC72C",
  "outputId": "9498e9f9-b8a0-4ebb-c40e-4267e7b4fd0d"
 },
 "outputs": [
  {
   "data": {
    "text/plain": [
     "array([[251,  55],\n",
     "       [ 51, 243]])"
    ]
   },
   "execution_count": 276,
   "metadata": {},
   "output_type": "execute_result"
  }
 ],
 "source": [
  "# check confusion matrix to true positive and negatives\n",
  "from sklearn.metrics import confusion_matrix\n",
```

```
  "confusion_matrix(y_test,y_pred)"
 ]
},
{
 "cell_type": "code",
 "execution_count": null,
 "metadata": {
  "colab": {
   "base_uri": "https://localhost:8080/"
  },
  "id": "OIbqSzd1C79K",
  "outputId": "c967b8e8-a51c-47de-b6c6-3abebb1b4fbe"
 },
 "outputs": [
  {
   "name": "stderr",
   "output_type": "stream",
   "text": [
    "[nltk_data] Downloading package wordnet to /root/nltk_data...\n",
    "[nltk_data]   Package wordnet is already up-to-date!\n"
   ]
  }
 ],
 "source": [
  "# lemmatization\n",
  "from nltk.stem import WordNetLemmatizer\n",
  "nltk.download('wordnet')\n",
  "ln=WordNetLemmatizer()"
 ]
},
{
```

```
"cell_type": "code",
"execution_count": null,
"metadata": {
 "id": "U9yD-PsmENaI"
},
"outputs": [],
"source": [
 "# reading the imdb movies review and store into df again\n",
 "df=pd.read_csv('/content/IMDB_Dataset.csv',)"
]
},
{
"cell_type": "code",
"execution_count": null,
"metadata": {
 "colab": {
  "base_uri": "https://localhost:8080/"
 },
 "id": "LUTG5cPBGuHU",
 "outputId": "c99ba6b8-93b8-40ae-e795-503a540565c6"
},
"outputs": [
 {
  "data": {
   "text/plain": [
    "(3000, 2)"
   ]
  },
  "execution_count": 279,
  "metadata": {},
  "output_type": "execute_result"
```

```
   }
  ],
  "source": [
   "df = df[0:3000]  # with 50000 records we are having resource limitations hence taken only 3000 rows\n",
   "df.shape"
  ]
 },
 {
  "cell_type": "code",
  "execution_count": null,
  "metadata": {
   "colab": {
    "base_uri": "https://localhost:8080/"
   },
   "id": "R-xbkI2lGL1s",
   "outputId": "af9bc3af-e9c7-4e49-87f3-9f8c079e7284"
  },
  "outputs": [
   {
    "data": {
     "text/plain": [
      "0"
     ]
    },
    "execution_count": 280,
    "metadata": {},
    "output_type": "execute_result"
   }
  ],
  "source": [
```

```
   "df.duplicated().sum()  # there are no duplicate records which we can drop\n"
  ]
 },
 {
  "cell_type": "code",
  "execution_count": null,
  "metadata": {
   "id": "U2uc7XMBC8AO"
  },
  "outputs": [],
  "source": [
   "corpus=[]\n",
   "for i in range(0,len(df)):\n",
   "  #print()\n",
   "  review=re.sub('[^a-zA-Z]',\" \",df['review'][i]) # we removed all the numbers and special characters only allowed alphabets\n",
   "  #print(review)\n",
   "  review=review.lower() # lowering all text\n",
   "  review= review.split() # splitting (converting each sentence into the list of words)\n",
   "  #print(\"before lemmatizing\",review[1])\n",
   "  review=[ln.lemmatize(word) for word in review if not word in stopwords.words('english')]\n",
   "  #print(\"after lemmatizing\",review[1])\n",
   "  review=\" \".join(review)\n",
   "  #print(review)\n",
   "  corpus.append(review)"
  ]
 },
 {
  "cell_type": "code",
  "execution_count": null,
  "metadata": {
```

```
 "id": "ynEcUs38C8Dp"
},
"outputs": [],
"source": [
 "# bag of words initialising\n",
 "from sklearn.feature_extraction.text import CountVectorizer\n",
 "cv=CountVectorizer(max_features=3000)\n",
 "X=cv.fit_transform(corpus).toarray() # transform cv to X via corpus"
]
},
{
"cell_type": "code",
"execution_count": null,
"metadata": {
 "id": "cDJ52y7tDTD7"
},
"outputs": [],
"source": [
 "y=pd.get_dummies(df['sentiment']) # converting the categorical output variable into numbers\n",
 "y=y.iloc[:,1].values"
]
},
{
"cell_type": "code",
"execution_count": null,
"metadata": {
 "id": "A4_mh3K_DWnP"
},
"outputs": [],
"source": [
 "# train and test split\n",
```

