```
{
"cells": [
 {
  "cell_type": "code",
  "execution_count": 1,
  "metadata": {},
  "outputs": [],
  "source": [
   "# Importing libraries\n",
   "\n",
   "from __future__ import print_function\n",
   "import pandas as pd\n",
   "import numpy as np\n",
   "import matplotlib.pyplot as plt\n",
   "import seaborn as sns\n",
   "from sklearn.metrics import classification_report\n",
   "from sklearn import metrics\n",
   "from sklearn import tree\n",
   "import warnings\n",
   "warnings.filterwarnings('ignore')"
  ]
 },
 {
  "cell_type": "code",
  "execution_count": 2,
  "metadata": {},
  "outputs": [],
  "source": [
   "df = pd.read_csv('../Data-processed/crop-recommendation.csv')"
 ]
},
{
 "cell_type": "code",
 "execution_count": 3,
 "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>N</th>\n",
"      <th>P</th>\n",
"      <th>K</th>\n",
"      <th>temperature</th>\n",
"      <th>humidity</th>\n",
"      <th>ph</th>\n",
"      <th>rainfall</th>\n",
"      <th>label</th>\n",
"    </tr>\n",
"  </thead>\n",
"  <tbody>\n",
"    <tr>\n",
"      <th>0</th>\n",
"      <td>90</td>\n",
"      <td>42</td>\n",
"      <td>43</td>\n",
"      <td>20.879744</td>\n",
"      <td>82.002744</td>\n",
"      <td>6.502985</td>\n",
"      <td>202.935536</td>\n",
"      <td>rice</td>\n",
"    </tr>\n",
"    <tr>\n",
"      <th>1</th>\n",
"      <td>85</td>\n",
"      <td>58</td>\n",
"      <td>41</td>\n",
"      <td>21.770462</td>\n",
"      <td>80.319644</td>\n",
"      <td>7.038096</td>\n",
"      <td>226.655537</td>\n",
"      <td>rice</td>\n",
"    </tr>\n",
"    <tr>\n",
"      <th>2</th>\n",
"      <td>60</td>\n",
"      <td>55</td>\n",
"      <td>44</td>\n",
"      <td>23.004459</td>\n",
"      <td>82.320763</td>\n",
"      <td>7.840207</td>\n",
"      <td>263.964248</td>\n",
"      <td>rice</td>\n",
"    </tr>\n",
"    <tr>\n",
"      <th>3</th>\n",
"      <td>74</td>\n",
"      <td>35</td>\n",
"      <td>40</td>\n",
"      <td>26.491096</td>\n",
"      <td>80.158363</td>\n",
"      <td>6.980401</td>\n",
"      <td>242.864034</td>\n",
"      <td>rice</td>\n",
"    </tr>\n",
"    <tr>\n",
"      <th>4</th>\n",
"      <td>78</td>\n",
"      <td>42</td>\n",
       "      <td>42</td>\n",
       "      <td>20.130175</td>\n",
       "      <td>81.604873</td>\n",
       "      <td>7.628473</td>\n",
       "      <td>262.717340</td>\n",
       "      <td>rice</td>\n",
       "    </tr>\n",
       "  </tbody>\n",
       "</table>\n",
       "</div>"
      ],
      "text/plain": [
       "    N   P   K  temperature   humidity        ph    rainfall label\n",
       "0  90  42  43    20.879744  82.002744  6.502985  202.935536  rice\n",
       "1  85  58  41    21.770462  80.319644  7.038096  226.655537  rice\n",
       "2  60  55  44    23.004459  82.320763  7.840207  263.964248  rice\n",
       "3  74  35  40    26.491096  80.158363  6.980401  242.864034  rice\n",
       "4  78  42  42    20.130175  81.604873  7.628473  262.717340  rice"
      ]
     },
     "execution_count": 3,
     "metadata": {},
     "output_type": "execute_result"
    }
   ],
   "source": [
    "df.head()"
   ]
  },
{
 "cell_type": "code",
 "execution_count": 4,
 "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>N</th>\n",
     "      <th>P</th>\n",
     "      <th>K</th>\n",
"      <th>temperature</th>\n",
"      <th>humidity</th>\n",
"      <th>ph</th>\n",
"      <th>rainfall</th>\n",
"      <th>label</th>\n",
"    </tr>\n",
"  </thead>\n",
"  <tbody>\n",
"    <tr>\n",
"      <th>2195</th>\n",
"      <td>107</td>\n",
"      <td>34</td>\n",
"      <td>32</td>\n",
"      <td>26.774637</td>\n",
"      <td>66.413269</td>\n",
"      <td>6.780064</td>\n",
"      <td>177.774507</td>\n",
"      <td>coffee</td>\n",
"    </tr>\n",
"    <tr>\n",
"      <th>2196</th>\n",
"      <td>99</td>\n",
"      <td>15</td>\n",
"      <td>27</td>\n",
"      <td>27.417112</td>\n",
"      <td>56.636362</td>\n",
"      <td>6.086922</td>\n",
"      <td>127.924610</td>\n",
"      <td>coffee</td>\n",
"    </tr>\n",
"    <tr>\n",
"      <th>2197</th>\n",
"      <td>118</td>\n",
"      <td>33</td>\n",
"      <td>30</td>\n",
"      <td>24.131797</td>\n",
"      <td>67.225123</td>\n",
"      <td>6.362608</td>\n",
"      <td>173.322839</td>\n",
"      <td>coffee</td>\n",
"    </tr>\n",
"    <tr>\n",
"      <th>2198</th>\n",
"      <td>117</td>\n",
"      <td>32</td>\n",
"      <td>34</td>\n",
"      <td>26.272418</td>\n",
"      <td>52.127394</td>\n",
"      <td>6.758793</td>\n",
"      <td>127.175293</td>\n",
"      <td>coffee</td>\n",
"    </tr>\n",
"    <tr>\n",
"      <th>2199</th>\n",
"      <td>104</td>\n",
"      <td>18</td>\n",
"      <td>30</td>\n",
"      <td>23.603016</td>\n",
      "     <td>60.396475</td>\n",
      "     <td>6.779833</td>\n",
      "     <td>140.937041</td>\n",
      "     <td>coffee</td>\n",
      "   </tr>\n",
      "  </tbody>\n",
      "</table>\n",
      "</div>"
     ],
     "text/plain": [
      "         N    P   K  temperature   humidity        ph    rainfall   label\n",
      "2195  107  34  32    26.774637  66.413269  6.780064  177.774507  coffee\n",
      "2196   99  15  27    27.417112  56.636362  6.086922  127.924610  coffee\n",
      "2197  118  33  30    24.131797  67.225123  6.362608  173.322839  coffee\n",
      "2198  117  32  34    26.272418  52.127394  6.758793  127.175293  coffee\n",
      "2199  104  18  30    23.603016  60.396475  6.779833  140.937041  coffee"
     ]
    },
    "execution_count": 4,
    "metadata": {},
    "output_type": "execute_result"
   }
  ],
  "source": [
   "df.tail()"
  ]
 },
 {
  "cell_type": "code",
   "execution_count": 5,
   "metadata": {},
   "outputs": [
    {
     "data": {
      "text/plain": [
       "17600"
      ]
     },
     "execution_count": 5,
     "metadata": {},
     "output_type": "execute_result"
    }
   ],
   "source": [
    "df.size"
   ]
  },
  {
   "cell_type": "code",
   "execution_count": 6,
   "metadata": {},
   "outputs": [
    {
     "data": {
      "text/plain": [
       "(2200, 8)"
      ]
     },
    "execution_count": 6,
    "metadata": {},
    "output_type": "execute_result"
   }
  ],
  "source": [
   "df.shape"
  ]
 },
 {
  "cell_type": "code",
  "execution_count": 7,
  "metadata": {},
  "outputs": [
   {
    "data": {
     "text/plain": [
      "Index(['N', 'P', 'K', 'temperature', 'humidity', 'ph', 'rainfall', 'label'], dtype='object')"
     ]
    },
    "execution_count": 7,
    "metadata": {},
    "output_type": "execute_result"
   }
  ],
  "source": [
   "df.columns"
  ]
 },
{
 "cell_type": "code",
 "execution_count": 8,
 "metadata": {},
 "outputs": [
  {
   "data": {
    "text/plain": [
     "array(['rice', 'maize', 'chickpea', 'kidneybeans', 'pigeonpeas',\n",
     "       'mothbeans', 'mungbean', 'blackgram', 'lentil', 'pomegranate',\n",
     "       'banana', 'mango', 'grapes', 'watermelon', 'muskmelon', 'apple',\n",
     "       'orange', 'papaya', 'coconut', 'cotton', 'jute', 'coffee'],\n",
     "      dtype=object)"
    ]
   },
   "execution_count": 8,
   "metadata": {},
   "output_type": "execute_result"
  }
 ],
 "source": [
  "df['label'].unique()"
 ]
},
{
 "cell_type": "code",
 "execution_count": 9,
 "metadata": {},
 "outputs": [
  {
   "data": {
    "text/plain": [
     "N                int64\n",
     "P                int64\n",
     "K                int64\n",
     "temperature    float64\n",
     "humidity       float64\n",
     "ph             float64\n",
     "rainfall       float64\n",
     "label           object\n",
     "dtype: object"
    ]
   },
   "execution_count": 9,
   "metadata": {},
   "output_type": "execute_result"
  }
 ],
 "source": [
  "df.dtypes"
 ]
},
{
 "cell_type": "code",
 "execution_count": 10,
 "metadata": {},
 "outputs": [
  {
```

```
"data": {
 "text/plain": [
  "muskmelon      100\n",
  "kidneybeans    100\n",
  "papaya         100\n",
  "pigeonpeas     100\n",
  "blackgram      100\n",
  "cotton         100\n",
  "mothbeans      100\n",
  "mungbean       100\n",
  "watermelon     100\n",
  "orange         100\n",
  "mango          100\n",
  "banana         100\n",
  "rice           100\n",
  "pomegranate    100\n",
  "chickpea       100\n",
  "apple          100\n",
  "jute           100\n",
  "grapes         100\n",
  "lentil         100\n",
  "coffee         100\n",
  "maize          100\n",
  "coconut        100\n",
  "Name: label, dtype: int64"
 ]
},
"execution_count": 10,
"metadata": {},
    "output_type": "execute_result"
   }
  ],
  "source": [
   "df['label'].value_counts()"
  ]
 },
 {
  "cell_type": "code",
  "execution_count": 11,
  "metadata": {},
  "outputs": [
   {
    "data": {
     "text/plain": [
      "<AxesSubplot:>"
     ]
    },
    "execution_count": 11,
    "metadata": {},
    "output_type": "execute_result"
   },
   {
    "data": {
     "image/png":
"iVBORw0KGgoAAAANSUhEUgAAAZoAAAExCAYAAABF3WROAAAAOXRFWHRTb2Z0d2FyZQBNYXRwbG9
0bGliIHZlcnNpb24zLjMuMiwgaHR0cHM6Ly9tYXRwbG90bGliLm9yZy8vihELAAAACXBIWXMAAAsTAAALEw
EAmpwYAABucklEQVR4nO3dd3xN5x/A8c+T2DIkkYnWrNYMQo0YESH23luMoigtihq1f9rqRlFFqVmtPUJstff
eNbIXiSRI7vP7497sm8h0h0b3neXvflnnO+55xvbm7u9zzPee45QkqJoiiKouQWE0MnoCiKorzZVEVRFRcpUq
NIqiKEquUoVGURRFyVWq0CiKoii5SShUaRVEUJVepQqMoivKWEEIsE0IECiEup7EFjYHEJIUz4n9qk
KjKIry9lgOeKazvDlQTCzaZLpR7GQZ5WYN+owyYIjZ3vQFjZ7U3QiZYTByfpNLQJMMapNu7Ld6XMvBMsV+T9
```

2ZInuxt4G70Mvmt01+0p6FTf0CmkycHMytAp6FXTrJShU9DrQ2Fp6BTS9K94YegU9IqQsYZOIU2r/t0ksrN+Zj
5v8tmWGYK2yyveYinl4kzsTl+u2f68U4VGURTFmGniMhyqKyqZKSwpPQJKJJkuDvhmY3uA6jpTFEUxblKT8U
f2bQH66Eaf1QaeSCn9srtR1aJRFEUxZpocKSAACCHWAI2AokKIR8BUIC+AlHIRsANoAdwGooD+ObFfVWgUR
VGMmMyZlopuW7L7K5ZLYHiO7VBHFRpFURRjFme8Ax0yShUaRVEUY5aJwQDGShUaRVEUY5aDXWeGogq
NoiiKMcvBwQCGogqNoiiKEcvJwQCGogqNoiiKMVMtGiWrvpg9n0NHT2JtVYS/Vy167fv/dv50mns2Jio6Gi+v
0Zw7n/qq4StX/EiNGlV5+fIlp06dZ+iw8cTGxtK6dVO+nDYWjUYSGxvLp59O5eixUzmS1/Q5E2jsUZ/o6BhGD5
/E5YvXUsX0G9idgR/1pmTpd6hc1pWw0PBky6tWq8SWPasZ5vUZ27d4ZymPag2r4zVtECamJuxd682mBRtT
xXh9OZgabjV4Hv2cHz/9nruX72DjWJRR347GytYKjZR4/7GLbcu2AvDpz+MoVlp72ajCFoV59vQZY5qPylJ+AK
UaVqHJ1N6YmJpwYe0Bji/cmmy5dRlHWn49GPuKJTn09QZOLt6RsMylfzOqdm8EQnBhzX5OL9ud5TwAKjSs
Spcp/RGmJhxdt489CzeniukytT8V3arxIvo5Kz9bwMMr9wBw698c127uIARH1+7DZ5k2T6+fPsG+tBMAhSw
KEfU0itktxmU6tyoNq9F76gBMTE04sHYvWxf+lSqm9zQvnN2q8zz6OYs/+4n7l+8C8O2RRcQ8i0YTpyEuLo4p
rbX7f+eDkvSfPYQChQoQ9CiQhaO+IzoyOtO5ZUjcy9zZ7mv0VhcaIYQE5kspP9VNfwaYSSmn5fa+27XwoEfH
Nkyc8XVu7yqV5p6NKVe2FO9XcOXDWtX5+ac51HVtnSpuzZq/6NN3BACrfv8ZrwE9+GXxSnx8jrB16x4AKlf+
gDV/LKJS5YbZzqtxk/qUKvMOri4tqO5ShTnfTKa1R49UcadOnGPv7oNs2PpbqmUmJiZMnDqagz5Hs5yHiYkJ
g2d+xLSekwnxC2He1vmc9D7Bo1uJ1xqs7lYDp5JODGswhPeqlWfIrKGMb/sZmrg4ls9cxt3LdyhQuCDfbP+W
84fP8+jWQ74ZPi9h/X5fDCAqIirLOQoTQdMZfVnbcy4R/qH02zKHW3vPEHIr8WohMeHP8J76O+81q5Fs3aL
vFadq90asaDOVuJexdF05jjs+5wm7H5DlXLpN9+KHXjMJ8w/h8y1zuOh9Gv/bjxNiKjaqhl0pB6Y2GkmpauXo
Pmsg89pNwum9Erh2c2du24nEvYxlxIqJXPI5S9B9f379+LuE9TtO6k10Fl4vYWJC3xmDmNvzS0L9Q5i+ZR5n9
p7C99ajhJiqbtVxKOXIpw2HU6bae/SbOZhp7T5PWD6r2xQiwyKSbXfg/4bxx6zlXD9xlQZdGtNySDs2frMm0/l
lyBvQdfa2X4LmOdBBCFH0de/Yxbkylhbmr3u3ALRu3YzfV2uP0E+cPItlEUscHOxSxe3c5PW/NSp8xQvrr1a+L
NniX/whQsVQvsdr+xr2sKNjWu3AHD29EUsLMyxs0/9q7ly6TqPHuq//FL/wT3YsdWb4KD0roSevnLO5fC770
fAgwBiX8ZyZOshajetkyymdtM6HPzTB4Cb527wVvhbWNlZEVhgGHcv3wEg5lk0j24/xbMbBJtU+6rVy5fDmg1
nO0dG5DGH3A3jyMAjNyziubj1OOY/kBSUq5Cn+F++ieZl8eKxNWSd8z90hNuYFMk7DgxPXea+ZS5ZzKelclq
B//Ql+GEjcyzhObz1G1aY1k8VUberC8U2HALh37haFqCJ03DzxDWcm9VKtY/qLetw
akvmDx7KOJcl4L4fQQ8DiHsZy/GtR6jhkXz7NTxqceTPAwDcOXeTwhaFKWKX/oVgHUs7cf3EVQAuH75Azea
1M51bhmk0GX8Yqbe90MSivQDdaEMn8joVc3JI9kH9+JEfxZwc0ozPkycPPXt2ZPfu/Qnz2rb15PKlg2zZvIJBgz
7NkbwcHO3xfeyfMO3nG4CjU9oy8WHaolmz5XtsaVrSTbthAa1+ACoWFXhb7R2v/IHY+PbvFOpIJeu3b3EE+lkaThK
tKFWxDDfP3Ug2v0KtioQHh+N3P+uXkDJ3sCLCL7GYRviFYu6QsatkB998RIla5SlQxIw8BfJRxq0qFk6pi2FGFb
G3Jsw3JGE6zC+EIvbWemISX68w/xCKOFjje+MhZWt9QOEiZuQtkI9KbtWwckyeS9laHxAR/ISg+/5klpWDDaF
+ibmF+oVg5WCdIsaakCS5hfqHYKXLXyL5fNVUZmz7CrfuHgkxD28+oLqHtph+2LIu1o65eKz6eq91live6q4zn
Z+Bi0KIeekFCSEGo7v89oJvZjKwT7pXcjBqQqS+Enh6rZKffpzN4cMnOHL0ZMK8zZt3sXnzLuq7fsiX08bSrHm3
155XStNmj2f2l9+iyeaRXVbzSBpToFABxv8ygWVfLknVd1+/bQMObz6UrRz1Xs09gy9VyG1fji/aRrfVn/PyWQ
yBVx+gic36lwIz9HrpiUFK/O88Zs+izYxc9QXPn8Xw6Nq/aOKS//5qtqmXpdYM6L/mfcrXKb38p3eYSHhgGBY2
loxfNRXfO4+5cfIqS8b+TJ9pXrQf1YWz3qeIfZmL39434pZKRr31hUZK+VQIsRIYCaR5Ni/p5beN8X40rzL0o754
efUE4PTp8xQv4ZSwrFhxR3z99PfPT/5iNLa2NgwdNlDv8sNHTlC69LvY2FgREhKW6bz6enWjB5ek8Ag7L98a+TE
nYoktK0cnewL8AzO8rSrOFfl56VcAWFtb0dijPrGxceze5e4fOKNZML8QumqFPiEaqNow2hgcm74kL8Q7BJchRr
42BDWIA2xjSPKeN+mcChvw5wfNc/ydYzMTWhtmcdPmuZvUZ0hH8o5o6JR+bmjtZEBGT89b+47iAX12m77
hqM7UKEf9a7GsP8Q7BK0iKycrThSWDyXML9Q7ByKgpoW3dWDjaE6/I9tn4/x9ZrW8ttx3YnLEkLxMTUBOd
mtZjT+nOyItQ/BOskLSRrx8TfU0KMXwg2SX7f1g42hOvyj///acgTzuw+QRnnctw4eRW/O4/5X+/pADiUcsS5c
fJuy5wkNf/9wQBve9dZvO8AL6CwgfPINQsXrcClZlNcajZly5bd9O6p/XD/sFZ1nj55ir+/fFtj1/eI5yDwd4a/bvT1KMRPXsN
T3aEWqZMyYTn1ZwrkS9f3iwVGYAVv66lWcNONGvYiV3bfejUrQEPE0ksCA4FdsIVHdap7UcW5GHe
dmbN+yh0ljZ2a6yADcunALx1JO2JWwJ0/ePLi2bsAp75PJYk55n8CtY2MA3qtWnqiIKMJ0H0rDvxrJo9sP2bI0
9cirqq7OPL7zmBD/kFTLMsPvwl2sSzlgWcIWk7ymVGhdm9veZzO8fiEbCwAsnGwo7+nC1c3HspzLvxfuYFfSE
ZvitpjmNcWldV0uep9OFnPR+zS1OzQAoFS1ckRHRPE0KBwAc10uVk42OHvW4nSS1sv7rpXxv+tLeBYL4d0Lt

