Estimation of COVID19 infection using Machine Learning Algorithms

Abstract

The COVID19 pandemic has affected the world very badly as the number of cases are not declining in any day. The growth and the mutation of virus in a different form has created a lot of concern among the researchers working on it. This project enables the prioritization of test by taking the symptoms from a person and feeding the input data to it which in turn gives an output of estimation of how much the person is infected with COVID19.

Introduction

COVID-19 is a viral infectious disease in which Wuhan was the initial epicenter and the maximum confirmed cases worldwide as of 8 November 2020 were 50,395,239 including 1,258,235 deaths with 35,629,514 recoveries. Coronavirus 2, also known as SARS-CoV-2 Extreme Acute Respiratory Syndrome, is the virus which caused the COVID-19 pandemic. This virus spreads rapidly when the infected person is in close contact. The lower concentrations respiratory droplets when an infected person coughs or sneezes, which may not be visible to the naked eye.

These droplets also may originate from saliva, which can be inhaled through the mouth or nose into a healthy person's lungs, thus spreading the disease from one person to another. If they touch their eyes, nose or mouth after they touch any contaminated objects, people may also get sick. The longevity of the virus is more on stainless-steel and plastic surfaces that are 72 hours, but the lifespan of this contagious virus is low on copper and cardboard. Tiredness, fever, and dry cough with moderate signs of runny nose, sore throat, pain and aches, and nasal congestion are the basic symptoms of the disease.

Old people with high blood pressure, heart attacks and diabetes have a serious risk of COVID-19 infection. The World Health Organization (WHO) suggests that everyone maintain a social distance of around 3 feet, as it will prevent droplets from being inhaled, wash their hands with soap and water for at least 20 seconds, as it will kill the virus, and also avoid touching their mouths. Aggregated data on the symptoms of other infected patients would play an enormous role in this study in order to speed up the early detection of COVID-19 infected individuals.

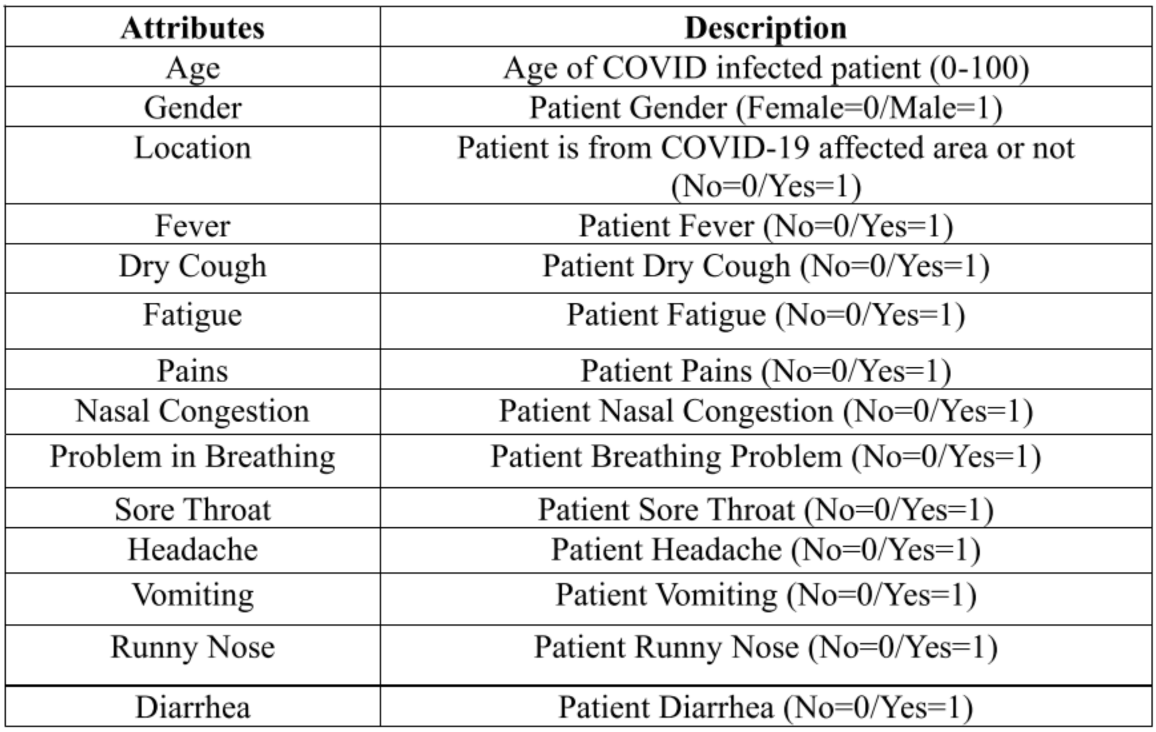
To identify a person with COVID-19, this paper focuses primarily on the symptoms as discussed above. After that, by applying various machine learning algorithms, we tried to estimate the likelihood of getting infected with the disease. We have developed a machine learning (ML) model for this analysis that uses Random Forest Classification to figure out the likelihood of a person getting COVID-19 infected based on his or her symptoms.

Motivation

In the early months of this year, COVID 19 was discovered. It's been months, but not a single vaccine or cure to this lethal virus has been available yet. If we can even identify the disease beforehand during these tough times, it might allow a person to contact medical officials in time and diagnose and obtain care in time.

Dataset

Table 1: Description about the attributes of COVID-19 Symptoms dataset.

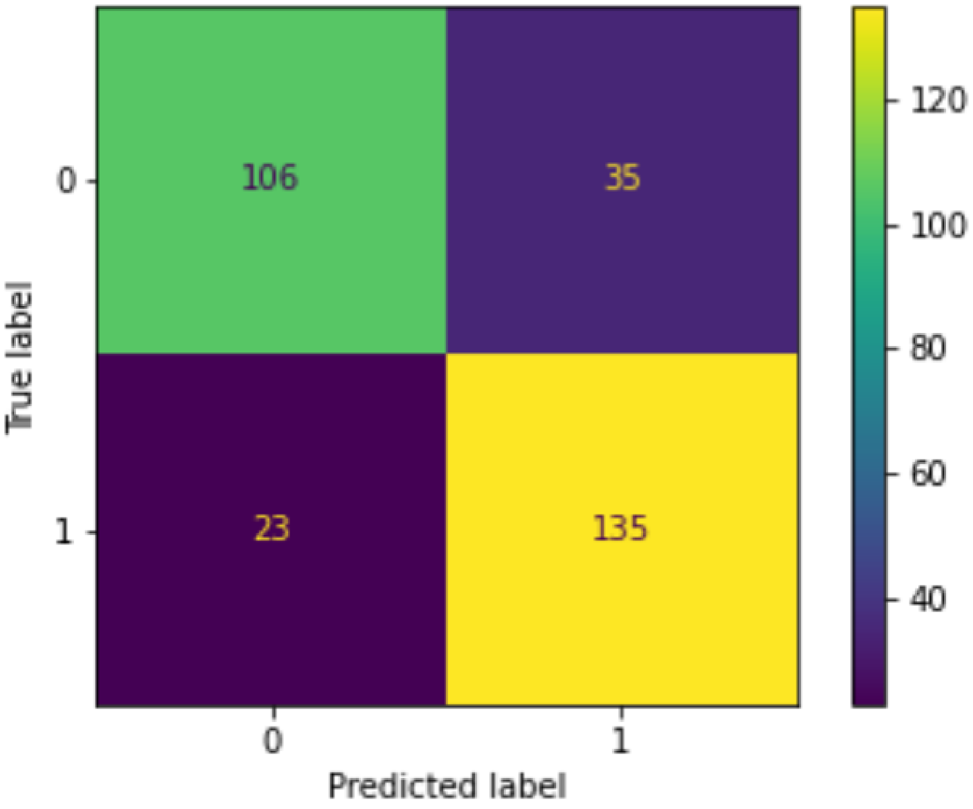


Attributes	Description
Age	Age of COVID infected patient (0-100)
Gender	Patient Gender (Female=0/Male=1)
Location	Patient is from COVID-19 affected area or not (No=0/Yes=1)
Fever	Patient Fever (No=0/Yes=1)
Dry Cough	Patient Dry Cough (No=0/Yes=1)
Fatigue	Patient Fatigue (No=0/Yes=1)
Pains	Patient Pains (No=0/Yes=1)
Nasal Congestion	Patient Nasal Congestion (No=0/Yes=1)
Problem in Breathing	Patient Breathing Problem (No=0/Yes=1)
Sore Throat	Patient Sore Throat (No=0/Yes=1)
Headache	Patient Headache (No=0/Yes=1)
Vomiting	Patient Vomiting (No=0/Yes=1)
Runny Nose	Patient Runny Nose (No=0/Yes=1)
Diarrhea	Patient Diarrhea (No=0/Yes=1)

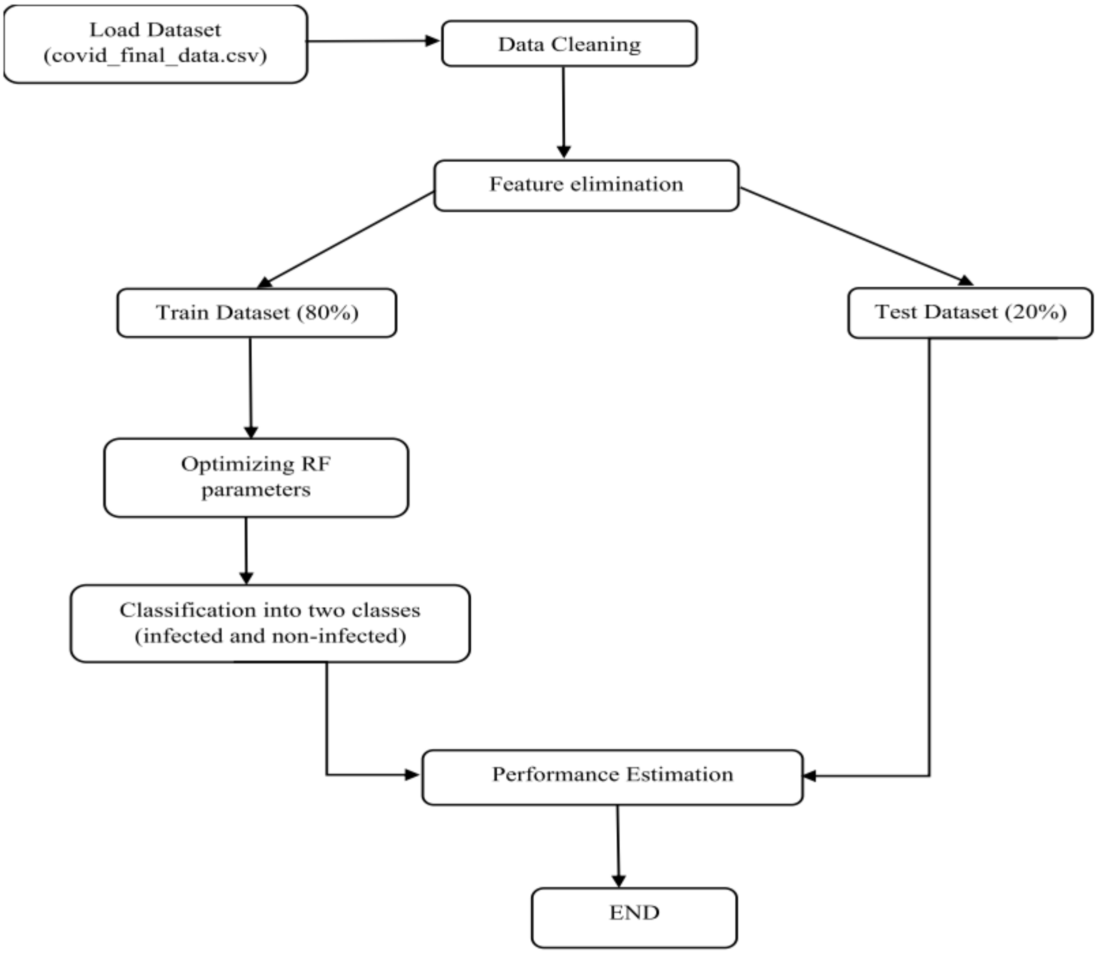
The dataset provides information about patient's symptoms and whether they are infected with COVID-19 or not. Considering all these details in both the dataset, another reliable dataset is created to maximize the data quality which consists of a total 14 columns with 13 independent variables and one dependent variable. It has 1495 rows. The dataset consists of 14 attributes out of which 11 are the patient's symptoms. As the output of this model will give whether the patient is infected or not with COVID-19, i.e. Yes or No. So, this problem is treated as Binary Classification.

Methodology

First of all, the COVID-19 Symptoms dataset is uploaded and the attributes are separated into independent and dependent variables, then the dataset is split into two parts - one being the train data and the other being the test data by which we can predict. The former being the larger part we process it through different machine learning algorithms to get the best possible output, and we observe that by using the Random Forest Classifier algorithm we get the best possible accuracy. The performance of Random Forest Classifier was the highest as compared to other algorithms. The output variable is the target variable and all the other attributes of the symptoms and individual's necessary details are fed into it and the desired result is shown. The confusion matrix for Random Forest Classifier is shown below in the below figure.



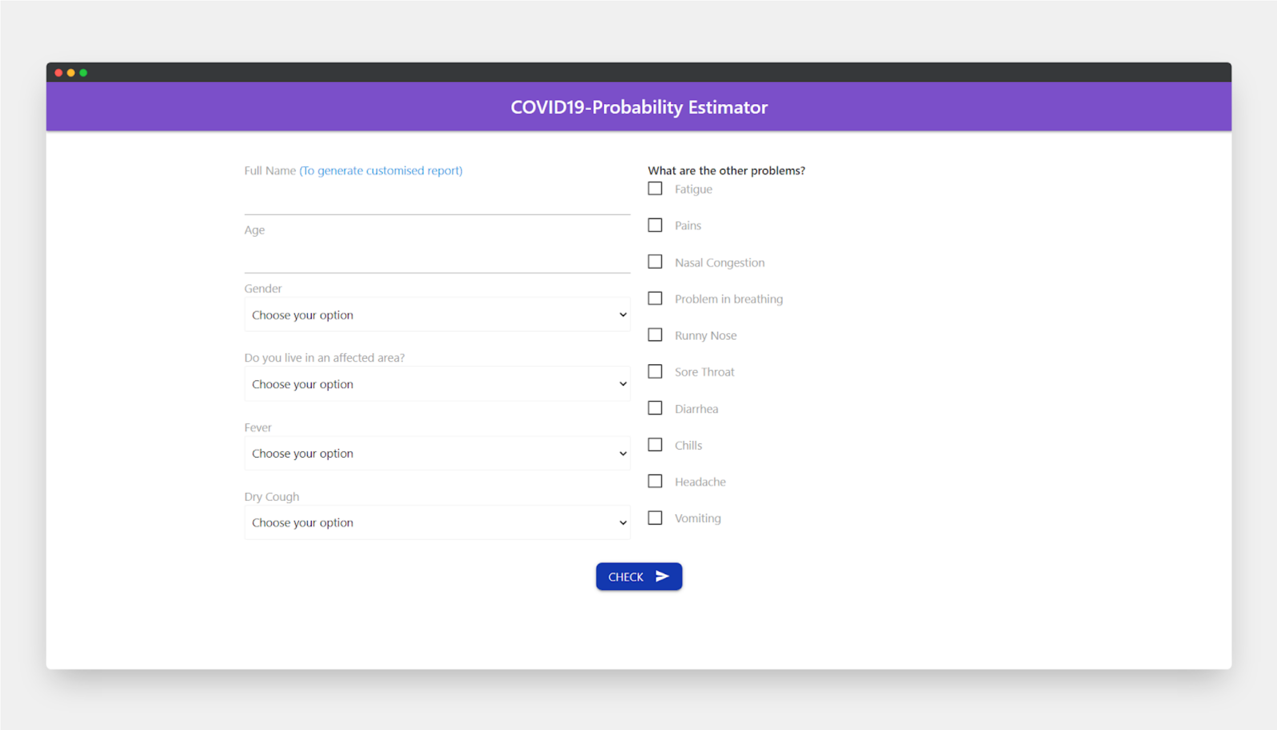
After the training of the model is complete, then the later dataset can be used for predicting the presence of COVID 19 in a particular individual. The Covid Output is the final result that we need to predict from our study. After the application of different machine learning algorithms, we construct a model that can be used to predict the patient's final Covid Output. For the last step, the test dataset is validated with the dataset that is trained and thereby calculating the rate of accuracy. The complete process is shown below.



Technological Stack

- Sklearn - Machine Learning Library
- Pandas - Data Manipulation Library
- Numpy - Numerical Computation Library
- Django - Python web framework
- HTML
- Material CSS

Demo



COVID19-Probability Estimator

Full Name (To generate customised report)

Age

Gender

Choose your option

Do you live in an affected area?

Choose your option

Fever

Choose your option

Dry Cough

Choose your option

What are the other problems?

□ Fatigue

□ Pains

□ Nasal Congestion

□ Problem in breathing

□ Runny Nose

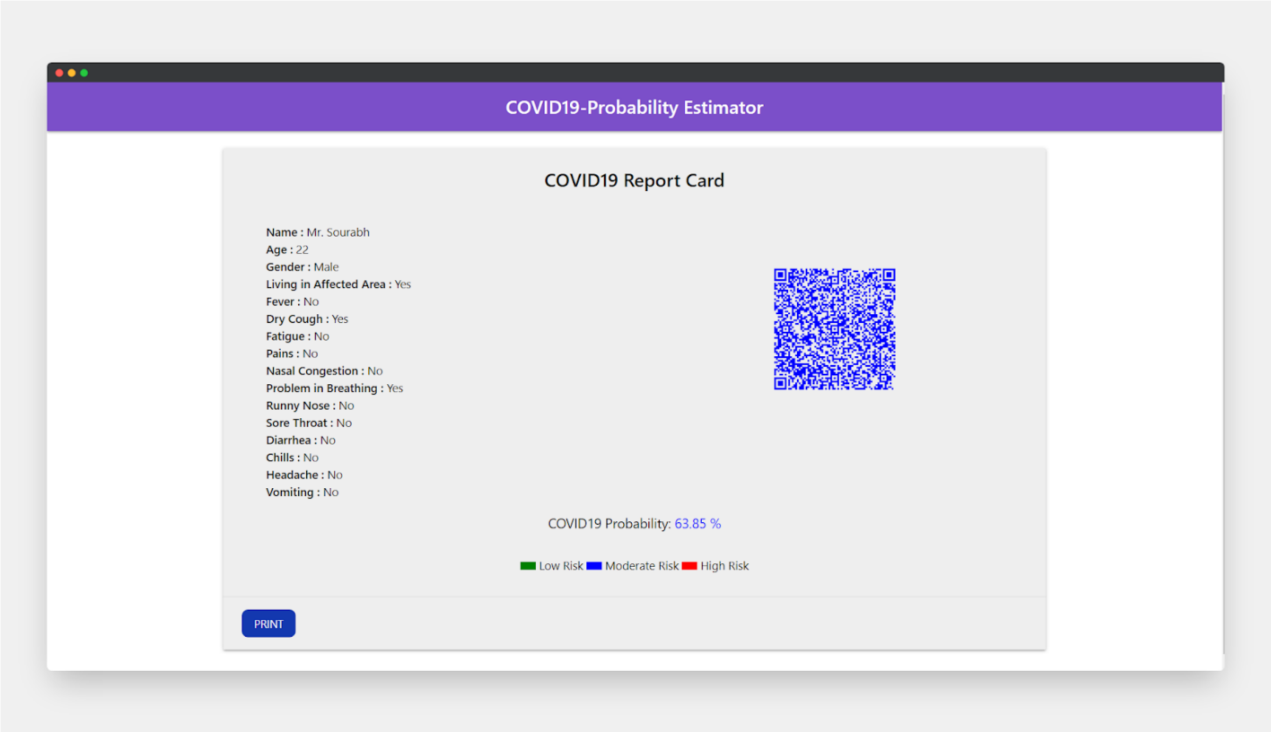
□ Sore Throat

□ Diarrhea

□ Chills

□ Headache

□ Vomiting



COVID19-Probability Estimator

COVID19 Report Card

Name : Mr. Sourabh
Age : 22
Gender : Male
Living in Affected Area : Yes
Fever : No
Dry Cough : Yes
Fatigue : No
Pains : No
Nasal Congestion : No
Problem in Breathing : Yes
Runny Nose : No
Sore Throat : No
Diarrhea : No
Chills : No
Headache : No
Vomiting : No

COVID19 Probability: 63.85 %

🟩 Low Risk 🟦 Moderate Risk 🟥 High Risk

Source Code

requirements.txt

```
alabaster==0.7.12
```

```
altgraph==0.16.1
appdirs==1.4.3
appmode==0.7.0
asgiref==3.2.7
astroid==2.3.3
atomicwrites==1.3.0
attrs==19.3.0
auto-py-to-exe==2.6.5
Automat==20.2.0
autopep8==1.5.1
Babel==2.8.0
backcall==0.1.0
bcrypt==3.1.7
beautifulsoup4==4.8.2
biopython==1.76
bleach==3.1.3
boto3==1.12.31
botocore==1.15.31
bottle==0.12.17
bottle-websocket==0.2.9
bs4==0.0.1
cachetools==4.0.0
certifi==2019.9.11
cffi==1.12.3
chardet==3.0.4
Click==7.0
cloudpickle==1.3.0
```

```
colorama==0.4.3
constantly==15.1.0
cryptography==2.9
cssselect==1.1.0
cycler==0.10.0
decorator==4.4.2
defusedxml==0.6.0
detectlanguage==1.3.0
diff-match-patch==20181111
distlib==0.3.0
dj-database-url==0.5.0
Django==3.0.5
django-heroku==0.3.1
django-qr-code==1.1.0
docutils==0.15.2
Eel==0.10.4
entrypoints==0.3
et-xmlfile==1.0.1
filelock==3.0.12
flake8==3.7.9
Flask==1.1.1
future==0.17.1
gensim==3.8.1
gevent==1.4.0
gevent-websocket==0.10.1
google-api-core==1.16.0
google-auth==1.12.0
```

```
google-cloud-core==1.3.0
google-cloud-storage==1.26.0
google-resumable-media==0.5.0
googleapis-common-protos==1.51.0
greenlet==0.4.15
helpdev==0.6.10
hyperlink==19.0.0
idna==2.8
imagesize==1.2.0
importlib-metadata==1.5.0
incremental==17.5.0
intervaltree==3.0.2
ipykernel==5.1.4
ipython==7.13.0
ipython-genutils==0.2.0
isort==4.3.21
itsdangerous==1.1.0
jdcal==1.4.1
jedi==0.15.2
Jinja2==2.11.1
jmespath==0.9.5
joblib==0.14.1
jsonschema==3.2.0
jupyter-client==6.1.0
jupyter-core==4.6.3
keyring==21.2.0
kiwisolver==1.1.0
```

```
lazy-object-proxy==1.4.3
lxml==4.5.0
MarkupSafe==1.1.1
matplotlib==3.2.1
mccabe==0.6.1
mistune==0.8.4
nbconvert==5.6.1
nbformat==5.0.4
nltk==3.4.5
notebook==6.0.3
numpy==1.17.2
numpydoc==0.9.2
openpyxl==3.0.0
packaging==20.3
pandas==0.25.1
pandocfilters==1.4.2
paramiko==2.7.1
parsel==1.5.2
parso==0.5.2
pathtools==0.1.2
pefile==2019.4.18
pexpect==4.8.0
pickleshare==0.7.5
Pillow==7.1.1
pipenv==2018.11.26
pluggy==0.13.1
prometheus-client==0.7.1
```

```
prompt-toolkit==3.0.4
Protego==0.1.16
protobuf==3.11.3
psutil==5.7.0
psycopg2==2.8.5
ptyprocess==0.6.0
pyasn1==0.4.8
pyasn1-modules==0.2.8
pycodestyle==2.5.0
pycparser==2.19
PyDispatcher==2.0.5
pydocstyle==5.0.2
pyflakes==2.2.0
Pygments==2.6.1
PyHamcrest==2.0.2
PyInstaller==3.5
pylint==2.4.4
PyNaCl==1.3.0
pyOpenSSL==19.1.0
pyparsing==2.4.6
PyPDF2==1.26.0
PyQt5==5.12.2
PyQt5-sip==4.19.19
pyqt5-tools==5.13.0.1.5
PyQtWebEngine==5.12.1
pyrsistent==0.15.7
python-dateutil==2.8.0
```

```
python-dotenv==0.10.3
python-jsonrpc-server==0.3.4
python-language-server==0.31.9
pytz==2019.2
pywin32==225
pywin32-ctypes==0.2.0
pywinpty==0.5.7
pyzmq==19.0.0
QDarkStyle==2.8.1
qrcode==6.1
QtAwesome==0.7.0
qtconsole==4.7.2
QtPy==1.9.0
queuelib==1.5.0
requests==2.22.0
rope==0.16.0
rsa==4.0
s3transfer==0.3.3
scikit-learn==0.22.2.post1
scipy==1.4.1
Scrapy==2.0.1
selenium==3.141.0
Send2Trash==1.5.0
service-identity==18.1.0
six==1.12.0
sklearn==0.0
smart-open==1.10.0
```

```
snowballstemmer==2.0.0
sortedcontainers==2.1.0
soupsieve==2.0
Sphinx==3.0.1
sphinxcontrib-applehelp==1.0.2
sphinxcontrib-devhelp==1.0.2
sphinxcontrib-htmlhelp==1.0.3
sphinxcontrib-jsmath==1.0.1
sphinxcontrib-qthelp==1.0.3
sphinxcontrib-serializinghtml==1.1.4
spyder==4.1.2
spyder-kernels==1.9.0
sqlparse==0.3.1
terminado==0.8.3
testpath==0.4.4
tornado==6.0.4
traitlets==4.3.3
Twisted==20.3.0
typed-ast==1.4.1
urllib3==1.25.6
vaderSentiment==3.3.1
virtualenv==20.0.15
virtualenv-clone==0.5.4
w3lib==1.21.0
watchdog==0.10.2
wcwidth==0.1.8
webencodings==0.5.1
```

```
Werkzeug==1.0.0
whichcraft==0.6.1
whitenoise==5.0.1
wrapt==1.11.2
xgboost==1.0.2
xlrd==1.2.0
xlwt==1.3.0
yapf==0.29.0
zipp==3.1.0
zope.interface==5.0.2
```

manage.py

```
#!/usr/bin/env python
"""Django's command-line utility for administrative tasks."""
import os
import sys


def main():
    os.environ.setdefault('DJANGO_SETTINGS_MODULE',
'covidpred.settings')
    try:
        from django.core.management import execute_from_command_line
    except ImportError as exc:
        raise ImportError(
            "Couldn't import Django. Are you sure it's installed and "
            "available on your PYTHONPATH environment variable? Did you "
            "forget to activate a virtual environment?"
        ) from exc
    execute_from_command_line(sys.argv)



if __name__ == '__main__':
    main()
```

base.html

```
<!DOCTYPE html>
{% load static %}
<html>
<head>


  <title> COVID19-Probability</title>
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <link rel="shortcut icon" href="{% static 'img/icon.png' %}">
  <link href="https://fonts.googleapis.com/icon?family=Material+Icons"
rel="stylesheet">
  <link rel="stylesheet" href="{% static 'styles/materialize.min.css' %}">
</head>
<header>
                      <nav style="background-color:#7b4fc9;">
    <div class="nav-wrapper">
      <a class="brand-logo center"
      href="{% url 'index' %}"
      style="font-size:24px;font-weight: 599;">COVID19-Probability
Estimator</a>


    </div>
  </nav>
</header>
<style>
  label{
    font-size: 15px;
  }
</style>
<body>

          {% block content %}

  {% endblock %}

</body>
</html>
```

index.html

```
{% extends 'base.html' %}

{% block content %}

   <div class="container" style="padding:40px 15px 0px 15px">


      <div>
        <form class="row" method="POST" action="{% url 'index' %}">
          {% csrf_token %}
          <div class="col m6 s12">
      <div >
        <label for="full_name">Full Name<a class="tooltipped"
name="full_name" data-position="right" data-tooltip="I am a tooltip"> (To
generate
          customised report)</a>
          </label>
        <input
          type="text"
          id="full_name"
          name="full_name"

        />
      </div>
      <div>
        <label for="age">Age</label>
        <input
          type="number"
          step="1"
          class="form-control"
          id="age"
          min=1 max=150
          name="age" required
        />
      </div>
      <div class="form-control">
        <label for="gender">Gender</label>
        <select name="gender" class="browser-default" required>
          <option value="" disabled selected>Choose your option</option>
          <option value="0">Female</option>
          <option value="1">Male</option>
        </select>
      </div><br>
      <div>
        <label for="cough">Do you live in an affected area?</label>
        <select name="lives_in_affected_area" required class="browser-default">
          <option value="" disabled selected>Choose your option</option>
          <option value="1">Yes</option>
          <option value="0">No</option>
        </select>
      </div><br>
      <div>
        <label for="fever">Fever</label>
        <select name="fever" required class="browser-default">
          <option value="" disabled selected>Choose your option</option>
          <option value="1">Yes</option>
          <option value="0">No</option>
        </select>
      </div><br>
      <div>
        <label for="cough">Dry Cough</label>
        <select name="cough" required class="browser-default">
          <option value="" disabled selected>Choose your option</option>
          <option value="1">Yes</option>
          <option value="0">No</option>
        </select>
```

```
          </div>
        </div>


        <div class="col m6 s12" >
          <strong>What are the other problems?</strong><br>
          <div>
            <label>
              <input id="indeterminate-checkbox fatigue" name="fatigue"
type="checkbox" />
              <span>Fatigue</span>
            </label>
          </div>
           <br>
          <div>
            <label>
              <input id="indeterminate-checkbox pains" name="pains"
type="checkbox" />
              <span>Pains</span>
            </label>
          </div> <br>
          <div>
            <label>
              <input id="indeterminate-checkbox nasal_congestion"
name="nasal_congestion"  type="checkbox" />
              <span>Nasal Congestion</span>
            </label>
          </div> <br>
          <div>
            <label>
              <input id="indeterminate-checkbox shortness_of_breath" name="shortness_of_breath"  type="checkbox" />
              <span>Problem in breathing</span>
            </label>
          </div> <br>
          <div>
            <label>
              <input id="indeterminate-checkbox runny_nose" name="runny_nose" type="checkbox" />
              <span>Runny Nose</span>
            </label>
          </div> <br>
          <div>
            <label>
              <input id="indeterminate-checkbox sore_throat" name="sore_throat" type="checkbox" />
              <span>Sore Throat</span>
            </label>
          </div> <br>
          <div>
            <label>
              <input id="indeterminate-checkbox diarrhea" name="diarrhea" type="checkbox" />
              <span>Diarrhea</span>
            </label>
          </div> <br>
          <div>
            <label>
```

```
              <input id="indeterminate-checkbox chills" name="chills" type="checkbox" />
              <span>Chills</span>
            </label>
          </div> <br>
          <div>
            <label>
              <input id="indeterminate-checkbox headache" name="headache" type="checkbox" />
              <span>Headache</span>
            </label>
          </div> <br>
          <div>
            <label>
              <input id="indeterminate-checkbox vomiting" name="vomiting" type="checkbox" />
              <span>Vomiting</span>
            </label>
          </div><br>
                <br>
        </div>


        <center>


          <button type="submit" class="btn" style="background-color: #1338af;border-radius: 8px;">
            Check<i class="material-icons right">send</i>
          </button>
```

```
     </center>
    </form>
   </div>
  </div>

{% endblock %}
```

result.html

```
{% extends 'base.html' %}

{% block content %}

{% load qr_code %}
<style>
 .card_data{
  padding:10px 0px 0px 20px!important;
 }
</style>

 <!-- Result Block -->
 <div class="container" style="padding: 15px 0px 0px 0px;">
  <div class="row">
    <div class="card grey lighten-3">
     <div class="card-content black-text ">
      <center><span class="card-title" style="font-weight: 500;">COVID19 Report Card</span></center>
      <br>
      <div class="row card_data">
```

```
          <div class="col m6 l6 xl6" style="text-align: justify;">{% for key,value in user_details_API.items %}
           <span style="font-weight: 500;">{{key}} : </span>
           <span>{{value}}</span><br>
          {% endfor %}
        </div>
          <div class="col m6 l6 xl6 center-align" style="margin-top: 50px;">
            {% qr_from_text user_json_data size="T" %}
          </div>
        </div>
        <center>
          {% if result %}
          <h6 class="Probability">COVID19 Probability: <span class="probability-value">{{ result }} %</span></h6>
          {%else%}
          <h6 class="Probability">COVID19 Probability: <span class="probability-value">0%</span></h6>
          {% endif %}<br>


          <div style="padding:5px 0px 5px 0px">
            <svg width="20" height="10">
              <rect width="36" height="18" style="fill:green"/><span> Low Risk</span>
            </svg>
            <svg width="20" height="10">
              <rect width="36" height="18" style="fill:blue"/><span> Moderate Risk</span>
            </svg>
            <svg width="20" height="10">
```

```
                  <rect width="36" height="18" style="fill:red"/><span> High Risk</span>
                </svg>
              </div>
            </center>
          </div>
          <div class="card-action">
            <a href="#" class="btn-flat" style="color:white;background-color: #1338af;border-radius: 8px;" id="print-btn" onclick="printDiv()">Print</a>
          </div>
        </div>
    </div>
  </div>

  <script type="text/javascript">
    function printDiv() {
    var printContents = document.querySelector('.card-content').innerHTML;
    var originalContents = document.body.innerHTML;
     document.body.innerHTML = `
     <div style="padding:50px 10px 10px 20px;border: solid 5px #000">
     ${printContents}
     </div>
     `;
     window.print();
     window.close()
     document.body.innerHTML = originalContents;
    }
    let val = document.querySelector('.probability-value')
    // Low, Mild, Moderate, Severe
    if(parseFloat(val.textContent)<=40){
      val.style.color="green"
      document.querySelector('path').style.fill="green"
    }

    else if(parseFloat(val.textContent)>40 && parseFloat(val.textContent)
<=75){
      val.style.color="blue"
      document.querySelector('path').style.fill="blue"
    }
    else {
      val.style.color="red"
      document.querySelector('path').style.fill="red"
    }
</script>
{% endblock %}
```

settings.py (Config)

```
"""
Django settings for covidpred project.

Generated by 'django-admin startproject' using Django 3.0.5.

For more information on this file, see
https://docs.djangoproject.com/en/3.0/topics/settings/

For the full list of settings and their values, see
```

```
https://docs.djangoproject.com/en/3.0/ref/settings/
"""

import os

# Build paths inside the project like this: os.path.join(BASE_DIR, ...)
BASE_DIR = os.path.dirname(os.path.dirname(os.path.abspath(__file__)))



# Quick-start development settings - unsuitable for production
# See https://docs.djangoproject.com/en/3.0/howto/deployment/checklist/

# SECURITY WARNING: keep the secret key used in production secret!
SECRET_KEY = 'ff45r5bt$3e%ztoo$x%b=v*$-z#elm#)&b@%2(k2w3(tt=f4l7'

# SECURITY WARNING: don't run with debug turned on in production!
DEBUG = True

ALLOWED_HOSTS = ['192.168.43.132','127.0.0.1']

STATICFILES_DIRS=[
    os.path.join(BASE_DIR, 'static')

]

# Application definition
```

```
INSTALLED_APPS = [
    'django.contrib.admin',
    'django.contrib.auth',
    'django.contrib.contenttypes',
    'django.contrib.sessions',
    'django.contrib.messages',
    'django.contrib.staticfiles',
    'qr_code',
    'covid'
]

MIDDLEWARE = [
    'django.middleware.security.SecurityMiddleware',
    'django.contrib.sessions.middleware.SessionMiddleware',
    'django.middleware.common.CommonMiddleware',
    'django.middleware.csrf.CsrfViewMiddleware',
    'django.contrib.auth.middleware.AuthenticationMiddleware',
    'django.contrib.messages.middleware.MessageMiddleware',
    'django.middleware.clickjacking.XFrameOptionsMiddleware',
]

ROOT_URLCONF = 'covidpred.urls'

TEMPLATES = [
    {
        'BACKEND': 'django.template.backends.django.DjangoTemplates',
        'DIRS': [os.path.join(BASE_DIR,'templates')],
        'APP_DIRS': True,
        'OPTIONS': {
            'context_processors': [
                'django.template.context_processors.debug',
                'django.template.context_processors.request',
                'django.contrib.auth.context_processors.auth',
                'django.contrib.messages.context_processors.messages',
            ],
        },
    },
]

WSGI_APPLICATION = 'covidpred.wsgi.application'



# Password validation
# https://docs.djangoproject.com/en/3.0/ref/settings/#auth-password-validators

AUTH_PASSWORD_VALIDATORS = [
    {
        'NAME':
'django.contrib.auth.password_validation.UserAttributeSimilarityValidator',
    },
    {
        'NAME':
'django.contrib.auth.password_validation.MinimumLengthValidator',
    },
    {
        'NAME':
'django.contrib.auth.password_validation.CommonPasswordValidator',
    },
    {
        'NAME':
'django.contrib.auth.password_validation.NumericPasswordValidator',
    },
]


# Internationalization
# https://docs.djangoproject.com/en/3.0/topics/i18n/

LANGUAGE_CODE = 'en-us'

TIME_ZONE = 'UTC'

USE_I18N = True

USE_L10N = True

USE_TZ = True


# Static files (CSS, JavaScript, Images)
# https://docs.djangoproject.com/en/3.0/howto/static-files/

STATIC_URL = '/static/'
```

asgi.py

```
"""
ASGI config for covidpred project.

It exposes the ASGI callable as a module-level variable named ``application``.

For more information on this file, see
https://docs.djangoproject.com/en/3.0/howto/deployment/asgi/
"""

import os

from django.core.asgi import get_asgi_application

os.environ.setdefault('DJANGO_SETTINGS_MODULE', 'covidpred.settings')

application = get_asgi_application()
```

urls.py

```
"""covidpred URL Configuration

The `urlpatterns` list routes URLs to views. For more information please see:
    https://docs.djangoproject.com/en/3.0/topics/http/urls/
Examples:
Function views
    1. Add an import:  from my_app import views
    2. Add a URL to urlpatterns:  path('', views.home, name='home')
```

```
Class-based views
    1. Add an import:  from other_app.views import Home
    2. Add a URL to urlpatterns:  path('', Home.as_view(), name='home')
Including another URLconf
    1. Import the include() function: from django.urls import include, path
    2. Add a URL to urlpatterns:  path('blog/', include('blog.urls'))
"""
from django.contrib import admin
from django.urls import path
from django.urls import path, include
from django.conf.urls import url
from django.conf import settings

urlpatterns = [
          path('', include('covid.urls')),
          path('result', include('covid.urls')),
    path('admin/', admin.site.urls),
]
```

wsgi.py

```
"""
WSGI config for covidpred project.

It exposes the WSGI callable as a module-level variable named ``application``.

For more information on this file, see
https://docs.djangoproject.com/en/3.0/howto/deployment/wsgi/
"""
```

```
import os


from django.core.wsgi import get_wsgi_application


os.environ.setdefault('DJANGO_SETTINGS_MODULE', 'covidpred.settings')


application = get_wsgi_application()
```

apps.py

```
from django.apps import AppConfig



class CovidConfig(AppConfig):
    name = 'covid'
```

models.py

```
from django.db import models
from django.contrib.auth.models import User
from django.core.validators import MaxValueValidator,MinValueValidator
# Create your models here.
class Coviddata(models.Model):
    age= models.IntegerField(validators=[MinValueValidator(0),
MaxValueValidator(1)])
    gender= models.IntegerField(validators=[MinValueValidator(0),
MaxValueValidator(1)])
    fever= models.IntegerField(validators=[MinValueValidator(0),
MaxValueValidator(1)])
    cough= models.IntegerField(validators=[MinValueValidator(0),
MaxValueValidator(1)])
```

```
    fatigue= models.IntegerField(validators=[MinValueValidator(0),
MaxValueValidator(1)])

    pains= models.IntegerField(validators=[MinValueValidator(0),
MaxValueValidator(1)])

    nasal_congestion= models.IntegerField(validators=[MinValueValidator(0),
MaxValueValidator(1)])

    shortness_of_breath=
models.IntegerField(validators=[MinValueValidator(0),
MaxValueValidator(1)])

    runny_nose= models.IntegerField(validators=[MinValueValidator(0),
MaxValueValidator(1)])

    sore_throat= models.IntegerField(validators=[MinValueValidator(0),
MaxValueValidator(1)])

    diarrhea= models.IntegerField(validators=[MinValueValidator(0),
MaxValueValidator(1)])

    chills= models.IntegerField(validators=[MinValueValidator(0),
MaxValueValidator(1)])

    headache= models.IntegerField(validators=[MinValueValidator(0),
MaxValueValidator(1)])

    vomiting= models.IntegerField(validators=[MinValueValidator(0),
MaxValueValidator(1)])

    lives_in_affected_area=
models.IntegerField(validators=[MinValueValidator(0),
MaxValueValidator(1)])

    result= models.FloatField()

    created_at = models.DateTimeField(auto_now_add=True)
```

urls.py

```
from django.urls import path


from . import views
```

```
urlpatterns=[
        path('',views.index,name='index'),
        path('result',views.index,name='result'),

]
```

views.py

```
from django.shortcuts import render
from .models import Coviddata
import pickle
import numpy as np
import pandas as pd

import json
def index(request):

    if request.method == 'POST':
        full_name = request.POST['full_name']
        age = request.POST['age']
        gender = request.POST['gender']
        fever = request.POST['fever']
        cough = request.POST['cough']

        fatigue = request.POST.get('fatigue')
        fatigue = 1 if fatigue else 0
        pains = request.POST.get('pains')
        pains = 1 if pains else 0
```

```
nasal_congestion = request.POST.get('nasal_congestion')
nasal_congestion = 1 if nasal_congestion else 0
shortness_of_breath = request.POST.get('shortness_of_breath')
shortness_of_breath = 1 if shortness_of_breath else 0
runny_nose = request.POST.get('runny_nose')
runny_nose = 1 if runny_nose else 0
sore_throat = request.POST.get('sore_throat')
sore_throat = 1 if sore_throat else 0
diarrhea = request.POST.get('diarrhea')
diarrhea = 1 if diarrhea else 0
chills = request.POST.get('chills')
chills = 1 if chills else 0
headache =request.POST.get('headache')
headache = 1 if headache else 0
vomiting = request.POST.get('vomiting')
vomiting = 1 if vomiting else 0
lives_in_affected_area = request.POST['lives_in_affected_area']
file = open("model.pkl", "rb")
classifier = pickle.load(file)
file.close()
user_data = np.array(
    (age,
     gender,
     lives_in_affected_area,
     fever,
     cough,
     fatigue,
     pains,
     nasal_congestion,
     shortness_of_breath,
     runny_nose,
     sore_throat,
     diarrhea,
     chills,
     headache,
     vomiting
     )
).reshape(1, 15) # 1 row and 15 cols.

result = classifier.predict_proba(user_data)
result=round(result[0][1]*100,2)
print(f"Result: {result}")



age = int(user_data[0][0])
gender = 'Male' if int(user_data[0][1]) else 'Female'
lives_in_affected_area ='Yes' if int(user_data[0][2]) else 'No'
fever ='Yes' if int(user_data[0][3]) else 'No'
cough ='Yes' if int(user_data[0][4]) else 'No'
fatigue ='Yes' if int(user_data[0][5]) else 'No'
pains = 'Yes' if int(user_data[0][6]) else 'No'
nasal_congestion='Yes' if int(user_data[0][7]) else 'No'
shortness_of_breath = 'Yes' if int(user_data[0][8]) else 'No'
runny_nose = 'Yes' if int(user_data[0][9]) else 'No'
```

```
        sore_throat = 'Yes' if int(user_data[0][10]) else 'No'
        diarrhea = 'Yes' if int(user_data[0][11]) else 'No'
        chills = 'Yes' if int(user_data[0][12]) else 'No'
        headache = 'Yes' if int(user_data[0][13]) else 'No'
        vomiting = 'Yes' if int(user_data[0][14]) else 'No'

        user_details_API = {
            'Name':full_name,
            'Age':age,
            'Gender':gender,
            'Living in Affected Area':lives_in_affected_area,
            'Fever':fever,
            'Dry Cough':cough,
            'Fatigue':fatigue,
            'Pains':pains,
            'Nasal Congestion':nasal_congestion,
            'Problem in Breathing':shortness_of_breath,
            'Runny Nose':runny_nose,
            'Sore Throat':sore_throat,
            'Diarrhea':diarrhea,
            'Chills':chills,
            'Headache':headache,
            'Vomiting':vomiting
        }

        return
render(request,"result.html",{'result':result,'user_details_API':user_details_API,
                'user_json_data':json.dumps(user_details_API)})
```

```
    else:
        return render(request,"index.html")
```