```
 "from sklearn.model_selection import train_test_split\n",
 "X_train,X_test,y_train,y_test= train_test_split(X,y,test_size=.20,random_state=0)"
]
},
{
 "cell_type": "code",
 "execution_count": null,
 "metadata": {
  "id": "4gySBq68DWzm"
 },
 "outputs": [],
 "source": [
  "#predict the model with test data\n",
  "y_pred=spam_model.predict(X_test)"
 ]
},
{
 "cell_type": "code",
 "execution_count": null,
 "metadata": {
  "colab": {
   "base_uri": "https://localhost:8080/"
  },
  "id": "7R6ZAv8EDapO",
  "outputId": "641d2a2a-245e-44f6-ba36-426f23b1d1ed"
 },
 "outputs": [
  {
   "data": {
    "text/plain": [
     "0.4533333333333333"
```

```
   ]
  },
  "execution_count": 290,
  "metadata": {},
  "output_type": "execute_result"
 }
],
"source": [
 "# check the performance\n",
 "from sklearn.metrics import accuracy_score\n",
 "accuracy_score(y_test,y_pred)"
]
},
{
 "cell_type": "code",
 "execution_count": null,
 "metadata": {
  "id": "Up250pRaFemE"
 },
 "outputs": [],
 "source": [
  "#tf idf on stemming and lemmatization using Bayes algorithm"
 ]
},
{
 "cell_type": "code",
 "execution_count": null,
 "metadata": {
  "id": "ZcNxn3C1b7rD"
 },
 "outputs": [],
```

```
"source": [
 "corpus=[]\n",
 "for i in range(0,len(df)):\n",
 "#  print(df['review'][i])\n",
 " review=re.sub('[^a-zA-Z]',\" \",df['review'][i])\n",
 " #print(review)\n",
 " review=review.lower() # lowering all text\n",
 " #print(review)\n",
 " review=review.split() # splitting (converting each sentence into the list of words)\n",
 " #print(\"before stemming\",review[1])\n",
 " review=[ps.stem(word) for word in review if not word in stopwords.words('english')]\n",
 " # Each data in english taken\n",
 " #print(\"after stemming\",review[1])\n",
 " review=\" \".join(review)\n",
 " #print(review)\n",
 " corpus.append(review) # appending to corpus everytime"
]
},
{
 "cell_type": "code",
 "execution_count": null,
 "metadata": {
  "id": "GLNl1RNSHU_3"
 },
 "outputs": [],
 "source": [
  "# TF-IDf iniatializing\n",
  "from sklearn.feature_extraction.text import TfidfVectorizer\n",
  "tf=TfidfVectorizer(max_features=3000) # tf object creating\n",
  "X=tf.fit_transform(corpus).toarray()  # transform tf to X via corpus"
 ]
```

```
},
{
 "cell_type": "code",
 "execution_count": null,
 "metadata": {
  "id": "0-vJVVCZJAk8"
 },
 "outputs": [],
 "source": [
  "y=pd.get_dummies(df['sentiment']) # converting the categorical output variable into numbers\n",
  "y=y.iloc[:,1].values"
 ]
},
{
 "cell_type": "code",
 "execution_count": null,
 "metadata": {
  "id": "346_IpbBHVFL"
 },
 "outputs": [],
 "source": [
  "# train and test split\n",
  "from sklearn.model_selection import train_test_split\n",
  "X_train,X_test,y_train,y_test= train_test_split(X,y,test_size=.20,random_state=0)"
 ]
},
{
 "cell_type": "code",
 "execution_count": null,
 "metadata": {
  "id": "KZKvQSX2HVJ2"
```

```
 },
 "outputs": [],
 "source": [
  "#predict the model with test data\n",
  "y_pred=spam_model.predict(X_test)"
 ]
},
{
 "cell_type": "code",
 "execution_count": null,
 "metadata": {
  "colab": {
   "base_uri": "https://localhost:8080/"
  },
  "id": "fcl74m6uJ3g0",
  "outputId": "59b4c30a-57ab-4107-f99c-d8615c60a1f9"
 },
 "outputs": [
  {
   "data": {
    "text/plain": [
     "0.8083333333333333"
    ]
   },
   "execution_count": 298,
   "metadata": {},
   "output_type": "execute_result"
  }
 ],
 "source": [
  "# check the performance\n",
```

