```
{
  "nbformat": 4,
  "nbformat_minor": 0,
  "metadata": {
    "colab": {
      "provenance": [],
      "collapsed_sections": []
    },
    "kernelspec": {
      "name": "python3",
      "display_name": "Python 3"
    },
    "language_info": {
      "name": "python"
    }
  },
  "cells": [
    {
      "cell_type": "markdown",
      "source": [
        "**Import the necessary libraries**"
      ],
      "metadata": {
        "id": "jw2aMqbc-wJv"
      }
    },
    {
      "cell_type": "code",
      "execution_count": 20,
      "metadata": {
        "id": "K77mW1n6-vFR"
      },
      "outputs": [],
      "source": [
        "import pandas as pd\n",
        "import numpy as np\n",
        "import matplotlib.pyplot as plt\n",
        "import seaborn as sns\n",
        "from sklearn.model_selection import train_test_split\n",
        "from sklearn.preprocessing import LabelEncoder\n",
        "from keras.models import Model\n",
        "from keras.layers import LSTM, Activation, Dense, Dropout, 
Input, Embedding\n",
        "from keras.optimizers import RMSprop\n",
        "from keras.preprocessing.text import Tokenizer\n",
        "from keras.preprocessing import sequence\n",
        "from keras.utils import pad_sequences\n",
        "from keras.utils import to_categorical\n",
        "from keras.callbacks import EarlyStopping"
      ]
    },
    {
      "cell_type": "markdown",
      "source": [
        "**Download the Dataset**"
      ],
      "metadata": {
        "id": "H1DO9j68_RI0"
      }
    },
    {
      "cell_type": "markdown",
      "source": [
        "Dataset Downloaded and uploaded to drive\n",
        "https://www.kaggle.com/code/kredy10/simple-lstm-for-text-
classification/data"
      ],
      "metadata": {
        "id": "4eXV8hsFHAqm"
      }
    },
    {
      "cell_type": "markdown",
      "source": [
        "**Read dataset and do pre-processing**"
      ],
      "metadata": {
        "id": "C6FrJHoSDOb5"
      }
    },
    {
      "cell_type": "markdown",
      "source": [
        "Read dataset"
      ],
      "metadata": {
        "id": "hz0U8RtoDY_U"
      }
    },
    {
      "cell_type": "code",
      "source": [
        "df =
pd.read_csv('/content/drive/MyDrive/spam.csv',delimiter=',',encoding='lat
in-1')\n",
        "df.head()"
      ],
      "metadata": {
        "colab": {
          "base_uri": "https://localhost:8080/",
          "height": 206
        },
        "id": "oYGNw8ok_sjy",
        "outputId": "f95810f2-1908-4007-dc0d-9ffa643904da"
      },
      "execution_count": 21,
      "outputs": [
        {
          "output_type": "execute_result",
          "data": {
            "text/plain": [
              "     v1                                                v2
Unnamed: 2  \\\n",
              "0   ham  Go until jurong point, crazy.. Available only ...
NaN   \n",
              "1   ham                        Ok lar... Joking wif u oni...
NaN   \n",
              "2  spam  Free entry in 2 a wkly comp to win FA Cup fina...
NaN   \n",
              "3   ham  U dun say so early hor... U c already then say...
NaN   \n",
              "4   ham  Nah I don't think he goes to usf, he lives aro...