Notebook.ipynb

```
{
 "cells": [
  {
   "cell_type": "markdown",
   "metadata": {},
   "source": [
    "### Importing the packages"
   ]
  },
  {
   "cell_type": "code",
   "execution_count": 1,
   "metadata": {},
   "outputs": [],
   "source": [
    "import numpy as np\n",
    "import pandas as pd\n",
    "import matplotlib.pyplot as plt\n",
    "from sklearn.metrics import accuracy_score,plot_roc_curve\n",
    "%matplotlib inline\n",
    "import warnings\n",
    "warnings.simplefilter('ignore')\n",
    "from sklearn.linear_model import LogisticRegression\n",
    "from sklearn.ensemble import RandomForestClassifier\n",
    "from sklearn.tree import DecisionTreeClassifier\n",
    "from sklearn.naive_bayes import GaussianNB\n",
    "from sklearn.svm import LinearSVC\n"
   ]
  },
  {
   "cell_type": "markdown",
   "metadata": {},
   "source": [
    "### Importing the dataset"
   ]
  },
  {
   "cell_type": "code",
   "execution_count": 2,
   "metadata": {},
   "outputs": [],
   "source": [
    "#Reading the DataSet\n",
    "df = pd.read_csv('covid_final_data.csv')"
   ]
  },
  {
   "cell_type": "markdown",
   "metadata": {},
   "source": [
    "### Dataset Attributes\n"
  ]
 },
 {
  "cell_type": "code",
  "execution_count": 3,
  "metadata": {},
  "outputs": [
   {
    "data": {
     "text/plain": [
      "Index(['AGE', 'GENDER', 'FEVER', 'COUGH', 'FATIGUE', 'PAINS',\n",
      "       'NASAL_CONGESTION', 'SHORTNESS_OF_BREATH', 'RUNNY_NOSE', 'SORE THROAT',\n",
      "       'DIARRHEA', 'CHILLS', 'HEADACHE', 'VOMITING', 'LIVES_IN_AFFECTED_AREA',\n",
      "       'COVID_OUTPUT'],\n",
      "      dtype='object')"
     ]
    },
    "execution_count": 3,
    "metadata": {},
    "output_type": "execute_result"
   }
  ],
  "source": [
   "df.columns"
  ]
```

````
},
{
 "cell_type": "markdown",
 "metadata": {},
 "source": [
  "**Patient's Attribute**\n",
  "\n",
  "```\n",
  "1. Age: Patient's Age\n",
  "2. Gender: Male(1) or Female (0)\n",
  "3. Fever: Yes(1) or No(0)\n",
  "4. Cough: Yes(1) or No(0)\n",
  "5. Fatigue: Yes(1) or No(0)\n",
  "6. Pains: Yes(1) or No(0)\n",
  "7. Nasal Congestion: Yes(1) or No(0)\n",
  "8. Shortness of Breath: Yes(1) or No(0)\n",
  "9. Runny Nose: Yes(1) or No(0)\n",
  "10. Sore Throat: Yes(1) or No(0)\n",
  "11. Diarrhea: Yes(1) or No(0)\n",
  "12. Chills: Yes(1) or No(0)\n",
  "13. Headache: Yes(1) or No(0)\n",
  "14. Vomiting: Yes(1) or No(0)\n",
  "15. Lives in affected area: Yes(1) or No(0)\n",
  "--------------------------------------------\n",
  "16. Covid Output: Yes(1) or No(0)\n",
  "```"
 ]
````

```
},
{
 "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>AGE</th>\n",
"      <th>GENDER</th>\n",
"      <th>FEVER</th>\n",
"      <th>COUGH</th>\n",
"      <th>FATIGUE</th>\n",
"      <th>PAINS</th>\n",
"      <th>NASAL_CONGESTION</th>\n",
"      <th>SHORTNESS_OF_BREATH</th>\n",
"      <th>RUNNY_NOSE</th>\n",
"      <th>SORE THROAT</th>\n",
"      <th>DIARRHEA</th>\n",
"      <th>CHILLS</th>\n",
"      <th>HEADACHE</th>\n",
"      <th>VOMITING</th>\n",
"      <th>LIVES_IN_AFFECTED_AREA</th>\n",
"      <th>COVID_OUTPUT</th>\n",
"    </tr>\n",
"  </thead>\n",
"  <tbody>\n",
"    <tr>\n",
"      <th>0</th>\n",
"      <td>19</td>\n",
"      <td>0</td>\n",
"      <td>1</td>\n",
"      <td>1</td>\n",
"      <td>0</td>\n",
"      <td>0</td>\n",
"     <td>0</td>\n",
"     <td>0</td>\n",
"     <td>0</td>\n",
"     <td>1</td>\n",
"     <td>0</td>\n",
"     <td>0</td>\n",
"     <td>1</td>\n",
"     <td>0</td>\n",
"     <td>0</td>\n",
"     <td>1</td>\n",
"   </tr>\n",
"   <tr>\n",
"     <th>1</th>\n",
"     <td>28</td>\n",
"     <td>1</td>\n",
"     <td>1</td>\n",
"     <td>0</td>\n",
"     <td>0</td>\n",
"     <td>0</td>\n",
"     <td>0</td>\n",
"     <td>0</td>\n",
"     <td>0</td>\n",
"     <td>0</td>\n",
"     <td>0</td>\n",
"     <td>0</td>\n",
"     <td>1</td>\n",
"     <td>0</td>\n",
"      <td>1</td>\n",
"      <td>1</td>\n",
"    </tr>\n",
"    <tr>\n",
"      <th>2</th>\n",
"      <td>35</td>\n",
"      <td>0</td>\n",
"      <td>1</td>\n",
"      <td>1</td>\n",
"      <td>0</td>\n",
"      <td>0</td>\n",
"      <td>0</td>\n",
"      <td>0</td>\n",
"      <td>0</td>\n",
"      <td>0</td>\n",
"      <td>0</td>\n",
"      <td>0</td>\n",
"      <td>1</td>\n",
"      <td>0</td>\n",
"      <td>0</td>\n",
"      <td>1</td>\n",
"    </tr>\n",
"    <tr>\n",
"      <th>3</th>\n",
"      <td>33</td>\n",
"      <td>0</td>\n",
"      <td>1</td>\n",
"      <td>0</td>\n",
"      <td>0</td>\n",
"      <td>0</td>\n",
"      <td>0</td>\n",
"      <td>0</td>\n",
"      <td>0</td>\n",
"      <td>0</td>\n",
"      <td>0</td>\n",
"      <td>0</td>\n",
"      <td>1</td>\n",
"      <td>0</td>\n",
"      <td>0</td>\n",
"      <td>1</td>\n",
"    </tr>\n",
"    <tr>\n",
"      <th>4</th>\n",
"      <td>33</td>\n",
"      <td>0</td>\n",
"      <td>1</td>\n",
"      <td>0</td>\n",
"      <td>0</td>\n",
"      <td>0</td>\n",
"      <td>0</td>\n",
"      <td>0</td>\n",
"      <td>0</td>\n",
"      <td>0</td>\n",
"      <td>0</td>\n",
"      <td>0</td>\n",
"      <td>1</td>\n",
"      <td>0</td>\n",
"      <td>0</td>\n",
"      <td>1</td>\n",
"    </tr>\n",
"    <tr>\n",
"      <th>5</th>\n",
"      <td>87</td>\n",
"      <td>0</td>\n",
"      <td>1</td>\n",
"      <td>0</td>\n",
"      <td>1</td>\n",
"      <td>0</td>\n",
"      <td>0</td>\n",
"      <td>0</td>\n",
"      <td>0</td>\n",
"      <td>0</td>\n",
"      <td>0</td>\n",
"      <td>0</td>\n",
"      <td>1</td>\n",
"      <td>0</td>\n",
"      <td>0</td>\n",
"      <td>1</td>\n",
"    </tr>\n",
"    <tr>\n",
"      <th>6</th>\n",
"      <td>55</td>\n",
"      <td>0</td>\n",
"      <td>1</td>\n",
"      <td>0</td>\n",
"      <td>0</td>\n",
"      <td>0</td>\n",
"      <td>0</td>\n",
"      <td>0</td>\n",
"      <td>1</td>\n",
"      <td>0</td>\n",
"      <td>0</td>\n",
"      <td>0</td>\n",
"      <td>1</td>\n",
"      <td>0</td>\n",
"      <td>0</td>\n",
"      <td>1</td>\n",
"    </tr>\n",
"    <tr>\n",
"      <th>7</th>\n",
"      <td>60</td>\n",
"      <td>0</td>\n",
"      <td>1</td>\n",
"      <td>0</td>\n",
"      <td>0</td>\n",
"      <td>0</td>\n",
"      <td>0</td>\n",
"      <td>0</td>\n",
"     <td>0</td>\n",
"     <td>0</td>\n",
"     <td>0</td>\n",
"     <td>0</td>\n",
"     <td>1</td>\n",
"     <td>0</td>\n",
"     <td>0</td>\n",
"     <td>1</td>\n",
"    </tr>\n",
"    <tr>\n",
"     <th>8</th>\n",
"     <td>66</td>\n",
"     <td>1</td>\n",
"     <td>1</td>\n",
"     <td>1</td>\n",
"     <td>1</td>\n",
"     <td>0</td>\n",
"     <td>0</td>\n",
"     <td>0</td>\n",
"     <td>0</td>\n",
"     <td>0</td>\n",
"     <td>0</td>\n",
"     <td>0</td>\n",
"     <td>1</td>\n",
"     <td>0</td>\n",
"     <td>0</td>\n",
"     <td>1</td>\n",
     "   </tr>\n",
     "   <tr>\n",
     "     <th>9</th>\n",
     "     <td>57</td>\n",
     "     <td>1</td>\n",
     "     <td>1</td>\n",
     "     <td>0</td>\n",
     "     <td>0</td>\n",
     "     <td>0</td>\n",
     "     <td>0</td>\n",
     "     <td>0</td>\n",
     "     <td>0</td>\n",
     "     <td>1</td>\n",
     "     <td>0</td>\n",
     "     <td>0</td>\n",
     "     <td>1</td>\n",
     "     <td>0</td>\n",
     "     <td>0</td>\n",
     "     <td>1</td>\n",
     "   </tr>\n",
     "  </tbody>\n",
     "</table>\n",
     "</div>"
    ],
    "text/plain": [
     "   AGE  GENDER  FEVER  COUGH  FATIGUE  PAINS
NASAL_CONGESTION  \\\n",
     "0   19      0      1      1       0      0                 0  \n",
"1  28    1    1    0     0    0          0  \n",
"2  35    0    1    1     0    0          0  \n",
"3  33    0    1    0     0    0          0  \n",
"4  33    0    1    0     0    0          0  \n",
"5  87    0    1    0     1    0          0  \n",
"6  55    0    1    0     0    0          0  \n",
"7  60    0    1    0     0    0          0  \n",
"8  66    1    1    1     1    0          0  \n",
"9  57    1    1    0     0    0          0  \n",
"\n",
"   SHORTNESS_OF_BREATH  RUNNY_NOSE  SORE THROAT  DIARRHEA  CHILLS  HEADACHE  \\\n",
"0            0       0        1      0     0      1   \n",
"1            0       0        0      0     0      1   \n",
"2            0       0        0      0     0      1   \n",
"3            0       0        0      0     0      1   \n",
"4            0       0        0      0     0      1   \n",
"5            0       0        0      0     0      1   \n",
"6            0       1        0      0     0      1   \n",
"7            0       0        0      0     0      1   \n",
"8            0       0        0      0     0      1   \n",
"9            0       0        1      0     0      1   \n",
"\n",
"   VOMITING  LIVES_IN_AFFECTED_AREA  COVID_OUTPUT  \n",
"0      0              0        1  \n",
"1      0              1        1  \n",
"2      0              0        1  \n",
     "3        0                    0           1  \n",
     "4        0                    0           1  \n",
     "5        0                    0           1  \n",
     "6        0                    0           1  \n",
     "7        0                    0           1  \n",
     "8        0                    0           1  \n",
     "9        0                    0           1  "
    ]
   },
   "execution_count": 4,
   "metadata": {},
   "output_type": "execute_result"
  }
 ],
 "source": [
  "df.head(10)"
 ]
},
{
 "cell_type": "markdown",
 "metadata": {},
 "source": [
  "### Correlation Matrix"
 ]
},
{
 "cell_type": "code",
"execution_count": 5,
"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>AGE</th>\n",
   "      <th>GENDER</th>\n",
   "      <th>FEVER</th>\n",
"      <th>COUGH</th>\n",
"      <th>FATIGUE</th>\n",
"      <th>PAINS</th>\n",
"      <th>NASAL_CONGESTION</th>\n",
"      <th>SHORTNESS_OF_BREATH</th>\n",
"      <th>RUNNY_NOSE</th>\n",
"      <th>SORE THROAT</th>\n",
"      <th>DIARRHEA</th>\n",
"      <th>CHILLS</th>\n",
"      <th>HEADACHE</th>\n",
"      <th>VOMITING</th>\n",
"      <th>LIVES_IN_AFFECTED_AREA</th>\n",
"      <th>COVID_OUTPUT</th>\n",
"    </tr>\n",
"  </thead>\n",
"  <tbody>\n",
"    <tr>\n",
"      <th>AGE</th>\n",
"      <td>1.000000</td>\n",
"      <td>0.074382</td>\n",
"      <td>0.022943</td>\n",
"      <td>0.047645</td>\n",
"      <td>0.097993</td>\n",
"      <td>0.081185</td>\n",
"      <td>0.130204</td>\n",
"      <td>-0.031658</td>\n",
"      <td>-0.021785</td>\n",
"      <td>-0.105755</td>\n",
"      <td>-0.048191</td>\n",
"      <td>0.023023</td>\n",
"      <td>0.016395</td>\n",
"      <td>0.022586</td>\n",
"      <td>-0.000310</td>\n",
"      <td>0.006081</td>\n",
"    </tr>\n",
"    <tr>\n",
"      <th>GENDER</th>\n",
"      <td>0.074382</td>\n",
"      <td>1.000000</td>\n",
"      <td>0.053795</td>\n",
"      <td>0.003112</td>\n",
"      <td>-0.048996</td>\n",
"      <td>-0.010474</td>\n",
"      <td>0.011494</td>\n",
"      <td>-0.016308</td>\n",
"      <td>-0.052529</td>\n",
"      <td>-0.062823</td>\n",
"      <td>0.017050</td>\n",
"      <td>-0.014599</td>\n",
"      <td>-0.022730</td>\n",
"      <td>0.025970</td>\n",
"      <td>0.212699</td>\n",
"      <td>0.013788</td>\n",
"    </tr>\n",
"    <tr>\n",
"      <th>FEVER</th>\n",
"      <td>0.022943</td>\n",
"      <td>0.053795</td>\n",
"      <td>1.000000</td>\n",
"      <td>0.191755</td>\n",
"      <td>-0.005902</td>\n",
"      <td>0.060148</td>\n",
"      <td>0.004159</td>\n",
"      <td>-0.123018</td>\n",
"      <td>0.010423</td>\n",
"      <td>0.001083</td>\n",
"      <td>0.009738</td>\n",
"      <td>-0.017796</td>\n",
"      <td>-0.068020</td>\n",
"      <td>-0.020995</td>\n",
"      <td>0.187202</td>\n",
"      <td>0.529185</td>\n",
"    </tr>\n",
"    <tr>\n",
"      <th>COUGH</th>\n",
"      <td>0.047645</td>\n",
"      <td>0.003112</td>\n",
"      <td>0.191755</td>\n",
"      <td>1.000000</td>\n",
"      <td>-0.027305</td>\n",
"      <td>0.052422</td>\n",
"      <td>0.070611</td>\n",
"      <td>-0.080566</td>\n",
"      <td>0.029718</td>\n",
"      <td>0.019849</td>\n",
"      <td>0.020887</td>\n",
"      <td>-0.030242</td>\n",
"      <td>0.056511</td>\n",
"      <td>0.014351</td>\n",
"      <td>-0.030356</td>\n",
"      <td>0.218632</td>\n",
"    </tr>\n",
"    <tr>\n",
"      <th>FATIGUE</th>\n",
"      <td>0.097993</td>\n",
"      <td>-0.048996</td>\n",
"      <td>-0.005902</td>\n",
"      <td>-0.027305</td>\n",
"      <td>1.000000</td>\n",
"      <td>0.015797</td>\n",
"      <td>-0.007595</td>\n",
"      <td>-0.045449</td>\n",
"      <td>-0.029837</td>\n",
"      <td>0.010381</td>\n",
"      <td>0.037218</td>\n",
"      <td>0.033071</td>\n",
"      <td>0.007328</td>\n",
"      <td>0.008849</td>\n",
"      <td>-0.016685</td>\n",
"      <td>0.070014</td>\n",
"    </tr>\n",
"    <tr>\n",
"      <th>PAINS</th>\n",
"      <td>0.081185</td>\n",
"      <td>-0.010474</td>\n",
"      <td>0.060148</td>\n",
"      <td>0.052422</td>\n",
"      <td>0.015797</td>\n",
"      <td>1.000000</td>\n",
"      <td>0.040673</td>\n",
"      <td>0.014355</td>\n",
"      <td>0.031633</td>\n",
"      <td>-0.009506</td>\n",
"      <td>0.023900</td>\n",
"      <td>-0.034958</td>\n",
"      <td>-0.030098</td>\n",
"      <td>-0.005686</td>\n",
"      <td>-0.037240</td>\n",
"      <td>0.057166</td>\n",
"    </tr>\n",
"    <tr>\n",
"      <th>NASAL_CONGESTION</th>\n",
"      <td>0.130204</td>\n",
"      <td>0.011494</td>\n",
"      <td>0.004159</td>\n",
"      <td>0.070611</td>\n",
"      <td>-0.007595</td>\n",
"      <td>0.040673</td>\n",
"      <td>1.000000</td>\n",
"      <td>-0.042787</td>\n",
"      <td>-0.043125</td>\n",
"      <td>-0.040672</td>\n",
"      <td>0.042957</td>\n",
"      <td>0.020311</td>\n",
"      <td>-0.048444</td>\n",
"      <td>-0.009234</td>\n",
"      <td>0.066466</td>\n",
"      <td>0.077812</td>\n",
"    </tr>\n",
"    <tr>\n",
"      <th>SHORTNESS_OF_BREATH</th>\n",
"      <td>-0.031658</td>\n",
"      <td>-0.016308</td>\n",
"      <td>-0.123018</td>\n",
"      <td>-0.080566</td>\n",
"      <td>-0.045449</td>\n",
"      <td>0.014355</td>\n",
"      <td>-0.042787</td>\n",
"      <td>1.000000</td>\n",
"      <td>-0.046551</td>\n",
"      <td>-0.047411</td>\n",
"      <td>-0.009285</td>\n",
"      <td>-0.022450</td>\n",
"      <td>-0.093514</td>\n",
"      <td>-0.034236</td>\n",
"      <td>0.024829</td>\n",
"      <td>0.016807</td>\n",
"    </tr>\n",
"    <tr>\n",
"      <th>RUNNY_NOSE</th>\n",
"      <td>-0.021785</td>\n",
"      <td>-0.052529</td>\n",
"      <td>0.010423</td>\n",
"      <td>0.029718</td>\n",
"      <td>-0.029837</td>\n",
"      <td>0.031633</td>\n",
"      <td>-0.043125</td>\n",
"      <td>-0.046551</td>\n",
"      <td>1.000000</td>\n",
"      <td>0.062148</td>\n",
"      <td>0.146668</td>\n",
"      <td>-0.040708</td>\n",
"      <td>-0.005887</td>\n",
"      <td>0.007113</td>\n",
"      <td>-0.017605</td>\n",
"      <td>0.013413</td>\n",
"    </tr>\n",
"    <tr>\n",
"      <th>SORE THROAT</th>\n",
"      <td>-0.105755</td>\n",
"      <td>-0.062823</td>\n",
"      <td>0.001083</td>\n",
"      <td>0.019849</td>\n",
"      <td>0.010381</td>\n",
"      <td>-0.009506</td>\n",
"      <td>-0.040672</td>\n",
"      <td>-0.047411</td>\n",
"      <td>0.062148</td>\n",
"      <td>1.000000</td>\n",
"      <td>0.015141</td>\n",
"      <td>0.064801</td>\n",
"      <td>0.052343</td>\n",
"      <td>0.065870</td>\n",
"      <td>-0.030366</td>\n",
"      <td>-0.008681</td>\n",
"    </tr>\n",
"    <tr>\n",
"      <th>DIARRHEA</th>\n",
"      <td>-0.048191</td>\n",
"      <td>0.017050</td>\n",
"      <td>0.009738</td>\n",
"      <td>0.020887</td>\n",
"      <td>0.037218</td>\n",
"      <td>0.023900</td>\n",
"      <td>0.042957</td>\n",
"      <td>-0.009285</td>\n",
"      <td>0.146668</td>\n",
"      <td>0.015141</td>\n",
"      <td>1.000000</td>\n",
"      <td>0.048589</td>\n",
"      <td>-0.026456</td>\n",
"      <td>0.125182</td>\n",
"      <td>0.021379</td>\n",
"      <td>0.044341</td>\n",
"    </tr>\n",
"    <tr>\n",
"      <th>CHILLS</th>\n",
"      <td>0.023023</td>\n",
"      <td>-0.014599</td>\n",
"      <td>-0.017796</td>\n",
"      <td>-0.030242</td>\n",
"      <td>0.033071</td>\n",
"      <td>-0.034958</td>\n",
"      <td>0.020311</td>\n",
"      <td>-0.022450</td>\n",
"      <td>-0.040708</td>\n",
"      <td>0.064801</td>\n",
"      <td>0.048589</td>\n",
"      <td>1.000000</td>\n",
"      <td>0.049161</td>\n",
"      <td>0.017500</td>\n",
"      <td>-0.009769</td>\n",
"      <td>0.016038</td>\n",
"    </tr>\n",
"    <tr>\n",
"      <th>HEADACHE</th>\n",
"      <td>0.016395</td>\n",
"      <td>-0.022730</td>\n",
"      <td>-0.068020</td>\n",
"      <td>0.056511</td>\n",
"      <td>0.007328</td>\n",
"      <td>-0.030098</td>\n",
"      <td>-0.048444</td>\n",
"      <td>-0.093514</td>\n",
"      <td>-0.005887</td>\n",
"      <td>0.052343</td>\n",
"      <td>-0.026456</td>\n",
"      <td>0.049161</td>\n",
"      <td>1.000000</td>\n",
"      <td>0.083667</td>\n",
"      <td>-0.282833</td>\n",
"      <td>-0.272482</td>\n",
"    </tr>\n",
"    <tr>\n",
"      <th>VOMITING</th>\n",
"      <td>0.022586</td>\n",
"      <td>0.025970</td>\n",
"      <td>-0.020995</td>\n",
"      <td>0.014351</td>\n",
"      <td>0.008849</td>\n",
"     <td>-0.005686</td>\n",
"     <td>-0.009234</td>\n",
"     <td>-0.034236</td>\n",
"     <td>0.007113</td>\n",
"     <td>0.065870</td>\n",
"     <td>0.125182</td>\n",
"     <td>0.017500</td>\n",
"     <td>0.083667</td>\n",
"     <td>1.000000</td>\n",
"     <td>-0.006384</td>\n",
"     <td>-0.066612</td>\n",
"    </tr>\n",
"    <tr>\n",
"     <th>LIVES_IN_AFFECTED_AREA</th>\n",
"     <td>-0.000310</td>\n",
"     <td>0.212699</td>\n",
"     <td>0.187202</td>\n",
"     <td>-0.030356</td>\n",
"     <td>-0.016685</td>\n",
"     <td>-0.037240</td>\n",
"     <td>0.066466</td>\n",
"     <td>0.024829</td>\n",
"     <td>-0.017605</td>\n",
"     <td>-0.030366</td>\n",
"     <td>0.021379</td>\n",
"     <td>-0.009769</td>\n",
"     <td>-0.282833</td>\n",
   "      <td>-0.006384</td>\n",
   "      <td>1.000000</td>\n",
   "      <td>0.302627</td>\n",
   "    </tr>\n",
   "    <tr>\n",
   "      <th>COVID_OUTPUT</th>\n",
   "      <td>0.006081</td>\n",
   "      <td>0.013788</td>\n",
   "      <td>0.529185</td>\n",
   "      <td>0.218632</td>\n",
   "      <td>0.070014</td>\n",
   "      <td>0.057166</td>\n",
   "      <td>0.077812</td>\n",
   "      <td>0.016807</td>\n",
   "      <td>0.013413</td>\n",
   "      <td>-0.008681</td>\n",
   "      <td>0.044341</td>\n",
   "      <td>0.016038</td>\n",
   "      <td>-0.272482</td>\n",
   "      <td>-0.066612</td>\n",
   "      <td>0.302627</td>\n",
   "      <td>1.000000</td>\n",
   "    </tr>\n",
   "  </tbody>\n",
   "</table>\n",
   "</div>"
  ],
    "text/plain": [
    "                      AGE    GENDER     FEVER     COUGH   FATIGUE
\\\n",
    "AGE                 1.000000  0.074382  0.022943  0.047645  0.097993
\n",
    "GENDER                0.074382  1.000000  0.053795  0.003112 -0.048996
\n",
    "FEVER                0.022943  0.053795  1.000000  0.191755 -0.005902
\n",
    "COUGH                0.047645  0.003112  0.191755  1.000000 -0.027305
\n",
    "FATIGUE              0.097993 -0.048996 -0.005902 -0.027305  1.000000
\n",
    "PAINS               0.081185 -0.010474  0.060148  0.052422  0.015797
\n",
    "NASAL_CONGESTION       0.130204  0.011494  0.004159  0.070611 -
0.007595   \n",
    "SHORTNESS_OF_BREATH    -0.031658 -0.016308 -0.123018 -
0.080566 -0.045449   \n",
    "RUNNY_NOSE           -0.021785 -0.052529  0.010423  0.029718 -
0.029837   \n",
    "SORE THROAT          -0.105755 -0.062823  0.001083  0.019849
0.010381   \n",
    "DIARRHEA            -0.048191  0.017050  0.009738  0.020887
0.037218   \n",
    "CHILLS               0.023023 -0.014599 -0.017796 -0.030242  0.033071
\n",
    "HEADACHE              0.016395 -0.022730 -0.068020  0.056511
0.007328   \n",
    "VOMITING              0.022586  0.025970 -0.020995  0.014351
0.008849   \n",
    "LIVES_IN_AFFECTED_AREA -0.000310  0.212699  0.187202 -
0.030356 -0.016685   \n",

    "COVID_OUTPUT          0.006081  0.013788  0.529185  0.218632
0.070014   \n",

    "\n",

    "                      PAINS  NASAL_CONGESTION
SHORTNESS_OF_BREATH  \\\n",

    "AGE                0.081185        0.130204          -0.031658   \n",

    "GENDER               -0.010474        0.011494          -0.016308   \n",

    "FEVER              0.060148        0.004159          -0.123018   \n",

    "COUGH              0.052422        0.070611          -0.080566   \n",

    "FATIGUE            0.015797       -0.007595          -0.045449   \n",

    "PAINS              1.000000        0.040673           0.014355   \n",

    "NASAL_CONGESTION       0.040673        1.000000          -0.042787
\n",

    "SHORTNESS_OF_BREATH    0.014355       -0.042787
1.000000   \n",

    "RUNNY_NOSE           0.031633       -0.043125          -0.046551   \n",

    "SORE THROAT          -0.009506       -0.040672          -0.047411   \n",

    "DIARRHEA             0.023900        0.042957          -0.009285   \n",

    "CHILLS              -0.034958        0.020311          -0.022450   \n",

    "HEADACHE             -0.030098       -0.048444          -0.093514   \n",

    "VOMITING             -0.005686       -0.009234          -0.034236   \n",

    "LIVES_IN_AFFECTED_AREA -0.037240        0.066466
0.024829   \n",

    "COVID_OUTPUT          0.057166        0.077812           0.016807
\n",

    "\n",

    "                    RUNNY_NOSE  SORE THROAT  DIARRHEA
CHILLS  HEADACHE  \\\n",
    "AGE                   -0.021785    -0.105755 -0.048191  0.023023  0.016395
\n",
    "GENDER                -0.052529    -0.062823  0.017050 -0.014599 -
0.022730   \n",
    "FEVER                 0.010423     0.001083  0.009738 -0.017796 -
0.068020   \n",
    "COUGH                 0.029718     0.019849  0.020887 -0.030242
0.056511   \n",
    "FATIGUE               -0.029837     0.010381  0.037218  0.033071
0.007328   \n",
    "PAINS                 0.031633    -0.009506  0.023900 -0.034958 -
0.030098   \n",
    "NASAL_CONGESTION        -0.043125    -0.040672  0.042957
0.020311 -0.048444   \n",
    "SHORTNESS_OF_BREATH      -0.046551    -0.047411 -0.009285 -
0.022450 -0.093514   \n",
    "RUNNY_NOSE             1.000000     0.062148  0.146668 -0.040708 -
0.005887   \n",
    "SORE THROAT            0.062148     1.000000  0.015141  0.064801
0.052343   \n",
    "DIARRHEA               0.146668     0.015141  1.000000  0.048589 -
0.026456   \n",
    "CHILLS                -0.040708     0.064801  0.048589  1.000000
0.049161   \n",
    "HEADACHE              -0.005887     0.052343 -0.026456  0.049161
1.000000   \n",
    "VOMITING               0.007113     0.065870  0.125182  0.017500
0.083667   \n",
    "LIVES_IN_AFFECTED_AREA   -0.017605    -0.030366  0.021379 -
0.009769 -0.282833   \n",
    "COVID_OUTPUT            0.013413    -0.008681  0.044341  0.016038 -
0.272482   \n",
    "\n",
    "                   VOMITING  LIVES_IN_AFFECTED_AREA
COVID_OUTPUT  \n",
    "AGE                0.022586           -0.000310     0.006081  \n",
    "GENDER             0.025970            0.212699     0.013788  \n",
    "FEVER             -0.020995            0.187202     0.529185  \n",
    "COUGH              0.014351           -0.030356     0.218632  \n",
    "FATIGUE            0.008849           -0.016685     0.070014  \n",
    "PAINS             -0.005686           -0.037240     0.057166  \n",
    "NASAL_CONGESTION      -0.009234            0.066466     0.077812
\n",
    "SHORTNESS_OF_BREATH   -0.034236            0.024829
0.016807  \n",
    "RUNNY_NOSE         0.007113           -0.017605     0.013413  \n",
    "SORE THROAT        0.065870           -0.030366    -0.008681  \n",
    "DIARRHEA           0.125182            0.021379     0.044341  \n",
    "CHILLS             0.017500           -0.009769     0.016038  \n",
    "HEADACHE           0.083667           -0.282833    -0.272482  \n",
    "VOMITING           1.000000           -0.006384    -0.066612  \n",
    "LIVES_IN_AFFECTED_AREA -0.006384            1.000000
0.302627  \n",
    "COVID_OUTPUT      -0.066612            0.302627     1.000000  "
   ]
  },
   "execution_count": 5,
   "metadata": {},
   "output_type": "execute_result"
  }
 ],
   "source": [
    "corrMatrix = df.corr(method='spearman')\n",
    "df.corr()"
   ]
  },
  {
   "cell_type": "markdown",
   "metadata": {},
   "source": [
    "### Visual Representation of Correlation Matrix"
   ]
  },
  {
   "cell_type": "code",
   "execution_count": 6,
   "metadata": {},
   "outputs": [
    {
     "data": {
      "image/png":
"iVBORw0KGgoAAAANSUhEUgAAAd8AAAF2CAYAAAAvE5J+AAAABH
NCSVQICAgIfAhkiAAAAAlwSFlzAAALEgAACxIB0t1+/AAAADh0RVh0U2
9mdHdhcmUAbWF0cGxvdGxpYiB2ZXJzaW9uMy4yLjEsIGh0dHA6Ly9tYX
RwbG90bGliLm9yZy+j8jraAAAgAElEQVR4nOy9eZxcVZn///4kEAgksgmIBA
lgRAFZo/BVUDYRHRxAUIi44IwiM+CCIjjCjJERRcFhd4kb4o9VFkGGERT
IAApCgLCEHcISFtlGCbIm/fn9cU4lN5Wqrlt9b7qrO8+b131x71mec253pZ86
2/ORbYIgCIIgGDxGDXUHgiAIgmBpI5xvEARBEAwy4XyDIAiCYJAJ5xsE
QRAEg0w43yAIgiAYZML5BkEQBMEgE843CIIgWGqR9HNJT0m6o02+JJ0
k6X5Jt0naso52w/kGQRAEQTDIhPMNgiAIgkEmnG8QBEEQDDLhfIMgCIJgkAnnG
wPP9VNkd+B0J64HVpa0VtV2w/kGQRAEwSATzjcIgiAIBplwvkEQBEEwyITzDY
7ZkHK8UpPXTy1yv34Tm/Wqn+8hpduQ/jqW6jjoCvK7laP67r628gUI4xNfw8
```

n3jtb5VtrL7MuEr1Xz9qbOU+vEpfZRtjahgr1dGPOjjn4d+oSv1u/t6MWX2Dz
5GmixtMsz2ti+Za9bXyP9NwvkEQBMHwom9+6aLZ0XbjbJuZA6xTeJ4APF
7BHhDTzkEQBMFww33lr+pcDHwy73reBvib7SeqGo2R7whH0p7ABcDbbN
+d094JfI+0bjEXeAL4mu3bJU0FPgs8XTCzve2/DmrHgyAI2tFX3/S5pLOA7Y
HXS5oDfANYFsD2j4BLgQ8C9wMvApH7uo91wviOfKcC1wL7AVElrAucCH7
P9JwBJ2wIbALfnOsfbPm4oOhsEQdAJ1zOizbY8pUO+gYNqazATzncEI2kc8
G5gB9LUyVTgYOCXDccLYPvaIelgEATBQJg/b6h7UJlY8x3Z7AH8zva9wH
M5MsvGwM0d6h0iaWa+rlrivQyCIGvvnlrx4lnO/IZgpwdr4/Oz8vgqQ/S7pL
0omF5ONtb56vHdoZl3SApBmSZvz09LPq7XkQBEE7BnfD1RIhpp1HKJJW
A3YENpFkYDTpbNovgS2BiwBsby1pb2C3btsobuGves43CIKgNDVuuBoqYu
Q7ctmbFBJtXdsTba8DzXdzAYuB/aX9K5C2RWGpIduUw
BjmlKOx/4GLAP8F1JawNPAc8ARxXKHSLp44XZn44XnPWw/tAT7GgRBUJ4RM
PIN5ztCsb19i7STCo/vbVNvKmlXdBAEQW8y/7Wh7kFlwvkGQRAEw4senk4
uSzjfoBaqCiMcN+Pblftw8OX+BqUFTPU/
DXMrCl0AjB+9fGUbq4yqZuOFGj4X82qQyxij6p+LnedXE5m4avTfK/ehFmL
aOQiCIAgGmRj5BkEQBMEgEyPfIAiCIBhc3Df8N1kcXdKaks6U9KCk
myRdJ2lPSdtL+lsh3u5MSR/IdR7OcXhvl/SLHMu3cFyUyivv6Srqq
S7pnoLN81rUu1NSv0HHgyAIBp2+V/JXjxLOtweQDgb9QmWBVcDzY217e9FUmFa
EIuck0h3OPmtv+Q018BPizp9W1MN8JETgLOAa6UtHohf7+Czb2b6wG7Az
+WtGxd7xoEQVCZERBeMpxvb7Aj8GrWjgTA9sO2T+5Qbx4pvOMhnRqwfQ
4putXHynbK9n0k/cpVytYJgiBY4oSwQlATnZSGtmuadt6gkHcqsJ+klUq4P
w1sLzGQWbxzYXzipI99l+qpWxorDCHXMfKNF8EARBDYyAkW9suOpBJJ0
KbAu8CnyVNO282xBpYOALr7u2M1YU
VvjixH1DWCEIgsGhh9dyyxIj395gFklpCpBtH0QKlLZ8yfrnkkIYj0lXfM9DOjTJVWPdBAEQVAX8+eVv3qUcL69wZXZ
WwF0l7B1ve0NSDOjTJVWPdBAEQVAX8+eVv3qUcL69wZXA8pL+pZBW
WmnI9nPAuSQH3BJJewG7AKWFd21fAMwAPlW2ThAEQe3EyDdYGthtkpbUFm
B7AeyXNlnQDSXe3ES+xec137xZmvg8073o+pHHUCPg4sKPtpwv5xfP9VfP9C
ao4AvS4rPShAEPYE9v/TVq8SabnFdRV28CNmxnJwiCYNDp4RFtWcL5BkEQBMOLHt5FXJZwvkEtVBVGeGn+q5X
qkeFeoVn28CNmxnJwiCYNDp4RFtWcL5BkEQBMOLHt7FXJZwvkEtPFdR
waaqIhHAKTO+W6n+j7f4j8p9mD6q+h+F1WvY31b19/Hgq89W7sNGy61R2
UYdakCuaKOvhj7M7Xu5so1RNSz8vKzSW0lacsqWz1XvRB3EyDcIgiAIBpke
3sVclnC+QRAEwfBiBEw7xw6nye/I4gnvb6r/JUk/yPVeaqrzy
VzmIUm3S7pN0kW9K/Wndo3jYIgqAFNR81kkrRrFpq5X9LXWuZvJOm3km6kbhmVNW1kh3wsDXtKqsPpWFE2hYcbtbHQKw/I0hJrqDsl+iQdc9HFdbjHpzz/WQoCdiPoRjM15b7KLr9F8zBPnnQyl3jwnL9C92j/mRLCaAV
EvSp6u+Qox8hx8vZdGDBUiaSIsoWDkM5b7AZYXkfUlRswAeaLZVYAfbz0j
6JnAk8Nka+h4EQVCdGtd8JY0mhel9HzAHuxbbvQ7CLjT9oeyOM0
9ks6wB765Ika+I5vzgN0kLQcLnPQbgWu7sHEdsHbtPQuCIBgo9cZ2fidwv+3UHAV1QGWwXZfb3MmEI3GnsvCNv5akG+DljyC1u89/Jt9ze2cfAa0c/VFgZpOkgp1xpHdTc99J/JRsTQs8y+2/O6/Ln
SAHbsCF5FGvefYdvoMsUFTnc/bvqapvV1JcodBEAS9Qb0brtYGHi08zqHiiQX8DgwHtjHrrbwHM53+LHYtHOmnfjCWSSn23C+/1TI62/a+SpJawJ
PkaadF0PSAcABAFuvujmTxq1X8hWCIAgq0MW0c/HvZ

```
bzZe8HZpLkXzcAfi/pGtvPl+5IEzHtPPL5DbBTlgcca7s/6cIiOwDrkkQfjmpVw
PY025NtTw7HGwTBoNHFtHPx71S+pjVZmwOsU3ieQBrhFvk0cIET9wOzW
VSetWvC+Y5wbL8ATAd+TheiCrnuS8CXgE9KWrX+3gVBEAyAenc73whMk
rSepDGkGcKLm8o8QlKaI88Ibgg8WOUVwvmOHPoTXzgL2Iy0kaDIBk11vt
BsNMecPou02y8IgmDoqdH52p4HHEw6FXIXcK7tWZIOlHRgLvafwLsk3Q5
cARxu+5kqrxBrvsOMophCIW06bcQXcv6FNK1r2H4IGNum/MSm589339M
gCIIlhKuH/FzUnC8FLm1K+1Hh/nGSJGtthPMNgiAIhhfzIrxkEACwvEZXq
v9Xv1a5D1WFET53S8t9ZV3xuy2rTxLMU/Vv9auno90D5vHRLSdFumJMxc
8EwOiaRzgDYX4NwgobLrNKZRtzXd3hXMb/Vao/7oZ1OhcqQeXwUCMgvG
Q43yAIgmB4EapGQRAEQTDI9MCMSFXC+QZBEATDixEw8o2jRj2MpDd
IOlvSA5LulHSppLdI2ljSlZLulXSfpH/PMUeRNFXSoU12HpL0+ny/pqQzJT0
o6SZJ10naM+dtL+mSprqnNR1bCoIgGFpqVjUaCsL59ijZmV4ITLe9ge2NgK
8Da5IOgB9j+y2k87vvAv61pM3fAFfbXt/2VqQD5ROW0GsEQRDUjufPL331
KuF8e5cdgNeazprNBN4C/NH25TntRdIB8cU0KFuwI/Bqk82HbZ9ca8+DIAi
WJDHyDZYgmwA3tUjfuDnd9gPAOEmv62BzY6BTbOdFImUB/9iuoKQDJM
2QNOPuuZUirQVBEJSnXknBISGc7/BDLK640cAd8hY1JJ0q6VZJNxaSr7G
9eeNi8RinCw0WApa/dfz6ZfsfBEFQjT6Xv3qUcL69yyxgqzbpk4sJktYHXrA9F
3gWaD7RPx74a667ZSPR9kGkYOGr19ftIAiCJUxMOwdLkCuB5SR9tpEg6R
3AfcC2knbOaWOBk4Dv5WJXA/8oaXzO/zBwq+/Ak7J9W6TdAp
mT+/PJXjxLnfHsU285HgE6Q9DXgZeAhksTf7sDJkk4FRgO/Ak7J9W6TdAp
wrSQDTwGfKdjcAzhe0mHA08DfgcMH9eWCIAiq0MMj2rKE8+1hspLGR9tk
b99PvR8DP26T9wTpeFGrvOkk7d9i2v4dOxoEQTCY9PBablnC+QZBEATDi
x7exVyWcL5BLYynmoLNw31zK/dh+qhq/yDrUCS68ObqR6Y/tdVXKtt4lmprX
SuMGlO9D30vV7YxXstWtlFVcWt1LV+5D3WodtWxQWfZilZuWrb6e0ANqk
Yx8g2CIAiCwcWx5hsEQRAEg0wP72IuSxw16hEkzS9S9GlpI0MacflIullSStJWq
2Q/6SkxwrPYyS9ULA3SdIlWZCPzuskvtDclzKhK4MgCAaHERBkI
0a+vcNLOaJUM1OAG4E9bZ8GbA7JgZICaxzXKJiFjZC0PPDfwKG2L85pm
5CCc1xdoS9BEARDzwiYdo6Rbw8jaQNgHHAkyQmA/guobjBbB9R3beQR
AEw5sY+QY1MjYLGQDMtr0nyeGeBVwDbChpDdtPlbBVRkChbF8AvmP7n
Ar2giAI6iOOGgU10mqqd1/SdHOfpAuAjwCndmtY0e2x+ms/hCqWl
nSQcABwDstOpkNh2/QbddC4Ig6J4eHtGWJaadexRJm5Ic5u8lPVRyxGWnn
psFFPYE9gdWzUn9iS+UpqhqFI43CILBwvPml756lXC+vcsUYKrtifl1Y1f
HVL1D0TeLekohZvUUChP/GFIAiC3mYErPmG8+1d9gUubEq7EOlt9fbkMOC6VmjJh
gnYDThQ0oOSriNt2vpWzr+NJMRwbV7bPZAsvpAZ23TU6JjqrxMEQVAT7i
t/lUDSrpLukXR/u6OVkrbPfw9nSfrfqq8Qa749gu1xTc/rtSZ5cL91P5s2L4b+G
A/7fUnvlAtHl8QBMGSpMYRraTRpL007wPmADDKutj2nYUyKwPHYnSjzKnTvoln
kNaq2G843CIIgGFa43unkdwL3234wdlL3234QQNLZJNnWWOwtlPgZcYPsRgJKnTvoln
G9QC1X/KYyrIZB/1QD481T9H3Qdogi/vOn7lW18bvJhleovP6r65Meyqr6qNb
cGQYJlXK0fz7q6QMQoVNlGVYEIgFcqCm68Ro8c8al3I9XawKOF5znA1k1l
3gIsK2k6aXPqibZPr9JoON8gCIJgeNHFyLd4JDIzzfa0wEbW1ZobWAbYCtg
```

JGAtcJ+l62/eW7kgLg0EQBEEwfOjC+WZHO62fInOAdQrPE4DHW5R5xvbf
gb9LuhrYDBiw843dzkEQBMGwwnbpqwQ3ApMkrSdpDOlEycVNZS4CtpO0j
KQVSNPSd1V5h3C+w5iC+tAdkn6dPxTkD8gzkr7TVH66pMn5/iFJ5xfy9pZ0
Wr5fMysi3SrpTkmXDuJrBUEQ9E+N53xtzwMOBi4jOdRzbc+SdKCkA3OZu
4DfAbcBNwA/tX1HlVeIaefhzYIwkJLOIJ3X/S9gF+Ae4KOSvu72X/8mS9rY9
qym9KOA39s+MdvedMl0PwiCYADUHDzD9qXApU1pP2p6PhY4tq42Y+Q7cr
gGeHO+nwKcCDwCbNNPneOAr7dIX4u0xgEsCMoRBEHQE3heX+mrVwn
nOwKQtAzwAeB2SWNJO/IuISki9RcP+lxgS0lvbko/FfiZpKskHSHpjUui30E
QBAOir4urRwnnO7xpSP/NII1yf0YKK3mV7ReB84E9cwSXVswnTaP8WzHR
9mXA+sBPgLcCt0havbmypAMkzZA04/a5D9T1TkEQBP3iPpe+epVwvsObl2x
vnq/P236VNNLdOSsh3QSsBuzQj41fAe8B3lRMtP2c7TNtf4K0G/A9zRWLq
ZvD1WjIAgGixBWCHoJSa8DtgXe1FBDAg6in6ln268BxwNfKtjZsbBzejywA
WlkHQRBMPTEtHPQY3wYuNL2K4W0i0jygcv1U+9nLLrzfStghqTbgOtI2+p
vrL23QRAEA2AkTDvHUaNhTAslpNOA05rSngMaoxnpfgeTyPKEeRAKFn5
S30QBEGdeF7vOtWyhPMNgiAIhhc9PJ1clnC+QS2sVFECeLkaVkCe86uV6
q/e78x8OZ6tqBoD1RWJAH4843uV6u+2xUGV+1AHdayLrTq6mmLWqlRTy
wJ4qu/FyjbG1KBqNHHUuM6F+mGZGtSZ6sDhfIMgCIJgkAnnGwRBEAS
DS4x8gyAIgmCQ8byh7kF1euqokSRL+n7h+VBJU5vK3CrprKa0bST9OSv83
NWizomSHpM0qpC2v6RTSvZrnKQfS3S3pA0ixJV0vaOudNkHSRpPty/olZlgpJ
2+d3+lDB1iWSts/3y0j6dq47M19HFMrOL6TPlPS1nL6bpFsKqkOfy2EgG+W
K9b4gaaqkQ3NdSToyt3lvDiG5caHNtmpHQRAEvYD7yl+9Sq+NfF8BPizpO7
afac6U9DbSF4b3SFoxCxsD/BL4qO1bcyjFDQt1RgF7Ao+SojRNH0C/fgrMB
ibZ7pO0PvA2SQIuAH5oe/fc9jTgaOCrue4c4Ajgty3sfgt4A/B22y/ngBZfKeQv
UC0qvM+yuY132p6Tz+9OtH1PbhdJLxTrNX0ZOQh4F7CZ7Rcl7QJcnNWN
Xs5l2qkdBUEQDDm97FTL0lMjX2AeybEc0ib/Y6RwiJcD/1hIXwN4AsD2fNt
3FvJ2AO4Afkj/IgMtkbQBSTj5SDv9ym0/aPu/gR2Bl23/otF27vs/NSJEAbcCf5
P0via7KwCfBT7fcHq259qe2qFL40lfmp7NdV7Jjrcsh+c2X8z1Lwf+BOxXKN
NO7SgIgmDoscpfPUqvOV9Iijr7SVqpRd4+wDksrtZzPHCPpAvzFGzxbMCU
XP5CYLc8cuyGjYnGD7eVIoxqJS0LeAI5vqvhl4xPbcftoe2zTtvE
8OmnEx8LCksyTtV5xO748cfnJF280qCDPyuzRop3bUbG+BsMJNL9xfpgtB
EASVGQnTzj3nfLPzOh34/LPdx4AKOl0/DFxBcg6r5DpHHAJNI+KPAb/Ldc
YAHwR+k+3+mSQ0XXXcW9jW5P5P9u1NSR9R9OjvYRyWtk5OLwgmb
2z4n2/sMSTbwBuBQ4Oc1v0dLtaNmisIKW43r108HQRDUhvtU+upVes75Zk
4A/hlYsZkYsZA2BXhrVut5AHgdsFcj0/b9tn8I7ARsJml10ijxTcDtkmYB
Z2V6rn9UsktN/tpvg/uBN1jwfeR+Hn0KKoK98D

z+V6rn9UsktNfQB5drpP7WORo0tpvg/uBN+V1Xmz/Iq/T/g3oeJre9u22jwfeR
+Hn0KHO88Df85p1kS2BO5vSWqodBUEQDDV981X66lV60vnmqdVzSQ64s
WnqI8CmBbWe3cmOVNI/5M1PAJNII7e/5vzPFOqsB+xSWI8t05cHSNOy32
y0IWmSpN1JI/AVJH0yp48Gvg+c1lhTLdi5HFgF2Cw/v0gSNDilMU2e6/cbjif
vvN6+kLQ58HDZ9yGNaE+SNDbb25n0peTMpv4upnYUBEHQC8S085Ll+8
Dr8/17gMds8ds67rT37z9qe5z+5jxn3xw75BP8Dx3m2F53bgaTo

O4ZrQpi+fIe1Kvl/S7SSR+cdtm7ST+iOS7gPuBV6m/Walo4FiG0eQNordIekW
4BrSzu3Hc37zmu8xpCniwyQ13vebwP5t2mvFySR93tsl3QP8O7C77ZdalG1W
OwqCIBhyRsK0s5L/CIJqTF13v0ofpHv4e+dCHaj6JbeO2M6Pt/wO0x0rq1osY
ojYzkXWHr1i50L98HINw6c6YjuPG1X9c/EGVYtTXVds55MeOqeSoUcm71T
6782bZlzRkx44RjVBLVzX91yl+iuo+kfxwVefrVT/buB1o8dWsrFCDX8glx9V
PYB+Hc7zkltOHfI+zG+5n7E7qgpuALxW0QE/P//lzoU6sEy5Qw39MpfqoaHq
+J1UpZdHtGUJ51tA0p9J09VFPmH79qHoTzC4VHW8I4mqjnckUdXxjiR6wf
ECPb2RqizhfAvY3nqo+xAEQRD0T4x8gyAIgmCQcQ9HripLL+92DoIgCILF
qPuokaRd8wmS+xsCNm3KvSML1+xd9R1qcb5ZUWeWpNvykZitJU2XNLlQ
ZqKkOwrP20q6QdLd+TqgkDdVSYVoppJqz5RCBKiZkl6VdHvj+I2SQlGfpE0
LNu6QNDHfP1QoP1PSSTm9pRqSpDWV1IcaqkGXdnj/jSVdqaQSdJ+kfy+c
Cd5f0tOFtk/vx85pkmbncndL+kYhb3r+cNwq6UZJReGElu+X85aR9Iyk7xR+
Vx3Vj5psv54gCIIeoc8qfXUix1c4FfgAsBEwRdJGbcp9F7isjneoPO0s6f8Buw
Fb2n4l/6HuFCjiDaSgDnvYvjnXuUzSY1mwAOB428dJmkSKn7xaQ8BAKVr
VDg3lI0n7s1A9aJ82zS4oX6CdGtJRwO9tn5jtb0obcrCKi4F/sX25UgCP84F/Jf
1CAc6xfXB/P5MCX7V9nlLgjTslnW57ds7bz/YMSZ8mBcsoijW0ej9I4TTvAT4
q6eu2j6ac+lEQBEFPUvO08zuB+20/CCDpbFIQp+aof58n/W1/Rx2N1jHyXQt
4xvYrALafsf14hzoHkaJA3dyoAxwGLDbct30f8CIpOlR/XAJsLGnDDuWKtF
NDWovkzBt9uK0fGx8D/pgjWDUiVx1Mi3fpksaBvFYHYK8D1i5pZwpwIknsY
ZuKfQqCIBhyag4vuTZJcrbBHJr+vkpamxRQ6Ud1vUMdzvdyYJ085foDSe8t5
J3RmNYEilO3i6kBsbiyDgCStgTus/1Uh370Ad+jfXSpqwpTrA3JwnZqSKcCP
1MSmj9C0hv7abeVstEDwDilOM8A+xTa/nSH9zg2/7zmAGe3ee9dgd90er88K
t+J9MWkWQmqHYcU7MwE2r67CqpGc154tF2xIAiCWukmwlXx71S+Dmgy
18pDN5+pOgE4vI263YCoPO1s+wVJWwHbkbRzz9HCBev9bM+AtOZLcgLQ
Xg2omHaIpM8C65OcTRnOBI6QtF6LvMWmZW0fJekM0tTsx0jOaXvblymJ
D+xKWge4RdImtp9uYbfduxTfZyDTzuOAKyS9y/afct4ZklYkCS9s2en9SMsB
V9l+UdL5wL9LOqTDB+h428c1HvIUf0tsTyPpL/P+dT7QGwcAgyAY8ZRZy2
1Q/DvVhjkkMZwGE1gY4rfBZODsvJXn9cAHJc2z3TwIKk0tG67ylO10298gT
bl2UtlZTA0I2IpF59iPt70haQ33dKlzXDTb80gxoQ/vou+t1JCw/ZztM21/ghQL
+T1l3yU77hc6aPV26tcLwHSS6EGD/UjiEGeycD25P6YAO2cHehOwGukLU
hAEwbDFVumrBDcCkyStpyRDuy9pH0+hPa9XEOg5D/jXKo4XanC+kjbMm
6IalFHZOZUkarB5trEaaRfZYgFpbV9AmpL+VMkunQbsDKzeqaDaqCFJ2j
FvnEJJ8m8D0pppK84AtlVSB2pM9Z7U6l26QdIywNY0SRNmtaEjgW0kva2f
+q8jOe43FT40B9G9pGIQBEFPYZe/OtvyPNKg8TLgLuBc27MkHSjpwCX1D
nUE2RgHnCxpZWAeSaf2ANK3g5bYfkLSx4GfZOcm4ATbv21T5SjgTEk/sfs/
uWX71XzU5sSmrKskNaZbb7P9SZIa0vGSXsx938/2/DyNfoqkeaQvKD+1fWO
b9l5Skhc8WdKppCnhXwGn9NfPfjhW0pGkHeNXABe0afP7wKFk2cXm9wO
uBK5sbITLXAR8T9JyTelBEATDhm6mnctg+1IW3ZeE7Zabq2zvX0eboWoU
1ELVNd9eEFaoI7ZzHcIKq42qpjwD8GxftUD+dcR2rkNYwTXEEq6qBlRHb
Oen571Q2caqy5SWIW/LahVVjeqK7XzmwxdW8p43r7N76Y5s+ehFPRkOK8J

