```
{
 "cells": [
  {
   "cell_type": "code",
   "execution_count": 1,
   "id": "160c9a6c",
   "metadata": {},
   "outputs": [],
   "source": [
    "import numpy as np\n",
    "import pandas as pd\n",
    "import matplotlib.pyplot as plt\n",
    "%matplotlib inline\n",
    "import seaborn as sns\n",
    "# roc curve and auc score\\n\",\n",
    "from sklearn.datasets import make_classification"
   ]
  },
  {
   "cell_type": "code",
   "execution_count": 2,
   "id": "e5c0a2b6",
   "metadata": {},
   "outputs": [],
   "source": [
    "from sklearn.model_selection import train_test_split\n",
    "X, y = make_classification(n_samples=2000, n_classes=2,
weights=[1,1], random_state=1)"
   ]
  },
  {
   "cell_type": "code",
   "execution_count": 3,
   "id": "e3411540",
   "metadata": {},
   "outputs": [
    {
     "data": {
      "text/plain": [
       "(2000, 20)"
      ]
     },
     "execution_count": 3,
     "metadata": {},
     "output_type": "execute_result"
    }
   ],
   "source": [
    "X.shape"
   ]
  },
  {
   "cell_type": "code",
   "execution_count": 4,
   "id": "af13df54",
   "metadata": {},
   "outputs": [
    {
     "data": {
      "text/plain": [
       "array([0, 0, 0, ..., 1, 1, 0])"
      ]
     },
     "execution_count": 4,
     "metadata": {},
     "output_type": "execute_result"
    }
   ],
   "source": [
    "y"
   ]
  },
  {
   "cell_type": "code",
   "execution_count": 5,
   "id": "ca1d70a9",
   "metadata": {},
   "outputs": [],
   "source": [
    "from sklearn.model_selection import train_test_split\n",
    "\n",
    "X_train,X_test,y_train,y_test = train_test_split(X, y,
test_size=0.3, random_state=1)"
   ]
  },
  {
   "cell_type": "code",
   "execution_count": 6,
   "id": "d7f333d1",
   "metadata": {},
   "outputs": [],
   "source": [
    "from sklearn.metrics import roc_curve\n",
    "from sklearn.metrics import roc_auc_score"
   ]
  },
  {
   "cell_type": "markdown",
   "id": "5eb70b84",
   "metadata": {},
   "source": [
    "# Random Forests"
   ]
  },
  {
   "cell_type": "code",
   "execution_count": 7,
   "id": "07671935",
   "metadata": {},
   "outputs": [
    {
     "name": "stdout",
     "output_type": "stream",
     "text": [
      "RF train roc-auc: 1.0\n",
      "RF test roc-auc: 0.9805055555555555\n"
      ]
     }
    ],
    "source": [
     "## Apply RandomForestClassifier\n",
     "from sklearn.ensemble import RandomForestClassifier\n",
     "rf_model = RandomForestClassifier()\n",
     "rf_model.fit(X_train, y_train)\n",
     "ytrain_pred = rf_model.predict_proba(X_train)\n",
     "print('RF train roc-auc: {}'.format(roc_auc_score(y_train,
ytrain_pred[:,1])))\n",
     "ytest_pred = rf_model.predict_proba(X_test)\n",
     "print('RF test roc-auc: {}'.format(roc_auc_score(y_test,
ytest_pred[:,1])))"
    ]
   },
   {
    "cell_type": "code",
    "execution_count": 8,
    "id": "db3a901c",
    "metadata": {},
    "outputs": [
     {
      "data": {
       "text/plain": [
        "array([[1.  , 0.  ],\n",
        "       [1.  , 0.  ],\n",
        "       [0.04, 0.96],\n",
        "       ...,\n",
        "       [0.97, 0.03],\n",
        "       [0.98, 0.02],\n",
        "       [0.23, 0.77]])"
       ]
      },
      "execution_count": 8,
      "metadata": {},
      "output_type": "execute_result"
     }
    ],
    "source": [
     "ytrain_pred"
    ]
   },
   {
    "cell_type": "markdown",
    "id": "72b427ed",
    "metadata": {},
    "source": [
     "# Logistic Regression"
    ]
   },
   {
    "cell_type": "code",
    "execution_count": 9,
    "id": "8027a57a",
    "metadata": {},
    "outputs": [
     {
      "name": "stdout",
      "output_type": "stream",
      "text": [
       "Logistic train roc-auc: 0.9863568922694498\n",
       "Logistic test roc-auc: 0.9885777777777777\n"
      ]
     }
    ],
    "source": [
     "from sklearn.linear_model import LogisticRegression\n",
     "log_classifier=LogisticRegression()\n",
     "log_classifier.fit(X_train, y_train)\n",
     "ytrain_pred = log_classifier.predict_proba(X_train)\n",
     "print('Logistic train roc-auc: {}'.format(roc_auc_score(y_train, 
ytrain_pred[:,1])))\n",
     "ytest_pred = log_classifier.predict_proba(X_test)\n",
     "print('Logistic test roc-auc: {}'.format(roc_auc_score(y_test, 
ytest_pred[:,1])))"
    ]
   },
   {
    "cell_type": "markdown",
    "id": "62f82d03",
    "metadata": {},
    "source": [
     "# Adaboost Classifier"
    ]
   },
   {
    "cell_type": "code",
    "execution_count": 10,
    "id": "5ac32b3b",
    "metadata": {},
    "outputs": [
     {
      "name": "stdout",
      "output_type": "stream",
      "text": [
       "Adaboost train roc-auc: 0.9975081174960356\n",
       "Adaboost test roc-auc: 0.9826111111111111\n"
      ]
     }
    ],
    "source": [
     "from sklearn.ensemble import AdaBoostClassifier\n",
     "ada_classifier=AdaBoostClassifier()\n",
     "ada_classifier.fit(X_train, y_train)\n",
     "ytrain_pred = ada_classifier.predict_proba(X_train)\n",
     "print('Adaboost train roc-auc: {}'.format(roc_auc_score(y_train, 
ytrain_pred[:,1])))\n",
     "ytest_pred = ada_classifier.predict_proba(X_test)\n",
     "print('Adaboost test roc-auc: {}'.format(roc_auc_score(y_test, 
ytest_pred[:,1])))"
    ]
   },
   {
    "cell_type": "markdown",
    "id": "2dabec9b",
   "metadata": {},
   "source": [
    "# KNNClassifier"