3Eo5YhtCTtM8+ahdmtXznonHyF5du8pXDs2AqBMtfeIiogiPDCM/AXzU6BwAQDyF8xPpQZVeXTjAQAWNtq
b0wkhaDuiM/tWZ2/UXrregHM0b32LBkBKGSqEWI+22Cx7HfscO3Uup85dJDz8Ke7tejHMqzcdWzd7Hbtmx
859eHo25sa1o0RFRzNw4JiEZVs3r2TwR2Px8wtgwc9z+fffRxw5rD1B//ffO5g56zs6tG9Br16dePkylpjoGHr0H
Jojefl4H6KxR32OnNlJTHQ0Yz6enLBs5boFjB01lQD/IAYM7snQkf2xtSuK9+FN7N97mLGjpuZIDgCaOA1LJi9i6
u9fYmJqwr51e3l48wHNenkCsHvVLs74nKaGmwsLDy/WDm/+7HsAPqhZAbeOjbl/7R7zd2rnrZq3krP7zwDg
2qYBh7dkfRBAPBmnYc+UFXRdOQ5hasLF9QcJvvUY557a4nd+tQ+FbS3pu3UG+c0KIjUaXAZ4srTJeF5ERtN+
0SgKWpmheRnLnikreP406yPgNHEa1k5ZxoiVkzAxNeHY+v343XpE/Z7acxqHV3tzef85KrlVZ/rBH3gR/YKVYx
ckrD944acUtjInLjaWtZN/Jerps4RlLq3rJSs8WcltxZSljFs5BRNTEw6u38fjWw9p3LMpAD6r93De5wxV3arzzaE
FvNANbwawKFqETxaPB8A0jwnHNh/m4sFzANRp40qTPs0BOL3rOIfWZ/6AJsOM+NxLRomcGjH0XySEiJRSm
ume2wP3gHmvGt5sjF1n6lbOmadu5Zx56lbOmZfdWznHnPozw583BWp2zNa+cstb3aKJLzK65wFAIQOmoyi
Kktob0KJ5qwuNoiiK0TPicy8ZpQqNoiiKMVM3PlMURVFylWrRKIqiKLlJSnWHTUVRFCU3qRaNoiiKkqvUqDN
FURQlV6kWjaIoipKr1KgzRVEUJVeprrO3k7Fe7iXa97ChU9Dr5cbvDZ2CXhUnHTB0CnpdFKaGTiFNL430SsJ2+
YsYOoXco7rOFGNhrEVGUZRsegMKjbpNgKIoijHL4TtsCiE8hRA3hBC3hRCpbvQjhLAUQmwVQlwQQlwRQvT
P7o+gWjSKoijGLAcHAwghTNHeVdgDeAScEkJsVJeTRI2HLgqpWwthLAFbgghVksps3zpblVoFEVRjFnOdp3V
Am5LKe8CCCHWAm2BpIVGAuZCe49rMyAUyFa1U11niqIoxiwTXWdCiMNJHoNTbK0Y8DDJ9CPdvKR+
Aj4AfIFLwCgpszf0TbVoFEVRjFkmWjRSysXA4nRC9N0YLeWN1ZoB54HGQBnAWwhxWEr5NMOJpKBaNIqiK
MZMo8n449UeASWSTBdH23JJqj+wSWrdRnvn4fez8yOoQqMoimLMpMz449VOAeWEEKWEEPmAbsCWF
DEPAHdIuMV9eeBudn4E1XWmKIpizGJzbtSZlDCWCPC0SbY8IIT7SLV8EzEzACWCyEuoe1qGy+lDM7Of
lWhURRFMWY5fAkaKeUOYEeKeYuSPAY9ow31Zwr6Y1ZueJH
rlw+xPlz+1iy+Bvy5NHW/tatm3L2jDenT+3h+D87qFe35mvJ+YvZ82nQshvten30WvaX1NH7QbRbfog2yw6x
7GTqLuEVp+/RddVRuq46SqeVR6jx3S6exCR+hyxOI+m26igj/z6T47lNnTOe/ae2svPQBipW0X9etM/Abuw/t
ZV7IRewsi6SMN+jeSN2HtrA9gPr2LzvD1w+rJZjeU2ZPRafk5vZfnBdmnn19uqKz8nN3Ak+myyvNp2as/as/3gOrY
fXMeGHb/xfsVyOZYXwJdzPufQ6e3e3sPvwnlap8oDem78DuHDq9nQehl5LlFq9KtYrcCzpPizYe2crlsxmj+OvYGt
bsW075yu/pjXEq4cjy7b+w6egfzF40jTx5E4/Da9RxZrX3MtYdWMkvm35Mtp6JiQmr9/zKtyv/l60c05Sz52gM
QhUaQAgRJ4Q4L4L4S4LITYIIQolN1tNvdsTLmypXi/gitDh47n55/m6I1bs+YvKlZqgHM1dwoWLIDXgB4A+Pgco
XoND1xqNmXQ4E/55Zevs5tShrRr4cGi+TNfy76SitNI5vpc5ad2LvzZ15VdN/y4ExKZLKavSynW9arHul71GFH
vPWoUt8ayQL6E5X+cu08pa7Mcz61RE1dKln4Ht5qtmTBmOmTBmOjO//kJv3OkT5+nVYQiPHjxONv/ooRM0b9CZ
lo
26Mn7EVOZ+PzWH8qpHydLv0LhWWyaNmcn0rybojTtz8jy9O37Eowfjfz5+vcx3dsMpGXDrvz0zRJmzdf/c
2WFW5P6lCzzLg1cWvL56C+Z9U1ar9k5erQfxMMUrxloP8AnTB3NQZ9j2cqlXuPa2ZNr7DwmzP1U
b9yILz7ij8Xr6VCvBxFPImjbjvRUAZhZmjJ/7KWP6fU7XRn34fNDkZOt1H9SZe7f+zVaO6crZUWcGoQqNVrSU0
llKWQl4AWT7cL5162b8vnojACdnubfu8ylhu8kl4furUeYoXdwTg2bOohPmCxVCvqajFRfnylha
mL+WfSV12T+cEkUKUbxIIfKamtCsvAMH7gSkGb/rhh+e5R0TpgMiYjhyL4j2lYrneG4ezd3ezd3YtG4rAOdPX8LC0h
xb+6Kp4q5eus7jhylHikLUs+iE5wULFcyx32WT5o34a/02bV5V5n0svrBo8f+qWaf/bURZ4+iQDg3OlLODjZ50he
AE1buPHn2i26bV/EwsIcOz25Xbl0nUd6XjOA/oN7sHPrXkKCQrOVS0NPV3Zs2AXA5bNXMbcww8bOJlVcTdf
q7Nt2AIBt63fRqLn2Ku2e7Zuwf8dBAh4HAhAWEp6wjp2jLfXc6/D3H9uulWO6VKF5Ix0GymZ3I8WcHJL9AT1
+5EcxJ4c04/PkyUPPnh3ZvXt/wry2bT25fOkgWzavYNAg/Udhb4rAyOfYmxdImLY3K0DQ5t4AQ/Udhb4rAyOfYmxdMmLY3K0BQ5HO9sdEv4zh2Pxj
3cokfjF8duMao+uUx0fd1tGyyd7TD73Fi0fPzDcDBMfVBQ3qatmzM3uN/s2ztT4wbkTMtGntHO3yT5OXvG4i
Do22WttWlVzsO7juaI3kBODja4ffYP2HaP5Ovmb2jHc1aurPqt/XZzsXWwRZ/38CE6QC/IOwckxcc9S2tLIp5EE
hcXB0CgXxB2DtqYd8qUwNzSnF/+/IHfdy+lZedmCet9On0kP8xcgMzFD3kZF5fhh7FShSYJIUQeoDnayy6kXJZ
waQeN5llGtpVqXnpHsj/9OJvDh09w5OjJhHmbN++mUxfvUWGdOzkxZfTxmbsh3iTpFE0Dt0NxNmpSEK32aG7g
VgXykcFe8vcSUNPHpltlezZ7kOT2u0Y0vsTxkwcnot5ZX47tV1d6NyzHfO+/CH7P+OrTZo9nzpffosmB
D/CM/C2mF5PH1JQPqpRnVK9xfNz9U7w+6cs7pUvg2qQuocFhXL94M9s5pusNaNGoUWdaBYUQ53XPx7xXp
O/pgxIemmHPPmK6f2TGfpRX7y8egJw+vR5ipdwSlhWrLgjvn76u4ImfzEaW1sbhg4bqHf54MnK36XWxsr
AgJCcvoz/SfYmeWn4CIxC6mgMgYbAvn1xu7+4Yfnu8ndpud9w3j4N1AjtwP4kWshmcvYpm08wKzmlfNcj69

vbrSrXcHAC6eu4JjscTWk6OTPQH+QVna7sl/zvJuyRJYWRchLDQ80+v3GtCFrr3bA3Dp/BWcitkTP/TBwcku0
3mVr1CO2d9OZkC3EYSHPcl0Pkn18epG9z4dAbh47jKOxRJb8A5O9gT4B6a1aiqVnSvw09J5AFhbW+Hm4Up
sbBx7dvi8Yk2tzv3a065nawCuXriOg5MdF3TL7B1tCfIPSRYfHhKOuaUZpqamxMXFYedoS1CANibAL4jw0CfE
RMcQEx3DueMXKFehDO9XKU+DpvWo516bfPnzYWZemOk/TWbKxzMy/HNmyBtwh03VotGKP0fjLKUckd
XLYS9ctAKXmk1xqdmULVt207tnJwA+rFWdp0+e4q/nD21A/+409WhEz17Dkx1llSlTMuF5NedK5MuX940t
MgAVHSx5EBbF4ydRvIzTsPuGP41Kp+5qiXj+kjOPwmhUJnHZSNfy7B7kxg6vRsxtUZWaJWyyVWQAfv91HS0
bdaVlo67s2bGfDl21H1rOLpWJeBpJUEDGv7/2bqnEK35UrPI+efPlzVKRAVi1bD2t3brT2q07e3YcoH0X7Qlr5x
qZz8uxmAMLl3/NZ8Mmc//Ogyzlk9TKX9fSvGFnmjfszO7tPnTs1gaAai5ViHgaSWAmcnOt1px6zp7Uc/ZkxxZv
vhg7K8NFBmDD8r/o6TGAnh4DOLDzMC06ewJQqXoFIiMiCQkMSbXO6aPncG/VCIBWXTw5uEt7M8GDu4/
g/GFVTE1NyV8wP5WqV+D+rX/5efYvtKzRkTa1ujDpo2mcOnI254sMgEZm/GGkVKHJJTt27uPuvQfcuHaURY
vm8fGIiQnLtm5eiaOj9gh5wc9zsbMrypHDWzh9ag9fTPoEgA7tW3DhvA+nT+3hxx9m06Pn0NeS99ipc+k5Z
DT3HzzCvV0v/ty6+7XsN4+JCeMbV2DYptN0WHGYpu85UKaoORsuPGDDhcQPwf23A6j9rg0F876+xvh+78
M8/PcRB05vY863U5k8dlbCsmVrf8LOQXtepN/gHhy7tAcHJ3t2Ht7A3O+052I8Wzdh99FNbD+wjunzJjLCa1y
O5HXA+wgP/n2Mz6nNzP72C6aOSxzZ+OuaHxLOMfQd1I0jF3fi4GTH9kPrmP2ddtTUiLGDKGJtyZfzJrB1/xr+
3rsqR/IC8PE+zIP7jzh8Zgf/+24aX4xNHMm4fN0C7HWvWf/BPThxeS+OTvbsOfwn//t+Wo7lEO/ovn94/K8vf/
+zli++HsfcCfMTln2/ah5F7bUDA36cuZCeQ7rw17E1WFpZsnnNdgDu3/qXf/afYI3PclbuWMzff2zjzo17OZ5n
mt6ArjPxukYzGTMhRKSUMsPjYtPqOjMkY76V88uN3xs6Bb0qTjpg6BT0MhGmhk4hTS81Lw2dgl52+YsYOo
U0nfY7nK0hKlHfDcnw502hT37JheEw2afO0QCZKTKKoiivlRG3VDJKFRpFURRjZsTnXjJKFRpFURRj9gaMOlOF
RlEUxZipFo2iKIqSm3LzqgOviyo0iqIoxsyILy2TUarQKIqiGDPVdaYoiqLkKtV1piiKouQq1aJRFEVRcpUa3vx2cjC
zMnQKqZR6rw03Z7sbOg298nYaZegU9Ioa/3qu45ZZo4u4GDqFNH0ddvLVQQaQp4DxXrYn21SLRjEWxlpkFE
XJHhmrRp0piqIouUm1aBRFUZRcpc7RKIqiKLlKtWgURVGU3CTfgEKj7rCpKIpizGLjMv7IACGEpxDiohDihRBi
zRiGgkhzgshrgghDmb3R1AtGkVRFGOWgy0aIYQp0ByoCKIoh9CiMFCiNNJHoNTbK4Y8DDJ9CPdvK
TeA6yEEAeEEGeEEH2y+zOorjNFURRjlomuMynlYmBxOiFC32oppvMANQB3oCDwjxDiuJTyZoYT0bNBRVEU
xVjl7KizR0CJJNPFAV89McFSymfAMyHEIaAqkOVCo7rOctH0ORM4cnoH3oc3UanKB3pj+g3szpHTO3gUehkr
6yKplletVol/gy7Qso1HjuV19H4Q7ZYfos2yQyw7eTfV8hWn79F11VG6rjpKp5VHqPHdLp7EvEhYHqeRdFt1lJ
F/n8mxnF7li9nzadCyG+16ffTa9pnUrP9N4vi53ew/upnKVSvojRkwqCfHz+0m4Ml1rFP8Luu61mLf4b84eHw
rf23/PUdyKtWwCoN8vmLIwW+oPbR1quXWZRzp/ddUPrv5G7UGt0i2zKV/M7z2zMHLey4uA5rlSD4pzZ73B
SfPe3Pw2BaqpPGaeQ3uxcnz3gQ/vYm1deI1BD8e6cX+I5vZf2Qzh49vIyDsGkWsLLOcy5gZI9hwdDWr9v5K+
crl9MY4lnDg120L2HBkFTMXTSFPXu1xePU6zuy9vo2V3ktZ6b2UAaO1PUn58ufj1+0L+d17KX/s/42Bn/XLcn
7pkbGaDD8y4BRQTghRSgiRD+gGbEkRsxmoL4TII4QoBHwIXMvOz/DWt2iEEJFJFSX/8Q20kwtyAbA94B7/ImwrGrcpD
6lyryDq0sLqrtUYc43k2nt0SNV3KkT59i7+yAbtv6WapmJiQkTp47moM/R7KSSTJxGMtfnKgs71MTevAA9//i
HhmXsKGNjlhDT16UUfV1KAXDwTiCrz93HskC+hOV/nLtPKWsznr2IzbG8XqVdCw96dGzDxBlfv7Z9xnP3aEC
pMu9Su1ozarhUZd78qTR375oq7uSJs3jvPsCmbSuTzbewNGfuN1Po3nEQjx/5UbSodbZzEiaCpjP6srbnXCL8
Q+m3ZTq39p4h5FbiwWlM+DO8p/7Oe81qJFu36HvFqdq9ESvaTCXuZSxdV47jjs95wu4HZDuveE2aNqR0mZ
LUcvagRs2qfPXtlzRr3DlV3MnjZ9izaz+bUxTfn374lZ9++BWAZp5ufDS8H+FhT7KUS53GH1KiVHE61+tJxeoVG
DdnNF6thqWKGz5pCGuWbGTvZh/GzR1Dm+4t2LRS+xl8/sQlPus7IVn8i+cv+LjzGKKjojHNY8riv3/kH5+TXDl7
NdW2syUHLwwgpYwVQnwM7AZMgWVSyitCiGVW5MWa2X17G1T1Q7TiAkWrsWp8/xkGbNhJnmNq3GYRQPz2AqaLe6lvH
NdW2syUHLwwgpYwVQnwM7AZMgWVSyitCiI90yxdJKa8JIXYBF3V7XyqlvJyd/aoWjY4Qwh34ed2Qvuxur2
kLNzau1b5Jz56+iIWFOXb2RVPFXbl0nUcPU7ZctfoP7sGOrd4EB4VmN50El/3DKVGkEMWLFCKvqQnNyjtw4
E7aHzC7bvjhWd4xYTogIoYj94JoX6l4juWUES7OlbG0MH+t+4zn2dKdDWs2A3Dm9AUsLC2ws7dNFXf54jUe
Pnican6Hzq3YsdWbx4/8AAgOzv7v09G5DGH3A3jyMAjNyziubj1OOY/kBSUq5Cn+F++ieZn8+xU2ZZ3wPXeH
2JgXyDgND05c571mOXvF6OYt3Fm/5i8Azpy6gKWlOfZ6XrNLabxmSXXo3IpNG7dnOZcGzeqxY6P2St1Xzl7Fz
NIMG7vUxd7FtTr7t2m/MrJjwy4aeLq+ctvRUdEA5MmbR9sCkjn/5UqpkRl+ZGh7Uu6QUr4npSwjpZylm7dIS
rkoScxXUsoKUspKUsrvsvszqEIDCCHHqA0uAllLKOzmxTQdHe3wf+ydM+/kG4OBon4n17Wje0p3ff1ufE+kkCIx

8jr15wYRpe7MCBEU+1xsb/TKOY/eDcS+XmPdXB64xqn55TPSdUnxDOTra8/ixX8K0n68/jk4Z/12WKVMSyyI
WbNq2kj0H/6Rzt5SjSTPP3MGKCL/EghXhF4q5Q8ZuXxF88xElapWnQBEz8hTIRxm3qlg42WQ7p6Qcnex5/Cj
x/e/7OCBTr1m8ggUL0LhJfbZuyfotHWwdbAn0DUqYDvQNwtYhedGztLYk4kkkcXHaohzolzymco0K/O69lG9
X/Y9S75VMmG9iYsJK76XsvPg3Jw+d5sq5bPUw6Zezw5sN4q3vOgPyo+2TbCSlvJ5TGxUi9SexzMTRzrTZ45n9
5bdoXsdtXNMoGofuBuLsVCSh2+zQ3UCsC+Wjgr0lpx+G5H5exkLP65OZ36VpnjxUda5Ipzb9KVAgP9v3ruXM
qQvcvXM/h5PK2Joht305vmgb3VZ/zstnMQRefYAmhy9Fn933f7xmzRtz8vjZLHebaXNJPS9lLnqHYulirl+6Sbt
a3YiOiqZO4w+Zt2wmnV17AaDRaOjjMRAzCzP+9+sMSpcvxd0b97Kcq17//WtqqkIDvASOAV5Amnfo0o1HH
wxQpJAjhfOnbnr39epGjz6dALhw7jJOxRwSljk62RPgn/HvPVVxrsjPS78CwNraisYe9YmNjWP3Dp8Mb0MfO
7P8BEREJ0wHRMZgWzi/3tjdN/zwfD+x2+y8bxgH7wZy5H4QL2I1PHsRy6SdF5jVvGq2cjJG/Qf2oFdf7TmF8+
cuUaxY4uvg6OSAv1/Gf5d+vv6EhoQRFRVNVFQ0x4+dpmLl8tkqNBH+oZg7Jr4HzR2tiQgIy/D6F9cd5OI6bTd
Rg7FdiPDPfnfegEE96d23CwDnz16iWPHE979TMftMvWbx2ndsyaaN2zK9Xsd+7WjbsxUA185fx84psXVi52R
LcEBwsvjw0CeYW5phampKXFwcdo6JMVGRUQlx//icIM+c0VhaW/IkNLH4RT6N5Ow/56ntVivHC4261tmb
QQN0AWoKISamFSSlXCyldJFSuugrMgArfl1Ls4adaNawE7u2+9CpWxsAqrtUIeJpJIEp3tzpqVvNkzrOzajj3Izt
W/YwaezMbBcZgIoOljwIi+LxkyhexmnYfcOfRqVTX2Ei4vlLzjwKo1GZxGUjXcuze5AbO7waMbdFVWqWsHkji
wzAb0v/wL1+e9zrt2fntn107q7t7qrhUpWIpxEEBgS9YguJdm3fR+26NTA1NaVgwQJUr1GFWzdSj/bLDL8Ld7
Eu5YBlCVtM8ppSoXVtbnufzfD6hWwsALBwsqG8pwtXNx/LVj4Ay5asxs21LW6ubdmxfS9durcHoEbNqjx9Gk
lAJl4zAHMLM+q61mTn9n2ZzuXP5X/Tx2MgfTwGcnDXEVp00o6sq1i9ApFPnxESmLqwnjl6DrdWDQFo0dmT
w7u1g3CsbRP/3is4v48wETwJfUIRa0vMLLSDaPIXyEfN+jX493a2T++mImNlhh/GSrVoAClllBCiFXBYCBEgpfw
1u9v08T5EY4/6HDmzk5joaMZ8PDlh2cp1Cxg7aioB/kEMGNyToSP7Y2tXFO/Dm9i/9zBjR03N7u7TlMfEhPG
NKzBs02k0UtK2YnHKFDVnwwXtH0jnqu8AsP92ALXftaFgXuN4i4ydOpdT5y4SHv4U93a9GObVm46tc2dYbk
p79xzEvWkDTpzfQ3RUDKOGJx6PrN7wC2NGTCbAP5CBQ3ozfJQXdvZF2X9sC/u8DzJmxGRu3byLz97D7D+2
GanRsHrlRq5fu5WtnGSchj1TVtB15TiEqQkX1x8k+NZjnHs2BuD8ah8K21rSd+sM8psVRGo0uAzwZGmT8byIj
Kb9olEUtDJD8zKWPVNW8Pxp1Cv2mDneuw/QpGlDTl3YS3RUNCOHJY7YWrNxCaM/noS/fyCDPurNiFGDsL
MvyqF/trB3zyE+GTEJgJatPDjgc5SoqOi0dpMhx/Ydp677h2w8tpqY6OfMHP2/hGXzf5/L7M++IjgghJ9n/cKM
hVMYMs6Lm5dvsWXNDgAat2pIhz5tiIuN43nMCyYPnQ5AUXsbJn8/AVMTE4SJCfu27ufo3n+ylateb0DXmch
Kv+mbJMXw5hLAIeATKeXmtNYpbl3J6F40Y76Vc95OafZIGlTxMi1eHWQAo4vk7AiwnPR12ElDp6BXWTMn
Q6eQpuO+B7I1dCakdcMMf97YbD1olMN0jONw1YDii4zu+UOglAHTURRFSe4NaNG89YVGURTFmL0Bd3J
WhUZRFMWYydd3AY5cowqNoiiKEVMtGkVRFCVXqUKjKIqi5C5plAPJMkUVGkVRFCOmWjSKoihKrpIa1aJRF
EVRcpEmThUaRVEUJReprjNFURQlV6mus7dUTTPju0pNxUkHDJ1CmqLGZ/2mVbnp0Z0dhk5Brx+qTzF0Cm
mqalHS0Cno9VLm7P10jMmbcDlKVWgURVGMmGrRKIqiKLlKDQZQFEVRcpVq0SiKoii5SqorAyiKoii5SQ1vVh
RFUXKVRrVoFEVRlNykus4URVGUXPUmjDozMXQCiqIoStqkRmT4kRFCCE8hxA0hxG0hxOfpxNUUQsJITpl9
2dQLRpFURQjlpPnaIQQpsDPgAfwCDglhNgipbyqJ+5/QI5c1kMVmhxUrWF1vKYNwsTUhL1rvdm0YGOqGK8
vB1PDrQbPo5/z46ffc/fyHWwcizLq29FY2VqhkRLvP3axbdlWAD79eRzFShcDoLBFYZ49fcaY5qOynevUOeNp
1MSVmOgYPvt4MlcuXk8V02dgN/oP6UnJ0u9QvVxDwkLDAfBo3ogxE4aj0WiIjYtjxsSvOH3iXLZzApj1v0m4N
21AdFQMI4dN4NKFq6liBgzqyeBhfShV+l0+KFWbUF1eAHVdazFjzgTy5M1DaEg47Vv2zpG80vPF7PkcOnoSa
6si/L1qUa7vL17JhlVwm9YbYWrC5bUHOLlga7Ll77erS62hrQB4+SyGvZOWE3TtAQDNvhpEaXdnokKessJjQo
7kM3z6MD5sXJPn0c+ZN/prbl2+nSrGoYQDXyyYiHkRc25dusXcUfOIfRlLiTIlGDf/U8pWKsuyecvZ8Evi307HgR
1o0d0TKeHe9XvM+/RrXj5/meU8R04fTu3GH/I8+jlzRs/j5uVbqWIcSzgwdcEXWFiZc/PSLWaOnEvsy1hcm9b
Fa2x/NFJDXGwcP05dwKVTl7OcS0bk8DmaWsBtKeVdACHEWqAtkPIPbQTwJ1AzakaabteZEKKIEEIqGEYTuwoN
wkhPhFCFDJkDiYmJgye+REz+k5jpPtwXNs0oHi5EsliqrvVwKmkE8MaDGHh5z8zZNZQADRxcSyfuYwR7sMY3
/YzmvdpmbDuN8PnMab5KMY0H8U/O49xfNc/2c61URNXSpZ+B7earZkwZjozv/5Cb9zpE+fp1WEIjx48Tjb/6
KETNG/QmZaNujJ+xFTmfj812zkBuHs0oFSZd6ldrRmfjZrCvPn6t3vyxFk6tx3Ag3+T52Vhac7cb6bQp/swGtZu
zaC+2S/IGdGuhQeL5s98LfuKJ0wE7jP7sqnvPJa7j6N8m9pYl3NKFvP0YRDrusxkZbOJ/PPD33jMHZCw7PKGQ
/zZ56scy6dW45oUL1WMPq79mT/+O0bNGak3btBEL/5csom+9fsT+SSS5t08AYgIj+CnKQuSFRiAog42tB/Qjq