```
 "from sklearn.metrics import accuracy_score\n",
 "accuracy_score(y_test,y_pred)"
]
},
{
 "cell_type": "code",
 "execution_count": null,
 "metadata": {
  "colab": {
   "base_uri": "https://localhost:8080/"
  },
  "id": "BGIm2aiFYJLU",
  "outputId": "3ef662dc-9590-49cc-9a43-573bf243217d"
 },
 "outputs": [
  {
   "data": {
    "text/plain": [
     "array([[236,  70],\n",
     "       [ 45, 249]])"
    ]
   },
   "execution_count": 299,
   "metadata": {},
   "output_type": "execute_result"
  }
 ],
 "source": [
  "# check confusion matrix to true positive and negatives\n",
  "from sklearn.metrics import confusion_matrix\n",
  "confusion_matrix(y_test,y_pred)"
```

```
 ]
},
{
 "cell_type": "code",
 "execution_count": null,
 "metadata": {
  "id": "Xqc4yW0BgxGC"
 },
 "outputs": [],
 "source": [
  "corpus=[]\n",
  "for i in range(0,len(df)):\n",
  "  #print()\n",
  "  review=re.sub('[^a-zA-Z]',\" \",df['review'][i]) # we removed all the numbers and special characters only allowed alphabets\n",
  "  #print(review)\n",
  "  review=review.lower() # lowering all text\n",
  "  review= review.split() # splitting (converting each sentence into the list of words)\n",
  "  #print(\"before lemmatizing\",review[1])\n",
  "  review=[ln.lemmatize(word) for word in review if not word in stopwords.words('english')]\n",
  "  #print(\"after lemmatizing\",review[1])\n",
  "  review=\" \".join(review)\n",
  "  #print(review)\n",
  "  corpus.append(review)"
 ]
},
{
 "cell_type": "code",
 "execution_count": null,
 "metadata": {
  "id": "Gjz4FI2KgxgJ"
```

```
 },
 "outputs": [],
 "source": [
  "# TF-IDf iniatializing\n",
  "from sklearn.feature_extraction.text import TfidfVectorizer\n",
  "tf=TfidfVectorizer(max_features=3000) # tf object creating\n",
  "X=tf.fit_transform(corpus).toarray()  # transform tf to X via corpus"
 ]
},
{
 "cell_type": "code",
 "execution_count": null,
 "metadata": {
  "id": "gQw7UuqSgxmW"
 },
 "outputs": [],
 "source": [
  "# train and test split\n",
  "from sklearn.model_selection import train_test_split\n",
  "X_train,X_test,y_train,y_test= train_test_split(X,y,test_size=.20,random_state=0)"
 ]
},
{
 "cell_type": "code",
 "execution_count": null,
 "metadata": {
  "id": "igb7ynn-gxpU"
 },
 "outputs": [],
 "source": [
  "#predict the model with test data\n",
```

```
  "y_pred=spam_model.predict(X_test)"
 ]
},
{
 "cell_type": "code",
 "execution_count": null,
 "metadata": {
  "colab": {
   "base_uri": "https://localhost:8080/"
  },
  "id": "5DXbNF3FgxsP",
  "outputId": "8691cec7-dfe3-4d1e-897b-fd369ba35f1a"
 },
 "outputs": [
  {
   "data": {
    "text/plain": [
     "0.4666666666666667"
    ]
   },
   "execution_count": 304,
   "metadata": {},
   "output_type": "execute_result"
  }
 ],
 "source": [
  "# check the performance\n",
  "from sklearn.metrics import accuracy_score\n",
  "accuracy_score(y_test,y_pred)"
 ]
},
```

```
{
 "cell_type": "code",
 "execution_count": null,
 "metadata": {
  "colab": {
   "base_uri": "https://localhost:8080/"
  },
  "id": "jwDOqrotgxu7",
  "outputId": "3589afc3-ecca-4085-d267-d5e28ad0a739"
 },
 "outputs": [
  {
   "data": {
    "text/plain": [
     "array([[ 99, 207],\n",
     "       [113, 181]])"
    ]
   },
   "execution_count": 305,
   "metadata": {},
   "output_type": "execute_result"
  }
 ],
 "source": [
  "# check confusion matrix to true positive and negatives\n",
  "from sklearn.metrics import confusion_matrix\n",
  "confusion_matrix(y_test,y_pred)"
 ]
}
],
"metadata": {
```

```
 "colab": {
  "provenance": []
 },
 "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": 1
}
```