   ]
  },
  {
   "cell_type": "code",
   "execution_count": 11,
   "id": "17fd8398",
   "metadata": {},
   "outputs": [
    {
     "name": "stdout",
     "output_type": "stream",
     "text": [
      "Adaboost train roc-auc: 0.981670071491109\n",
      "Adaboost test roc-auc: 0.9426111111111111\n"
     ]
    }
   ],
   "source": [
    "from sklearn.neighbors import KNeighborsClassifier\n",
    "knn_classifier=KNeighborsClassifier()\n",
    "knn_classifier.fit(X_train, y_train)\n",
    "ytrain_pred = knn_classifier.predict_proba(X_train)\n",
    "print('Adaboost train roc-auc: {}'.format(roc_auc_score(y_train, 
ytrain_pred[:,1])))\n",
    "ytest_pred = knn_classifier.predict_proba(X_test)\n",
    "print('Adaboost test roc-auc: {}'.format(roc_auc_score(y_test, 
ytest_pred[:,1])))"
   ]
  },
  {
   "cell_type": "markdown",
   "id": "47063591",
   "metadata": {},
   "source": [
    "# No we will focus on selecting the best threshold for maximum 
accuracy"
   ]
  },
  {
   "cell_type": "code",
   "execution_count": 13,
   "id": "5befeb9d",
   "metadata": {},
   "outputs": [
    {
     "name": "stdout",
     "output_type": "stream",
     "text": [
      "Ensemble test roc-auc: 0.9849\n"
     ]
    }
   ],
   "source": [
    "pred=[]\n",
    "for model in
[rf_model,log_classifier,ada_classifier,knn_classifier]:\n",
    "    pred.append(pd.Series(model.predict_proba(X_test)[:,1]))\n",
    "final_prediction=pd.concat(pred,axis=1).mean(axis=1)\n",
    "print('Ensemble test roc-auc:
{}'.format(roc_auc_score(y_test,final_prediction)))"
   ]
  },
  {
   "cell_type": "code",
   "execution_count": 14,
   "id": "eca4d196",
   "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>0</th>\n",
       "      <th>1</th>\n",
       "      <th>2</th>\n",
       "      <th>3</th>\n",
       "    </tr>\n",
       "  </thead>\n",
       "  <tbody>\n",
       "    <tr>\n",
       "      <th>0</th>\n",
       "      <td>0.99</td>\n",
       "      <td>0.991861</td>\n",
       "      <td>0.559186</td>\n",
       "      <td>1.0</td>\n",
       "    </tr>\n",
       "    <tr>\n",
       "      <th>1</th>\n",
       "      <td>0.02</td>\n",
       "      <td>0.000008</td>\n",
       "      <td>0.463282</td>\n",
       "      <td>0.0</td>\n",
       "    </tr>\n",
       "    <tr>\n",
       "      <th>2</th>\n",
"      <td>0.97</td>\n",
"      <td>0.966929</td>\n",
"      <td>0.538202</td>\n",
"      <td>0.8</td>\n",
"    </tr>\n",
"    <tr>\n",
"      <th>3</th>\n",
"      <td>0.91</td>\n",
"      <td>0.761539</td>\n",
"      <td>0.509875</td>\n",
"      <td>0.8</td>\n",
"    </tr>\n",
"    <tr>\n",
"      <th>4</th>\n",
"      <td>0.55</td>\n",
"      <td>0.779443</td>\n",
"      <td>0.490344</td>\n",
"      <td>0.4</td>\n",
"    </tr>\n",
"    <tr>\n",
"      <th>...</th>\n",
"      <td>...</td>\n",
"      <td>...</td>\n",
"      <td>...</td>\n",
"      <td>...</td>\n",
"    </tr>\n",
"    <tr>\n",
"      <th>595</th>\n",
"      <td>0.00</td>\n",
"      <td>0.024239</td>\n",
"      <td>0.461121</td>\n",
"      <td>0.0</td>\n",
"    </tr>\n",
"    <tr>\n",
"      <th>596</th>\n",
"      <td>0.05</td>\n",
"      <td>0.000003</td>\n",
"      <td>0.441377</td>\n",
"      <td>0.0</td>\n",
"    </tr>\n",
"    <tr>\n",
"      <th>597</th>\n",
"      <td>0.99</td>\n",
"      <td>0.984385</td>\n",
"      <td>0.532403</td>\n",
"      <td>1.0</td>\n",
"    </tr>\n",
"    <tr>\n",
"      <th>598</th>\n",
"      <td>0.01</td>\n",
"      <td>0.001147</td>\n",
"      <td>0.441720</td>\n",
"      <td>0.2</td>\n",
"    </tr>\n",
"    <tr>\n",
"      <th>599</th>\n",
"      <td>0.99</td>\n",
"      <td>0.989540</td>\n",
    "      <td>0.559890</td>\n",
    "      <td>0.8</td>\n",
    "    </tr>\n",
    "  </tbody>\n",
    "</table>\n",
    "<p>600 rows Ã— 4 columns</p>\n",
    "</div>"
   ],
   "text/plain": [
    "        0         1         2    3\n",
    "0    0.99  0.991861  0.559186  1.0\n",
    "1    0.02  0.000008  0.463282  0.0\n",
    "2    0.97  0.966929  0.538202  0.8\n",
    "3    0.91  0.761539  0.509875  0.8\n",
    "4    0.55  0.779443  0.490344  0.4\n",
    "..    ...       ...       ...  ...\n",
    "595  0.00  0.024239  0.461121  0.0\n",
    "596  0.05  0.000003  0.441377  0.0\n",
    "597  0.99  0.984385  0.532403  1.0\n",
    "598  0.01  0.001147  0.441720  0.2\n",
    "599  0.99  0.989540  0.559890  0.8\n",
    "\n",
    "[600 rows x 4 columns]"
   ]
  },
  "execution_count": 14,
  "metadata": {},
  "output_type": "execute_result"
 }
],
"source": [
 "pd.concat(pred,axis=1)"
]
},
{
"cell_type": "code",
"execution_count": 15,
"id": "e897a615",
"metadata": {},
"outputs": [
 {
  "data": {
   "text/plain": [
    "0      0.885262\n",
    "1      0.120823\n",
    "2      0.818783\n",
    "3      0.745353\n",
    "4      0.554947\n",
    "         ...   \n",
    "595    0.121340\n",
    "596    0.122845\n",
    "597    0.876697\n",
    "598    0.163217\n",
    "599    0.834857\n",
    "Length: 600, dtype: float64"
   ]
  },
  "execution_count": 15,
     "metadata": {},
     "output_type": "execute_result"
    }
   ],
   "source": [
    "final_prediction"
   ]
  },
  {
   "cell_type": "code",
   "execution_count": 16,
   "id": "b7dcc39f",
   "metadata": {},
   "outputs": [
    {
     "data": {
      "text/plain": [
       "array([1.91188114, 0.91188114, 0.90908893, 0.90827475,
0.81446108,\n",
