```
{
  "nbformat": 4,
  "nbformat_minor": 0,
  "metadata": {
    "colab": {
      "provenance": []
    },
    "kernelspec": {
      "name": "python3",
      "display_name": "Python 3"
    },
    "language_info": {
      "name": "python"
    }
  },
  "cells": [
    {
      "cell_type": "markdown",
      "source": [
        "# **Importing Model building libraries**"
      ],
      "metadata": {
        "id": "MhbOo52DAWRe"
      }
    },
    {
      "cell_type": "code",
      "execution_count": null,
      "metadata": {
        "id": "XZyIORX4AGEp"
      },
      "outputs": [],
      "source": [
        "import pandas as pd \n",
        "import numpy as np\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 to_categorical\n",
        "from keras.models import load_model"
      ]
    },
    {
      "cell_type": "markdown",
      "source": [
        "# **Importing NLTK libraries**"
      ],
      "metadata": {
        "id": "9KrfALGSAfHP"
      }
    },
    {
      "cell_type": "code",
      "source": [
        "import csv\n",
        "import tensorflow as tf\n",
        "import pandas as pd\n",
        "import numpy as np\n",
        "import matplotlib.pyplot as plt\n",
        "from tensorflow.keras.preprocessing.text import Tokenizer\n",
        "from tensorflow.keras.preprocessing.sequence import
pad_sequences\n",
        "import nltk\n",
        "nltk.download('stopwords')  \n",
        "from nltk.corpus import stopwords\n",
        "STOPWORDS = set(stopwords.words('english'))"
      ],
      "metadata": {
        "colab": {
          "base_uri": "https://localhost:8080/"
        },
        "id": "dkpUF8JBATxq",
        "outputId": "84e9e3cb-21c2-4418-fd85-f52206bf30e1"
      },
      "execution_count": null,
      "outputs": [
        {
          "output_type": "stream",
          "name": "stderr",
          "text": [
            "[nltk_data] Downloading package stopwords to
/root/nltk_data...\n",
            "[nltk_data]   Package stopwords is already up-to-date!\n"
          ]
        }
      ]
    },
    {
      "cell_type": "markdown",
      "source": [
        "# **Reading dataset and preprocessing**"
      ],
      "metadata": {
        "id": "EOx_wA_fAOcw"
      }
    },
    {
      "cell_type": "code",
      "source": [
        "from google.colab import drive\n",
        "drive.mount('/content/drive')"
      ],
      "metadata": {
        "colab": {
          "base_uri": "https://localhost:8080/"
        },
        "id": "L5de_CWPAj1-",
        "outputId": "5de09c6f-a645-4036-afb9-406051205928"
      },
      "execution_count": null,
      "outputs": [
        {
          "output_type": "stream",
          "name": "stdout",
          "text": [
            "Drive already mounted at /content/drive; to attempt to
forcibly remount, call drive.mount(\"/content/drive\",
force_remount=True).\n"
          ]
        }
      ]
    },
    {
      "cell_type": "code",
      "source": [
        "cd/content/drive/MyDrive/Colab Notebooks"
      ],
      "metadata": {
        "colab": {
          "base_uri": "https://localhost:8080/"
        },
        "id": "96FFZiUsA5_h",
        "outputId": "471ac98e-18e8-456f-8c90-14fe67a91d3e"
      },
      "execution_count": null,
      "outputs": [
        {
          "output_type": "stream",
          "name": "stdout",
          "text": [
            "/content/drive/MyDrive/Colab Notebooks\n"
          ]
        }
      ]
    },
    {
      "cell_type": "code",
      "source": [
        "df =
pd.read_csv('/content/drive/MyDrive/AI_IBM/spam.csv',delimiter=',',encodi
ng='latin-1')\n",
        "df.head()"
      ],
      "metadata": {
        "colab": {
          "base_uri": "https://localhost:8080/",
          "height": 206
        },
        "id": "r6L00VYbA97L",
        "outputId": "a6cb887c-9bc0-4a4c-e846-9da6d7394c82"
      },
      "execution_count": null,
      "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-9bf3cfc5-e3d7-40f7-a5eb-4b50ab09a8c4\">\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-9bf3cfc5-e3d7-40f7-a5eb-
4b50ab09a8c4')\"\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-9bf3cfc5-e3d7-40f7-
a5eb-4b50ab09a8c4 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-
9bf3cfc5-e3d7-40f7-a5eb-4b50ab09a8c4');\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": 5
        }
      ]
    },
    {