LBrUwRtWUeObWoDyz0XJrVKpf9R2gutMDWLYG9Zqq9IIqEsAXJlc9sQcT
vGyl+te6mmIXwIRlX9e5UAdG17BF5/mK/85WVLWfZV3UPfIdCsL5lkTS20
nTyUVeGYgYQ56efndT8omNICJBEARBe0ZCbOdwviXJsoKbdyxYzlZvKJU
HQRAMQ2LkGwRBEASDzEjYqRTONwiCIBhWzO8b+n0RVRn+b7CEKSj/3
CHpt/lIFZK2l3RJU9nTlKWmlFSIZhTyJkuaXqhrSR8q5F+S078t6buF9HUl
Pdhot0X/2raTn/tTj9ow12+oOk0r9O9vxTCTjXPMQRAEQ01fF1evEs63My/Z
3tz2JsBzpEAVZVlD0gfa5DVUmJr5T2D3QgCNE4F/t/3XbtvRQvWoA22/lRR
043OS/iEXOYkUSWxz228DTi5UvyanN64/9NN+EATBoGFU+upVwvl2Rzdq
QpBk/45sk3cr8DdJRVlAbL8EfBn4QXao422fMcB2OqlHNas33d6hnUUoBix/
+IV2AcCCIAjqpc/lr14lnG9JlPR+d6Ip5mcHrgNekdQunvK3aOE0c7SV54DT
SbrAA22nk3rU8cCVkv5H0iFNU9vbNU07b9Cin9NsT7Y9ed1xbyrRzSAIgur0
odJXrxLOtzNjlaT1ngVWBX6f0zspGTVo6WABbF8DIGm7FtmnAjfavqdkP1u
10696VD5X/Dbg18D2wPWSlstlmqedH2hhJwiCYNCJaeelg5dsbw6sS4q33Fjz
fRZYpansqkCzbOGVwPK0F7I/mtZrv13tF2jTTkf1KNuP2/657d1J8a03KdtmE
ATBUDAflb56lXC+JbH9N+ALwKGSlgXuA97Y2BglaV1gM2Bmi+pHk9ZaW
9m9nOTEN6uhm83t9KseJWnX/C6NzVmrAY/V0I8gCIIlxkjY7RznfLvA9i2Sb
gX2tf2rrMz0i6w1/Brwmeykm+tdKunpfkwfTVIcqtq/RdopoR61C3CipEZA4q/a
flLSW8lrvgXz37LdVqkqCIJgsOhlp1qWcL4dsD2u6flDhfs/0mY62fb2Tc9bFe6
nA9MLzxfDovMjzWX66V/bdvLz1cA72tT9MmlndXP6dGClTm0HQRAMBb2
8lluWcL5BLTzx2mID/q4YP7q6jN68ikHnRtcgrzm+BtWXuX6tso2q60l1SMfV
oUh00oxjKtv41EqJI5kAACAASURBVFZfqVR/vdHjK/fhtRp+nsvW4HAeqfjZ
epn5nQsNAn3D3/eG8x0uSLoQWK8p+XDblw1Ff4IgCIaKXj5CVJZwvsME2
3sOdR+CIAh6gd4Yf1cjdju3QNIRkmZJui0HmNg6p4+RdIKkByTdJ+kiSRM
K9drFgZ4o6aWmoBWfbGrzwpx+f1Nc5Xfl+MuTC2UnSroj3zfiMN+SYzcf12
R3j/wed0u6XdIeTfmrS3pN0ufy86m53Tub+rx33T/nIAiCgdAnlb56lRj5NiHp/
wG7AVvafkXS60nnewG+DYwH3mJ7vqRPAxdI2tq2WXgmGEm/JJ0JPjrXfa
CR14rGyFbS9sChtncr9KlTt6+xvZukscAtki60/UdJmwHHAe+zPVvSesDvJT1
o+7Zc9yPA9cAU4FL+utzEATBUNDDUSNLEyPfxVkLeMb2K
5DiIdt+XNIKwKeBQ2zPz3m/AF4Bdmxhp9s40JXJcaFnFto9FPi27dk5fzbwH
eCrhWpTgK8AEyQNan+DIAgGwkg45xvOd3EuB9aRdK+kH0/h6b05/M/CI7e
ebyhdjJQNt40Bv0DTt3CqkZH+c0agLXNqqgKRVgEnA1Tmp39jOktYB3mD7
BuBcYJ8u+xQEQTDo9Kn81auE823C9gukEIwHwAKQ
552LlzXdNm1/Rp1gQ825W0n6TbgSdJU8ZMt+taqv/uSnC7A2aRRcGmKqkZ
Pv/hk5wpBEAQ1EOElRyi259uebvsbwMHAXsD9wRLo5UlSRLVkYK7ldHOglz
TW2NwXeDvxLI5wkrWM7F/s7hRR+8iHSKH0zSZPKNlpUNVp9hTdUeoEg
CIKyxMh3BCJpwyYHtglJj9mfIITR+q/dmnQfAg/+8YPW/HkDrOHhwAFZsHPWH9GVjEghy/pa3kxt3xVxUSaNmL0PJ6lGqy6Qz5t3gQAbxF37ofOjsSVd/pw6oX+
17Sm
u7hOek44N/y5qnGJqqvA9+XtCGwou21bU+0PTHX3Xew+hsEQTAQYs13ZD
IO+GU+anMbsBEwNef9f9G/AycK+k+0g7hffMO50XwfYtwK0sdGbNa75fWEL
9/xHwHknr2Z5JcsS/kr1uLo5lS+OT4o7g9b6gfcsB7M0lkjJW7qrBs8jBQpSMHvfNS+4U9BfHHRm3sQ/pNLOHGc2rCxIRi10gX1jhrzHwBkKGz0
Sqs3Bwk0Fr5bslXTfx28jL/TNjFXk+TGhhqa7Q3HVhmBXY6MIO5nK+
9oaFlgRUNdUbtKaKg60XyGU66zdRh+Z2i+TEH22i4P2PBiJQ1VwPRVSBfYwz7sUrXC9R9jeENehaSq5xTCkPpq5TJm2Etd3Hyq7RBYAxA5D6R6K5ObJWAIlZ52f1H0ygJiG6MNRI9a4PsM4GyDSR3Fw/r1

```
96TTFdG4Rv/khsvRfizjRL1HYZW37AuCCFu1MbZHW+LKxSBtBEAS9RJ3
TyXkm81TgfcAc4EZJF9u+s1BsNvBe2/8n6QPANGDrKu2G8w2CIAiGFTVP
J78TuN/2gwCSzgZ2Z1Hd8z8Vyl8PTKAi4XyDWlh9mXGdC/XDKqOqCyu4B
47eL5+2A1RiGVdfDVp19JjOhfrhOb9auQ8TXH27Q1VRBIBf3vT9SvX/dfLh
nQt14Mm+lyrb2GJUdaGx57VcpfrLqzdWKud3MfKVdADp9EqDabanFZ7XB
h4tPM+h/1HtPwP/U74HrQnnGwRBEAwruhn5Zkc7rZ8irVx5y2/yknYgOd9t
u+hCS8L5BkEQBMOKmqed5wDrFJ4nAI83F5K0KfBT4AO2n63aaG/MIQR
BEARBSWre7XwjMEnSepLGkE6oFKMTIulNpI2rn8hHOisTzncJU1A6miX
pVklfltLCSVYkuqSp/EWSrmtKmyrpsYLa0JRC3mmSZue8WyXtVMgro4ZUP
P60c6HsnpIs6a31/1SCIAgGTp1BNmzPIwVTugy4CzjX9ixJB0o6MBf7D2A14
Af5b+WMqu8Q085LnqLS0RrAmcBKwDeaCypJEG4JvJDP6c4uZB9v+7gcA
OQmSefZfi3nfdX2eXk9YhopvnMZrimqJzUxBbiW9C1wakl7QRAES5y6g2fYv
pSmmPm2f1S4/wzwmTrbjJHvIGL7KdKuu4PVWidwL1IgjLNpE2nK9n3Ai8A
qLbJrUVKSNA54N2ljQUS8CoKgp5jfxdWrxMh3kLH9YJ52XqNF9hTgm8Bf
gPNI4R4XQdKWwH3ZkTezK/CbprQzJDXOOYxh0S+N22UhiAZ72X4A2AP
4ne17JT0naUvbN5d5vyAIgiVNL8dsLks436FhsY+OpDVJsoXX2rakeZI2sX1
HLnKIpM8C65OcbJFjJX2P5NC3acrbz/aM3MZEoLjG3G7aeQpwQr5vqB0t5
nyL5+c2Xnlj1hm3TnORIAiC2unlmM1liWnnQUbS+qTZkOaR6z6kqeTZWW
VoIotO+R5ve8Nc7nRJxagUXyU57iNJ4g9V+rcasCPw09yPrwL7tJomL6oahe
MNgmCwqDu281AQzncQkbQ6SfjglBZiDFOAXQsKQ1vRYr01x2qeAXyqKb0
POBEYJen9Fbq5N3C67XVzX9YhxTWtfKg8CIKgDvpw6atXCee75BnbOGoE
/AG4nLSuu4A8HfwmUsxQAPJO5+cltQpzdhSw4MhSoY6BbwGHlezbdk1Hjf
amvdrRx0raDIIgWKLEhqugI7bbBvttUiRabJey7S3z7Z+b0m8CNsyP+zflnU9
ylmXUkFoFiz2vRT9OavcOQRAEg81IWPMN5xsEQRAMK2K3cxBkXj+qlFx
xW15YEC9k4FRd35lfw/rQ6qquzvSsX65sY1Wq9eM1Vx9bXOvnKttYb/T4yjaq
qhL9YMZ3h7wPAPf7xco2Xq44EdvXfiJvUOnltdyyhPMNgiAIhhXD3/WG8w
2CIAiGGSNhzTd2Ow8TJL1B0tmSHsjiCpdKektDKKFQbqqkQ/P9aXkH82Ii
CzltMWGHnL6bpFuyUMOdkj63JN8tCIKgG+bj0levEiPfYUAOcHEh8Evb++
a0zYE1l0Bby5LEGd5pe46k5UgBP4IgCHqCkTDyDec7PNgBeK1JZWNmPh9
cN+NJn4tnczuvAPcsgXaCIAgGRGy4CgaLTYCb2uRt0CSO8AbguIE2ZPs5S
RcDD0u6ghQL+qwcQSsIgmDIGf6uN9Z8RwIP2N68cZHCV1Yia1fuBNwAH
Ar8vFU5SQdImiFpxn0vzG5VJAiCoHb6urh6lXC+w4NZpFjPg4bt220fD7yPp
DPcqswCYYVJ49YbzO4FQbAUMxI2XIXzHR5cCSyXJAyJQUBkPQOYN26G5I0
TtL2haTNgYjARhhVjzHQZkfd89gRMkfQ14GXgI+FKXpv5bUiO
U1HXAqcBOkuYUykwBDpP0Y+Al4O80xY8OgiAYSnrXpZYnnnO8wwfbjwEd
bZG3SVG5q4X7/wv32bUy3igt5TdcdDIIgGCR6eURblnC+QRAEwbCilzdS
Wcb1ALr1b85zCvhm+yc/uqCRJsuMwqlfvw1xoEIkZRXbLlqb5qQfifn19d3G
HCsq+rbOO1Gj4XT/a9VKl+HaIIdYgzHDb565VtnPx4tUmtUapHTujMivUdI
98gCIIgGFx6eRdzWcL5BkEQBMOKmHYOgiAIgkGmz8N/5BvnfGtG0gtNz/t
LOiXfT5X0mKSZhWvSFLVkMaeyEeaUerRdzTW2sLum1ZvWhfpSQJ
```

pZQQ5pd6OOf6vsJBUEQVMNdXGWQtKukeyTdn49zNudL0kk5/zZJW1Z9h
3C+g8/xxXCQtv8KkB3unsCjwHua6pxjewvbk4BjgAskva2Q/xHgetIZXbK9hh
LSdNsb2N4I+DrllZC+WujjuzoXD4IgGBzqDLIhaTQp5sEHgI2AKZI2air2A
WBSvg4Aflj1HcL59g47AHeQfqlT2hWyfRVJ8u+AQvIU4CvABElrF+wtpoRk
O87wBkEwrHEX/5XgncD9th+0/SpwNrB7U5ndgdOduB5YWdJaVd4hnG/9jC
1OKwNHNeUfUsi/qpA+BTiLNFrdLevqtuNm4K0AktYB3mD7BuBcYJ9cpj8l
JMhqSIV+HtiUf2wh/4z+XjgIgmAwmYdLX0UBmHwd0GRubdKMY4M5Oa3
bMl0RG67q56WsLgSkNVtgciH/eNuLSP5JGgN8EDjE9lxJfwZ2Af67TRvFw3
b7kpwupG9sPwP+q0Q/H2jq59Sm/K/aPq8/A/lDfADAVqtuxgbjJpZoNgiCoBr
dnPO1PY00W9iOVoeXmxsoU6Yrwvn2BrsCKwG3p6VaVgBepL3z3QK4K99P
AdaUtF9+fqOkSSQlpL2XWI9Z9EO9z7p7DP/th0EQDAtqPmo0B1in8DwBeH
wAZboipp17gynAZ2xPtD0RWA/YRdIKzQUlvZc02vyJpA2BFW2vXaj7HdJou
KUSUq4fBEEwbLFd+irBjcAkSevlWch9gYubylwMfDLvet4G+Jvtj6q8Qzjfwa
e45jsz76p7P4VRru2/A9cCH8pJ++Sy95J2LO9l+y6S076wyf75wBSnT92ewPvy
UaNZwFTKf1s7tqmfYwb4vkEQBLVS525n2/OAg4HLSDOK59qeJelASY29M
JcCDwL3Az8B/rXqO8S0c83YHtf0fBpwWr6fSnKAzazaws6HC4+ntWlrMVu2
byNtl+9PCQlKqiEFQRD0GnWHl7R9KcnBFtOKJ0UMHFRnm+F8gyAIgm
FFSAoGQWZMxRWMMTXMao+quIgy1/Oq96GyBVheoyvbGFPRxjKq/iaja/
hpLFuDwtMWo1aqVP9+V1OIgnoUib4349uVbTy61Zcq1X92fjWFqLoouZbb
04TzDYIgCIYVIawQBEEQBINM6PkGQRAEwSAzEtZ846jRICBpuqT3N6V
9SdIPJG0r6Jd+frgEKZqZIs6c2FtENy2uT8/JCkNQtHgp5sUk4ao6y0lNW
MLOnzBXun5Chcjecv537cLulWSf/VIdRlEATBoDLffaWvXiWc7+BwFungd
pF9c/qZwIG23wpsC3xO0j8Uyt3eVHdv4M4mW/MbCkTAj1hUOenVprJPAV9
sdW43n2nbBdjG9tuBd+TyY7t41yAIgiVKzcIKQ0I438HhPJJYGYfDNwPYfgY4DCjqSf6GrLAhaX3gb8DTF
mO7jTbNwPYfgY4DCjqSf6GrLAhaX3gb8DTFfryNHAF8KkWeUcA/9KQOb
T9qu1jbD9fob0gCIJa6bNLX71KON9BwPazJz
e4HngUUmbkCJanVNDl44BvpJ1LAGQNB4B4YZ3t2WSNFCiCoB
Lu4upVwvkOHsWp58aUs2j9+WhOOzvX2YPFw0l2TXawNwAfKyQv0hdJ789
rxg9JelcbO9NsT7Y9edK49ap2KwiCoBR1hpccKsL5Dh6/AXaStCUwNk81z2J
RuUGArVh8Tfe3wCeAR2qcAv42cDj5M5Dt/l3Sevn5sryGfAcQcZ2DIOgZwvk
GpbH9AjAd+Dlp1AtwKrC/pM0BJK0GfBf4XlPdl0iO8uga+3M3ycnvVkj+DvB
DSSvn/ghYvq42gyAI6mAk7HaOc76Dy1nABAFkrs8/JGkJ/1nSK8ALwB+byg
RBEAwpvbyLuSzhfAcR2xcCanq+Gxw9mHf6m2ua3Ol2UPGZHXfklGGFnLG1B+oerC8KWfZWFx5HaCZRUW
22uaPvsJdCfo4FbCs8/JGkJ/1nSK8ALwB+bygRBEAwpvbyLuSzhfAcR2xcCo
sFqbV9NOtLTqvzUNunbF+4ndqrTUFqy/RAFNSPbt1KY/cjKHcflKwiCoCeJ
2M5BEARBMMj08lpuWcL5BrXwasVQ5zvPH9e5UAde1gqV6l/G/1Xuw7I1b
KN4hfmVbUwcVe3nOZfqCk/PLxbfpXse8WvV+5GO1w+Yl2v4fZz8+DWVbV
RVJAI466YTKtW/auPq6kx1ECPfIAiCIBhk5o8AXaNwvkEQBMGwopcjV5Ul
nG8QBEEwrBgJu53O7LlA1FHGa0qZKOlTS/pLOasp7vSnJS2PAPaH7Hka0UtF/20v
6WyHtD4V3fqypfOM87DslXZ3f9W5JP5V0UKHcq1klaKakY/J7PN1ka6Pc/k

uSbsk/lxsktYrB3OodL5J0XYvfU6PPd0qaUsg7TdLsQvt/6mQvCIKgFxgJsZ2rj
nwvAI6TtILtF3Pa3sDFtl9JMRrYz/aMpnonkZR3LgKQ9PYu2pxDEgDYp4s6
U4Br8/8vK6S/lKM4LSA78mtsF4NPNDje9nFN5dcEfg3sa/u6HJhir2zj1FzmI
WCHLJxAlvA7x/bBLdp+wPYW+Xl94AJJo2z/ot3L5S8BWwIvSFqvKT7z8baP
kzQJuEnSefaCXSxftX1el/aCIAiGlKVi5NsfOSTh1cCHCsmNuMX9sRbJiTbs
3N5Fs5cAG0vasEzh7Az3BvYHdpFUd8Smg4Bf2r4O0llZ2+fZ/ktVw7YfBL4M
fKFD0b1IISgbMaBb2boPeBFYpUTTHe0FQRAMFSNh5FtHeMkFggGS3gi
8BbiqkH9GYWrz2Jx2PHClpP9REodfuYv2+kjhF8vueX83MNv2A6Twjh8s5I
0t9K0oWLBdIf2IQvohhfTGO27C4spEZdinadq5nWbuzcBbO9iaQvo9nJXvF0
MppvR9tp8qJB9baP+MbuxlmwtUjR544aEOXQyCIKiHCC+ZuAT4gaTXAR8
FzrNdPBi32LSz7V9Iuowksbc7SUB+M9uvlGzzTOAIZRGADkwhjeDI//8Eabo
cWkw7Z0pPO1eg1bRzq3ItEwt11gTeDFxr25LmSdrE9h25yCGSPgusz0JJww
aLTTuXsLcA29OAaQD7rLtH737FDIJgRLHUTzvDgqD/vwP2pNyUc6Pe47Z
/bnt3YB6FsIcl6s4Dvk8SG2iLkl7tXsB/5HXXk4EPKMVRrotZJCWiJcUWwF3
95O9Dmkqend9xIotOFR9ve8Nc7vQS0+6d7AVBEAwpdl/pq1epS9XoLNLa5J
rA9Z0KS9pV0rL5/g3AasBjXbZ5GrAzsHo/ZXYGbrW9ju2JttcFzifp4tbFKcCn
JG3dSJD08fxelcgbsI4jfWloxxRg1/x+E0lfBBZzlrYvAGYAnXZPl7IXBEEwV
AyWpKCkVSX9XtJ9+f+L7ZmRtI6kq/IJlVmSvljGdhnnu4KkOYXryy3KXA68
kTSV2vy2xTXfP+S0XYA7JN1K2n38VdtPlulwA9uvknZNr9FPsSksLj5/PouK
yHdDcc13pqSJeWPVvqRd3/dIugvYDuiku9u85tsQrN+gcdQIOBc4ud1O5+yc
30ThC0/emfx88ctAgaOAL0tq/N6PberDW7q0FwRBMOjYLn1V5GvAFbYnA
Vfk52bmAV+x/TZgG+AgSRt1MqyRECMzGHqqrvnWE9u5Wv3LFLGdG8x1
b8R2nl/D2t7KFWM7v1hDnOuLnxjInsxF2WOt6qtbvRLbeZe/nF3pX+uEVTc
p/cGY89wdA25L0j3A9ln+dS1gel7G66/ORcAptn/fX7mIcBX0BFeN/ntlG6ds+
Vyl+uNuWKdyH25atroQwGs1xK1dpv99eh2pw+mtmFaWKlGHqMHyqvaFq
M+jK/dhVOvNlF3x7PyXKtuo6jx3mPXtyn2og/l95f+NSDoAOKCQNC1vFi3D
mrafgAX66/3NtDZmI7cA/tzJcM84X6XoU99tSp5te8+S9f8MNH/F/USXZ4h7
FkmfBprXEv5o+6Ch6E8QBMFQ0c1u5+KpjFbk5dBWe3SOaJHWFknjSMu
aX8oxMPqlZ5yv7ctYNPpUt/VH9JpkXvdtG+UqCIJgaaHO5VLbO7fLk/QXS
WsVpp2falNuWZLjPSNvbu1IXbudgyAIgmBQGKzdzsDFLDwh8ingouYCOY
riz4C7bP9XWcPhfIMgCIJhxSDudj4GeJ+k+4D35WckvVHSpbnMu0nBm3Ys
nBz5YGtzC+mZaeehIJ/FPQF4B/AK8BDwJWBZ0tnaCaQIU6cD3wLeC3zH9
v8r2FiGdEZ5c+A7wCW2z5M0nRTD+hVgDPAH4Ejbf+2nPxOAU4GNSF+
MLiEdw3pVSYxhcjEqVm7j0FxnOWBVYCwLz0zvQQqpOZcUlvMvwCdtPyn
pBdvjCrb2ByYDTwAfyclvBxpr5j+3fVL7n2YQBMHg0M2GqyrYfhbYqUX64+
RQxbavpUMkwlYstSPfPFVwIWnr+Aa2NyLFi16TNNVwjO23AJsB7wL+lSQ
iMUGLShjuDNzR2BHXxH62NwU2JTnhxaYsmvpzAfCbfKbsLcA44OhO72J
76xwm8z9IZ603z9dDucgOtjcjBdnod7uj7aMb9cnhN/MVjjcIgp5gEKedlxhLrf
MFdgBes/2jRoLtmSSn90fbl+e0F4GDga85xSr7NYvKGXYMqZkDghwGvEnS
Zm2K7Qi83AiokeNjHwL8k6QVBvB+rbiaFLc5CIJg2DKI085LjKXZ+bZTI9q

4Od1JEWlcFo8oqjgtR5p6OL9TY9mZ3kp7haJW7T4PPEJ9DnM3Fk4jVyZUj
YIgGApCUnBkImg7V2HbN5Ic8YbAB4DrbZcNjdTfukC7dhvpbftUot2rJM0E
Xkdal25HV59U29NsT7Y9eYNxE7upGgRBMGDcxX+9ytK84WoWsHeb9PcU
EyStD7xge25OaojMv42SKk5ZYenttFcomkVSYCrWeR2wDvAAsC5JbajIqsAz
JZrfwXZzuZckjclT4t3YCoIgGFJ6eURblqV55HslsFzWugVA0juA+4BtJe2c08
aSBBy+V6h7FvBx0jrtxZ0aygewvwM8avu2NsWuIIlYfDLXGU2STTwtrzvfCL
y7oZYkaTJph/Ojpd94Uf43v0PjHT8KXDVAW0EQBINGn/tKX73KUut8s/rSn
qQzXA9ImgVMBR4HdgeOzEG1byc5vlMKde8EXgSutN1fUOIzJN0G3AGsm
O126s9H8pmye4GXybuTs3rSF4FL8xTyCcAUD1yw8ovAh7Ot64Ff2756gLaC
IAgGjZGw4WppnnZunNX6aJvs7TvUXWzXsu39C/f91m9j81HgQ/3kX0Q/x5
Vsn0bSOS6mTWxT9jHSBqz++lNdaigIgqBmetmplqabbxBxxTXQCzhgJNjoh
T70io1e6EO8x8j8WSwN11I77TxUSFqtScC+ca021H1bwhzQuciwsNELfegV
G73Qhzps9EIfesVGL/RhqWCpnnYeCpzClW0+1P0IgiAIho4Y+QZBEATBIB
PONxgs2opZDzMbvdCHXrHRC32ow0Yv9KFXbPRCH5YKlBfIgyAIgiAYJG
LkGwRBEASDTDjfIAiCIBhkwvkGQTAskPThoe5DENRFON+gVrIYRLu8N
w1mX4IRx5FD3YHBQFJPHAHNse47lTl4MPoyEgnnG9TN9MaNpCua8n5
T1oik0ZJeX3gek/WD26lClUbSypKOKFHuw/1dJds6rHD/kaa8b3ff+wWBWv
aUtFUXdX4h6edtrp+VqL9qf1fJPuxYuF+vKW9IR7WS3i3p1JJlPytpUr5X/tk
+L+k2SVuWtHFt4f5XTdk3lKg/WtK4wvM2kt6Tr/Fl+tDG7kaSjsrx5X9Yoso/D
bStpZ2e+IYVjCiKmsXNf5T70zNeWEjaF/gx8Pf8R2Aq8CuSwMV+pTsirQP8
O/BGkuM/E/hP4BOUk4L8UNP9bwvPBi4oYWNfFipi/Rvw60LermThjP6QdA
nwNdt3SFoLuBmYAWwgaZrtE0r045IWaW8CvgSMLlH/JtI7C1iLJEDS+H0
aWL+EjeOAhnM6v3APaVTb6ef51ixU0oxI2iSblujDwkrS5sDHSPHdZ5dov8
EXWRhDfQqwKbAesAVwIrBdCRsrFu43bu5aifrfBZ5i4WfrLJKAy/Kkz8fhJ
WykxqR1Se8xBZhHki+dbPuhsjaC7gnnG9SN29y3em7HkcBWtu/PI4nrgH1t
X9hlX04nSSeeT3J015N0kze1/WSnyrY/3biXdEvxuQvU5r7VczvWs31Hvv808
Hvbn8wjnD+SFK76xfb5CxpN+tRfJ+lWHwN0HPnaXjBSzT+LLUr2vUjVn8
Vs+hEeKdUB6S2kL0RTgGeBc0hHLnfowsw826/l+92A03Pkuj9I+l4/9Yr092+
hzL+TnYDitPBfbX9IkoBrSvYBSX8CViJplO9t+z5Js7twvJtKer6VadIXorbLU
Es74XyDullD0pdJ//ga9+Tn1UvaeNX2/QC2b85/DLp1vACr2p6a7y+T9BfgHb
ZfGYCtgR6Ir+PLyGuF+52AnwDYniuptKSkpLcBR5BGaMcCB9qeV7Z+gaH
6Wbxq++EBtt3gbpJz+lDjMybpkC5t9OUZiP8j/T6OLuSNLWljZUl7kpb+Vi5
Mu4vkDDsxqul3dzgkb1ecji7B08AEYE3Sv8/76O73e/sAv4gt9YTzDermJ8D4F
vcAPy1po+i0AcYVn23/V9nOSFqFhaOqJ4EVJK2Y7TxX1k4FNssjAwFjC6M
EkaYIy/CopM8Dc0hTtb8DkDQWWLaMAUm/BiaTpn4PWa
wv6WLSuzfuyc/rta+2gD82J0jagDSK3df2JiVs7EUa+V4l6XekEV/ZGYgG/0G
a9h8NXGx7Vu7Le4EHS9r4X+AfC/fFEX0ZXe0xksbbngtg+/Lch5Uo/7nC9u6
5zl7ANyW9mfRl4J22O649BwMnIlwFPYekb/SXb/ubJe08BPTR+o+rbfe7Tinp
tywcBbyHpj+Ktv9xsUpLAElrAEeR1lpPLfyh3E0PX9cCRsPsfBdGmu3Dcr8
LIpfhr4MLPIFqMwXouyc2mL7fzvZyHbWAvYhrdduCnwHuMD27WXqZ1Zxsr

AnuQHPeOwC+BCxs/2xL1lwHG2/6/Jpub217sS0Ld5N/HzqTZi0dy2rqkTVJX
2P7+AO2uQfrZTgHWsb1Oh/Jftz2gjYNLO+F8g1qRtDGwge2L8/PxLJxGO8
X2zUPWuS6pw1m02Als0vrcsPqHV9cXoiabywKbAI/ZfqpE+c+SnMIE4Nx8X
VRcjx4I+Xf0EWAf2zt2Kt/B1iO2Ox6pa/oysxglv8wcSFq7b2zeegE4xnaZXcod
kTTJ9n0dypzM4ssJzwBX2b62da0AYto5qJ9jSCORBu8n7ThegTRdt0cnA5LO
tf3RfP9d24cX8i63vUuZjkj6uO3/L9+/uzgikXSw7VP6q192JNaB4i7hBuMlzQ
Q+U2Zji6Rf0H4dzrb/uUxHJI0h7RbfONu7Eziz5Br4s51+XiXa/xFwsu1Zearz
OtL096qSDrXdaQf6qbnOx2zPt7q8swAAIABJREFUyDa7+hIjaUfbV+b79Wz
PzlPuP5b0TLfv1KqJkuWKyzGfI+3u7wrbPwJ+lNd41ZiC7gZJ19reNt//yvYnCt
nnsOiO9FbMaJG2KnCspHNK7sRfKomRb1ArkmbYnlx4vt72Nvl+wT/0DjYW
7KaVdLPtLVvllbCzoG4LO4s8t6m/OzDB9qn5+c8s3DR2mO3zyvSjje0PAwfY3
rVE2b1aJC84JmR7QgkbGwEXk9ZNbyI5iS2BdwO7N9Yt+6nf8edVog+zbG+c
778EbG97D0lvAP6n0+9V6dz3R0ij3zVJI9/9O02NNtmo9JkoYb/UyLepTte7xy
V9sr9826d323aVf2st7I4F/hSbsdoTI9+gbhY54N9wvJk1StqoegyjQdWjLYeRN
uc0WI50vGNF4BfAgJ2v7QsklYrYVPWYUOZk4F9s/76YKGln4BSgm6M2A+
XVwv37yGeebT/Z2PjVH7afIa1p/lDpDPc+wFNKgVcutN3xzDQ1HP1q2gvQX
H+1MjaaGMgIqFX0KZE2bq1NOmZXte0Bj8xsv1Tmd7o0E843qJvHJW1t+8/
FREnbkAIzlGEFSVuQjmGMzffKV9mjHFD9aMsY248Wnq91Os/5bGPH9E
DJU4WlI8zVcExo7WbHC2D7D3ndrhN1nOf8q6TdgMdII+5/hgWbl7r5vZJ/L
8cBx0nakEW/JPVbtc19q+d29LfBrePmtzqw/fnGvZKX24903Oh6Fj361ImqR5
4WI/8+P0HanR+0Iaadg1qR9E7SWtFppEg7AFsBnyJtaCkTOu+q/vJdMiCCp
BeB+0l/SDbI9+Tn9W3360Al3W/7zW3yHrC9QYk+tNpYswrpmMkptn9Swkbx
mNC5pHXSBZQ5JiTpXuDtzeu7kpYnndWc1KH+gKcgCzbeApxE2rV9vO3Tc
vr7gV1sf6WEjU1IMxIbsXDd+riyO50l/ZW0a12kSFSNHewCtrW9Sjfv1ML+O
bb3KVHudhY6+zez6GfTLhGtKzu5/YGvAH8GvmP7ni77+4v+8t0hsIykuSy6p8
HAS6TjU1+yXfYL91JHON+gdiStCRzEwo09s4DfA1NsHzSI/Vi3v3x3CNgg6
QxgerODlPQ50nrllBJ9aN4lbFJkpau7cBgPUeGYULZxJLANcHBjk5ekiSRn
OMP2UR3qt3W+kpb1wohPS4y8Bn8caUPfDNLPYStS2M5DbV9UwkYtx536s
V92t3PVz+ZBpDCXV5B2OFcNPhIMMuF8gyVGni6ewsLYueeX3TEraTXSO
c635qS7SDtzSweDkHRKrvOnrjq+sP4apJjQr7DoKH45YA/bfxmI3aFCSYHm
MNLOc5GOphxnu+O0c/N5zjzVuQPpd/Qh22uW7MMHgK+x6I7r79q+tETd
W0mbwx5qSp9IOnK0WZk+LEm6cL5vBtZ005lgSdsBj9t+oEP9PlJs56dZdLq8
6zjXkkYDq+Q19cau+P2BQ2y/rUPdNUh7EN4M3Eb6ItBqeSJoIpxvUCtqHTv
3UNv9ftNvsvE24ErgMuAW0h+ULUibdHa0fXdJO1/MfVkr9+Ms2zPLv80CO
zuyMPj9rMZRlZJ1tyVNcZ+en89joeDEt8rYUgelHHd5dlpZ9WaAR1O2JjncP
UnvcRApytP/9Vsx1f0s6VjNYSw8ojKZtHHsp7andah/p+2Nus1rKlec7l2MktO9
7X4fAi6xvVYJG5cAX7d9W1P6ZOAbtvuNYV115Fyws0DEhBRacioLRUz+s9
NnSylK2E2k6fvdSIFH9i/T9tJOON+gVvI38mv+//bOPEyuqmr3vxcUiASQIIq
KoAiIKERlMAryqRiUwQkcmC4gOCsyiCgyCPPcqiohMooj3U/ATyIciQRy+i4o
BcUAQQSCAzJMigyiDEATf+8felT5dXVXnVNep7k6f9XuePDlzrVR3atXee6

33BfbyiHbuzVWmRgvP+B5wlu2z2o7vQOrx7NR60+t5a5KS8I4k6b0zgXm2/1
Ry38ACGUq2invbXpj3ryKNKpYnffhWaTXqtQZuVxCG6LL2XHxIT1EHSZ8
jzWDcTnr/ziFNV1cWuJC0kLSu+re246uQitnKRllXkkbZt7cdXxM4r2LibCUt
AT8Ctimer5K06qhJkHS1u8hhSrrK9gZlz6gDSVeTZnHGZWIi6QrbLyvsD9yu
1RSi2jmomzq0czew/Y72g7bP1jg8cPMH6lHAUXkq/JvAZyi30isKZLT+npmT
QCWBDGDFVuLN3GD79wCSPt/lnvb4u36YK6lEVWHcHq+Z9wPXk1p9fmj
7MfUpcEH6sj9m2cD2/RXbUj5Dcg46kpGfzSakaexKFnrF5Cpp0XjWSqsW/J
XQS3+5tPK7UOg05hT9uQkNamIijdZPX7q4388yUdOI5BvUSv6Pe45GtHP3
A54l6WtU1859ZJznOpIT1JtIXwq2JFVilsohdhvV5XaMk/Mzy3h62zOLpvGV
1kk7vP6o9daKzxlUoWo1YCvScsJxefQ3Q9JTXL3l6UFJs21fWTwoaTZQOgV
ue76kW0jVvXuTPuCvBt7V/sxhUmjH6YjtKr7Al0p6X4divr1IXyzKmFVTkdug
JiYrMSLa0qI1VV3V57mRxLRzMHTUp3aupDtpE+5vnSK1L1RSNJI0l5Qstg
V+RxqFz7fddwLv8OxK02tKggwn2/5R2/HtSKIX2/bxmoOst9Y2HZjbk7Yjvbeb
k4T8d65w3+bA6SSBkuLIdXdgVw+gBSxpzYpTxsX34HTS+7k4cVRZPy9pz7
HtPSs841mkqfvHGUm2GwPLANvb/kvJ/bX8PDtU4xdxhSr4Su97MJZIvsGU
o+QDoR9Xo18AZ5CqrGub/lISyLi4uNbV49q1SWuLv2Z0xfSrge3K1p3zM+p
Ybx3KWpykFYG32z6t4vWrAR8mFbCJ1IZ2ku27K97/KpKC00W275G0IWna
+TVVvpTVsX5eJ0rOVK2138rFfL1av+pC0ia2Ly25JtZ4x0kk32DaImlL2z/P2y
+wfUvh3PZl04NdipT6EsjIz1mWEUMDSAnnDNuPVbz/XtJ663GMrLf2W8T
2BPDPTqeosEYo6Tjb++btfWwfXzh36kRUuEo6mjTivoLU2vJDUiI/Evh61fez
hjgGdiTq8tzK3sQ9ZocGjWF9RroV/uGCTnuX64f+JWC6Emu+wZRD0mE9T
tv2/6n4qKMZcWU5m9EOLYcAZWtz7UVKBu4mTZFW9o51UpX6ZtXrO1D
HeutVA35IblHY3h04vrBfqac0x93LnWnLkkdsC7w8f/lYmSRXuqFLbO8qxja
XZJYxt8LlAzsSFV63kzdxqXgLqVhwJv0XM3aKYc38mjsBTwBrAhtXLCh8rq
QTup20/bFB45uuRPINpiKd1mSXJ2kBrwJUTb4DiegXp7fzVLP7XS+uoyrV9
pPAT4CfFNZbnwbcJanSemsN9Hovq3JAh2NzSH2/pX6+wKOt0a3tByRd32/i
VerZPhl4DklA5UiSCYGoqInc9nvxtqrLIG1xtHsTv5ckFFL1WX8pW4+tGMe
vSUVT84B32L4hVzzfWvERj1KtQCxoI5JvMOWwfUxrW0kQYh/gPaQPiGO6
3dfpUV22O+13RNKHSPKFy+f9h0mKTF+tGMPAVak54X6QERWhb9r+Xm
u9teJjvjtIDMBSebS5VGF7cXtJlQe0WqwAlGQeDyWphX3Q9k8qPOKFkn5Q
2H9+cd/2Wyo84xhS29RvgK1JRgSHFqfR+2S863aDehPXZRl0L+kLwLNIdp
k30N+/6f6q6/3BaCL5BlOSXCG9P2mt9DTgFVWqettYK384q7BN3i8tVlLSQ3
41Scf55nxsLeB4SbNsf7ZCDJdQbkhexmnAv0jiJduQTAX2dZLxq/rBd6+kdfL
IRqRp8B2AW0meuGVVvu0tJX2parVQMlE4FHgM+JztnoIVbby1bb+fL2Itb
HtB3p4v6d4BEu8gPIfUAfDlXPl8FlC1ZxtgK40VgVlM1QJD22+VtBLpd+GI
XCD4dEmbuoIJCqNtIoM+iIKrYMqRC2u2B04hVcI+PM7nDCSiL+l6YHZ7I
Y+SUfiVttetEEMdbkCLFY+UnGx+12+FqZKS0ctt/0vSzqRe2a1Isp2fsf2aQW
KsGMOlpNHV0aRR3yiqtPnUEMPNjJ7+/lJxv0qPrro7ErWeUVlXOT+v5U28
E2k5odSbOPc7t5tsFEKoXozX9txnFmJ5XlkFuaSNGDvDdJ9HW3EGHYjkG
0w5lCQqF5GKPzqJxldS75G0htukCPuM43rbL+py7jrb63U613bdwFWp7e0c4

2nvUEEGUNIZwCWtEd8g7SJKXroH2H5fhWsX0NudqWebj+rRZa6jR3cd0j
Rte4JZk2SKcOPYu6qR388dx7OOXDfFHl5JJ7rgIVy4ptPMxSxSv/JOHoeWe
lOIaedgymG7ssl8CfPJU76SznafmtDAncV2pRaStgR6iiAUqKMqdbZGjOxFq
nR+kP6+jPw7V9Y+QFL5KhYXVZEz3JA0SmwVKp0IfBV4JRWnf22/tsp1kub
a/mmHU9u1LqGDLnPFGHr601bkWJIu9yhxCUmr5nM9TREK17+ItP5cdO
46pUriVc1mG12eUfz3bdblmo5Sm0oGEScwuko+KBDJN5hySHq9s9jAePpzi4
8qbI9nGu5jwLmSLma0ItNmjF1/7MbAVam2KxU0lXAYyUloaZIq1jWweGr+
5gr3f4Ok6/wbkqzm5SQBk12G0F97FMn/eRSuQZc53ztuC73M893mRpTju0z
J3rBKDK8itbp9nbS80nLuWpB/x39b8ojLSP3i97YeWQwFmFCxkHbyezFzM
mOY6kTyDaYiX2Kw/twWvaqdq7CI9KG8LiOKTBcB/0kqGKpCxxFvXufb0fb
R44irb2z/MPdzrtBWuHYZaY2vjGVtn5q3r5d0APCp3AZVN3VV8o59cMFCT
1K7hd4uFR8zkClC5jDStOyCwrH5ki4gGUhsXXL/x0lFUo+SugDOGW9txDD
IRWSxptmDSL7BVGSg/twCswvTszPapm6rTNceR5peHCWQkafUjqPa9OJi4
QhJzyBVuLb6O6t+iRgYSQfa/iLwgKR32v4ugO1HlFyCehb4AMspOUK13v+
HgQ1z5XTdxVIdP7TbplpntMVTNYZDgI08Tgu9zKCmCAAvbEu8QCoClNT
T1zhfdyxwrKQXkH6ffi7pNuDIIa2zdvsSeSJjf16zSF0C+wwhjmlDJN9gKjJwf
y7UMl078PQi8C9Ju5EUjNYl6TKvZXv1AWPrlx2BL+btgxjd9/smypPvXxhdO
HZ3YX+ipjmLa8vF1+8nhkEt9AD2JTl37cJYU4Sqfde9XJwqC7nYvkXSuaQR
9/8i/Y4NI/l2a8e6rD0k4H5gf9tVhFMaSyTfYCoyUH9ujdQxvXgPyVHpEJIZg
yVV/YCuk0HVvgb2sJW0Yu5NLuPWLsc/bXtMi1KfDGqhh+2/Aq/WaFOEH7
miKULmeeosyyiScURPcr/5jqTagztIU8+fG8/6u6TdSaPUVmX/tcAJtr/duqaw5
DCKlsBGFoJZm5R8bxpCHcC0I1qNgilHl/7c1i+qyvpza4zjTOCCLtOLW9kuX
SuVtB/pQ3J5UoHSfwM/HW8f5ngpthONt3Uprxk/Yvs+SXNIdoI32p5fMYabg
INtzxv03zBeNKCFXl3khNcrkJ7iKbkd74/AucCDtM0IVfkSkZ+zG8lze39SEZ1
INRZHA8cXE3CX+59CkujcE7iNpIC2Osk28mDX4zk8LYnkG0w5JL0VWN3
2SXn/dyRxBgOfbK1XTkAcvTxX3+6KNnj5WWuRHWuAdUhFNee4gqVgHU
h6kjSdKdKoveVwJGA52z3VlZTMLnYn/QzmAW8AFpBaja50djwqecaapLXy
mSQv4776YesQLCl5fqmF3rDJI8g3l/2OSzqc3j3PVW03f0ta87617fjzgXm255
TcfyzJaGI/2w/lYyuSiiYftR3rvl2I5BtMOST9ivSBcEfev4JUuLQ88C2Xu9/UHc
+4PFfzvWOEPiRtQErE77b9wvoiHR6SFgIvIykw3Q6sZvufeeRzhUss8Nqe9S
aSLOalwL9bx12izSzp76Rq846U3d/lmX1Z6A2D3PrUcq16I/BL2++YoNdeaHv
9fs8VrrkBWNdtiST/m66zvU590U4vYs03mIos49HydBfbvh+4X9LyEx2Mk/5w
PxrERcYIfTjZEV5FeZHT0JD0XEYMEf7scmvCx2w/Djwu6Sbb/wSw/YSkyvq
+WVjiQJJO9UkUkm8F7mV8es7tMQxioVcbkrYgFeJtS6oL2Ax4Qeu9rXD/1q
TiufVJo+CFJNOPH/cRxqPjPNfC7Yk3H3xS/RlFNI5IvsFUZOXiju2PFnZXn
eBYBmVQoY96gpAOAp5aWNP8DfAPkpj/aSQf2V48XdL2pH/PinmbvL9SxRi
+ALwF+LiruRi189Cg6/0a3EKvFpRkR28nCZd8wvZDOY6qifd9JC/hAxmpO
N4Y+IKk1W2XtitlXixpTEU/udixwv0LJe3WvjYsaVfguooxNJJIvsFU5JIufZQ
fII0QliQGFfqoi3cCRfOE+22/PE8PXkh58r2Qkb7mixjd49x1KriNJ0nuVOOth
L11nPcVGdRCry7OBt5GEjh5MrcL9RPHfsDmHu1edEEeDV9MUs2qQhVFr

158BPi+pD0ZrQI3g+ptV40k1nyDKYeSs8p8ksJUSzhhI5L369tyq8cSQUmhU
xWhj7riaK9w3qPVPiLp97Y3qul1du9WqZunWbtiu2oSHwiNWOjtRGqPeTrw
Rlez0KszDgGvy3FsA6wI7AX82CVqVZKudRcpzF7nOly7nu3r8vaythcVzs1xu
cxl69rXM6ICd43H6qGv7P4tQac1kXyDKUvhPzT0WegUjEbSn4CXtLd+SFo
WuLquwphe7UCSzutw2MBsUnV7HRrWfZEr2t9NKroqtdAbYhxPJYmd7ERq
Y3tGyfWXAO+3fWXb8dnAN2xvWvF1B25B6/d1gkRMOwdTlpxsI+HWw/eAr
0v6aGtdMRevfSWfq4uugh22R8lxStocOJiknvXRjjcNmTyLcgJwQi7EmhDaB
Ufyl6LzgPMkVRm17g/8QMkisTjduzuwaz+hdNnutD8IQ9PrXlKJ5BsEzeBQk
o3g7UoawAKeRzKJOLTG1ymdSlOyZDw0X3ukO9sHdrpvV9vfydub2f5V4dx
HbX+lwjPOK4mx73alcbKAkSr4n7e1z53OaDORTuxEqpR+IynhLkVyOZrTT/
85NUm59vk6AZF8g6AROLkPfUrSEaR1TkjqVFXaSfqh6whH0rakke4/SOpH
v+p2bRf2B76Tt09kdILakzSKL+NLfb7msCi+T7N6nOvGDYz4K88DzvT4DBV
WzzKXKmy3YiiVuQzGTyTfIGgAXYqdNkk1P7UWO/VKqOcBd5KE9z/Zeu0
WFUQyBp4iLbYqSVo1H7u3+x1DY6ARp+3jgePzVPmOwLeyOtaZJGWqqsp
pnyhst5sktO8PQkw7txHJNwiawSc6HFtc7MSI4EZHJJ1qe4+83bWiua0nu51
BzRlqmSLN+s57kxLCUpKeAE70BOk6Z1oGD2K02YPoo5fd9m3AUcBRSha
L3yRJl1YtXnuR7YHFXrJq23p591rbV7ddMqGqdEsCUe0cBA2kUOy0MskNp
1MlcvH6xbrKk1W5KumfwI2kBPXCvE3eX8t2qfqZktHFNqRK4VvysbVIYhf/
4+STO3TU2+ChH23mVpX0jqQEdyFpCrqq2cVAP8vctnUuqX7gj6SfxQYkAZ
G3upqLVSOJ5BsEDWKAYqeuLSl9vPYv6D5CdZlmd1k1ch4FlsXwB2Cu7fva
jq8KnO8hGjfUiaS5pKKrljTlPGC+7cpewPk5VwKvpcu0cJuIR6f7TyAZjxxo+9
/52FLAF4AZtvfuJ54mEck3CBpAW7HTZ/stdpJ0D+kDXqS+2FG2gLY/VuEZ
nYQ85pAkEu+xvUk/MRWeuzTJiOP0Ctde7S4mEL3O1Y06e/kupuz9zF9kzgD
OLkuQJc9ZBNxF5+Rrl1hfKhlubNiuDa5kuHFVVbGPJhJrvkHQDAYtdupV
mFMJ2y1bRpQ8mw8lqZZ9sIrWs5JV3UdIVbg/AH5K6g8+ALiC1KJTRi8TiM
oGETXw+8L2EaR12srYHnT9vMXCAUf7j7cnXlhsuLGo0w1BIka+QdAAcrL
rSj+GBZJmplv6m+LM976RlHQfI601V3aLyvrHD5BMIbYkrVcvA+xTtc2mIP
c55hQVfI2HgYbsUzzM15Z0HWn6u1P1+Xdi5NudSL5B0CByO8rapLXXm/o
xOZD0IZKFXauw6WGShd1XK95/KamS92hSAh2F7cvH3DT6/qtsb5C3lwbu
A9ZwNnFfUplM6UVJHwa+295ulfXVHyz7/ZC0gB6V5jWO0KcdMe0cBA0gr
8EdSRKjuI2kiLR6lic8uF3zucP9hwCvBl5r++Z8bC1Sr+ks25+tEMYjpIT9jvyn
iIHXl9y/OEYnv9hblvTEOwV4GXA38P2243OBzYEP9brZ9muHE9b0J0a+Qd
AAJB0LrADs10pYeQ31S8Cjtvcpuf96YHb7SEjSDOBK2+sOJ/JRr1V0iIIRl6g
JdYiqA0kPMTJifBqT53a10Pb6Xc5dY/slnc4Vrtm+13nb7Uk9yMTINwiawXb
Aui5827b9YJ5Kvg7omXzz9WOmIG0/KunfVQLo8EFt0tTxFVVGsJPhejQsbK
8w2TFkeilPLVXh/jf3OGfGjqiDTCTfIGgGdodprjx9W2X6605JW3qsT+vrSa5
EVej0QT0L2FDSXq5oGSnpdSSrSZOsJhdUfP1gLPdI2tRtXsaSNgGqyG5+yk
uQv/ZUIqadg6ABSJoPfN/2t9uO7wq8q6zVSNJLSEpGFzPawm4zkpLRNQPE
tiZwlu1Xllz3XNJI6rEcg0jmCjOAt9u+a7wxNBVJmwJnAacy0v60MbAbqXf6k
pL77wauImlKn237H8OLdnoRyTcIGkAhcT3K6ORZOXHlSumdSaNOkSzsTu

+nYrrHs0srfiWdA5xr+9S247sBO9h+66BxNJFc2fwRoCUwcg3wFdv3VLh3a
eANJHnLbUhV7GcCP3D9jlnTiki+QdAg8jTx4uTZPo1cw/N/Y/tVfd7zIuDUsv
skXW/7Rf2eC7ojacVu+suS1rB9ex/PWgbYmpSIXwf83PYu9UQ6/Yg13yBoA
HkN7xlZSeqCwvE3A38uqk8NyHI9YuhkZD8LeDawa4Vndyy4ylrC06YYa4JZ
QPZFlvTzNn3t+Yz2TO6J7cez3OS1wEZAxyrqIBHJNwiawdHAHh2OXwucQ
nmPbVV6TaW1G9mbJHd5g+0q0o7nSfoGsG9LXUvS8sCxwI/HE2wwqtp5Vo
9z3R8grUHS+96JJMAyj1QHcG0tEU5TIvkGQTNYxfat7Qdt3yhplYkIoJuEpa
SlJe1SwRjhQODzwG2SWg5GawCnAQN70jaUgTySJf2apLX9PZJN47h0v5tI
JN8gaAYzepwr9cHtg66jpUGNEbIK1wGSDiVJZAq40fY/e90X9OSZkvYnvZet
bfL+qhXuPwi4yLYlzZS0/Hg0v5tIFFwFQQOQdDJpiveQYr+vpCOAZ9t+f5/P
WwXYAri9za3opbav7nLPQMYIed36Dtt35/3dgB1IcpmHD2Kt11Qk9XRTsn1
EhWcUNb8FPEQfmt9NJZJvEDSAvDb6f4FNSaNMgNkke8D32n645P4fkgQ
Vrpb0bODyfO8LgVNsH1chhoGMESRdDrzB9t8kbUFaW9ybpE/8YtvtetHBkJ
F0MKnX+6Ptmt/AJRU1vxtJJN8gaBD5g7Gl13tN6wOzcP4lnQQzijq/kj4NrGd
7N0krAL+yvWGF1x7Vy9uvm4+kK23PztsnAffaPjzvX2H7ZVWfFSQknW97q
7x9kO3P93n/pGt+L6lU0e4MgmCaYPtm2+flPzd3uOS/utxadD3aklxdnEetlbS
dgdmSHsx/HiLJSj4o6SFJHXtN21g6uzO1YijKUUb9yvgoruu+czwP6Kb5TfXf
i0YSv7BBEBTpVjB1h6S9gTtJvZ//A4tHOJUM6GswRjgTuFDSfSSlrl/mGNY
GQtZwfAw69dlN83tLqmt+N5KYdg6CYDHdpoKzBOH/JglinGT7/Hz8dcBGttt
7eIcV35wcw/mFXt91gZm2L5+IGKYTkv4OXET60vWavL2YydT8nu5E8g2CY
DH9rMNKWhn4eye3pGDJQNJ/9DrfrTe7cP/awGrAuozW/L4BuMv2TTWFO
u2IaecgaDiSnmP7z3m3o9KUpMNIzkPXSVqWNO08G3hC0s62fzZB4Qb1cks
/+s0dOA74tO1vFg9K2jif6+X322ii4CoIgt+2NmzP6XLNu4Hr8/bu+e9Vgf8Ajh
xeaMGQmd/akHT2OO5/vu0/th/MSlfPHyCuaU8k3yAIqmj4Pl6YXn4jMM/2k
1m/N2bQllyKP/u1xnF/VyMNequqNZ5IvkEQVFmzXSTppZJWJdnFnV8497
ThhBVMAL20natwqaT3tR+UtBepACvoQnxjDYIGIOlEOn+4Cnh6hUfsSxLP
XxU41vYt+bnbAH+oK85gwpmde6wFzCj0Wwuw7RVL7t8XOEfSLowk241Jsq
FvH0bA04Wodg6CBiBp917nbZ82UbEE04/ccvbSvHuN7Qt6XR9E8g2CRiNp
OeDNtr9bct1xtvfN2/vYPr5w7lTbeww30mCiyDrgbwN2tr3tZMczXYk13yBoGN
k/d2tJ3yY5Ar27wm1bFLbbR9Glus7B1EbSMpLeJukskjLVG4CTJzmsaU2s+
QZBQ8hOQDsD2wK/I6kQvaCiH666bAdLMJLmAjuRKth/QdL23tT2eyY1sA
YQyTcIGoCkO4Hbga8Bn7D9kKRb+jCiXyorWi1V2G4l4UE1m4PJ4/+RNLI3
LxTRHd/7lqAOIvkGQTM4m7SO927gyWxs30/Bx0qkatZWwi3qKEfhyJLLRs
COwM8k3UyWY1taAAAGxElEQVTySI4vUxNAFFwFQUOQJFKP7k7ANsC
KwF7Aj20/PJmxBZOPpM1Ivxs7AFcA59g+ZXKjmr5E8g2CBiLpqcCbSB+2
W9l+Rsn1a/Q6P6A+cDCFkLQUMBfYsbX2K+kl4VBUL5F8g6ChtFyJgOWy
+Xmva68iTS8Xi61MEt14Zg1evcEUph+3q6Aa0WoUBA1A0mGS1svby0r6BX
AT8FdS1XNPbG9ge8P89wYkt5pfAQ+TVI6C6U1UuNdMJN8gaAa1uBJJWk
fSqcBPSAVY69s+scY4g6lJTJHWTFQ7B0Ez6OhKBFwrqfRzQNJLgYNJhulf
BPbK9wdBMA4i+QZBM1iUE+hfSRXPBxTOVXEluhK4A/gRsCmwqaSag==

```
WH2hBlOQxyc7gOlGJN8gaAb7MJgr0Z5DjC2YJKpWsdueMzERNYeodg6C
hiPpWbb/WnLNkbY/PVExBRNDVLFPHlFwFQQNRNJKkvaU9DNGq1V14
03DjimYeKKKffKIaecgaAiSZgBvIZkrvAJYgSQ5eVGF25du03Mehe2/1RVn
MPFIWodUUPdK4BjgY7b/NblRTW9i2jkIGoCk00m2gOeT9HsvAG60/YKK9y
8C7qJz8rXtteqKNZg4OlSxnxlV7BNDjHyDoBm8FHgAuBa4zvaTkvr55r3Q9s
uHE1owiUQV+yQRyTcIGoDt2VnhameSg809wAqSVrN99ySHF0weexECGp
NCTDsHQQORtDEpEb8DuNP2q0uu34NktL6y7fvysWWAPYD9bL94qAEHw
TQjRr5B0EBsXwZcJunjpLXgMh4D/gY8IukG4HBSMr4U2GVYcQbDRdJ59
Bj52n7LBIbTKCL5BkEDkHRYySUXlpw/BNjI9o2SXgH8hmQ5d04tAQaTxZ
cmO4CmEtPOQdAA8gi3neVJa36r2J5Zcv8oSzlJ19ler+YwgymKpLNt7zDZcU
wnYuQbBA3A9jGtbUkrkOQm30NqOzqm230Fnilp/8L+zOK+7S/XFWswJYl
WspqJ5BsEDUHSLGB/0hrtacArbD9Q8fZvkEQ5uu0H05uYIq2ZSL5B0AAk
HQ1sD5wCbGD74X7ut33EUAILgoYSa75B0AAk/RtYBDzB6FGMSApVK1Z
4xtbAQcD6+RkLgaNs/7j+iIOphKQ/hMhKvcTINwgagO2BTFQkvQ/4AHAgc
Fk+vDHwBUmr2z5lwBCDKYCkp5LU0O6yfU/h1CcnKaRpS4x8gyAoRdJCY
PN2AwVJqwAXh8jGkomkk4ETbV8jaSVSC9mTwCzgANtmqA05iwFAyCo
Arq5Fxk+/7JCCaojdfYviZvvwf4U7YW3Ig0yxEMiUi+QRBU4UFJs9sP5mMP
TUI8QT08XtieC8wHCL3v4RNrvkEQVOHjwA8kfQv4PangahNgd2DXyQwsG
Ii/S9qOZBe5GUl0BUlPAWZMZmDTnUi+QRCUYvtiSa8EPPkwyUxBwDTAn
RklLNB8ATgBWA/Yt/Cy3JNkMBkMiCq6CIAiCYIKJkW8QBKVIuorP
uENJzikoAYknWX7XXn7KNufLJw73/ZWkxjewdgu/y3SdOQ2kxh
LUB/rFLbnMrqfd9UJjqVRRPINgqAU27e1tiUtKu4HSzS91h1jTXKIRPINgiB
oLk+T9HJS2+mMvK38J6qdh0gUXAVBUIqkVxR2Tyc5Iy3G9uUTG1FQB5I
W0GOEa/t1ExdNs4jkGwRBKZJ+QfqQVj406oPD9usnPKhgYCQ91fa/JjuOJ
hLJNwiCUiRtCtxh+y95f3/gB+BW4PBO0pPB1EfSPcC5wbnAAkdCmDBCX
jIIgiqcTLIkRNIWwFHAt4B/ApVUrIIpybrAb0kKZndIOlbSKyc5pmlBjHyDICglr/
hF0pW2Z+ftk4B7bR+eD19je8PJjC9YeiS9gCSDuSvwHOAs259a8l3BeImRbxAEVZie/dx
AiqsHTW+4UkPXhB4Vx0TUwDbP8Z+E/gaySzjPdObkTTm0i+QRBU4UzgQ
knnAo8CvwSQtCvwSQtDZp6jlYQpG0nKR3Svo+cBPpy9VBwHMmN7LpTUw7B0FQ
CUlzgGcD59t+JB9bF5ZgZrUZLJpLOAN4AXATMA35o+7HJjaoZRPINgiBoK
Llq/fu2w5N5gonkGwRB0FAkfZzeIhtfnsBwkGQRAEYY5B0u+01JjuO6Uq0GgVBEAUSdUPklGQRAEY5B0u+01JjuO6Uq0GgVBEASdUPkl
wXiJ5BsEQRB0IqZFh0gUXAVBEDQUSQ/ROcmGn++QiTXfIAiCIJhgYto5
CIIgCCaYSL5BEARBMMFE8g2CIAiCCSaSbxAEQRBMMJF8gyAIgmCC+
f+9ax9dTFEsQwAAAABJRU5ErkJggg==\n",

    "text/plain": [

     "<Figure size 432x288 with 2 Axes>"

    ]
```

```
     },
     "metadata": {
      "needs_background": "light"
     },
     "output_type": "display_data"
    }
   ],
   "source": [
    "sns.heatmap(corrMatrix)\n",
    "plt.rcParams[\"figure.figsize\"] = [10,9]\n",
    "plt.show()"
   ]
  },
  {
   "cell_type": "markdown",
   "metadata": {},
   "source": [
    "### Visual Analysis of Common Symptoms"
   ]
  },
  {
   "cell_type": "code",
   "execution_count": 7,
   "metadata": {},
   "outputs": [
    {
     "data": {
```