       "       0.81412833, 0.8044465 , 0.8030558 , 0.77640513,
0.77571156,\n",
       "       0.76053202, 0.76037124, 0.74946856, 0.74887362,
0.72210041,\n",
       "       0.72143711, 0.6642442 , 0.63743537, 0.58779968,
0.58675354,\n",
       "       0.5840152 , 0.58236186, 0.57577749, 0.57550386,
0.55494674,\n",
       "       0.54889283, 0.54845371, 0.54127948, 0.53365858,
0.49116892,\n",
       "       0.4346034 , 0.40068654, 0.39879719, 0.37475652,
0.35586612,\n",
       "       0.24217239, 0.24140421, 0.24119003, 0.23896893,
0.19125353,\n",
       "       0.19098417, 0.12284501, 0.1228351 , 0.10498954])"
      ]
     },
     "execution_count": 16,
     "metadata": {},
     "output_type": "execute_result"
    }
   ],
   "source": [
    "#### Calculate the ROc Curve\n",
    "\n",
    "\n",
    "fpr, tpr, thresholds = roc_curve(y_test, final_prediction)\n",
    "thresholds"
   ]
  },
  {
   "cell_type": "code",
   "execution_count": 17,
   "id": "e54eccf1",
   "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>thresholds</th>\n",
    "      <th>accuracy</th>\n",
    "    </tr>\n",
    "  </thead>\n",
    "  <tbody>\n",
    "    <tr>\n",
    "      <th>30</th>\n",
    "      <td>0.434603</td>\n",
    "      <td>0.961667</td>\n",
    "    </tr>\n",
    "    <tr>\n",
    "      <th>29</th>\n",
    "      <td>0.491169</td>\n",
    "      <td>0.958333</td>\n",
    "    </tr>\n",
    "    <tr>\n",
    "      <th>28</th>\n",
    "      <td>0.533659</td>\n",
    "      <td>0.958333</td>\n",
    "    </tr>\n",
    "    <tr>\n",
    "      <th>27</th>\n",
    "      <td>0.541279</td>\n",
    "      <td>0.958333</td>\n",
    "    </tr>\n",
    "    <tr>\n",
    "      <th>25</th>\n",
    "      <td>0.548893</td>\n",
    "      <td>0.958333</td>\n",
    "    </tr>\n",
    "  </tbody>\n",
    "</table>\n",
    "</div>"
   ],
   "text/plain": [
    "    thresholds  accuracy\n",
    "30    0.434603  0.961667\n",
    "29    0.491169  0.958333\n",
    "28    0.533659  0.958333\n",
    "27    0.541279  0.958333\n",
       "25    0.548893  0.958333"
      ]
     },
     "execution_count": 17,
     "metadata": {},
     "output_type": "execute_result"
    }
   ],
   "source": [
    "from sklearn.metrics import accuracy_score\n",
    "accuracy_ls = []\n",
    "for thres in thresholds:\n",
    "    y_pred = np.where(final_prediction>thres,1,0)\n",
    "    accuracy_ls.append(accuracy_score(y_test, y_pred,
normalize=True))\n",
    "    \n",
    "accuracy_ls = pd.concat([pd.Series(thresholds),
pd.Series(accuracy_ls)],\n",
    "                        axis=1)\n",
    "accuracy_ls.columns = ['thresholds', 'accuracy']\n",
    "accuracy_ls.sort_values(by='accuracy', ascending=False,
inplace=True)\n",
    "accuracy_ls.head()"
   ]
  },
  {
   "cell_type": "code",
   "execution_count": 18,
   "id": "d226c83b",
   "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>thresholds</th>\n",
       "      <th>accuracy</th>\n",
       "    </tr>\n",
       "  </thead>\n",
       "  <tbody>\n",
       "    <tr>\n",
"       <th>30</th>\n",
"       <td>0.434603</td>\n",
"       <td>0.961667</td>\n",
"     </tr>\n",
"     <tr>\n",
"       <th>29</th>\n",
"       <td>0.491169</td>\n",
"       <td>0.958333</td>\n",
"     </tr>\n",
"     <tr>\n",
"       <th>28</th>\n",
"       <td>0.533659</td>\n",
"       <td>0.958333</td>\n",
"     </tr>\n",
"     <tr>\n",
"       <th>27</th>\n",
"       <td>0.541279</td>\n",
"       <td>0.958333</td>\n",
"     </tr>\n",
"     <tr>\n",
"       <th>25</th>\n",
"       <td>0.548893</td>\n",
"       <td>0.958333</td>\n",
"     </tr>\n",
"     <tr>\n",
"       <th>26</th>\n",
"       <td>0.548454</td>\n",
"       <td>0.956667</td>\n",
"     </tr>\n",
"     <tr>\n",
"       <th>24</th>\n",
"       <td>0.554947</td>\n",
"       <td>0.956667</td>\n",
"     </tr>\n",
"     <tr>\n",
"       <th>31</th>\n",
"       <td>0.400687</td>\n",
"       <td>0.951667</td>\n",
"     </tr>\n",
"     <tr>\n",
"       <th>32</th>\n",
"       <td>0.398797</td>\n",
"       <td>0.950000</td>\n",
"     </tr>\n",
"     <tr>\n",
"       <th>23</th>\n",
"       <td>0.575504</td>\n",
"       <td>0.950000</td>\n",
"     </tr>\n",
"     <tr>\n",
"       <th>19</th>\n",
"       <td>0.586754</td>\n",
"       <td>0.950000</td>\n",
"     </tr>\n",
"     <tr>\n",
"       <th>22</th>\n",
"       <td>0.575777</td>\n",
"       <td>0.948333</td>\n",
"    </tr>\n",
"    <tr>\n",
"      <th>18</th>\n",
"      <td>0.587800</td>\n",
"      <td>0.948333</td>\n",
"    </tr>\n",
"    <tr>\n",
"      <th>21</th>\n",
"      <td>0.582362</td>\n",
"      <td>0.948333</td>\n",
"    </tr>\n",
"    <tr>\n",
"      <th>20</th>\n",
"      <td>0.584015</td>\n",
"      <td>0.948333</td>\n",
"    </tr>\n",
"    <tr>\n",
"      <th>33</th>\n",
"      <td>0.374757</td>\n",
"      <td>0.943333</td>\n",
"    </tr>\n",
"    <tr>\n",
"      <th>34</th>\n",
"      <td>0.355866</td>\n",
"      <td>0.941667</td>\n",
"    </tr>\n",
"    <tr>\n",
"      <th>17</th>\n",
"      <td>0.637435</td>\n",
"      <td>0.938333</td>\n",
"    </tr>\n",
"    <tr>\n",
"      <th>16</th>\n",
"      <td>0.664244</td>\n",
"      <td>0.936667</td>\n",
"    </tr>\n",
"    <tr>\n",
"      <th>15</th>\n",
"      <td>0.721437</td>\n",
"      <td>0.913333</td>\n",
"    </tr>\n",
"    <tr>\n",
"      <th>14</th>\n",
"      <td>0.722100</td>\n",