EtP2Zgk8GYmJrQuE2jLOdZu3EtipcqTg/XPnw1fj5j5uh/jwyZNIj1S/6kh2tfIp5E0rJ7cwDOHDlLf49BeDUdwtx
Pv2bc159mOZeMkjLjjwwoBjxMMv1INy+BEKIY0B7IsaOmV52jKQIYvNAIrfRy/QTIVKERQuRoa66cczn87vsR
8CCA2JexHNl6iFpNP0wWU6tpbfb/6QPAzXM3KGxRGCs7K8ICw7h7+Q4AMc+ieXT7ITYONqn2Ua+VK4c3H
8x2rh7N3di0Tnv0e/70JSwszbG1L5oq7uql6zx+6JtqftSz6ITnBQsVRObQVf88W7qzYc1mAM6cvoCFpQV29ra
p4i5fvMbDFMUPoEPnVuzY6s3jR34ABAeH5kher+LiXBlLC/PXsq94Ds5lCL8fwJMHQWhexnFj63HKNq2RLM
b3zC2eP4kCwO/cbcwcrROWPT55g5jwyBzLp17TuuzZ6A3AtbPXMbMojLWddaq4avWcObj9EAB7NnhTr1ld
AMJDwrlx4Saxsakvjmmax5T8BfJjYmpCgYL5CQ7I+u/VtVk9dm/cA8DVs9cwszTDRk+e1etV4+B27d/arg17q
N+sHgDRUTEJMQULFXgtV7zUSJHhhxBisBDidJLH4BSb09c8SvlDfAeMlzLnrlT6qg/buUAZIcR5wBsIBLoA+YG/
pJRThRAlgV3AEaA2cAH4DfgSsAN6SilPCiGmAWXQVs8SwDwp5RIAIcTYNLa7E9gP1AQgHfRoDfOPNt2K4kxQ3Knz
Z/zGvNAz7ZnA6ITE8zovaovkn45a2kY9hofWQ6vEFmsZB76AM56bwWvwnL4pKzz2BnqDLqMN+s7dFjHfaJYyQX
…

Stpi1SyuVCiBHAcMBPNxigE9ovGz0ELgNmSQYDbIuv0LpW0kK03XB5gbVSyumvetHav9Pa6H73559l7ijvdYq
Kff7qIANQt3LOvB2ajHdlvk7GfCvnQ4/3Zavv66hDpwx/3tTz32iU16t55TkaKWWPFLO+1xOWUAyklP2SPL+f
dBlwU0qZcrgdUsrvX7XdlNtOMf9H4Mck0xvRnsRKGdcvxfQ9wFPfNhVFUYzBG3CXAHVlAEVRFGMm9X715b
/ltRUaKeW017UvRVGUN4XG6DrqM0+1aBRFUYyYRrVoFEVRlNykus4URVGUXBWnCo2iKIqSm9SoM0VRFC
VXqUKjKIqi5Cp1jkZRFEXJVZm4S4DRUoVGURTFiKnhzW+pD4WloVNI5aIwNXQKaRpdxMXQKehlrNcUG3n2l
ZfbM5jd1Qx+w129yuYxvr/JnGK8V3HLOFVoFEVRjJhG343Y/mNUoVEURTFib8AVaFShURRFMWZqeLOiKIqS
q9SoM0VRFCVXqUvQKIqiKLlKtWgURVGUXKXO0SiKoii5So06UxRFUXKV6jpTFEVRcpXqOlPSVKphFZpM7Y2J
qQkX1h7g+MKtyZZbl3Gk5deDsa9YkkNfb+Dk4h0Jy1z6N6Nq90YgBBfW7Of0st05nt+U2WNp1MSV6OgYxo
2YypWL11PF9PbqSv8hPXi3dAlc3mtMWGg4AG06NWfIiH4ARD2LYvLY2Vy/civbORnra1ayYRXcpvVGmJpwe
e0BTi5Intf77epSa2grAF4+i2HvpOUEXXsAQLOvBlHa3ZmokKes8JiQYzllxBez53Po6EmsrYrw96pFubKPYV8O
pWbjmjyPfs7XY77h9uXbqWIcStgz8ecJmBcx59bl28wb9RWxL2PTXb/DwPZ4dvMEJPeu3+frT7/h5fOX9B7di
+Y9PHkS8gSAZf9bzqn9p9LNsWJDZ7pM6Y+JqQlH1u1j98K/U8V0ndqfSm7VeRH9nOWf/czDK/cAaNy/Ba7d
3BFCcGTtXvYt25FsPY9Brek0qQ9jqg3gWVhEZl++DIl7A1o0Jq9rR0KIkkKIy7mw3elCiCZ65jcSQmzTPW8jhPh
c97ydEKJCTueRbN8mgqYz+rK+7zyWNBlHhTa1sSnnlCwmJvwZ3lN/5+SS5G/cou8Vp2r3RqxoM5VlnhMp61
4Nq5L2OZpfoyb1KFn6HRrXasukMTOZ/pX+D8AzJ8/Tu+NHPHrgm2z+o38f073NQFo27MpP3yxh1vwvsp2T
sb5mwkTgPrMvm/rOY7n7OMq3qY11iryePgxiXZeZrGw2kX9++BuPuQMSll3ecIg/+3yVI7lkVrsWHiyaPzPXtl
/TrSbFSjnRv/4Avhv/PSNnf6w3zmuCF5uW/kX/Bl5Ehkfi2a1ZuuvbONjQrn9bPm41gsFNPsLExIRGbRolbG/T0
r8Y6jmcoZ7DX1lkhIkJ3ad78WO/WUzzGE3NNvVwLFs8WUylRtWwK+XI5EYjWDXxF3rOGgSA03slcO3mzpy
2E5jR/DMqN66BXUmHhPWsHG34oH4VQh4FZfq1ywxNJh7G6rUVmtwipZwipdz7ipgtUsq5usl2QK4WGkfn
MoTdD+DJwyA0L+O4uvU45TxqJIuJCnmK/8W7aF4mv2SeTVknfM/dITbmBTJOw4MT13mvWc5elLJJ80b8t
X4bAOfPXMLC0hxb+6Kp4q5eusHjh36p5p89dZGnT7RHb+dOX8LBKfsf6sb6mjk4lyH8fgBPHmjzurH1OGWb
Js/L98wtnj+JAsDv3G3MHK0Tlj0+eYOY8MgcySWzXJwrY2lhnmvbr9u0Dt5/7gPg+rnrFLYww9rOOlWcc72qH
Np+GADvjXup26zuK9c3zWNK/gL5MDE1IX/B/IQGhGQpx1LOZQn815/gh4HEvYzl9NajVG2a/L1RtWlNjm86
CMC9c7coaF4YC9siOJQtxr1zt3gZ8wJNnIabJ67i3KxWwnqdJ/dj05xVyFw+XZ/ThUYI4SmEuCGEuB1/AJ5ieU
8hxEXd45gQomp2f4bXXWhMhRBLhBBXhBB7hBAFhRAHhBAuAEKIokKI+7rn/YQQfwshtgoh7gkhPhZCjBFC
nBNCHBdCWOvilgshOumgeewohrgshjgAd4neq29ZPQoi6QBvgKyHEeSFEGSHE2SRx5YQo7Q5o7WBHhF
5owHeEXirmDVYbWDb75iBK1ylOgiBl5CuSjjFtVLJxssptSMvaOdvg+DkiY9vcNxMHRNkvb6tKrHQf3Hc12Tsb
6mpk5WBHhmzwvM/u086rctRH391/MkX0bOxsHG4J8E4/mg/2CsHFI/rpbWFkQ+fQZmjhNQkxRXUxa64f4
h7Dhl42sOv47a8/8QVTEM84cSvgzpU3fNizas5AxX4/GzNIs3RyL2FsT5ptYpML8Qilib5MqJjRJTLh/CFYO1vje
eEi5Wh9QuIgZeQvko7JbdawdtQdkVZq4EB4QyqNr/2botcoOmYnHqwghTIGfgeZoD7i76+nhuQc0lFJWAW
YAi7P7M7zuQlMO+FlKWREIBzq+Ir4S0AOoBcwCoqSU1YB/gD5JA4UQBYAugPuBAClLKY8AWYKyU0Ke
Qd4IoRw1oX0B5Zn5QdLTk+nagYPekJu+3J80Ta6rf6crivHEXj1AZrYnL1QuL6LwcosHJTVdnWhc892zPvyh+w
nZaSvmdD7YumPLVHnAyp1bcihOWtzZN/GTv9rI18ZI3Uxaa1vZmlG3aZ16FO3H91delKgUAHc2zcGYOvv2+j
n2p+hzYYRGhjK4MmDXpGknnkZyhH87zxm96LNfLJqMqNWTOLhtfvExcWRt0A+WnzcgS3z16W/7xyiERl/Z
EAt4LaU8q6U8gWwFmibNEKeUxKGaabPA4UJ5te92CAe1LK87rnZ4CSzGJgG/ELJKEr/4jfL6WMACMACKW
Lf123/FoAQYhUwOAM5LQX6CuGOA03slcO3mzpy
0TfkGY7sQ4R/6ijVerdeALnTt3R6AS+ev4FTMnvimm4OTHQH+metnLl+hHLO/ncyAbiMID3uS7fyM8TUDXcv
KKXlekYGp8yr6fgmazhvIpj5fGayr7HVo3bc1Lbp7AnDjwk1snRJbwkUdbQkJSP66Pwl9gplFYUxMTdDEaZLFB
PsF612/mms1/B8G8CRU+746svMoFVw+YN9fPoQHhyfE7/xjFzOWf5luvuH+oVglad1aOVoTHpg8xzD/EKyd
bLijmy7iYEO4Lsej6304ut4HgHZjuxPmF4Ltuw7YFLdj8k7tuTcrBxu+2DaPOe0m8DQonJyWw+deigEPk0w/Aj
5MJ94L2Jndnb7uFs3zJM/j0Ba62CR5FEgnXpNkWoP+IpmVztI/0TYjWwFnpJR6O4OllIullC5SSpf0igyA34W7
WJdywLKELSZ5TanQuja3vc+mu05ShWwsALBwsqG8pwtXNx/L8LppWbVsPa3dutParTt7dhygfRftKCnnGp
WJeBpJUEBwhrflWMyBhcu/5rNhk7l/50G2cwPjfM0A/C/cpUgpByx0eZVvXZs7KfIyd7KhzeJP2PnJIsLu+efIfo
3V1hVbE07EH9v9Dx4d3QF4v9r7PIt4Rmhg6gJ/4dhFGrSsD4BHpyb8s+cfAP7xPq53/aDHgbxf7X3yF8gPQLV

6zjy4pf1sTHoOqJ5nXe7fuJ9uvvcv3MaupCM2xe0wzZsHl9b1uOB9Onl+3qep3aEhAKWqlSM6IiqhYJjr3ldWT
kWp5vkhp7YcxffGA8a6DGSS63AmuQ4nzD+Ema3G5UqRAe0HZUYfQojBQojTSR4pD7b1tvH07VcI4Ya20IzP
7s9gDMOb7wM1gJNAp2xs5zpQSghRRtcl1j2NuAgg4QyplDJGCLEbWIj2Rc02Gadhz5QVdF05DmFqwsX1Bw
m+9Rjnntrm//nVPhS2taTv1hnkNyuI1GhwGeDJ0ibjeREZTftFoyhoZYbmZSx7pqzg+dOonEgrwQHvIzRq4orP
qc3ERMcwfuS0hGW/rvmBCaOnE+gfTN9B3Rg0oi+2djZsP7SOA3uPMPGTGYwYO4gi1pZ8OU87Wi0uLo52T
XplKydjfc1knAafySvo+Ps4TExNuLzuICE3H1Ollzavi6t8qDOqPQWtzHCf2Q8ATVwcq1tp797Z8sfhFK/zAQWtz
Bh84geOzf+Ty7qWV24bO3Uup85dJDz8Ke7tejHMqzcdWzfLse2f9DlJrcY1WX5kmXZ48qfzE5bNXDGd+eO+I
zQglKVzfmXizxPoO7Yvdy7fYdfa3emuf/38DQ7vOMyCnT8RFxfH7ct32PGH9qB64EQvylQsjZQQ8CiA7z9Pv9t
WE6dh7ZRfGbVyEiamJhxdvx+/W49o0NMDgEOrvbm8/yyV3aox8+CPvIh+wYqxPyesP2ThZxS2MicuNpY1k5
cS9fRZjr1+GZWZL2xKKReT/jmVR0CJJNPFAd+UQUKIKmh7e5qndfCdGUJmpXM+KzsSoiSwTUpZSTf9GWCGt
o9wPRAJ+AC9pJQlhRD9ABcp5ce6+Pu66eCky4QQy3Xb3SiE8AS+A4KBI0AlKWWrFPH10J7LeQ50klLeEULU
RtuyeUdK+crO/bnv9jK6q0IseXbV0CmkaVDhXB3kl2V5je63qGXMt3JuaaS3ci5lmnuj67Lrl/sbsvVNmMx83nz
+76p09yWEyAPcBNyBx8ApoIeU8kqSmHfQfhb30Z3XzrbX1qKRUt5He3I/fvrrJIuTnm/5Qrd8OUlOzEspSyZ5
nrBMStkvyfxdaM/VpNx30vijpB7e7Aosy0iRURRFeZ1y8nhIShkrhPgY2A2Yov3cuyKE+Ei3fBEwBbABFugGSsR
KKbP1fQFj6DozKCHEX0AZoLGhc1EURUlJk8Pf05FS7gB2pJi3KMnzgcDAnNznW19opJTtDZ2DoihKWt6Ebpa
3vtAoiqIYM2O+tExGqUKjKIpixNRtAhRFUZRcldPnaAxBFRpFURQj9t8vM6rQKIqiGDV1jkZRFEXJVXFvQJtGFR
pFURQjplo0iqIoSq5SgwHeUv+KF4ZOIZWXmpeGTiFNX4edNHQKelW1KGnoFPTabaTXEwPYfm6BoVPQq+I
HXQydQq7575cZVWgURVGMmuo6UxRFUXKVGgygKIqi5Cp1jkZRFEXJVf/9MqMKjaIoilFTLRpFURQlV6nBAI
qiKEqukqpFoyiKouQmNepMURRFyVWq60xRFEXJVRr532/RmBg6gTdJhYZVmbbvO7488ANNh7bVG9Nlan+
+PPADk3Z+RYmKpRLmu/VvzuTdXzN5zzc0HtAiYb7XT58wccc8Ju6Yx8wjPzFxx7wcyfXLOZ9z6PR2dh/+k0pV
PtAb03dgdw6d3s6D0EtYWRdJtbxKtYrcCzpPizYeOZITwOx5X3DyvDcHj22hStUKemO8Bvfi5Hlvgp/exNraKm
H+xyO92H9kM/uPbObw8W0EhF2jiJVllnMZPn0YK4/8xhLvRZSrVFZvjEMJB37a+gMrDv/GFwsmkiev9titRJkS
/Lj5O3be2UbnIZ2SrdNxYAd+3beYpXsXM+mnCeTNn/eVuQz7cii/HV7Goj0LKZtmLvb8sOU7fjv0KxMXTEjIJb
31Owxsz+K9v7B47yIm/PR5Qi69R/fij1OrWLjrZxbu+pmabjVfmWNGfTF7Pg1adqNdr49ybJuZ2/9neJ/8iy0H
1lChSnm9Mb28uuB98i9uBp3GyjrxPVS67Lus27GMy4+OMWBYr9eSr8zEw1i99YVGCHFfCFE029sxEXSb7sV
P/WYz3WM0NdvUw6FssWQxFRtVw66UA1MbjeSPiYvpPmsgAE7vlcC1mztz205kVvOxVG5cHduSDgD8+vF
3zG4xjtktxnFu5wnO7zqR3VRxa1KfkmXepYFLSz4f/SWzvvlCb9zpE+fo0X4QDx88TrXMxMSECVNHc9DnWLb
zidekaUNKlylJLWcPxoyazFfffqk37uTxM3Rs048H/z5KNv+nH37FzbUtbq5tmTntG44dOUl42JMs5VKrcU2Kly
pGH9f+zB//HaPmjNQbN2iiF38u2UTf+v2JfBJJ826eAESER/DTlAVs+GVjsviiDja0H9COoS0/ZmCTwZiYmtC4T
aN0c6npVpNipZzoX38A343/npGzP9Yb5zXBi01L/6J/Ay8iwyPx7NYs3fVtHGxo178tH7caweAmH2FiYkKjJLls
WvoXQz2HM9RzOKf2n8rIy5Yh7Vp4sGj+zBzbXmY0bFKPkqVL4FGrPZM/ncWX8ybojTtz8gL9Og7j0QPfZPPD
w58yc+LX/Lpg1etIF9AOb87ow1i99YUmp5R0LkvQv/4EPwwk7mUcp7ceo2rT5EeBVZu6cHzTIQDunbtFIfPC
WNgWwaFsMe6du8XLmBdo4jTcPHEN52a1Uu2jess6nNpyNNu5Nm3hxp9rtwBw7vRFLCzMsbNPXWuvXLr
Oo4e+qeYD9B/cg51b9xISFJrtfOI1b+HO+jV/AXDm1AUsLc2xt7dNFXfp4jW9xS+pDp1bsWnj9iznUq9pXfZs9
Abg2tnrmFkUxtrOOlVctXrOHNyu/Z3u2eBNvWZ1AQgPCefGhZvExsalWsc0jyn5C+THxNSEAgXzExyQ/mtYt2
kdvP/cB8D1c9cpbGGmNxfnelU5tP0wAN4b91JXl0t662tzyYeJqQn5C+YnNCDk1S9ONrk4V8bSwjzX96OPu2
dD/lq3A4ALZy5jbmmOrb1Nqrhrl27w+KFfqvmhwWFcOn+V2JexuZ5rPJmJf8bqrSk0QoiSQojrQogVQoiLQoi
NQohCusUjhBBnhRCXhBDvZ2X7ReytCfNN3CMN/CMN8wuhiL21npjgxBj/EIo4WON74yFla31A4SJm5C2Qj0pu1b
ByTP7mL1vrAyKCnxB03z8r6SXj4GiH3+PE7fj7BuDgaJfh9e0d7WjW0p1Vv63Pdi5JOTrZ8/hRYl6+jwNwdLLP9
HYKFixA4yb12bpld5ZzKepgQ5BvUMJ0kF8wRR2S/04srCyIfBqJJk6TJCb9xnGwfwgbftnAmhOr2HB2LZERUZ
w5dCbddWxS5BLsF4SN3lyeJeQS7BeUkG9a64f4h7Dhl42sOv47a8/8QVTEM84cOpsQ16ZvGxbtWciYr0djZ
mmWbo7/FfaOtvj7Jr7HAnwDsHfI+HvfEGKRGX4Yq7em0OiUBxZLKasAT4H467EHSymrAwuBz7KyYSFEqnky
5Uk8PTFIif+dx+xZtJmRq75gxIqJPLr2b8IHRryaberlSGsmrTwyc75x2uzxzPnyWzSanB0Pk6HXMAOaNW/Mye
Nns9xtpktGTy4pQ/TEvOKP3czSjLpN69KzTh+61OhOwYIFaNLB/RWp6H/fvDIXXUxa62tzqUOfuv3o7tKTAoU