"image/png":
"iVBORw0KGgoAAAANSUhEUgAAAnAAAAJMCAYAAAB6q5t1AAAABH
NCSVQICAgIfAhkiAAAAAlwSFlzAAALEgAACxIB0t1+/AAAADh0RVh0U2
9mdHdhcmUAbWF0cGxvdGxpYiB2ZXJzaW9uMy4yLjEsIGh0dHA6Ly9tYX
RwbG90bGliLm9yZy+j8jraAAAgAElEQVR4nO3deZg0d10u/PtLwiaySkAkC
U/QIIsiathEPSwKeNATVNDwqiQKh9dX3I5r4Hgp6InivkcFRIPiwSgqETgC
RsBXEUJAJKyCEEhMhKi4IBBI8j1/dD1kmMzM05l+erp++Plc11zdXVVdfU
9Vd889VdVd1d0BAGAcN9h0AAAArh8FDgBgMAocAMBgFDgAgMEocAA
Ag1HgAAAGc+ymAxyk2972tn3o0KFNxwAAOKLXvva1/9jdx+007j9VgTt06F
AuvPDCTccAADiiqnr3buPsQgUAGIwCBwAwGAUOAGAwsypwVXVxVV1U
Va+vqgunYbepqdW1duny1tvmf5JVfWOqnpbVT1sc8kBAA7OrArc5EHdfa/u
PmW6fWaS87v7+CTnT7nT7dTVXdPclqSeyR5eJKzq+qYqS+qYTQQGADhIcyxw252a5
zp+jlJHrll+HO7+8rufleSdyS5zwbyAQqLkVuE7ykqp6bVU9YRp6++6+PE
mmy9tNw++Y5JIt9710GgYA8Altbt8D94Duvqyqbpfk+pVX11j2mrR2G9XUmW
hTBJyTJiSeeeHRSAgBs0Ky2wHX3ZdPl+5L8YRa7RN9bVXdIkuny
Ycvdj09y2Q7zfHp3n9Ldpxx33I5fZgwAMJTZFLiqullV3fzw9SRfmuSNSc5Kor
X2a7PQkz5+un5fktKq6cVWdlOT0kJBccbGoAgIM3p12ot0/yh1WVL9Tnf/S
VW9Jsm5VfW4JO9J8ugk43Vd5Sd6c5KokT+zuqzcTHQDg4MymwHX3
O5N8zg7D7D/ynJQ3a5z1lJzlpzNACAWZnNLlQAAJajwAEADEaBAwAYjAIHA
DAYBQ4AYDAKHADAYBQ4AIDBKHAAAINR4AAABjObMzHMyaEzX3h
U53fx0x5xVOcHAPznZgscAMBgFDgAgMEocAAAg1HgAAAGo8ABAAxGg
QMAGIwCBwAwGAUOAGAwChwAwGAUOACAwShwAACDUeAAAAajwA
EADEaBAwAYjAIHADAYBQ4AYDAKHADAYBQ4AIDBKHAAAINR4AAA
BqPAAQAMRoEDABiMAgcAMBgFDgBgMAocAMBgFDgAgMEocAAAg1
HgAAAGo8ABAAxGgQMAGIwCBwAwGAUOAGAwChwAwGAUOACAwSh
wAACDUeAAAAajwAEADEaBAwAYjAIHADAYBQ4AYDAKHADAYBQ4AI
DBKHAAAINR4AAABqPAAQAMRoEDABiMAgcAMBgFDgBgMAocAMBg
FDgAgMEocAAAg1HgAAAGo8ABAAxGgQMAGIwCBwAwGAUOAGAwCh
wAwGAUOACAwShwAACDUeAAAAajwAEADEaBAwAYjAIHADAYBQ4A
YDAKHADAYBQ4AIDBKHAAAINR4AAABqPAAQAMRoEDABiMAgcAM
BgFDgBgMAocAMBgFDgAgMEocAAAg1HgAAAGo8ABAAxmVgWuqo6pq
r+uqhdMt29TVS+tqrdPl7feMu2TquodVfW2qnrY5lIDAByseWRW4JN+R5C1b
bp+Z5PzuPjnJ+dPtVNXdk5yW5B5JHp7k7Ko65oCzAgBsxGwKXFUdn+QRS
Z65ZfCpSc6Zrp+T5JFbhj+3u6/s7ncleUeS+xxQVgCAHZpNgUvyc0m+L8k1
W4bdvrsvTT5Lp8nbT8DsmuWTLdJdOwwAAPuHNosBV1ZcneV93+b3v3bZu+ww
rHeZ9xxOq6sKuKquvCKK67Yd0YAgLmYRYFL8oAk/6OqLk7y3CQPrqrfSvLeqrpDkkyX
75umvzTJCVvuf3ySy3aacXc/vbtP6e5TjjvuuHXlBwA4MLMocN39pO4+vrsPZ
XEXQ6sKK9ZLJlfGIt82SnYcVzQ4RZMhpNxYdHFI4Hc/jE2/Dq+h+/fHDIx
KTp+j2YjwT7gJF7WpiVE/cvp4CVmPO2cgFjaNchQuyKNYSSH3PrdE7y1i0+JPN2AtE6bqCb54UuH+kE3f6DTKTpvdtbUQ
iB6o3dg5VHhdfTEbgLj1aqn66FLHvLnHNG+/7mUk74a8OLSlscldjGrQgaYjgKrkiKaGqZbjUrnEjmNJAoAfYmvdhYlWYRA1cjfGUPEjJvUdBuC6
uACRp/VWWwGYt2eNPWEExpu9Y2t3UjlTQuaSiaAF4Bq4AxXz9vw/XP1bm
S9+/TyG8W1aWS2sgwnvz6f1dnvVtQ62u+ifyhhaXIAs/z9g6ghVWKmpNGq
tdJ9mgv5+nDCEDwTlKsjy39+6pIkKbqwlXeTUEM0o1IR6JAyqrwLe9utu5ivnu5nJRuTgwLXGYx9iJzjrFL9O7sW
gqt0cyxvXI0XYH69XR/pl9YhN5y8vXzzyqfqR8zHasHlIsOwmdRJ/YzeE1DTa
Ngm7QqV1mSqr2t7WwcZ6mz6ZQYowPKu+2UvmiDRRHxUHnctM9CUs4Brku/6RwL5/6ni8HnMzC7Y
Vf5+JbP+sXxn7yQsgYSaSLDQ2M6q2qDMJ3Fm9XBHoDkL0bDKe/Wnf/4ev+rl8
oWp6OSeRpUHiAJ7sCPnq7ZsuWaDNx2xy7kCeenf4Aqx8P5OS2uWK5Kb0TcdG
yzN+KqDfEdhjGSuhCtYbZw2JxVFw10+u0FoYcdc6NObUJxiZlTJL2dkjKLvYN4
gvrRNCMq7zPXzBW+sFv/yWcOFzxJ6+nQV0EV2Eo8IyJDrEMTcOr87K0W+
qzYFQXZvMdWTwcXmfDCQGwWfqDwWbt2xShtE6wHR6yaHHKfJp1e9Vdm
DKcKWSs+sxVR6ka0fCwHaGPXpD0x6G8NR/hYLnzlZ/oOAhfhPNbd27gBjVRgp
0S79/ZNwE8yBE8PuSGzfNpjiVRkxiNW8VX+c9OFUuCsJAmrPcywI9ljqNNhy
7JK6xF/Gl9CUIgYRlpJQTLW0svIxg+4hW6xuHcXNGbeSBO9sbWaFXJxeA18Jet
ODAlkZ1GGglx7cGUqyMv66tvuaoGkvl1cqiHw7qgpH4u/DLWj/tuHFobKV5QEsdS3WMeYGANd5r+Ycll4nE3/mt
b9/b4UHL9rJN9nNAmgr2OSzmL2QaX7lRAHMcMCYY8Uf++ZYUwKZXrJt2CJ+3vqodcqa1HIvOpZDcS/GS/hjNjAW9sWD39DgzojnStvLqyHhV36Ft0zeflHAsggH2XDqNI4LOhuMO4pImJ+TytyhT2CD4AUiz10+gH1E6uvFKOvVH8D3VnGcnD1Mhu0cx09jOSGIXjFcjHPt9eglFxXW5/9iJSJubxHq06RRc7wwYRcdGJJxDoE+vHBzI/P0FuP2QvaGEK1F55Xdcu8rmBFRg0GaSrtA9P6y/sKQcIeG6xngzZ+EBfjSqMbyHGA2JrdsWr77JLiUyrMD72GsMwvGBRVy+3M9KXfbmcYeSkO8IrcEeZXhqanWbrxTMHdwVvgV4O0YmrK3hmVazxS09qtQYh4lB0cMk7e0VxKCTaSX4oSfC4ys3sOdKBmAHhJpi0ovKdTVy96mmq2eyHemvlfqWN3EPWjsPWNZXg9Bu/uOXe/w9ulvQwBsiodJwkL2mycSdtqUtoBCqmoK1RcQswRWtxKNA4wQdDA6bpMdkzSYlPv1+5dA5XZA/1TnG2CaLcmW+Hs/tVqbnkmtZahWhD2a8P6iMyeg8wM3YAY4Xsi14aT93eAgbmSOaavw8JRFUmXkQA5/06ZzMrKGq92ocMNsWOhbyMLFMQFNpPc6vKwKFwVjb6jCk1uFVvWWxFrfI0DORyDUC0r+0mVe95e3ItWW8T10Wio6ETHM9I3t9tbs5bpzVS9+0MfiDNtuSH9wn9AumPFHbEYzd2+aFyqzGIGxKQfuLxhxgKcjw91ZR9JjPiXd9JNCuHO6THDXQJj9DFLLGfhqMb2xQF36bzfY8DlYufuSyjmm3G79KtkJZzl00H90NYTzuSkqHUyK8lZPhGWwGiVZzQEx7/0fuQ39Mm0hudu3XN0/YDgkmZXcwpGC8X5xuf2yb2LN9bCmvmiChv4nh34Qm2VcQpt+OnC2xxaYVq0kDVTlL9ZTgEo0Ou8DxnU7mYnMfjYSRx88xrjlBkj6b03Q0v54HANoGDTC4q/AvDsI2d9RUeHG6RwhJJEGxGY27ZdM1WMoCZMJnfPUMdEQOq2EH4+fUbOZqKFKCLnIwi2jt3t5rZfYHODl6P42+2cOVPm5d9W01RobQo6gy/gKkbUGIgRk1hlgtxsXldq/BLotwObjb7xSMBqbnyOgcn/cwDvb+tydtgW3EPJDkuF9cvJ9n7iktEkgmTvN0cMbcZcIIcH7l8+NtEvN20LDLvNLLCJnrhGc9LcXG6Qmeo7kKmpLaYTi8XJn7CmFGlfB4jOojLWNcFPZYTe6oG1nRpKZ8HOE0lVcbTA9CMgeAakiaZlxb13n3iRN8D7Lv5nwa+w2yk4Qrf6hXkV6pmRbOmN8bQe01ccL/vzBpv5WjhB8Nrg1LF6T8ODx4HNfQGlfkmfTbHY6P9+ZyP3GnA56xCJNSOHZqLGKflf3A3KsmqtwnsPBjAWpjAPfDcZqW1kmvOeMfOPmytP1xtTgqubzx+ACOgU5MUyynGvMdDr6QRjpFWwSyt3M6dJu7FcrHJd6ZvJ4JSRwrsfVCKf7IHqNcLkg4VFWVPJY6aYNpIH+Ff4XwrRI6ONWsTKjeCX0KdHHw60qHrgzuHYWTb4Y9MpVGYt5+2Oi8+Mhhv/Ak4RdEQfo7QWSaHRtTj7VyJ56XjMsKxmIbD/1eVtUF1FbUDO5e2FD7YjpTnaqpBlcnKjrTqy2KETnr6F0G3UJE2gdMXmjDyKW2+p2jEbwbWPxJm4v4a3HQHIP+9P7CHSUtCZAaNm0eTd0o0Wb65kQS7f+1GHk1ssP3S1UJvCIu1htsNx4YeCqXaJlLeGnUGX7/xtJtuWCnnOYKJ6ifc0D+YjFC2j6D0GzOEGDj2zqoSqbPHqrR/LzExGfVAlx4ZwNb2LBM7TgPvUpH9PDpzT0mvFQvR/QPUl2zJyL6QY+oLhXh5hOVTsQM+j8DQD7BEF0vJ1/5yKSKrIewVFRXTDaC3JRoe2VvtWmpbBgZaCJBjtpPMoAiXmkeYNw5F/LxzJyo9jBpCt5I4WkTEZ5TkrJfPVxc53N8HQtaZOHSeqNUqFFYsLrhUR54dR/YmuWZoCr77S1GBrPRz3V2hOpnkHr/RJq94jI7Zey3TUqdPzHqi7OfZkqRPqOvKO06NzaFJazAiXgGcQm2xaVdR3rsNBqmxXRa+GWoK7KOWt9dTUaaVPJdtXQOUX7h6G2jY4HR+AsB/YpzpqZgFgzvTEH1DzN9EHfRsvPv2aeLySfJrOpsH2pNB+72UuV+pD9Fhz+Kn0E0kXU1RvWzK6+4w0Pn1fVn8yvd17p4CzVGhH6RmgxWb9h3gWXUTb47fXmNSbxPtptWmrVmjYlFDRDeHLQ8QgqUcqf+a6X4a2tQfNB/wKGDkULDBvQCf3EA8xY/UvnI1n9/b00y4ymjTf+F8fZW2rbxFk7YFU5g8+V0ecz1SLQK/Wm9Bl8Jz1ZnMbW2KwLF05hk+E0vQVkfYfSqrjY1zwN1x5bnyYbPBrhvdIVZwlV+RYLnLt05Bk3RFX5BTWr9XoqNfgYRCrbpF5jCEfT2zlfSe13zLiyx0BAoYIf6xgXS2Mou/kEi5hqeVM1gPSQqL0tyW3xW5Fme/3yLaKmPmOh6KplEF5nMHpqk1HXpLDP5d/i9rK5UUPi43Kr39sE9nhVi+0UhkMo16FH5jp2jgB7N3HIVS2U7ExmnmYWYwRzE98DOWoJJptWPkpaONxrZ9+HrXLaxQR5MTNmmrT5m/tg2R6X4iUbx1j3g19+oC0/gJVNxVBtq3Xy7iSKDX+vMoYR7/xJlkobnoZ5MvqLmT5G0P2X9wSfF8uHcSNFJRCOnGeYbmHGbcj/7Ua01D1l3nu30I8ST2Pwdc1v6Dwp8d9RZPmdP3oxlVUtyxjsU2mqFKxBtrTTnWOCk1ccP/AjzxuV2u6ypj2pCjwDnXEEQRLXiqmwHsy4cUibUc8uhHVBq5GxLrK7z2hhfAX5L/T3cO6ZCvW7ph7FKAdFbQNhiktKBJyfaY6vXnUdEvXYrlSQ1PbLnHNZyc+oZyVs8Nu6g+PCgr2gDhCh+ixNJWm4JEc1NDpqc/0+LSowqrEG8y8cVmlXpa0l5AGv8NodAjFqXX1SNDHFKGPzaXdaXR1ac4+wFV/cKcWNTaR3yWAxwRobJcpGrsAHGmO/SodYnIQEeTwd4i9q3Nnc0pBd/RhHXQ8cIqfq8Fqs8IV5b0uW0pyMJ8tjvqN6N3vngd+LHLhl2O7ZqY0HuRjJDSdAbMzW5C29Tse6e82mdlTyIO1bNYMfnPIsRHt0szK/4qZYItb2RmU81Ju1D5S7Yo3HSHz4LaRdHSuL0gWImwdAFLW2zPJb0cbtK0jmVbTBrr+v/XvyfKDnstLGbg8RqVGm2/Wjjk3pJr+RJUbPbLVfEbyBBwYDxmPpnsh9p5c8bQkcubY4ddHR1q6WMKkNHS7WNwimNuK6DLttQwKDDlxvRfq3GQ1mhOJF8apZM84CdC0XGpg/mOdDt/2Sor/1b0qS2m0oPAqE+fSDsudbDcWA2WOrRXB02kEyDFM1e6TvBcx9FT11nEnHRfi5Mj2kdOpJsF7Hg+nd/sA4JmgH9FBdr7ls+eNZUMBaLmhpWRdYknQ3Q9DlgA+VwyP6TnsnbCmNcZL4V9OXf5xJw77/qS7z5g+ZQ8H9PpRyRdyqRWKhUlQ8xqBlb+LZl+1DuxApD7d/IMHeEYBmv1FoobzjHNoZ7dkt2vxsB9Grfe1f9Dng44+DiJfaRbWNE9GrcI4jQ6EsZdEhO2T4xqTFSRSVbfAIdanyN+n8LnmCXZ/vuPTBEJtD2QyFh6tffiXUN4NcAOzdKbhKmi/2f7qk2ND3sdf/K/ZUbavgOpPGnazyAtL7ykPOOgrHvtBmVDNE/Wt/zOuX01LWr3fhK0S/PRRF+mZt3cP4F4UxaM95EEbh+H2nfsrzqcZF0y63GgKI/5Ko4KQHqtuXc/Y5ooetA28aBmmdXNKeAxO8DLOWbqZQL9KnUKkQwDMFS+rLA7Q8dbLtoHhmGwEvvawhh5pVmHUe9n4M1FcB8uvDtrSTj+lY60VpAyoRcwP3JBkqCVIhNdNNKTgqvw5C8QbtM3Md3rmGj8EB5YgEZ4X8FDmvHVFUqgyNAC2yTbqxwNE+3SS70tF7uVM8k2DwDLHBBn3SQHhHwdKhhVGrhzwqUGwI7O6Ibmq03TBoQEpPwKu6ziMpJeRqjvqKGVb7/FwxmXE+YEB2bSd5i4HjvFrZj7e29TyWHwaLlU6G57yWg1GVbQ6pEq1YuMcTTqjN2MFQDvvpamwqkjOAbbJD4TZr/T7kqPmhbbzFnM3WyYbRWejhK7+K/0cNOtDI7KOjp9PGJ2Yw2bnOK0lENjrYbTKkOjsbHvZUvD4Vf8Pw9f9xvQy8XXWb3smLNWUUn9kQLTQdmAlDJO2SKRr1LyY/j7hg+vW/OIZ0BWeEA3phjo8FDnlntwnm2NBrIubYKGjaUkIBNMFGhTWRJzT9DJ8CXXAUFhYQB1VUhEZ9Xs1PnoiAZLjR/J4Ki3jpZfPvjkmGxu4H08mLc2EoyPYMShMHMqDK90s5eFPqVqIzuA82tMDpjuDGsKPOC4h/DcbQe/Y8xZUKqtCy3iBIZvmpXmnFPB7/x4rBe2fShRJffH3JS7DwMTfGmVBcpsyRjr4ygOe7C2d1UYt9rKqJaGG0iZuePLdj0ctxaFrAz7CfTlTYRGfOuxVrGHK4VCA0/J5CtArp7NmyMB0e2Qb7/CWHJoLs4x0xK5UFaSNb2jQw2rX3GnBhkbjhHvP55v8iyPDI6K38/A2rqjvJYpKHbi8s+5uSThzk51uq0J0hBeLJ4VCgLdYGJbWPmFb2cl3HZcRxA99AJ7hyJhctNhvV+iA39OMvT3/D2SaTIlfLv1dsmp8pNrhOhSaIjStrCqqjqvq+WUBwhLk0KhMlq6eHM4CudIEbQs9/5+IfFxTiYo40CTa5IsWtMEIUXJ5n+Y3iC0f3O6bnprNd2WxWP/Kl0sryVtfwb1YjDZxU4HAYfLYDsFoz26mOiCGKeBP0zU76RB7pjlhWd4DZH0EHwB8GWGYc3lU7TC1c8snlJ7/jqSeaYxtxXkOjRrWXVsOuQ0ey+EaEtpMnhcyHrs4oUdxaGb9ntptngAs0MvzvYHwcGzoCwbJiMWG2n+IfNqxBA54R5zeIR2wdBtdILA+YYybF2vIQpzv7cCEHHaEEp4H9Qz1+NlphQ0WwNS/mAXFqVBHGuEH5g1N41Yvgp8/OEoyMp3PCtLuA+JM2kNzxq8XD2L8l6w/Fw3a/dgTIY2qGZkQDtCaAK59z3m6Q3qDLp9FaCLBtlQCpz3j0jPJs9ApMcprjpmhrpAyNnAu/gR5mNJqwnqWpFW5SFuY4ZEiGUGVsumGH60RIpL9i4FzWMmP37kJjIOTa0yOiWkG9EjNMY8mXvaS5f6l0q+9GhUFNCtdGuGBhMA/Ggyi6sPnu0qnlSbijSy3IQ7BrB9YkRWoXzeKXUyvDcSwzJYthrHtDo8vS0TOvnHc9cPcyUO7fR/TP5FQa0ADFQx1ATKu4kJNdgA2M2g6qE6h+7yRdqv4AJ/7Nh08L0/L+fpjmUAn5UAHZVLAiU6MuHpu6nzWIJmIgsO3ZuGGA8Gf7RaZqOjqxyt5HvxqfqlCV5A7Zcb8uBMqC7ZMRM7ypPGsN6ooiOc4D4HtgPYdHFsAjxTbvgZpLQBtjbgHRFHQRNkVqyUSi3aUzWVOt1JJQmHSpUQ9a2w3M7LCjjLUn4sswnOUlPVbX0Lqy8oN1S6sxrBM8H6Mgt5fLqcfpAK+sVCxd8hQN2HqlADp2mGqEOe9OP5uw8Q/d0EtVfAvk2Bw6sBfKRKKW0HddmdeKxNJIS88vdsjNIeKdcH4VvVRb7NP2z8kEWm3YpPqrrX/ua/mWwsQNDDjqQW8tJC8TFZDDxrJZpQv7xsLR81oeYDeGtNNNyfpqbN/+l/yQ/N5zfwu4GNbWbALL4tOJKXNHmmNvL/DZtI0uSGWPWJZQGBbJ+u/Ao9jW5fphD8eMKJoj6FRwnWO5Ju2Nn26k+bkmWb/47nGSZ73YbyXhS4Q0IDf9TjRKrmgN5pbPzpKeHHTezXbCVyH5D39mXw5qKtc0LYx7r0j4y93LftVyX2hdHKC1w59OIo7tzdk1Rbpq2Cv6P/8YrTKhdpUCTuytTJHqO1r2K5oMP7MOg6NH7ahhm28A5JhVqdCd2F9tPrtEYH2ltO8mzZ9yH6oE/3IgSQ4AlPc7/TskWVnMPUzCD45odPuOWQldvMBhGHQFAZDr4Lk2K5WzlIY2o/zlIX3jvKOO2/w/lUgRtdp6PFfcMlfxXWfKJKlKKqt+NU0mG46q/FmJ30MZeG1R1R2ObSYWH7t0g8r1xNKTQDLVfzVEWU38OhUezQ5y+WsH8XFe9VLKhBaDkYaTS+qD4wK7fS1I0Hqq/n3FlCl8Ctm0x31XA5ZzE2nuy+QVqkOWGrHT8Jn0VZzu6aLLvmYlt2iMmQ9C2NNf2hEmQp7bErjvZIjPrjEXVk0/MNXEBDsFr1EQmjYPqsLC+LOyNk1P+EgfbTyFuEiDrjdcrMoGpMtwX7Hy2JHMTrCqc1Kx5XIz3Do7VnIPHzYB69lxhbpHNNqsqH8XUBUBWbXi08F4b2aHBFHSGlxaDAVmEAYvaEHE2LGRJUsH2HA+u1u+pQ7FwmJR3n1Qd2tBwIxOlRjIKZHmwVKHtbbfRpTM8eXE2gnnxjvfKcRYDRXQUFg8VkUB/S5vUUAIKsrZJW28Pa5tYZ11Ak1BRp/gnTCL0QWtzPAibk+7QCFOy7mRdV29a/c1wqAqSjv8mhFWl3OMpWEljL9QcwKtgaTPrPbSyiqyeyG3Os2sBhSuqCIP2y4Vf8yK4sDTxxHd1XFH5uwA4vAnCp+L2bFhj+1nV5Q/SPgO1q8f/yCavvzDdEgEtlXCQMvy6dDpnBeg5NTSzNIlMEyGjn1xqMKM1VSYiRflm3ct0O2lSbzAGQhZQaWqdiCw0I44dupUHxvAq0bmPtHo4yb+AYdg+bQ7A44EvO7e9QBz6JT8ZkG02bFGNi1Cm7JrSyiM/xPu6OrvT5k33Iv+3OP7ohjzKt/dJ94MYXuLbiOGeX5p+Lj5WyLzNkT6SsHjX3LSO+ht9fN/L0MqNrmxJmcu9wd5TGZDNQq7PMRjD/6ajnXwHtvQ4rvFYN7XxlmVBKpz7ozc5g7kFDpF+5Do+7YNHDlgvHQpZ6pQF8pmJFMcijx7n01n4Ieti3FcuVBpXIHRdn1cuCOJ2+OxRzYpwC8kpNRYPhgjBzYjLWjaIjJH+D3CaK7gOfpxj5DGiwedh+QlKSOLl03EaYwxSV8X7o19rS4n2o9bHXFZUfJxVbwOGd8+MUIt1l+OSsV/vzdS8mlUDsnxRgz9wtqn0Rr6lCsMS+QbNfD2zDlm2i+O6UFq0N8VKwMc3nBTSjgeNh6XJYN/c8+D5cWbYaDaoRo+OnhfQH4vKp6Jhy3sDLwvzCmzdItubUIEU0tJeXl8+fbXiL1bQnPC3EPazGfpIYqlt4F3TrY1KE3pthW5KUFGwpE0b6LNHrAnv1uYLEx8rEqF4mMoZlD7V4u5FSyUlaeG8h1TvBKW3vS7n+4hMU1+P7/QZ8mFA7JfvMHnSv5lgzPV8SHzN/B4aGJ8P1fPY1Ud3rmQpTJdI61p5Zeu8ZDkfQ6pL8J7EyqMPxadyGq0MBz+TYbXp+bjBnBzy3DFS6H7yeR0ObrsxcrnGt9lTK3Aoy3bxbTwOe6S2Xj30l5iwxMM1m2NpKKwHcM3mmDDg0wTs+hb+c2NUAP7YKrFVYp98m1ey4AUE6QJG3EV0Fz5TSnWPnkDJqgs1Jc5jVhdULEhkNVO/Wju9UtIczVn3iG7tHHmQdrFRVFJFH3hB7M9vp7ZmUeWLnDfLw82kCQAPPt2k4rzeyaJFOF2yiW+WDBCYaVOI9FSSIQ6gHDddBw1eAU3ho21AzEEjM5DYmYnJRAHhTmfQ0W+FiFKkXYgQ3H0EVakk1jafADXEnJHHqe0hYpFtoRCFjXaS3tRgDHZ6xBeu5J4jZW9VqddtbA5zmZxmMXF3iUDLTnzYYvcBhbfGTjCj6Hf3P/hmsdpPXkZ3WiR3MKw7GDBFiFzhgoO0UuNwspuTPQbxl1ib9yAr5CMdoVMDFixc+1zNWB8qnkCNbxGLPTtzDg2vwLzEXkZNxYiGM3pBavVMz6R/DMUaFo8/Fu+RhH4Q7RUyedBxAntzHTY3VVL6l75R5WAhHObrTrnVFMY6+uP80S2PC0hpIYmJXGuJOpwZNP3ZvHu6VPaNBR6ts+VH9omNeYZAUnWWNPOPczv2KZEH3l4uW2r4O8MkBEgIjmY3lqFxr0eOBXnAUEu8c9i4IjHiZRHWzWZV/n0mm7E6Bdkz1ocFz3/4iXtmOKVwaxFUY6wW5wsLX3JiHV/H+VkaBT/FvJhbyoNLj76wdqTQBmbiVv19y0I3tYxWbpbSRfrB3HiBeYkGwCMIyUpyTGJbEhVCTCPRvQTtUOvhKxd8LB39mOeE5kuGQqfIczLQgVt1oQkJLyu5URwMI6SfBCHwMvIIcpTrqy9pgk7OdiNRVeYjm7q53UqlXozW6SJDjfZ0Jz6tMvK3dntm39bI/QkOr6NZFHoQXIxWd9dTwdUvFHHTX+wJRnZ3FAmczowLV0Qkz2VBS3Y1cPnDvkWqy8tvNyG7bZtY5cX4sowNP5a8z3Q2OWnRcCWAZaNVutdB6UAw4s1sKXGn3BdlIEy2qHfmlPcQjA9B0MtxIF64RLyx39WG1zg0egyKG/ZupmjM9DM9d5wLYHPbrGR8eEaCdDA9ROvYrAO73RV6Mf7awyQ9t86fHsWXlwNYNT5Nq3lm/Thz5Mp0Q8T5SJ4YNr8LcMoFJDbZBwzA+BkUjWuZV+TSS36pyyvtB09aSh94/i2LRDJQlDX7N4BPbEkQqe8CcqiYAQTQ1DErgsi0G8wlHdDo0sHzPD7B9vwZGPZrUp1bc0zLNJ04UFaNwVZF6zuqBgxgHHMH3+bkSLTh/uWj9PSGRbG8SNcq1vs1jvAP+T0Yx4fCG2jbvXEhMxTT1Fzs/jugeSX0aBeOiFpEi/tLfNKhhTDkhRb4mNdpJTfJZZk/YdWTPmcVbL42/KA8QSW47Hf2QW7JoXdP2AXR+gKdF0f6HknG6Q3SPIm/3eJ1HOcyzEd8y1d9gU+8V7SzPwIWCvCgoD1QNH+h2o69wsCf6ZW/WTW2DgO3A7Dm36Rr1bgb5gXqP/7Nxh+pOMm1k1o6OEhYMIrOBJFZOEL6RaF3UVtKeUH0SQf7ChbTw6lJmbOPjMAAhYcmw/Ps7+0xqzvZIz8uw79LHJo7YHD+yd5ql+IJeM+K0ynT1j+1yvu3xMOHqITr39DvGQJbh4W8mvWOjNbPFz6Dz/PsvvzCklO0k8S+xrE3ySl+QsmWdZ0RmmQ55W2Lw16uSHnSZ36qbeOcmzGTPZZOuuaXuJHJeYTNdKj3MZWEJOuBJjNXMBu19bO5jWm7zHlFZ2VVpPjY0kqHX8hEWh+jVazJWfQiVrXR4oJKt0HXygxRhT8gsNLlf2GgkKHHkNm5j1+2wnLNnnGyGCo+g75bO2AtmA6cNM5K8v2BQ1Iac4y7GyMjyrCV39cL9Bc5vnKiW2ZJbd8FQ+6kfe+OrT5E95ukwAM8zZ/tHNM8Ta7rXL7dSjbK3jBhHA6lglMFqgwIVqnjgjvHyJoOHYlQNxqDeC0ibvCvM/6oHg9MNWBqTLE4dcNIKEtlg3EvXPIfrtjvHE+vb6xdI+SaiNSEpAMefYJ2p33/TGyKCg3z6ilfhYz+pyaq/TJ3VRBBgxvJKXsU2ZNL8MxAp2a4dGkCPkHiMODLQxtsLbCdzU4KjR+A/kEapVHPYzTXm0X2u+9qhGUJu8BWXX6ZoEHtWf9H1ZE3SwjbibCpx6dWhrtGQWVpSIDgvtsZD5rdSORo+/4IVXE5vkJ6XnVk0Ac7RFt4GcPKPvLwNsY5Z3jF3I4PqgYLHptxYXIZN9VFPc8EiWD/vY8O+HGI7sCMLiI5XKu8DQn9Z+nvs64fHRBrfnRY9bStNq31QGpIuwWW8FKP9gdkHJFtYBINbbsiubuhcgkxlkfi+rwgzKAlAg46Igs64yBYyRkbaYtYZD82oIrlp5mgOSZ5TafHhZBq2rGeo2KtIgvRDkUTyPVY/LWzrdaZx5NpprRDZKTQtaZJv8v8CUJehCYPBc5FH/8zfrSz3C7f2Rzt+kCnbmeJphPNn/rtTvmC2jY2I6ha9tnuiPT+kD3Fp7chqBpFXt9F7y62H35fyrp0gsSXTBWL04fXl2xsaBRB/KKcQNNVpzaAp8aG8oVohf44ynHJfedmDu5I7KpjVPYNkoDdjzRDmKxJ8HcbNUEunA8S8/c1GrxuJKqr2Uf/jBHkbBzelQMxzeuxGyGNj1KPcxlvLdBfsaGkI0Tv4VbGEVXNFYUZdf5qz4eUntXrXvV1IUpsR/ZZQ7WZvpeK2EGAP6lQf/4jw5brgD8okUmoDxV6JAkwI35owVsaz8aG7hHPdIiG+OhUvpOXmsGZbz9fuCOPBbcIiTaHLTsdhAeAtMPPqFA2yfIPxiUyuYj/lCE4tXF8kLipsx7BFmOWOiN1mhXHx0ZRXanGl1+uyqE9bmjoMekzDDTXLjEMvqpxNyCPfRObv6lQgmf/iGsi7qqrfgeVUhZ+sebzfoLmZp9jsKcHd7yj9pqqDZ/WjbtY+ZzOgy/pGVJBm/CntmcYexJxhLxq1AlryVBR7UFpBDN3k8E8uHgPn2wtBpTVnYFyOO92i/KrQKO3gqaqmEcA72B07B1uCu/QPuIWZ5xzUfTzVZqVI4qRyKZH5XjROi7/ZvyzDspxH8+H/NdHKZa7t4NTbOUIF+A4PmeqDjV3SiIoU3IrXSTUESTl80o1ZDCp2upzkEOXRg3l3NoWfRvVtVp23q8Y+n+rmhG3QcjkaE+uHEyGj6fJ3j09v3J+BkQavCrJ4lqlf4z6PmIrgyowpTWAyj6/CV81ndgwg/h+/T4CdVy5NbZaZwvsUUAdM7oHSZKSrJGP7hcAhFtyiiB+QQ6xz/SyfE0Ks9aqcSD7ixZGrMxUGXaqVoNzOIo/PJqWUddumJFW8eIRZu0F3N2dKM5zyoE+9jwFvgtDVb5Z6LNUH0ydq5vqx4MyvHrqqOXe+EwjYBTK1/MAcR2skKpG5s9mP/2SqFHvXpvdf+Dyxk5Of89MYmI57o9gyKfmBCj5GJEN9vDbBJwYMvCRuT9iiyzbyEDbb5CO7HOVkAK9lojoRKL1jTPGyKxsFVQzx4cD7XKhKDb6AHoZzu6mewQCFbP4gIp1EEU4CrTUO2AV5e0PIS7FCa2a8UtUcgZrhTT9XPMdSTDwnV5Z53cnX+KFm6pvKoERv+RmK3w8VgvT7kwcD09WbjnnUbm86rNaA/gdYs1Ai3wxXrGoS/bsk0qBQp/tm14u8pDnQ7EZMBNNBk3ZgEuPiNJDiyN1JLOqqEArDKlxXiW08f2PiR6g0zjl7TkZ/yQXO+WVddhq85bmz+oN86HBSmq2eFlNrVy0gwa0XfqHGNeP3sc4KF9XGFb0nmZ5WLb3J2xwvMNqL4E5RktyhO0xyWbSGUIyVkzc5hcUGpgjmDZEdd4Dq5c1oCnv8OaslP6bgN5aAhwkMoxGdU8ddSiQuBMpDFVH5BXNMw0Tk5KrnSSiB6btAHC84ovFGqm2rRsQ0lsM6rI98t7x+NpBd1JR7r9ETD5Y5zIcxeqyJt2CyMt3fZL3TKYNWvhDqYvpr8gqXJaPjpLFqsLt41NbOR8/6PiTbVW/UZIF3iDOq0vnjMIqhCMqETGFn0xbqEBRKkyc1SNvxNcd7uwqgPFgsRQwIVKJV9cETMUxPUwAiqK3yGWXP0yvfVW/DV5bO1P1KXy1shmLbgcftWpQxfEYR4iE8YhG+UmcG7+X9lr+3CTaTtKpgXLG7Pqv/ymJYILFYjtJprrEcULZ9VXHpHbzgRBFAY8q0lFEcnBCCwpz1yYrVjsYw3j6GWxH3bx/jBJ+W1lngPHsG43P8nEvW/p2wrdEuOHCh3RazCEUzQXdAmpRXE+bl5NGLs2p10UsGGoHLovp8YSOQC0sX8Q7eK7r6Rx1lR9MXhnxrHSRCEDjiqO8MO6LbSIyl2qDkyvfKy+vJbZLk0+KzgaO8KLgsG3mXLR4AEqxYeN2O+1ZPtZb4JqHZEArmpTSE6blzIGtz1zFPGNPIRRjwn9hymtvF5GgIt0RzjCpvQ5hVN87UnAcjFY6UzlnbtFdg3ABjYFAI2OULQCF/dUEK5hgZn2H55jv8Yk9w5A0LcvltU4oDhvHRu5z0cHsZGdAHowqbljA17YkOZv4uCjEPFkUkFbukoScBk7VBfoOyfgffu7bBX0gj+2oKFsNKh3WDfRJ59TXTU6gJtPZx/sXgSNaIsEyehAQdJNR0EmBfPR3eGDK1GGjdZzR27VEd8dGFE9ibqTpnp8ZPCv+VkQmOQZdkZuvCnpMStIN7SMbmlhqcWpBB/SNiEwUrI1aT6jEQaKiHQLspFJZRMuzzoa2jNYnCt9BYf9ylQXSPxZ5sOLDeaCmNufnTj3BBCs6oRFNRhKgLAxr/bLb9L4lSJKQaT7SEeq24yaLCfsyLZgV3f4aXZLDyI/YrQyMnjbRzIFv9yc8PSBE3SvJu/rr6xkEzXaj2EDEgXVf7otnuTPBb3H6lw2Pjq9kcnOm5TiyqgTYuiehvITbAA5BBIAM9vcYf+GTWe1QKMP9/n+BAt5V2S7KTdA3fpm0znOwzG62LK5mpgVPdHSiaY/HeRVx+3J+5BPEXSzIFn9f5QUDsQ0TR61RgAIxHLPLB3NAq1qs+m6A76EUkJ/ys/r2rtM8kcKoRp3XHnqWH58zbyE9j4nwdnGwDmpUaGMdiV0T+Ud8/LxGHfH1dTPS+nPjuWtfFSlhe2fBkbbzH9GxlvNHf4+X3YN0Tn5jtLojwrsB7oVHEA49aEIv3H3uRWbppEE7ahv7IQ3JoUwb9xVNoEXv+OcmvMT/oyFnCS6mJcgIuLNHtBrXfwcjzSGBVeKXeWt4cQ0vpCw/f9VbCS8IVvS8OdIlsCAGhTZd03s/tmGVyhGIOuI0VIpPPxPaYYVB/4TYOuyZ0ArBuZYCEDz5ZXLr1c4j2hJnfSeMdMvGbDeBd6OPHlh0CxiWuOI98jGmg8+bhf3cvk/nmaK6dE5d14tyE6cfk0YwlMSGf8BnLq10KPjIixjrySjKMDSXEQVW7IsnSNQUfL7Zi0wS/pkQKMYglqI8dHRPFF5YEJ4VStoWF/JhW6k6uu/AjUHqVYavEMgGObROvTd+FjRELtAihIBIDx6VUwFjUrLqqSmawnIzEmG7qLPnYJRlaGeF4ViDBqXNUxHvwxgI3wngwPo6MbECGiK2gFDP8EOKHDeE2mY6BAyMcUb2h4LOVGcL1cy0zcPxX4GZnhSo7TJxMEG4jtv1ZcHs2S3bt2cvxp1o+hw5LkKkhyR6ttJGk+4xmvmgFa0U0iVl2b/9Hl+GrW2uJY/bpx1RUIT+GaEH8GxU1WfZH9MU/8HqvjSkA3Qe7Oj6qaYPbmYJPp0AfLiDxR+LShtB9yFDeMRJs54qhCHs/uqPS6Ntz6kDZfpihOzkt6j5drbpxTGrTymMo+h7e9rx2DaJJOdSh7kADTWGsHGDMSh+VFNRhAvtkL8tHNiMGYbh96gx5nYs+XbUAAAA==

+AAAAajwAEADEaBAwAYjAIHADAYBQ4AYDAKHADAYBQ4AIDBKHA
AAINR4AAABqPAAQAMRoEDABiMAgcAMBgFDgBgMAocAMBgFDgAg
MEocAAAg1HgAAAGo8ABAAxGgQMAGIwCBwAwGAUOAGAwChwAwG
AUOACAwShwAACDUeAAAAajwAEADEaBAwAYjAIHADAYBQ4AYDAK
HADAYBQ4AIDBKHAAAINR4AAABqPAAQAMRoEDABiMAgcAMBgFDg
BgMAocAMBgFDgAgMEocAAAg1HgAAAGo8ABAAxGgQMAGIwCBwAw
GAUOAGAwChwAwGAUOACAwShwAACDUeAAAAajwAEADEaBAwAYjA
IHADAYBQ4AYDAKHADAYBQ4AIDBKHAAAINR4AAABqPAAQAMRoE
DABiMAgcAMBgFDgBgMAocAMBgFDgAgMEocAAAg1HgAAAGo8ABAA
xGgQMAGIwCBwAwGAUOAGAwChwAwGAUOACAwShwAACDUeAAAAa
jwAEADEaBAwAYjAIHADAYBQ4AYDAKHADAYBQ4AIDBKHAAAINR4
AAABqPAAQAMRoEDABiMAgcAMBgFDgBgMAocAMBgrleBq6pbV9U91x
Gkqm5SVRdU1d9U1Zuq6qnT8NtU1Uur6u3T5a233OdJVfWOqnpbVT1sHbk
AAObmiAWuql5eVbeoqtsk+Zskv1FVP7OGLFcmeXB3f06SeyV5eFXdL8mZS
c7v7pOTnD/dTlXdPclpSe6R5OFJzq6qY9aQCwBgVpbZAnfL7v63JF+V5De6
+/OTfMnRDtILH5hu3nD66SSnJjlnGn5OkkdO109N8tzuvrK735XkHUnuc7R
zAQDMzTIF7tiqukOSr0nygnWGqapjur1Sd6X5KXd/eokt+/uy5NkurzdNPkd
k1yy5e6XTsMAAD6hLVPgnprkxUne0d2vqao7J3n7OsJ099Xdfa8kxye5T1V9
1h6T106zuM5EVU+oqgur6sIrrrjiaEUFANiYZQrc5d19z+7+liTp7ncmWccxc
B/T3f+S5OVZHNv23mkLYKbL902TXZrkhC13Oz7JZTvM6+ndfUp3n3Lccce
tMzYAwIFYpsD94pLDVlJVx1XVrabrN83iOLu3JjkvyenTZKcneeF0/bwkj1X
VjavqpCQnJ7ngaOcCAJibY3cbUVX3T/IFSY6rqu/aMuxWSdbxac/Uk1N0uyXuSPDpJuvtNVXVukjcnuSrJE7v76jXkAgCYlV0
LXJIbJfnkaZqbbxn+oU/kedbSDdPcbknzuDsP/KclDdrnPWUnOOtpZAADmb
NcC192vSPKKqvrN7n73AWYDC0AGAPe22BO+zGVfX0JIe2Tt/dD15XKAAAdr
dMgfu9JL+a5JlJHGMGALBhyxS4q7r7V9aeBACApSzzNSJ/XFXfUlV3qKrbbHQsvgc
sf5vpvKgAAGzAMlvgDn8H2/duGdZJ7LyHDdfdJBBAEAYDl
H3IVaVZ9UVT8wfRI1VXVyVX35+qMBALCTZY6B+40kH8nirAzJ4hyk/2tti
QAA2NMyBe7Tu/snknw0Sbr7Q0lqrakAANjVMgXuI9PJ5TtJqurTk1y51lQA
AOxqmU+h/lCSP0lyQlU9K8kDkDkpyxzlAAAOxumU+hvrSqXpfkflnsOv2O7v7H
tScDAGBHu+5Craq7Tpefl+ROSS5PclmSE6dhAABswF5b4L4ryROS/NQ4z
qJk9kDAGzArgWuu58wXf2y7v7w1nFVdZO1pgIAYFfLfAr1lUsOAwDgAOy6
Ba6qbpfkjkluWlX3zjXHvd0iyScdQDYAAHahM2OoOeDAOy6
Ba6qPjXJHZPctKo+N9d+99stknzkTSAWQDAGAHex0D97Asvi7k+CQ/s2X4vyd
58hozAQCwh72OgTsnyTlV9QwCAQAwB6W+R6451XVI5/8lcI8lNtgz/4X
UGAwBgZ0f8EENV/WqSr03ybVB/foLL4XDgCADVjmU6hf0N2PTfL+7n5nk
qkvsnOWG9sQAA2M0yBe5D0+rTknw0Vk
l+MsnrsjgLwzPWmoAgF0t8yGGH5muqS+qXpDkJt39wfXGAgBgN8vsgftPdfb1
TZn1Lki/MYuvbX1TVr2w/vRYAAdjmV2oz87iy87iy3t/cbr9mCS/lcWnUQEAOG
DLFLjP7O7P2XL5ZVX1hXcFFAPmY2sJvAeT/BG1bU2qg5GIt3G+zmVbXhFLRVkO1
kvPKrzu/hpjziq8+MTzzIF7r5JHltV75lun5jkbVJu5jkLVV1UZLu7nuuLR0AANexT

7+NpTAACwtGW+RuTdBxEEAIDlLHMmBgAAZmTXAldVNz7IIAAALGev
LXB/lSRV9VsHlAUAgCXsdQzcjarq9CRfUFVftX1kd//B+mIBALCbvQrcNyf5
uiS3SvIV28Z1EgUOAGADdi1w3f0XWZz39MLu/vUDzAQAwB6W+R6436qq
b0/yxdPtVyT51e7+6PpiAQCwm2UK3NlJbjhdJsk3JPmVJI9fVygAAHa3TIG7
97aT2f+Zk9kDAGzOMl/ke3VVffrhG1V15yRXry8SAAB7WWYL3PcmeVlVvT
NJJblTkm9cayoAAHa1zLlQz6+qk5N8ZhYF7q3dfeXakwEAsKNltsBlKmxvW
HMWAACW4GT2AACD2bPA1cIJBxUGAIAj27PAdXcn+aMDygIAwBKW2
YX6qqq699qTAACwlGU+xPCgJN9cVRcn+Y8sPona3X3PdQYDAGBnyxS4L
1t7CgAAlnbEXajd/e/e4kJyR58HT9g8vcDwCA9ThiEauqH0ry/UmeNA26YZLf
XmcoAAB2t8yWtK9M8t+yOP4t3X1ZkpuvMxQAALtbpsB9ZPo6kU6SqrrZei
MBALCXZQrcuVX1a0luVVX/PcmfJnnGemMBALCbZU5m/1NV9aVJ/i3JXX
ZL8YHe/dO3JAADY0VIns09yUZKbZrEb9aL1xQEA4EiW+RTq45NckOSrkj
wqizMzfNO6gwEAsLNltsB9b5P++sSpKq+pQkr0zyrHUGAwBgZ8t8iOHSJ
P++5fa/J7lkPXEAADiSXbfAVdV3TVf/Psmrqur3szgO7pFJzn1J0hn5hfVQ
hHv6y3r+bfg57/vriAABwJLsWuO5+6kEGAQBgOUf8EENVnZLkfya509bpu/
uea8wFAMAulvkU6nOy+CTqG6RUmuWW8cAACOZJkCd0V3n7f2JAA
ByxS4z+nuz157EgAAlrLMmRheVVV3X3sAACWsswWuC9McnpVvSuLY+
AqSfsaEQCAzVimwD187SkAAFjaMgWuD3a5154EgAAlCLVPgXphasknN0
3JbnHGnMBALCLIxa47Z9Arar7h/dN3JAADY0VIns09yUZKbZrEb9aL1xQEA4EiW+RTq45NckOSrkj
wqizMzfNO6gwEAsLNltsB9b5P++sSpKq+pQkr0zyrHUGAwBgZ8t8iOHSJ
P++5fa/J7lkPXEAADiSXbfAVdV3TVf/Psmrqur3szgO7pFJzn1J0hn5hfVQ