"      <td>0.911667</td>\n",
"    </tr>\n",
"    <tr>\n",
"      <th>37</th>\n",
"      <td>0.241190</td>\n",
"      <td>0.880000</td>\n",
"    </tr>\n",
"    <tr>\n",
"      <th>35</th>\n",
"      <td>0.242172</td>\n",
"      <td>0.880000</td>\n",
"    </tr>\n",
"    <tr>\n",
"      <th>38</th>\n",
"       <td>0.238969</td>\n",
"       <td>0.878333</td>\n",
"     </tr>\n",
"     <tr>\n",
"       <th>36</th>\n",
"       <td>0.241404</td>\n",
"       <td>0.878333</td>\n",
"     </tr>\n",
"     <tr>\n",
"       <th>13</th>\n",
"       <td>0.748874</td>\n",
"       <td>0.871667</td>\n",
"     </tr>\n",
"     <tr>\n",
"       <th>12</th>\n",
"       <td>0.749469</td>\n",
"       <td>0.870000</td>\n",
"     </tr>\n",
"     <tr>\n",
"       <th>11</th>\n",
"       <td>0.760371</td>\n",
"       <td>0.866667</td>\n",
"     </tr>\n",
"     <tr>\n",
"       <th>10</th>\n",
"       <td>0.760532</td>\n",
"       <td>0.865000</td>\n",
"     </tr>\n",
"     <tr>\n",
"       <th>9</th>\n",
"       <td>0.775712</td>\n",
"       <td>0.846667</td>\n",
"     </tr>\n",
"     <tr>\n",
"       <th>8</th>\n",
"       <td>0.776405</td>\n",
"       <td>0.845000</td>\n",
"     </tr>\n",
"     <tr>\n",
"       <th>39</th>\n",
"       <td>0.191254</td>\n",
"       <td>0.836667</td>\n",
"     </tr>\n",
"     <tr>\n",
"       <th>40</th>\n",
"       <td>0.190984</td>\n",
"       <td>0.835000</td>\n",
"     </tr>\n",
"     <tr>\n",
"       <th>7</th>\n",
"       <td>0.803056</td>\n",
"       <td>0.816667</td>\n",
"     </tr>\n",
"     <tr>\n",
"       <th>6</th>\n",
"       <td>0.804447</td>\n",
"       <td>0.815000</td>\n",
"     </tr>\n",
       "    <tr>\n",
       "      <th>5</th>\n",
       "      <td>0.814128</td>\n",
       "      <td>0.791667</td>\n",
       "    </tr>\n",
       "    <tr>\n",
       "      <th>4</th>\n",
       "      <td>0.814461</td>\n",
       "      <td>0.790000</td>\n",
       "    </tr>\n",
       "    <tr>\n",
       "      <th>41</th>\n",
       "      <td>0.122845</td>\n",
       "      <td>0.675000</td>\n",
       "    </tr>\n",
       "    <tr>\n",
       "      <th>42</th>\n",
       "      <td>0.122835</td>\n",
       "      <td>0.673333</td>\n",
       "    </tr>\n",
       "    <tr>\n",
       "      <th>3</th>\n",
       "      <td>0.908275</td>\n",
       "      <td>0.505000</td>\n",
       "    </tr>\n",
       "    <tr>\n",
       "      <th>2</th>\n",
       "      <td>0.909089</td>\n",
       "      <td>0.503333</td>\n",
       "    </tr>\n",
       "    <tr>\n",
       "      <th>43</th>\n",
       "      <td>0.104990</td>\n",
       "      <td>0.501667</td>\n",
       "    </tr>\n",
       "    <tr>\n",
       "      <th>1</th>\n",
       "      <td>0.911881</td>\n",
       "      <td>0.500000</td>\n",
       "    </tr>\n",
       "    <tr>\n",
       "      <th>0</th>\n",
       "      <td>1.911881</td>\n",
       "      <td>0.500000</td>\n",
       "    </tr>\n",
       "  </tbody>\n",
       "</table>\n",
       "</div>"
      ],
      "text/plain": [
       "    thresholds  accuracy\n",
       "30    0.434603  0.961667\n",
       "29    0.491169  0.958333\n",
       "28    0.533659  0.958333\n",
       "27    0.541279  0.958333\n",
       "25    0.548893  0.958333\n",
       "26    0.548454  0.956667\n",
       "24    0.554947  0.956667\n",
      "31    0.400687  0.951667\n",
      "32    0.398797  0.950000\n",
      "23    0.575504  0.950000\n",
      "19    0.586754  0.950000\n",
      "22    0.575777  0.948333\n",
      "18    0.587800  0.948333\n",
      "21    0.582362  0.948333\n",
      "20    0.584015  0.948333\n",
      "33    0.374757  0.943333\n",
      "34    0.355866  0.941667\n",
      "17    0.637435  0.938333\n",
      "16    0.664244  0.936667\n",
      "15    0.721437  0.913333\n",
      "14    0.722100  0.911667\n",
      "37    0.241190  0.880000\n",
      "35    0.242172  0.880000\n",
      "38    0.238969  0.878333\n",
      "36    0.241404  0.878333\n",
      "13    0.748874  0.871667\n",
      "12    0.749469  0.870000\n",
      "11    0.760371  0.866667\n",
      "10    0.760532  0.865000\n",
      "9     0.775712  0.846667\n",
      "8     0.776405  0.845000\n",
      "39    0.191254  0.836667\n",
      "40    0.190984  0.835000\n",
      "7     0.803056  0.816667\n",
      "6     0.804447  0.815000\n",
      "5     0.814128  0.791667\n",
      "4     0.814461  0.790000\n",
      "41    0.122845  0.675000\n",
      "42    0.122835  0.673333\n",
      "3     0.908275  0.505000\n",
      "2     0.909089  0.503333\n",
      "43    0.104990  0.501667\n",
      "1     0.911881  0.500000\n",
      "0     1.911881  0.500000"
     ]
    },
    "execution_count": 18,
    "metadata": {},
    "output_type": "execute_result"
   }
  ],
  "source": [
   "accuracy_ls"
  ]
 },
 {
  "cell_type": "code",
  "execution_count": 19,
  "id": "e363bebe",
  "metadata": {},
  "outputs": [],
  "source": [
   "def plot_roc_curve(fpr, tpr):\n",
   "    plt.plot(fpr, tpr, color='orange', label='ROC')\n",
   "    plt.plot([0, 1], [0, 1], color='darkblue', linestyle='--')\n",
    "    plt.xlabel('False Positive Rate')\n",
    "    plt.ylabel('True Positive Rate')\n",
    "    plt.title('Receiver Operating Characteristic (ROC) Curve')\n",
    "    plt.legend()\n",
    "    plt.show()"
   ]
  },
  {
   "cell_type": "code",
   "execution_count": 20,