K4N6+MQBbf99GP9f+DG02jNDAUAZPHpRujv8VOfUee53ehBbN2zbq7KGUMv7TehUQ3/G+Sff/GaCDvhW
FEIOBwQANrGtQwbx0suVh/iFYOSUeZVo52vAkMCxZTLh/CFZORYEb2hgHG8IDtDHH1u/n2Pr9ALQd250wv
8TWkYmpCc7NajGn9eeZ+2mT6OPVje59OgJw8dxlHIs5JCxzcLInwD8ww9uq7FyBn5ZqByVYW1vh5uFKbGw
ce3b4ZDqvAYN60ruv9l4i589eoljxxLycitnj75fxvOK179iSTRu3ZXq9tn1b06KHdiDGjQs3sHVK7LazdSxKSIpup
SehTzCzMMPE1ARNnEYb459+11N112r4P/TnSai2CB7eeYQKNSqwd9O+ZHGt+7amRXdPXS43k+VS1NGWk
BTdbdpcCifkkjQm2C9Y7/rVXKvh/zAgIZcjO49SweUD9v3lQ3hweEL8zj92MWO5/vNl/wU9B3SmS+92AFw6d
xUHJwfgAgD2TvYEBgSlvbIReBOGN79tLZqUJT9++rnu/zjSKL5SysVSShcppUvKIgPw74U72JV0xKa4LaZ5TXFp
XZeL3qeTxVz0Pk3tDg0AKFWtHNERUTwNCgfA3MYCACsnG5w9a3E6SevlfdfK+N/1Jdw/6+dDVv66luYNO9
O8YWd2b/ehY7c2AFRzqULE00gCA4JfsYVErtWaU8/Zk3rOnuzY4s0XY2dlqcgALFuyOuEE/o7te+nSvT0ANW
pW5enTSAIy+SFgbmFGXdea7Ny+79XBKWxesZUhzYYypNlQju46RtNO2tF0H1R/n2cRzwgNTP36nz92gYYtt
b/Tpp09OLbnn3T3EegbxAfV3id/gfyAtvA8uP0gVdzWFVsTTsQf2/0PHh21rZ73q6Wdy4VjF2nQsj4AHp2a8I8
ul3+8j+tdP+hxIO8nyaVaPWce3HoIkOwcUD3Puty/cT/dn8uYrV62gbZuPWnr1pO9Ow/Qvqv2YKJqjUpEPo0
k6DWcl8oOKWWGH8bqbWvRvCOEqCOl/AfoDhwBquXEhjVxGtZOWcaIlZMwMTXh2Pr9+N16RP2e2g+rw6
u9ubz/HJXcqjP94A+8iH7ByrGJdyscvPBTCluZExcby9rJvxL19FnCMpfW9ZIVnuzy8T6Mm0cDDp/ZQXR0DJ99
nDjqbPm6BYwfNZUA/yD6D+7BRyMHYGtnw57Df+Kz9zDjR03LsTxS8t59gCZNG3Lqwl6io6IZOSxxRNCajUsY
/fEk/P0DGfRRb0aMGoSdfVEO/bOFvXsO8cmISQC0bOXBAZ+jREVFZyuXEz4n+bBxLX4/spyYmOd8NebrhG
WzV87km7HzCQkIZcnspXyxYCL9x/Xl9uU77Fy7CwArWysW7viJQmaFkBpJx4HtGeA2iOvnrnNox2EW7VpAX
Gwct6/cZvvqHenmctLnJLUa12T5kWU8j37O15/OT1g2c8V05o/7jtCAUJbO+ZWJP0+g79i+3Ll8h11rd6e7/vX
zNzi84zALdv5EXFwcty/fYccfOwEYONGLMhVLIyUEPArg+89/yNbrmdTYqXM5de4i4eFPcW/Xi2FevenYulmO
bT89B7yP0rBJPfae/Jvo6BgmjExsqS1Z8z2TPplBYEAwvQd1ZdDHfShqZ8OWg2s5tPcok0bPpKidDZu8V2JmX
hiNRtJvSHea1+vCs8hn6ew1e3J6NJkQwhP4HjAFlkop56ZYLnTLWwBRQD8p5dlUG8rMPo25CuYkIURJYAdw
CKgL3AJ6A1cBFyllsBDCBfhaStkovW0NLdnF6F607U+vGTqFNEXFPn91kAEY662cTYXxdjSoWzln3s2g03pOk
mVcq3daZvjzZtuD7enuSwhhCtwEPIBHwCmgu5TyapKYFsAItIXmQ+B7KeWHWUg9wdvWotFIKVN+S6xk/B
Mp5Wmg0etMSFEUJT053KKpBdyWUt4FEEKsBdqiPeCO1xZYKbWtkONCiCJCCEcpZerx3hlkvIdOiqIoSqbO0
QghBgshTid5DE6xuWLAwyTTj3TzMhuTKW9Ni0ZKeR+oZOg8FEVRMiMzo86klIuBxemE6OtaS9lkykhMprw
1hUZRFOW/KIe/H/MIKJFkujiQ8vIfGYnJFNV1piiKYsRy+Fpnp4ByQohSQoh8QDdgS4qYLUAfoVUbeJKd8zOg
WjSKoihGLU7m3Fc2pZSxQoiPgd1ohzcvk1JeEUJ8pFu+CO3o3BbAbbTDm/tnd7+q0CiKohixnL60jJRyB9pikn
TeoiTPJTA8J/epCo2iKIoRexNufKYKjaIoihH775cZVWgURVGMmjHf0CyjVKFRFEUxYqrQvKUi5Ou7u15G2eUv
YugU0pSngKmhU9DrpUx990tjUDaP5auDDMRYryl25VrO3oTPmOTkqDNDUYVGURTFiBnzDc0yShUaRVEUI
/YmXGFfFRpFURQjps7RKIqiKLlKtWgURVGUXBWXqes3GydVaBRFUYyYujKAoiiKkqvUqDNFURQlV6kWjaIoi
pKrVItGURRFyVWqRaMkU6VhNXPpPHYCJqQkH1u5l68K/UsX0nuaFs1t1nkc/Z/FnP3H/8l0Avj2yiJhn0WjiNM
TFxTGl9TgA3vmgJP1nD6FAoQIEPQpk4ajviI6MznRun80YRT332sREP2faJ7O5celmqhinEo7MXjQNiyLmXL90
kykjZhL7Unu5nRp1nBkzfSR58uYhPPQJQzqMSFjPxMSE33ctIdA/mNF9xmc6tzEzRlCncW2eR8cwY/Rcbly6lS
rGsYQDMxdOwaKIBTcu32TaiNnEvoyleh1n5v02E9+H/gAc2HGIZd+uJF/+fCzc9D358uXFNI8pPtsPsvTr5ZnOL
d7l6cOp3fhDnkc/Z87oedy8rD/HqQu+wMLKnJuXbjFz5FxiX8bi2rQuXmP7o5Ea4mLj+HHqAi6dupylPCo2dK
bLlP6YmJpwZN0+di/8O1VM16n9qeRWnRfRz1n+2c88vHIPgMb9W+DazR0hBEfW7mXfsmS3JMFjUGs6Ter
DmGoDeBYWkaX8kvpi9mc0bFKP6KgYPh85jasXb6DuvNuqRJ8WN6dsNAnAJQu+y5zfphKxSrv
M3/2ApYtWJXtfF6d73wOHT2JtVUR/l616NUrvCZvwiVo/rO3chZCLBVCVHhFjK0Q4oQQ4pwQn46cQeEEC
665/eFEEUznY+JCX1nDGJe35mMazKK2m3q41SueLKYqm7VcSjlyKcNh/PrhEX0mzk42fJZ3aYwqcWnCUUGY
OD/hrFu7u9MaDaa07tP0HJIu8ymRr3GtSlRujjt63Zn1th5TJj7qd64EV98xB+L19OhXg8inkTQtnsrAMwszBg/
91PG9Pucro368PmgycnW6z6oM/du/ZvpvADqNP6QEqWK07leT+aM+4Zxc0brjRs+aQhrlmyks2svnoZH0qZ
7i4Rl509coo/HQPp4DGTZtysBePH8BR93HkNvj4H09hhInUa1qFg93bdLmmo3rkXxUsXp4dqHr8bPZ8ycUXrj
hkwaxPolf9LDtS8RTyJp2b05AGeOnKW/xyC8mg5h7qdfM+5r/a//qwgTE7pP9+LHfrOY5jGamm3q4Vg2+Xu
sUqNq2JVyZHKjEaya+As9Zw0CwOm9Erh2c2dO2wnMaP4ZlRvXwK6kQ8J6Vo42fFC/CiGPgrKUW0oNm9SjZ

OkSeNRqz+RPZ/HlvAl6486cvEC/jsN49CD5LenDw58yc+LX/PoaCky8di08WDR/5mvbX0bJTPwzVkZdaHT3rN
abo5RyoJTy6is24Q5cl1JWk1IezvkME5VxLkvAfT+CHgYQ9zKW41uPUMOjVrKYGh61OPLnAQDunLtJYYvCFL
GzSne7jqWduH5C+2NePnyBms1rZzq3hp6u7NiwS7uNs1cxtzDDxs4mVVxN1+rs26bNb9v6XTRqrq3Nnu2bs
H/HQQIeBwIQFhKesI6doy313Ovw9x/bMp0XQINm9dixcTcAV85exczSDBs761RxLq7V2b/tIAA7NuyigafrK7
cdHaVt+eXJm4c8efNAFrsgXJvVY/fGPQBcPXstzRyr16vGwe3aHHdt2EP9ZvV0ecQkxBQsVCDLeZRyLkvgv/4E
Pwwk7mUsp7cepWpTl2QxVZvW5PgmbQ73zt2ioHlhLGyL4FC2GPfO3eJlzAs0cRpunriKc7PE92fnyf3YNGdVj
n1YuXs25K912hbThTOXMbc0x9Y+9Xvu2qUbPH6Y+nb0ocFhXDp/NaFF/Tq4OFfG0sL8te0vo6TUZPhhrIyu0
AghSgohrgkhFgBngV+FEKeFEFeEEF8miUvaCokUQswSQlwQQhwXQtgLIZyBeUALIcR5IURBIcRCfdvKCVYON
oT6hSRMh/qFYOVgnSLGmhDf4MQY/xCs7LUxEsnnq6YyY9tXuHX3SIh5ePMB1T1qAvBhy7pYO2a6sYWtgy3
+voEJ0wF+Qdil2I6ltSURTyKJi9Ne0TjQLwg7B23MO2VKYG5pzi9//sDvu5fSsnOzhPU+nT6SH2YuQGqy9ia3d
bAl0DfxKDrQNwhbB9tX5pY0pnKNCvzuvZRvV/2PUu+VTJhvYmLCSu+l7Lz4NycPnebKuWtZyrGoQ9FkOQb5
BVHUIcXrZ2VB5JNI4uI0emPqe9bj94O/8b8Vs5j76ddZyqOIvTVhvonvsTC/UIqk+PAuYm9NaJKYcH/t+9D3xk
PK1fqAwkXMyFsgH5Xdqie8l6o0cSE8IJRH17LWKtXH3tEWf1//hOkA3wDsHexybPtvEw0yww9jZaznaMoD/a
WUw4QQ1lLKUCGEKbBPCFFFSnkxRXxh4LiUcpIQYh4wSEo5UwgxBXCRUn4MIISYlIFtZYnQNzPF712I1FHxl5
eY3mEi4YFhWNhYMn7VVHzvPObGyassGfszfaZ50X5UF856n8rSEV56+81ITB5TUz6oUp6hnT8hf8H8/LZ1IZf
OXOWd0iUIDQ7j+sWb1KjjnOm8tPtNPS9VbnrWi4+5fukm7Wp1IzoqmjqNP2Tespl0du0FgEajoY/HQMwszP
jfrzMoXb4Ud2/cy5Uc0whKeHp411EO7zpK1Q8r4zW2H2O6jUsd/8pE9MzL0O8R/O88ZveizXyyajLPn8Xw8
Np94uLiyFsgHy0+7sB3vXO2yygj7zklY96E181YC82/UsrjuuddhBCD0ebqCFQAUhaHF0B8380ZwAP9MrItvXT
rDQaoZe1MObNSyZaH+odg7Zh4dGntaENYQGjyGL8QbJwSj3KtHWwIDwwDSPj/acgTzuw+QRnnctw4eRW
/O4/5X+/pADiUcsS5cY2MpEvnfu1p17M1AFcvXMfByY4LumX2jrYE+Yckiw8PCcfc0gxTU1Pi4uKwc7QlKEAb
E+AXRHjoE2KiY4iJjuHc8QuUq1CG96uUp0HTetRzr02+/PkwMy/M9J8mM+XjGenm1rFfO9r21J7/uXb+OnZ
Oia0TOydbggOCk8WHhz5JlVt8TFRkVELcPz4nyDNnNJbWljzRnVQGiHwaydl/zlPbrVaGC037vm1p1VN7Huj6
+RvJcrR1tCUkIPnr9yT0CWaWZpiamhAXp8HW0ZbgFDEAF05coti7TlhaWfAk7GmGcokX7h+KlVPie8zK0Zrw
wOTvsTD/EKydbLijmy7iYEO47n14dL0PR9f7ANBubHfC/EKwfdcBm+J2TN75lXabDjZ8sW0ec9pN4GlQeKby
6zmgM116twPg0rmrODg5gO5dZ+9kT2BAzpz/edsYc0slo4yu60znGYAQohTwGeAupawCbAcK6Il/KRPLfhx6
CmgmtqWXlHKxlNJFSumSssgA3L1wG4dSjtiWsMM0bx5qt3blrPepZDFn957CtWMjAMpUe4+oiCjCA8PIXzA
/BQprU8lfMD+VGlTl0Y0HAFjYWMbnT9sRndm3eneG8t2w/C96egygp8cADuw8TIvOngBUql6ByIhIQgJTfwi
ePnoO91ba/Fp18eTgLu1prYO7j+D8YVVMTU21+VWvwP1b//Lz7F9oWaMjbWp1YdJH0zh15OwriwzAn8v/
Tjh5f3DXEVp00nbFVaxegcinzwhJ8eEJcOboOdxaNQSgRWdPDu8+CoC1bWL3ZAXn9xEmgiehTyhibYmZhRk
A+Qvko2b9Gvx7+0GGXjuAv1ZsxqvpELyaDuHw7qM069RUu4/qH/AsjRzPHTtPw5baHD07N+XInmMAFCvpl
BDzXqVy5MmbN9NFBuD+hdvYlXTEprj2PebSuh4XvE8ni7ngfZraHbQ5lKpWjuiIqISCYW5jAYCVU1GqeX7IqS
1H8b3xgLEuA5nkOpxJrsMJ8w9hZqtxmS4yAKuXbaCtW0/auvVk784DtO+qLdRVa1Qi8mlkwoGLkjlxGk2GH8
bKWFs08SzQFp0nQgh7oDlwwAi2lYomTsOKKUsZt3IKJqYmHFy/j8e3HtK4p/YDymf1Hs77nKGqW3W+ObSA
F7rhzQAWRYvwyWLtsGDTPCYc23yYiwfPAVCnjStN+mhHL53edZxDuiPSzDi67x/qudfm73/WEhMdw5ej5yQ
s+37VPGZ8+j+CA0L4ceZCZi+axtDxA7lx+Rab12wH4P6tf/ln/wnW+CxHajT8/cc27mShC0qfY/uOU9f9QzYeW
01M9HNmjv5fwrL5v89l9mdfERwQs+zfmHGwikMGefFzcu32LJGe6K5cauGdOjThrjYOJ7HvGDyUG3rr6i9
DZO/n4CpiQnCxIR9W/dzdO8/Wcrx+L4T1Gn8IWuO/s7Bjm+Pr4MKqYdm8lbP539hvCAkIYdGsJUxb8AUDx/Xn
1pXbbF+zE4CGLRrQrJMHsbGxPI95wbShry7G+mjiNKyd8iujVk7CxNSEo+v343frFO/WPbBMSHjOVM8gbvbyXSEN+A7DRzrb1dzm/Mrtze0Ke/9vd2+9Ab8t/NxHl/NS47GnOXtPGNMxcVbyr1Q+gN6xVvu
hWZpS95fsZ/rwJeHI9W/vKLUbXopFS3gcqJZnul0ZcoyTPzZI83whs1D1fDizP5LZKZiFtRVGUXPEmnKMxukKjKIq
iJDK2XqesMNbBAIqiKAqvbzCAEMJaCOEthLil+z/Vt8mFECWEEPt133W8IoTQf5mMFFShURRFMWKv8Qubn