NaN   \n",
              "\n",
              "  Unnamed: 3 Unnamed: 4  \n",
              "0        NaN        NaN  \n",
              "1        NaN        NaN  \n",
              "2        NaN        NaN  \n",
              "3        NaN        NaN  \n",
              "4        NaN        NaN  "
            ],
            "text/html": [
              "\n",
              "  <div id=\"df-028b22c9-38b1-42c3-b51d-d45b0b743d3a\">\n",
              "    <div 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>v1</th>\n",
              "      <th>v2</th>\n",
              "      <th>Unnamed: 2</th>\n",
              "      <th>Unnamed: 3</th>\n",
              "      <th>Unnamed: 4</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",
              "      <td>NaN</td>\n",
              "      <td>NaN</td>\n",
              "      <td>NaN</td>\n",
              "    </tr>\n",
              "    <tr>\n",
              "      <th>1</th>\n",
              "      <td>ham</td>\n",
            "      <td>Ok lar... Joking wif u oni...</td>\n",
            "      <td>NaN</td>\n",
            "      <td>NaN</td>\n",
            "      <td>NaN</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",
            "      <td>NaN</td>\n",
            "      <td>NaN</td>\n",
            "      <td>NaN</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",
            "      <td>NaN</td>\n",
            "      <td>NaN</td>\n",
            "      <td>NaN</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",
            "      <td>NaN</td>\n",
            "      <td>NaN</td>\n",
            "      <td>NaN</td>\n",
            "    </tr>\n",
            "  </tbody>\n",
            "</table>\n",
            "</div>\n",
            "      <button class=\"colab-df-convert\" 
onclick=\"convertToInteractive('df-028b22c9-38b1-42c3-b51d-
d45b0b743d3a')\"\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 0 24 24\"\n",
            "       width=\"24px\">\n",
            "    <path d=\"M0 0h24v24H0V0z\" fill=\"none\"/>\n",
            "    <path d=\"M18.56 5.44l.94 2.06.94-2.06 2.06-.94-2.06-
.94-.94-2.06-.94 2.06-2.06.94zm-11 1L8.5 8.5l.94-2.06 2.06-.94-2.06-
.94L8.5 2.5l-.94 2.06-2.06.94zm10 10l.94 2.06.94-2.06 2.06-.94-2.06-.94-
.94-2.06-.94 2.06-2.06.94z\"/><path d=\"M17.41 7.96l-1.37-1.37c-.4-.4-
.92-.59-1.43-.59-.52 0-1.04.2-1.43.59L10.3 9.45l-7.72 7.72c-.78.78-.78 
2.05 0 2.83L4 21.41c.39.39.9.59 1.41.59.51 0 1.02-.2 1.41-.59l7.78-7.78 
2.81-2.81c.8-.78.8-2.07 0-2.86zM5.41 20L4 18.59l7.72-7.72 1.47 1.35L5.41 
20z\"/>\n",
            "  </svg>\n",
            "      </button>\n",
            "      \n",
            "  <style>\n",
              "    .colab-df-container {\n",
              "      display:flex;\n",
              "      flex-wrap:wrap;\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",
              "    [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-028b22c9-38b1-42c3-
b51d-d45b0b743d3a 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-
028b22c9-38b1-42c3-b51d-d45b0b743d3a');\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",
              "  </div>\n",
              "  "
            ]
          },
          "metadata": {},
          "execution_count": 21
        }
      ]
    },
    {
      "cell_type": "markdown",
      "source": [
        "Preprocessing the Dataset"
      ],
      "metadata": {
        "id": "61IHxF1NDkh2"
      }
    },
    {
      "cell_type": "code",
      "source": [
        "df.drop(['Unnamed: 2', 'Unnamed: 3', 'Unnamed:
4'],axis=1,inplace=True)\n",
        "df.info()"
      ],
      "metadata": {
        "colab": {
          "base_uri": "https://localhost:8080/"
        },
        "id": "46xw0pCwDibS",
        "outputId": "df4efcad-7eee-4a9a-d00c-9f475187b7a2"
      },
      "execution_count": 22,
      "outputs": [
        {
          "output_type": "stream",
          "name": "stdout",
          "text": [
            "<class 'pandas.core.frame.DataFrame'>\n",
            "RangeIndex: 5572 entries, 0 to 5571\n",
            "Data columns (total 2 columns):\n",
            " #   Column  Non-Null Count  Dtype \n",
            "---  ------  --------------  ----- \n",
            " 0   v1      5572 non-null   object\n",
            " 1   v2      5572 non-null   object\n",
            "dtypes: object(2)\n",
            "memory usage: 87.2+ KB\n"
          ]
        }
      ]
    },
    {
      "cell_type": "code",
      "source": [
        "X = df.v2\n",
        "Y = df.v1\n",
        "le = LabelEncoder()\n",