MRoEDABiMAgcAMBgFDgBgMAocAMBgFDgAgMEocAAAg1HgAAAGo8
ABAAxGgQMAGMyxmw4A8J/FoTNfeFTnd/HTHnFU5weMwxY4AIDBKH
AAAINR4AAABqPAAQAMRoEDABiMAgcAMBgFDgBgMAocAMBgFDgA
gMEocAAAg1HgAAAGo8ABAAxGgQMAGIwCBwAwGAUOAGAwChwAw
GAUOACAwShwAACDUeAAAAajwAEADEaBAwAYjAIHADAYBQ4AYDA
KHADAYBQ4AIDBKHAAAINR4AAABqPAAQAMRoEDABiMAgcAMBgF
DgBgMAocAMBgFDgAgMEocAAAg1HgAAAGM5sCV1XPqqr3VdUbtwy7T
VW9tKrePl3eesu4J1XVO6rqbVX1sM2kBgA4eLMpcEl+M8nDtw07M8n53X1
ykvOn26mquyc5Lck9pvucXVXHHFxUAIDNmU2B6+4/T/LP2wafmuSc6fo5S
R65Zfhzu/vK7n5Xknckuc+BBAUA2LDZFLhd3L67L0+S6fJ20/A7Jrlky3SXT
sMAAD7hzb3A7aZ2GNY7Tlj1hKq6sKouvOKKK9YcCwBg/eZe4N5bVXdIkun
yfdPwS5OcsGW645ctMMuvvp3X1Kd59y3HHHrTUsAMBBmHuBOy/J6dP
105M8f8vw06rqxlV1UpKTk1ywgXwAAAfu2E0HOKyq/neSBya5bVVdmuSHk
jwtyblV9bgk70ny6CTp7jdV1blJfF2m
PyvJWetLBAAwT3PfhQoAwDYKHADAYBQ4AIDBKHAAAINR4AAABqPA
AQAMRoEDABiMAgcAMBgFDgBgMAocAMBgFDgAgMEocAAAg1HgAA
AGo8ABAAxGgQMAGIwCBwAwGAUOAGAwChwAwGAUOACAwShwAAC
DUeAAAAajwAEADEaBAwAYjAIHADAYBQ4AYDAKHADAYBQ4AIDBK
HAAAINR4AAABqPAAQAMRoEDABiMAgcAMBgFDgBgMAocAMBgFDg
AgMEocAAAg1HgAAAGo8ABAAxGgQMAGIwCBwAwGAUOAGAwChwAw
GCO3XQArr9DZ77wqM7v4qc94qjODwBYL1vgAAAGo8ABAAzGLlSOOrt4A
WC9bIEDABiMAgcAMBgFDgBgMAocAMBgFDgAgMH4FCrMiE/wArAM
W+AAAAajwAEADEaBAwAYjAIHADAYBQ4AYDAKHADAYBQ4AIDBKH
AAAINR4AAABqPAAQAMRoEDABiMAgcAMBgFDgBgMMduOgAAsF6Hz
nzhUZ3fxU97xFGdH9efLXAAAINR4AAABqPAAQAMRoEDABiMAgcAMB
gFDgBgML5GBIAkvmoCRmILHADAYBQ4AIDBKHAAAINR4AAABqPAA
QAMRoEDABiMAgcAMBgFDgBgMAocAMBgFDgAgMEocAAAg3EuVGB
pzpUJMA+2wAEADEaBAwAYzNC7UKvq4Ul+PskxSZ7Z3U/bcCQA1sQu/E9
c1u31N+wWuKo6JskvJ/myJHdP8/dsGLkh2ZqiXJTTn28h8A+wdOxIsdDzp6+6V0kxAc6GN1P7bRX+/jw5NMq7sjmq7zxY8dHGYnDFq6JsX5AN/2HdtDl3xf2eLavq5tzvT1r2lrRRZ5O1DyDqebPtcmXAVGanEiJ7l3pyQd4+vh+dGCmvj6gn4Gma9zOg2ubPaabvDbVnqXtefKkH88crupV33RqZ4cFcyyr5fP/2d87Hz5+Xvs4Ly7xSvzvb7c98t3zJXoD1wo7T05LW8aA9JC15/FB+r58f1e7mzH9Lr8d6+ghEO2/TI8jbPNF3SzjTdPh7LL36cy5dU54GPlb+Tx2cBg5SnSjS2k6PztudZ/ytNzrPjqDemxXTd2PzZp2fPj3D5fHhQ9d6d19PjTdOV9mzHnz6bqW5Pdmbt3/zsZQL3z+eB9yS/cPBjXvUt2afuOPdi9Tq2cMvZ81KsuY+21oWD0fMYm0FbSbLhLwnvm8AnHxuUh7Jhf+x3fx8lbpGrxqB9RPQ1lyVlfz+9/zvUndQ/+uwhy7vR+7vHX4vQcw82Btv4=

duT/P00+Pgkn5HkW7r7JZvKdiRV9Z7uPnHDGWa9/Oaeby/W7/5Zdqux/I5s7
vn2Yv3u30EtO1vglvPzSb6kuy/eOrCqTsriTBB320SoLTl+YbdRSW51kFl2Mev
ll5nns373z7JbjeW3slnns373bw7LToFbzrFJLt1h+N8nueEBZ9nJNyb57iRX7j
DuMQecZSdzX35zz2f97p9ltxrLbzVzz2f97t/Gl50Ct5xnJXlNVT03ySXTsBOSn
JZkU5/O2eo1Sd7Y3a/cPqKqnnLwca5j7stv7vms3/2z7FZj+a1m7vms3/3b+LJz
DNySqupuSU7N4kDKyuK/gvO6+80bDZakqm6T5MPd/cFNZ9nNnJdfMu981
u9KuSy7FVh+q5tzPut3pVwbX3YKHADAYJwLFQBgMAocAMBgFDgAgM
H4FOoKqupHk/xrkmd29z9tOs928q1GvtXMOd+csyXyrUq+1ci3fweZzRa41Vy
Q5KokP7vpILuQbzXyrWbO+eacLZFvVfKtRr79O7BsPoUKADAYu1CXVFU
PS/LIfPwJdZ/f3X+y0WCTuefbTVX9YHf/8AxyzHr5zT3fbuayfncyl2xzX7dzz7c
b63c5c8+3m7ms350cVDZb4JZQVT+X5C5Jnp1rT+txfJLHJnl7d3/HprIl88+3
l5mcMHnWy2/u+fYyh/W7mzlkm/u6nXu+vVi/Rzb3fHuZw/rdzUFlU+CWUF
V/29132WF4Jfnb7j55A7G25ph7vn/bbVSSm3b3RrcED7D85p5vtut3ztmSIdbt
3PNZvysYIN9s1+8csvkQw3I+XFX32WH4vZN8+KDD7GDu+f4lycndfYttPzd
Pcvmmw2X+y2/u+ea8fuecLZn/up17Put3NXPPN+f1u/FsjoFbzhlJfqWqbp5rN
zOfkOTfpnGbdkbmne/ZSe6U5L07jPudA86ykzMy7+V3Ruadb87rd87Zkvmv2z
My73zW72rOyLzzzXn9bjybXajXQ1V9aracULe7/2HDkT7O3PPN3dyX39zzsX
9zX7dzzzd3c19+c8/HzhQ4AIDBOAYOAGAwChwAwGAUOACAwShwK6iqt0
w/37rpLDuRbzXyrWbO+eacLZFvVfKtRr79O8hsvkZkBd19t6r6lCT323SWnci
3minfbZPcd9NZdiLf/s05WzLMa0O+fRrk+SffPhxkNp9CvZ6q6jZJurvfv+ksO
5EPxjT314Z8MC92oS6hqk6squdW1RVJXp3kNVX1vmnYoc2mk2+dquqiTWf
Yi3z7N4dsc39tyLc+c3j+7UW+/TuobHahLud3k/xckq/r7quTpKaOSfLoJM/N5j
fTy7eCqvqq3UYl+dSDLJjCPn2bc7ZJrN+bUS+lcz9+Sff/s0hm12oS6iqt+92Ut
+9xh0U+VZTVR9N8pwkO70YHjWd225j5Nu/OWdLhnhtyLeCAZ5/8u3THLI
pcEuoqucm+eck5yS5S5ZBp8QpLTk9y2u79mU9kS+VZVVa9Ncnp3v3GHcZd09
wkbiLU1g3z7NOdsU4a5vzbkW8EAzz/59mkO2RS4JVTVjZI8/8LsmpufZ8cZck+
eMkv97dV24wnnnwrqqovSvLu7n7PDuNO6e4LNxBrawb59mnO2aYMc39tyLe
CAZ5/8u3THLIpcAAAg/EpVACAwShwK2gqk6tqtmdyuM
w+VYj32rmnG/O2RL5ViXfauauNbv4PM5ot8V3PfJJ9dVcd295dtOswO5FuNfK
uZc745Z0vkW5V8q5Fv/y4saMxXrmXO+OWfbS1V9aXe/dNM5diPfauRbzZzzHVQ2x8Atoaoem+R
1SR6Y5JOu3bsnr/ekF/9UDZ8SiRz9hD59EANR6kSPrUn5kK2lEJ+6qeL7d4ey2Vp0Mr+e0oNqbKF/1GqDGS4o6fWWvvR5mJh1fG/7Nt61tmdJ9BlLv5hSI
rmrKq6gHqU6RI2xUuUWMuetcn0ePIBM4Ch8sYJ+NPr7wzlLyUKuU9Ah5Ka4bnBYdKTkiaN6h0nXhD0zk5vgKjvSiqF/2ZrkmL/ssz+
Zhe+TmoOOuU5uZK/NuDtPc9bMbsN5J3wMQ8lfLfcN3Ag/ORMj3pgdm08L9vx
/JGpzGxnl21eMgn5LEe0TVFJkpIe0O/53fssEwmFa5hZRqjF5q6SfGL6v3tzPw
sPa7T0bl1nLjfrTtFJzj6ZrdQdSgTLzbdfl2Glz3M7I3PzqubMC1l2xL4QJ9QzhH
6zJP7EMmqY86E2kMmlEaD+nM1r+tEXJ5flCWZU0WCbWTrDEhVTYPtd3aAc7UyCD

U8++eabbw7ZFLjrqapumOS2ST40xy8ClW818q1mzvnmnC2Rb1XyrUa+/dtU
NgUOAGAwjoEDABiMAgcAMBgFDgBgMAocAMBgFDgAgMEocAAAg1H
gAAAGo8ABAAxGgQMAGIwCBwAwGAUOAGAwChwAwGAUOACAwShw
AACDUeBYi6q6uKouqqq/qaqXVNWnHqX5PrOq7j5d/8Au0/xmVT3qes73sV
X1xqp6U1W9uaq+5yhkPVRVb9xl3Md+j33M94FV9QVbbn9zVT12vzn3q6oe
XVVvqaqXbRt+qKo+VFWv3/Jzo4PON2X5sqq6cMr51qr6qTU8xsVVddvrMf0
NquoXpufbRVX1mqo66Shn+sNpub+jqv51y3r4guub93o85sc9L4/yvM+oql/aY
fjtq+oF0/vMm6vqRVuyvOAoPOanrTKPo6mq7jqtw7+uqk/fNu7w++3rp8tTj+
LjPnnL9V3f0zh4x246AJ/QHtTd/1hVP5rkyUm+fdUZdvfjV4/18arqy5J8Z5KH
dvdlVXWTJN9wtB9nqxV/jwcm+UCSV07z+tWjkWkfHpfkW7r7ZTuM+7vuvte
6A1TVMd199S7jPivJLyV5RHe/taqOTfKEdWdawtcm+bQk9+zua6rq+CT/sey
d9/qdD+vur5ymfWCS7+nuL99y/2Ue49juvmrZTJMHZsvz8oD8cJKXdvfPJ0l
V3fNozLSqjklyRpI3JrnsaMzzKHhkkud39w/tMv7w++1nJnlJkudvHVmLFV/d
fc31fNwnJ/nR652WtbMFjoPw50k+o6puUlW/Mf2H+NdV9aAkqap7VNUF03
+Pb6iqk6vqZlX1wuk/6zdW1ddO07768qk45POq+umqel1VnV9Vx21/4Kr6/Kp
6RVW9tqpeXFV32CHfk7L4I3dZknT3h7v7GdP97VVr5py/WFV3XpLjp+tqj
+ftu7cu6r+oKreXlX/a8u8j62qc6b7/35VfdL236OqPlBVZ02/66uq6vbT8K+oqld
Py+pPp60Nh5J8c5L/MS2vL6qqp9S0xfAIeX98Ws5/W1VftNuy32EZPmZaZ2
+sqh+fhv1gki9M8qtV9TP+vq9qvrkWmwhO3fLNA+sqj/ebf
pp+MVV9YNV9RdJHr3HQ35fkrO6+63Ter2qu8+e5nn+e5nGn6TnzhunyxGn4x229
rWkrby22mp1diy20L6iqF9XHb+X9tinnRVV11yMsijskufzwH9LuvrS73z89zn
WW9eEcVfXDVfXqJPevqq/fst5+rRaF4/q4Tt7pefT0qnpJkmfvsYWel5ufbCq
uk9VvXK6zytrUTIOb+X6k+m18xNb7vON03Lfr/UKKckec70+9x0Gv/UHZb
xnzy95t5aVc+pWjTXqnrIlOOiqnpWVd14Gr71efnefWYJKckec70+9x0Gv/UHZb
bzab5vGaa76nT8KXf17Ytq+u8hqvqv2bxT+bja9sW7x3cIsnh59OhWrw/nZ3kd
UlOqKrvnbK+oaqeuuVx/6gW75NvqqonTMOeluSm0+/wnGnSY6rqGdN0L6m
qmx4hD+vS3X78HPWfJBcnue10/ZeS/HiS707yG9OwuyT/NJ0/TeTePCrJDbPYGnDcPxrkzxr
b9Rkpsm+eokz9gyv1tOly9Pcsp0vbfc7weT/NJ0/TeTePCrJDbPYGnDcNPxrkzxr
h6z/fHj+O4x7Q5L/Ml3/4SQ/tyXHj0/XvyOL/9LvkOTGWfxB+ZQkh6aMD5im
e1YWRXGn3+Mrpus/keQHpuu3zuI/5iR5fJKfnq4/5fB8tt8+Qt7D9/+vSf50un6
dZb/t9/+0aT0dl8UW+z9L8sjtv8O2+xjEkr59+fjnJbbMo8jebpvn+aZ0dO8
3/8PBfSfL1u02/5bn1fUs8B1+X5HN2GffHSU6frn9Tkj/a+tzZMt0HpstHJXlR
Fv/0fmoWfyAftSXPt03XvyXJM4+Q6/jpPq9P8tNJPneJZd1Jvma6frck/y/JDbPanRQlDu
cneewuj/XAJC/Y4bV5nbzT8+i1h58DeyyjpZ6X2x7zFkmOnm7cha5/SZLnTdfPSPL
OJLfM4r3g3UlOPyOK1dHhZ3CjJX2Z6fW+b78OS/EuSlyX5n0k+bcvv/a/Tsr5
Bkr/K4h+OmyS5Jmem8tQ7d3peZdvze4/l9qNJvn66fqskRwnJsZvke72tLvuf
stXwvTnJRFlsMP5jky7e8Fq9Jcr/p9kOTPD1JTcvlBUm+eBp3m+nH+eBp3m+nyptN8Pm
Xra2DL/K5Kcq/p9rmHf3c/B/9jCxzr9LKqen0Wb94/lsUb6G8lSS+2irw7yV2ye
HN9clV9f5I7dfeHsngz+pJabDX6ou7+1x3mf02S352u/W+M8lnJXnplOM
HsnhDX0pV3TLJr7Fd0gc5J88ZZJzpsuL0rypu6+vLuvvEOP+vLuvzOIP0gnTuEu6+y/3

```
yJgkH8nijTRZ/AE9NF0/PsmLq+qiJN+b5B4r5v2DHR5jp2W/1b2TvLy7r+jF
LrXnbJvnbv6uu+81/Twxyf2S3D3JX07r4vTp8a5K8idJvqIWuzgfkcWunx2n3z
L/381q7p/kd6brv5Wd18tWX5jk97r7mu7+hywKw1Y7LdsddfelWTw3n5TFc/j8
qnpI9l7WVyd53nT9IUk+P8lrpmXzkCR3PkL+7XbLe96W58Buy+h6PS8nt0zy
e7U4fupnt93n/O7+1+7+cJI3Z7Ge75trl8VHssv67u4XZ/G7PyOLfwr/uq7dEn9
BL7ZuXpNFWT6UxXJ/V3f/7TTN9tfIkZ5XOy23hyY5c1oXL8+iJJ6YfbyvLfE
a3suDuvuzknx2kl+qaYt1knd396u2ZH1okr/O4h+cuyY5vNX926vqb5K8Kov3r
+tsjZ+8q7tfv8Ny4IA5Bo51elB3/+PhG4d3YWzX3b8z7Rp6RBZ/GB7f3X9WVZ
+fxdaiH6uql3T3Dx/h8Xrb7cqiWN3/CPd7UxZ/EP/sCNNtd+V0ec2W64dvH35
tbc+0/XaSfLS7Dw+/est9fzHJz3T3ebU4lukp1zPfbnk/9hi7Lfst9znyAVPLqSyO
VXrMDuN+N8kTs9gS+pru/vfpubLb9Mlyx4wdXq9/s8S0h5f/VZkOLZkyHP7
wxZGWw3WW7Z4Ptij6/yfJ/6mq92ZxfNP5e9zlw33tcW+V5JzuftKRHmcfefda
roeX0X6elz+S5GXd/ZXT7taX75Ble56dXivXDdX9z1kUzd+pxQcXvjjJP+0y3y
OtxyM9r3ZabpXkq7v7bdumfctRel+7Xrr776bn1N2TvC8f/ztVkh/r7l/bep9pPX5
Jkvt39wer6uVZFNGdbF+udqFuiC1wHKQ/T/J1SVJVd8niv9S3VdWdk7yzu3
8hi61a96zFp78+2N2/neSnknzeDvO7QRa7tpLk/0nyF9vGvy3JcVV1/+kxb1hV
O20t+LEkP1HTJ2Wr6sZV9e3Tf8fvr2uP5/mGJK/Y4f57OfHw42dxXM32jHu
5ZZK/n66fvmX4vye5+faJ95N3p2W/bZJXJ/kvVXXbWhxn9ZgjzXMXr0rygKr
6jOlxP2l6DiSLP+afl+S/59otIHtNv/13+Naq+tYdRv1kFltA7jJNd4Oq+q5p3Cu
TnDZd/7pcu14uzqL0JcmpWeyGzzT+q6d53D6LXXR7qsVxX8/eYfjnTc/vVNU
Nsljm787yy/r8JI+qqttN87hNVd1ph+lWtdsyul7Pyx3uc8YSj/3qJA+sqk+pqhtm
l2Mdq+rBde1xpTdP8ulZ7HrdzVuTHDr8vMrer5G9fp+tXpzFMYWHj7H73O
nyer+vHY33nOl5cVIWz6mdsn5TXXs86R2n6W+Z5P1TebtrFlvAD/votA6YGV
vgOEhnZ3HQ+0VZbOk4o7uvrMWBvF9fVR9N8g9ZHPdx7yQ/WVXXJPlokv
9vh/n9R5J7VNVrszje5eMOCO7uj9TiQPNfmHZNHJvk57LYMrN1uhdNf5T/
dHoT7iyOV0sWf6B+dfoj8c4k33g3JDm9qn4tyduzOMZrWU/JYrfWWYXn+HsvmcMxIsLnLIHvyQU/JYrfT32dRa
A5/1cQfJ/n9Whws/W3b7nN98+607D+muy+vqidlscuwkryou59/3dnsrbuvqKoz
kvzvmg4az2KX9t9299XTlpMzpvx7Tr/D7O+axTFS2x/zDVX1ndM8PimL9frC
afS3J3lWVX1vkity7XJ6RpLnV9FUFWRSlw1svnpfFrso3ThlencVzbi8nZnEs4
Ha3S/KMLb/XBVkc3/XhZZZ1d7+5qn4gyUumAvjRLLZg7vQHexW7LaOnZI
nnZXf//1vm9RNJzpkK9BG3dE/Pu6dksRvy8ix29+30QY3Pz2J34eEtp8/s7tdM
W5R2mu+Hq+obp/zHJnlNkt0+xf2bWbyWPpTF7uTd/EgW7ytvmN4/Lk7y5dn
5tbXM+9p+33NeVlVXZ/FPx5nd/d5pa+fHdPdLqupuSf5q6psfyOKY0z9J8s1V
9YYs/vF91Za7PX363V6V6XLGzMThA1EBhjSVv6+ajpVa5+N8cnd/oKo+JYv
S9YDpeLjdpv/JJL/VH//JSICjQoEDWMJ0XNCtsjgu7ie6+zc3Ggj4T02BAwAY
jA8xAAAMRoEDABiMAgcAMBgFDgBgMAocAMBgFDgAgMH8X1KCVJk
Azk4HAAAAAElFTkSuQmCC\n",
      "text/plain": [
       "<Figure size 720x648 with 1 Axes>"
```

```
     ]
    },
    "metadata": {
     "needs_background": "light"
    },
    "output_type": "display_data"
   }
  ],
  "source": [
   "plt.clf()\n",
   "plt.rcParams[\"figure.figsize\"] = [15,5]\n",
   "df.groupby(['COUGH','FEVER','SHORTNESS_OF_BREATH','SORE THROAT']).size().plot(kind='bar')\n",
   "plt.xlabel(\"\\n Possible Combinations of Fever, Cough, Sore Throat and Shortness of Breath\")\n",
   "plt.ylabel(\" Number of patients \\n\")\n",
   "plt.show()"
  ]
 },
 {
  "cell_type": "markdown",
  "metadata": {},
  "source": [
   "#### Seperating the features and independent variables"
  ]
 },
 {
  "cell_type": "code",
 "execution_count": 8,
 "metadata": {},
 "outputs": [],
 "source": [
  "X = df.iloc[:,0:15].values\n",
  "y = df.iloc[:,15].values"
 ]
},
{
 "cell_type": "markdown",
 "metadata": {},
 "source": [
  "#### Spliting the dataset into test and train"
 ]
},
{
 "cell_type": "code",
 "execution_count": 9,
 "metadata": {},
 "outputs": [],
 "source": [
  "from sklearn.model_selection import train_test_split\n",
  "X_train, X_test, y_train, y_test = train_test_split(X,y,test_size=0.20)"
 ]
},
{
 "cell_type": "markdown",
 "metadata": {},
 "source": [
  "#### Creating Different Models"
 ]
},
{
 "cell_type": "code",
 "execution_count": 10,
 "metadata": {},
 "outputs": [
  {
   "data": {
    "text/plain": [
     "GaussianNB(priors=None, var_smoothing=1e-09)"
    ]
   },
   "execution_count": 10,
   "metadata": {},
   "output_type": "execute_result"
  }
 ],
 "source": [
  "rfc = RandomForestClassifier(n_estimators=15,random_state=42)\n",
  "dt = DecisionTreeClassifier(random_state=42)\n",
  "lr = LogisticRegression(random_state=42)\n",
  "gnb = GaussianNB()\n",
  "svr = LinearSVC(random_state=42)\n",
    "\n",
    "#--------------------------\n",
    "lr.fit(X_train,y_train)\n",
    "rfc.fit(X_train,y_train)\n",
    "dt.fit(X_train,y_train)\n",
    "svr.fit(X_train,y_train)\n",
    "gnb.fit(X_train,y_train)\n"
   ]
  },
  {
   "cell_type": "markdown",
   "metadata": {},
   "source": [
    "### Area Under Curve Analysis"
   ]
  },
  {
   "cell_type": "code",
   "execution_count": 21,
   "metadata": {},
   "outputs": [
    {
     "data": {
      "text/plain": [
       "<sklearn.metrics._plot.roc_curve.RocCurveDisplay at 0x23c8bed9b38>"
      ]
     },
```

"execution_count": 21,
"metadata": {},
"output_type": "execute_result"
},
{
"data": {
"image/png":
"iVBORw0KGgoAAAANSUhEUgAAAmEAAAHICAYAAAAGDj3wAAAAB
HNCSVQICAgIfAhkiAAAAAlwSFlzAAALEgAACxIB0t1+/AAAADh0RVh0
U29mdHdhcmUAbWF0cGxvdGxpYiB2ZXJzaW9uMy4yLjEsIGh0dHA6Ly9t
YXRwbG90bGliLm9yZy+j8jraAAAgAElEQVR4nOzdd3hcxdn38e/sale76l2yr
Obeu9wLNp1AIJTQEiAEQg0hjYcACTF5krwkIaQACSGBhBBanoQOoQW
MCzYu2Bj3KlnFsnpdbZ/3j7NW88qW7F2tyv25Ll/SnnM059YKWz9m5sworT
VCCCGEEKJvmSJdgBBCCCHEUCQhTAghhBAiAiSECSGEEEJEgIQwIY
QQQogIkBAmhBBCCBEBEsKEEEIIISJAQpgQot9TSq1QSt0Y6ToGOqXXU1
5RSq0+xjXuVUn8JVU1CDGUSwoQYpJThgFJqRx/fN+gveqVUkVLqzL6spT
tKKatSarlSaq9SarVQ21NKqYJI13aUUupo70rV0pLX+udZaArEQIS
AhTIjBawmQAYxUSs3u7qJAW+UV/xYopaK6OfDD9h1936mlFoDOD
CC2nil1HtKqVql1G6l1OUdrj9fKbVeKaU+UUqN1fNSKvME35sQIkBCmBCDkFIqBrgMeDbw50ANS
+UUqN6nD+LKXULqVUg1LqUUB1+fqvK6V2KqXXlFLvKKXyO5zTSqnbVZ
7gb1B7n0mcBZwkdZ6g9baq7u0Fo/prV+MnBNp167QK/ZPwKfFwTucYNS
6hDwgVLqbaXUN7vc5zOl1CWBz7t9b7txKVAP/IRjf3bLlVL/VEr9PfDebVdK
FXY4/wOl1P7AuR1KqYuD3KNDxOSDw+t1KqLNDObqXUGUHe
C5tS6h9KqRqlVL1SaoNSKvME35sQIkBCmBCDkFIqBrgMeDbw50qllLXZ
dcANwHxQBXwHvAcRu/ZVcAflFKTAte2ANcCocD5wG3K6Uu6XLb8Ay
cA+4GeB2tOAfwM/B/NKA/cDCDt/bl4B7nvmc
B6rXXJKdZ/GjABOAfjfbuqQ40TgXgzXgDGK6Vmdjl/Ye
BcEvAa8GiHc/8DhAdeuF/1BKDQtyr4BZwp4BqsCkuXAwNKjrCQQhgxhwg+bCmq0Grw0Sqnr1cHqj8Kn4Xn50etsNZ8RcmhpAVaF7PmsxQ/
f1Y5bAb5rGlLG8oBKMRN9yeufdSUWNlvVjRFLSQWPbrgwNTxT35hJQaOgJL6qVJcX2CBzj/wUGuPv9V0zCMFhNVV5pM5VGhMSRi/6UKsZIHNJUjYWs9mSJTLjW26mR01SHJTgYDWs5GSMgxxKxDsmiugjqG3ZuaXmWMNjBA3fBRQIJF1SjhVcXnYm4sVmSOY6qCf1IGOp7R0ng3SM6C3qUjo7jzUwSkqFTmWa3HUg9bO/HFa7ukzWuXhTFQuyxXvHx/Cm8KdOcHAMRQBe5WcAi9OXo1QLkGrHwqxL15Pe9qfJ3gaTz7aWRxiU5o6JJg5Hp3Xfnoh7q/ZMSoxudB+eLDENhT+m/8ym38Fy/6h+5/Ly/ZD1Pdor7pHCLkXp/JAEmxGLjlxz6m5MeeO5Bvz1XvTbxDtROrPXS3b4w9MCe2yd+WTKbGawA9AgIRyqk9n2HUo4+U2iDoU1tM8N6ZDzP5bYKlsZ/h1zi1IYbd2z8x3bbd8eV9cpuyrrq/IHWDgpcWBnhqsy4iFx8lxZ09ZRQ8Ev38zDhtFr2Wp1thF7w3f9Gm4uwwP7ZXgJ3eM+/rQe0cgKfMNe7TCSRpRIb7DGxaiaRg9yO4gxEJwwsrAdd9dtIcMwDXNYRjvwYB9OFYlJnh3TQkQRFNbdEtyKy/vFxR9ZeygPbYP1WSyL+/g3k2sQ2v6JJ73K0VvcnrhwTYsa7dqmJ1YsvSMvVUiAZ6nJbJk3NeSs0fwN2FZE6FVi+NHz6uWhDHzLOC1kjzq3pbbEgbsNhwP2AG3O4ZW5P0p9EhxFvaaWnk+Yv/v3c51JhNbqj+5mkG7c8vWt6SCGMgLhhjqgG4Np9ldasPSPA+ns4JuLkE4lgqsDEjQpymzMRm3rrCk9xVIf4qLhZr5Jlj5cCk2N6A5V6UYf9FWL9v1HhvPv4dMZmcIDcehWJ+6IWBmlDH0J3hJ/H7ZDO3vXtV9mR2nKbMVy0LtvuP9wCt7+Nj/sP/lbz7Db56eUiavD6UHT+6k5+DL2/8dK1VuJbiSNnR2oLpGiY6HuoX0P2uv2RCrLBjlixNJfDNo7zz2lptCm5CymWp4TiK4F6/fgvzQn49XMR+v5xRf2XncEuGAmpF6rSm3sayAb8s7yipz2IQZ8PBD31+cZWYsl4eM1WlhCmUtUvpqZNzXLdl9TU7LlMyQ8JlEQjzv6VOmMvx3uFBbqzzBU+ix1eSm4FBHIskXyR3zXhH4KzXD/ysnD5W/tawtRvnsyaxUa/SbVbbYAgfU8gqytBVSGCKaafhR+ETz/2cZp3YlS2KLnVUSqg/sp+IxY3YmJUYv9lLkbTE2JtRHutpx2hGMUYdYTNPeTpJIt28YVodTIalWpjmqIWhgcPeKy+N4MzUSakeTKLDffmQNTJz2x+/mvCQWeSbdIqNemYkWgslUyQXsuvHUJPpR0rA30UaXPp8AhJnhEfMLgmr+SfNHrcsQcXsiWzEXgOfVPa8A9aPZhCY9kJNk3PjWbgyBb1g6nNiVMv3qFcVh6axgcHZeduyiEDJaa9ODzBxhPyYMiBnnI7eeQK0e8+fh31Eh/CyxWqN8PM8uDFfi75GETn68eH+v2xQpohIF+0KnvmwlD3Oa2qtg4s1V0uwJ9ZM+G2lCt/eTwhc17vxDN5qW0+jmYHlvxT3hnxjOsuoHqMnWsV9UN7wHPhHAiQJKufjYXDQa7fjYd9kDrhODGiC4tXpZ2I3+WlRebd/99nQWqQ0nsAHjhVB94sB4x86yKM+UFZm97uqB36gqtTKU2zyRM5E3Q+FU+fc3XQ8tjT2qI6TDSdNTWgOFvt4pzLewJ8fZjyLqtbRg2VUXNM/eKXIPQSm/lxJgEMDlm4sn+TNDpnqmelMcyF9VyVRPGk7v+ADDWt4y7QuaJ/xmGrjXVBP+RRVb6EUaB+QtCAXP+DfkL4JKTMOjCWfbtkk6xR3Vf2bGOdrYTKyBEqAhb7zNuCkBr6JHv5KR3ROXySnfqzvN8Q+hnOKT7B7FuC1K9plOOq/+nbk8SjlKvdoXusJ4rtHZYVnRWWNMkMpzGTbAqgXC6hpQ7cfSAMIvBLG7IEHX9B9sr0GLTwmPbvi/XaHiKv2m6uNaFRcfv5KdvbK+4gSLj2wX+U6NXzu73uV0EWdTBX6w8u6WMTlNUTbxfsKOCjFQE+QyILO3lfuk8jKb7F1n4bv9aMBK0BKRvQg8Fj4qF7pW0CXhlOPZLKRZ2cRn1cKsYhEyXWNSuytZyc9CZTCb2GA9YmjHuyp95wJUfZlFPZo3d0jyx2W+4R7n1nmUsjkJS5rNVUHuZJzkx4dPJqGtv4m2RbCqIG/tGoSY7ND0hT7C/x4cH3f+mOoe1Ip9dyDr8EX9fYnFm1PI7kvvVWWDQaTJPpu6qaU4rgOYk7QYfe3UzgqzVSxNFOeUUo3ev7r4fPTCcipaKBv/hrwUrHSsCr7+wtsNwdbG5sS3GTtX8CGXOsbeMwGEC5F9yAvR7+qCBRuFvIQ7x/jhqEhHOJ/07qEbZUIBr+qT9CCiHxy2YwYCeoelz6OAbPYXojAgKlFSFUoxB52RrQsk/Sql7LvBDJcuAkl1SAVTyuJDFbcmF4WHzvTSRy8+rG1OJJM1kbSjSx6Uv8jCOM2BVpcJ6trOjc7Tx6eIvl2Qrv2WWDJQSXYeOBX0TVjsb98nWW3M53c2ttFJYu15jfHqG6CP7O3BZtGbODuTq4C6Ow+4gtKh9vu8PLcp6B/n21+wjj+7uBfHwYr9s6/xtq8yddEtB9Km3o8b+1yf1KMvXz+hSOavxa4vrJ/ZyXd1L6APc52vW8uLyaNIMsH5wZ1Df2K/8H9JFQP7XXAy7A/XiG2j8jmjMwbE6hOSjgmh9YtWvH0JQPTIF2RQGsNvwCbiqCRwzGU97d0LINqAIywLHV9MbSLYgN0chFO53ddRz3YFJ6hvXSo+9u03BJ04RVZRthZWGgn6Bf6dhNvyM7vlp4r+pfS8f52LwYGSJCXE61tvn8Kyn8Kq8bNyk+lXrRbHkg5ng4XD/H7UJHRoBsNpw9BwH1yqqU5pmHZB3zGBufBuFzR0fPVq/KG1eYzxlsPuGdakJu8GD9HSrlyDXHbvbRONj6LxFXT+vAeAPVpY1RnRuOrqbSCWTQ9C52Ng/98iGddZm95J4o+wkCw0Klg5zGV4lcE2hIQ0c9G3u4nGbfqqNfXnN5ANdVptXDkAsK0Q9Y9zgp8DHxg9MROqSmDpLIwe9w16/lO5zKFfTMFk1y07CP6Vsx3KIGDJtE7FezzOIH00nP9p8QKHTLaQg+rFRF+3/oycfLfPtuepsJVGUr3q4pA4pRQSPcwkUU4JY2bABqsYRfFhkZlwWs6dPnCEH2J2X8lIHTMhqGo+ZAJD5mv7DNjcU3L0gbFO8FWymKKMtzpMeXKSa7gK7TDJnC3zCfyo0lG80IgPRcVgS+jSS/BTfpOMzNjnqXr3Ha9R0m9uxJD+xjSNhyfJncP1TedVxOsj/k8+2H4ZoT5IY7ahTQcgOx5f4fJO7DTtvF4Gj0dVnJOiVtcIMDtbTcDmBz+/XkOTIGzLptFkPJ1FHtS2XowMBxgAbn2HK1H0zkw/aPVPTk58yd12/3W8ZNSRdMa+dc+Ye0Y/Awy7A6BVDvPCXBTDIWyPGjVhHfeG22s5v7xQ/2zv79zsyE7W1k79dPvz8cfGDCWSo3vLNQLMM8oyqv+V/jmFKUXjfF4M0fEvmGoiTL9+pgm3+QqZomXWXfxOnwaLVgQsA7SzIy8oWvTQ1lFuBFi+Ue3NXNRg1t3RxMZLQ6hhk4S6pypdfKPyYHC14khaxgJvOZT/LJm2OxX45SI8lEHG08yyeBbrozM81XPP6fVJOGB8P8Bt8AmmZYZ1djR1wRoG+NKpC+wpJPqLcPw8G1T+yfrSzGzYefcC1DHltS1U4cDQwRXYcN49DJHoe2ypmr7KDs2UbLhkPyKM7+GLlpBXo2Z/L+3JoKRTNjzfGfH1GK7Wgt+Fj3wiLsW5yfHBBhtZ2nl7tBCo0BR1FZb6CnSKQqHpyhpsoWK1j9yOXbbZdX7IY1GAd/cgYrcwVaQ04fs1lJvAbQFb5KeN6rXkmBSP07Jy09fnHqkyAJJ9rW8gDvrIcrTM54c8x54wpKJNZ2vf9XpeZFvFjUfxmpo9S+e8H34XAaVsp/VIQQmcnCaKXl05Ry6UDi8bgpLBCNEUNjtXHPeiJuD08cRdAkxJ4/hLJ+mk69/GaLUB3kWQa+4HZrGk2QIM/6iWAEiuRcS9BvAjhXUP5wR0Etoch0ZlBGo8OlPp8M7dnmJyZ0U9/2R/zTpO63kP2Sir4BLaYchZ/j9d2TSdC+aLZXeLLbCgGHyNUZV0A6J/QMB9/JSy3b3D/yF7YDPEi8gI3XnbT6eLbA06CaVUWKGpb3mJ/Bl1s0xm50cJjaJaLk24x/bKxs3/s+r1JdS7JJEoH6Hws4AtuhOtyBaVbtxu17HekTJrdzAeCNUHNekebhbIONBTzhlq8Klyj1f2P7ZtdkC+sz6HtxWwN9thyPpZ1ZRhSnzSjsPBH7TgyCvBXdXfeSbfujRQ4NsD4M6p8F8LuZ/BVvfqcrm1GSmvdFjxxj5A1ffgwOymHGVJiW0E3N8q6ZCGwAmc36yROfyCPwEwJVVKdJxfMVVNPa+vkcXmiZPeL/7U5/dcAi4pFKu1PK+Iq0lJcX1fQ9NxsipFYNljK+8Ze0Xm7hMAldZyPWs6mu/5qXPLGV07yNy6KMNhyIf7+KbHgX80/1OPyn7AD/AMUpDqWRwNtgu+N9ZNCwTUWyGNgubiyNEn2lfj1Ej+fH1fI2M9lwvV3nV1H5kpmJBP1IyJrxAhybq8gc5yIyUTk/kZg1WtNlF4Yj98hHdC9sPXSbqv57UxaldoXoWWQ7gpIRQV4ZNkqNLkNqQVfw28wy0Axhk0EWe7gp3GGy1QO2c9Q5cfnwaR8aBEFZdPQRGiz5+Y4LZc2WXzMzQv0PwU5aMUZqhAm+Y3KoN3KSa3p61YJiBSI2bb2WWRQs+LiEf+SmKQsoHN0rSbe8MDYtXIPf5/DZ5Qh6gmNUzEOhNaDZZ9S6yRtgMLa9YVp5xTBt6KzJk1i2WPHHojAgbMvwNsDgRd6xi1EuKVOf/D7u8ppC1KxYX7FrhKnKJYPTyJanpTePHP7zmY61Q1WAtyMgZ17Q4ivnRp+FUqGHjLuhnWyGvrD03/uH3QAVoSKEeNKVd+3LjPLx6W/u5jhcUtsEMWhdJ8OGpmYWeD1uPNIkvkWzVVsgSrjwJ7ye4RqU3uQRRVqvbRHEm/gzDe1Mc2Bo/eOlkUbTlc8Pjy/ZFm/HJ5cuunD1ZanjCdrcL7Wtb05bxkVwzSE64fKJwNSmv0JsMm3FUYalgs55+qbnXn/LZnGR0G+PWX4AFxpk1x/D4NwTBTrwrTAf++nl9n1/pPDlKehWxSvkNLpkMm+fQG5BHyqN3Mc7bC8O9Iq59L1Ngs2urKaqbb8tYRwH3PjdGWnQVnpYB2I5ARiLs/I77S5wwWMLYN2t0r+8GVuuXHkAzRv+7mMqT/PcvXIkC7oJbfLrG4RjnF/PPcUJaTeXhSF+KS/IP7/wfgk2rqvqRsAtqN0ss1I1+yXmUSnGy8i8y20qRu8h6A3l3EJz+4EmCfiS3fkMeW6GUYi0LFjNehRy9MehnpB4L6DJpkYHOaU+AFJpYSy81QNFYvNWEJMlk0TonkC6P8vCFcCp5+yOM+vqhuNOmazD8EgKnubOz/tm02+m/k1/y0JTR6Ss46Br0yKWCUaSRFbSEWvbWhPFlutkJyn7aOiKfORBbeohs1nSZ9fRQ6AGwlDFZzKXDtvrYl+dK8G6wDUBLWnp6Wrutcr7hDVQgvLBxHDxwB5hmtZzUiLwCJaK3U3pOFpR7MNGuM9LoY09MVsbyPsGrCLNcBFmUt1Ca0oXJtcfzlDoHVCTNW31+SXzpLrYi3n08/XkEk+6+qimfAwrK6nEGa2+T7P+O7dU9o4s4HSFQGyjgkKQhtzlljnPoLMvHGCw/NUD5fkMo44/AWfndLaLFumfFspj0iXRUPEmOjJsUp6Hhc29ioyePEsO1HMaAb+79oc6cyOWjdmHKB7U/bmfn1Nf0kZ8OafuB7x5sGq0hC6ukFmSxIkcgnePW6Uz4g8f7K9HUY7VKdNl8/1w/07/WHPf4fvsPjhPBn1XLzjOawE0XV4xT3RxHgI6OvJyqnv5nrubTa7bMUQPbqijEMtJFubOgjQiklf+jnk0yqCnr5dnlTxuc8JAEWCuVpVPFcJvrvuP/iWZvZMQdD4UI0qWMRR1thTbTVWAzc5B/8OmC+fvEGNj67JfaW6UzHpc5M8ZE2hbatmE28gWsPCofs8odb2ulwx3fGd4fsA2/eW5hSncSFYPdNvTgVE3ytIcjEH5fQ2l5p1LgAcLAhDQmNA1Mk+PSeG9Ze7VThgj/Dt6jLd2ZLNcdpXFwZ3y5ZHR58nlCpTpFqaKMS5xy2Lpqit/Bd/iT29+32bC6odttWmWf4s6a7Bcfq5wM7JX1Lqi6n/6b96V7MmC+wdSD/ZeLFrp+07SRjCEW3Ca/FvLJ6EwGk0gdZRyk+CXz9e9CGGCKF+ZUJPMwZBvBMvP49Xnx94rt+8Yp8YFFmt87DtXYfHS+95ff5B8K/lf/rYpUyAsahZ0ptGCH+aNRzO04THtUOxAD91trW6p6oVh+dvS3sSStXyBy+CBfZ5oAyrT2UmBxOyNJtaNEBXUVkT/4bdC9WXnuZ/bvDRSP7qmjtI+bQ/sSw/bfVqSxfh+/l4+Bc/6pSw3GXv+NuqjKTWm8U/WZv0ULehb/Ab+ry3fO3dPfk/l/vg4T5BHRHhi+jdq/OmI0zI2oBm9PuItbCMo+S1j+b+zHctcNsHHDuiLkZwZeHgRq5jWzsbIZKdsdHYrJwxq6zjq8DndJ2WNtnGKR4nTt5v8lHRy8H1+LQwcNE3JmQfWbTY9YGA5xF2c+4m4ejQ9ErUy5/tkzm9z9tsiYaRQpvc4nXm23HZ4w1UnT1dWs7jlmQK2LqqVvAOMShRdeX7Q77fnGN77//7YI/PXx1JgDfbUbGl6fLgH2LHnIvE9hV95GonlmuPqI8LcDn+Dsyb6zfA7Z32LUYJZofBNUfPf2cRWpaSYsk3XrzUhnyuv6uDR43PN1nsSFP6kRzIyiu1vZWzPYV3+tjWP1YEvb/YCVZKhsGF8KdwMd0gnRMmtqgBxOjfp8Ms3dgvt8lMv2V6xT/ssuvvcvjXbaaaZWH1jTSNNcb/YJ+Uem5xJ/J/z+fvP+ZPfvoeOeHdLF06WHN3Ilbm3YzHmzLsXPO0mk1eqHt6U5m4ZK6Wd7GoXaP8PPOJs3kFyPffoZvlXVwEpjFdZ7nfaRuAU0f78vu9BlMHAlT3HZV2BOc4MNzQVwMpGFNPGpHbhbr7tX7XO6NKmbhUBrVRGTMPWlyMkbUs4Mw3c72OvV1f2cMGaB7E0Tv85G84gpQgo4x8yUdxW+nd+SOiVp5RX9wP3QKKAeyDwQRq8oyVD/gXeLM6m/fnYKGcp/rxqDLF/aHm8DLeJNE8xU9Xe/8fPk7xQflekLpzKpdzHLSy3LkBd+uRHNeq19H6aFOOupk8nYl8DtfbVyLU5m8LoE91Ltyf9/fsyuwuawzj3i1snn5wdHKl6arm9U6tv6gf8tIvgCBhwdnSLvVGMWK/rdTYAfu0A35O+8NJmIm5hjy2T9CjcFbsgHhbM9J1EcdsGqNjXz9A4UhPaJG7kdKV8xmZvt4/VD7G4TRk3iNcztEKXahrQVwJRqLHH04ASULpcMfZxlBktj70+ri8PB/vrddf4MW1cpwZeHCI8uJk6mp2BSaalgMYQjOHkmTOvg+9nr7U3O5Ck4wsCq09mtp//tazVdMm8dnswpmEf7U7Sf5r9NnbfUNvbs6IMu3d94rBzR4M65WRPdWV8vLDg8rA7IKvXGtfWPtfk7Q3C/OfTPe4ttjG4eY2cKV3qrOGMDIuWpdgdGDk+9V3+yJ8GSE9PP/I33i9OTu9VR79dxHk6XyU9/V9MJpgT7xjdmZ2nDR+9c75OhKlFAb/XrvGnhGXaMd5aZkLjh6T2oufmcvvfrJqRdSltumIF/ccaNZJBBMPuXRvvSmbWC9MDdtbd4VkwpzPJ/RN1/dkJI7/9AwkGXIaC5HDgDtjDzhl7VhAP/BVMdOpkHRbi6CrQmR8DDgxwyKHixxcdA4BK3VhtqyuZKo1u1JXb0Gy0ek+P+VVPzS+ycEY2Pp8Ad7u49ePMPi1NVpIy4Fd9uQ5mtvMAfMQd7/rd91fsTo6MDTCjoP9yJUi80p4gnrzc5d+lX3/fVF3LcxVmRgo84P/+rF6k1FQQpVJZl/lFP1swHYHfMcsFiBcsw6L+VIA4XU8jUN9jXSyU1PK+vntIuoqbPXdfa6ct/Sn6pfzX7gxqTd/q05K9nmhJHusB+xpzq6PAH4P8GFsy5fOAuHlSyiJFjDsS//K/VFLJRkdDtsDRwBg3nMgiAnN/PT/wjmdn3uEdxP9e+OJb0/2J1ijcJCAGOJ2eR5g0fLn/zIB7n8vOUh/O9p1sUS2KLLtTv2dw5t99/Z/0wftcjMw36G/B97iH2mcp5hwiewsGd2JVR0sUV0tT8F9pWNvsNW4fk1/XLf/gNo5SmgWJi0WRDhsKXT5sNyJKX3zUxdVd+/z/0+f64Pvr6aeo1Wz5bFZl57xj/Pfx7K6Cj1lXkhBwzHtk2YX7/rTlYFN34Ql/+E0A1fWUlfqPSdJh69YeYdkp5QD2v8UhHsY8ZFPM56v3feaMSsMTMj3nv8HeMehQzxf/K9yv/8vZ03+O/kD93+5xi5GtOr5gTEgRXlLM/kubQwG7rOf0P0vAPuvc2D+qbTrUpB9fy47wm/vc/LEB/EUtWsvX19+qh8cTzx70BY9UHf/Xh2nFb/TPM/IfuTnSdM55tiqYuLnvUtZXhQ/rvWfhXn/w4sX8jG2a8x3hxmFtLMXdpUkZtTrJNWIoXuEttKP1Q/Vg1/e+M8mTvM9X7zZzgJ2qhm5nE3Sj1/kC4aYwY+YY2eZ8c6vwc7vbOLbDQiBhzEFnBI4oaXVNZAyT2T/OPQDeF5y2y/Ms4HnhKRZZgiWzb2+PHafyFIgj3yBHs5OkJaOgjnOPxfshMqEd8xL8uxT+PFcKqUHnxsYk7evUr+4Y8qYtbHCsfXjxVwBzKdoy+9Xm1EoDvVG52WxKlz8AyD/Drc+4Db1Xl6iOVXqtD/OFnW3lKQ/HGv2y7E+sdCjoPM0jI74i3yUQG8KvrqE9Tms6TtcZS+jmaprjDZuBNxZ/VZ4Mkj4RkPDc2B4ng+1tLynu2uhWSvljeeOM7+29k5PbwQ43bLrBEkvfX/3mdJ+a5X7Y3vz9+1Pztm8vxR6Jz7p8tAIm8bsSKEuCNCjxnoGdgzwcSJyVIFbRCgAiaT5oInSEo+cWWgjCWmZ1RT4L8PB+Z5H2Nzzy6umSrFVh9O9hNfGqqrDXIvbqpqQPUX8XXZh33QWqexbtOYbQHvKdORx1BB1L1m8k3ZrnqrtNaotoTb3D/9TOwbzAbXY60ZX3ACCMdd8+O+vv1zatm8MeFtLlLzWKV3tTfUqpIT9LQnpwLpbDvV1o7uhVEq5mD6bQgQmPvjvl2vuzE6VvebvA3N20CFvqvULeg61ZFxfFnoDz1pdQ2ay2/H9cGmAGkeUB1X6Du+TBZdx3YpTdh4IWrNWhHpJnOPAW6p88+kxJ+DM5k1LkyvJR9SNcbzbQ2BzPz/h/4E
s5iEA7U8CY+l1RhpR2LFCsfx6FRHD/9f
p5XIbqrzQnV7wDQNm+p6SJ+OnKqX0yzqqcJr/2MmGKrPNFuV3n6A10k
JqXXhurwmKPsZbjtfspe8MGcV+DQSxoOQ8oqN2ZwH+77m6mlfxdqSSD4Lq
e+d5UoWvt66RMP4S9D7U0drBXJpx1l5MfUcmJJ4XrqjVKc4EV3MpV0h+Z5
CIrRzCx27u6ea2jUTE6LdPPO0v3rW0H/bXV8q2O3s9f8tBXlRqp3NX1n7r
uFz+9dV3vwvhm9Eo7U4v0/J1hC+y91mbrp1rh+GAeaJ23Qc/qXz7t/NQ5Fz9/
+XQ/YW+t/6fx94wC6v+v4ZWVzgD1hrt7O16ub9t7tGE/yAEvS+79aV9cA
8l/gqOqJ2TTEXm+7DR7z18IT/Io/OdHJnN+rm2ozwGHk9mN2gpP3oN99iS9D8/UgZ8OqHW6ELGPAnTqfYhbcLnP38AHoprT3Qyl2sa8TXqz
+OXn6MD5vYxBL43Tkc7DY/VtPq4hzGzPeSJsbLUrCl6OHPYB5vlhlQ0T
QGmXnJQmcCMKTrgCXlqGf1OJ9xGbqSxG8Cu4bR9vAYYaehpRqYlt8rQ
XL6Qnf73R4UbZ+8LT3vIy+xUjXTZ6aAbyhHyOPQKhXnpUs2tYqQAHiiP
KRdp9qbDgRkfHv3jE56Xd27+vPdSD9n/7L5SqAVw9PM7Nz/8T+8x7Jrpm9/sf
3rsPTaGu8z8HcCd3lp2Mr3HVsF+HxrpG8xuNTpDqPTxMLnR/vh//bXyV
Nvn1Wn2zdecqT9cs4+d8/X58/88fn9/6e7+d/j92b2fsF5k/+75wAz/p/T
2nm3G3+rNSNdCrsy2B+Wrmy6eLCXbo8XXsW1vvdPeQD9iZN7EdJy3JP5f
i6EaWcsk0CrzT6Xf8WE1a6qzlOSEJY8hd0nYRD3j/IxwezgrTI9MmuP5N0+n
z2w1vfq7m3k58NLVZVQPdm6JhVpWdEqbmVS6Bdj8ZY1TaLyFFv93zSr
N99Se5xrPWwNjHqpRb5ZN/VqMZOOfkGDFxK5tDyV/Z+fZpnmZ0LPKIz
h0+ewQvE9wWq4zhw6VuF5qZ0d6Ts3H1Mhi0fiFXWMnaQ6pqh6LgMz1q8v
5sJSahA3Ni0jWKv83Hz2XXkBw48w8vxGjXnvs0m3lujx/uc2fpVqcAq5OK/Fs
XTv9w+zz71vP+v7/zPIrMXbdbH4tUg65+xo9xkXS0i6JPEwfbF+YZv7A
0vPv0b76+vP6CF1d4Kfr2l8cz9nf/4L3X/3yVz34GbC+7Nhj5b97Dgf+vh
b/jH9v8PCq/9Pvr3qvqPf38iP0DxrBSm25H/75sz/e/vnPw9m/+9//7//
t+f//K//yKs+6iWf+F//l+x39vr3zA9vP4/+/u39/7+1UN/88r3Y/fEv/v
8X/53/8wv3/Kr++/yMv6lRe/lV/+v0v/s/V3n7z1v/+fK3///+v8r1l9X
Pzav/7Er//38J//L/tlX/+K//W/+Lqd/mW+++M//FXnvoWv/93Ee//9/5w
dzXn++/5PvvtX9///cp/9z4/vN6n//v/f/6P/6/6f///8/ff4//9//++/+d/
+dP//X//8/v/5/f//vn/pX//+7/+r/+d//d//+t////3/v//f///H/+//++/