wP7pJTlgH266ZRigU+llB8AtYHhr7oUGKhCoyiKYtSklBl+ZFNbYIXu+QqgnZ5c/KSUZ3XPI4BrQLFXbVido1EUR
TFir/E2AfZSSj/QFhQhRLrXDNKNEK4GnHjVhlWhURRFMWKZ+R5N0iuY6CyWUi5Osnwv4JBqRZiUmZyEEGb
An8AnUspXfvtYFRpFURQjlpkWja6oLE5neZO0lgkhAoQQjrrWjCMQmEZcXrRFZrWUclNG8lLnaBRFUYyYRm
oy/MimLUBf3fO+wOaUAUJ7tdRfgWtSyvkZ3bAqNIqiKEbsNQ4GmAt4CCFuob0w8VwAIYSTECL+dqz1gN5A
Y93tV84LIVro31wi1XWmKIpixF7XFzallCFobxaZcr4v0EL3/Ahp3BUlPUZ3rbO3jRBicNKTdcZC5ZV5xpqbyitzjD
Wv/zLVdWZ4g18dYhAqr8wz1txUXpljrHn9Z6lCoyiKouQqVWgURVGUXKUKjeEZa1+wyivzjDU3lVfmGGte/1l
qMlCiKIqSq1SLRlEURclVqtAoiqIouUoVGkVRFCVXqUKjGC0hRPF0lrV+nbkoipJ1ajDAaySEmJLOYimlnPHakl
CCFEA+AgoC1wCfpVSxhoil6SEEDeAZlLK+ynmDwAmSSnLGCSxxDzeAxaivY9HJSFEFaCNlHKmIfMCEEKcBn4
D/pBShhk6n3i612ws8C5JLoElpWxsoHys01supQx9Xbm8yVSheY2EEJ/qmV0IGAjYSCnNXnNKAAgh1gEvgcN
Ac+BfKWWG7gWem3QX6/seaCGlvKWbNwHoATSXUj4ycH4H0X5o/iKlrKabd1lKWcmQeenyKAv0B7oC8UV
njzTwH7wQ4gKwCDgDxMXPl1KeMVA+9wCJ/ut3SSll6dec0htJFRoDEUKYA6MAL2A98I2UUu/9H15DLpeklJ
V1z/MAJ6WU1Q2RS0pCCHfgF7S3lR0I1ARaGcNRuhDilJSyphDiXJJCc15K6Wzg1BIIIUyAVmhbXhpgGfC9oY7
UhRBnpJQ1DLFvxXDU1ZtfM11TfQzQE+19uasbwYfmy/gnUspY7S0njIOUcp8Qoh9wADgGuEspYwyaVKJgIU
QZtEfECCE6AX6GTSmRriuvP9or7/4JrAZcAR/A+TXnEt9FtVUIMRzYBDyPX27AwpfuAZWU8uzryuVNplo0r5E
Q4iugA9pvHv8spYw0cEoACCHigGfxk0BBIEr3XEopLQyUVwSJ3Rr50RbEOEPnFU8lURrt77IuEAbcA3pKKf81Z
F6gbTkA4WhvUvWnlPJ5kmWbpJQdXnM+Kbuokn3wGKqLSgixP53F0lDnjt40qtC8RkIIDdqjuFiS/6EZxQenkn
FCCFNgrpRyrBCiMGAipYwwdF7xhBClpZR3U8wrJaW8Z6icdDkUBIahbVlJtOcFF0kpow2Zl5K7VKFRlCwSQvg
Y6xGvEOJsyvNsxnB+RAixHniKthsPoDtQRErZxXBZaQkhKgEVgALx86SUKw2X0ZtDnaNRlKw7J4TYAmwgsesR
KeUmQyUkhHgfqAhYCiGSdo9ZkOQD1IDKSymrJpnerxuJZlBCiKlAI7SFZgfa0ZdHAFVocoAqNIqSddZACJC0VS
PRnug2lPJoR5kVAZJ+qTUCGGSIhFI4J4SoLaU8DiCE+BA4auCcADoBVYFzUsr+Qgh7YKmBc3pjqK4zRXkDCSH
qSCn/MXQeKQkhrqEthg90s94BrqEdei2llFUMlNdJKWUt3SAKN7SF+bKUsqIh8nnTqBaNomSREOI3UoyeAp
BSDjBAOgAIIcZJKecBPYQQ3VMul1KONEBaSXkaeP9pOS2EKAIsQftl0kjgpEEzeoOoQqMoWbctyfMCQHvA1
0C5xLum+/+0QbNIgzEM/U5KCFFPSnkUGK0bAr5ICLELsJBSXjRwem8M1XWmKDlE9y38vcY6Ek1JLX4knr5R
ekrOUS0aRck55dCeczAYIcRW9HTnxZNStnmN6fwXvNR1gRYTQvyQcqERdDW+EVShUZQsSnLlgnj+wHgDpR
Pva93/HQAHYJVuujtw3xAJGblWQBO0IwcNcmHPt4HqOlOUN5AQ4pCUssGr5ilaQoiqUkqDf5/nTaVaNIqSR
UKIfVJK91fNMxDbpJehEUKUAmwNnJMx8xVCTARKkvw+OQYbQfgmUYVGUTJJd6O4QkBRIYQViReKtACcDJ
ZYcqOBA0KI+OudlQSGGC4do7cZ7XXX9pLkPjlKzlBdZ4qSSUKIUcAnaIvKYxILzVNgiZTyJwOllowQlj/wvm7yetI
rOCvJGdt9hN40qtAoShYJIUZIKX80dB5JCSEaSyl9UlznLIEhr8NmzIQQM4FjUsodhs7lTaQKjaJkg7Fd8VcI8aW
UcqpuyG5KUp1z0E83grAw2tt4vETduiNHqUKjKFmU1hV/pZSdDJmXohgbVWgUJYuEEJdIvOJv1fgr/kopW79
i1Vynu25XH1KPolJfQExCCPG+lPJ6Wrd0VrdyzhIq1JmiZF20lFIjhIgVQlgAgYBBbkmsxw7gOHAJ7ZWRFf3GAI
OBb/QskyS/BYSSRarQKErWGfMVfwtIKccYOgljJ6UcrPvfzdC5vMlU15miZIEQQgDFpZQPddMlMaIr/gohRqM
tfNvQnuAGQEoZarCkjJyxDex4k6hCoyhZFH/lX0PnoY8QYjgwCwgn8XpsUkppLF17RkUN7MhdJoZOQFH+w4
4LIWoaOok0jAHKSilLSilL6R6qyKStE+AK6Md5JHfsCm9OdQ5GkXJOjfgIyHEfeAZid+9MMjtiFO4AkQZ
Oon/kBgjHtjxn6cKjaJkXXNDJ5COOOC8EGI/yc/RqOHNKejOt1004oEd/3nqHI2iZIMQwhUoJ6X8TQhhC5hJK
e8ZQV599c2XUq543bn8FyQ932ZsAzveBKrQKEoW6U4guwDlpZTvCSGcgA1SynoGTk3JJCHEz8ByKeUpQ+fy
JlKFRlGySAhxHqgGnJVVSVtPNu2gM52iEEPfQc0tnNSBAPyHEVeA94F+M73zbf546R6MoWfdCSimFEBJACFH
Y0Akl4ZLkeQGGM2BtoFz+C4z5fNt/nmrRKEoWCSE+A8oBBHsAcYADwh7HdOiCeEOKIlNLV0Hkobx9VaBQlG4
QQHkBT3eQeKaW3IfOJl+IikSZoWzhDpZRVDZSS8hZTXWeKkj2XgIJoz4dcMnAuSX1D4jmaWOA+2u4zRXntV
ItGUbJICDEQmAL4oD153BCYLqVcZtDEACFEAaAjyW8TIKWU0w2UHCHEDqCulDNFN26C9H
XB5w2YGQohdaK9zdhbtlzcBkFLquxy+ouQq1XWmKFn3CIhIMh0BPDRQLikVl1J6GjoJRQFVaBQlOx4DJ4QQ
m9GeD2kLnBRCjAGQUs43YG7Hb8Wfe/uQFyARJuLy3R/m33F0LcRXut
M/UFRMVg1DkaRXmDCCHeTW+5lPLf15WLosRThUZRskgI4QJMAt4lSe+AajUoSnKq0ChKFulGnY1F+/0ZTfx
81WpQlOTUORpFybogKeUWQyehKeyehKMZOtWgUJYuEEO5Ad2AfyW8utslgSSmKEVItGkXJuv7A+0BeErvOJKAK
jaIkoQqNomRdVSllZUMnoSjGzsTQCSjKG+CEEKKCoZNQFGOnztEoShYJIa4AZdFeWPMF2vvSSDW8WVGSU4VG
UbJICOGW1nwp5YPczkVRjJkqNIqiKEqOUudoFEVRlByVGIpWXLhdfldDygD3UF+KVJQ0qUKjUKFmU1p

```
cj1fBmRUlOdZ0pShbpCkoJoLHueRTqb0pRUlEtGkXJIiHEVLR3riwvpXxPCOEEbJBS1jNwaopiVNTRl6JkXXugD
fAMQErpiwEvqKkoxkoVGkXJuhdS2yUgAYQQhQ2cj6IYJVVoFCXr1gshfgGKCCEGAXuBJQbOSVGMjvrCpqJkn
S2wEXgKlAemAE0MmpGiGCE1GEBRskgIcVZKWT3FvIvqezSKkpxq0ShKJgkhhgLDgNJCiItJFpkDRw2TlaIYL9
WiUZRMEkJYAlbAHODzJIsipJShhslKUYyXKjSKoihKrlKjzhRFUZRcpQqNoiiKkqtUoVEURVFylSo0iqIoSq5ShUZ
RFEXJVf8HKxtBprhjCwsAAAAASUVORK5CYII=\n",
     "text/plain": [
      "<Figure size 432x288 with 2 Axes>"
     ]
    },
    "metadata": {
     "needs_background": "light"
    },
    "output_type": "display_data"
   }
  ],
  "source": [
   "sns.heatmap(df.corr(),annot=True)"
  ]
 },
 {
  "cell_type": "markdown",
  "metadata": {},
  "source": [
   "### Seperating features and target label"
  ]
 },
 {
  "cell_type": "code",
  "execution_count": 12,
  "metadata": {},
   "outputs": [],
   "source": [
    "features = df[['N', 'P','K','temperature', 'humidity', 'ph', 'rainfall']]\n",
    "target = df['label']\n",
    "#features = df[['temperature', 'humidity', 'ph', 'rainfall']]\n",
    "labels = df['label']"
   ]
  },
  {
   "cell_type": "code",
   "execution_count": 13,
   "metadata": {},
   "outputs": [],
   "source": [
    "# Initialzing empty lists to append all model's name and corresponding name\n",
    "acc = []\n",
    "model = []"
   ]
  },
  {
   "cell_type": "code",
   "execution_count": 14,
   "metadata": {},
   "outputs": [],
   "source": [
    "# Splitting into train and test data\n",
    "\n",
    "from sklearn.model_selection import train_test_split\n",
    "Xtrain, Xtest, Ytrain, Ytest = train_test_split(features,target,test_size = 0.2,random_state =2)"
  ]
 },
 {
  "cell_type": "markdown",
  "metadata": {},
  "source": [
   "# Decision Tree"
  ]
 },
 {
  "cell_type": "code",
  "execution_count": 15,
  "metadata": {},
  "outputs": [
   {
    "name": "stdout",
    "output_type": "stream",
    "text": [
     "DecisionTrees's Accuracy is:  90.0\n",
     "              precision    recall  f1-score   support\n",
     "\n",
     "       apple       1.00      1.00      1.00        13\n",
     "      banana       1.00      1.00      1.00        17\n",
     "   blackgram       0.59      1.00      0.74        16\n",
     "    chickpea       1.00      1.00      1.00        21\n",
     "     coconut       0.91      1.00      0.95        21\n",
     "      coffee       1.00      1.00      1.00        22\n",
     "      cotton       1.00      1.00      1.00        20\n",
     "      grapes       1.00      1.00      1.00        18\n",
     "        jute       0.74      0.93      0.83        28\n",
     " kidneybeans       0.00      0.00      0.00        14\n",
     "      lentil       0.68      1.00      0.81        23\n",
     "       maize       1.00      1.00      1.00        21\n",
     "       mango       1.00      1.00      1.00        26\n",
     "   mothbeans       0.00      0.00      0.00        19\n",
     "    mungbean       1.00      1.00      1.00        24\n",
     "   muskmelon       1.00      1.00      1.00        23\n",
     "      orange       1.00      1.00      1.00        29\n",
     "      papaya       1.00      0.84      0.91        19\n",
     "  pigeonpeas       0.62      1.00      0.77        18\n",
     " pomegranate       1.00      1.00      1.00        17\n",
     "        rice       1.00      0.62      0.77        16\n",
     "  watermelon       1.00      1.00      1.00        15\n",
     "\n",
     "    accuracy                           0.90       440\n",
     "   macro avg       0.84      0.88      0.85       440\n",
     "weighted avg       0.86      0.90      0.87       440\n",
     "\n"
    ]
   }
  ],
  "source": [
   "from sklearn.tree import DecisionTreeClassifier\n",
   "\n",
   "DecisionTree = DecisionTreeClassifier(criterion=\"entropy\",random_state=2,max_depth=5)\n",
   "\n",
   "DecisionTree.fit(Xtrain,Ytrain)\n",
   "\n",
    "predicted_values = DecisionTree.predict(Xtest)\n",
    "x = metrics.accuracy_score(Ytest, predicted_values)\n",
    "acc.append(x)\n",
    "model.append('Decision Tree')\n",
    "print(\"DecisionTrees's Accuracy is: \", x*100)\n",
    "\n",
    "print(classification_report(Ytest,predicted_values))"
   ]
  },
  {
   "cell_type": "code",
   "execution_count": 16,
   "metadata": {},
   "outputs": [],
   "source": [
    "from sklearn.model_selection import cross_val_score"
   ]
  },
  {
   "cell_type": "code",
   "execution_count": 17,
   "metadata": {},
   "outputs": [],
   "source": [
    "# Cross validation score (Decision Tree)\n",
    "score = cross_val_score(DecisionTree, features, target,cv=5)"
   ]
  },
  {
```

```
 "cell_type": "code",
 "execution_count": 18,
 "metadata": {},
 "outputs": [
  {
   "data": {
    "text/plain": [
     "array([0.93636364, 0.90909091, 0.91818182, 0.87045455, 0.93636364])"
    ]
   },
   "execution_count": 18,
   "metadata": {},
   "output_type": "execute_result"
  }
 ],
 "source": [
  "score"
 ]
},
{
 "cell_type": "markdown",
 "metadata": {},
 "source": [
  "### Saving trained Decision Tree model"
 ]
},
{
 "cell_type": "code",
 "execution_count": 19,
 "metadata": {},
 "outputs": [],
 "source": [
  "import pickle\n",
  "# Dump the trained Naive Bayes classifier with Pickle\n",
  "DT_pkl_filename = '../models/DecisionTree.pkl'\n",
  "# Open the file to save as pkl file\n",
  "DT_Model_pkl = open(DT_pkl_filename, 'wb')\n",
  "pickle.dump(DecisionTree, DT_Model_pkl)\n",
  "# Close the pickle instances\n",
  "DT_Model_pkl.close()"
 ]
},
{
 "cell_type": "markdown",
 "metadata": {},
 "source": [
  "# Guassian Naive Bayes"
 ]
},
{
 "cell_type": "code",
 "execution_count": 20,
 "metadata": {},
 "outputs": [
  {
   "name": "stdout",
   "output_type": "stream",
   "text": [
"Naive Bayes's Accuracy is:  0.990909090909091\n",
"              precision    recall  f1-score   support\n",
"\n",
"       apple       1.00      1.00      1.00        13\n",
"      banana       1.00      1.00      1.00        17\n",
"   blackgram       1.00      1.00      1.00        16\n",
"    chickpea       1.00      1.00      1.00        21\n",
"     coconut       1.00      1.00      1.00        21\n",
"      coffee       1.00      1.00      1.00        22\n",
"      cotton       1.00      1.00      1.00        20\n",
"      grapes       1.00      1.00      1.00        18\n",
"        jute       0.88      1.00      0.93        28\n",
" kidneybeans       1.00      1.00      1.00        14\n",
"      lentil       1.00      1.00      1.00        23\n",
"       maize       1.00      1.00      1.00        21\n",
"       mango       1.00      1.00      1.00        26\n",
"   mothbeans       1.00      1.00      1.00        19\n",
"    mungbean       1.00      1.00      1.00        24\n",
"   muskmelon       1.00      1.00      1.00        23\n",
"      orange       1.00      1.00      1.00        29\n",
"      papaya       1.00      1.00      1.00        19\n",
"  pigeonpeas       1.00      1.00      1.00        18\n",
" pomegranate       1.00      1.00      1.00        17\n",
"        rice       1.00      0.75      0.86        16\n",
"  watermelon       1.00      1.00      1.00        15\n",
"\n",
"    accuracy                           0.99       440\n",
"   macro avg       0.99      0.99      0.99       440\n",
"weighted avg       0.99      0.99      0.99       440\n",
    "\n"
   ]
  }
 ],
 "source": [
  "from sklearn.naive_bayes import GaussianNB\n",
  "\n",
  "NaiveBayes = GaussianNB()\n",
  "\n",
  "NaiveBayes.fit(Xtrain,Ytrain)\n",
  "\n",
  "predicted_values = NaiveBayes.predict(Xtest)\n",
  "x = metrics.accuracy_score(Ytest, predicted_values)\n",
  "acc.append(x)\n",
  "model.append('Naive Bayes')\n",
  "print(\"Naive Bayes's Accuracy is: \", x)\n",
  "\n",
  "print(classification_report(Ytest,predicted_values))"
 ]
},
{
 "cell_type": "code",
 "execution_count": 21,
 "metadata": {},
 "outputs": [
  {
   "data": {
    "text/plain": [
     "array([0.99772727, 0.99545455, 0.99545455, 0.99545455, 0.99090909])"
```

```
      ]
     },
     "execution_count": 21,
     "metadata": {},
     "output_type": "execute_result"
    }
   ],
   "source": [
    "# Cross validation score (NaiveBayes)\n",
    "score = cross_val_score(NaiveBayes,features,target,cv=5)\n",
    "score"
   ]
  },
  {
   "cell_type": "markdown",
   "metadata": {},
   "source": [
    "### Saving trained Guassian Naive Bayes model"
   ]
  },
  {
   "cell_type": "code",
   "execution_count": 23,
   "metadata": {},
   "outputs": [],
   "source": [
    "import pickle\n",
    "# Dump the trained Naive Bayes classifier with Pickle\n",
    "NB_pkl_filename = '../models/NBClassifier.pkl'\n",
    "# Open the file to save as pkl file\n",
    "NB_Model_pkl = open(NB_pkl_filename, 'wb')\n",
    "pickle.dump(NaiveBayes, NB_Model_pkl)\n",
    "# Close the pickle instances\n",
    "NB_Model_pkl.close()"
   ]
  },
  {
   "cell_type": "markdown",
   "metadata": {},
   "source": [
    "# Support Vector Machine (SVM)"
   ]
  },
  {
   "cell_type": "code",
   "execution_count": 24,
   "metadata": {},
   "outputs": [
    {
     "name": "stdout",
     "output_type": "stream",
     "text": [
      "SVM's Accuracy is:  0.9795454545454545\n",
      "              precision    recall  f1-score   support\n",
      "\n",
      "       apple       1.00      1.00      1.00        13\n",
      "      banana       1.00      1.00      1.00        17\n",
      "   blackgram       1.00      1.00      1.00        16\n",
   "    chickpea       1.00      1.00      1.00        21\n",
   "     coconut       1.00      1.00      1.00        21\n",
   "      coffee       1.00      0.95      0.98        22\n",
   "      cotton       0.95      1.00      0.98        20\n",
   "      grapes       1.00      1.00      1.00        18\n",
   "        jute       0.83      0.89      0.86        28\n",
   " kidneybeans       1.00      1.00      1.00        14\n",
   "      lentil       1.00      1.00      1.00        23\n",
   "       maize       1.00      0.95      0.98        21\n",
   "       mango       1.00      1.00      1.00        26\n",
   "   mothbeans       1.00      1.00      1.00        19\n",
   "    mungbean       1.00      1.00      1.00        24\n",
   "   muskmelon       1.00      1.00      1.00        23\n",
   "      orange       1.00      1.00      1.00        29\n",
   "      papaya       1.00      1.00      1.00        19\n",
   "  pigeonpeas       1.00      1.00      1.00        18\n",
   " pomegranate       1.00      1.00      1.00        17\n",
   "        rice       0.80      0.75      0.77        16\n",
   "  watermelon       1.00      1.00      1.00        15\n",
   "\n",
   "    accuracy                           0.98       440\n",
   "   macro avg       0.98      0.98      0.98       440\n",
   "weighted avg       0.98      0.98      0.98       440\n",
   "\n"
  ]
 }
],
"source": [
 "from sklearn.svm import SVC\n",
    "# data normalization with sklearn\n",
    "from sklearn.preprocessing import MinMaxScaler\n",
    "# fit scaler on training data\n",
    "norm = MinMaxScaler().fit(Xtrain)\n",
    "X_train_norm = norm.transform(Xtrain)\n",
    "# transform testing dataabs\n",
    "X_test_norm = norm.transform(Xtest)\n",
    "SVM = SVC(kernel='poly', degree=3, C=1)\n",
    "SVM.fit(X_train_norm,Ytrain)\n",
    "predicted_values = SVM.predict(X_test_norm)\n",
    "x = metrics.accuracy_score(Ytest, predicted_values)\n",
    "acc.append(x)\n",
    "model.append('SVM')\n",
    "print(\"SVM's Accuracy is: \", x)\n",
    "\n",
    "print(classification_report(Ytest,predicted_values))"
   ]
  },
  {
   "cell_type": "code",
   "execution_count": 37,
   "metadata": {},
   "outputs": [
    {
     "data": {
      "text/plain": [
       "array([0.97954545, 0.975     , 0.98863636, 0.98863636, 0.98181818])"
      ]
     },
     "execution_count": 37,
     "metadata": {},
     "output_type": "execute_result"
    }
   ],
   "source": [
    "# Cross validation score (SVM)\n",
    "score = cross_val_score(SVM,features,target,cv=5)\n",
    "score"
   ]
  },
  {
   "cell_type": "code",
   "execution_count": 27,
   "metadata": {},
   "outputs": [],
   "source": [
    "#Saving trained SVM model"
   ]
  },
  {
   "cell_type": "code",
   "execution_count": 28,
   "metadata": {},
   "outputs": [],
   "source": [
    "import pickle\n",
    "# Dump the trained SVM classifier with Pickle\n",
    "SVM_pkl_filename = '../models/SVMClassifier.pkl'\n",
    "# Open the file to save as pkl file\n",
    "SVM_Model_pkl = open(SVM_pkl_filename, 'wb')\n",
    "pickle.dump(SVM, SVM_Model_pkl)\n",
    "# Close the pickle instances\n",
    "SVM_Model_pkl.close()"
   ]
  },
  {
   "cell_type": "markdown",
   "metadata": {},
   "source": [
    "# Logistic Regression"
   ]
  },
  {
   "cell_type": "code",
   "execution_count": 29,
   "metadata": {},
   "outputs": [
    {
     "name": "stdout",
     "output_type": "stream",
     "text": [
      "Logistic Regression's Accuracy is:  0.9522727272727273\n",
      "              precision    recall  f1-score   support\n",
      "\n",
      "       apple       1.00      1.00      1.00        13\n",
      "      banana       1.00      1.00      1.00        17\n",
      "   blackgram       0.86      0.75      0.80        16\n",
    "    chickpea       1.00      1.00      1.00        21\n",
    "     coconut       1.00      1.00      1.00        21\n",
    "      coffee       1.00      1.00      1.00        22\n",
    "      cotton       0.86      0.90      0.88        20\n",
    "      grapes       1.00      1.00      1.00        18\n",
    "        jute       0.84      0.93      0.88        28\n",
    " kidneybeans       1.00      1.00      1.00        14\n",
    "      lentil       0.88      1.00      0.94        23\n",
    "       maize       0.90      0.86      0.88        21\n",
    "       mango       0.96      1.00      0.98        26\n",
    "   mothbeans       0.84      0.84      0.84        19\n",
    "    mungbean       1.00      0.96      0.98        24\n",
    "   muskmelon       1.00      1.00      1.00        23\n",
    "      orange       1.00      1.00      1.00        29\n",
    "      papaya       1.00      0.95      0.97        19\n",
    "  pigeonpeas       1.00      1.00      1.00        18\n",
    " pomegranate       1.00      1.00      1.00        17\n",
    "        rice       0.85      0.69      0.76        16\n",
    "  watermelon       1.00      1.00      1.00        15\n",
    "\n",
    "    accuracy                           0.95       440\n",
    "   macro avg       0.95      0.95      0.95       440\n",
    "weighted avg       0.95      0.95      0.95       440\n",
    "\n"
   ]
  }
 ],
 "source": [
  "from sklearn.linear_model import LogisticRegression\n",
    "\n",
    "LogReg = LogisticRegression(random_state=2)\n",
    "\n",
    "LogReg.fit(Xtrain,Ytrain)\n",
    "\n",
    "predicted_values = LogReg.predict(Xtest)\n",
    "\n",
    "x = metrics.accuracy_score(Ytest, predicted_values)\n",
    "acc.append(x)\n",
    "model.append('Logistic Regression')\n",
    "print(\"Logistic Regression's Accuracy is: \", x)\n",
    "\n",
    "print(classification_report(Ytest,predicted_values))"
   ]
  },
  {
   "cell_type": "code",
   "execution_count": 30,
   "metadata": {},
   "outputs": [
    {
     "data": {
      "text/plain": [
       "array([0.95      , 0.96590909, 0.94772727, 0.96590909, 0.94318182])"
      ]
     },
     "execution_count": 30,
     "metadata": {},
     "output_type": "execute_result"
   }
  ],
  "source": [
   "# Cross validation score (Logistic Regression)\n",
   "score = cross_val_score(LogReg,features,target,cv=5)\n",
   "score"
  ]
 },
 {
  "cell_type": "markdown",
  "metadata": {},
  "source": [
   "### Saving trained Logistic Regression model"
  ]
 },
 {
  "cell_type": "code",
  "execution_count": 35,
  "metadata": {},
  "outputs": [],
  "source": [
   "import pickle\n",
   "# Dump the trained Naive Bayes classifier with Pickle\n",
   "LR_pkl_filename = '../models/LogisticRegression.pkl'\n",
   "# Open the file to save as pkl file\n",
   "LR_Model_pkl = open(DT_pkl_filename, 'wb')\n",
   "pickle.dump(LogReg, LR_Model_pkl)\n",
   "# Close the pickle instances\n",
   "LR_Model_pkl.close()"
 ]
},
{
 "cell_type": "markdown",
 "metadata": {},
 "source": [
  "# Random Forest"
 ]
},
{
 "cell_type": "code",
 "execution_count": 36,
 "metadata": {},
 "outputs": [
  {
   "name": "stdout",
   "output_type": "stream",
   "text": [
    "RF's Accuracy is:  0.990909090909091\n",
    "              precision    recall  f1-score   support\n",
    "\n",
    "       apple       1.00      1.00      1.00        13\n",
    "      banana       1.00      1.00      1.00        17\n",
    "   blackgram       0.94      1.00      0.97        16\n",
    "    chickpea       1.00      1.00      1.00        21\n",
    "     coconut       1.00      1.00      1.00        21\n",
    "      coffee       1.00      1.00      1.00        22\n",
    "      cotton       1.00      1.00      1.00        20\n",
    "      grapes       1.00      1.00      1.00        18\n",
      "        jute       0.90      1.00      0.95        28\n",
      " kidneybeans       1.00      1.00      1.00        14\n",
      "      lentil       1.00      1.00      1.00        23\n",
      "       maize       1.00      1.00      1.00        21\n",
      "       mango       1.00      1.00      1.00        26\n",
      "   mothbeans       1.00      0.95      0.97        19\n",
      "    mungbean       1.00      1.00      1.00        24\n",
      "   muskmelon       1.00      1.00      1.00        23\n",
      "      orange       1.00      1.00      1.00        29\n",
      "      papaya       1.00      1.00      1.00        19\n",
      "  pigeonpeas       1.00      1.00      1.00        18\n",
      " pomegranate       1.00      1.00      1.00        17\n",
      "        rice       1.00      0.81      0.90        16\n",
      "  watermelon       1.00      1.00      1.00        15\n",
      "\n",
      "    accuracy                           0.99       440\n",
      "   macro avg       0.99      0.99      0.99       440\n",
      "weighted avg       0.99      0.99      0.99       440\n",
      "\n"
     ]
    }
   ],
   "source": [
    "from sklearn.ensemble import RandomForestClassifier\n",
    "\n",
    "RF = RandomForestClassifier(n_estimators=20, random_state=0)\n",
    "RF.fit(Xtrain,Ytrain)\n",
    "\n",
    "predicted_values = RF.predict(Xtest)\n",
    "\n",
    "x = metrics.accuracy_score(Ytest, predicted_values)\n",
    "acc.append(x)\n",
    "model.append('RF')\n",
    "print(\"RF's Accuracy is: \", x)\n",
    "\n",
    "print(classification_report(Ytest,predicted_values))"
   ]
  },
  {
   "cell_type": "code",
   "execution_count": 37,
   "metadata": {},
   "outputs": [
    {
     "data": {
      "text/plain": [
       "array([0.99772727, 0.99545455, 0.99772727, 0.99318182, 0.98863636])"
      ]
     },
     "execution_count": 37,
     "metadata": {},
     "output_type": "execute_result"
    }
   ],
   "source": [
    "# Cross validation score (Random Forest)\n",
    "score = cross_val_score(RF,features,target,cv=5)\n",
    "score"
  ]
 },
 {
  "cell_type": "markdown",
  "metadata": {},
  "source": [
   "### Saving trained Random Forest model"
  ]
 },
 {
  "cell_type": "code",
  "execution_count": 38,
  "metadata": {},
  "outputs": [],
  "source": [
   "import pickle\n",
   "# Dump the trained Naive Bayes classifier with Pickle\n",
   "RF_pkl_filename = '../models/RandomForest.pkl'\n",
   "# Open the file to save as pkl file\n",
   "RF_Model_pkl = open(RF_pkl_filename, 'wb')\n",
   "pickle.dump(RF, RF_Model_pkl)\n",
   "# Close the pickle instances\n",
   "RF_Model_pkl.close()"
  ]
 },
 {
  "cell_type": "markdown",
  "metadata": {},
  "source": [
    "# XGBoost"
   ]
  },
  {
   "cell_type": "code",
   "execution_count": 39,
   "metadata": {
    "scrolled": true
   },
   "outputs": [
    {
     "name": "stdout",
     "output_type": "stream",
     "text": [
      "[14:16:03] WARNING: C:/Users/Administrator/workspace/xgboost-win64_release_1.4.0/src/learner.cc:1095: Starting in XGBoost 1.3.0, the default evaluation metric used with the objective 'multi:softprob' was changed from 'merror' to 'mlogloss'. Explicitly set eval_metric if you'd like to restore the old behavior.\n",
      "XGBoost's Accuracy is:  0.9931818181818182\n",
      "              precision    recall  f1-score   support\n",
      "\n",
      "       apple       1.00      1.00      1.00        13\n",
      "      banana       1.00      1.00      1.00        17\n",
      "   blackgram       1.00      1.00      1.00        16\n",
      "    chickpea       1.00      1.00      1.00        21\n",
      "     coconut       1.00      1.00      1.00        21\n",
      "      coffee       0.96      1.00      0.98        22\n",
      "      cotton       1.00      1.00      1.00        20\n",
      "      grapes       1.00      1.00      1.00        18\n",
      "        jute       1.00      0.93      0.96        28\n",
   " kidneybeans       1.00      1.00      1.00        14\n",
   "      lentil       0.96      1.00      0.98        23\n",
   "       maize       1.00      1.00      1.00        21\n",
   "       mango       1.00      1.00      1.00        26\n",
   "   mothbeans       1.00      0.95      0.97        19\n",
   "    mungbean       1.00      1.00      1.00        24\n",
   "   muskmelon       1.00      1.00      1.00        23\n",
   "      orange       1.00      1.00      1.00        29\n",
   "      papaya       1.00      1.00      1.00        19\n",
   "  pigeonpeas       1.00      1.00      1.00        18\n",
   " pomegranate       1.00      1.00      1.00        17\n",
   "        rice       0.94      1.00      0.97        16\n",
   "  watermelon       1.00      1.00      1.00        15\n",
   "\n",
   "    accuracy                           0.99       440\n",
   "   macro avg       0.99      0.99      0.99       440\n",
   "weighted avg       0.99      0.99      0.99       440\n",
   "\n"
  ]
 }
],
"source": [
 "import xgboost as xgb\n",
 "XB = xgb.XGBClassifier()\n",
 "XB.fit(Xtrain,Ytrain)\n",
 "\n",
 "predicted_values = XB.predict(Xtest)\n",
 "\n",
 "x = metrics.accuracy_score(Ytest, predicted_values)\n",
    "acc.append(x)\n",
    "model.append('XGBoost')\n",
    "print(\"XGBoost's Accuracy is: \", x)\n",
    "\n",
    "print(classification_report(Ytest,predicted_values))"
   ]
  },
  {
   "cell_type": "code",
   "execution_count": 46,
   "metadata": {},
   "outputs": [
    {
     "name": "stdout",
     "output_type": "stream",
     "text": [
      "[08:54:44] WARNING: C:/Users/Administrator/workspace/xgboost-win64_release_1.4.0/src/learner.cc:1095: Starting in XGBoost 1.3.0, the default evaluation metric used with the objective 'multi:softprob' was changed from 'merror' to 'mlogloss'. Explicitly set eval_metric if you'd like to restore the old behavior.\n",
      "[08:54:45] WARNING: C:/Users/Administrator/workspace/xgboost-win64_release_1.4.0/src/learner.cc:1095: Starting in XGBoost 1.3.0, the default evaluation metric used with the objective 'multi:softprob' was changed from 'merror' to 'mlogloss'. Explicitly set eval_metric if you'd like to restore the old behavior.\n",
      "[08:54:46] WARNING: C:/Users/Administrator/workspace/xgboost-win64_release_1.4.0/src/learner.cc:1095: Starting in XGBoost 1.3.0, the default evaluation metric used with the objective 'multi:softprob' was changed from 'merror' to 'mlogloss'. Explicitly set eval_metric if you'd like to restore the old behavior.\n",
      "[08:54:47] WARNING: C:/Users/Administrator/workspace/xgboost-win64_release_1.4.0/src/learner.cc:1095: Starting in XGBoost 1.3.0, the default evaluation metric used with the objective 'multi:softprob' was changed from 'merror' to 'mlogloss'. Explicitly set eval_metric if you'd like to restore the old behavior.\n",
      "[08:54:48] WARNING: C:/Users/Administrator/workspace/xgboost-win64_release_1.4.0/src/learner.cc:1095: Starting in XGBoost 1.3.0, the default evaluation metric used with the objective 'multi:softprob' was changed from 'merror' to 'mlogloss'. Explicitly set eval_metric if you'd like to restore the old behavior.\n"
     ]
    },
    {
     "data": {
      "text/plain": [
       "array([0.99318182, 0.99318182, 0.99318182, 0.99090909, 0.99090909])"
      ]
     },
     "execution_count": 46,
     "metadata": {},
     "output_type": "execute_result"
    }
   ],
   "source": [
    "# Cross validation score (XGBoost)\n",
    "score = cross_val_score(XB,features,target,cv=5)\n",
    "score"
   ]
  },
  {
   "cell_type": "markdown",
   "metadata": {},
   "source": [
    "### Saving trained XGBoost model"
   ]
  },
{
 "cell_type": "code",
 "execution_count": 40,
 "metadata": {},
 "outputs": [],
 "source": [
  "import pickle\n",
  "# Dump the trained Naive Bayes classifier with Pickle\n",
  "XB_pkl_filename = '../models/XGBoost.pkl'\n",
  "# Open the file to save as pkl file\n",
  "XB_Model_pkl = open(XB_pkl_filename, 'wb')\n",
  "pickle.dump(XB, XB_Model_pkl)\n",
  "# Close the pickle instances\n",
  "XB_Model_pkl.close()"
 ]
},
{
 "cell_type": "markdown",
 "metadata": {},
 "source": [
  "## Accuracy Comparison"
 ]
},
{
 "cell_type": "code",
 "execution_count": 41,
 "metadata": {},
 "outputs": [
  {
   "data": {
    "text/plain": [
     "<AxesSubplot:title={'center':'Accuracy Comparison'}, xlabel='Accuracy', ylabel='Algorithm'>"
    ]
   },
   "execution_count": 41,
   "metadata": {},
   "output_type": "execute_result"
  },
  {
   "data": {
    "image/png":
"iVBORw0KGgoAAAANSUhEUgAAA70AAAHNCAYAAADSYBxEAAAAOXRFWHRTb2Z0d2FyZQBNYXRwbG90
bGliIHZlcnNpb24zLjMuMiwgaHR0cHM6Ly9tYXRwbG90bGliLm9yZy8vihELAAAACXBIWXMAAA9hAAAPYQ
GoP6dpAAA7m0lEQVR4nO3de5x2c73/8debW7bTfddudyBJRD/awo6Oig5KJemgwm67kajdUaWtM7UL7V
RUuxIpJaWUEtkh7GRXklNIyCGncrrH8ebO5/fHWlOXy8zcM3Nfc8/M8no+Husxc631XWt91jXLbd7z/a7vlap
CkiRJkqQuWma6C5AkSZIkaaoYeiVJkiRJnWXolSRJkiR1lqFXkiRJktRZhl5JiRJUmcZeiVJkiRJnWXolSRJkiR1lqF
XkiRJktRZhl5JiRJUmcZeiVJmiJJ3pakklww3bXMRkkelWS/JOcnuT3JOcnuT3J3Un+kOSzSdaZ7vqmWpL57f2z5nTXIk
mzWapqumuQJKmTkpwDbNi+fHpV/Xlay5PxqhJKm/JDhVI4DNi8fVEuBqGpn2ROxE4gdq8wTDtZJfZbv6wPqBT3EhcW
TXY8kzVaGXkmSpkCSTYBfAz8GXgocWlV7T0NWYCVj0Wdq8O88Kmhpm61eVldqoU18UwwF0t1+T9ADVlUB+9M/3Gml1Ux9pWwA
3F3+kiZJA+HwZkmSpsau7df/AH4BvC7Jiv2NkjwmyZeSXHtkMqUwsbeQcZi8BxZcCvNp9D62RnirHU0mSKQkyK+c1zn
7Rd+x12zXz+9Zd3g7VHiDJP+T5Dbg5HbbkmOTfKndijxpUm+OSfRqj7/yX5Vpb2pquSvL1JMu3512UZO8R9
ntOW9N2Y7x3uwGPBvYaKw0C9AfeJNskOTPJPnUluS/LTJM/oa/NTUs+/+tzdbCefZ4hHWSZGZsTsqRAvRtbE6sCAKT1f
DkE0n+2N5T1yT5fJKH9p37ivZ42zHj1eS4xL6+IyfjUiiAkAGiAx/3OWRcOcyRK2tGTj6XUaldpxkqvUKA+YHwXuRiAAFFvRwL9mbxlIAo
WuVhTkd1vFvQgMb4p7SLN5ax1KRv7p4SZWNw6IGY2SIU4+EnYfUZMA/ECQbd+7/+7ytU3B9FAHA5xbC43jutOocB2v6mLzv/xWaMY
AgcCLgXcAC4CHtW1WAX4O7A58FXbZsAdwCbDqJEt8CPdqMU3PBDw8hJhCbiElaYYgMUBqGxbn86ND1JVYl8/8AGD1Fe6hfffX
8AGHRh2Y4UjEKiBdGTGhEjo6sygu2LbzEuvLObp0fPaGVLBP3Fmn8HmfULE5hNz6F2KSzhvqdKmBnWGHoe
GCBiKSxkOlMDQDd1PqfkK2JUvcDVbFQI1lkHeZgU8ARXAhJmw+gs+H/kRXk6C1oJDXYTUOWfo2rxzVI/6wrAKjoYH8bJi4xEkabqHG
ia02l5/Xb0Y4oXfE1MbYVtKMnxbeBZx3eR1o1OJ8J1jwuSWHfRO4TcXNmbGxWkOz7Nzl+W4qlD16FCZZ0xtYRRTFuwjvWbsdFGpFe
hVB9SPiv7oC0/5xJD5iyjnQamAC+d9X6MR6fG3SDPpG8UESSTx9o+4NCG6jXr2cHWyo4csxNdu+AqI+5YG1jxFSOgUvPcIXbP3tDeh21A2XT
IULfRrn4a8KSzhCOQ4jIcYMNlUD6Q9EN7psXsVxaysvMGVpkM2rh2p4a6Psu9pP/p63zoeEvHX+hVBckS+7k3Df8J/fmW3WZ1kLe+o
WC74/wPIg1wP7K4xMYqLbpn1XTcCsYp6dKr8Hqy+Z0Qm9K3+xm/eY3QHmBzgd1lfy1YMndM7FPXM9/yrDHiaRk5o2Mg/6l6RiUfZg0YjLV4
0AAvmIjAz4UDhGpcqxRNpTo18vYp6/vTfFqsAPlqysxEVHEyu5q/0oCX2NDnL0tHtTwlHG7L8m8tZSwDzxrlRIpWZHJyJjejPFY0U40
7n5CqhPHXNt87M70S/HLJH4Pl3WC5wQfq4m62dQhn95SmWvbVEgPx2j/hC+vq9NZEMvGNSdsaqZzvDGh6s8d0TaN4yrtQBhSKFJwG
YVeNVrW5J36C3QprSzSBjXptTJMjrMjGu6Lm7Wa+ejqQcO8AP4cBwjkb8yP3pa5LT2Obel4V21y96QjKhd5+1AP6ti/SqHZM7ETuQ2CjqUq6kBsnG1U
```