   "id": "d7ed05a3",
   "metadata": {},
   "outputs": [
    {
     "data": {
      "image/png":
```

"iVBORw0KGgoAAAANSUhEUgAAAYIAAAEWCAYAAABrDZDcAAAAOXRFWHRTb2Z0d2FyZQBNYXRw
bG90bGliIHZlcnNpb24zLjUuMSwgaHR0cHM6Ly9tYXRwbG90bGliLm9yZy/YYfK9AAAACXBIW
XMAAAsTAAALEwEAmpwYAAA3cElEQVR4nO3dd3gUVffA8e+hJkDoRTqoICQgIF2pAnYFBXlRFE
URsL7+sKEogiIWUBApigVURFRUehNfERWRonQEUEioPSakHZ+f8wEl7BJFshkk93zeZ482dm
5M3PuzO6cmTuzd0RVMcYYE77yBDsAY4wxwWWJwBhjwpwlAmOMCXOWCIwxJsZIjDGmDBnicAY
Y8KcJYIcRkTWi0ibYMeRU4jIUyLyTpCWPFEhgRj2VlNRLqLyIKznPasP5Mi8oOINDibac+Wi
DwkIi9l5zJzO0sEGRCRbSISJyJHRWS3u2Mo4uUUyVTGVRd5uYxUIlJQRF4UkT3ku1FYWgHanT
a8jvbz6SIXA8cUdVf3OFBIpLofp8o8OisgSEWmeZpriIjLO/b4dF5Gl1ItLTz7xvFZEV7rx2ichc
EWnhjh4P3CYiZTOILVsM+AdsA1QBRwO9AbeN2DG
ERExtrn7XXgv8BDQEmgJjANuDarF5TBNvBcEJfdF/gwzXufufN+n0sA3OJ9BAESkALAQqAo0B4
oBjwEviUg/n3L9gJHAUKAcUAUYC3C3QEUNV4YC7QI4PYsmzbB3PbZhlVtb90/oBtQHuf4VeA2T7
DzYAlwEFgNdDGZ1xJxgATPMZdx2wyp1uCXBx2mUCP/O3wXsGd/3yg
qk9ZBe4HfgP+8FO3dkA8UDnN+02BZ5uzEeAr
UAft2xht0wKcNT9qwAMAia5Zaq59boD+NNdFwN8lhcJvO2HdgCHgZVAS59+4CfC+gCfCju652AaOBAj7TxABfAfuBv4GngZ4GngKuABCD
RXSer3bLFgHfd+fwFDPS/uuPudNf5CHdeqMTPs9APK6
cf3urpOVpPkMueUKuNuzUpp1MslnONrdnmXc4bvdmAqnmdd/3HiKuvU+CtycyXe3O/DNOWz7R
UAvn+GT68/f9wt4ExieZh7TgX7u6wrA58Aet/xDwd6/nRJrsAPIyX9pvgCVgLXA6+5wRAfzt
F0HqCDO5z6oZ4NfAKUAPIDrd33L3E/7E3dL9Ud7nIK+lnm/4B7fOIZBrzpvu4EbAfqA/mAp4E
laT6oX+EkpEg/dXsJ+Dac/n3x30IpwdTR2cnfXn/LtjzmwdLMLZYce4MebHOeK6AGdn1Bo4
Dlzilm9Dmh03/hPB2zg7/XrACaC2b53V4JZweXXiLoC2zPZPtPxNmRNnHj/wiY4jP+NqCUO
+4RYDcQ4RN3orud8rjxNsRJnPncumwEHnbLR+Hs1B8BItzhpmcnXgc+ypwFvudukLE6iTt1mdw
JJwIPusiI5NRFcibMDL+5uh9pAeZ86D8nge/AYzvfgInfaekApP+suBjiWwbYs4G6vvUA+970
pwPt+5pXPrc+VOIkxKXWaDLbdJcD+c9j2i8g8EZz8fgGtcA4KxB1fAicRVnC3/0pgoFv83EO
gq4M9j4u9S+nnarnRNNE5AjORv4RHZ9/zZgjqrOUdVKUYvKWAFcIyLlgauBu1V72t14FRnO
7p7gLdU9SdVTbVbV93F2Zs38LHsH4sycAs4TStAN/c9gD7Ai/lBxGu53kdfNpk/ruZmMpYiu7uuu
fepXF2PP7scsen+lBV16nqMeAZOaI5M1FvtZUOwqoSCmxGGGCW+ubc
4VV2NcxZSz32/KzDM5lEqg/r7+kJVl7nr+COcJkIAVHVHWmUzqajqbsT4zIHYYnUkz3x4sqqL
Eefq5z6/YqUBBnB5nqR1Wd
5q6bOFVdqapL3fLbcHbkjrym1Okq+pO/gESkHM7ZztAc4XfzKbZTV
d9wl5V2+yfiJAiJaOduujaoayLoA58zmaVXd5G7D1aq6z0+54jDGPm0Au7rqFdD
6T7vi97vhSwF6fadJzBOfswZ9At31mfL9f3+Ekh9TPchec7b8TaIxzcPScqiao6lacg5lufuc
aBJYIMtdJVaNwjlZr8e8Osipws3vR66D4A4xyrAI2mmq4xz5JDWVKC5iFTA
OeJQnA9c6nxe95nHfpwjtIo+0+/IoF573Vj9Ke+O9zeF7ThH9qXNPN6k6u8J1S3Mf7hYThZ4S6QTB
acmnUDs9nl9HEi9gF8hzfIyqv8+0q9/IMtCRB4RkY0icsitSzFSFOrUvautcUkVnuhdDDOMk7tX
xlnOaWQFTF2Qa7fNb7WzhnBn6X7UtV/4fTLDUG+FtExotI0QCXHWcB3CSTVqfqmpxnLb9dTh
nSan8fibdNvjS7vh9QOkA2uWjcJq9/Al022fm5DpW5zRgCu6BG3ArzoEDONurQprvyVM46yBH
sEQQIPfodSIw3H1rvB86tZqx3AC2mmK6SqH/tZ5kGcI+auOB+sj90PX
Op8+qSZT6SqLvGdRQZVWgg0FZHKvm+KSBOcL/v/fN72LVYRKQgTtPScKCcu
OYg5PAMos3ELtwmoT8xZ3W10AlEWl0NgsSkZbAEzjbpaoRbb2AEzjbpoRbl0P8Wxc4vT7jgF+BGqpaFGdkn
+B06TmT9p57MD5yyytM96L6qqMRlMc+oMVUepakOcJpyaOE0+mU6Z4XNHu6sg2kqZt/Z3aOaVFMSPPAa6DEsM5ka3ob6sq7sU5
qx3knkGD85m8WkQKpyneGae+S3GuscTjNLL2aI/gWz7Y0Ahn+Hz/JRJu64+Brq4Z+VNc

T7r4KyzP9J8T6JU9RpyCEsEZ2Yk0EFE6uNcBLxeRK4UkbwiEuHe/ljJPc2eC4wVkRIikl9EWr
nzeBvoKyJN3TtpCovItSLi7+gJnKagHjhfhsk+778JPCkiMQAiUkxEbg60Iqq6EOcL8bmIxLh
1aIZzFDNOVX/zKX6biESLSCHgOWCqqiZntA7SWWwBnOaTPUCSiFwN+N7S+DdQSkTSO6XPzKc4
66SEuwN6IL2Cbv3GAh+7MRdw4+8mIv0DWFYUTlv1HiCfiAzEuZiZ2TSHgaMiUgu412fcLOA8E
XlYnNt6o0SkqTvub6Ba6l1X7udrAfCqiBQVkTwicoGItCYAItLY/fzlx9nhxeNcPE1d1vkZTP
4O8LyI1HA/vxeLSKm0hVQ1EWfHnm5Mqvorzk0Oj7tvfQjEAp+JSDX3e3MlThPfIFU9pKqHcNr
ax4hIJxEp5Ja7WkRe8Zl9a5zvoL/lBrLtVwE3ufO/EOdCdobUuU12j7uO5rsHcuBcvzksIk+I
SKT7XakjIo0zm2d2sURwBlR1D/AB8Iyq7sC5Xe0pnI2/A+eoKnWd3o5z5PwrzrWFh915rMBpG
x2Nc/q8BedCVHpm4Nzl8LfbJp4ay5fAy8AUt5lhHU678ZnojHML3zycOzEm4dyJ8mCach/inA
3txrmQ+ZAbQ2br4BSqesSd9lOcut/q1i91/K84R1Vb3VNof81lGXkOZ0fyB85OaCrOkWR6HuL
fJpKDOE0eNwIzA1jWfJwdzWac5rJ4Mm6KAngUp85HcA4IPkkd4a6bDsD1OOv5N6CtOzr1Fst9
IvKz+7oHTmLdgLMupxJ4c0dRd/kH3Nj38e+Z7rtAtLv+p/mZ9jWc7bcAJ6m9i3Ox1J+3cL4HG
RkG9BaRsqp6AueOuR04d2gddpc3QFWHpU6gqq8B/XBukEj93D2AcwEdEYnAaXJ8P4PlZrbtR+
DcPfW3O5+PTp+FXx+7dTh50OYeNF2Pc33pD5yz6XdI/xpGtku9wm2MXyKyCOdOj6D8uvdciMi
9QDdVDehI2WQ9EfkeeNA9Ws6uZT6Ic0vr45kWNoBzW5YxIcFtaz4fpx25Bs6tmKODGlSYU9UW
mZfK8mW+kd3LzO0sEZhQUgCnOaI6zun+FJy2YGNMBqxpyBhjwpxdLDbGmDCX65qGSpcurdWqV
Qt2GMYYk6usXLlyr6qW8Tcu1yWCatWqsWLFimCHYYwxuYqIbE9vnDUNGWNMmLNEYIwxYc4SgT
HGhDlLBMYYE+YsERhjTJjzLBGIyHsi8o+IrEtnvIjIKBHZIiJrROQSr2IxxhiTPi/PCCbiPPeFzC
uPVfj9AdTA+dZqeM8jMUYY0w6PPsdgaouFpFhlqYGRHzgPMudoyHrZN
zrycCarEJOGP3YWoWbcGNByZ5fMP5g/KKnJq/+2x7nunJQIR6Y1z1kCVKlWyJTjjIdv55Bz/u
I/SLms9dedUv2wpyl3D6/HPwYJsnraPtI9vywrBTATi5z2/PeCp6nhgPECjRo2sl7ysFIydsu
18co6yraHarXBh72BHYtKIj09i8OAlDBu2nNKlIxk7vj2FW9T0ZFnBTASxnPpM2UrAziDF4q2
cfAQcjJ2y7XyMyVSnTtOYP38bPXvW4dVX21CiRIRnywpmIpgBPCAiU3Ae9Hwox18fONsdek4+
AradsjE5xpEjCeTPn4eIiHz079+ERx5pRIcO1TxfrmeJQEQ+BtoApUUkFngWyA+gqm8Cc3CeK
7oFOA709CqWLLNtMhxYBSXqn9l0trM1xmRi/vw/6N17AbfdFs0LL7SkTZvsux7q5V1Dt2QyXo
H7vVq+Z0rUh/aLgh2FMSZE7N8fR79+i3j//fXUqlWSa689P9tjyHXdUAdFapPQ2ZwNGGNMOr7
+ejvdu89m3754BgxoxtNPNyMiIvt3y5YIMrNlPCzr47xObeIxxpgsULZsIapXL8a8eV2oX79s
0OKwRJAR3yTQ5C1r4zfGnBNV5f331/Pzz38zalQ76tYtw5IltyLi72767GOdzmUk9Q4hSwLGm
HP0xx8HufLKqfTsOY9Vq/YQF5cIEPQkAHZGkLmyrS0JGGPOWnJyCmPGrOLJJxeTJ48wdmx7+v
SpR548wU8AqSwRGGOMh/bujWPgwB9o3boyb77ZgSpVigY7pNNY01B6toz/94dgxhhzBhITk5k
4cR0pKUq5coVZsqQD7nqLXnpOkXLtBgMOxm4LGQUoYk1kYbb8z6xRQsYA7Wb/dFjKbDKSi8uLPI4ZO
6lTp+ksWLCNXr3qMmxYa4oX966TuKxmiSAtu0vIGBOgw4dPUKBAXiIi8uu/4DXUVdm+ggR2F+8uU
hhnTG7WGyMMWdhzpyt1Kkzkeee+weed+xGA1q0r58okAJYIjDHmjOzde5QBA76ncuW3WLhwB6VK