7v5nvrqgYI1jPUWx1rbALexAgyBD4+26Hebt/bIK4BdmqttwRePwtcrZTq+J5
2fe9sR+emKaWuVUpt6XCvyXT/s3oa+Grg869iBDq01vuAbwPLgUql1AvdPC
DwDPAO8IJSqlwp9csudQohjkNCmBCDjFIqBzgd+KpSqkIpVYHR6/KFwJD
TUbrD5yXAR1rrpA5/4rTWtwbOP4cx7JWrtU4EHqfLPK0ODgF5Sqm284Hh
0QyguAffwmGM4a2jX6s6vg7UenOXWu1a64+7+d66eh+YE3ifutMCxHR4HS
wwdb3H8xjDe/MBO/Bhh3qP9952dS3GHL2jP7uHMULUecepF4BAT9yfMY
YSU7XWScA2uv9Z/QO4SCk1DWNo9ZW2b07r57TWizBCpAZ+0fWLtdYerf
UDWuuJwAKMXr9rT1SnEMIgIUyIwecaYA8wDmOIbzowFiilw7ylLt4Axiqlrl
FKWQJ/ZiulJgTOxwO1WmunUmoOxlOF3fkEcAI/CEzcjgUeBDbSsxD2JjBJ
KXVJoHfnW3QOQY8D9xydU6WUSlRKfbkH7QKgtX4fY47Wy0qpWUqpKK
VUvFLqFqXU1wOXbcGYR2cJTHoPOnTYxVsYgeUnwItaa3/g+Ine2zaBADc
KY/jw6M9uMkYI7tqTGUwsRmCqCrR3feDrg9JalwIbMHq0/n10+FYpNU4pd
bpSKhrjZ9mKMUTZtd5lSqkpSikz0IjR03nMdUKI4CSECTH4XAf8QWtd0fE
PRngJ+os8MJx2NsYwWjnGcNcvMOYFAdwG/EQp1QTcD/yzu5sHhj/PB5Zi
BL8DGEOMlweGFo9La10NfBkjuNUAY4A1Hc6/HKjtBaVUI0ZPzwl7ibq4D
CM0vYgxL2obUIjRSwbwI4wwVIcxuf25HtR9dNj3zI7X9+C97eg64NXA/LuO
P7vfARcopVJOUMMO4NfAWuAIMIUO7103ng5c90yHY9EY7391oN4MjIch
usrCWO6jEdgJfITRuyaE6AHVg38ThRBCDFJKqSUYwamgQ++dEKIPSE+
YEEIMUYFJ9HcCf5EAJkTfkxAmhBBDUGBOWj3Gk6K/jXA5QgxJMhwph
BBCCBEB0hMmhBBCCBEBEsKEEEIIISJAQpgQQgghRARICBNCCCGEi
AAJYUIIIYQQESAhTAghhBAiAiSECSGEEEJEgIQwIYQQQogIkBAmhBB
CCBEBEsKEEEIIISJAQpgQQgghRARICBNCCCGEiAAJYUIIIYQQESAhT
AghhBAiAiSECSGEEEJEgIQwIYQQQogIiIp0Ab2VlpamCwoKIl2GEEIIIcQ
Jbdq0qVprnR7s3IALYQUFBWzcuDHSZQghhBBCnJBSqri7czIcKYQQQQggR
ARLChBBCCCEiQEKYEEIIIUQESAgTQgghhIgACWFCCCGEEBEgIUwII
YQQIgIkhAkhhBBCRICEMCGEEEKICJAQJoQQQggRARLChBBCCCEiQ
EKYEEIIIUQESAgTQgghhIiAsIUwpdRTSqlKpdS2bs4rpdRvVL7lFJblVIzw
1WLEEIIIUR/E86esL8B5x7n/HnA2MCv24Hfh7EQIYQQQoh+JSpcDWutV
yqlCo5z/gXgb9qYr7FOKZWslMrWWpeFqyYhhBDi5Kr0kp0VqUfBb3q2LDVa
XqPbzDBsdbOHNy3uJuyGDKbs0OL90fq/6u3KYatOvlxwfZaOvB4pBdIaOE63SgFIbrbWuHqSvymJtB1h
WgyPvKwnY9pVa6aQVgWqmT2gcTyM5cYnkU9nr1X1sBwjXd22gOqVVF2
TWCNC9wLxO6qyy6ph1PSdnixEy86EDVq8Ko6DG9urbhq1Otqr2rBVq9O
KXOKCIuHXEyRW8MkR/lT4OaKKhkO/ak/ywlzNX2FURhjVFJcLlzbsGJfo
ajyqCG/nRlTvBtutz5mg9q4FHzPpGLhNd62WRqOQIw0nGsfVEj7cAjOZG
CIQoaGQqyL6w1KhyP2RcY/zjs/bWvYg6mtgeWYHrbOl1sZ

Ca/fS52zLmiIqnHWHBO4vH7vMW0oFMm2ZCNU2VKZnDaZVFtqIGCldP
o8xZaCLSoQOltqoGgVFK2Gojehapdx3BoHefNh+nVG8Bo2rd+Erq7CFsKU
Us8DS4E0pVQp8GPAAqC1fhx4C/gCsA9wANeHqxYhhBD9y7aPSinZWQdA
c50TgNnnF5CcFRv0eo/Pz7oDNby7/QiVTc5e329ltJuzJ2Vy15Uzjjm375zl2Kd
MZfhDd/e63cGo1dvaqXeq1ll7TLg6GqzqXfVB27CYLG3BKc2exriUcW2BKs
We0qknKzk6GXNPQpKjFva+Z4Sug6ugcnvgZrGQNw+mXgEjlsCw6WCOZB
9Tz4Xz6cirTnBeA7eH6/5CCCH6r60fltJS7yI+1ejVGD42ibjkzsNqTo+PlXuqe
Ht7Be/vOEKj00uM1UxuckyvO6nyU2NYMiY887P6u47DgF1DVLDhwFZva9
B24ixxbcFqROIIZmXOItWeGjRYxVniUKfYk0hrPRR/HOjtWgUV2wANUXb
ImwuTf2T0dA2fCebw9KCG28CIikIIIQacigMNHDnYGPRcc6Mbf6YNx+IM
wBgOeWbDIQD8WrP5UD0f7q7E4faRYIvizImZnDd5GIvHpGGznLjXxO9w0
PDa62h3h6UmthdRuz3ItQ3Ba+zPPH5Pp2HAYMOBHT969bHDgCZlIik6qS
1YTU2f2haiug4HpthTiDaHeejT2QiH1sLBlUboOrwV0GCOhtw5sOze9tAVFd
lh2FCRECaEECIsVjy7i5qylm7Pr69o5aM3qoOeS4uL5uIZwzl3chbzRqZiMfd
ubfHmlSupWL68x9dbsrN71X44ODyOtqG/Ew0HNrgagrZhNVnbwlN6TDrjU
sZ1DlQdglVSdFLPhgHDxdUEhz6BopXG8OLhLaD9YLZCzmw47W5jIv3wQ
rAMjIcPektCmBBCiONqdHooqQ3+pNrxtLp8pI1LYtyF+aA1lJWAxwPAQ+/s
Ij89lo1nBV8iIMZqxmRS4K3Gt6caXy/v7S4qBqDg//6JtQdLG5kSEnp5hxPzaz
+Nrsb2YHV0ODBIsKp11nY7DBhviW8LT6OSRjHbNrvT0F/HYBVriT31YcB
wcbfAoXWBifSroOxT0D4wWWD4LFj8PaOnK3cOWI59gGIwkhAmhBDiuK
7/6wY2Fdf16NpEn2Ksx8wYj4nhPjMbGpq5+8+HmXlkNz9b++e26+4LfKx6K
gwFdxCVloY5MTFk7Xn8Hmpba4OGqI7hqqa1hjpnXbfDgMnRyW3zqHIzctu
H/YIMB1rN1pDV36c8rVDyidHLVbQayjaB3wOmKMieCQvvNHq6cueCNfg
DGYOdhDAhhBDH1djqYWZeEjefNuqYc1pr3LUuWvY30by/EXe9MQcrOt1
G7Kh4zp+YxEWxFmLWNMBaqL3hDnypxjywURmxJNjCN6HanJyMZdiwE
17n8Dg6DwEeZziwu2HAaHN0W4jKjMlkQsqE9knrHYcB7akkWhMjOwwYL
h4nlG5oXzaidAP43KBMkD0D5t8GBUuMJxmj4yJdbb8gIUwIIYYg565dHLr
xG2hn5+UeNFBrzWTb+Bvxm4yAdGng3MFPPz22IaXwmaNB+0lsLia/fjsZd
duwuzv3nGmPBw0UXnoe0SNHhP4b6sCv/TS4Gqh01lJzeP0Jg1W3w4DW+L
YQNTppdOdeqg5PBKbaU4mJium/w4Dh4nUZvVsHA08vlqwHn8sIXVlTYe7
NxvBi3nywhX64dzCQECaEEEOQu6gYX3U1CRd+kajkZACONLpYvbeKZrK
ItSYQ4y/DhLEJdaLdgt0avPcm3uIk216HPcoDpAPLgl5nTkrCWpB/UvV6fJ5j
w1Q3Sy10NwxoVuZOi4LmJuQGfQrwaPAasMOA4eJ1Q/nm9on0JevB2wooy
JoMs280hhfz5oM9KdLVDggSwoQQIsI85eU0vfcejo0b0d7eTkFv59Vm9jCBJ
oxeB7/WtLi86CCbvZm9HmyTb+aQdwKeRhs+DbUtbmy5JkbH23FWOrlw+c
WkZodn2EhrjcPraAtRXXuqugarRnfwZSRsZltbiBoWO4xJaZOOmVvVNgwY
nYhJ9e4pyyHN5w2ErkBP16F14Ak8oJExCWYFVqTPXwAxKZGtdYCSECa
EEBHgOnCQpvfeo+m993Bu2waAJT8PU+zJTVCusuaxLX4pTlM88d5qFBqf
X+PxmzEFGSXzR0XjsiTg9UVBqw8zUGCPJiPBhklB2vjkXu9JeHQYMOh6
VUF6rZy+4CvfJ1gT2oLT2OSxx8ypOjocmGpPxR5lH3rDgOHi80LFZ+0T6Q

+tBXezcS59Akz/itHTlb8IYlMjW+sgISFMCCHCqHXLFhyfbm577auro+nDD
3Dv2w+AbdpUMr7/PaxLT+eVKhNOz4k3sfZXOdHV7YuQemscqEMuPPFeD
k8ppjHZ2ErmSKOTz8sauGnJSDLigweqbEo7vXYHPjqB5/dvDfo1RxcK7Rqs
6px1+PSxPXlmZW7b9y/Vnkp+Qv4xc6o6Ph1oGaCrnw84fh9UfN4+kb74Y3A
FehzTxhrbABUsMnq74obmbgPhpnSwfup+rLCwUG/cuDHSZQghRI8cvOzL
bT1dAJhMxBQWEn/WWcSfeUbb03vv7zjCjX8P9m+bD2VpwBRVb3y01HNV
6TwyXO3DPz7lZfPw9/l0+Lv4TSc/nNkb9ih7t3sBdv08ITpBhgH7A7/f2G/xaE
9X8WpwBp72TBll9HIVLDaCV3xWZGsdRJRSm7TWhcHOSU+YEGJI8TU2
4i4+1Gf38zc3E7tkMcMf/g1ev599ta00RFlpALRX03ywmBrnEdYU78OSvJ2L
Z8fgUbVU1dXiqPVQ76rDeGaxnV3NpTqjiOr5n5Mek0FGfCYXxS/lppgryIzJI
tbSPo/LYlZER4V2OQSTMmGPGhqLaQ5ofj9U7TJ6ug6uhOI10Bp4ajW5AC
ZcaGx4XbAIEiK/Y8BQJCFMCDGklH7rThzr1vXpPUuyUrj79afYXnkIl67BZ
KlHWeqNjyZP23W2LHivzEJ27DCWbLqGuJqMbtucNXUiZ5339b4oXwwUW
kPV7vaJ9EWrwVFjnEvMg3FfaO/pSsqNbK0CkBAmhBhi/E1N2CZPJu3220
6tIQ2NnkbqnPXUOWupc9VR66yj3llHdWst1Y4aWn3GpOZ92Z/R6N4KSZAS
lUxKdAbJ1vEkWTNIjs4g2ZpOsjWDvMThLBszEpMy8cL29USNMFF4XkH
Q22eOkHWXhjytoWZfoKcrELpaKo1zCcNh9FntQ4zJJ7c0iAgvCWFCiAFvX
2UTl/9pHa3u9vlQce4WCsu3M6/scyZV7cPsNya8W30eNg2byP/7wNVdc0Etaz
Iz1hmY16R8oPyAFcgI/AEwntJL04q0wOcmZWJOvcKsTMd9is+NZh+l7AtMl
Pd6/Iycnk7B1LRe1SkGMa2h9kB7L9fBVdBcYZyLHwYjlxq9XCMWQ/IIkKd
G+z0JYUKIAe9QrYPaFjdXjYphSvFnDN+6lrR92zD5/TiS0jgyazFeW0zb9Y4p
c7hmRO96BuI+qsKv3OxN2Yjb30S8eRgx5jQsyk6UsmNRdszKgkJhNilyku1k
JthOafmEUTPkibQhr66ofUX6otXQWGYcj83oMJF+MaSOktA1AEkIE0L0S/
6WFqof/xPO3buCni+vd1LdbPRmebx+ftPUwLhXS1FaY83PJ/6GG4g/+yxsky
ef8jpS9U2NPPPBQcps+9gz4b/8aN6PWJKz5JTaFCKo+pLOw4sNgYdIYtICv
VzfNUJX2lgJXYOAhDAhRL/TvGo1FT/+MZ7ycmwTJ4L52Kf7Go80ob1+oq
NMWAEdYyfhiltJP/9crKNHh2wBzzdXfMSOV2qxO+PJWBDDKxe9Qqzl5BZ
UFeIYDWVG2Dq6FVB9sXHcngIFC2HBHUaPV/p4CV2DkIQwIUTYtW7Z
QsuGDT261rVzF41vvYV1xAicv/4jK2KDP8X19MdFTBgWz1+vnxOSGl0OD0
Wf19BSb/Su1Tvr2bh9G7GHhuGP9TL1xkSWFN4YknuJIayposPw4ipjjheAL
cno6Zp3q9HTlTERTLK22mAnIUwIEXZHfvkrWj/9tEfXKouF1FtvIe2WW7js
yY18eij4cCTAuZNPbUFJR6Obg59VcWBzFaW76vD7O6/HZVPpMKuab11zI
XZb77bwEQKA5srOE+lr9hrHoxONPRcLbzB6ujIngym067mJ/k9CmBAi7Nx
+E+6FXyT9rruOOedweilraG0/YDZzxGyGT6sw13k4b1gy95w3IWi7FrOi4kB
Dr2rRGiqLGtm/uZLD+xtAQ3xaNNYZzaywvM42/0aSbclcOvZSLh93KVnxm
b1qXwxxLdWB4cVA8KoK/E+ENR7y58PMa40er2HTJHQJCWFCiPDbHHs
m1ZZc+O3nvfq6pQA4eX3n5uNfeBJSh8cy9oxktsSt4h81z9PsbWZS6iR+MuE
Bzik4B6vZGvJ7ikHIUWusRH90iLFyh3HcEgt582DalVCwxAhdZvmVKzqT/y
KEEGHnVVYSdS2Lv7XsmHMPvL4D7ddcMmt40K8dkRpLvD10Gzpr7eeAfx
f/qnya1WWrMTvMnF1wNldPuJqpaVNDNqFfDFKt9cZG10efYDyyDdAQZY

e8uTD5UmMroOwZIBuRixOQECaE6Nb+qmYu+cPHnRZB7YnhjUf4xYrfY/
O5aY7NwTHlFvzOWs56bv0x17p9fpaNS+eL54wKVdmd7Kvbx7VvX4vT6wR
Aa41Xe0m1pXLztJv58tgvkxHT/fZAYohzNsChdcbei0Wr4PBWjNBlg9w5sOxe
YyL98FkQJb2nonckhAkhulVW10pDq4eLZwwnK7HnE9PTd9YR818vW+fdR
p11LCY8qAnp3DBlRNDrz5wQvhBU1lxGk7uJC0ddSLrdWPx0dPJozs4/W4Y
cxbFcTUboOtrTdXgLaD+YrZAzB06725hIP7wQLPKwhjg1EsKEGCRqW9zc
89JWHL3stTpRmwBfnZfHrPyUbq/zt7RQ/cc/4tyxE4DWBgdrCu+h1ZrJhIXD
WHDJaGyxkR2auXr81UxKmxTRGkQ/5G5pD11Fq6HsU9A+MFkgpxAWf8/
o6cqdAxZ7pKsVg4yEMCEGiR3ljbyz/QhjM+OIiw7NX+3oKBNLxqYzKj2u22
uaV67k8PLleA9XYJsyBWUy0WDPoTUmk2WXFzDx9JEhqUWIkPC0Qskn7
SvSl20CvwdMUZA9ExZ923h6MXcuWGVRXhFeEsKEGGR+dvEUZhd032t1
KhybN+P45JO2187du2n6z9tYR40i/9lniZk5AwC1vQYe+YyUgtSw1NEbXr830
iWISPI4oXRD+/Bi2UbwuUGZIXs6zL/dGF7MnQfR3f/PhhDhICFMCNFjlb9
6qNOiq8pqJe2220i95WZM1v41v0przWv7X+OXG36JzWwjM1bW+xoSvC6jd
+vokhEl68HnAmWCrKkw92ZjyYi8eWBLiHS1YoiTECbEYKE1I+vLMG37D
EdNeH65+BobiF0wn9w//ck4YDKhAvs6Ohrd1Fc6AKgtawnL/XuqtKmUn6z9
CWsPr2V6+nSWL1hOmj0tojWJMPG6ofzT9p6ukvXgbQUUZE2B2TcaPV15
88GeFOlqhehEQpgQg0TUnh08tuI3sAKKw3if6BEjUZb2SfZ+v+bzD0tZ99oB
vK7ODwVYovt2RXCv38uzO5/lsS2PoVDcN/c+Lh93OSYle/ANGj4PlG8xNrwu
Wm1MqvcY4Z/MyTDra8acrvwFEBOeYXkhQkVCmBCDhGo1tv5x3XQHY+
ZND9t9oie0byFUU9bMh//YxZGDjeRNSmXa6Tkok7HYqdUWRUp2301s3l27
mx9//GO212zntJzT+OG8H5IVe2p7S4p+wOeFis/ahxcPrQN3s3EufQLM+Krx
9GL+QoiN/BxEIXpDQpgQ/dzhBx6g/sV/nvC6eG1sPu0bP5nYBQvCXRYNVQ
7++fMNWO1RnHXDRMYUZkZktXmXz8WfPvsTf932VxKiE/jVkl9xTsE5svL
9QOX3QcXn7cOLh9aCq9E4lzYWpl5hDC/mL4K49MjWKsQpkhAmRD/n2rM
XS1YWCRddeNzrSmpbeW5bNVeOGdcndTVUtuL3ac69aTLDxyb3yT272lixk
QfWPkBRYxEXjrqQuwrvIskm834GFL8fKre393QVrzFWqQdIHQ2TLzF6ug
oWg2ymLgYZCWFCDACWvDwy7ryTj/dX8+eVB3B6/MdcU5/iYeeYRq60hG9
R1M9XlLJ/cyUAzmZj6QdzVN/Pt2pyN/GbTb/h//b8H8PjhvOns/7Eguzw9/6JE
PD7oWqnMZ/r4EojdLXWGeeSR8CEC429FwsWQUJ2ZGsVIswkhAkxAHh9
fu7+11Ze3FjCsEQbuckxx1wTHx3F6eMzGJMRvrWOdn9SQf0RBynZsVjtZv
KnpJKcdWwt4fTfQ//l5+t+TrWzmmsnXsvt028nxtK3NYhe0BqqdgdWpA8skOq
oMc4l5cG4843ANWIxJOZEtlYh+piEMCH6GW9tLQ0vvYTf6QKg5VAJO81J
/OvTUm45bRTfPnMMNkv4njos31dP6a66oOea61xkFCRw4bdCM/G/rLmMte
VrqXJU9ej6XbW7+KDkA8Ykj+G3y37LlPQpIalDhJDWULMvsOH1auNPi9F
7SkIOjDk7MLy4CJLzI1urEBEmIUyIfkJrTePrr3Pk5/8PX31923ETUDFuPK/
evpDJwxPDXse6V/ZzeF9Dt+dHzTz5ydAOj4MNFRtYU76GteVrKWos6tXX26
Ps3DHjDq6ffD0WU2T3ohQBWkPtgfaJ9EWrobnCOBc/DEYuNXq5ChZDcg
HIAxNCtJEQJkQ/4Ckv5/D9P6Zl9Wps06fTevtdtGbnAfDjV7eTGh/NN8MUw
LTWVB1qwhNY48vZ4iVnfDIX3hm8t6s3Tx36tZ9dtbv4uPxjPi7/mM2Vm/H6v

dij7BRmFnLFuCtYkL2AgsQCFD1rV556jDCtob64fSJ90WpoLDPOxWUaPV
wFi415XSkjJXQJcRwSwoToByp++jMcmzYR9e27uIdxrHvzMHC47fxZaeFbb
+vIwUb+/ctNnY4lZ8WcdNipclS1ha51h9dR66wFYHzKeK6ZeA0LsxcyI2MGV
nP/2uZIHEf9ocBE+kDwaigxjtOouRYAACAASURBVMekBXq5FhlbAaWNk
dAlRC9ICBOiH/A2NFCbO4avlQ7DFtXMTy6axOgOE+wnZIVvjzuP0+gBW3
T5GFKHG/dMy+n55H6Xz8WmI5tYW76WNeVr2Fu3F4AUWwoLshewIHsB
87Pny7ZBA0lDWftE+oOrjJ4vAHuKEbgWfMsIX+njJXQJcQokhAkRYa2fb6
N2z34O2jI4c0IGy784iYwE2zHXeT0+Nv2nmM3vHcIXZImKU5VZkEDWyB
MPeWqt2V+/v21e18YjG3H5XFhMFmZmzOQ7s77DguwFjE0eK9sFDRRNF
R2GF1cZc7wAbElG6Jp3m/ExYyKY5GcqRKhICBMiQvytrVT9/hFqn34arz2e
zad9iT98ZVbQa8v31bPiH7uoq3AwelZGyJeFsNqjSM+P7/Z8nbOOdYfXtQ0z
VjqMp91GJI7gy2O/zPzs+RRmFspSEQNFc2XnifQ1Ru8l0YnGnouzbzTmdW
VOltAlRBhJCBOij9S98AKNb7zZ9tpdWoq3ooKkK67gSRaT0mLm5V9/eszX+
X1+Kg40Ep9i44t3TCNvUnj3x3tu53O8W/xu2+tmdzN76vag0SRYE5g3bF7bM
OOwuGFhrUWESEt1YLmIQPCq3m0ct8ZD/nyYea0xvJg1FUx9u+m6EEOZ
hDAh+kjjG2/i3L0bW2AD7OhxY8n+5S+InTOH5O+tINbpDzq9xhxlYuY5ecw
6rwCrLfx/Zd86+BYHGg4wPmU8AKn2VG7Nv5WF2QuZlDoJs/yS7v8ctcZK9
EeHGCt3GMctsUbomn6VMZF+2DQwy68BISJF/vYJESZ+t5u6v/8dv8MBgL
u8DNuECeT//emg1zfbTXzpuzP7ssRuTU6dzBNnPxHpMkRPtdZB8dr2nq4j2
wANUXbImwdTLjOGF7NngFnWVxOiv5AQJkSYOD/7jMqHfm28CHRxxS8
7PYIViUHD2dAeuopWweGtGKHLBrlzYNl9xkT64bMgSpYCEaK/khAmRAh
5a2tx7Tbm2zh37gIg7+9PEztnTiTLEgOdqwkOrWvfCujwFtB+MFshZw4s/UE
gdBWC5dgna4UQ/ZOEMCFC6PA999L80Uedjpnjwrehthik3C1G6Dq6In3Zp
6B9YLJATiEs/r4xkT5nNljska5WCHGSJIQJ0Utaa/D5gp7zt7QQPX48WT+8
DwBTbCzR48cHv9avjS1gAAXosFTbzuf3oXtwl55cI0LM7YDS9e0T6cs2gd8L
pihjSHHRt405XblzwSrLgAgxWEgIE6KXKpY/QP2LL3Z7PmbePGIKC4/bRv
0RBy/+dD3ewKKriUBdfPjWY1pZupLvf/R9Wr2tPbp+4fCFYatFAB6nEbqOb
gVUthF8blBmY/L8/G8aPV258yBaelKFGKwkhAnRS+6DB7FkZ5P05cuCno9
dsOCEbTTXu/B6/ExYMIyENBsvrC+hITE8Sz+UNJbwg1U/ICc+h3Pyz+nR1
ywavigstQxZXheUbmxfq6tkPfhcoEzGMhFzbzF6uvLmgS18W1QJIfoXCWFC
nARLdjZpt97aq6/5+9oiXt1SDkBSs4+ZwLNHaqhvMbFbtzIl9sRbBvVWq7eV
76z4DgrF75f9npz4nJDfQwThdUP5p+3DiyXrwdsKKMiaAnO+0R667EmRrlY
IESESwoToI29sPczeI01MzUki2gzgITrKhN1iZnpuEhdOzw7p/bTW/HTdT9lT
t4fHznhMAlg4+TxQvgWKVhrBq+QT8Bjrw5E5GWZ9zRhezJsPMSkRLVUI0
X9ICBOijyQ2+zhP2/lSWhpNyslenNx3/gSGj00OSftOr5NndjyDw2v88q9yVP
Ha/te4ddqtLM5ZHJJ7iACfFyo+a+/pOrQO3M3GuYyJMOOrRk9XwSIJXUK
IbkkIE6KPFBz2ktTsZ8t7hwCIjo0iPiV0azp9VvUZv9/8e8zKjAosDntewXncMu
2WkN1jyPL7oGJr+0T6Q2vB1WicSxsH0640Alf+IohLj2ytQogBQ0KYECHU0
uCiprQ56Lkor6Y+zsR9Dy0N2f2qW6vZXWssDruzdicAT53zFDMz+8f2RwOW
329s/XN0In3xGmOVeoDU0TD5UiN0FSyG+MzI1iqEGLAkhAkRQh8+s4vib

TVBz8UBjSmhfQLyh6t/yJryNZ2OxVpiQ3qPIcHvh6qd7cOLxWuM/RgBkkfA
xIuMDa8LFkJCaOfuCSGGLglhQvSA9njQfmNNL7/Ph48omh2eY65ztnpJyYl
jweWjASitbeUPK/ZxsNrBnBHJfP+yySGtq9XbysTUidwz5x4A4ixxjEoaFdJ7
DEpaQ9Xu9r0Xi1aDIxCek/Jh3PnGRPqCRZAoDzQIIcJDQpgQJ+AuKuLAR
V9Cu1wA7Bx7NYezF8J3VwW9vjjKxz1PrW17nR4fzU+um855U4aFtK4WTw
tHHEfIT8hnesb0kLY96GgNNfsCey8GQldLlXEuIQfGnN0+kT45P7K1CiGG
DAlhQpyAp7IS7XKxf84ZrGyOJjtlAlarHz0++FON+Zk2/ic1GgBblJlLZ+aQG
GMJaU0un4tvffAtKloquHfuvSFte1DQGmoPGIHrYCB0NVcY5+KzYdTp7X
O6kgsg8CCDEEL0pbCGMKXUucDvADPwF631g13OJwL/APICtTyktf5rOGs
Sg1tti5s7nv+UZlfwvR1PxuiSvcyadgdF9jyyhyVgd2oy8uK5+LbITH73+D18/6P
vs6FiAz9f/HOW5CyJSB39itZQV9Q+kf7gKmgyFsYlLtMIWyMWGx9TRkroE
kL0C2ELYUopM/AYcBZQCmxQSr2mtd7R4bLbgR1a6y8qpdKB3UqpZ7XW7
nDVJQa3fZXNrNlXw9ScRJJjrCFpM9Fkoy45j+GJmpQcY0uZ0bMi80ScX/u
5f839rChZwX1z7+OCkRdEpI5+of5Qey9X0SpoKDGOx6a393IVLIa0MRK6h
BD9Ujh7wuYA+7TWBwCUUi8AFwEdQ5gG4pWxqFEcUAt4w1iTGCLuPnc
8C0en9fh6v8tFzZNP4m9pQWsoa0mi1mU8ZdjS5KMMWDjPzuiLIzf3SmvNg
+sf5I0Db3DHjDu4cvyVEaslIhrKOgwvroL6YuO4PcUIXQvvNEJX+jgJXUKI
ASGcIWw4UNLhdSkwt8s1jwKvAeVAPHCF1trftSGl1E3ATQB5eXlhKVYMb
c5t26j+/SM44zLYPfpyapJGYfJ7jGEuwOpvJmlUQURrfHTLozy/63mum3gd3
5jyjYjW0icaDwd6uVYaH2sPGMdtSUbomnebMcSYPgFMpsjWKoQQJyGcISz
Y/4rqLq/PAbYApwOjgPeUUqu01o2dvkjrJ4AnAAoLC7u2IUSPaK1p3bgRX0v
LMedce/dSMvw0Dk64HMxmFn9pJJNPy8Fk6h89Kk9vf5ontj7BJWMu4XuF
32tbEX9Qaa7s3NNVs884Hp1orM81+0ajpytzsoQuIcSgEM4QVgrkdnidg9Hj1
dH1wINaaw3sU0odBMYD68NYlxiinNt3UHzNtUHPNcXlsLfwHrKHWTnz1s
KQbid0ql7e+zIPbXyIs/PP5v559w+eANZS3XkifbWx8j/WeMhfYGx6XbAIsqa
CKbSL3AohRH8QzhC2ARijlBoBlAFXAld3ueYQcAawSimVCYwDDoSxJjGE
aWcrAJk/+iH2qVM7nasod8PLDcy+dGK/CmDvFr3L8rXLWZi9kAcXP4h5II
cRR20gdAWCV2VgeqglFvLnw/SrjeHFrGlgltVzhBCDX9j+pdNae5VS3wTew
Vii4imt9Xal1C2B848D/wv8TSn1Ocbw5d1a6+pw1SQEQPTIkdinTOl8zFYHb
I5MQd1YU7aGu1fdzbT0aTy89GEs5tCuNRZ2rXVQ/HH7E4xHtgEaLDGQO
xemXGZsBZQ9HQba9yaEECEQ1v/d1Fq/BbzV5djjHT4vB84OZw1CDERbKr
fwnRXfYVTiKB4941FiLDGRLunEnA1QvLZ9K6DDWwENUTbInQPL7jN6u
rJnQlRolg8RQoiBTPr8xYD3tzUH+denpQC0hHCR1kipclRx2/u3kRGTweNn
PU6CNSHSJQXnaoJD69q3Ajr8GWg/mKON0LX0B8ZE+pxCiIqOdLVCCN
HvSAgTA957O49QUttKYX4yxMPk4YlMyu6nwaUHDjQcoMnTxEOnPUSav
edrnYWdu8UIXUcn0pdvBu0DkwVyZsOSu4yJ9DmzwWKPdLVCCNHvSQgT
/Z7Wmje2HmZLSX3Q8werWhiTEceTX5t93Ha8Xs2BgvOp2ODGUrq307nm
OlfI6j0Za8vXsqrM2BC8osXY49BqjvCQndsBJZ+0T6Qv2wR+L5iiYPgsWPQd
I3TlzgXrABguFUKIfkZCmOjXSmod3Pvy56zaW43dYsbczbpd5+QknrCtmmof
RQVfwLzbh+lA19VSICbRSnxqZHpw/rT1T2yu3Iw9yrh/hj2D7Ljsvi3C44TS9

UboOrgKyjaCzw3KDNkzYMEdxvBi7lyIjuvb2oQQYhCSECb6rb+uOcgv396N
ScFPLprEV+fmn3DxVK01ZXvqcbceu/vVkQpjvtgZZ1gZc8misNR8PBUtFeyo
2RH0XJ2zjsLMQp4858m+K8jrgtKNgYn0q6FkPfhcoEwwbBrMvQVGLIG8e
RAd33d1CSHEECEhTPRLOw838sDrO1g8Jo1fXDqV7KSe9VBVHWri1d8cf
6kJqzUyi50uX7ucNWVruj0/InFEeAvwuqH808CSESuN0OV1AgqGTYU53zB
6uvLng+3EPYtCCCFOjYQw0S+1uIyerJuWjDxhANN+jdtpXO9odANw2tXjy
CzoPDnfuWMHlffcRdp1vwhDxQa/9tPsaQ56rsXdwsTUiSyfvzzo+fyE/J7dxO0
whglPSEP1PiNwHVxlzO/yOIxTmVOg8OvGnK78BWBP7tm9hRBChIyEMD
HgffiPXez8+HCnYynDYkjP6zyE5qg009xaFdZa7vzgTlaUruj2/Pxh85mQOuH
kb7D/A3j+qkAPVi9kTIQZ1xjrdOUvhJiUk69BCCFESEgIEwNeU62T+FQb0
043tiq1RJvJHNH3w2klTSWsKF3B2flnMz1jetBr5mTNOfkbuB3w+rchYbixm
XVPJGQbvV2x/WipCyGEEICEMDEAtTa5+c+fPsfdaky0b6huJT0njmln5B5z
be3TT1P/0ssA+B2OsNb1yr5XMCkTd82+i6zYrNDfYOUvob4YvvamEayEEEI
MaBLCxIBTd8TB4X0NZI1MxB5vISHNxqgZ6UGvbfpwBd6KCuyzCwGwz5i
ObeLEkNfk9Xt5Zd8rLMxeGJ4AdmQHfPwITP+KBDAhhBgkJISJiHF6fDz6
wT6aXUGWk2jsPOepeFsNxdtrAHDUGwurzvniCHInnHhuk3X0aHIffTQEF
Xfv4/KPqXRUcu+ce0PfuN8Pb3wbohPgrP8NfftCCCEiQkKYiJjt5Y08+uE+Yq
1mosymY85nJ9rIS4nB5fDw3lPb8Xr8RFmM6+KSo0lI6z9b47y09yVSbCksyV
0S+sY3/914svGiP0BsaujbF0IIERESwsRJKa1z8Hlpwym1sb/KWMrhj1+dxZK
xwYcTAda9uh+Xw8sVP5xNWk7/WzS0urWaj0o+4pqJ12AxWULbeHMlvHc/
5C+C6VeHtm0hhBARJSFM9JrX5+fap9ZzoKolJO0l2rsPLo5GN599UMrowo
x+GcAAXtv/Gl7t5eIxF4e+8XfuM56KvOA3oCKzyKwQQojwkBAmeu3lzWUcq
GrhZxdPZlb+qS3yGWOJIi+1+82fP32nGJ/bx5wLwryafBc+v48md9MJr9NoX
t77MjMzZoZ+xfv9H8Dn/4TT7ob0saFtWwghRMRJCBO94vb6+d1/9zJleCJXz
8lDhbF3prnOybaPyhg3fxjJWbFhu09Xnxz+hAfWPkBJU0mPv+aGKTeEtgh
PK7z5PUgZBYu+G9q2hRBC9AsSwkSvvLjhEKV1rfz0S5PDGsAANr5VhNaa
2V8oCOt9jmpwNfDwpod5ae9L5Mbn8v3C7xNlOvFfEXuUnQtGXhDaYlb9G
moPwLWvgsUW2raFEEL0CxLCRI85PT4e+WAfswuSOe04E+lP1pb3D3Xaf
qiuwsGkxdlhewryb9v+xqv7X217XemopMXTwtcnf51bp92KLSqM4aepAl74Cr
i7mVdXsxemXgEjl4avBiGEEBElIUz02DNri6lscvHIVTPC0gtWvK2GlgYXO
WONeWZpuXHMPj98c8FWla2iurWa2VmzARifMp5rJ157ans79lT1HijbGNj
HMciyE7lz4Iwfh78OIYQQESMhTASlteaVLWV8Wlzfduz1reUsHpPG3JHhW
6sqOTOWc2+eErb2uxqZOJKHlz7cZ/dr4/MYH5fdKyvgCyHEECUhTByjqLq
Fe176nLUHakiwRbUtpGq3mLn73PERrm4QOLIDXr8TLDGQ3LdPfQohhO
g/JISJTv688gAPvbsbq9nEzy+ewpWzczGZBsf6VB6/h4/LPqbV1wpATWsNid
GJfVvEvvfhn1+D6Di4/j+QOLxv7y+EEKLfkBAm2uyuaOJnb+1k6bh0HrxkKl
mJg+epvO3V27n/4/vZU7en0/HTc0/vuyI2PAlv3QUZE+HqFyWACSHEECch
TLRxuI2NtK9bUDCgA1izuxmnz9gA3O1zU9lUxm1vXU2aLY1fnfYrxiSNabt2
eFwfBCG/z9h6aO2jMOYcuOxJiO6fq/8LIYToOxLCxKBysOEF796MT7/Azn/BzM8tdOCCGEhDAxy

NQ6a/FpH1+Z8BVGJIyg4I2niVXRXDL//r4vpvEwPH8lVGyFc38B827p+xqE
EEL0WxLCRMQ4Gt28+dhnuJ1Gr1VzrZO03ND0VC3NXcq8YfMosr4ekvZ6r
WIbPHc5tNbDlc/DuHMjU4cQQoh+yxTpAsTQ1VDVSmVxEzEJVtJy4yiYlsb
UZTkha9+1fz+tmzdjnz4tZG32yN734KlzQPvh6/+RACaEECIo6QkTferQ9hr2
b6kCwNHgBmDWefnkTQz9ArBVjzyKyWYj9cYbQ952t9b/Gf7zP5A5Ca7+Jy
Rk9929hRBCDCgSwkSf2vLfEsp21WGLswCQmG4nMT0m5Pcx7S2m6e23Sb
vtVqKSk0Pe/jH8Pnj3h7DuDzD2XLj0SWMtMCGEEKIbEsJEWGm/pnhbDW
6nsfxFS72LjIJ4Lv2fwrDe1/bUy5gSE0m5/vqw3gcAVzO89A3Y/RbMvcV4AtJk
Dv99hRBCDGgSwkRYVZc28+YftnY6VjA1Laz3HFOqsaz7jNTvfhdzfJiXpGg
8bEzAP7INzvsVzL0pvPcTQggxaEgIE2Hl8/oBWHbNeLJHJwEQlxJ9yu02uB
pweo0FWbXXi66tM47XHuSqlX78yQmkfPUrp3yf46r4HJ67ApwNcNULMPa
c8N5PCCHEoCIhTPSJuKRokjJDM/frYMNBvvTql/BrI+B98zUfS7ZrAIYF/rj
vuBBTTOjnmrXZ8y7863qIToCvvw1ZU8J3LyGEEIOShDAx4NQ56/BrP1+d
8FVGJY0i/62/4sqqp+bSxQBE2WNZcO33wlfAJ0/A23dD5mRjD0h5AlIIIcRJ
kBA2xD2xcj/PfnIIAJfHH+Fquvfk50/y773/BsDldQGwJGcJ87PnUxzzJjo7iel
3PBjeIvw+eOde+ORxGHseXPoXeQJSCCHESZMQNsSt2VdDQ6uHpWPTA
bBbo5iZ2wdLOvTSusPraHY3s2D4AgBiomKYnDa57wpwNcO/b4A9b8O82+
Dsn8oTkEIIIU6JhDBBfmosv71yRsjaK95Ww/5PKwFwNLlD1m5+Qj4PLg5z
b1cwjeXGBPwj2+ALD8Gcb/R9DUIIIQYdCWEi5D77oISy3XXEJFgBSM6K
Cdmk/D53eKsRwFyNxgr4Y86KdEVCCCEGCQlhQ4zWmvd2HKEpsHjqkUY
n0ZYQD6tpTXpePJfdHd4FWcNu99vwr6+DPRm+/g5k9eHwpxBCiEFPQtgQ
s/NwEzc9s6nTsTPGZ4Skba/bx4Y3D1K6u54R08K7IGvYrXsc3rkHsqYaT0DG
Z0W6IiGEEIOMhLAhxu0znoD8xaVTmD/SCEqZib1fPNXv89Nc72p7XV/hY
OULe2ioamXCgmEsuHR0aAruhreuDu1wAKCdLlAqNA37vEb4Wv8EjL8AL
nkCrLGhaVsIIYToQELYEJURbyMv9eTnaX3wzC52r6vodCwhzcaF355O7vi
UUy3vuFwHDnDg/AtA67ZjMfPmhaDhJmP4ce+7MP+bcNZP5AlIIYQQYS
MhTJyU1kY3CWk2Cr9QAIDZYmLEtHQs1vCHFl9dHWhNyvXXEz3a6HGzz
5h+ao02lBkT8Ct3wPkPw+wbQlCpEEII0T0JYeKk2eOtTFjQ+9XiPX4PGys28
t9D/+WTw5/g9p14GYvq1mompk7sdCxu8SJiFyzo9f2PUb7FCGDuFvjKP2H0
mafephBCCHECEsJEn3B6nawtX8v7h95nRckKGt2N2KPszB02lwRrQo/aWJ
a7LPSF7XrLWIQ1JhVueAcyJ4X+HkIIIUQQEsJEWDk8Dh5Y+wAflnxIq7e
VeGs8S3OWcmb+mSzIXoAtytajdpo++JCmP7xDOe/jrakJTXGbnobX74Ts6X
DVixCfGZp2hRBCiB6QECbCak/dHt46+BZn5p3Jl8d+mdlZs7GYLb1up+4fz+
DYsJGoDGM5jegxY7AWFJx8Yc1V8PY9MGIJXPUCWAfoYrJCCCEGLAlh
ok9cNvaytn0fT5Zt8mQKnn8uNAWt/g14W+H8X0sAE0IIERGmSBcgRJ9rKI
MNf4FpV0PamEhXI4QQYoiSnrBBqrrZRYvLe8zxiobWCFTTM9rtxlNREfSc
v9UZuhut/BVoP5z2P6FrUwghhOglCWGD0KEaB6c99GHHtUyPYY3qm07Q
WmctAOYeLHpafu99NL7xRrfnQ7Iga+1B2PwMzLoekvNPvT0hhBDiJEkIG
4TqHG60hhsXjWBi9rHLP8RYzcwdEd5V7cHYLPzJz58kKzaLmRkzT3i9r7Y
GS34e6bfdFvS8berUUy9qxYNgssCS7596W0IIIcQpkBA2SPxhxT7+/nExAJ7

A/pALR6exLESbc5+Mj0o/Ymv1VpbPX47VbD3mvLeqiuKvXY+/uRkAX20ttil
TSLzoovAUVLkLtr4IC+6QDbmFEEJEXI9DmFIqVmvdEs5ixMnbcLAWl9f
H2RONcGG3mpmZnxyxevzazyObHyEvPo8LR18Y9Bp3SSnu/fuJXbSIqCxjja
74pUvDV9SKn4M1DhZ+O3z3EEIIIXrohCFMKbUA+AsQB+QppaYBN2utg
48ZiYjJTYnhF5eFYMguCI/bx8f/2oer1ZjsX13aTHxq9wutvlv0Lnvq9vDg4gex
mI6/LljK175G3KKFIa33GIc/gx2vwml3Q2xqeO8lhBBC9EBPesJ+A5wDvAag
tf5MKbUkrFWJfqemtJltK8uISbRisZqxRJvJnRB8XpnX7+WxLY8xOmk05xa
c28eVduODn4ItCebfHulKhBBCCKCHw5Fa6xKlVMdDvvCUI/q706+dQP6k
4/ckvXHgDYoai/jt0t/26KnIsDv0Cex9F85cDrbESFcjhBBCAD0LYSWBIUmtl
LIC3wJ2hrcsMZC9U/QO+Qn5nJ53eqRLAa3hg/+F2AyYc1OkqxFCCCHa9G
SxqFuA24HhQCkwHejRfDCl1LlKqd1KqX1KqR90c81SpdQWpdR2pdRHPS1
c9F9+7ScxOpEuvaeRcWAFFK0ylqSwxka6GiGEEKJNT3rCxmmtv9LxgFJqI
bDmeF+klDIDjwFnYYS3DUqp17TurZTXa31IKRW59RTE4NFS
A7vfgp2vw4EPITEXZn0t0lUJIYQQnfQkhD0CdF1pM9ixruYY+7TWBwCUUi
i8AFwE7OlxzNfCS1voQgNa6sN9uVxSnaxXYZxFv4
HqOhIVyiEEEJ00m0IU0rNBxYA6Uqp73V4KQEw9+Dcw4HDHZ6XA3O6HDMWsCilVgFx
wFqXUCiAe+J3W+u9BarkJuAkgLy+vB7cWkeLxeahqrSImKqbvbvr+A7D6Ye
PztHGw6Lsw8ULImgr9YUhUCCGECOJ4PWFWjLXBojAC0lGNwGU9aDvY
b7+uuxlGAbOAMwA7sFYptU5rvafTF2n9BPAEQGFh4XF2RBSR5PK5+M6
H32Fv3V4eWPBA3924bBOkjIKrnof0cX13XyGEEOIUdBvCtNYfAR8ppf6mt
S4+ibZLgdwOr3OA8iDXXVAdW4m9RSq0EpgF7EBF3YEsVezceAcDZ7Dnut
Q6Pg299+C3WH17P/fPv55Ixl/RFie1i0yWACSGEGFB6MifMoZT6FFAJaFs
iXWt9ovUHNgBjlFIjgOPAlZC5YB29Cjyql
+Ol58SRnHTvM2Oxu5vb/3s6Wqi38dNFPuXBU8G2KhBBCCNGuJyHsWeB
F4AKM5SquXo6bBGqv9SmEfjVBkCWY2B4IZwFYzgp1Hp5tOSKQh
B/6itd52ct+KCIfUnDi+/IPCbs83uBq49f1b2Vmzk18u+SXnFJzTh9UJIYQQA
1dPQliq1vpJpdSdHYYUtv8QtLX2KaVuBzpEu2CQrxtIHHfRq5ng8UdMFUSEq
z8NLH2ZZ3rJIlySEEEIMGD0JYUcnAx1WSp2PMa8Ma8rJ3wlib7k9fhorHIGPe
dxdd6dqqKlghZPCwBOr5P7Vt9HaXMpj5z+CAuHh3kDbiGEEGKQ6UkI+6l
SKhH4Hsb6YAnAXT0pPWcMeT3Q9XoE4o2Nxbp6UQkBPFoVk0Hb3zsHqD3V4IA
yOys2WGt8bj8fmg+YkzMF0IIIQaQE4YwrfUbgU8bgGXQvmN+O6Ao9SYVQs2
u3Mu2hk0PMZ+cbqJA3uBgBunHIj41KMpxAnWu6T+AAAIABJREFUpkw
kLyHC67bteAWqdsHCOyNbhxBCCNFLx1us1QxcjrHn49ta621KqQuAH+KM5v
9Jrx/9m78/gYr/2B458nk0RWIZEoUkvsIhKE2NdaYym3uLRFf12ubrpcii62uq
q9qlp6q7pQXVBVpap2am9iCWKngpAgsu+ZzPn9MUwT2YbMTILv+/XKS+a
Z85zznUmYr/Oc53xtE6KwNidXB+oHVzWrbcuqLelQo4OVIzKTPhu2TAcff2
g2rKyjEUIIIe5IcTNhX2Hc5ysM+ETTtAtAW2CiUuoXWnRLEOLIaE8zB
iBdiZU8RBCCGEKD+KS8KCgWZKKYOmaU5AHFBPKRVrm9CEKEZmM
vzxPtTuCPV7lHU0QgghxB2zM0QmmwclrOhCgFM3gjNSZVJHaGR8BDGKQpGkqTyG
UhhRBC3JOKmwlrpGnaGnakZvfa0Ddm481QCmlmlk9OmFxyqA4FRZLdoYegJ

QbmTg4mXOTbDmSEgt754P/IKjRsqyjEUIIIe5KcZ++jW0WhbCZuOhUtiw+
ke9Y3eb32PYO22dBbjZ0e6esIxFCCCHuWnEFvO+maLewoeiEdFIyjTNaqVl
6s84x5CoAej7tz8ONPQFwdLmHZsLizsDBJdDqafCqW9bRCCGEEHftHvr0
FXlFxaXRZfb2fMda1/Y0+3wHJx1Obg5mt8/UG3fV12llfBfilmng4Ayd3ijbOI
QQQohSkiTsHpWUYawm9WLXugTU8ADAv7qH1cY7dO0QGhqNPcvwKvW
lMDjxK3R5E9zusUuoQgghxG3MSsI0TXMGaiqlTlk5HnGHWtaqTLdGxW+
0GvbrXxzZFg38fTlSu8M7CsNiw2jk2YhKTpXuLtDSUgo2TQZXH2j7YtnEII
QQQlhQiUmYpmn9gdmAI1BH07QgYLpSaoC1gxOWcTUqBTudZtoV376Cj
mr1zJ81y9BnEHEtghGNRlgrxJKd+h0u7oXQOVDBreziEEIIISzEnJmwqUB
rYDuAUipC07TaVotIWIW7pxMdhzW4q3MjrkWQY8ghpFqIhaMyU64eNk8
Fr3rQYmTZxCCEEEJYmDlJmF4plXSnl6/E/ePPmD+x1+xpWbWM9uSK+B
7iTsHQb0Fn/s0EQgghRHlmThIWqWnaCECnaVp9YCywx7phifIkLDaMAO
8AXBxcbD94djpsfw98W0Pj/rYfXwghhLCS4soW3fIy4A9kAT8AScCr1gxKlB8
p2Skcu3GM1g+1tnjfmZGRAOgqFbPYf9//ICUGekyX8kRCCCHuK+bMhDV
USr0FvGXtYETxMrJzOXc9FcD0p7Xtj92PQRksvh7MkJ5O3MKFuLRujZN/
k8Ibpd2A3R9Dw75Qq61FxxdCCCHKmjlJ2BxN06oBK4BlSqljVo5JFGHSz0f
4JeJKvmNODtbdPDUsNowKugoEegdatN+EH34gNy4O708+Lnq7jJ2zITsVu
k+x6NhCCCFEeVBiEqaU6qpp2kPAUGChpmkVgeVKqRlWj07kk5ypp5aXC2
+HGmeOXBx1hNTxsuqY+2L20dynOY46R4v1mZuSwo0vvsS1U0dcWrQovF
FCFIR9AUGPg08ji40thBBClBdmbdaqlIoFPtE0bRvwBjAZkCSsDFR0cqBH
k+I3Z7WUuIw4ziaeJdQv1KL9xi/+htykJLxfeaXoRltngJ09dH3TomMLIYQQ5
UWJC/M1TWusadpUTdMigfkY74z0tXpkosztj90PQMhDllsPpk9IIH7xYtx79s
TZ37/wRlci4OgKaPM8VKxusbGFEEKI8sScmbBFwFKgp1LqSkmNxf3jz9g/c
XNwo7GX5epFxn+9CEN6Ot5jXy660eYp4OwJHeQmXCGEEPcvc9aEtbFFI
KL8iUmLoXbF2tjbWa7Oe+axYzg1bUqFevUKb3B2C/y1HXq9B07WK0guhB
BClLUiP101TftRKTVU07SjgMr7FKCUUs2sHp0oc9aolKDpirij02AwzoJVqg
mtnrb4uEIIIUR5UtwUx61V0/1sEYgQRP4EsUdh8JdgX6GsoxFCCCGsqsiF+
UqpmJvfvqCUupD3C3jBNuGJB4Y+C7a+Cw81g6b/KOtohBBCCKszp2xRj0K
O9bF0IOIBF/4lJF6EHtPAzpxfSyGEEOLeVtyasOcxznj5aZp2JM9T7sBuawc
mHiAZibDjv+DXFep2K+tohBBCCJsobk3YD8DvwHvAxDzHU5RS8VaNSjx
Yds+FjATjLJgQQgjxgCguCVNKqShN0168/QlN0zwlERMWkXQZ9n0GAUO
hmmXrUwohhBDlWUkzYf2AAxi3qMi7V4EC/KwYl3hQbJ8JygDd3i7rSIQQ
QgibKjIJU0r1u/lnHduFIx4o105AxA8Q8jxUrlXW0QghhBA2ZU7tyPaaprne/
P4JTdPmaJpW0/qhibKWa8i17gCbp4GjG3QaZ91xhBBCiHLInL0APgPSNU
0LBN4ALgDfWjUqUaYSMxN5c+eb7IvZR82KVsq3L+yB078b60O6eFpnDC
GEEKIcM6cooF4ppTRNGwh8rJT6StO0UdYOTJTOjSupZGcYZ7Ky0nPMO
kcpxe/nf+f98PdJzkrmuWbP8Vyz56wT4KbJ4F7deClSCCGEeACZk4SlaJo2C
XgS6Khpmg5wsG5YojQSYtNYNj0s37EaDSsVe05Magzv7nuXnZd3ElAlgC96
fkGDyg2sE2BGPESHw4B54OhinTGEEEKIcs6cJGwYMAL4P6VU7M31Y+9YNnQcVYP
+1bliiNLIzjTNgrfvXoWqdigB4VXcrtK1BGVh2chkfH/wYhWJCqwkMbzQcnV