6T676eqFtD8weEcmiHWv2uf6d2nZ/j5w4FF/X+QaEN6f43Q9Or2WzhKjZI0Kxl6JUkarF1ohga/mqZXbnj5cbt
9p57nXP8CrP6AI9zfeNoMh7X+5zVHenYUml7Nfv9M83zne6rq4Ko6tap+zQND0c00z9suriZogt9NNL2929F
MTvX5cex3Ik2wftk42g7XN9LzzasB99G8/4P06FHW3QSQZB6wNXBAVe1XVSe37+X5wD+OcsxxzdRcVedX1
etowutGwLdpnqt+V9vkJmBOmo87+pv2Wd9HAzeO5zyS1CWGXkmSBqQNszsBl9EMde1fPkUTzl7c7nIC8N
wkTxzjsCcA6yZ53hhtrmi/Prlv/TYTKH84dPV/Huzu92tUdRdwGrDdKBMy9ba9m2Yo807AnjRDac8YRy2H0vR
KHtBO9vUAPRNM/Z7mmd4d2mA3vH0lmomqzqzBfxzT9n3nehzwTODUdlXR/OGj/718A02YX2LVOLeq3gn
cCvxLu+nk9uu/9u3yKpqh1ScjSQ8yc6a7AEmSOuTFNL2L722fab2fJBfQDHndlea50uGZj09P8nGansCHAlsBB
1bVxcBngNcCxybZD/gVzdDTzYHjqupnVXV9kpOAvZPcQvMc7vNphiqP18U0YX2/NtDdTNPTuuUIbfekmVH6
l21Nl9LMKLwNzeRdt/W0/QKwF80kTG8YTyFVtSDJy2neo98m+RzNrNL3AOvQBLoNgWOq6r72o4W+CRyX5
Es0Pd7voXkv/2Pc78D4PRL4fpJDgHnAPjS97Z9o6x9KcjrwniQ30vxRYnOan/utkz1pkq2BN9PMznw5TbB+Jc1
1/rRt9lOanvL903ze8Rk0fwzZh2bCriMme35Jmq3s6ZUkaXB2pQlmXx1pY1XdSPNRPVsneVRVXUPzrOdxNO
HsJ8DBNEHq5naf24DNaHo/30gzTPoQmgmcru05/OtpevH2p5m06DE0H98zLlV1L03IvQT4EvAtmnD3ghHa
ntvW/RuaoPeT9rwL2+vvbXsNTUC+mWa483jr+RWwAc1H+byGJuidSPPRPxcDz+5peySwLc2Q32/TvP9DwH
Or6ufjPecEvI/mDwtfbeu7rj3XZT1tdgB+RvN89DHAJjR/QFiwBOf9A01o3ovmI6GOpunhnV9Vh8Dfnt3dluZji
nYGjufvH1/0vKrq732WpM5Lz+SDkiRJA5XkkTQB8eCq2mu661kSSbagCbLbVdV3x24tSZopHN4sSZIGLsnqw
Fo0w4zvAz47vRVJkh6sHN4sSZKmwhtoJnZ6ErBjO8xZkqSlzuHNkiRJkqTOsqdXkiRJktRZhl5JkiRJUmcZeiVJkiR
JneXszZo1kgRYDbhtumuRJEmSNO1WAa6txUxUZejVbLIa8KfpLkKSJEnSjLE6MOYnBBh6NZvcBnD11Vczd+7
c6a5FkiRJ0jQZGhrisY99LIxjFKihV7PO3LlzDb2SJEmSxsWJrCRJkiRJnWXolSRJkiR1lqFXkiRJktRZhl5JkiRJUmc5
kZVmnTU2fi9ZdvnpLkOSJEkddcsln5nuEjRA9vRKkiRJkjrL0CtJkiRJ6ixDryRJkiSpswy9kiRJkqTOMvRKkiRJkjrL0
CtJkiRJ6ixDryRJkiSpswy9kiRJkqTOMvRKkiRJkjrL0CtJkiRJ6ixDryRJkiSpswy9kiRJkqTOMvRKkiRJkjrL0CtJkiRJ6i
xDryRJkiSpswy9kiRJkqTOMvRKkiRJkjrL0CtJkiRJ6ixD7wyR5Iok7xh0W0mSJEl6MDP0jiHJ4UmqXe5NckOSny
bZJcmg37tNgS9PQdsJ67vuEZepOrckSZIkDZKhd/F+AqwKrAm8GPgZ8FnguCRzBnWSqvpLVd056LaT9Haaax
5eAHYeYR0ASR4yhbVIkiRJ0qQZehdvYVVdX1XXVNXZVfVx4OU0AXj+cKMk85J8OcmfkwwlOSXJhr0HSrJNkr
OS3J3kxiTH9Gy735DlJB9JclWShUmuTXLQGG3XSHJsktvbc38nyaP6jnVOkte3+y5IclSSVUa64Kpa0F7z9VV1f
bv61p7XRyX5XJIDk9wI/LQ9z/pJjm/ruCHJEUn+qaeOJNkryeVJ7kpybpJXT+inIUmSJEkTYOidhKo6BTgXeCU0
YQ74MfBo4CXAU4CzgZOT/GPb5qXAMW27jYHnA2eNdPw2CL4T2B1YB9gWOH+UtgF+APwjsDmwJbA28O
2+pmu3x9m6XTYH/mNCF35/OwGLgGcBuydZFTgNOAfYBNgKeBTwnZ59PkbTY/wm4EnAp4FvJNl8CeqQJE
mSpFENbHjug9DFwJPb758LbAA8sqoWtuvenWRb4NU0z9++Hziqqj7cc4xzRzn2GsD1wElVdS9wFfCrUdq+o
K3j8VV1NUCS1wO/S7JpVf26bbcMML+qbmvbHEETvN8//ku+n0uraq/hF0n2Bc6uqvf1rNsFuDrJusA1wJ7A8
6rqzLbJ5Uk2own3p/WfIMnywPI9q0bsmZYkSZKk0djTO3kBhid0egqwMnBTO7T39iS3A4+n6WEF2Ag4eZzH
PhpYgSYUHpLkFWM8P7wecPVw4AWoqguBW9ttw64Dryt64BHjrOekfT3Uj8FeG7f9V/cblsbWB/4B+Cnf
W3+jb+/R/32Bhb0LH9agnolSZIkPQjZ0zt56wF/bL9fhiZEbjFCu1vbr3eN98BVdXWSJ9IMVX4B8AXgPUk2b3t
+e/WG77HW9+9XLNkfPe7oe70M8CPgvSO0vQ745/b7l9L0+vZayMg+ARzY83oVDL6SJEmSJsDQOwlJnczn
PnT7aqzaZ7nXVRVV4yy23k0w4m/Op5zVNVdwA+BHyb5PE2v6QbtuXpdCKyR5LE9w5vXB+YBF433mgbgb
OBVND3Ki/o3JrmQJtyuUVUPGMo8knao+N8CcfP4siRJkiSNn6F38ZZP8mhgWZqjmbaiGXZ7HPD1ts1JwJnA
D5K8F/g9sBrNpFY/qKqzgH1oJra6DDiK5r1/cVUd0H/CJPPb8/0SuBN4PU1P8ZUj1HcSTaD+Zjuj8xyanuHT2v
MuLZ8HdgO+leSTwI3AE4DXAbtV1W1J/gv4dPsZxz8H5gLPBG6vqq8txVolSZIkPUj4TO/ibUUzPPcKms/sfS7w
NuDlVfVXgKoqmoB7OnAYcAlNsF0TuKFtcyqwHbANzQzHpwBPG+Wct9IEyDP4ew/xy6rqpv6G7bm3BW5pz
38ScDnw2sle8GRU1bU0MzkvC5wIXEDzecYLgPvaZh8E9qX5o8FFbbuX8fdh4pIkSZI0UGkykzTzJZkLLJi31h5k
2eUX216SJEmajFsu+cx0l6DFGBoaYt68eQDzqmporLb29EqSJEmSOsvQK0mSJEnqLEOvJEmSJKmzDL2SJEmS
pM4y9EqSJEmSOsvQK0mSJEnqLEOvJEmSJKmzDL2SJEmSpM4y9EqSJEmSOsvQK0mSJEnqLEOvJEmSJKmzD
L2SJEmSpM4y9EqSJEmSOsvQK0mSJEnqLEOvJEmSJKmzDL2SJEmSpM4y9EqSJEmSOsvQK0mSJEnqLEOvJE
mSJKmz5kx3AdJEXfXb/Zk7d+50lyFJkiRpFrCnV5IkSZLUWYZeSZIkSVJnGXolSZIkSZ1l6JUkSZIkdZahV5IkSZLU
WYZeSZIkSVJnGXolSZIkSZ1l6JUkSZIkdZahV5IkSZLUWYZeSZIkSVJnGXolSZIkSZ01Z7oLkCbqiFc9mRXm+PcaS