RbFkyS1oxvcMHFT0uEOv+ZKH6urjJ5R5TvtrPnXPBlHrXmt/vzNc8Scf2Vg/PwBcpxiDgJUNGX6q
DfKa6cPqzvPcsgmY1VafsA+Zw6pK/2jpbVX6a4xUA7M2zNqbKMPG6ysQZ0mQEYz0HI6PTXKaX
ZLQOQbV0Rk9pmbfMEB3SFDtTutyzzF7cxzIqsLzNtD47jrKNdVq0CuWu+tu2kA4FyFd2MVkdgHa
dNvsiIQqy7WyYZw7EFyD98upDu6aZnt/bGbJqIDAcRb2VD2EvzeUkZ/ChjRm3crCPuwTVSGcP
CqKLl5lVzlnKyvaPv1bdaSPkQzc5CuYcxPCwqUH7Mf9ZCtv6Lv7U5SZuEKg5mOYkSKaHgvKHLe
Gaqfk1YM3qa0HBMvk3mBVy2Tq9SDaxb5sDx98qFLMRSYpkk9HZhvHeE5mxYHVkAk4LKsSkTkhlg
3xcwdlXLf2dVE6qKw6zl6h7bqtgbpDV6mR09d8yBmKKQYhDdBaAZ7aQL5CP6yStl97rxxvq3tu
YQ2vBjWjG2S5OcDKB1eMw7mJxAaAm+XGWKL4iQpvAQsaiSJ0jHXQsaYq9G9IhH0hcdx8XoqC0
NJj+jP0yuAXAj7pHzQh7K6dJSsQw5Gfr+3dtE91ebSw04MBkEjpiHeXz2wvdkmxGiOVeIVbg
8mjRDl+1aWG0IvYTGDmkZqj7PamGfYz4xHi7PSkSxvmA+ugd56X8PWpjLW4BZFMV5VWCwW5UXm
nNpIxXQ==

```
jRxN4+OH/cdllkzl2LJF58zrTvn3VYIdl/LAzAmOMJzp1msbXX//JAw80YOjQlkRF2aNGcipL
BMaYLHPgQDwREXmJjMzPoEGXMmjQpbRoUSnYYZlMBNw0JCKFvQzEGJO7ffHFZqKjJzBo0BIAW
rSoZEkgl8g0EYjIpSKyAafjOESknoiM9TwyY0yusHv3Mbp0mU7nzjM477zCdOtmPcnnNoGcEY
zAeYDMPgBVXQ208jIoz6X+mMwYc07mzt1KdPQEZs3aytChLVm2rDsNGpQLdljmDAV0jUBVd6T
51V+yN+FkE/sxmTFZomrVojRoUJYxY9pRq1apYIdjzlIgZwQ7RORSQEWkgIg8ittMlKvZj8mM
OWMpKcro0T9zzz3zAYiOLs3XX3e1JJDLBZII+gL34zx4Phbn2cL3eRiTMSYH2rRpP61aTeHBB
//Hjh1HrJO4EBJI09BFqtrd9w0RuQz4wZuQjDE5SWJiMsOHr2Dw4CUUKpSfiROvokcP6yQulA
RyRvBGgO8ZY0LQgQPxDBu2nOuvv4ANG3pyxx11LAmEmHTPCESkOXApUEZE+vmMKgrYs+OMCWH
x8Um8995a+vatT9myhVmz5g4qVYoKdljGIxk1DRUAirhlfD8Bh4EuXgZljAme77+P5e6757N5
8wFq1ixJ+/ZVLQmEuHQTgap+C3wrIhNVdXs2xmSMCYIjRxJ48snFjBmzimrVirJgQRfrJC5MB
HKx+LiIDANigIjUN1X1cs+iMsZku06dpvHNN3/y3/9ewpAhLShSxDqJCxeBJIKPgE+A63BuJb
0D2ONlUMaY7LF/fxwREfkoVCg/zz9/GSItaN68QrDDMtkskLuGSqnqu0Ciqn6rqncBzTyOyxj
jsalTN1G79r+dxF16aUVLAmEqkESQ6P7fJSLXikgDIPd2KWj9DJkwt2vXUW66aTo33zyTypWj
6N69drBDMkEWSNPQEBEpBjyC8/uBosDDXgblKetnyISx2bN/57bb5hAfn8zLL7eiX79G5MtnD
yoMd5kmAlWd5b48BLSFk78szr2snyETps4/vziNG5/H6NHtqFmzZLDDMTlERj8oywt0xeljaJ
6qrhOR64CngEigQfaEaIw5W8nJKYwe/Qtr1uzh3XevonbtUixYcHOwwzI5TEZnBO8ClYFlwCg
R2Q40B/qr6rRsiM0Ycw42bNhLr14L+PHHnVxzzfTXXi45OIiLCn05rTZfSpaARcrKopIhIB7AUu
VNXd2ROaMeZsJCQk8eabP/Piiz9SuHA+Jk5sR/ful9jU3yZdAn3kOi1FNQVAVeOBTc/aQAAAA==
dkkIltEpH86ZdqIyCoRWS8idjuPMefo4MF4RoxYyY03XsiGDXfSvXu0JQGToYzOCGqJyBr3tQ
AXuMMCqKpenNGM3WsMY4AOOM8xWC4iM1R1g0+Z4sBY4CpV/VNEyp59VYwJX3Fxibz77lruu68
BZcsWZu3aO6lQoUiwwzK5REaJ4FxvLm4CbFHVrQAiMgXoCGzwKXMr8IWq/gmgqv+c4zKNCTuL
F++gV68F/PbbAWrXLkW7dlUtCZgzkm7TkKpuz+gvgHlXBHb4DMe67/mqCZQQkUisisiecIIsiEQGvg
Yj0FpEVIrJizx7r3cIYgMOHT3DffV/RuvUnJCWlsHDhzbRrZ53EmTPn5S0E/hol1c/yGwLtcG
5J/VFElqrq5lMmUh0PjAdo1KhR2nkYE5Y6dZrGokU7U7+L//a8jzz19G4cLWSZw5O14mglic209
TVQJ2+imzV1WPAcdEZDFQD9iMF1K7lyjb2pPZG+O1vXuPU6hQfgoVys8LL7REBJo1s/6BzLkJ
6LflIhIpIhed4byXAzVEpLqIFAC6ATPSlJkOtBTdZD5QTOupcwuZSqMmXKr9SuP
YFnn3UeF968eQVLAiZLZJoIROR6YBUwzx2uLyJpd+inUdUk4AFgPs7O/VNVXS8ifUkr1tmoz
vfNTg/XHtHVdedZV0CY91LmFzmr7+O0KnTNG65ZRbVqxejR4+YYIdkQkwgTUODcO4AWgSgqqt
EpFogM1fVOcCcNO+NS/N6ODA89nZE4GDAskPkZE25mzfqd7t1nk5iYwvDhrXn44YWJmsFUgiSFLV
Q/aDFGOy34UXFufSSyvwxhvtuPDCEsEOx4SoQA4t1onIrUBUgR0SMJ4KJBE8iPO84hPAZJzuqB/2MCZjwtL69Xu57LKP6ddvEXv3xhEfnx
7jzzrkA1KpVirlzu1gSMJ4KJBE8iPO84hPAZJzuqB/2MCZjwtL69Xu57LKP6ddvEXv3xhEfnx
TskEyYCKRp6CJVHQAM8DoYY8JRQkIyL730E0OGLKVYsYJMnnwt3brVsv6BTLYJJBG8JiLlgc+
AKaq63uOYjAkrBw/GM2rUL9x880WMHNmWMmUKBTskE2YybRpS1bZAG2APMF1orI014HZkwo
O348kddfX0lycorbSdwdfPTRtZYETFAEdB+aqu5W1VFAX5zfFAzZsF1dzXmXjQtk33/xJ3boTefjhb
1i0yOmOq3x56yTOBE8gPyirLSKDRGQdMBs3dOgEffos4PLLP0VE+OabLbz9XaYm/5ppFqJQUZjhUBd
kRArlGMAH4GLhCVdP2FWSMCVCnTtNYvDiWxx5rzKBBzSlUyEYON8F/nhs4PLLP0VE+OabrtZJnM
kRArlGMAH4GLhCVdP2FWSMCVCnTtNYvDiWxx5rzKBBl1KoUP5gh2QMEEAiUNVm2RGIMaFoz57
jFC7sdBL34ostyZtXaNy4fLDDMuYU6TYNicin7n/XisgV8hbHM7/jfAKBv+1vtIhOMIGSJi8+Hx
WBIwOVJGZwT/df9flx2BGBMqYmOPcO+9XzFr1laaNi3PnfWCXZIxmQooyeU7XJf3ufn6WT3Z
U94xuQuM2ZsITp6Av/735+MGNGWH364hZiY0sEOy5gMBXL7aAc/712d1YEYEwpq1ixBixYVWb
v2Tusp1OQa6TYNici9OEf+56e5JhAF/OB1YMbkBklJKYwcuZI1a/bwwQfXUKtWKebM6RzssIw
5IxldI5gMzAVeBPr7vH9EVfd7GpUxucCaNXu4++4vWRVKe/VQHzHHwXcnY1nXpqJFxjD9u/qiK4
3SYi96cdISIlLRmYcHXiRBJDh/7E0KE/UbJkBJ9+ej1dutS0TuJMrpXZGcF1wEpAAd9PuQLne
xiXMTnW4cMJjB27iltuqcWIWEpVSoy2CEZc07STQSqep37v3r2hWNMznTsWALjx6/hoYcuoU
yZQqxbdyflyhUOdljGZIlA+hq6TEQKu69vE5HXRKSK96EZkzN8/fV26/379n79FvHtt7EAlgR
MSAnk3rZxwHERqczNdmwHPvQ0KmOKmNygIMH4+nVaz2TzAOnVaz7t239Gvnx5+Pbb/3D55Xbk
CLyuqu+KyB1eB2ZMsN1443RmzdrG6qu35OmnLyMyMm+wQzLmrATSERwRkceB24CWIpIXsDkDjclt
yRYVmEVsFwhXobAf2BtfWCEpkVk8Fb/ROOExhHMsDxdXTZ2xgqVqxK7mGsHyUuSUVBTpTqtcU
f33MYoUyU/hwgV46aVW5MsnNGx4XrDDMsZTgTQN/QfnwfV3qepuoCIwzNOojMlmqsHH64nOn
oCzz67BICmTctbEjBhIZBHVe4GPgKKich1QLyqfuB5ZMZkkz//PMy1135bjx5zueiiktx9d91
gh2RMtgrkrqGuwDLgZqAr8JOIdPPS6MGOyw/TpW4iJmcDixbGMGnU5333Xjdq1+1SwU7LGOyVSDX
CAYAjVX1HwARKQMsBKZ6GZkxXklOTiFfvjz06FGHpk3LU7dumWCHZMw5C6TXUL7UJODaF+B6xuQo
ybHSUpK4eWXf+L22cAcNFFJZk58yZLAiasBXJGME9E5uM5cAmRnmGYHc+yGRxMKBrrnhRo7LhnX5
9//psbb6xhncQZ4wrkmcWPichNQAuce/+NV9UvPI/MmCyQmJjM0KHLGDx4KY8+2piBA5ty442x
cAl3bfvrtZ92WhtKlrRO4ozxlVb/3vA
LKAzTg+kb5zpzEXkKhHZJCJbRKR/BuUai0iKiHTxJfIlrPDBjK+6wh0oNJjT9yzvAAAAAAAAAAAA
yZQqxbdyflyhUOdljGZIlA+hq6TEQKu69vE5HXRKSK96EZkzN8/fV26tZ9n379FvHtt7EAlgR
MSAnk3rZxwHERqQc8DmwHPvQ0KmNygIMH4+nVaz7t239Gvnx5+Pbb/3D55XYMZEJPoA+vVxHp
CLyuqu+KyB1eB2ZMsN1443S++y6WJ55owjMPjcQ+yJpNG6OrfgAsw4GrxKiHi2wdDMsZTgTQN/QfnwfV
eeYIVbOTYaNNCzPG+w4VqmmTmGbpdcPKSUkKeC/p5Y6+DXaGD8f/XvYz7cpffJLqPc+ErCcGFvsQ
GYq1SwU7LGOyVSDX
CAYAjVX1HwARKQMsBKZ6GZgxXlJVRIRatUrSpk1l3nijbjQ+BjJWdqvZbvacmIrMoPdXr3NmF4d