        "Y = le.fit_transform(Y)\n",
        "Y = Y.reshape(-1,1)"
      ],
      "metadata": {
        "id": "SJyPaFp3DrKG"
      },
      "execution_count": 23,
      "outputs": []
    },
    {
      "cell_type": "code",
      "source": [
        "X_train,X_test,Y_train,Y_test =
train_test_split(X,Y,test_size=0.15)"
      ],
      "metadata": {
        "id": "UTIBHKccDtkN"
      },
      "execution_count": 24,
      "outputs": []
    },
    {
      "cell_type": "code",
      "source": [
        "max_words = 1000\n",
        "max_len = 150\n",
        "tok = Tokenizer(num_words=max_words)\n",
        "tok.fit_on_texts(X_train)\n",
        "sequences = tok.texts_to_sequences(X_train)\n",
        "sequences_matrix = pad_sequences(sequences,maxlen=max_len)"
      ],
      "metadata": {
        "id": "Yu_EY1VsDvuu"
      },
      "execution_count": 25,
      "outputs": []
    },
    {
      "cell_type": "markdown",
      "source": [
        "**Create Model and Add Layers (LSTM, Dense-(Hidden Layers), 
Output)**"
      ],
      "metadata": {
        "id": "e5q3TWuKD1V5"
      }
    },
    {
      "cell_type": "code",
      "source": [
        "inputs = Input(name='inputs',shape=[max_len])\n",
        "layer = Embedding(max_words,50,input_length=max_len)(inputs)\n",
        "layer = LSTM(64)(layer)\n",
        "layer = Dense(256,name='FC1')(layer)\n",
        "layer = Activation('relu')(layer)\n",
        "layer = Dropout(0.5)(layer)\n",
        "layer = Dense(1,name='out_layer')(layer)\n",
        "layer = Activation('sigmoid')(layer)\n",
        "model = Model(inputs=inputs,outputs=layer)\n",
        "\n",
        "model.summary()"
      ],
      "metadata": {
        "colab": {
          "base_uri": "https://localhost:8080/"
        },
        "id": "QDvxp0K8Dxsw",
        "outputId": "352033f4-0b19-47aa-9e56-ae29a5cd3ead"
      },
      "execution_count": 26,
      "outputs": [
        {
          "output_type": "stream",
          "name": "stdout",
          "text": [
            "Model: \"model_1\"\n",

"_________________________________________________________________\n",
            " Layer (type)                Output Shape              Param 
#   \n",

"=================================================================\n",
            " inputs (InputLayer)         [(None, 150)]             0         
\n",
            "                                                                 
\n",
            " embedding_1 (Embedding)     (None, 150, 50)           50000     
\n",
            "                                                                 
\n",
            " lstm_1 (LSTM)               (None, 64)                29440     
\n",
            "                                                                 
\n",
            " FC1 (Dense)                 (None, 256)               16640     
\n",
            "                                                                 
\n",
            " activation_2 (Activation)   (None, 256)               0
\n",
            "
\n",
            " dropout_1 (Dropout)         (None, 256)               0
\n",
            "
\n",
            " out_layer (Dense)           (None, 1)                 257
\n",
            "
\n",
            " activation_3 (Activation)   (None, 1)                 0
\n",
            "
\n",

"=================================================================\n",
            "Total params: 96,337\n",
            "Trainable params: 96,337\n",
            "Non-trainable params: 0\n",

"_________________________________________________________________\n"
          ]
        }
      ]
    },
    {
      "cell_type": "markdown",
      "source": [
        "**Compile the Model**"
      ],
      "metadata": {
        "id": "q0j3QGMLEPWr"
      }
    },
    {
      "cell_type": "code",
      "source": [

"model.compile(loss='binary_crossentropy',optimizer=RMSprop(),metrics=['a
ccuracy'])"
      ],
      "metadata": {
        "id": "2LhDB_DTEKSU"
      },
      "execution_count": 27,
      "outputs": []
    },
    {
      "cell_type": "markdown",
      "source": [
        "**Train and Fit the Model**"
      ],
      "metadata": {
        "id": "E867FGpHEYW3"
      }
    },
    {
      "cell_type": "code",
      "source": [
        "model.fit(sequences_matrix,Y_train,batch_size=128,epochs=10,\n",
        "          validation_split=0.2)"
      ],
      "metadata": {
        "colab": {
          "base_uri": "https://localhost:8080/"
        },
        "id": "7T67tLLbETts",