0RRbZLTUFiNHg3gsARVhpDCCGEKP9KTMJuJl7fA600TesHhCmlllg/NF
Fa3jXdqdnEq8jnzyWeY8qeKRy+fph21dsxue1karjVsG5QyVdAnwmPzAadOf8
HEEIIIe5PJX4Kapo2FOPM13aMe4XN0zRtvFLqJyvHJqwkJzeHL49+ycKjC
3F1cGVmh5n08+uHpmkln1waWakQfx4quEOD3tYdSwghhCjnzJmKeAtopZ
S6BqBpmjewGZAk7B6Uoc/giXVPcDrhNH3r9OWNVm/g5Vz0bJlF7Z0Pudn
g1QCsnfAJIYQQ5Zw5SZjdrQTsphuYt7WFKIfOJpzldMJpJraeyOONH7fdwK
nXYM88cPM17g0mhBBCPODMScLWa5q2AVh68/EwYJ31QhK28LD7w7Yd
8I8PICcDPP3AvB0zhBBCiPuaOQvzx2uaNhjogHFN2EKl1CqrRybuHzfOwY
FF0HIUXEiGnPSyjkgIIYQoc0UmYZqm1QdmA3WBo8A4pdRlWwUm7g9K
KbK+n4ghzgUq9iU3eSGavdwVKYQQQhT3afg1sATYAfQH5gGDbRGUuH9
k7VrN+f9FAm6wcSwAru3alW1QQgghRDlQXBLmrpT64ub3pzRNO2iLgMR
9RCkMf8wDwHvsizg3bwlAhfr1yzIqIYQQolwoLglz0jStOcZ1YADOeR8rpSQp
E8U7sxFiDwPeOAUE4dq2bVlHJIQQQpQbxSVhMcCcPI9j8zxWQDdrBSXu
A4Zc2DwVKtYAsss6GiGEEKLcKTIJU0p1tWUgwjaSspNsM9DhZXDtODkPv
wr8aJsxhRBCiHuI3Kb2ANl7ZS9v/PEGXk5eNPFqYr2BcjLI3TCDa8frkrjsRx
x8fXFq6m+98YQQQoh7kOx8/4BYeXolL2x+gaquVfkh9AeqOFex2ljp303lr+
W5JEZm4jl6NH5rVmNfubLVxhNCCCHuRTITdo/IzMkl8nIS/tUr3tF5BmXg
44Mf83Xk17Sr3o7ZnWfj7uhupSiB9Hjivv8ZZVeB2suX4hwQYL2xhBBCiHtY
iUmYpmka8Djgp5SarmlaTeAhpVSY1aMTJkv2RnEtJYt5neuafU52bjbj/hjHp
gubGNJgCG+GvIm9nZXz7p0fgj4Xx1p1JQETQgghimHOJ/L/AAPGuyGnAy
nASqCVFeMSeaRk5vDZ9nN0rF+FED8vs8/7+ODH7NRtZlzwOEY2GYkxn7
aixIsQthDc64ODs3XHEkIIIe5x5iRhIUqpFpqmHQJQSiVomuZo5bhEHot2R
5GQnsO4ng3Nah+dfAmAK6lX+OjRj+hes7s1wwOM5YnYPAOwg8p1ICvX6
mMKIYQQ9zJzkrAcTdN0GPcGQ9M0b4wzY8IGEtOz+WLHX/RsUpXAhyuV
2H7PlT38d+enPMLTvBb8Gp1qtrB6jDlXrvBXaCiGjEzAEziIcwvrjyuEEELcy
8xJwj4BVgE+mqb9B3gMeNuqUQmTz3f8RWq2nn+bMQv20+mfmLFvBo9ef
RkHZztCgmyzJivn6lUMGZlU9NPj2HMMODjh2qaNTcYWQggh7lUlJmFKq
e81TTsAdMdYsuhRpdQJq0cmSEjLZvHuKAYEVqfhQ8Xf0bg4cjEfHviQR5z6
432tDi0frU0FFwfbBHolAgCPgQNx+9e/bTOmEEIIcY8rcZ+wm3dDpgO/Am
uAtJvHSqRpWm9N005pmnZW07SJxbRrpWlarqZpj5kb+IPgUkI6GTm59Gt
Wvdh2mfpMFh5dSIcaHeh05TGc3R1o1vVh2wRpMEDYzTrvDfvaZkwhhBDi
PmDO5cjfMK4H0wAnoA5wCih2C/Sb68g+BXoA0UC4pmlrlFLHC2n3PrDhj
qN/QNiVcFPj5oubSclOYbDLk5w+lUiHIfVxqKCzTXDHfoYbpwFv0Mn9GkI
IIYS5zLkcmW9hkaZpLYB/mdF3a+CsUuqvm+ctAwYCx29r9zKy5UWprDqzC
l9XX5J2OeJWWeHfqfiZM4vRZ8PWd8HTD+POJUIIIYQw1x2XLVJKHcS8
hKkGcCnP4+ibx0w0TasBDAIWFNeRpmnPaZq2X9O0/devX7/DiO9vl5IvER
YbxkDHx7n6VzLBfWtj72D9WTB9QgKpi6eQeiKGTI+eVh9PCCGEuN+Ys2P
+63ke2gEtAHMyocIuoqnbHs8FJiilcovbSFQptRBYCBAcHHx7Hw+0VWdX
YafZ8dClhiRWyqZRu2o2GTf2nbdI2bwN8ILtKwGwc3O1ydhCCCHE/cCcNW
F5b8vTY1wjttKM86KBvKvDfYErt7UJBpbdTMCqAH01TdMrpX4xo/8HXrY+

h9Wn19CxWid0SQ64eoBOZ7ma7EopyMkp9DlD9AkcK+ZQbcYMqNIAO2d
nKjRqZLGxhRBCiPtdsUnYzUXzbkqp8XfRdzhQX9O0OsBl4J/AiLwNlFJ18oy
1GFgrCZh5UhMy+XbyHgbnvAnAReKpWufOinuX5Op/ZpLw3XdFPu9c0xO
Xnv+06JhCCCHEg6LIJEzTNHullP7mQvw7dvPclzDe9agDvlZKHdM0bczN5
4tdByaKl5aUjSEHoqoeZkjIAOw0HTXql7yj/p3IvnAB+6pVqTx8eP4nTq6Fm
MO4vPBfi44nhBBCPEiKmwkLw7j+K0LTtDXACiDt1pNKqZ9L6lwptQ5Yd9
uxQpMvpdRoM+IVNyVlJQFQI8id1n3rWm0c+6pVqTImz82w10/D/yZBz2egc
z+rjSuEEELc78xZE+YJ3AC68fd+YQooMQkT1hOXEQdAPY96th14yzRwcI
HOb9h2XCGEEOI+U1wS5nPzzshI/k6+bpE7FG1AGRQhmfZc23WVvSfz78
N15WqG8ZsSNnK1qIv7jJciu74NrlVsOLAQQghx/ykuCdMBbpi31YSwgrSrG
XTKdCD+UBxJdvnvejRgINM+DcdKll2MXySlYNNkcHsI2r5gmzGFEEKI+1
hxSViMUmq6zSIRBShlzHV9Q2syMLQecRlxnLhhrJ1+ITmaz8Lf51PvT20Tz
Mnf4NKf0G8uOMp+YEIIIURpFZeE2fJClyiBQRkYs2kMpxJO5Tvu5uBm/c
Fz9ca1YFUaQPMnrT+eEEII8QAoLgnrbrMohIlSiswcAwDZuX9f9d0YtZFT
Caf4d8t/06KqcdcQZ3tn6lWywcL8iO8g7jQM+x505tzLIYQQQoiSFPmJqpSKt
2Ugwmjar8dZvCcKgKp6jZE4gWbg04hPqVepHk82eRKdnfVrQ5oYcmHbe/B
wCDQKtd24QgghxH1OpjXKmUvx6VStWIGn2tdB3cjCsDGWZE4SlRzF3C5z
bZuAAaRehdRYGPoNFFPfUwghhBB3xnKFBoXFeLtXYEznuvyjpS8Amy6s
x9/Ln241u9k2kNxsYxLWMBRqtrHt2EIIIcR9TpKwe8CNrHhebv4ymq1nouL
Pg8qFR6bYdlwhhBDiASBJWDmWpc8CoJ5HXdpVb2fbwePPQ1I0uFQB74a
2HVsIIYR4AEgSVo5tjNoEQD+/frafBdv6rnENWMXqth1XCCGEeEBIElZO
pWansvrcagDqV65v28EvH4TIlVCpJugcbTu2EEII8YCQJKyc+vbEt6TmpJTc
0NKUgs1TwNkTKte2/fhCCCHEA0KSsPJGKbTcNJYeXk5rrzK4I/HcFji/Azq/
AXayg4kQQghhLfIpW45EXErE5XgKvVMcgcmm43Y6G60HMxhg01SoVAu
C/w++etk24wohhBAPIEnCyoG0LD0fbjzNoj3nGaa3J80xjWz/WPrV7Ye9o47q
DSrZJpCjP8LVo/CPr8C+gm3GFEIIIR5QkoSVsdQsPX0/3snF+HRGtq1F5a
MnSUpI4/+G96NmxZq2CyQnE7bOgGpB4D/YduMKIYQQDyhZE1bGzl9P42
J8OjMHBfB8dy9i02KpVKGSbRMwgPAvIOkS9JgGdvJrIYQQQlibfNqWEz7
uFfj8yOcAeDt723bwjATYMRvqdge/LrYdWwghhHhASRJWTlzPjOaXs79Q3a
0aDjoH2w6+6yPITDLOggkhhBDCJiQJKyfWX16Co86Rmu42vgyZFA37FkC
zYfBQgG3HFkIIIR5gkoSVA3YVYjkUv40RjUbgaOsd6re9Byjo+qZtxxVCCC
EecJKElQOO3hupoHPmqaZP2Xbgq8fh8A/Q+jmoXMu2YwshhBAPOEnCyti
55BM4uB+nS9UheFTwsO3gm6eCozt0/LdtxxVCCCGEJGFl7cdzX2DQu9D5
IRvvzRW1C85sgI6vgYunbccWQgghhGzWWpb2x+7naHwY2Tf64qRztcmYuY
mJXHrxRQwXjoChOhzYCh9sL7RtTnQ0jvXq2SQuIYQQ4kEjSVgZUUox79A
8Kjl6cSmhrc3Gzb54kYwDB3H2ysa+bgBUerjIto61auLWrbvNYhNCCCEeJJ
KElZE9V/Zw8NpBnqr3Ogf2OXN1eww7jiRx43IaTm5W3CcsVw9AlXaVcHt/J
djprDeWEEIIIYokSVgZuDULVsOtBq0culMh6yxJJxLJcEwGoHaAl/UGP7n
W+GfQ45KACSGEEGVIkrAysOXiFo7dOMa77d9Fl2D8EVTv5cuj/ay8/iorB
Q59C9hDjRbWHUsIIYQQxZK7I20ox5DDF0e+YMKOCdT1qEs/v362DWDP

fGOdSABNs+3YQgghhMhHZsJsJDIukil7pnA64TQ9a/VkUsgk7O1s+PanXI
U986BOJ+CE7cYVQgghRKFkJswGrqVf48nfnyQxM5GPu37Mh10+pIpzFds
G8cf7kJsFrZ6x7bhCCCGEKJQkYTYQlxGH3qDnzTZv0q1mtzII4CwcWAwtR
4OHr+3HF0IIIUQBkoTZkF1Zvd1bpoGDM3SeUDbjCyGEEKIAScLud5fC4c
QaaPcyuPmUdTRCCCGEuEmSsPuZUrBpMrh6Q9sXyzoaIYQQQuQhSdj97
PQGuLjHeBmygntZRyOEEEKIPCQJK2vKSv0acmHzVPCsa1yQL4QQQohy
RfYJK0PXL6ZwZt1FAHSuFv5RRPwA10/AkG9AZ8ValEIIIYS4K5KElQF9d
i7hv53n0KZL2DvrWO2SxRtVnS03QHY6bJsJNYKhyUDL9SuEEEIIi5HLkW
Vg909nObjhIo3aPkTwGH9OOxosO8CfCyDlCvSYLuWJhBBCiHJKkrAykJq
QSZWH3ej2ZGMcnC08GZkeD7vmQoPeULu9ZfsWQgghhMVIElZGNGvNU
O2YDdkp0H2KdfoXQgghhEVIEnY/SbgA4V9A4Aio2qSsoxFCCCFEMSQJu
59s+w9odtD1zbKORAghhBAlkCTsfhFzBI78CCFjwKNGWUcjhBBCiBLIFh
VlICMnl8T0bFYdiuZSfIZlOt08BZwrQYfXLNOfEEIIIaxKkjAbSM5OBqCCf
QUAzl5LJS0pm5nLD5vaVHZ1vPsBzm2Dc1uh53+MiZgQQgghyj1JwmzgwN
UD2Gl2BFQJAMBgUDjZ27F9XBcAnB11VK3odHedGwzGIt0eNaH1sxaKW
AghhBDWJkmYDYTHhtPYszHujn8X0dY0qF3FtfSdR66E2CMwaCHcnGkT
QgghRPknC/OtLFOfyZHrR2j1UCvLd67Pgq3ToWoABAyxfP9CCCGEsBqZC
bOyiOsR5BhyrJOE7f8aEi/CEyvBTvJpIYQQ4l4in9xWFh4bjk7T0cKnhWU7z
kyCPz6AOp2hbnfL9i2EEEIIq5OZMCsLjw2niVcT3BzdLNvx7o8hIx56TCtQp
Dv74kXiv1mCIT29wGm58fGWjUMIIYQQd0WSMCtKz0nnaNxRRjYZadmO
k6/A3v9B08egenPTYaXXE//NEq7PmweAzrNyoac71q2LY506lo1JCCGEEHd
EkjArirgegd6gp/VDrS3SX3Z0NOlh4RDxHVzUQe028PMq45OGXBKWLiPz2
DHcunfnocnv4FC1qkXGFUIIIYTlSRJmReGx4dhr9jT3aV5yYzNce/8DUjZtu
vmoIuybk+95nZcXNeZ+hHuvXmi3XaIUQgghRPkiSZgVhcWG4V/FHxcHF4
v0p7KzqeDjhG/HGzB6HTjnv9xoX8ULO6e73PRVCCGEEDZl1bsjNU3rrWna
KU3TzmqaNrGQ5x/XNO3Iza89mqYFWjMeW0rPSedY3DGLXYoEICMBLSc
Zx15jcazfFEffGvm+JAETQggh7h1WS8I0TdMBnwJ9gCbAcE3TmtzW7DzQ
WSnVDHgXWGiteGzt4LWD5Kpcgh8KtkyHSkHcadA5QJvnLdOnEEIIIcqM
NWfCWgNnlVJ/KaWygWXAwLwNlFJ7lFIJNx/uA3ytGI9NhceGY29nufVgn
FwLmYngXg0cLVDuSAghhBBlyppJWA3gUp7H0TePFeVp4PfCntA07TlN0/
Zrmrb/+vXrFgzResJjw2lWpRnO9s6l7yxXD5ungYMruHiWvj8hhBBClDlrJm
GF3Z6nCm2oaV0xJmETCnteKbVQKRWslAr29va2YIjWkZqdyvEbxy13KfLQ
ErhxBrwbUPjbKoQQQoh7jTWTsGjg4TyPfYErtzfSNK0Z8CUwUCl1w4rx2M
yt9WAWWZSfnQbbZ8HDbcDVp/T9CSGEEKJcsGYSFg7U1zStjqZpjsA/gTV5
G2iaVhP4GXhSKXXairHYVHhsOA52DgR6l+5mT6UUyfPHk3g4hUR9d3Ji
YiwUoRBCCCHKmtX2CVNK6TVNewnYAOiAr5VSxzRNG3Pz+QXAZMAL+
N/NzUX1SikLXcMrO2GxYTTzboaTfem2jMiK+JPLC7cBlSB8EQBunTtbIEI
hxP0uJyeH6OhoMjMzyzoUIR4ITk5O+Pr64uDgYPY5Vt2sVSm1Dlh327EFe
b5/BnjGmjHYWnJ2MifjT/KvZv8y+xx9XByGtLQCx7M3zAeg2qSxuPZ8FAB
7Ly/LBCqEuK9FR0fj7u5O7dq1pYKGEFamlOLGjRtER0eR2

8OpBDMpAq4damdU+Ozqacz16GvcBK4JD/UAcqlWzVIhCiAdAZmamJGBC
2IimaXh5eXGnOzhIEmZhYbFhONo50sy7mVntcxOTQCk8R43CyT/PXrZ/Lo
TYI2gDP8SllXkJnRBC5CUJmBC2czd/3yQJs7D9sfsJ8gmigq5CiW2VUqTvD
wfApU0I7l27Gp+4fAAO7oTHx0O3IdYMVwghhBBlxKq1Ix80SVlJnIw/adalS
F1uLpeeeZZrs97HuXlzXFvf3M5CKdg0BVy8oN1YK0cshBDW4+bmVuo+9u/
fz9ixRf9bGBUVxQ8//GB2e4DatWsTEBBAs2bN6Ny5MxcuXCh1nJayYMECl
ixZYpG+YmJi6NevX75jr7zyCjVq1MBgMJiOTZ06ldmzZ+drV7t2beLi4gCIjY
3ln//8J3Xr1qVJkyb07duX06dLt6FBVlYWw4YNo169eoSEhBAVFVVou6VLl
5p+Vr179zbFtHjxYry9vQkKCiIoKIgvv/wSgOvXr9O7d+9SxWZLMhNmQfuv
7kehaPVQK1K2byf517WFtvO5EYC9HjKOH6LqO29TefhwNLub+fDZzRC1
E/p8AE4VbRi9EEKUP8HBwQQHF33T/K0kbMSIEWa1v2Xbtm1UqVKFKV
OmMGPGDL744otSxamUQimFnV3p5jbGjBlTqvPzmjNs8++6zpscFgYN
WqVTz88MPs2LGDLl26lNiHUopBgwYxatQli1bBkBERARXr16lQYMGdx3
bV199ReXKlTl79izLli1jwoQJLF++PF8bvV7P+8wvHjx6lSpQpvvPEG8+fP
Z+rUqQAMGzaM+fPn5zvXMGzaM+fPn5zvH29ubatWqsXv3btq3b3/X8dmKJGEWtD92P046Jw
KqBHB1+auk7dqFQ/XqBdpV8KxHmrMnfr+tzb/g3pBrnAWrXAdaPmXDyIU
Q97Npvx7j+JVki/bZpHpFpvT3v+PzIiIiGDNmDOnp6dStW5evv/6aypUrEx4e
ztNPP42rqysdOnTg999/JzIyku3btzN79mzWrl3LHH3/8wSuvvAIY19/s2LGDiR
MncuLECYKCghg1ahTNmzc3tU9NTeXll1/mEbnH7GlClT+Mc//pEvnrZt2/L
JJ58AxlmUMWPGcPHiRQDmzp1L+/btuX79OoN9Oi9evXc+D
AAVJTU+nTpw9du3Zl7969/PLLL/z444/8+OOPZGVlMWjQIKZNm0ZaWhp
Dhw4lOjqa3Nxc3nnnHYYNG8bEiRNZs2YN9vYKNB+cAAAgAElEQVT29Oz
Zk9mzZZN16lTc3NwYN25cke9Vly5dCAkJYdu2bSQmJvLVV1/RsWPHAu/1y
pUrmTFjhunxtm3baNKcOGDWPp0qVmJWHbtm3DwcEhX3IYFBR0xz/32
61evdqUTD322GO89NJLKKXXyrau6ldimpaXh5eVFcnIy9erVK7HvRx99lO+//
/6eSMLkcqSFKKVIX7+JJ89UJW3FKnIuX8axjS8QQ
L27t2LTqcr9NzZs2fz6aefEhERwc6dOdO3bt2FfYuDX4bBePDGDxIPUz5fm7oiqE9twRIRJ2fXPrsN
8vh2jHo/g7YO5bNCxFCCCsaOXIk77ZIBp06MmxRj1qbV2Uez0+ZQ6ON
L27t2LTqcr9NzZs2fz6aefEhERwc6dO3F2dmbWrFl07NiNiRiIgIXnvttXzt3333X
Tw8PDh69ChHjhyhW7duBfcv349jz5q3ALolVde4bXXiM8PJyVK1fy3HPPATBt2jQ6dOhAZGQkjz/+uDmJutqGIdlk
HZSmTZtGt27dOHjwINu2bWP8+PGkpaXxxRdf0LVrV+Oex8+JCQkAAGDFjBhMmTOD48ePk5OTwmeffWaK
iljIgwIEDz7Nixg/Xr1yMyu3Dl4pnRoxLBkJTAK8KHVo3MnvggzQMmUfyaj3
9u5FsasVYgyEEokVMnz/GB7W/xI2j3mQrJXDR/UgnAHUeHq0O
29nh4eHDjRv6iOQ4ODnz22WcEBARQvXp1jh8/ztNPP216fuLs
nuMS5f+LlUdHBzMzp07zYq5rMlMWAlz7ZrXsyfYrFs36ij+m7jPb4d
HoYJ+fNl+V4tTyR7kp6+mGTsI3R18sf6vFGdrMpS/Iukb4A9FuNvzqpR9c3
kPv8zXkv3Or2xcXh5pr7Vt3Gb3pZH8vkjcsuOXz0DmgBvnJdq2ja+sNS9
0tLLSczs9q/4HHxnAGvBT2zZTW4Prpb
bq7qzdHHdhVUS5+zpX3MeyMfYwVb3fe0tT8LqXyzbZ2NfXJrcsB96RT91ZhexlrA0G
TqUvjVP2P5Ueqr/nvh3QYBM6O5Z7W8mG1FfaKuX0OUm8yS8pY6G
Pbf9PHeEOCl1Kp5erbp4Cx2sNk/bPYrZ3TUZA6oXdjt2/DHP6+5GpbF38NW0RHH91Gih5r
OvfTRYCkFCPGzp1q1dWYDT3/dN/tT

PPvoId3d3QkJC2LhxI6GhoUXe1Xend/vdSeKjCtmW6fbxcnJy+Oyzzzh06BB+
fn68/PLLvPfee7z99tv079+f4cOHU6FCBRYsWMCoUaPYunUrAD4+Ply5Uq
BKYrkkSVgpqOxssqOjATh9eBteQM7Ef9G4r3Ftgn2ev6hXkzNJydQDoDeog
m982EJIjoZH/welXFMghBD3ksI+kAszceJEQkNDWbduHW3atCl0huX2fot
KJLZt24arqyujR49m8uTJzJkzB4PBwN69e3F2djY7PldX13ztJk2axL/+VXCz7
gMHDrBu3TomTZpEz549mTx5MmFhYWzZsoVly5Yxf/58UxJhjlszXDqdDr1
eX+B5Z2fnfNUS1q9fT1JSEgEBAQCkp6fj4uJCaGgoXl5exNxWFi8lJYVKlSr
h7+/PTz/9ZFZMHTt2JCUlpcDx2bNn88gjj+Q75uvry6VLl/D19UWv15OUlIS
np2e+NhEREQDUrVsXgKFDhzJr1iwAvPJsXP7ss88y8yYcIE0+PMzMwCP8Py
Sj7tSyFm6jT+6hvKX31D8Xr3KwBq1gnCwccHBx8ftJulCy7FpxMycwuPzPm
DR+b8QUJ6Tv5/GDISYOeHUO8R8JOyREKI+5OHhweVK1c2zZh8++23dO
7cmcqVK+Pu7s6+ffsA8s1e5XXu3DkCAgKYMGECwcHBnDx5End390I/+AF
69uyZb+F2QkJCvuednZ2ZO3cuS5YsIT4+vkD7W0lAhw4d+PHHHHwHjbNft/
dzSq1cvvv76a1JTUwHjJbdr165x5coVXFxceOKJJJxg3bhwHDx4kNTWVpKQ
k+vbty9y5c01jlfRematBgwb5ZsiWLl3Kl19+SVRUFFFRUZw/f56NGzeSnp5O
p06dWLNmjel9/PnnnnwkMDESn09GtWzeysrLy3bgQHh7OH3/8UWDMnTt3
EhERUeDr9gQgQMYXYzzTcA/PTTT3Tr1q1AwlyjRg2OHz9u2gB106ZN
NG7cGCBf0rhmzTzz/TtGlTs9+rsiQzYaWQm5iIQ/XqeP/7debsn4OLe
2XeaN+hQLukDON192c71iHAtxLX113CMSfP/6x2zoHMJHik4HV9IYS4V6
Wnp+e7ZPj666/66/zzTffmBab+/n5sWiRsS7uV199xTBw8OjQH9
z585l27Zt6HQ6mjRpQp8+fbCzs8Pe3p5ujPbbb/Piiy/PAsSA+wsPbuhHQ2xm9yuVQ1
nw4YNfP755/me79ChA79/++ivDhg3jpZDeokOHDmiaho+Pj2nLnLB03TWLVqFa
+++iqzZs3CycmJ2rVrM3fuXLNjKczTTvMhoTTz/Nk08+Sb169fD09MyXeAcFBRERE
UH16tWZMmUKnTp1wsHBgVq1arF48WLAAbkF48WIAPvnkE9NNDZ6enqbjBjZlDA0N
LVV8tqKZOw1cXgQHB6v9+/+/eXdRgAXHrhRXJiYvD44Qt6fx/KU2en4pTrWq
Cd3qBITM+hopM9jvZ2ZKXr8arhxtA3W0HiJZjXEpoOhkELChlFCCHu3Ik
TJ/LNDpR3qampn3FZs2aRUxMDB9//DHR2WUlZWFTqfD3t6evXv38zzzz
xeYuSqPVq1axYEDB/LdIfkg6NSpE6tXrza4vbDiq4rV+jxXJbtLb8o4vppHWqa
bVGW8m83v/h/X7Eqv1t4Y5p+ZXP1Y7uEOndnppZ+HFn58s7Y0jtVTZwY2Y
BzY+5ShGsV9jdHoc0/rk+Q8fidLx4i7UCcXZYxDtCVGlLjT7IK
6fiSVuNeBcyCEKWRPVxdhdnRoaBQBhD4
D1OyCX59sWObNFCBEB87W4PuGBCVxZF0pcSUcYUDixV0ToX2RPfpg9aMNucbRHp4l+4otNJRp0mr3F4r
5E98X3rAAcOt8vfwNX6ZcRRdsWYdYq0ubU0Fb2SU76vV05fpwUbJJU9sl1kC
fWh5aFbWfiUvW9sWD6vBPLi1Gkbj/6rkAGYmQlwCrn3ChGkYNcEbD3fmTvdNH
IkjJq8p4dMWW7NKqb7jE7t9hLb2RwZHDFJYbZmDfTgCnUGC8aYBRTvWt8m
P25M1Ta3k9J/Jt1o0MdXctHWXcsO9GBDDnb+aqubM7m5fY7Y6a6iTOzw6pH7

L24/MYELp8/wpR/wuw2PzBi0X7eGxpIl4bGXYo9nB0K7+jYKrhyCAb+Dxzuj
Q3lhBBCCGFZslnrXXB0ssfZ3ZEKWjY5WXH4P9wIJ3cnsjVwd7LHy60CXm
4VsNcV8vbqs2HLdPDxh8B/2j54IYSwEZ1OR1BQEP7+/gQGBpp2pb8bkydP
LnaH/AULFrBkyZI77nfDhg2mGodubm40bNiQoKAgRo4ceVdx5jV79mwaN
WpE06ZNCQwMNMXXpUsXLLXV0v79+xk7dixg3ErjkUceISgoiOXLl/PMM
8/kK8N0N25tZnuLXq+nSpUqTJo0KV+72rVrExcXZ3q8ffv2fKWYfv/9d4KDg
2ncuDGNGjVi3LhxpYoFqFevHmPHji20ssH333333+fr46lnZ2daXuR3
r17ExgYiL+/P2PGjCE3NxeA+fPn39GebKVyq0r5vfLVsmVLZHLlZHoxNVrQlr1YbImOI72fe5UlMqKnVqg
w0iFkI8qI4fP17WIShXV1fT91evXlXdu3dXkydPLsOIite5c2cVHh5e4e7/jv
j777DPVs2dPlZSUpJRSKjExUS1evLjYcUpr7969qlOnTnd9/u2qdNQUKWUUkeP
HlV+fn7qxIkTpn4//fTu47zllatWqk9/Yog8Ggevfurdat
Ht/62RgMBjV48GC1dOlSpZRSaWlpKigo6K5iKuzvHbBfFZHTyExYKZy+Hg
vAnF9zGb7QWG6j2IKnmcnwx/tQuyPU72GLEIUQAn6fCItCLfv1+8Q7CsH
Hx4eFCxcyf/58lFLk5uYyfvx4WrVqRbNmzfLt5v7BBx8QEBBAYGAgEycaxx
k9erSphuHEiRNp0qQJzZo1M82oTJ06ldmzWwPGckNt2rShgBo0yFR
mqEuXLkyYMIHWrVvToEGDYgtO165dm+nTp9OhQwcgW1ds+yNHjqg6deqY
Ht/62RgMBjV48GC1dOlSpZRSaWlpKigo6K5iKuzvHbBfFZHTyExYKZy+Hg
vAnF9zGb7QWG6j2IKnmcnwx/tQuyPU72GLEIUQAn6fCItCLfv1+8Q7CsH
Hx4eFCxcyf/58lFLk5uYyfvx4WrVqRbNmzfLt5v7BBx8QEBBAYGAgEycaxx
k9erSphuHEiRNp0qQJzZo1M82oTJ06ldmzZwPGckNt2rShWbNmDBo0yFR
mqEuXLkyYMIHWrVvToEGDYgtO165dm+nTp9OhQwdWrFjBxo0badu2LS
1atGDIkCGm3fIPHDhA586dadmyJb169TKV/w4NDWXcuHG0a7devat27dqp7du3K2ZvQ0M1/3mfFAIA41gHmQeY
gC4zz//PMEBwfj7+/PlClTTMcle40rVqwzap16tQJ+HvG6dq1azzzxBBNERE
QQFBTEuXPn8s24FRX/Ca8zr61bt9KiRQvs7f9eubR06VrWhbCtfZlvLnkQlffLLnbx6R+ZybNsvOqefa5bidu2RaQvs
XHwEbBaJodTo46QvyKudNxzzzxIj4Me0+AOq9MLIcS9zs/PD4PBwLVr11ZmNhAz46mzJKGsfx4UnfxbaLs09YDp7/JKjxZqZJ++a6KmH9p
3YS5xOTk7s2rWLuLg4BBg8ezObNm3F1deX9999znpWaLtc2n4ijTKfzVTW9S0kMJiMhYwuWYdSx08qbKtwCOmidJbIAxaKCmoDjmGnsxZr4ZVkY2+LftQ6ZMPOiq3dOWyTuVxcMU6wJvvDK02vmoPPL5iGnrPNAsfCmBNlk2XrQQPvEe6qBzhXu+1+7Z7Hq+CwoBpzkcx59cXzhPFIwJ+uhxnxMeBBgc1ZhptVSdTFm3i6RtHXibv2hPpg3g3X7O/oL3SejIJzJWLHrSR6YnEL3Y3rKcZTs0M38CowLhM7qeegLpkyFEdZu8Sy4yvrJJ3VSNaz58D9bUbs2HDaerYQ3C
1atGDIkCGm3fIPHDhA586dadmyJb169TKV05k5cyb/+9//9y//qFixImAsQzRBRE
gC4zz//PMEBwfj7+/PlClTTMcle40rVqwzap16tQJ+HvG6dq1azzzxBNERE
QQFBTEuXPn8s24FRX/7a8zr61bt9KiRQvs7f9eubR06VJeeeUVatasaSozVZI
PPviAt956i0aNGtzOaNGgFgb2/PCy+8YNa5RYmJiSE5OZm2bduiaRojR47kl19+Kfac
pUuXMnz4cNPjWjWz8bvV5PdnA26fPbxcWF2rVrExYWVqoYzSFrwkohJScJd2
XHwEbBaJodTo46QvyKudNxzzzxIj4Me0+AOq9MLIcS9zs/PD4PBwLVr11i9e
jUeHh6Eh4eTlZVF+/bteeihhzhx4gQ//fQTf/7zJ559aSX37t6/5vXmzZuOjsz6cEmSNw//ddqvuOoivAYHEdBYqsKgLJ0eTXv0vr0HCIvEcfVaxc2jQi
k6ePImmaSQmJhYZVenHRDt/9L7Lbu/3SOZZrhMWC3+IhePBy4bM+BySiMfPHGs5s+OLRQKPPKhdI/OO/LYL22pIxRF5O1LiMfTGvZ9a4q
3YS5xOTk7s2rWLuLg4Bg8ezObNm3F1deX9999nzpw5TJo0iZdffpnVq1fj7e3
N8uXLeeutt1JmTWbgo7GThn3F8vS/Pzgg7vR3FlP2Jzc2le/fuHDlyBF9f30Jf4/Tp09m
wYQM1atQo8LaavPFnvsSSqrzdXFyQ3MY7XfW8zp3kU8XwvUQ4CH8rrQ3Z1F2+3evZu
WLVuaHmdkZKSqrbvAtfgMTbP0qkv8uO7hb2qWwXxvWyOO9K0PqPY8WLD5kcO3pE6dY
U47uLiwp49e9/Idu3z5cr6SWL6+vly+GfCvbdCDiP1oH58eAjS2fahxSoSKwXmUMyK7a2bS3AcH2
Hxx57zHQ8ODOiYnTt30rp16xLDjwJkohh1TsqMD0R5uVV3DglFvbOB/9BU
KNlye2FEMJS+swq6whM1M11Oxs3buTIkSmWa2kpCTOOnDnD5s2beOaZZ3B1dQUwfb3lVTLn5+dHs2bNGDBgAAMGDMjXnpSUxM2bN+nc2bhiW8Bl
BxMe6x6OmZ/z+2FStWxMnJiWeeYGbQ0NB8645u9ZOiGDBl
iev5W7ciWLVvmK3Bt7r7nwKt4d/jx4Kzs2nbti2i0iMjKSHj2M
VzZyc3OpVq0aSqnir4k8UL0ej0xMTF+cP36cJk2aFPoa27ZtM4xiZQBM4a6OBtvdpsdt2+i7Vg0hP6jNa2haPoA27dvz+jRo
xk6dGiBpjfKSr+21/n7WJiYvKV4Fm7di1du3bFxcWWf/zjH7+73rt89NFH6
HS6Qd+vue/BLV27djW7HNSt3yNzx7uV1N9e2HvDhg1kZmby+OOPs3rtVrp6zLr8kf5zvDq2bm35KgyV
P0sfHh5MnT95B9HHdHkrC7F5sJdi4EcNFWwyGyhts+C3Gzo9o51AxNCiHLq
r7/+QfT4ePjg1KfefPm0atXr3tXLmdv7XVvb2ZSoVly5Y5f/58tm