      "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": "erhXmm58CR8r",
        "outputId": "c76e0df2-2ae5-43e8-f8e1-8f789ff32ffe"
      },
      "execution_count": null,
      "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": [
        "df.groupby(['v1']).size()"
      ],
      "metadata": {
        "colab": {
          "base_uri": "https://localhost:8080/"
        },
        "id": "i6E3-2vgCXLi",
        "outputId": "a564a7ae-dd3d-427b-d1c4-208f0373fc68"
      },
      "execution_count": null,
      "outputs": [
        {
          "output_type": "execute_result",
          "data": {
            "text/plain": [
              "v1\n",
              "ham     4825\n",
              "spam     747\n",
              "dtype: int64"
            ]
          },
          "metadata": {},
          "execution_count": 7
        }
      ]
    },
    {
      "cell_type": "code",
      "source": [
        "#Label Encoding Required Column\n",
        "X = df.v2\n",
        "Y = df.v1\n",
        "le = LabelEncoder()\n",
        "Y = le.fit_transform(Y)\n",
        "Y = Y.reshape(-1,1)"
      ],
      "metadata": {
        "id": "qfBLVkujCZWm"
      },
      "execution_count": null,
      "outputs": []
    },
    {
      "cell_type": "code",
      "source": [
        "# Test and train data split \n",
        "X_train,X_test,Y_train,Y_test =
train_test_split(X,Y,test_size=0.15)"
      ],
      "metadata": {
        "id": "RyFCWqR2CbSJ"
      },
      "execution_count": null,
      "outputs": []
    },
    {
      "cell_type": "code",
      "source": [
        "# Tokenisation function\n",
        "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 =
sequence.pad_sequences(sequences,maxlen=max_len)\n"
      ],
      "metadata": {
        "id": "35ZUG70XCdYy"
      },
      "execution_count": null,
      "outputs": []
    },
    {
      "cell_type": "markdown",
      "source": [
        "# **Create Model**\n",
        "# **Add layers (LSTM ,Dense-(HiddenLayers),Ouput)**"
      ],
      "metadata": {
        "id": "zMQYBl1bCf6h"
      }
    },
    {
      "cell_type": "code",
      "source": [
        "#LSTM model\n",
        "inputs = Input(name='InputLayer',shape=[max_len])\n",
        "layer = Embedding(max_words,50,input_length=max_len)(inputs)\n",
        "layer = LSTM(64)(layer)\n",
        "layer = Dense(256,name='FullyConnectedLayer1')(layer)\n",
        "layer = Activation('relu')(layer)\n",
        "layer = Dropout(0.5)(layer)\n",
        "layer = Dense(1,name='OutputLayer')(layer)\n",
        "layer = Activation('sigmoid')(layer)"
      ],
      "metadata": {
        "id": "OVUDcMoJCz-h"
      },
      "execution_count": null,
      "outputs": []
    },
    {
      "cell_type": "code",
      "source": [
        "model = Model(inputs=inputs,outputs=layer)\n",
        "model.summary()\n",

"model.compile(loss='binary_crossentropy',optimizer=RMSprop(),metrics=['a
ccuracy'])"
```

```
      ],
      "metadata": {
        "colab": {
          "base_uri": "https://localhost:8080/"
        },
        "id": "UKam-C9ZC1rQ",
        "outputId": "73bc4ec5-38bb-49b9-90dd-41942f5f5c3d"
      },
      "execution_count": null,
      "outputs": [
        {
          "output_type": "stream",
          "name": "stdout",
          "text": [
            "Model: \"model\"\n",

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

"=================================================================\n",
            " InputLayer (InputLayer)     [(None, 150)]             0
\n",
            "
\n",
            " embedding (Embedding)       (None, 150, 50)           50000
\n",
            "
\n",
            " lstm (LSTM)                 (None, 64)                29440
\n",
            "
\n",
            " FullyConnectedLayer1 (Dense  (None, 256)              16640
\n",
            " )
\n",
            "
\n",
            " activation (Activation)     (None, 256)               0
\n",
            "
\n",
            " dropout (Dropout)           (None, 256)               0
\n",
            "
\n",
            " OutputLayer (Dense)         (None, 1)                 257
\n",
            "
\n",
            " activation_1 (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": "code",
      "source": [

"model.fit(sequences_matrix,Y_train,batch_size=128,epochs=25,validation_s
plit=0.2)"
      ],