ZIkaWnY5YTLp7uEJWJykCRJkiR1lqFXkiRJktRZhl5JkiRJUmcZeiVJkiRJnWXolSRJkiR1lqFXkiRJktRZhl5JkiRJUm
cZeiVJkiRJnWXolSRJkiR1lqFXkiRJktRZhl5JkiRJUmcZeiVJkiRJnWXolSRJkiR1lqFXkiRJktRZhl5JkiRJUmcZeiVJki
RJnWXolSRJkiR1lqFXkiRJktRZhl5JkiRJUmcZemeAJKcm+cx01yFJkiRJXWPonaQkhyepJP/Rt37bJDXBw70S+O
DgqnugnnqHl5uS/CTJk6fyvJIkSZI0nQy9S+Zu4L1JHrYkB6mqm6vqtgHVNJafAKu2y/OBRcBxS+G8kiRJkjQtDL
1L5iTgemDv0RokeXiSbyX5U5I7k5yfZPu+Nn8b3pzkE0n+b4TjnJdkn57XOye5KMndSS5O8uZx1Luwqq5vl3O
A/YHHJnlEz3H3T3JJW+vlST6aZLl225pJ7kuySV9tb01yZZK0r9dPcnyS25PckOSIJP/U0/7V7ftwV9vjfFKSlcZRvy
RJkiRNiKF3yfwVeB/w1iSrj9LmH4DfAFsD/wx8GTgiydNGaf9N4GlJ1h5ekeRJwAbtNpLsBvwn8H5gvbaGjybZ
abyFJ1kZ2BG4FLipZ9NtwHxgfeDtwG7AOwGq6gqaoL9z3+F2Bg6vqkqyKnAacA6wCbAV8CjgO+15VwW+BR
zW1r4FcAyQ8dYuSZIkSeM1Z7oLmO2q6vtJzgH2AXYdYfs1wH/1rDo4yVbAdsAvR2h/QZLzgB2Aj7ardwR+XV
WXtK8/CLyrqo5pX/8xyfrA7sDXxih36yS3t9+vBFwHbF1V9/Wc/2M97a9I8ingtcAB7bqvAF9MsmdVLUyyIbA
RzXPJAG8Czq6q9w0fJMkuwNVJ1gVWprnvjqmqK9sm549UbJLlgeV7Vq0yxrVJkiRJ0gPY0zsY7wV2aoPn/SRZ
Nsn72+HJN7Wh84XAGmMc75s0QZd2yPD2/L2X9xHAY4FD2+HDt7fH/ACw9ijHG/YzmoC6EfA04H+AE5I8rq
feVyf5eZLr2+N+tK/WH9A8C/yK9vUuwM/aXmCApwDb9vUuv4nbbb2sC5wMnA+UmOTrLbGM9E7w0s6Fn+tJj
rkyRJkqT7MfQOQFWdDpwIfHyEze+iaIfHyEze+iGR58APA8msB5IvCQMQ55JLBukn8BnkkTco9qtw3/zHbj7wF2I5qh00
9fTKl3VNWl7fIrmp7pldpjkeTp7XlOoBmOvTHNMOq/1VpV9wBHADsneQhNj/RhPedYBvhRX20bAesAp1fV
X4EtgRcDFwJvBX6f5PEj1PsJYF7PMtoQckmSJEkakcObOB2dv4LfAJX3rnw0cW1XfAEiYDE0AvGi0A1XVn5KcTt
PbuwJwUlXd3G3/Ick1wAK3ZEHwsnmJjMk4RvGO3O
nrSrgDOCMJPsCV9L0HB/Y124hsLCnlolcmyRJkiQZegelqs5L8k2anstelwC/r
Ovm0fAQ5KMkTTK7s8zaRRD6uqAxnd8kke3X7/MOAtNM/Y/qin1jWSvA74NfBS/j6MufdaL2pnmN4fOKyq7
urZ/HmanuNvJfkkcCPwBOB17fpNaD4u6X+AP9MMs34Ei38/JEmSJGnCHN48WB/kgbMQf5Sm9/NE4FSajz
6wTiOdTTwcGDF/vZV9RXgDTSzLJ9PM1vyfOCPiznmVjSTV11HM4nWpsB2VXVqe9xjgU8Dn6PpqX1gOzZ/H0yrX
6H0gTy3qHNVNW1ND3Gy9Jc8wXAZ2meyb0PGAKeAxxP0yv+MZpJuU5YTO2SJEmSNGFpRppKE5Pk/cDrq
mqDpXjOucCCz73gcawwx7/XSJIkSUvDLidcPt0lPMDQ0BDz5s0DmFdVQ2O1NTloQpKsnGRTmmHcB013PZI
kSZI0FkOvJupzwM9phnEmSJElSZxl6JUmSJEmdZeiVJEmSJH
WWoVeSJEmS1FmGXkmSJElSZxl6JUmSJEmdZei
VJEmSJHWWoVeSJEmS1FmGXkmSJElSZxl6JUmSJEmdZeiVJEmSJHWWoVeSJEmS1FmpqumuQRqXJHOBBQ
sWLGDu3LnTXY4kSZKkaTI0NMS8efMA5lXV0Fht7emVJEmSJHWWoVeSJEmS1FmGXkmSJElSZxl6JUmSJEm
dZeiVJEmSJHWWoVeSJEmS1FmGXkmSJElSZxl6JUmSJEmdZeiVJEmSJHWWoVeSJEmS1FmGXkmSJElSZ82Z
7gKkiXrS2zZmmYcsO91lSJIkSbPelV++ZLpLmHL29EqSJEmSOsvQK0mSJEnqLEOvJEmSJKmzDL2SJEmSpM4y9
EqSJEmSOsvQK0mSJEnqLEOvJEmSJKmzDL2SJEmSpM4y9EqSJEmSOsvQK0mSJEnqLEOvJEmSJKmzDL2SJEm
SpM4y9EqSJEmSOsvQK0mSJEnqLEOvJEmSJKmzDL2SJEmSpM4y9EqSJEmSOsvQK0mSJEnqLEOvJEmSJKmz
DL2SJEmSpM4y9GpESR6Z5EtJrkqyMMn1SU5MsnmySG5N8YJT99m63PyTJ/CSV5KIR2r2m3XbFlF+MJEmSp
ActQ69G8z1gQ2AnYF1gG+BUYGXgG8D8JBlhv52BI6rqnvb1HcAjkzyjr90wuFVTULckSZIk/c2c6S5AM0+Shw
KbAVtU1Wnt6iuBX7XbrwB2B9hgCg9s223OrAF8Glg+6m7EkmSJEkPdvb0aiS3t
8u2SZbv31hV5wO/punV7bUL8KuquqBv/aHAa5Os2L6eD/wEuGGQRUuSJElSP0OvHqCqFtEE052AW5Ocke
TjSZ7c0+ww4NVJgZov27H/Xt5h493DnBZ2z7tsQ9bXB1Jlk8yd3gBVlmiC5MkSZL0oDPh0Jvk4Uk+n+TCdsKi
m3uXqShSS19VfQ9YjeZZ3hNphiOfnWR+2+Q44NVRUCHDXKIQ+j6RnenOa54OPHUcbewIKe5U8TvA
xJkiRJD3KTeab3G8DaND7ND16NwA10Io0Y1TV3cBP22XfCb9gEOr6oFSb5LE2QPbb9+t6qGRjncN4EDgI8AX6+
qRSPPg3U/nwAO7Hm9CgZfSZIkSRMwmdC7KfC8qjp30MVoRjuf5jlfaIY4n5zkpcCzgfeNtlNV/TnJsbSzLI/nRFV1F3DX8OtxhGRJkiRJ/g3U/nwAO7Hm9CgZfSZIkSRMwmdC7GbBZVZ076GI0410IbNvz+lDg1CRbA88jV1c5IfAq8B9hjP
yapqIbBw+PU4QrIkSZIk3c9knum9GFhh0IVo5miHsJ+S5F+TPDnJ45Ng68Clg8ClVXX6Yg49H
/inqrp4qmqXJEmSpF6T6el9M7Bfkn2BC4B7ezeOM0bxVs8ftwC+Bd9IMZV8OuBo4BPh4X9vD2nWfXNxBq+o
u4K6BVipJkiRJY0jVxB7JTbIOzSRGG/dvAqqqlh1QbdL9tDM4L1h9p7VY5iHeZpIkSdKSuvLLl0x3CZMyNDTEvH
nzAOYtruN1ZbfAiawkSZIkSTPYZELvMwP4PwSJkiRJGrTJ9P
QeDHw2ySeB83ngRFbnDaIwSZIkSZKW1GRC77fbr4f1rCvaiawAZxiSJEmSJM0Ikwm9jx94FZIkSZIkTYEJh96q

unIqCpEkSZIkadAm09NLknWBLYBH0jcZVlXtu+RlSZIkSZK05CYcepPsBvw3cCNwPff/nN4CDL2SJEmSpBlhMj
29HwDeX1X7D7oYSZIkSZIGaTKf0/sw4OhBFyJJkiRJ0qBNJvQeDbxw0IVIkiRJkjRo4xrenORtPS8vBT6a5OnA+
cC9vW2r6qDBlSdJkiRJ0uSN95ned/a9vh3YvF16FWDolSRJkiTNCOMKvVX1+KkuRJIkSZKkQZvwM71JPpRkxR
HWr5DkQ4MpS5IkSZKkJTeZiaw+DKw8wvoV222SJEmSJM0Ikwm9oXl2t9+GwM1LVo4kSZIkSYMz3omsSHIL
Tdgt4JIkvcF3WZre3y8OtjxJkiRJkiYvVSN12o7QMNmJppf3MOAdwIKezfcAV1TVmYMuUBqWZC6wYMGCBc
ydO3e6y5EkSZI0TYaGhpg3bx7AvKoaGqvtuHt6q+prAEn+CPyiqu5dzC6SJEmSJE2rcYXeJHN70vNvgRWSrDB
S28WlbEmSJEmSlpbx9vTekmTVqvozcCsjT2Q1PMHVsgOqTZIkSZKkJTLe0Ps8/j4z83OnqBZJkiRJkgZqXKG3q
k4DSDIH2AI4rKqunsK6JEmSJElaYhP6nN6qWgS8G4cwS5IkSZJmgQmF3tbJNL29kiRJkiTNaOP+yKIeJwCfSPL
PwG+AO3o3VtUPB1GYJEmSJElLajKh97/br3VaEBVpIkSZLUWZM
Z3kySzWlmcV6PZkjzRcAnq+p/B1ibNKKPPuFxLL9MprsMSZIkaan72PU3T3cJs86Ee3qT/CtwEnAncBDwOeA
u4OQkOwy2PEmSJEmSJm8yPb3vB/aqqk/3rPtskj2BDwJHDqQySZIkSZKW0GSe6V0L+NEI638IPH7JypEkSZIk
aXAmE3qvBp4/wvrnt9skSZIkSZoRJjO8+VPAQUk2BN35An5HVZsB84O0Dq0ySJEmSpCU0mc/p/e8k1wPvAl7
Trr4IeG1VHTvI4iRJkiRJWhKT+siiqvo+8P0B1yJJkiRJ0kBN5pleSZIkSZJmhQn39h9Ca5heY53n4F3A1cChxeVV9d
wtokSZIkSVoikxnevC/NZ/WeAPwKCLApsBXweZqPLfrvJHOq6pBBFSpJkiRJ0kRNJvRuBnygqr7YuzLJ7sALq+p
VSc4D3gYYeiVJkiRJ02Yyz/S+CDhphPUnt9sAjgfWm6ZNwmRC783Ay0ZY/7J2G8BKwG2TLUqSJEm
SpEGYzPDmj9I8s5sHODmj9I8s/tcmmd6C3gq8BJgj7bNlsBpA6lQKiRJkqRJmnDorapDklwIvB/Yvan1fVL9o2nxpolZ
IkSZIkTcJkenqpqjOAMwZciyRJkiRJAzWu0Jtk7ngPWVa9Dky9HkiRJkqTBGW9P7600z+6OJW2bOUtUkSZIEmSJEm
SBmW8ofe542y38WQLkSRJkiRp0MYVeqvqlNG2JZkH7A28GdgU+OJAKpMkSSJIiiuAz1TVZya5//x//x2/4cOrqpuSHI
qcE5VvWOaS5EkSZKkEU3mc3oBSPK8JN8FrqXpiTseeOqgCpMkSSJIiSVDmc3oBSPK8JN8ArgPeChwPbDLBYxye5AeTrWGcNgW+PJ6GSa5I8o6+1d8G1p3sy
ZPMT1I9yw1JfpTkSZM95gzySuCD012EJEmSJI1mQrM3J1kdmA/sAqwPfAFEfAdYDnhVVVV048OoGoKr+soT73wX
ctYRlDAFPpHnu+THAAcCPk6xbVfcs4bFHlWS5SZ5qrp3qo5fP1bTdP1bElSZIkaRDG3dOb5cP3N
VWHvOzZP8KsnCJNcl2S/JnJ7tP+w55+2/vf/722SjyS5qj3T5ZqqS5qj3mtUkOatefCjwO+PRwr2/w/e2+x+T7DDCtVWSPZI
cm+QO4APt+pcNL7kly7JEkMb34h8BXgw1X146ra66r66xCQOBZwHbAFs2wHbAFs
Czwb+ZYxjvhp4J7A7sA6wLXB+u/mVwZJ7sA6wLXB+u/mVwJ+ADwGrtstIx3gpcAzwYY5qJu54PnDWB63oosEP78t523QbAie1xnwy
8FtgM+FzPrl8HVgO2AF4FvBF45Ain2Ac4FtgAOCzJi4BvBvAAfR/MFid5re+ve35x71PUmySbvfh2gC+lbA6WNc3
uE0Q9y3AZ5B07N9fJLletqsCLwbeD3wHGAN4L9GOliS5ZPMHV6AVcY4tyRJkiQ9wESGNz+bZzc4tyRJkiQ9wESGNz+bZljzWUkuBo6ge
d51qrwZuBp4S1UVHGS1YD9K+xLM7x6J2CHqjozOwLVjHHMN4X9GM
d4P3BUVX24Z925i7mWeUlupwmBK7brflfVF3ffVc4smeyrT8keRtwWpI3AWsCLwA2bXuKSfIG4A8jnOvIqj
ps+EWSI4D9qupr7arLk3yQbH7gCOq6rbgCuB3450gW2P7jbAs6rqF+26HhtsCR7dNlwP2qKr
L2jafownVI9kb+PAo2yRJkiRpscbd01tVuwY2SCiw8Xgdc1UBEWBlYH1qIJUMMBja
paAPx+jGMeDaxAE/wOSfKK3mG+47QRcPIE97mt3e8pwB7AZe3XYU8B5rfDfW9vA/KJNO/t42I6WRcBZw/v
UFWXAreMcK7+XuenAB/qYhwKpJVmTs9+SnNEH38iRHJNmx3Wck67U1/rKnxptofh7r9bS7czjwtq5j5B5
rgE8A83qW1UdpJ0mSJEkjmvDszVV1Z1UVlWb0Qyh/RTwH8Cfk/xwgLUFqBHW0a7v/X6X6MMh/
p5mc6gvA6X3DbxdnMpNa3VdVl1bVxVX1JR7YS74MzR8SNupZZZnwZo1/TSOvv6Hu9DE1vae+xN2iPff
dY70nbu/svwPY04XRf4Nx2iPZ4ahle3/sz6p9Yq/dnef8NVQuramh4ofnjgSRjiSN26Q/sgigqn5fVXvR9MBtP5
iS/uZC4JlJegPRpRSAjRDu8qBnPn5s9Zhu2FJnlRHvnPtdhDFV1V1X9sKreDPPcCyi6nrPJrneJfEp4ENk7yif
X028KQ2GPcv9wAX0wxF33j4AEmeADx0HOc6G3jiKMe+D8Z+T6pqUVWd1P6cn0wz1Pp5I5znwbGp/XU+XU+
HCaj3u6aJzviyRJkiQN1ESH9o6ondTqqyFKhdb+7wvMwuiN4Tnsyt5v+DSnRbWnRGL5unHp+gEDu+1xEsPaX+E
+Os1Q6pfQ/KHkAUPIq+oPSY4FDkmy32dtpvqPHK5P8/b8dtE2OpJ2K/Bv08uN9nE0sSbBlx5eZfDH
vt9PLD3F2hmD6YJtb8E7qSZTOkumaCCnuQw4iuY9fHvVTeC66qoSRfAfZJ8
znF+wP/l+TzNEOP76AZErxlVb21qi5OchLwSkntJ1D+DW3bSlCahLAZs4iRpOUSMGgKgpW7OL3oG+9fUGoveSakdjJ/TjxcFNUkQTPcqZj+PhX2iGoYuW
+Os1Q6pfQ/KHkAUPIq+oPSY4FDkmyO80fJ/aj+QPFseN9byRJkiRpkJaop3dAtqCZHKl32beqUJWU+lmSjqi
8ChwMd69t0TOBM4DjiJ5pnfi4C7RznXrcBubbvhtuXtc+eQHsWfSjmwQV