ybHSUpK4eWXf+L22+cAcNFFJZk58yZLAiasBXJGME9E5uM8txici8dzvAvJGG+sXv0Pd901n5
9//psbb6xhncQZ4wrkmcWPichNQAuc/obGq+qXnkdmTBaJj09z1BvoLLDD8Tq5sixTMyQ4a+ZZBZ
msMMyJsfI6HkENYDhwAXAWuBRVf0ruwIzwP2rz2WhtKlrRO4ozxlVFb/3vA
LKAzTg+kb5zpzEXkKhHZJCJbRKR/BuUai0iy3Y1kssrRowkMH76c5OQUupYLA
sb4kVHTUJSqvu2+3iQiP3s/ljEoBcRGao6gY/5V4G5p/J/I1Jz4IF2+jdewF//n
mYhg3L0bZtFcqUKRTssIzJsTI6I4gQkQYicQIXAJEphnOTBNgi6puVdUEYArQ0U+NjVpYrRRU+5B4HPgX/
8jDMmYPv3x9Gz51yuvHIqERH5+O67o1bv1lqjCyKR7Aj8XnmTSOwOWLsisMNnOBZo
6ltARCoCN7rzapzejESkN9AboEh1FqG38cTo/T3z3Vn3Te5UiT2e6wkTR/T2SPfC4n2Pkc
```

```
7l9mmZ4JPCEqiZn9Jg/VR0PjAdo1KhR2nmYMLZ79zGiopxO4oYNa02BAnmpX79ssMMyJlfx8o
dhsUBln+FKwM40ZRoBU0RkG9AFGCsinTyMyYQIVWXixHVER09g4MAfAGjSpLwlAWPOgpfnzsu
BGiJSHfgL6Abc6lvA9zGYIjIRmKWq0zyMyYSAbdsO0afPVyxYsI0WLSrSu3e9YIdkTK7mWSJQ
1SQReQDnbqC8wHuqul5E+rrj3/Rq2SZ0ffnlb9x++xxEYPTodtx7b33y5Em/WdEYk7lME4E4j
ffdgfNV9Tn3ecXnqeqyzKZV1Tmk6Y4ivQSgqncGFLEJS6mdxMXElKJ9+6q8/npbqla1/oGMyQ
qBXCMYCzQHbnGHj+D8PsAYzyUmJjN06FK6d58NQM2aJZk2rZMlAWOyUCCJoKmq3g/EA6jqAaC
Ap1EZA/z88980afIRAwZ8T3KycuJEUrBDMiYkBZIIEt1f/yqcfB5BiqdRmbAWF5fIk08upkmT
SezefYwvv+zIJ59cT8GC9rsAY7wQyDdrFPAlUFZEXsC5zfNpT6MyYe3YsUTefXctd9wRw/Dhb
ShRIiLYIRkT0gLphvojEVkJtMP5kVgnVd3oeWQmrBw5ksC4cat45JFGlC7tdBJXurT1D2RMdg
jkrqEqwHFgpu97qvqnl4GZ8DFv3h/06bOAHTuO0KTJebRpU8WSgDHZKJCmodk41wcEiACqA5u
AGA/jMmFg3744+vX7hg8+2EDt2iX54Ydbad68QrDDMibsBNI0VNd32O15tI9nEZmwcdNN01my
ZCfPPNOMAQOa2cVgY4LkjL95qvqziKTbU6gxGdm16yhRUQUoUqQAw4c7ncTVq2f9AxkTTIFcI
+jnM5gHuATY41lEJiSpKhMmrKNfv0XcdVcdXnutLY0blw92WMYYAjsjiPJ5nYRzzeBzb8IxoW
jr1oP06fMVCxdup1WrSvTta53EGZOTZJgI3B+SFVHVx7IpHhNivvhiM7ffPoe8efMwblx7eve
uZ53EGZPDpJsIRCSf24NoII+lNOYUqZ3E1a1bhquuqs7IkW2pXLlosMMyxviR0RnBMpzrAatE
ZAbwGXAsdaSqfuFxbCYXSkhI5pVXlrF+/T4mT76WGjVK8Pnn/h5VbYzJKQK5RlAS2IfzXOHU3
xMoYInAnGLFit3cffd81qzZQ7dutUhISLZbQo3JBTL6lpZ17xhax78JIJU9N9icFBeXyLPPLu
HVV1dw3nmFmT69EzfccGGwwzLGBCijRJAXKEJgD6E3YezYsUQmTlzH3XfX5ZVXWlG8uHUSZ0x
uklEi2KWqz2VbJCZXOXz4BGPHruKxxxpTunQhNm68i1KlIoMdljHmLGSUCOweP+PX7Nm/07fv
QnbuPEqzZuVp06aKJQFjcrGMHkzTLtuiMLnCnj3H6d59Ntdd9yXFihVgyZJbadOmSrDDMsaco
3TPCFR1f3YGYnK+zp1nsHTpTgYNupQnn2xKgQJ5gx2SMSYL2L19JkN//XWEYsUKUqRIAUaMaE
PBgnmpU6dMsMMyxmShQJ5ZbMKQqvL222uIjp7AwIE/ANCw4XmWBIwJQXZGYE7z++8Hueee+Xz
zzQ7atq3M/fc3CHZIxhgPWSIwp5g6dRM9eswlf/48jB9/Bb161UXEbiAzJpRZIjDAv53E1atX
lmuvPZ8RI9pSqVJU5hMaY3I9u0YQ5hISkhk8eAndus1CValRowSffXaDJQFjwoglgjC2bNkuG
jb8kEGDlpAvXx4SEpKDHZIxJggsEYSh48cTefTRRTvPpkDB+KZOfNGPvroWusp1JgwZd/8MB
QXl8SkSRvo3ftiXn65FUWLFgx2SMaYIPL0jEBErhKRTSKyRUT6+xnfXUTuH9LRMQeZuuRQ4d
O8MILS0lKSqFUqUg2bryLceM6WBIwxniXCNznHY8BrgaigVtEJDpNsT+Aqlqp6MfA7e/reMLZ
zJm/n/xh2PffxwJQooR1FW2McXh5RtAE2KWqP1U1AZgCnPPLMQlVdoqoH3MGlZoFI7iX
vuuct44okm1kmcMcYvLxNBwE82E5G2OImghb/xqjoet9moUaNG9nS0DMTGHqF4caeTuJEj21K
wYF5iYkoHOyxjTA7mZdNDHOyZ7gSsDNtIRZ6gRZZxgeNuMOaAeGQOqv9xTlfmbP+mZWr9Ec4k
JZeUsyRgjMmUl2cEy4EaIlJVRAoA3YDpvgVEpAowFeihqr94GIsxxph0eHZGoKpJItIHmAPkBc
aq7heR3u74UcBgoDQwQkQAklQ11quYjDHGnEpyWs9psbGxGhcXF+wwjDEmRxGRFeldaFvPosYYE+
YsERhjTJizRGCMMWHOEoExxoQ5SwTGGBPmLBEYY0yYy3HNR0VkH/DHGU5eBtifheHkBLbO4cHWOT
yczTpfoKpl/Y3IcYngbIhIXLjdsGbrHB5sncODV+tsVUPGGBPmLBEYY0yYC7dEMDrYAQSBrXN4sH
UOD56sc1hdIzDGGHOqcDsjMMYYk4YlAmOMCXNhkwhE5CoR2SQiW0Skf7Dj8ZqIjBWRvSKyLtixZB
cRqSwi80Rko4isF5GHgx2T10SkkIgsE5HV7jo/F+yYsoOI5BWRlSLyVbBjyQ4isk1E1orIKhHJ8n
74w+IagYjkBX4B2gM7cB6jeYuqbghqYB4SkVZAPPChql4c7Hiyg4hUBCqq6k8iUgxYAdyYy/ezAE
VUNV5E8gM/Ag+r6pIgh+YpEekHxALFVfW6YMfjNRHZBsSqqic3zYXLGUETYIuq/q6qCcAUoFOQY/
KUqi4ADgY7juykqrtV9Sf39b/ARqBScKPyljri3bf53b9c/etORKKAa4Exw44ltwiXRFAJ2O7zfg
e5/AAR7kTkQqA+sDTIoXjOrSZZBewF5qpqbl/nocATQEqQ48hOCnwrIitEpFdWzzxcEoH4GZarfz
WFMxEpCnwOPKKqR4Idj9dUNVlV6wFRQBMRybVVgSJyHbBXVVcEO5Zs1lxVGwBXAw+4Vb9ZJlwSwQ
6gss/7KGBXkGIxHnLryT8HJqrq1GDHk51U9R9gPnBVcCPxVHOgo1tnPgW4XEQmBDck76nqLvf/Xu
ALnOruLBMuiWA5UENEqopIAaAbMD3IMZks5l44/QDYqKpvBDue7CAiZUWkhPs6EmgH/BzUoDykqg
NUNUpVL8T5Hv9PVW8LclieEpEibuMHRKQI0AHI0taAYZEIVDUJ6APMwbmA+Kmqrg9uVN4SkcnAYq
CWiOwQkXuCHVM2aA70wPmVuMr9uybYQXmsIjBPRNbg/OCZq6ph0aQyjJQHfhSR1cAyYKaqzs7KBY
RF81FjjDHpC4szAmOMMemzRGCMMWHOEoExxoQ5SwTGGBPmLBEYY0yYs0RgTDpEJNmnKeqqrOy1VkQuDKdeYU
1oyxfsAIwJYf+5XToYk6vZGYExp8ntG/41t4+9lyJS3R1+gYh8LyJr3P9V3OHlReQLtz+g1SJymTurvCLyvtt
H0LfuHbnG5DtLBMakLzJN1VBXn3FHVLUJ8A5Ob5i4r/9UFRvAhP/G8Pe7O1DQ1+BOYDNd3hN4AhqtoE+Afo
7OmaGJMBu7PYmHSISLyqFvUzfBtwuar+7nZy95eqlhaR/TgPw0l0h+9W1TIisg+IUtUTPvO4EKe7hxru+ye
B/Kr6YvavmglnTgj8J/O+ttjPu3rmmvlU+09zIR8T+SQemznWOHiecfgnLPW+vx5jz8vu4e/ccfndn/3Bsu6v
PF3HqjdtDfOxRZlzX+B3aQaoFKDhqecNp9tX7uT6q6L0DMQVTfYIcW20wbVzcHl5H9+jX9cB92eRPvCBW4t2k
lqE9yHFGmOy0+ufxnxQw5i/sXyUpSAQCQOMIrl8H+5+9nkq4SyNxcQoSMbLP4XUtuvBlwG5gW4+uHOhWM0
+S2jdP1kxHL60HQoGv3Vh4/uBKGwGQnDzi6Wl7KuNxXYm0KGg6vWr+IVfEqCIVZc9rnE6yyGW7f5cMOdBfy3k
cAJAJTU95u6ZqfjLw9qP9eD7DCe0KuQqrs1eo6OGPsm0bhm0xUaRwzJdQGD1BPXWlHTJt3QnS9TKU22gy+6J
ozWtHjUwS0GY+R8g1C0IX0QhxJw+1TaAU92MK7NIF2U7Uy1U8XAtuqJ4/PjBMW6DtD7GO8mJdLmMDqUbXM4
vwJ69+UlejyC/UP0CfJ8HLXRY4nOkCoIrkxJN8DFlhpd1Y0YgRnA4Lfml6KF8hYrKKKk3JbB+Xw0Qk9WqRyzC
lQFTD86GbtcdCOyT6A7wt5+gQyVXSDl8pYgMbX3Po8CY8qTI4HGrnVVKFjk1rh2X1+jm6ilNFnG9mQC+XxrgA6
tmqdhDeoKqEZq6XP3w3vH4i2oeUvXp0aAyVKvZ5YdlN3umLm/TASeJU4mIOS0QGU7Ve3UFGMcB5ZMAXi6Z5
AAAnG7++ZU6P+tx4hPUPtjfTVOAI+ZCWAGEXMwOGDMqGdUZuP9nbjTuuYzD9kfOlbqfLm3qpPQeV4qKbLXV
gXHqP9SZ4rjIgGv1e79K3QV/ZKeMfgsKV2MYQ8h0mYp0rRBVjWmOH6tYcLQGCgnJw4ymDQIDlhMVtOkv0iKWS
EmvPSnxDTCbhoxLUJUoF/DJK4jd8b2XvL5PMm3JPUSm6ITmNTqmAxsL3WNukJdgC+t/9Y8JbPfl2p3lxsT7GWO
ddP5oBMS+oUGqfUZkXMyqA8wAM0jwk6cOPHoRnmvb4SJ8/gBgqVNOb6zuEKQSfrcH87bGvpbvNNlHrBCYyZL