      "metadata": {
        "colab": {
          "base_uri": "https://localhost:8080/"
        },
        "id": "XU0fZWRCC3ML",
        "outputId": "b55d7118-743d-4d38-d843-958316b2b6b7"
      },
      "execution_count": null,
      "outputs": [
        {
          "output_type": "stream",
          "name": "stdout",
          "text": [
            "Epoch 1/25\n",
            "30/30 [==============================] - 28s 720ms/step -
loss: 0.3323 - accuracy: 0.8772 - val_loss: 0.1085 - val_accuracy:
0.9715\n",
            "Epoch 2/25\n",
            "30/30 [==============================] - 18s 588ms/step -
loss: 0.0818 - accuracy: 0.9807 - val_loss: 0.0794 - val_accuracy:
0.9800\n",
            "Epoch 3/25\n",
            "30/30 [==============================] - 12s 384ms/step -
loss: 0.0421 - accuracy: 0.9884 - val_loss: 0.0518 - val_accuracy:
0.9842\n",
            "Epoch 4/25\n",
            "30/30 [==============================] - 9s 291ms/step -
loss: 0.0293 - accuracy: 0.9921 - val_loss: 0.0461 - val_accuracy:
0.9884\n",
            "Epoch 5/25\n",
            "30/30 [==============================] - 9s 288ms/step -
loss: 0.0261 - accuracy: 0.9921 - val_loss: 0.0517 - val_accuracy:
0.9873\n",
            "Epoch 6/25\n",
            "30/30 [==============================] - 9s 291ms/step -
loss: 0.0161 - accuracy: 0.9952 - val_loss: 0.0582 - val_accuracy:
0.9863\n",
            "Epoch 7/25\n",
            "30/30 [==============================] - 9s 291ms/step -
loss: 0.0110 - accuracy: 0.9971 - val_loss: 0.0660 - val_accuracy:
0.9895\n",
            "Epoch 8/25\n",
            "30/30 [==============================] - 11s 369ms/step - 
loss: 0.0087 - accuracy: 0.9974 - val_loss: 0.0765 - val_accuracy: 
0.9863\n",
            "Epoch 9/25\n",
            "30/30 [==============================] - 9s 294ms/step - 
loss: 0.0059 - accuracy: 0.9982 - val_loss: 0.0815 - val_accuracy: 
0.9884\n",
            "Epoch 10/25\n",
            "30/30 [==============================] - 9s 290ms/step - 
loss: 0.0051 - accuracy: 0.9987 - val_loss: 0.0902 - val_accuracy: 
0.9852\n",
            "Epoch 11/25\n",
            "30/30 [==============================] - 9s 318ms/step - 
loss: 0.0038 - accuracy: 0.9987 - val_loss: 0.0964 - val_accuracy: 
0.9884\n",
            "Epoch 12/25\n",
            "30/30 [==============================] - 9s 290ms/step - 
loss: 0.0039 - accuracy: 0.9984 - val_loss: 0.1214 - val_accuracy: 
0.9863\n",
            "Epoch 13/25\n",
            "30/30 [==============================] - 11s 363ms/step - 
loss: 0.0011 - accuracy: 0.9997 - val_loss: 0.1153 - val_accuracy: 
0.9895\n",
            "Epoch 14/25\n",
            "30/30 [==============================] - 9s 294ms/step - 
loss: 6.9965e-04 - accuracy: 0.9997 - val_loss: 0.1322 - val_accuracy: 
0.9873\n",
            "Epoch 15/25\n",
            "30/30 [==============================] - 9s 292ms/step - 
loss: 0.7710 - accuracy: 0.9739 - val_loss: 0.1286 - val_accuracy: 
0.9884\n",
            "Epoch 16/25\n",
            "30/30 [==============================] - 9s 294ms/step - 
loss: 5.0771e-04 - accuracy: 0.9997 - val_loss: 0.1294 - val_accuracy: 
0.9895\n",
            "Epoch 17/25\n",
            "30/30 [==============================] - 9s 296ms/step - 
loss: 2.4364e-04 - accuracy: 1.0000 - val_loss: 0.1362 - val_accuracy: 
0.9895\n",
            "Epoch 18/25\n",
            "30/30 [==============================] - 9s 293ms/step - 
loss: 7.7019e-05 - accuracy: 1.0000 - val_loss: 0.1435 - val_accuracy: 
0.9863\n",
            "Epoch 19/25\n",
            "30/30 [==============================] - 9s 294ms/step - 
loss: 4.9329e-05 - accuracy: 1.0000 - val_loss: 0.1585 - val_accuracy: 
0.9863\n",
            "Epoch 20/25\n",
            "30/30 [==============================] - 9s 310ms/step - 
loss: 3.0667e-05 - accuracy: 1.0000 - val_loss: 0.1735 - val_accuracy: 
0.9863\n",
            "Epoch 21/25\n",
            "30/30 [==============================] - 9s 316ms/step - 
loss: 1.8201e-05 - accuracy: 1.0000 - val_loss: 0.1857 - val_accuracy: 
0.9852\n",
            "Epoch 22/25\n",
            "30/30 [==============================] - 9s 295ms/step -
loss: 7.7908e-06 - accuracy: 1.0000 - val_loss: 0.2049 - val_accuracy:
0.9884\n",
            "Epoch 23/25\n",