Gdpgu12VXUesDlNL/X/0rwHH6UZUjzs34AbaALo92nC8W2Mfq3D9Z4IbE0zu/Wvgf+jed+Gg/6tjP6e3Eozq/
UpNO/rHsD2VfW7UU63M/Abmp/HmTTDll8ylZ8VLEmSJEljyf3niZrd2s+1vQZ4V1UdOt31TKUkq9PMjPyC4d
mru64dvr7g3Y94KMsvM+qj25IkSVJnfez6m6e7hBlhaGiIefPmAcxr5/8Z1UCGN0+XJBsD/49mBud5/P2jbzo3
nDbJ82hmrj6fZgbtA2iGY4/1ubmSJEmS9KA2q0Nv6900z/veQzO09tmjPIs72y0HfJzmGdvbgF8AOzp0WJIkSZ
JGN6tDb1X9luZzaDuvfTb3xOmuQ5IkSZJmk5kwkZUkSZIkSVPC0CtJkiRJ6ixDryRJkiSpswy9kiRJkqTOMvRKkiR
JkjrL0CtJkiRJ6ixDryRJkiSpswy9kiRJkqTOMvRKkiRJkjrL0CtJkiRJ6ixDryRJkiSpswy9kiRJkqTOMvRKkiRJkjrL0CtJ
kiRJ6ixDryRJkiSpswy9kiRJkqTOMjPdBUgT9cFLr2Tu3LnTXYYkSZKkWcCeXkmSJElSZxl6JUmSJEmdZeiVJEmSJ
HWWoVeSJEmS1FmGXkmSJElSZxl6JUmSJEmdZeiVJEmSJHWWoVeSJEmS1FmGXkmSJElSZxl6JUmSJEmdZe
iVJEmSJHXWnOkuQJqonTZ5B8st+5DpLkOSJEl6UPjORV+c7hKWiD29kiRJkqTOMvRKkiRJkjrL0CtJkiRJ6ixDry
RJkiSpswy9kiRJkqTOMvRKkiRJkjrL0CtJkiRJ6ixDryRJkiSpswy9kiRJkqTOMvRKkiRJkjrL0CtJkiRJ6ixDryRJkiSps
wy9kiRJkqTOMvRKkiRJkjrL0CtJkiRJ6ixDryRJkiSpswy9kiRJkqTOMvRKkiRJkjrL0CtJkiRJ6ixDryRJkiSpswy9Gog
khyepdlmU5Kok/53kYT1truhpM7z8aTrrliRJktRtc6a7AHXKT4Cdae6r9YHDgIcC2/e0+RBwSM/rvy6t4iRJkiQ
9+Bh6NUgLq+r69vs/Jfk2ML+vzW09bSRJkiRpShl6NSWSrAVsBdy7BMdYHli+Z9UqS1qXJEmSpAcXn+nVIG2d
5PYkdwGX0Qxx3r+vzf5tm+HlbWMcb29gQc/i87+SJEmSJsSeXg3Sz4A3ASsCbwDWBQ7ua/NJ4PCe1zeOcbx
PAAf2vF4Fg68kSZKkCTD0apDuqKpL2+/fluRnwIeBD/a0ubGnzZiqaiGwcPh1koEVKkmSJOnBweHNmkr7AO9
Ostp0FyJJkiTpwcnQqylTVacCvwPeN82lSJIkSXqQMvRqqh0I7JbksdNdiCRJkqQHH5/p1UBU1fxR1h8JHNm+X
HNp1SNJkiRJYE+vJEmSJKnDDL2SJEmSpM4y9EqSJEmSOsvQK0mSJEnqLEOvJEmSJKmzDL2SJEmSpM4y9EqS
JEmSOsvQK0mSJEnqLEOvJEmSJKmzDL2SJEmSpM4y9EqSJEmSOsvQK0mSJEnqLEOvJEmSJKmzDL2SJEmSp
M4y9EqSJEmSOsvQK0mSJEnqLEOvJEmSJKmzDL2SJEmSpM4y9EqSJEmSOitVNd01SOOSZC6wYMGCBcydO
3e6y5EkSZI0TYaGhpg3bx7AvKoaGqutPb2SJEmSpM4y9EqSJEmSOsvQK0mSJEnqLEOvJEmSJKmzDL2SJEmS
pM4y9EqSJEmSOsvQK0mSJEnqLEOvJEmSJKmzDL2SJEmSpM4y9EqSJEmSOsvQK0mSJEnqrDnTXYA0UTtstj
bLLevfayRJkqSl5fu/vWG6S5g0k4MkSZIkqbMMvZIkSZKkzjL0SpIkSZI6y9ArSZIkSeosQ68kSZIkqbMMvZIkSZK
kzjL0SpIkSZI6y9ArSZIkSeosQ68kSZIkqbMMvZIkSZKkzjL0SpIkSZI6y9ArSZIkSeosQ68kSZIkqbMMvZIkSZKkzjL
0SpIkSZI6y9ArSZIkSeosQ68kSZIkqbMMvZIkSZKkzjL0SpIkSZI6y9ArSZIkSeosQ+8Ml2TZJL9I8r2+9fOSXJ3kYz3
rXpXklCS3JLkzye+THJZk454285NUz3J7kt8keeVSvq5Tk3xmaZ5TkiRJ0oOPoXeGq6q/AjsBWyXZsWfTwcDN
wL4ASfYHvg2cA2wDPAl4I3AZ8PG+ww4Bq7bLxsCJwHeSPHHKLkSSJEmSpoGhdxaoqj8AewMHJ1ktycuB1w
E7VdU9SZ4O7AXsWVV7VtX/VtUfq+q0qvpP4CUPPGRd3y5/AD4A3Ac8ebhBkocl+XpPr/EJSdbpPUjbs/y7JA
uTXJHkXX3b35zkD0nuTnJDku+26w8HNgfe3tPjvOYA3zJJkiRJAmDOdBegcTsYeAXwdWADYN+qOqfdtj1wO/
CFkXasqhrtoEmWBf6tfXl2z6bDgXVoeo2HgP2B45OsX1X3JnkK8B3gIzQ9zM8EvpDkpqo6PMkmwEHA64FfA
P8IPLs99tuBdYELgA+16/4yQm3LA8v3rFpltOuQJEmSpJEYemeJqqokbwIuAs4H9uvZvC5weVUtGl6RZE/aoc
+tx1TVgvb7eUlub79fAbgXeGNVXdbuOxx2n1VVv2jX7QhcDWwLHA3sCZxcVR9tj3NJkvWB99AE5jWAO4Djq
uo24Ergt+21LEhyD3BnVV0/xmXvDXx4HG+PJEmSJI3I4c2zyy7AncDcX7tvX35h4GbATsDqwEpGfbbe22jW
ie6X0f8KUkL2u3rwcsAn75t4NX3QT8vt023OaMvnOeAazT9h7/lCboXp7kiCQ7Jllx/JcKwCeAeT1L/zVLkiRJ0p
gMvbNEkmcA7wReDpwJHJpkOMj+AVg7yXLD7avq1qq6FLhmhMPdV1WXtst5VXUg8DPgvcOnG60M/h6ue
7/v3T58/tuAf6EZen0dTa/zuUkeOp7rbY+xsKqGhheasC5JkiRJ42bonQWSrAB8DfhSVZ0EvAHYlKYXF+BbwF
rAm5fgNH+lGeoMcCHN0Pen9dTwcJph1Bf1tNms7xjPBC5pZ5ymqhZV1UlVtRfNJFlrAs9r294DLLLsE9UqSJEn
SYvlM7+ywH80fKN4LUFVXtTMlH5jkJ1V1ZpJPAZ9K8jjgGJrnb1cFdqXpkb2v53hJ8uj2+xWALYEX0T4DXFV/S
HIscEiS3Wl6WPej6TU+tt3vU8Cvk3yQZivk3yQZwKrZwBvoQ3eSbYG1gJOB26hmUF6GZoh0gBXAE9rrzZ22+Hbi5qnprlC
RJkqQlZk/vDJdkc+DfgflVdcfw+qo6Ta6EOTpKreDexA84zucTRDno+m+Rk/ox0ePGWwuzZDj62h6bt9FM4v
yf/a02Rn4TXusM2mGLr+kqu5tz3828Bqaj066gCYwf6iqDm/3vxNxV4KXBKe449gO2r6nft9v+i6V2+kGbm5jUm
+x5JkiRJ0mgyxqfZSDNKkrnAgr5Agpdu8E8st6x/r5EkSZKWlu//9obpLuF+hnJ5VyEK5tHWqLolTa6bFxFdoo
SZIkSeosQ68kSZIkqbMMvZIkSZKkzjL0SpIkSZI6y9ArSZIkSeosQ68kSZIkqbMMvZIkSZKkzjL0SpIkSZI6y9ArSZIk
SeosQ68kSZIkqbMMvZIkSZKkzjL0SpIkSZI6y9ArSZIkSeosQ68kSZIkqbMMvZIkSZKkzjL0SpIkSZI6y9ArSZIkSeq
sOdNdgDRRR/78MubOnTvZUiSJEmaBezplSRJkiR1lqFXkiRJktRZhl5JkiRJUmcZeiVJkiRJnWXolSRJkiR1lqFXkiRJktRZhl5JkiRJUmcZeiVJkiRJnWXolSRJkiR1lrM3a

9YZGhqa7hIkSZIkTaOJZIJU1RSWIg1OkjWBP053HZIkSZJmjNWr6pqxGtjTq9nk5vbr6sBt01mIOmMV4E94T2l
wvKc0SN5PGjTvKQ3adN9TqwDXLq6RoVez0W1V5RhnLbEkw996T2kgvKc0SN5PGjTvKQ3aDLinxnVOJ7KSJE
mSJHWWoVeSJEmS1FmGXs0mC4F92q/SIHhPadC8pzRI3k8aNO8pDdqsuKecvVmSJEmS1Fn29EqSJEmSOsv
QK0mSJEnqLEOvJEmSJKmzDL2SJEmSpM4y9GpGSfLmJH9McneS3yR59mLab962uzvJ5Un2WFq1anaYyD2V
5JVJfprkL0mGkpyZ5EVLs17NbBP9N6pnv2clWZTknCkuUbPMJP6/t3yS/0xyZZKFSS5LssvSqlcz3yTuqR2TnJvk
ziTXJflqkocvrXo1cyV5TpIfJbk2SSXZdhz7zMjfzQ29mjGSvBb4DPCfwMbA/wInJFljlPaPB45v220MfBw4KMmr
lkrBmvEmek8BzwF+CrwEeArwM+BHSTae+mo1003ifhrebx7wdeDkqa5Rs8sk76nvAM8HdgWeCGwPXDy1l
Wq2mMTvUpvR/Pt0KPAkYDtgU+ArS6NezXgrAecCbxlP45n8u7kfWaQZI8kvgbOr6k096y4CflBVe4/Qfn9gm
6par2fdF4ENq+oZS6NmzWwTvadGOcbvgG9X1b5TVKZmicneT0mOAv4A/BXYtqo2mupaNTtM4v97WwFH
AWtV1c1Lr1LNFpO4p94NvKmq1u5Z91Zgr6p67NKoWbNDkgJeUVU/GKPNjP3d3J5ezQhJHkLTs/Y/fZv+B3j
mKLs9Y4T2JwKbJFlusBVqtpnkPdV/jGWAVQB/uXyQm+z9lGRnYG1gn6mrTrPRJO+pbYCzgL2SXJPkkiT/lWSF
KSxVs8Qk76lfAKsneUkajwJeDfx46ipVh83Y383nTOfJpR7/BCwL3NC3/gbg0aPs8+hR2s9pj3fdIAvUrDOZe6rf
u2iG9nxngHVpdprw/ZRkHWA/4NlVtSjJ1Fao2WYy/0atBWwG3A28oj3GF4B/BHyuVxO+p6rxO+hR2s9pj3fdIAvUrDOZe6rf
N71A/BN46hXWqu2bs7+b29Gqm6R9vnxHWLa79SOv14DXRe6pplGwPfAR4bVX9eQrq0uw0rvspybLAkcC
Hq+qSpVGYZq2J/Bu1TLttx6r6VVUdD+wJzLe3Vz3GfU8lWR84CNiXppd4K+DxwhcQ/IOZrNYemAOzSG4QjlKXEH8k8mwT+TSJ1nGmA/jaxMQnhxiSPnUngoLvG7ZRKpHmlT20VL8bDA2w8rtd0T2UIgs6i5Qbv55yP2VftQjc4N3Wpm0A8kwvTGhO9+IW8jwgnSJQUWq8qgMiXtyY4cCPw3AGBV/z8LIKL0tGSTOD/qUHYa5Kz0ZDJaAp0VTQWSkcqz/ZTaFQcHyaS0h+K1o33a/B0hD4ZqBYqoO3JTrs53Uzq8X8+7wL2KA1/lFDWzYpyQLJ5VFsTpjB4VFAgVZMr/SYk6KPYUDMd6bHhH1qwQZeS1ldxDRZwUZ9c/hhXh9ADkf45dPbAbOUFAcG8B8T0H3Db/sIr2ZSFxp/0xq2bUbr+ZKmmsf/lE/C1F6tC6iIZaGFJhGPdvbZ5RJLb0HWlCO8OhgHU4xcNTqCQK2d7jVNu/5Cc3h/Qq/4GM9/Fcv57Xc8l8xdqx4f1lq6X/9cONDddP/H/SvDSDFRzprHKn2ONGT5t/PDVAFxp5Ma+3nLOn5N2AqaNxZdd7d8zW8w5Wj0N7BcEYRSdD0/KSG8D2YVRQb/Ni+bgOaOk6ED9dDR+3aI3Hx9mDBYf2pVk/tRE+Sj4Y4oW0vf8xKx3XNUTD4fc3zX38rv/QW5iLhB/eUh7Ez1UeJp3IU/1yq3/jYX7qH6BM1kINd8Bju8uM7NuaAkU91+l3PNc+UEiA7u7ruMadtGfrn0+1nQuV5Z5nlW0pb5kQA4x4G6AnRyYONQzN0+G6lM+lqmvBT1TK0PqU+F4+CeB/M4dEL5x4hkTQX4Ty2+lHWGmYP6Dv2Gn1DR3Hm/8ujLDEQ9fK/Dx/pz1XAkQf4nFlQ4HL4jq1lOO6RkQwIAQn84DTgcNp4f5A9/4hrY/7xHfy13KrdBGkqqJ1DeLrDkSKqbbJHHPjKqR0FsOP1YyJvYj5eLbHUmmfpNzQn1QCftj4dQvo0jYQJVV49t/PlqIRmXqBmVx4cFzUWyqbJyHSDsO9HCJY4G6Ba6PvoaqYGlm3nQ4M+8uNWkbuH2SIcz8T8PtDyYmV4d8cSUg0cHRhlCWE2Z/B3vkG9mwNK2J/cV2xVFL9JGXG8BuKzhn0qTb7Mo6j7rrJ3PADq6f2ac1ygkSd+7Yf6Q9UiXnKtMoKbEkVuKz9dRIHKE8LAzoeCEYEDJ6pDtJzbHgE1wnzMBd8mQvdYz6mYtG+1g8zMz5g+7lR2ewSPbGi+cUGWdZ44Zpf2n4qIfyHY7ZjGVzR7ntkYUFWeZQ9IDYnX7VnZ9lkXAmZmqJIf4dBxyX5mD+W5h4EZGHhtC3TKkE5+NN5HfEPDsBG/5h9byB84GuCPGFd9XUJdhbGb8LjgL6MvNfNNm4tLPG8MGEyBl8NEg6DmtYqOJ7Gkd3CF4zSPvqo6Hb/4B2wQXV8XyZ0FTg2ebDG+XhsJm9qbq/7t3NhE8uWwsDNu7K1dTxtlwUtfe5B7wT/A6T9dIjmrX1hJ7VY4mjdG+X6tCVm5twO8qRzxQHQv8tJwSpqXT1e0JPx7L0dSwffbOcLzfDHDKEe8ZK8lGcr2vSScQ4EEvyLR5kYbC3Vr3NDv2eAMxzmkRFKkbwSHhZTu88Fs3a5U8OJGn33/CSRkrTTHSRaYpOHvMiqr6xwYrVoK4N0BSP5c5NoyD5PzQO6j3rtxKzG4o/pjnKd0Zt2ap7lv/MgH3T8y1r2Yqvl6ZLoBAbUd3R7sYTz0bF6JaZi7r7q9Czy7BPW6hmK6AQDZR6kpMdw0DAPUqFo/xD7DYVy6VbbO6sd4b89e1BBfW0iTQABdZyYyVUSJ8yZdcSGp3fWLP2o1dGaiK8oqkTk3Lk4TSMu8eR0h7mTTnmgzqkkQUUZ3+4kP/MZPhK+bPcEK/4Jc4SJPKYAPOMfWQEmjLR0lfNm7uRmS0qdBSNEFpu0LK0yUhqTo5hX5nqZbfxdgPlw2+PSVJRX5bOcV0f2D0G7D9c3ea3IsUpPHmQoIY5Bxq/nbmIGe9Tk2jPN8QRfMCGzv0L35zR9fMR8P1XYOMDpZCdvSMtNEVh4Bq6smS4M1EBV4ZxiqIN3WFuTLqnT+nMw8q35D+pTBzTIrsMgqrxp1lMHmY3C86Ni3LP2BGkB6IeRYFq7T+nP6ddNDp2yqsO6Lr3xSKu8E6hMjhzkQBIzh/Sq69ydPLXqtzQGLk2k4d/eBF8dzLUCWCcU3TeunxNWdVDZ4Nn/KeBwBGShRK5Mv3hz7ggaf5zHhzDLPwZeMvS+9MN6Q8eqZlYnEjj7DndMpHKzL1/1cIxcyuypIgOpOpT3a/gncHz5/+hGsE80tCFqKuv/6ocjwt0BAhH3b1aZ5EgPrdTHn/AW/hW9oUcXwdjf3kEgiPTdr/Nx2F2jxrBc9ruPw7lW3pgnHojwjj9uCbPpPCO4PKY6PhM8jRw+0W8X/P4Npzz2fJcTJCwGN1sHSVsYOhyBkLOQh2SYfsJ4eVuSqUkRMm9Xcoqvk6uO2XATAHtFspCfyuW1Mgv0XU/07v2LD2CjHFwWSG5ebPDMiX5X5m2FUhbRKE3Q5fxb/YoBRUmbc3c6V1nQCKLCnJhPi5MDEIkfL/wt+fy3C3UdqPtkDFBALG0MoZYDJ33p93Cu9jrRpRyf1TwE1MpsjlpHM7yTU9rDEkq1ga4fNAwA7vbWtmPRF5R3R9TYjcEE/S2mKDPw0uIi0mD61lnVZpJnTfF0jTpxQ4x+lP3cGJn3Ltc=
s/w8mRwbV7dPk3QAAAABJRU5ErkJggg==\n",

    "text/plain": [

     "<Figure size 1000x500 with 1 Axes>"

```
      ]
     },
     "metadata": {
      "needs_background": "light"
     },
     "output_type": "display_data"
    }
   ],
   "source": [
    "plt.figure(figsize=[10,5],dpi = 100)\n",
    "plt.title('Accuracy Comparison')\n",
    "plt.xlabel('Accuracy')\n",
    "plt.ylabel('Algorithm')\n",
    "sns.barplot(x = acc,y = model,palette='dark')"
   ]
  },
  {
   "cell_type": "code",
   "execution_count": 42,
   "metadata": {},
   "outputs": [
    {
     "name": "stdout",
     "output_type": "stream",
     "text": [
      "Decision Tree --> 0.9\n",
      "Naive Bayes --> 0.990909090909091\n",
      "SVM --> 0.9795454545454545\n",
      "Logistic Regression --> 0.9522727272727273\n",
    "RF --> 0.990909090909091\n",
    "XGBoost --> 0.9931818181818182\n"
   ]
  }
 ],
 "source": [
  "accuracy_models = dict(zip(model, acc))\n",
  "for k, v in accuracy_models.items():\n",
  "    print (k, '-->', v)"
 ]
},
{
 "cell_type": "markdown",
 "metadata": {},
 "source": [
  "## Making a prediction"
 ]
},
{
 "cell_type": "code",
 "execution_count": 43,
 "metadata": {},
 "outputs": [
  {
   "name": "stdout",
   "output_type": "stream",
   "text": [
    "['coffee']\n"
   ]
```

```
    }
   ],
   "source": [
    "data = np.array([[104,18, 30, 23.603016, 60.3, 6.7, 140.91]])\n",
    "prediction = RF.predict(data)\n",
    "print(prediction)"
   ]
  },
  {
   "cell_type": "code",
   "execution_count": 44,
   "metadata": {},
   "outputs": [
    {
     "name": "stdout",
     "output_type": "stream",
     "text": [
      "['jute']\n"
     ]
    }
   ],
   "source": [
    "data = np.array([[83, 45, 60, 28, 70.3, 7.0, 150.9]])\n",
    "prediction = RF.predict(data)\n",
    "print(prediction)"
   ]
  }
 ],
 "metadata": {
  "kernelspec": {
   "display_name": "Python 3",
   "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": "3.8.5"
  }
},
"nbformat": 4,
"nbformat_minor": 4
}
```