        "outputId": "4bfc6b3a-6f2a-4f8a-e300-cbbcafc63d45"
      },
      "execution_count": 28,
      "outputs": [
        {
          "output_type": "stream",
          "name": "stdout",
          "text": [
            "Epoch 1/10\n",
            "30/30 [==============================] - 10s 264ms/step -
loss: 0.3182 - accuracy: 0.8788 - val_loss: 0.1571 - val_accuracy:
0.9715\n",
            "Epoch 2/10\n",
            "30/30 [==============================] - 7s 247ms/step -
loss: 0.0805 - accuracy: 0.9786 - val_loss: 0.0742 - val_accuracy:
0.9778\n",
            "Epoch 3/10\n",
            "30/30 [==============================] - 7s 237ms/step -
loss: 0.0403 - accuracy: 0.9881 - val_loss: 0.0670 - val_accuracy:
0.9821\n",
            "Epoch 4/10\n",
            "30/30 [==============================] - 7s 245ms/step -
loss: 0.0272 - accuracy: 0.9929 - val_loss: 0.0806 - val_accuracy:
0.9778\n",
            "Epoch 5/10\n",
            "30/30 [==============================] - 7s 242ms/step -
loss: 0.0220 - accuracy: 0.9937 - val_loss: 0.0820 - val_accuracy:
0.9800\n",
            "Epoch 6/10\n",
            "30/30 [==============================] - 7s 240ms/step -
loss: 0.0178 - accuracy: 0.9955 - val_loss: 0.0787 - val_accuracy:
0.9789\n",
            "Epoch 7/10\n",
            "30/30 [==============================] - 7s 243ms/step -
loss: 0.0150 - accuracy: 0.9958 - val_loss: 0.0969 - val_accuracy:
0.9800\n",
            "Epoch 8/10\n",
            "30/30 [==============================] - 7s 241ms/step -
loss: 0.0162 - accuracy: 0.9958 - val_loss: 0.0901 - val_accuracy:
0.9768\n",
            "Epoch 9/10\n",
            "30/30 [==============================] - 7s 246ms/step -
loss: 0.0099 - accuracy: 0.9968 - val_loss: 0.1284 - val_accuracy:
0.9789\n",
            "Epoch 10/10\n",
            "30/30 [==============================] - 7s 247ms/step -
loss: 0.0355 - accuracy: 0.9905 - val_loss: 0.1264 - val_accuracy:
0.9726\n"
          ]
        },
        {
          "output_type": "execute_result",
          "data": {
            "text/plain": [
              "<keras.callbacks.History at 0x7fe4e3d6a4d0>"
            ]
          },
          "metadata": {},
          "execution_count": 28
        }
      ]
    },
    {
      "cell_type": "markdown",
      "source": [
        "**Save The Model**"
      ],
      "metadata": {
        "id": "lMPVOjDyE4cR"
      }
    },
    {
      "cell_type": "code",
      "source": [
        "model.save('sms_classifier.h5')"
      ],
      "metadata": {
        "id": "AUKPE7vAEoFD"
      },
      "execution_count": 29,
      "outputs": []
    },
    {
      "cell_type": "markdown",
      "source": [
        "**Preprocessing the Test Dataset**"
      ],
      "metadata": {
        "id": "UzPGuTYkH2p8"
      }
    },
    {
      "cell_type": "code",
      "source": [
        "test_sequences = tok.texts_to_sequences(X_test)\n",
        "test_sequences_matrix =
pad_sequences(test_sequences,maxlen=max_len)"
      ],
      "metadata": {
        "id": "sRbKOsA0FATn"
      },
      "execution_count": 30,
      "outputs": []
    },
    {
      "cell_type": "markdown",
      "source": [
        "**Testing the Model**"
      ],
      "metadata": {
        "id": "aAGE1gxpH9GW"
      }
    },
    {
      "cell_type": "code",
      "source": [
        "accr = model.evaluate(test_sequences_matrix,Y_test)"
      ],
      "metadata": {
        "colab": {
          "base_uri": "https://localhost:8080/"
        },
        "id": "HWiyIj56H_df",
        "outputId": "a278f75e-9cfe-4155-94bd-42a8cf4931b4"
      },
      "execution_count": 31,
      "outputs": [
        {
          "output_type": "stream",
          "name": "stdout",
          "text": [
            "27/27 [==============================] - 1s 20ms/step -
loss: 0.0886 - accuracy: 0.9821\n"
          ]
        }
      ]
    },
    {
      "cell_type": "code",
      "source": [
        "print('Test set\\n  Loss: {:0.3f}\\n  Accuracy:
{:0.3f}'.format(accr[0],accr[1]))"
      ],
      "metadata": {
        "colab": {
          "base_uri": "https://localhost:8080/"
        },
        "id": "XXSI6-AgIFrH",
        "outputId": "2d20c4cd-59de-4d5a-dd54-6a3788f40694"
      },
      "execution_count": 32,
      "outputs": [
        {
          "output_type": "stream",
          "name": "stdout",
          "text": [
            "Test set\n",
            "  Loss: 0.089\n",
            "  Accuracy: 0.982\n"
          ]
        }
```

```
            ]
        }
    ]
}
```