            "30/30 [==============================] - 9s 295ms/step -
loss: 7.4443e-06 - accuracy: 1.0000 - val_loss: 0.2257 - val_accuracy:
0.9873\n",
            "Epoch 24/25\n",
            "30/30 [==============================] - 9s 298ms/step -
loss: 1.8775e-04 - accuracy: 1.0000 - val_loss: 0.2443 - val_accuracy:
0.9810\n",
            "Epoch 25/25\n",
            "30/30 [==============================] - 9s 292ms/step -
loss: 1.6095e-06 - accuracy: 1.0000 - val_loss: 0.2496 - val_accuracy:
0.9810\n"
          ]
        },
        {
          "output_type": "execute_result",
          "data": {
            "text/plain": [
              "<keras.callbacks.History at 0x7f0dc2ac8190>"
            ]
          },
          "metadata": {},
          "execution_count": 13
        }
      ]
    },
    {
      "cell_type": "code",
      "source": [
        "model.save(\"Ai_Spam_Identifier\")"
      ],
      "metadata": {
        "colab": {
          "base_uri": "https://localhost:8080/"
        },
        "id": "YHIM235qC4wt",
        "outputId": "b07591db-78ae-4d44-e5fe-535ca42ba663"
      },
      "execution_count": null,
      "outputs": [
        {
          "output_type": "stream",
          "name": "stderr",
          "text": [
            "WARNING:absl:Function `_wrapped_model` contains input
name(s) InputLayer with unsupported characters which will be renamed to
inputlayer in the SavedModel.\n",
            "WARNING:absl:Found untraced functions such as
lstm_cell_layer_call_fn,
lstm_cell_layer_call_and_return_conditional_losses while saving (showing
2 of 2). These functions will not be directly callable after loading.\n"
          ]
        }
      ]
    },
    {
      "cell_type": "code",
      "source": [
        "test_sequences = tok.texts_to_sequences(X_test)\n",
        "test_sequences_matrix =
sequence.pad_sequences(test_sequences,maxlen=max_len)"
      ],
      "metadata": {
        "id": "bAIssoULC6Jm"
      },
      "execution_count": null,
      "outputs": []
    },
    {
      "cell_type": "code",
      "source": [
        "accuracy = model.evaluate(test_sequences_matrix,Y_test)\n",
        "print('Accuracy: {:0.3f}'.format(accuracy[1]))"
      ],
      "metadata": {
        "colab": {
          "base_uri": "https://localhost:8080/"
        },
        "id": "YYfPeJHoC7r8",
        "outputId": "ffcf3b94-fd73-40cc-a394-bb24a4eca3c1"
      },
      "execution_count": null,
      "outputs": [
        {
          "output_type": "stream",
          "name": "stdout",
          "text": [
            "27/27 [==============================] - 1s 27ms/step -
loss: 0.3614 - accuracy: 0.9833\n",
            "Accuracy: 0.983\n"
          ]
        }
      ]
    },
    {
      "cell_type": "code",
      "source": [
        "y_pred = model.predict(test_sequences_matrix)\n",
        "print(y_pred[25:40].round(3))"
      ],
      "metadata": {
        "colab": {
          "base_uri": "https://localhost:8080/"
        },
        "id": "TAd2boE7C9iz",
        "outputId": "395e247e-4eff-43a4-f7bf-7a152b2e8299"
      },
      "execution_count": null,
      "outputs": [
        {
          "output_type": "stream",
          "name": "stdout",
          "text": [
          "27/27 [==============================] - 1s 25ms/step\n",
          "[[0.]\n",
          " [0.]\n",
          " [0.]\n",
          " [0.]\n",
          " [0.]\n",
          " [0.]\n",
          " [0.]\n",
          " [1.]\n",
          " [0.]\n",
          " [0.]\n",
          " [0.]\n",
          " [1.]\n",
          " [0.]\n",
          " [0.]\n",
          " [0.]]\n"
        ]
      }
    ]
  },
  {
    "cell_type": "code",
    "source": [
      "print(Y_test[25:40])"
    ],
    "metadata": {
      "colab": {
        "base_uri": "https://localhost:8080/"
      },
      "id": "TobFDYACC_LF",
      "outputId": "01314bb6-79e0-4206-b67f-a4dc1187c725"
    },
    "execution_count": null,
    "outputs": [
      {
        "output_type": "stream",
        "name": "stdout",
        "text": [
          "[[0]\n",
          " [0]\n",
          " [0]\n",
          " [0]\n",
          " [0]\n",
          " [0]\n",
          " [0]\n",
          " [1]\n",
          " [0]\n",
          " [0]\n",
          " [0]\n",
          " [1]\n",
          " [0]\n",
          " [0]\n",
          " [0]]\n"
        ]
      }
    ]
  }
]
```

```
}
```