EtWhfWy4jL9E7bPTA7XD1YhpANc/nvsbYqZ8BCJTRpQ6rt6ZCL7fC2xA8MeNomUHFwRtd1H6MfrSwsJo4PmO
JsH8NpXxDMKzYPxhMkOd/x9lBajGy1MtOGVMOzUrU6hT/ehxY3FU4zE0E3uIuCRcdGS4qXzDMBFBw8m8UFsD
StnTVWSPKuInPVbxy1DHHnhsoEfA9XgAwELIcIWeh3NYfQkhgjprRDjGMutphAz+V3AhhsCaBzjJrXqqF0uu
hTtzp0PTTIJUjQxYiPIh3fB+aepTqTYZuaQBoM8/cY6LQtgLXL7Bmq/XHD6yo6r9kzoj9vuhXuXa5/4/sZKVa
vQj7QkIymYIzIVuT7uc2pDo0azeuzHx3AR0gW4aILvm2T8ZzbtXu3Kf4qDBnqc9QzXhZ5nRU6MrQkU/tCYyOq
eM4aPiz9nBkEZsP/GAUgDYrSGLVrsCe1Z0dfqJwRqZI2MjnZPjd0ps7i1+/6ng7hZQzOmkGGHCaXFcsGlbjLq
uz6tn4C9bvc0aTw8lTuU9hCsW1q5zC7aJ1qgbDBqsVBw8mdxNC0zRWS1mPpeFbWNxjFWwWhhXvHu1wRjLQ9
e8O+5m0RqlSDuMGCSfPhEOWgCTCx7Q9qrUFsIyORdqnBd9JEC1UQK5MF5M9Z3fiLYqwJc6hlDfpG3ZNoxl5
Ej/5XtTqwGTlp8FnTwt5SdeHKmYONNUIcdyY6SLE7ZRA7IpQ4SqWxrkTnnQzTgsTO3E1jOjnlqLQ5X96bG39
vMqz2QxGNbhcGv+sYJOLqRMcDx8FPVSBOAwpUDqy7r3HnqbnuN7cHKmBMVqdkjt3gFSZd6A9vDXi9U4Q8jT
Ej8XhlLzlXhyJ3Y8tJiPBoTS54Rd5fBtXqWK4sMdOXQp2JrDBc2P+YXTT4gZd9V0tAPDchvo8jbqkmmC5SZfdA4
5mY3FYUJ6ciAvOtNd1oRJWkDeyx0Wo2amTxCcbL0E9Mhfix8n2xPIm+qCtAaeWQ5ns6E6O6ShqZ/BQ3Lp+66
NSLgFmfXWXrdzShtXDF9m3bXrhTFKlF9kEdmQoNuDTyrcuZsPT1H9TSYj26wz5qpOaHXnMuSHoJD6yS8Wd8+
xhIJXxyChTeKMQAF0Vy7Kqj3CSmQlSIzRfSDjnYNn9TWWzk7ZKMXc4AsfJqB1DLq6nWZrAdV6j0pMN7gTw6c
ZsyXtY7rzI7tKb8hZ8uwTAEEvHwVMUDgfkYYvHwyNN9SgP2XnYn9mMv2BKdZzHjX3sPTDZLfAkO4V5u6QSMkg
0rxkbcUmYmZGDQeC1JCB6Y3dZEPNuXQOJAF8Lk7W5Vz4NMcQGvx4ZUzqwrNUGCA0Ib6tUgT3zmGK3B7AJqO3
Y+iUq8LMApb5HhRm5DSfTu85bbLhJ8w1dn5t+YhkVvHf7vLzTGz1AfZz4N+dtu1i+1Hu7G8SE6ap4ZRDiFNYhZ
```

```
6lSJFihAVFUVMTAwrV67M8ng8SwQikhcYA1wNRAO3iEh0mmJXAzXcv97AOK/iYdtkdh4oyk3D
OtG160wqV45ixYrbaNmykmeLNMaYjBQrVoxRo0YxfPhwEhMT6d69O99//z0LFy4EnDOHhx56i
McffxyAxx57jKFDh7J582YAUlJSeO211845Di/PCJoAW1R1q6omAFOAjmnKdAQ+UMdSoLiIlP
cqoK7j7mfed/DKK61YurQ79epZFxHGmOBq0KAB9erVY8qUKURGRjJ9+nSGDBnCRRddRN26dWn
cuDEPPPAAABdffDEjR47kkltuoXbt2tSpU4ddu3adcwxeXiOoCOzwGY4FmgZQpiJwEpDf
GQNVqlQ5u2hK1GfMgEQiG/agZk3rJM4YEzxHjx49ZXjmzJnJX9etWzf5dxTTiT1113Hdddddl
6XxeJkI/F3Z0LMog6qOB8YDNGrU6LTxAWk4knrWRZAxxpzGy6ahWMD3KS2VgJbIunUcYYY4yHvE
wEy4EaIlJdRAoA3YAZacrMAHq4dw81Aw6p6rk3eBljTAZ8u2wINWdtN8+ahlQ1SUQeAOYDeYH
3VHW9iPR1x78JzAGuAbYAx4GeXsVjjDHgPLhl3759IdkVCCIIs5oOsltmbFRo0aa9j5b
Y4wJVLg+oUxEVqpqI3/ThO0vi40x4Sl//vxn9PSucGB9DRljTJizRGCMMWHOEoExxWW
ET2ANvPcvLSwN4sDCc3sDqHB6tzeDiXOldV1TL+RuS6RHAuRGRFelfNQ5XVOTxYncODV3W2pi
FjjAlzlgiMMSbMhVsiGB/sAILA6hwerM7hwZM6h9U1AmOMMacLtzMCY4wxaVgiMMSbMBeSiUB
ErhKRTSKyRUT6+xkvIjbLKHb9GRC4JRpxZKYA6d3frukZElohIvWDEmZUyq7NPucYikiwiXbIz
Pi8EUmcRaSMiq0RkvYh8m90xZ7rezEWkfdE5B8RWZfO+Kzff6lqSP3hdSvwMVAMKAKuBtDX
Hn9O3A+UABYDUSnKXMNMBfnCWnNgJ+CHXc21PlSoIT7+upwqLNPuf/hdHneJdhxZ8N2Lg5sAK
q4w2WDHXc21Pkp4GX3dRlgP1Ag2LGfQ51bAZcA69IZn+X7I2gCbFHVraqaAEwBOqYp0xB
4QB1LgeIiUj67A81CmdZZVZeo6gF3cCnO0+Bys0C2M8CDwOf/AP9kZnEcCqfOtwBeq+ieAqub2
egdSZwWixHm4QBGcRJCUvWFmHVdjFOH9GT5/isUE0FFYIfPcKz73pmWyU3OtD534xxR5GaZ1
llEKgI3Am9mY1xeCmQ71wRKiMgiEVkpIj2yLTpvBFLn0UBtnMfcrgX+q6op2RNeUGT5/isUn0
fg75FDae+RDaRMbhJwfUSkLU4iaOFpRN4LpM4jgSdUNTlEnkQVSJ3zAQ2BdkAhIEZdkAk8KOILFXVzV4
H55FA6nwlsAq4HLgA+EpEvlPVx7HFixZvv8KxUQQC1T2Ga6Ec6RwpmVyk4DqIyIXA+8AV6vq
vmyKzSuB1LkRMBa0QkSVWnZUuEWS/Qz/ZeVT0GHBORxUA9IlLcmgkDq3BN4SZ0G9C0i8
gdQC1iWPSFmuyzff4Vi09ByoIaIVBeRAkA3YEaaMjOAHu7V92bAIVXdld2BZqFM6ywiwiVYAvgN
tz8dGhr0zrrKrVVbWaqlYDpgL35eIkAIF9tqcDLUUkn4gUApoCG7M5zqwUSJ3/xDkDQkTKARc
BW7M1yuyV5fuvkDsjUNUkEXkAmI9zx+8F7qrpeRPq649/EuYPkGmALcBzniCLXCrDOA4FSwj3
CDlJc3HPjQHWOaQEUmdV3Sgi84A1QArwrjVr6vQ0xNwhwOz8PTBSRtTjNJk+oaq7tnlpEPgbaA
KVFJBZ4FsgP3u2/rIsJY4wJc6HYNGSMMeYMWCIwxpgwZ4nAGGPCnCUCY4wJc5YIjDEmzFkiMD
mS21voKp+/ahmUPsr+bxTMpX19
ErsmKZZvQZbePmhxJRI6qapGsLpvBPCYCs1R1qohcAQxX1YvPYX7nHFNm8xWR94Hnu
BBqp6gNZHYsJHXZGYHIFESkl+7R+trReS0nkZFpLyILPY5Ym7pvn+FiPzoTvuZiGS2g14MX
OhO28+d1zoRedh+cGRySsiw0Mv+kIo1E5CUg0o3jjI3jfcUff/J75H6O6ZSGcRyRyssiw0
RkuTh9zvZmcJYLX8iNvZmIg0Eec5E7+4/y9yf4n7HPAfN5b/uLG/5y7nF3r0YShYPe9bX/25+8
PSMbpSGwV8CXOr+CLuuNK4/yqMvWM9qj7/xFggPs6LxDlll0MFHbffwIY6Gd5E3GfVwDcDPyE
03nbWqAwTvfG64EGQGfgbZ9pi7n/F+F8+5ydk59i6XFGtrPwI3twjA65OgCr7i
fOoT/0+A65yh4sC+dzX7YHP3dd3AqN9ph8K3Oa+Lo7T1HB8h1MWFCRpyq1k8dEJ
H8wFARaYXTdUJFoByw22ea5cB7btlpqrpKRFoD0cAPbtcREji/IPGEYHKL7jhPn2qoqokisdAbeZ5cuWUPyDJYJxmxwCzQJ9dvzhXqRlnw0l4fX6Y5pzyZPhtgbRXsamp7ttJxBPAWl3ZS
uiMgXQEtgHjBcRF7GaUTA2GaU767gzqNRcYJSIFgauAxaoa7zzZjHnzHXSz/PkWtGFAD+CPN9JEisgqoWhFL
vvIp/76I1MDpiTJ/Osu/ArhORPq5wyFAFdzHtEl9eAmFR6uoeyX/KuxRXN94ozA1Z7Fy7yvUVXrKsTuByjJ6XpACq0A0pwyEAMhNzqANNyrPsi8RRMyMIVkWabt0KTcc1T
8W1wIciMgw4AHylqrcEsIzHVHVq1jxaEzzqqc1nhmDbE3JWFiSaqnuYgFMyMY3LLvHpnF3wLQKwrLZ5NvdqZg
CPUxcHPKiq8zOZRZyq1heRYsAs4H5gFE5/i6+J6Bu2fPeb8rmnrKFg9J2QnlLNNNc3sUnIRa59PAS/5
xk0BboGraAiJS1S3zNvAuzuP+lgKXiUhm38hEaZZZA53caQrjNOGkjaq4M4aFbi4k6Z4
rUT3zMSfKTgdhbXE6UwN9/+9qdOISE13mX6p6iHgIeBRd5Bvmbm1NSw7FmfscUH9HXPnqcn6JtYmzhL5WJFsNgGnSzabmtmkqM57gFRxOiIqO
xFpkN4yTPiwRGByi4+ARiKyAufs4Fc/ZdoAq0TkF5x2/NdVdS/Ol8bQm3hBIsJsSIREZQ4I1qV
rBMpxrBu+o6i9AXWCZ20QzABjiZ/LxwJrUi8VpLMB5Lu1CdR6/CM5zIjYAP4vz0PK3yOSM3Y1
lNU7XzK/gnJ38gHP9INU3QHTqxWKcM4f8bmzr3GET5uz2UWOMCXN2RmCMMWHOEoExxoQ5SwTG
GBPmLBEYY0yYs0RgjDFhzhKBMcaEOUsExhgT5v4fRmS68wGKsaUAAAAASUVORK5CYII=\n",
     "text/plain": [
      "<Figure size 432x288 with 1 Axes>"
     ]
    },
    "metadata": {
     "needs_background": "light"
    },
    "output_type": "display_data"
   }
  ],
  "source": [
   "plot_roc_curve(fpr,tpr)"
   ]
  },
  {
   "cell_type": "code",
   "execution_count": null,
   "id": "d6b6fe87",
   "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
}
```