7danYMFSpUAIw3Dej1+mLburoaawErpejRowdLly7N9/zRo0fx9/dn7969hZ7
7119/4efnV2T/58+fZ/bs2YSHh1O5cmVGjx5NZmZmka9xwYIF/Pnnn/z222+
mItfmKCr+21/n7ZydncnMzDQ9Xrp0Kbt376Z27doA3Lhxg23btvHII4/g5eVF
QkICVapUAYwzlre+9/f358CBAwQGBhYb553MhPn6+hIdHW16HB0dTfXq1
Yvse9myZfkuRebl5OTEgAEDWL16tSkJy8zMxNnZ+rsXyJqwuxQeGw6Apiq
U3DjuDBxcAi2fAq+Sp6eFEOJ+c/36dcaMGcNLL72Epmn06tWLzz77jJycHA
BOnz5NWloaPXv25OuvvyY9PR2gwOXI1NRUkpKS6Nu3L3Pnzi2QiHh4eFC
5cmXTeq9vv/3WNCt2t9q0acPu3bs5e/YsAOnp6ZrQGDRty/fp1UxKWk5P
DsWPHAJg0aRIvvvgiycnJACQnJ7Nw4cJ8/SYnJ+Pq6oqHhwdXr17l999/L/Y
1njt3jpCQEKZPn06VKlW4dOQlSqeIvSePGjU3nJCcns2vXLi5evEhUVBRRUV
F8+umnpsSuS5cufPvt4BxRvC7776ja9euAIwfP56ZM2eaxjQYDMyZM6fAeL
dmwm7/uj0BA6hWrRru7u7s27cPpRRLlixh4MCBhb4Og8HHAihUr+Oc//96RI
DU11bR+T6/Xs27dOtOaNTD+Pt4+a2YNMhN2l8Jiw2iADg3HkhtvmWbcD6z
zBOsHJoQQ5URGRgZBQUHk5ORgb2/8Pk08+yeuvG8u0PfPMM0RFdGiR
QuUUnh7e/PLL7/Qu3dv1IiiCA4OxtHRkb59+/5jJuykpBo/Xgv0PvfDmP9
WvW/jlyaOX9ffUXC9+2E7GXsmu0eYTyFPhtNZNTsY+SxAp/0Ndr0QhZUYa7ta+NYmn0HG6GnzsYYVJFs1mqgpCjvv5TR9/Bll+/BqpUbtlEKmn0IqOM18XOvdL0F/D8iGvWSG3I5F3UV8E+TkD4eP2/+W2kyLmmc67e7P8+P1zL7zo6ewb4AocdFYCvmYHj6tvhfR/Etd8GYhGnyFOB6Xfi7+iShFHhtjEF4CQ6u4Qvy/4EUx7KNT3xdLx9/42aU8Ju4K7ZrCXqk/PaTsGHX2vDNq0Wqd5mH8kKBwsv6CL5ILCWm4wt7XC+pJpiwJtK+xtwvN1QHT+0KJv40ifQ8l4AsaHNuoTWN0lBbp0HaIbIlFNwS0oBVhvnQmHnnTQoOKUDu8DIYi+ja5GHKFkXbCtbXhpwDQ5i3VcEd4HTlj1ZdjfBDjJt1OGWxlXzbHmPQQCQxE9yHICSbHSVlVswdKehNR9Psyb+XfB4Wq0bRtgf4Bmy1G9KoUoqR5nHH0vkFgAAz7hEqHD3yfYcBJgiEjT7lwwU+K5jfNAY5uGOvuDUrJZPV3Ky1Kf78rQdsPfKxJXZoybqo+bGmx0Kc2qvPJi+y1iZvUoxAaHhBKwaZAGGzcDlzx1WIBOt2l0hPgrFu2bkAZt+l3fzSUkoKoUlvkyevv/1DC+1l62f2QQOHGaDpgTyCk5CCgYVbdfMgQ4MfO2mdthtQ2ByQaIhTuzTNRwGMsMTCJkRt2j3OlvWh1oWUMTJkqU7WbDnrM6yBg7EKR5ut0DEXTcxTeVJfv3e+2/Y7IGf8Mnvgqrl8LP+8aBRxVRYz0VgHWFddVi/pHvSg1tiEQEQZNzmMNdVEPenUh9p4e6RLyeCpUQZW5jMMZEADsDuKgJH0EU7sDBTt2uuk4unSnbSbs4jVvnYSyZ/ruwMgdOkwk5tSFLCRsOcYUs/SpZHZ8vW37B6gSeW83qTMxrcPb4Vu8SUvVV42wNTm2XfOi5+oV7FNvWMvPXOTmoo6Ddr1Ot+AN2eJ+ANlKGEXNjlBCgEdr05y7lLvkVOaqczqHjDaDj0s+zGv/TcFNUHMxZkvLIGzdvTAU0EOPJkfZQ6qJKw3fNXB0ea3hmCbEh2VaUBuIQqZ+4z7l8cKRA4XpA4D/X1G9XRNaxUrPJ4MBo9LHpRVV/eFWpLf7wAKOZYdZy8VcbZa7HBWT31T9/V6EPBZVgRYYaaQeRB5BHt9YKrX+ujQN3xKTUVmgaxnQrCMDtJiRhC8+TNz0wydCmJzKDM9gaBzMnqj8XIc1R2XrFtXjeNXX5NNFV+Sn7+LrYY+R1xUe7L7+rSTDdGtN21Y+I70ykc8Gq2P+QzkANCsi5lGdrqKMRpsCzVD72IGI6ChdEv0AybhPYhZGS+4d2z4d1BhF0Xjm9YsHbTJyMOqZjy66t6Oic+ieMpMm6jjHGy5UUGtKw8Rvy8mfj5Q2xmtUffOM5+nALuNl62dRZzTZWKUvE8AJlMy7HPXm/Z4Pb6kdK+pZ1qPfaVOL6b8hASCYvtXPjN9Jyxeg39oDXtoCX26eJa1n0zhbCfqwHXKQLfjrtcVmTrIwJr6sBjvvhJqRcYo7mFfeAYOzscrkGptJBetNcLCTKWzLi5oLQRcDVQLgN4QVhNN3yuR9tHRUx7mQeTwkO9GdGmLc+WY9BBv3gPkJCa0tDJJWGO+dIcwpd0r5MLR8thF9n3uDSJP4x8dWM7Bqwu76V9jOXTL1rhfUFHt5KwtmjZ0c0vd9EXqQKV5BM8GSWiHyJV8SfyAK2z9NpncNnePSDhn/CIM/ez+KK2Xl5cR3mjThxiYkRSNyHbGiLWsAOrPCO0jJMeZYd2Opt3dB3+C+wXx2M+gaW+mA0uDxYw3CRfgBIaQ6PHI9Vts0Nsiaq+XRKNyQjIvPcP3lYfhnUPzn8ai5mCFa/Zu8Ar6i1/bYRjGgBS80vKU/TC/TuuxMAYX+bGTg5XSTNpGAkyJQgDtBQpEnHB4bIoEw5ZW63w1DbdQwQ2OcEX0t9q0W1z4BcfgjbMG2oMEw/QvYwS8MqC9R0a/dMHVoPyCYY/nMhkDRKx6PAOZHKRTxDZchr+WUkv+xkaGfjhNl2wcw9PzaR0Sz8MqqV0wjRVdSCL+9TAGdsMCDu0r80KxBjZxnySdDFpvAWUZn9vHqVC5Y6LjOhKsOkjtBy9OBP6HzhaXrMCZPVwtgcQKdEYPNm5DNpKupcJUSavWD5FLyMxLnHkULkQmQp3Bn0LkhTmqDhCKM7Wzzm8VKTu+Fg+GovXv+KTdsQlPxSRTWhjbvWbxrDgn34e6eSphSi/s8PZJNDOdk4Oos+5ixsLDNe5Qvv8UexTk2Y+zvAZ3+Sai/0xq8YDNxkADfUnePnjZfcZcVqH6tRuOoqD6l3AbEbJ9YHmWWvwxQeVQtFY6bJfp+y/PR8RzkzTNhSUgDSUc55DVmCqh/BtLFABRI1Ur0wzWoxEf/VdkqTQqRSJOqPG6Wa7UiU0cmBEWhWb3n6fGbSIsuzt1amETd2R/5CQGGT36u+u4Xf1gmWFXVUHX7cLvqLIJ4Tay4OVB43EjgjgNSb6G1D8ZSwyHkOGyqsDCm0VH6FiBR8iGCY1DQ5qRdcTVUQ6xYJSRGn47huAKmaOm8Lek67yk4ecEVO9xDb0CC6cDSCCANJGiQFWrD9w3DoO0oEnzQpWL7EV4Wfe6KnDe0I6uVJApEhAEjx6ZA8Gfv0cM/WjMVp1Zf7vSmB6TnJDGCoMmDB6VKSpW4cfuD4phGz/KhcN7+UmIoDQ9T8W5n+HpD8+bPsvt/3kW87tN1c7EFqnN3NqjjU7Lzu2hVa/6Zjw60DAQIEW4RsU1e4M9xL2aOIEm6SFRwPPWpQbqGDS2Qs1BgIUMvpmVQUBx6VTTcBV1HMP6KrPh5DVfXPJNPoDMX3ddIXGR9iDVrXMNkZFNBi5UaSxXpGZaHTzHqEBOTwH5YOe2eVMSnNFRwMsJ0dWMGLQQxGWSAIhtpjdkI7Vq8Gr7BPhtPYKaq5oDDL9d7bQlPPjXd0x+FGHFXS/ybEVO8n+q5RNdzthv57Tl1wmuYYI0iH6dcbuFhGOqOfxkLiRmkBuW3q1HgCd4ZqGAwnm90aRHp6cqDA49ktoAW+CnYyjBdbHkG9xXZl0vsO9QYPEzqR7CRDCNYKVbGkAYFlFpDxJ4rcBE2DoNGpRcvfQ4TsSgOcZrc4nGAOwNhb6HkGxxWDUkyLTEV8ZqPTrSUb3W2DAtZHi0Gu+cdMsWOeoUrWDdSTsJMGvJ9uyE2OoaSRlrY+PvYCtEd3vCUVLsJ1k6A7WFbEBfO/CYYlHgj/UN1VJBBUXGGRDPOtI+iZk1brJYIZ9PsF5Qk3XBuTAlxuROvgFYe9UeW/QW/4lEpm81zB3NQlCU+AVh8wzzz5pE+jNU0ZXdZZZMqN3MP6LzlzZmnJt0/6qScsnoYPDj6Qv3hxj9aZCX4rqFHTYtPdGfc3/bZt9vdTqJp3h1v9fF1HCs9Y2Ib9Hbwv3uQEBRUCfgwQEvzyEdNWl0sZl7sfKHfDNP0Y8xsQpZm75ZaOv+2pzZiDLJGFfNXzEhK9Ld+Sl22a2gYQ1zoxZXS8mZ3pWPbG2Q/xY/ssdvU9Vqo+FwVBFxE/hvUdUoLWDOMdzsbZs4SZtQOdGJ1+32ptnb3Yr0ZVhknoXlYlQ7Rc0jV0cQAL3j+9g3G/ZpYfNrJD6pkWzjt/W3pYWt/1b3bwb8ax4ZV3FBj3p7sqyzAZ9A+Z5QvC/vrZKZOY9F93K6xSqj3NgePa1IeLN6vc5ZNiazBrbZa2gSNajRgC5bJLWKb/FbeGCNsMFRMYLNvT5hAeyqAOb4AfbDrbrTaNZT4k3XIvy85EpmWYsD5SE0IuX9+P87GHpbB+ShcDSZF8wn3f/LP5bGgpf7SD8pBsI/iJHqWTpXvW7WNqcFDYk02Q1yMrRl+jiYfSO4QDJGg7zm3Kq5Up+2NjYMxr2FLpiNmjHK2uBk3/8D5WMnU0HqwHo5fQn8S/m/AuM3dRVv/k4JF37OcQW6L1DXHv8PhBXgcjMdkPvZYvdDJGJTVqfSXVM6dRbU1cWjCuHC27MHwQE36NvqKQOVdxxLkbhWRKS0DWoSU+ufIPvB4rIYbVAwQyBKhvSevd6mJ6TsuZNWeMGjtNbfLtjrJWxfaULeHuRxJcqO7gEZ0OIgNSe3Ok+QDTUGK6KgJQmdWdjnIMbD0XcRE7FDDGqBfEUW42LbGnxMHRnqMnUy1W4RNqzkhBgI4FXNYa5tyNGEkDZeIQ5jcAzz8Dh4/EAnumX5+9x1cdR7rGm8u8JFCE0Nc3/CqZH0fMfY2SAb8Rjke5CR9PeLtqbhUMQmP0PdM5xbWuYqW3Mfh6ZBQTfkXTVG/MI2X7BwYqPm+/IKHvTG4i0z4nDhp2++K+yk0wX0bJhvXFmjrBGPq/EzmJD1EJeo8l9zcA7NTRVeyiLWPNJ+YcRVy7mT/A/g7+tIiK9c0XNtqe4hcS9aJ+rd7aSQEr9Fhmn2+ZrXzTp0b2IOmTmPYgnN7XAj3gO+t4qfk6PGfvXCNdZzqb5DlJoASO/cCm+KS4g+hFaUiIIBVVcPaThJxGpp7o4ybDKYZZWKmyZwOq/uFz1EGjDkyUCCS9Cq5Pb3i+Zg8O1LSVfGlOhiEK7hJ+NQ4Jg4HA0kVvz0G9XRF6RhSnAv7EDdhWnZtaBwE0ZfmQOuBwM+TQOVWGVoAyBaJnUHvy8K0YASghPgiWA/KVfbuAYJk3Fk8LjM1eRBR2Jjz6Gy+NfZi5KJ0Vd5dbQ9cY/gRXwBU9WwtFl4CZ+B6/NJmAQ1eA1fw9rvdmlFtK7AuUEXzOo3jTlMPr0KZ8/u+7M5Nm3bumuPmH3uIwQXiT3x+DIVVmhUTVMO3CoVfmU5mTJ4Y5CP13ifR60Jl0MbAKdEmCq3Y1tcWCKlSbcpRQvAN5H+6A3YmzB8RA4YqwQYZH0L6HUsvhW17QkOrjbs7dKjHOrBAwIKt4z1Jv6X8QLjJhuRwLNb25TRYB9gT7M3TA6ZOh0UvZnOvlAl6NeRkaiEDYhqzdTw3AiSXwS8bHmk1Um0s+7XWA1T+VYR3i3rJJR7V4Hkv0ZLjOLOB0kyOBNjcXBOW6OvB24aNDjBZc4oJKXMKoD5wLbqLVeLzHJaCx7xHIaNaNCfQMZ5b4zZ5JiX1yPjmVqLD0Jh0ycbqydxiuqkT6ydYp+b9WXwmpr5DwN1uNqWKHhX6XsCkyNBjcXEDgJDPdfmJkAz5cDQOOfRPHNWk3YzLQphBVI0D6AUDhm0JyRNEsYpIK89akdqv7J0y8xLb+MiIAC3PhNYPhRYoYfmHXB/QJaYPAkd3MljeTNmvz0U8zdwyhxo/pZV7BYrJZH7kiMnaP0+8tIcQjmh3RYgcNoIUBWmNwTt6sZgcFmuE9/sO4lTXUxk0l6Z+xdQ6OFNqFyI1j1XunYz/g1gMOWnI2DoMvUBEq97wvqCvqkaSVJK24n2i5K+ymzp2XIHJP57JFCdeeH81Z3ahfIXyfxtqR9S7Zkw3pNvXzG4qkkJaBbIUkf+JMCT0WuvyaTyCfoPj5bVFGLNwtSy9gKWqAbH0g3ezRqtD1fJMnsDPsQjoT/ZxKF6CeQWuHHlFhr/x6Bq60NpYNEKHEHyWFuIw9BZ8IFxnnGI8WrEwvpNlZvhUzG0pdk7zkebzOV5JmSE5S2FkYVmNuKjjOm6A1lhxWcuWHjI6ZZhDLaX7K0fG5gazZm8UJGCGHG9pLf/jU1u+mJ2/qVr1H20DfZV7oOIQGMPEDSl+JLyFl0ZKRSXLWT/eW/L5gX6D2VA5DPvd5oTyLJwAOIPbhCpAGr/y1N48dtnU7KGfI1KBwXwH9T+cvgKtzI9VukqVDlSKwP9ZXbA94Wn7k0LB4dD2o6GmPpk1j5s/fgMVPK5jNt9rkYbCYMEVi7qq8K2n1UgIUrFjDvCDTmpfFGrTa61nDtvjgLxmaY9ZjGjHrIqHtzn5yJPMkPdIp8lLjZMQZwVDMsFVMNBGSFUqMr0HpeJdhLBvTqGqgYaaZApJL6DmwRS32b0h3AIuNcnLbn+XMajGk8xkxRNJPeTSSCBkgQq6DYBTkDkhqkpjmlAF1FNjxjSrUn5PaIqB7Gcvn7SfOjlh1yAEEL2CJbQRvfDJ3aWJl09BG/E/NsHNQdqMEy2UUYwrtE4+Ygp3QTnPF5lfRQA5bjvzGTz+xmOk0bW7DSH3WYMc2gtjIhAyLQl3a0SkwwmCDjVKPkSt8SFJv6ccQCzzsKKF8y6tYOdjlg0WK6wN7z1CtAeBoFw/zPIJBGxIX4BjDLR7r0D26wHmLWjhbG0DI+aPhJeUKw4uuDN0sLQDuT4ge6LTHn6WuE9p2KbqUKQkm6Nmvqx6ebUMb/Ot/qzm6KMjOxLWENRCUDbVRndt9GaQJtx3CHSYZNGR36EJuIhj2X8+XRi3/cNKzzx+n8MHWpJYkgUhYTD9cnnHNZ6q7+hojbeaUFgNa8sLIizFqTBKhLdhBO+o+VAAO7eUrUDt6T3DwyakRj/lUkcaTXW3rl8/lATLWIWHTpMzZ1CySHm0M2JjUGqgw6fF5eMKqEg2YmsmBdgr3bIRkInvoZRt+Xb4+9Z1iMcX4Ja+GVDGqzxL4tHpBQcCyxSsLtjHPWexO1d+d27TbQRGaTVzOqkiy2Vq4E+yUNd8Ujj6HjSJkjyAUBzeNsamvtLV2Di5KRD6Hc0cNwZdYbJOF9KkkPEUCrGmHxk7sgvEObnujmDwZiA9vC8HJ8DkLFsTpCrHi6jjN4R6yzjvPKemX8lBbQ1Cfq0pUxNAh6X/XzN6yoUDV38e6bkr7sEEE0AlU9N8fFoKTB3odpKOB7Atxk5Ldn8f7q9ZWq/VE1KqA9x5aKFvtUMX+HtwWVH0yHM0HtBqfDF5fXzOc47BOX0n+cbMRWWbTNp0TyNgE2EOVAe7WrWOXHoAWmW/DtQqoaKYyDM8hAwPxWoC/pzm+YsSfpDF4pmntIuKpzSoS/xOADPa2+HGi72+EV+O8o+BOLrKRB5pSCxcwgSUvvX/6Eac/cnW86gW7R2GzOdVPoEStq9F+EfA6XNkk8Iu3hAvv8f+5jEp3hkRXdLJgkS+J5rjI7Py/K/iQLAGE7GmDDw0iEIrrkvb7azCkRDfYMo2M2FCMaKOvP6rPNkIsVukJoMWm3UK7/o6f0zl3Bc5bYCBS3qP3JlHNyp6oRj/WEwyv/LZy5/IfoDcv0DAPqpbCXWGCCSZAQ5c+HvMbNIaPkdVzhw/z+GPJfijbDxaJBSdE5RmQWbGbgj6tNtw8wC1dsgX/K3ZLcXEKR2PtGPgfh9RGPdxL6yyRCPEBxDVcw7MaqZvN5nK8VGKJnKq7zCN6/dXMxN6sLhpkbClV7FsK3WPu0XdCJ1aaBGd+OFvN0qzDb8yRLCHNSuU42VqXNqPIZmiClHkX7JvHQRaL1vgxzcrx3Z7+3BDjH0Yc9GUI6vhfQ3cXZrxkG/5bfvA6q68J2vXOlqZkJT5EwE4+HRsNUO6Xl+xJUd6YdmD2Fs6RAOaZLrLMImSWuJlQ9ISIqjBO8o7GKVwYdH8Vd8ePJKxQR9XXgEsaTyhNMsC6Bd6fRpE/mgeI4nVfQWX9jSFTNS8k2qnhdKHnO4NtPlq0jdz7vf4P7SgqL0f6ZGMjbAPI7dg6Y6XQ+c3n1+A2F7cPdESAR9oATrKzUPZ5yrgDMHAX6Jm8yfdGKvpQDTb/nx3j+zzI0vD3lHymhlaVvJ2C6Re7rvAcgFXIHJkFHJVy6HRaKVdLF2X/3qf1Cvd7VGYXgzI2D+wb6jRpGPYHSj98SG9KgQN7TBoR6P86Rz5cdGHkvCkVXA6lEdBDNBv/Ma3N5i0EAe0TmzhNbR8tY9zSO1K3jNHOOzUOPn7n9vBi7sO4L3FH7xT0R2zDtvD/ewY3uDs6JfC9c6ESl9o7lvWxa1i8CT+6zz9S47NY2cIEbn9+WZXjY8nn9ReKJQYXzW0FxJmnVIMvvnh+XbMwaEbzCpm2NuDHvYpYAZiMkmamStsTDqmEYzN5JWX5xsbJg03ZFZ32sw3V7B5yN2f4aKKQLCQ0hExnFMmj9GcnkYT3oEKzUTPMRo6EfCPYA+eYAYGd6vPOgt1GRfyVkgTyZ5qEgpOx6eeQ6mUD1CeHtnkakB7I8ZXBqF/ko/sGJiKmv6wyZi5o0AzK5ax96HbBRCVKPwbTNoLGeTaRiZaaHyuHa8RuhZ3wlxdg7ltyw1hMnpgEJuoRb0KFPgqx2AlTw2sHHeqSPnv/rOXJO5n4ObfX1YMHLGCZs3dmSm/4y2Lm5C6+cwTWGYNhVYeULgp6NqeCtuSBvn4JpwiBldvSNvEydq6z27U2sGFYtoU5zvR2u/73kKojUgA8DuCpvW1G4NXzFpO8SXYnJWVRcR9yFdOR8MNB24zUr+cX0IWpPTz6W7IqSekTm4rMjClO/kF/iCb09rAmWnkuJmBhzsYjvSN3v1Gft9ZtN0CYzQ+7k+GyWNwxdcNVPqWF1FzDDdyGqOI4+d2PrdvW3oMxkXZoRBvuWE7eBr2j1Mgd8nAO1bq1GCLRgRMwMHD9Rlk/N4WP2wmIyFwr8srGMp7uwLI38K6K33ZSzw+d9cGfJxLc1vE5C4yJNMGTaX2sBtvO+K2RSehtnvjzVZJLmaM3eb6j3+fd9TmhjFJZVBSsyfqBiVYgfEN7BFzqThD70lVCxy3N+b3OSsBt/jhhAWuVsIqIHn6Jvkhcsg/I/v9xU6y5vUQ+ypnpYC5Wo8Yqf4hPc+jDKpV5tkpwp9LlRyW3ljkj4AN5bRM4lsTPdKmfjE92U2WINydqvAk7PKbHdbH9wA3dTdcRyFtGWhBaJqhXnOcu73nmnSmO+rjsT8XqvB5/AEaDzb3z6nqaAg4gGj1Smq0TfLtakOpzp9dQ+ZLYHaRYOK8EzJKdeRLr5P1qlJuBbEJtQr8m7J7hNe4Vkwx3wvvCiBP/wFbJwemiHLvhyQ/LbQJeHRU7IMcqUV3RW5/qj0wHj8VQe4qvDODTOu5uKDIq1P7BS3xhQyWh8e2D3zwk19WqMeCbXH9c2R6s8L0nSwbKBpG0LY76AwtUMAeRrGvSjn5WBfZbbdMmeFJY7ZBqfm/fbx4fxd+8MUBx/HqbGv2cXM6dfQ+WH6xxjkN5gTeKvT64tAJfafWQRfZS7vF3Kzqq99/N6+DTeh7D6Plwph1pcOO88q2sNG7tUThXxoXkibPZ3M8jOEcs9X6bkDsJ3wd3CiMlqV81sWVW7r3o2mC8qa7Qsij8iAbFQvIb3xnAuL+qVD23OUWSeuuAWlZ9l4DYOpP0ve8I7JGR46SBEuXmsqnTXvVt9l4egM/qzOH7mjUHamCIdPJOuGFpgyMq6CzP4D4RLpMaJeGqTNoB3xqq+AtY6CIRm1AJnszTjXKVrjW4pF6z2sERDqiOyOqUCRfXDhJtRJvo97ggZbfYcxC2bYDhGCcC6PqFhRPgFz6oKU5B8cgDyk9Smpy9q3fz0vD54L0sAcEeAyTYPt7DsMGaeA8X/mRj6zOUvMm+5tOExpxZYx1VMMVE20m+EvGGgS5A9bFuv4uPFe7PuITnH0btLjPEAJr6k9vaUAETNvXN1F8ToPbjzGmsyzRXnSVcQQ3UmAmlLp0DK6WIvQGhuUUmp5eZgtnBJPO4hhYDbYEqrEmnrOJvGvYIV9VYaxHZQjWyNA2g2Et9qcFhx8CtuNvDg2udQiu8mRx7l+TXKUNrpzOvTbkfECIy5WcjmSIyXGqzqSN3v5FDc+u06h7M/O7jzqaQ6lLGCKq9nc7j33f5anm8rMPMt6Tz9bZRvv83X1T5N4OxS6W5v8XjHV2u8vKcWUtfiM8efS9zHHhVQDf3RVIn1jpBhMIKm+m4yAcMbPdQbSxnKq1cd1dwlgwm5drjTlt1jCYcPx1PYk1EDLcfnH3ScnyY8Fu+CHWa0KYOW5enLEKTXxpLwXkZ9g1sgP8WTD8UIXLUZTrHzLI9/1sEOWdm9hGjGMjMDEY1X99Hc5ADVDvmHRa3fbW6G9rLMh1x2i1GAbW8v89V8C5NIU3Rj+b5xvVQ0gDpZz08uSRShrr4QFhQrPLLa0WJdOoF0sx9T1hZNA3cv/zkDrdWd/K+p7Ko0+W0cSmCH2LXdrgO/Gw4n4hZSdXIOqJtF8dRaV49fHVzeJujCrhphFbQ1ldLjL2Qn5RGFDrBgE1xIOKs3xWbGvAqbDmaQ9a4FoxKU94izOMn0Y0bVLvpxLyr5c/69J98rJ3yUb5ojmLm4dHYTpYZPwPGX2QWW6nD93KuXBIijXQHTKOv/R5vDFhwjKYkGQU8FZQrqJ1+/D4dFgqBlvc+cxXj1lzoB3NpBEM8zSdLLJEFlD4NEM4SAjN+2uwTXyHUaJe5v6RcHqh4zWsxJYIJNF8cNfbkSnhdz+PGuEl9m6EORGv2p3+MGqB7Wb/GFXWXKTiB2f7PxB0pL1jAbKYbx6Ar1MDN1t7KX3T2A0cP0n2H2ebYjOMG7KALLM+K9TIO8Nsg48Mp6VFYbxOO4C35dVhc2eZfLXgfHsBT6DZnbjaXNLoBxVAcdA06pUvPWuKZtOGkUgPOhtaV8ilQ9T2nBIAC4vWKZRBRsdnqZN/KgO9LsT9ABOGOf2NpBROUf9jV3KgsjwotyHbSqNXTVTWUqtK1Br4P6gqD8XQE2Iz9yvvnBZVwIrwmWx6BM0nOPPa9rRAHXCzdBxH9c9e5fSpEMPG5HbsLQVwkzMwqbD7A0ZKp+Fy6bkgftNn/oG9iH9zCbLwSfZh0POxWFZq8tmyH72X9T7Y6OsxQw/I2XExRjajGomVnDh9lXMkMrTFq0EXbiaCp3SbLqFiu4wqjU8JkadQKyrSJDvbS/3kmnR8fxLs/BI6cYnkn8i7RWGScqpx8pGc6xuJTFRGuSqRxUHtgMgpgXYYZmrgZ4QL7A2CRovBHGwTr99JyQZqPdOemffwNXhwobRiK3U6TWqO8P2D8jFJNt+UqBjmUADL6D7i/4XhvvHsqcwRGj6N3sP2N9YCCz3KpzRrn3WyZo3XP1wbWLoagzjKP7NXSq9mpIqeEAWNKShvg9WxRrAJzmsX6EZPWcHkt2bXTXgOhq5ke88Xnxxy0c3F9VRYhd+Dk8qDTmpMVGJsCJe7bHwxSp5uR5q/6NZ5q7kjW4Ig3iPqBEeRbHYSvBJcyIXxu3k2uDnLNeJsYWAZDT+bzgA4ahNzn76crQl2Thfh2ua96WmfqyPYm7vLvZQm3ov0oc3Ndu0LbzprIHhm+7xKu0q29RT4RZn9ZtAn3ujrLv3JcgJyZI+akVnupZmGqsz0c5jFrIK8Hi56xIe7x9OMRdxqBWCKgM4VIJhEUBQkASGCZA/2xZk/I/pUFS4VVpamsFSVO4z03dUm2XwEP1uvbEqmZn4bnPm4rmjs+Kkg/aM/cm0vnB8JQbQvmK0Tv/mDnDQs5Zr/LB3kIfN37ZJQnCo8l9wy2vn/s6FX3FtdN96NXhlDI8M/sYqIkZ/SYb5dvCmLbtTdJvTVXPsn/H0+q72K84xh8UqVtMVC9T2AyxBxJa4/9+PRDUqlnHvKGCJ4+xKkPs8zEgPA2VNZrz3+VYDvtz9Wsp+ZaxXOsq3rPpSaJsJPALaEQe9GWf30/wSPV2qSYgt+vC2MqIx2kx07eNxLo8V0cRPp0kSt8QGP2wgNpD0w6WeeNtpcS/CFwqowYwK/nmzyyfzMeQ1dvNs2pZs1sr5l8Llt1JqgFQ95YqzKmLDAC0qlI1Hg1rCbQlXPOTD/sWPHt42bj9ClJ1JwPYJ8L0Dz2rrkmPQ+ZaA3Xas5+rtqWPd2ryW2eXxtSwIGf1s5OxNNcrXvbFpsSC9r9sMqSd5K21bIRW/E6jGrYIb0bzTBjxFsuvB2M5cWX9R5MqwZIWtPvfmk8ol/IJl77phvBN7vsWo+kY63bDWAm1Dw3SXK4Hc7W7xsr5tIlY9AwnfAxpp/ksxbn6z92/1VIbxD9jqUj2LeQ0Zx4kiqRhCr3B/ZIqWePTsE4yoMg9d/tf5wvpzHwAXm2kJvmI0gXV5yO9MzT1G+kzRyfgbFlGsbW4FnVdyHF4NwjWG3TWKYgAa6ZjInZHnS9eBR/6K+gHAx5Lqb9fGWLCb7oedTakKo4V90p/rmnRUsCUf3G9FeoHhv2TtJYqDzzySQWc1lRKJSjaUJS3O/HANJh8LhutHLtDglBnZMSpX0WhHaBpNhgBWNg4n4grnjZkeSvY11aPlK9hTcxUnmgV5/ZxDUnRZb9aE4+bhL4grNsoSRUORCBTcRwXzz6+73aX2CU87VnJEHy5hPgPIMHIMfwtTPwnyrFeP9aLFD8hVZnM9zcyTBEEcRaEZVErRKgxNzDx+7FSyo6Gxa8F0T0HVaoVAd9eopsLVD4IzbhybkjmRsnB4dMixXyp0Jeh6gfPJ1FnuUGchVHY1C1BTmbuCQLuFvUuNbpm3g/wS8EQ/cepRvPYLqoxuo8qhtCgXRuV4gyJ1kVUvGxKFSLYh3EY8DWJSULqLgvqOVSlIq9DdMbt6UvGsgCtsVDXOvSIBbCv34ygAW9Kt01fvlqsm18PphQrH6szb5t5kH51aRiPL/ZJuQQjnT+sgJUSQ9I2TaHwcvu+3CkRTTmqTZ0O3q6Z3HsewlG+sUxrYhlQbGmRvC+Wz7cOqV1a0Shu6VM0AnrnC6GdCwi3qOCXIK2o55LoM+QHHxQ0uX0R/OZ/GZBLy9WRUvDMkS1CmbWpm6Rz82rRkbbijkrtd4qUmTZsfEJ8qIKjK/dcpw2bu6rkQ0mCWnQgGlj7YpMaQOqcmNkOuTl3vKLqb4Sb07FUm5EfSrY6xpw5BTVXMLJN+p0eCKmPbV91Z9D6w3dwQwyUVhFAZGfEzzhv4JfoWu9wgVFQ8JDbL4Xq2dT0mnHUX2ptCHHwJwb9yX3nhQd2Wa1/GO4ztpmOjLqqIb93nLKY0wBEG6iIOl8Ne2t8MRpOQqb5ZOTpjNkoeVXQBAcuF0qpYeOeA2+Xs1nQX9DeCsqpzH+dzX3OdUdvQ97JiZu8yj9zpEHDvJqWhm2/zhg3NRWAx4W17eDQ5eNl4CAQfkNYWq1X4CsmIY9GH0Gt/RfgEV+p+Dd9WmQCoYzlr7Z8dZK2P0B3uxG6FbbwTKkkmFnb+5cD63u3c5cEcPjiJiHtQlbt7mWGfl+gMrNgNGZmtZRZ/7kKSpTNqm8v+q6X8R3+Ln8xmGx2VeL8K5UA17g7a5yZaWbtn2RbDr4zIJEzDyvjQW0Pbdtt/JX0lc3hR4KzSsxV8nVtBajnu0bGnMx1VXhEH6I3GJm1z0q5KvqLr3gROWtYpUz9M2qqkzd9UjNn9N1bBzv3Krn9z2vPHrS2GABKl0h9Ky7A8/S2ll0k/IX5EBKUWtg6kVvqbIGb+wbL4oKdFsCbQQyvXOzqbcbSYkTHqFp7bRMIXvVtWCUrshKgdfdsMF9FOgoWQBnjzRFuy0jYnyjVHJ3K5NL2uGS0KKJEqhIAjwYqDxy66gW6NspqjR6Ad2+mlOgLV/gI5qhWDwiLXmIcWjClYuVF9OYlZMhwB7PP5MSWQ3T0TVjf9U7wF6hUP0vUpQyqRNzD7+SWnsmCmB7fcu2QPmRzWWfQCTqXOu3lkpumiiDX4ul1sX2c9eE3R2NjmJZfj3ogljwR5WOYkcpSX/0vXLDnQJoAgsIB8IsjGAkA8kCW9pMV/hUSRZuN9rPyfgIlcBWoNKDIzAvWVJ9kUDYjpTJPv99Yn3vUKrBLswpKkeHkK7KRhN8wcKPPPgK2WqmI3kBkwo1+S19tRVVGasW6IOqNjWfcwxu6GQX2UphsWPxmKtxyh/xG0Cr/XipBXjbxeMh/X0pTqEP3ykXHhRdMnszGQE3e6VGNnuJ7PsRCKVTu6fPTiV1uLJVwVqwwZPpJTAmiOXnn3J4HJR1ZVfpIyGGdSxLhPGaz7jyGhP94j9obdgBVgLVa06jdnRR2AYMTGcCSnGDCPoICPbyghjOTEpChmR6DsM2XjS4uP/D8BtCOaZzAjDE8BXBfhsjhmpFGdP1yUoiCOKJPd5CNZbDm+SCuwfvyJhlw7f3wMUpDFt0EFXDVkUzZBm2hkNVE2h2dTQZBWqTx5XoOEPgYbQ1EJ5fFJ7uuDC7HCRUybGDhJCMp5kE18SstHJRsJJYORSp3ZLCRn8s6bCqTw70uT3ooEQ4D5wwkYyiqp+/DWBqzRAqyBBZTZaLhRtl3pW/8Rhbf8iGfDuo2jwxNSrjf0NPEbgU3dOC5O5MUAjX0BQR4CkXRy4nDmE1Nrwv/tkjKLHY2YZK5UKlVRbuMkn+1tQ72J9oeVwBvFXAe9d7aPxYVqzhl1Qxmkb9HqrhsJ/Q13fjSNRyLbeh9E80qXhm6fWrBiuBuAoq8lVvwsgQwb3kZ+qrFdAHlrjdjM7RHlbcDRx+9ndFO+2g5TTIxv7DB0mW3mC4gYyRnqydZw/fRsE5wrw3ZyWWNEiNVzyqUtTHJuVzxcKUnmFxPnrDWixQcB5JsU5Ngp+dTq5PE1kr6mKVhyoESeKi0X2z5l/ODnQr81EUKKqHOFsRAKpnqtD0tE3GzwGZmNR4Q2fN9pnLJmUq5RfPs5x5xmhFEy9sAQ6yUt1MtHjFlh2/UGA5wM1q3nE6W3Ug84Sx7j1ByB/aYcevIUb8b1ep4Wh5aY7jYQbVsEbYNbozw9IPz95uhTR6rsS76aEMXcPKL6O+6QDmbTAs3Xm07y2ZS3vO6Ua67bD5dA+3Lzp6OZ6xXzXEiNGUJOy7KBDVTnG2xa4B5kHtJYRnlpXhp8amwaoKEDWUuqwmodLMz0pQ80fz0LU5tiUC8HadyE57B1f3Xg3n72tnGEgbZCHgt+ZGxJ6ANnwmyqK+IgMfdcB9oYHYFh14YVGqrS5G/7H3ikTjT3CKzCXeGoCq8pNjjd5yxN3Zm4Hhcds3BdEXPhVmcCaiBmTx8VLWPJa2zHg/PqtdCLxW9DgkmCT7vtV/t5WB8u26EotuEw73zgE3XjlsQ5jzUqbJLNO1LZ4xvRRKVuApENOuGQ4ih7Y5i8pz9UdsXtMJIM/rhxeENwSsRuOHHOG5kpmrAWnzUW1LhN5VrnZ9i7F8MjTf0jQ9jkmeSNjTlz2d9AnEL3Q1UM3nyZ9AYFwF4dzpFSbrA8Bu5AB0PqMkzW1s3Hw11KsCaIM1+sI5nf6ePLwq5dyBfd1g8SgNnuG5Tw2aH0Lk+C5EAKhdK1BOkwC8a9hbZgEZx4J6T49JUvpHG5vVpdAoeRHRhO5IN5nvbEb8tpaHzLp4ptjmA9cCbTmeBfLt3tBgGZlKF5hiUWcWvhSfDuHJvEWRjwv5LPrxnCHL0fphsBRBVYa1x09iXhbQqdnKFhqd9B9rqBSAZaRJ4fIaiwjaLtrrGbw6xZ4Y16p3Py8gdZm4cRsEhyFx7XCAqSoBNjV43NmvYIQhD4Ec1nJBX8fzTZDpubV+mKshpdNhxXoa8dN3ij5sZdqVQN6cphiz5X9S/6p4DNMHvc7EZdlAGEAvuaVMNPTe4TFxeRgeazV3FvVhAgtjkIrKV7tu91N8dlFr0iPJ3N4sJN4g8j+DWXBhuH1+QhM2o0IV1GmAl4D5EwFvTgFt4Ulh6XNCRkhOIaofuX8aeAHdIzkKOp5XtTPNIbMQNuAobh9RFiulWjG1rtkVA6jTHrJR1ffpYY1yVLM1bKsxdJCv1C7cCchtSuWwmf0xaJFZTBcV7XaIaQJ8fiEHvFxTLe7KcbPV0vwlMGYzBCVFiObmRvcx1v6Uc8MgcHYNpMsHM3gqNvY9EQSKNtqhITQfyGyRmHOOLuzNE7u+3MSzlP9Bw2hYF/5pZZB3UuUu0i6PtHt23oyDQdVMcH3XaPl7xMyBwlf1vEZSGeJ8j/kxKmOKp0l8UDYlCXOqY8YN5o5ejpLJrBE79lbM1g0O6qbE8bKDgFRtJKjfiIB6VcbD2XQ1fC7rlrLlgO2EprugjnHmALcbz6hCX9hCY5R78vQSo4ycUTwm6cG/2OB3YnEsWQLsTLO3OJn8rR1t/f6s+UCKmjgDNNDNz5nSy8ECxOUHGyFB3sT+AyZkmA/OA1ZAo5mxb3LANR1K0T9FGZ4RNQQ2b4aPzuhqtqLjyLGdR7yqdN2GDKpz4IU6E5ah8tSTN6Oum6KFMFH7PjxaF9nhCPMFmvyqDtBnLlpP3Cvz/aMc0t4A5iqK3K25q12QjbVnPO3Uq0M2Vi8bFQ2rQo1oAUoiiFBj8Ej1x6Tqgb1pO9X7jZLm7X0MwuO6lsE8V3lY4vkJO2A7SQNCrbj3rGaFF0C4JKtRx9mQBWwOqNqdmvgo2pzs0vdO7tpFK9JuMuIMuc4cJvWCYn4XmK4+PXDjjoUVC9HLDc8S9lfmHWRsEhBZX7dZiP0qA0Dd+nlvZf2Vqsf9FCl9QCh9C8aAJqlGD+H7PNbjD5C98w/+VmY/Xp8JiYJmOZx7o0bmk/P1R/zbSDvzcbz2jg+K+Tci6BTkfOhA8EwnqESS3L9w5D3s7CfeqHH8HpHTgj0IjY1zTW2iROSq+e0qHRRLWh9IHRG5NNqd0mAavo45sfB1vGvCvHdvJdvuKfnG4XwEUOIWhC/PzH3nmLXKv8pAjrdIw8TUoiKHE+vO1gz3aOBUlmEGn4JQcsYzk7ehFxOHh2OMJwJZl+7AzIzfXv8rLPlT3zdeSVb0o+vG46eU/N1JeA/QBHbAuR8zSTZoStzF13G6/3pMQL6AHx1HTRjtUdTMW3C2kdQ/X9kR8jtzVv2MobK86hVs27KqTSlpH61zm7+aEE0IBMZqdyDRJrOy2ujUwIZqmHX1PAQSdJT7H+DWHoJNL1Pvx+V6CaT4xGmQsbuEGnPhfoGOE0b5tOuPz+mgIJm2M0U8fkSg20RJSNJ3nyozLxlq/Zm9MWUsJ3LrNqk/RhDpfW9/ZyVsc1jHs6s00XT70nrhPWdW4vwzBw77UMI4HKZqhpQzc8yScEA5EEGiOUd6rSKmSyu3ABYvwoHvSPIT8iDvRo9ZM1Km/Ie0VN0o6WLm7L/2ZxUnZ1SlfBDLzN9zjHA3wXFfKrMv7VDhIexJ5sp3ldxgGKlYzStqdb6JqDq3R8XW9uZcOjg/VF+zJPAsmaA+a2VfDP20bp4zMUKdEWHjX6ugJNgw3HoJOGwsvo3uKh0R6gcX1v3NsDLh5HDAy43rtuqdFlHTpvRKPn7YF0hd2dwtPgGKNnYD+fXh/3ixdiNPL7iiCEtbgnBgdo9hVhDq17mkpbLTrAhYqnrb+fOlVfxEDUZZ1Yh4kjWcUyu0/4RnjrDCCBLiw3gn9zHLM5iiNe2kGAw9+rVHEHaRS3yrHtnexCA+jSAEPtQw1zKxvkvnv12pDtjbcvBMHXC5nSnUJ6g3sb/UoJcE7Hh6P+ESiQjs4vVqcNSjKxwy5cNnHWsSIsgzlT73IfE4eJw94dMtbA3Oxgj9D/xUfRdE6PZ3OpwCJ+YFSnWxp3ZMNkNe8XUuYG2ZcgJuiRCgD9VR8hnOdU6Wo0dmx6d0ADtcH81bLKdhjzb6WEPwxFCgU78EAxCPvRCS4qKWBF6cS0mgmuOXOr5HDT0eKQ7MKmXlHgVDFOsMkTMg0/o9Y6MLn9L3CnrGRf9SRqgwumO6Q09b5KlHBt3JGVh0NcqiFWVZrvRC5zXGgWnlUOibnDsMuXf6p51qSI3Tl1QpW9lVQWk8LG0ZnY6Qt6CbRqc2y6cPwBR3mrvzKFUpm2QH/NLUvIrnahx59yFPU01FF1DmzMRjS8Snuv5mpDthZOdbt7yKwh1YVmxsmfCTeqDJAJlmYjt5OzUm8dZ02FRN4JPn6zmhfXa/X1TNDQG+sbnvbs0vN94fBzBhIHjgwFPBGEPc+TOCrVyH9LHkeGyR9/9TbzK7+D2UchHeu+Xfz2j0fuPkwEyCsqbA7UXJmcZKX8qNcNN2ZSVfZKYdvkqUiXdnCmn+Aw1qOlUySo6PI+2XkbUO/fOimB/lUIUWlBhGUnxBIWZzZYVMB1ZZVYvpc8RHrS58ZzZP0VrZ4l7gG8Y4rIGPVQfwIFnhnnc0+o8f6dSoeNE1Js3AkOTNt/SNOpUYyO4FKl63ySQ/pJ70/hO6ZmEL9bL2l4bwxBrrm/XzSAsfm/uJ/ztdSxEDBrhhNsYa+bNS2G+Y4PK9jDKuDrFtgyGTUhRR5HxyvflfpdUePVnyFyqxpfKrEe/bGp9sJV9+Da7ELh7+KHUkSGfXGUl2rE9kUPUS0ZnrUTbz/lPu6xmPG+JqYkUaDzxcFIbxBk3bTs3pwrRwf2O7jdvjHh3+WUu/L+qo5ao6pUv0uY3tIClOVIGI3iMdGY6T1PCZz1iUUGQ0sh75YjKyPiAnsaDvfPOvn1LVAiPUZuucj8QT0aJ3hQo0z0e7/fVz9UzN1XE6l8M+6Tx7avPnV6jKvSJhW5B6lClcO2O9zoM3a0aDs2zh9c5GXEuH6M2Ahv46G7xmbBHGbIxHQVqlSq/zWwB4dHkRuJ9ka/eXsnhGxYN+MPfc2ep1sF/ZNH5L6qq9DiO/1jpqOsqhhmm1ApEmMw3wMlyFvF0oSkV2TU4UlJhcJ6WhN+kLqKhmRR3M3mD2oIV97mGAIXiYEIcPvumnHT43JCuU1eQtLtfMF+7hVlDQp8/5YUIFamjG0XT16pZ1HcWn7jAm2e7ZzRWxLTlkbQuXv4TXfhjRtn/jXpnOUqK0rsBamwMhOBg8qN3aNs4lT9jZBWDv0NR2K3nMHJ46jyHJv0y8dgA85RCbwFWjc89q6ySpNuVJHlS2fR70OO0VC9eo8FYd1NDTuKFqyo5cQ2PmCKbSaN4CqQPspoxw6M/eHQ/wECEjqxmdMAiAAm9rzB0UfxVzThi8nYmTn0N96ZWfJGE2Pv7NfoSlbtnJNTZQumrXNpG2YC9q8bixpWxfDwPpYn1CALHJb1/sutWEx7HzEV4yfB0I7d9IoVv8QRb+Y6GlH9kxvQLTNQ8fBJDMDHMdUtxZkxxJs+bJ4c8ZhjhCNqlEu27ZUz3M4bVKJVkhBsR8Wyb3ij7QTsLI/YIH26jcGRsqxpTI8OXbsJjnBV1JgNsNFvPm/Y9lxyqvXtP1nnDVnZJcUZD5hsSwydt6hBNfuHTSHbtDOX9L3lhTsddXxqy3dOZQYHf/PLnOWZ0N6z+tGlzfh0QfIhuNY/E0y/xSzL1G7/SSHc9u+t6+sWwh9PDEazxWrZR2xRW1ZoBoXFDbjDNKS+wJfzVBz6glmt5xhKI3n1ST/xVJjb1Sx2Zu5MUFg+TF0rrHIzvWIXFMU9gtNQ7XuOvgkfgwozLBaaX6VYvMqkOLF1J14SjOLJgWGqNHN0sTr9PdHHBXlixDbEnT8xsp2fqHs3zXr3KZHN6RZNhT80j7j6yjBCNW0zp79vG2tzgFlnBq0RwvqBSjxtyJ2dbv7Vs1ixyNpvA9SN9d1uJ7FmF0xDw8p7yBybuIZnOzIY4WhCY6u1YJkl06QfbZ6SQO2WzZbW7CWf0xMdZm5e+QEd6fv7YJpAq0jc1cJOLEjdC3pvJ5nxr5w+jyX+FTSP9l15gLVydvuKZPsYc1+sLKIDnBoCTo0qH4GRfw/aRMkP3V82sr3cUAU4S1mTUL4f9gbpSnUIzYX9aq1J1Ovu0JJxySa14JcJ+nXEWOUS9B6mWwEjkC6F2uFj6oKbdugu6d3tJNDMZz4etXmVNzFp2mVQfp0ZOoODazhjrJpXUH2nVjqY2HHt8QlHmaD9ZB5Q9Sbh7nm3yHhKj0+QiKEZB0m07RnA4SeZNmNz2YFWxxS3EyXZHdqw0ChRe+Nn9j2KAv/wC/LxRk99NJ6eMi8r6EwXTXNALJyMmRgHmtOc/LRWJmbjHpd6+x5R2ZwhXFGGVb3L+RnNS0aQAAAA==

Qnn/+ec6ePcuJEyd4/vnnuX79utmxFWbWrFl0796dM2fO0L17d2bNKlhD9fL
ly3zyySfs37+fyMhIcnNzWbZsGQBNmzbl559/plOnTvnOCQgIIDo6mosXL5Y
qPpCF+XfkasZl7BySqOdeRP2rhCgI+wKCHgefRoW3EUIIG3s/7H1Oxlu2G
HEjz0ZMaG3+BtRt27blyJEjAISFhfHqq6+SkZGBs7MzixYtomHDhixevJg1a
9aQnp7OuXPnTBuFAixatIj33nuPatWq0aBBA9PO7RcuXOD//u//uH79Ot7e3
ixatIiaNWsyevRonJ2dOXnyJBcuXGDRokV888037N27l5CQEBYvXlxkrMX1
6enpyaFDh2jRogUvvPACL774ItevX8fFxYUvvviCRo0asWLFCqZNm4ZOp8
PDw4PNmzczefJkMjIy2LVrF5MmTeK3337jrbfeMpXKsbe354UXXigQyxdff
MHChQvJzs6mXr16fPvtt7i4uBQYY8eOHRw7doynnnqK7OxsDAYDK1eupH
79+ri5uZGamsqAAQNIS0sjJCSESZMmce6LEEdzc3Bg3bhznzp0r9LXc/po//P
BDU2wpKSkcOXIkX03IlStX0r9/f6pWrcqyZcuYNGlSib8bP/zwA23btqV//7/X
UN8qeVQaq1evZvv27YCxkHuXLl14//33C7TT6/Os/e/fn2XLlvHGG2+UKkaZCbsDxxIOAlC3qCRs6wyws4eub9owKiGEKN9yc3
PZsmULAwYMAKBRo0bs2LGDQ4cOMX36dN588+N588+9/MyMiIli+fDlHjx5l+fLl
XLp0iZiYGKZMmcLu3bvZtGkTx48fN7V/6aWXXGDlyJEeOHOHxxx9n7Nixp
ucSEhLYunUrH330Ef379+e1117j2LFjHD16NF/Nya5duxIUFERISEiJfZ4+f
ZrNmzfz4YcWnzjdNvz4Ycf8txzzzFv3jwOHDjA7NmzTUnBhA4cPH2bNmjU4Ojoy
ffp0hg0bRkREBMOGDSMyMpKWLVuW+N4OHY8PBwDh8+TOPGjfnqq
68KHQNgwYIFvPLKK0RERLB//+358fX3z9bVmzRrT7OSwYcPyPyPVfUa7n9Ne
e1f//+AvUVly5dyvDhwxk+fHiRBcNvZ+57kZKSYrpsfPtX3t+JW65evUq1atUA
Y63Ja9euFWhTo0YNxo0bR82aNalWrRoeHh707NmzxFiCg4NN9UlLQ2bC7
sCJhEMY9O74OD1c8MkrEXB0BXR4HSoWXcldCCFs7U5mrCzpVu3IqKgo
WrZsadoEMykpiVGjRnHmzBk0TTNVlgfo1q0bHh4eADRp0oQLFy4QFxd
Hly5d8Pb2Bb2BmDYsGGmYtB79+7l559/BuCJJ57gjTfeMI3Vp08fNE0jICCAqlWrE
hAQAIC/vz9RUVEEBQUBxsuRVapUMZ1XVJ9DhgzBzs6O1XXJ9DhgxBp9ORmprKnj17GDJk
iOm5W3UZ27dvz+jRWMu1XsYGRnJ22+/TWJiIqmpqfTq1avIMdq1Ky0dOPQaGfNeMzk
2bct//vMfoqOjGTx4sNm73Bf3WvK+5tvFxMSYfiZGgTHrOnj1Lhw4d0DQNe3t
7IiMjadq0aaFLeIpc1lMEd3f3/vx5KlWqxJAhQ/juu+9M
dS+L4uPjw5+Sm+RX+i7N5Cjh7QodXbR+cEEKU
Q7dmXS5cuEB2drarZ5Tdg777xD165zRTdiYyM5NdffzUthbBYjW6XSmdUjmfm
DnbXerLzs7u3z92tnZ3dGarrx9urq6AmAwGKhUqZJpLJVlERAQnTpwAjDN
M2bM4NKlSwQFBXHjxo0CttYZ5v41mqmlg3/PgwIESxx49ejTz58/n6NGj1TJkyxfReFTb
GiBEjTLNdvXr1YuvWa9vujJeS97XfDtnZ+d8P7Nl
FRUab1VV5eXvkWxsfHx5sSX3PfizudCatatWqMHz+e+fOnAsKdhNj3NLMSSmSGTZgUk0cXQotW8t1G6PRxy0mHhAT2RwyZp4PN56OOf3QLD3PD1vMRXRJ2Hu9xUJxvVYGJjmdM9FwPkmVSOX8cpq2rbKvbq0o3pRG4qdYMz+M2Zy5NLBwk7cTmr0PZCP6M8Ub2Mh6nDDzqu+zFu3FeyWnB+PMf1r4ANhOnbuX05t4cu5JlfbSkvTTGrMMO1M96vKbgkNS64TZl/4iVEJvh+rkhB6nE9pJ3i7uFvkDC5MhuufJ/GuUfWXpoexQm5nGE9+tVezL/Hi9Wi+XafoQR0p7zfNZ7gZ8s8fNTy/kHzTzn5eflXLIGl9H0E+mffywTgbb2zoqnGDz1xNDXGVqxWdI3J2F0VHKYQhRKkHqL+9ekLJlSqO49jD2L8i/3Vwl5sJa0IK32yMBNzffyFfOepwL/qSlbwzl5/DTpt3x3VBzqr6jmZ5OhuBGcXZfS+WoOdQOCWb4Z8Wk5Cpa5i9lXBWHUtPW+n17L3zvmWkkcZvqlbTQCmUczcbXLOa2ZhaBzpmP5aGTwR5vzuylEIdEq/fw2sdjFs54/nHeqWOf2gHgnOHWZjfnJnQSqS0nVYbRqEf6fhMiffuJ5ilz9M1qwhaTlvCF0c1cqlMcXNXd/K4dyvK0uuy1MsEnv0VBZmrs7qGlMO94B3RyjwUGtXf4FJQ4SzC2fEqbGjQUcKm/G2uszQRDPdSwy/9UHYy4bTtE4y77GFLWmXV1W4Qm5E+mPdsZZ+ebRlPoubfCSQXs/rqqMRlcUWvKvM8QQtEI51BuUeJYdFBzdnrtfrCm8f1twpa8/qnUyanNUgymtcyjKZNzKR/ZswgbRgHyWhVT32pv2ThP4kyhZoeUAaXhCDCfDs2k8ZVmeyqbd8uv0Z1WYF3BuEdiTsDy2RX2qfYIBaflrUg73IcUvw2AckE3JIU7JyDP/r6Q8mu8ZXiNlLh6pWCJBbj0w7dbcszdZXg9E2b4xy1IEnDnWh3w+RpBb9qMFYSNQJqIa+G1p6ag2S60P6VdqaOdvG0KOV62cFKSJZbyPu7BD46ySCdIdg6Xcm0ANYw71DRj68SWjZzLMvcK83Nor6jfBqyrmIQi6XBKKirQ3bOgwn6Vs5ymNAw0mL9MBcO1gnfmAgvWqbbKOiHY8tc2KE/FNJWRlHnkJNJKihXTV6N3tTgXoJIBE9syhRlvabeTgcI/eEpr8M3kD3L/RTGYPEV8jqrwyXzW3bIaqDyxzx9wHSaMXfLAnpE/5PwSjWqubDiZh6VDATJBMD1IvNSC5cG3lrFWy/JR+kPAozA9kt8xwQB39JiGnOKtXJJxBxR1G+nGx6Pg0WECyN85vsPdUeD8aAH1dhpMvMBX/1c8m+EIDYsXGGHdMLkJPpGQpALXE/GnPsEQQHCFcOfH4TgX1DBY6p5bDH3r/vSS17L7O+ODUIhUczA2ECFOCoVaXYIIOOB1x4skzFqjdTyvwXFqmMGBiIXsGQaHrFhRIjwtTyl/ve4xxPvkKfivPOpYEABYSgbkgUyT6ygyhgVjHP1/CT9Xx9nTR3fvXMHM80xJIGMiQNrKHhWdqXtPAcjPh4wUcpzYv1wUKUyLnJtnFIHwwBzbUfgU07wtKJvjLsPAhmuOEOOHHm4Q/3P5aWUWsTmnNSFoq8SK4hW7jlbyNkN6Em7KqiY6Wuf5VFS++7EkzPNzuzrO0mb6V0i8zwK9M/EgVuVRB9rbtcqFhIcB8mwIEVzM4lwS4i/8Rs6sWrp3l0gxHx8Ns4c0x3YX5PW6y4JSrcCVrWZSqt7E0jx5e

Rh+fLlZGcbNzhfvHixad3XiBEj2LNnD7/99pupr/Xr13P06NF8492aCSvsq0m
TJgXiGzBgAN988w0A33zzDQMHDizQpmbNmuzbt4/09HSUUmzZssWswven
T58ucCn2bkgSVoKryZlcik/npZ+Nv5z6tLoAqOxscm/cMBbp3jwFKtWEVk+X
ZahCCFFuNW/enMDAQNNi5kmTJtG+fXtyc3NLPLdatWpMnTqVtm3b8sgjj
9CiRQvTc5988gmLFi2iWbNmfPvtt3z88celjtXXcPr///nu++uorAgMD8ff3Z/Xq1
YAxwQoICKBp06Z06tSJwMBAunbtyvHjxwkKCmL58uU0a9aMuXPnMnz4c
Bo3bkzTpk1NszZ5vfvuu4SEhNCjRw/TIv6ixli+fDlNmzYlKCiIkydP3tGdj0W9l
uI0atSIpKQkUlJSiIqK4uLFi7Rp08b0fJ06dahYsSJ//vkn/fr1o2PHjrRs2ZKgo
CB2795tWiTv7OzM2rVrmXrzwDAAAB2vSURBVDdvHvXr16dJkyYsXry40J
mrOzFx4kQ2bdpE/fr12bRpExMnTgTgypUr9O3bFzAm+I899hgtWrQgICAA
g8HAc889B8CqVavw9fVl7969hIaGmi4Fg/ESdmhoaKniA9CUUqXuxJaCg4
NV3v06rG3mmzvITspm/f+3d//xNZf/48cfT5vZGCNmZH7F8mu2lflVGavkV0
hE6iu8uVWievtEyTdFb3knKknyTXojjNKb8Cm/CkNfiyNNcTM0GKbMTM
2u75/nLPTfpzZEdsZ53m/3Xbj9Xpdr9frec7lOM9d1/W6rnsWken2O3dfmcq0Z
uW4Ou0dLh8+Qo1+HfF1j4DH50JQ/1KLSymlriU2Ntah3+iVuhEffvghlStXLj
RX2O3s8uXLdOzYkW3btuHunn9ovb3PnYjsMcaE2ruWDswDswvICfHsDTqBGm
XLANFL17OppJ7OTwrx9Op9n2M3h5JynMLca9Vi7qzP8Z7/2jwDILAfk6OX
CmllCpdI0aMyNc97AoSSEhJ49913CCyVgf4cmYQUc/jOd8Sv+6od+7LIHPh6Q
nJlMhysNSFkwkyq9elLrzbdw+3U+nEuAQR9BOe3ZVUop5Vo8PPksPT0ZNKly64
8MoVQEBAQ4/eVoczRwKyM7JAeDjgffw27+68lCzmlSpaNnXoqqlP75Kly64
uWVB5DS4KxwaPei0eJVSSil1a9IkrAge7uXwLO+GbWMHhIzuDWpVq4eVZ5
Dg9hlwKRU6T3JekEoppZS6ZWl3pAMysi6i2hdGbWu1/mtnzHIwOfDgG84LTCmll
yullFJKXYO2hBUjPesiaZnZ1PHWzXU9x3Cipga
aZnhN18SdiwSWg+HavWdFJVSSpV9SUlJPPXUU9x11120atWK9u3bs2LFi
hK/7+7du/Ot93i9GjRoQN++fW3by5cvZ8iQIYBlklFfX19CQkJo0aIF/fr1IyMj
w+51Vq5cydtvv51vX3BwMAMHDsy3r1OnTuSdeik+Pj7fRKC7du0iLCyMJk2
a0LRpU4YPH17kPR117kPR117Ngx2rZtS0BAAAMGDLBNolrQq6++SosWLWjWrB
kvvfQSudNazZo1i8aNGyMinD171lZ+zZo1vPXWWzcUmyvRJKwYWUl/UOu
ccPXlt/hjkvXD5O4JHcY4NzClFCrDjDE89DE89thjhIWFEcfToUdvs6i9ezEiSV+79DQ
UGbOnHlD19i9ezcHHDhyweyx3Ie4DBw7g4eHBBsmXL7JZ777338i2EHRsbS0
5ODpGRkVy8eNGhOJqcXFxxMbG0rVr13yz7/GqNNHj
+bw4cNUq1bNtjB4Xj/99BL6pUbN26kfv38
jRE9evRg1apVN5wkugodE3YTzYS0Dk4YKMWe6ZxzV1HxhygUbhFRF
u7s8JRSyiF/TJnCpf0HSvSaFZo1pdb48UUe/+GHH/Dw8OC55/Jzo7M2jaM2jGCMCOQHb
wNLaM2jQIFsyMmvWLO677z7S09OnO677z42b97ReO0iNDSUIUOUIUOG
G7cOFatWo06dA5dP5+uvv2bGjBmEh4cTGxtLUFDQjYbE0aNHqampWb1cw0YzNRLpgVa4KEXZ4LSG0VBTx+TqvDUU5v
zn1y6dAkvLy+PyN98Ry7W+wrEn7//+y7//wDgfOnKp6EOrq7b0mpl4uWgjxAfw9n6W8edsTTSmsc
QVlZ2dz8eJFqlWrVuVYoWoeOHqFChQr6FwZC6sSgWWsXngQOIjY1lPrlapVhrVrE7Pn
kk08YPHGw7du3BWzrzWPPzPUhpj+PHHH1myZAknLEiYwYMSJ
fOREhMzOTK1euYLPz8/wLICgj0iQqdOnVizZ39+sE5sXRJKwAY

wwhl91IiTrD+rhErrr5QjlosDAC5nWGtArQfayzw1RKqTLtwIED+ZYXKqhm
zZps2LABT09PDh8+zMCBA7nWaigpKSmsWLGC3377DRHh3LlzALz99tus
W7eOOnXq2Pbl1bRpUyIjI3F3d2fjxo2MHz+eb775BoDo6Gj27dtHhQoVaNK
kCS+++CJ169YFoH///syePTvf2oi5li1bxrZt2zh9+jR33303PXv2LFRm+/bthV7
/smXL2LBhA3FxccyaNcuhJCwmJobBgwcXWy4uLo4BAwbYPbZ582aqVq1q
205OTqZq1aq2yUb9/f05efJkofPat29PeHg4tWvXxhjDqFGjHFqFITQ0lK1bt2
oS5gBNwgq4+GcmnS95kLLzDCkA5WtR6VIcxK6CxCjo9TF4VHR2mEop5b
BrtViVlpEjR7Jt2zY8PDyIiooiKyuLUaNGER0djZubm20h66JUqVIFT09Phg8
fTo8ePXj00UcBS7fYkCFD6N+/P48//nih89LS0hg8eDCHDx9GRMjKyrIde+i
hh/Dx8QGgefPmHD9+3JaEubm5MXbsWP7973/TrVu3fNccMGAAs2bNwhj
DyJEjmTZtmm1dwlynT5/G19fH39+ff/zjH6SmplKtWrW/
nrzPw96+a2nSpAnR0dEOlbW3XKG9+x05coTY2FhbF3LnzsnzB1fbF3Lnzp2JjIwkLCzsmte
vWbMmp06dcigWV6djwgrI/cdZq6s/cX2/pUbSa9x7bh1snAS+TSH4KSdHqJR
SZV+LFi3Yu3evbfuTTz7hhx9+4MyZM4BlzUE/Pz9++eUXdu/ebRsY7u7uTo5
10myAzMxM2/5du3bRt29fVq5cSdeuXQGYM2cOkydP5sSJE4SEhJCcnJwvjg
kTJhAeHk5MTAyrV6+2XQ+gQoUKtr+7ubmRGRlJQkKC3d
coIvTs2ZPIyMhCx7y8vPLdKyIigt9++40GDRrQqFFEjzp8/b2uRq169Oqmpqba
yKSkptm7MFi1asGfPHrv3zysuLo6QkBC7PwVbCGvUqMG5c+c+dsrzcxMZE77
7yz0DVXrFhBu3bt8Pb2xtvbm27durFz585iY8nMzMTLy6vYckqTsCKZcjnsPh
vFHZ5VkEspkPI7PDwR3LTxUCmlivPggw+SmZnJp59+atuWkpJSqcwsWNxbMz97sdWqOYpa
Xply5cnz55ZdcvXoVsIwbO3jwIJcvXyYtLY0ffvgBgPT0dNLS0ujevTszZsywtfr
8/vvvtG3blrfffhtbW1tvbm27durFz585iY8nMzMTLy6vYckqTsCKZcjnsPh
8/vvvtG3blrfffpsaNWpw4sSJfHGkpaVRp04dwPJk4/UoX748o0ePZsaMGUW
W2bZtG40aNSq0v1mzhl7ZrauzJycHL7++mv2799PfHw88fHxfPvtt0RERACWpy
MXLVpkawRYDB4eHhgGVM3IIFC/j5559t1160aBF//Hvvvvlt0TZ+8nbF
QmW5DE8PJzly5fb7te7d+9Cr6FevXps2bKF7OxssrKy2LJlJli0PdkYcOHcr3dK
cqmiZhRUi9/CcXrlygmocPpP8J9drD3V2dHZZSSt0SRISVK1eyZcsWGjZsJ
s2bRg8eDBTp04F4IUXXmDBggW0a9eOw4cP2/7jbt++ndDQUB59rPozU6UalSJQDq1q1L//79CQoK4u
mnn7YNAL9w4QKPPvoG9C7PzP8//BCAsWPH0rJlJSwIDAwkLCyM4
OP8E2q+++iqvv/46999/Dhg0r1EK2bNkyQkJCCCAoKYt++fUyYMKH
QeWFhYezbtw9jDJGRkdSpU8eWDOPe3jwIKdPn+bZZ5+lcuXKBAcHExw
cTHp6OmPGWJ7A9/PzSztm7dSpUqVa77teQ1dep
UPvjgAxo3bkxycjcjLDhg0DLE+FDh8+HIB+/frRuNobaGrZT6bMJkz8ff3Jz
ExkaCgINs5AJs2baJHjx4+rEHt9w2VZaGioudbgzRu1c+9p9nwWS/J9J/j
aTGfF/9akXMJx7vp6CdRtU2L3VUqpmyk2NtahVgtVcl7Vcl5++WV69w
xQSk3u3HC5LZiuxt7nTkT2GGNC7ZXlrAinL50jPre/lRIPQ6eVTQBU0opd
V3Gjx/vcvNlJSQk8klJEejF8P/0h9cSnLDvV4tJwellF
LqVuPn50evXr2cHUapBagpat26dGFLlI22hBV0xjxyjoIODvHRURmWo/vF5GiZ/YrZZGB5fDvmk
GDr1ZQ3TpgUSmllFLqZimxJExE3IBPPgG5Ac2CgiDQvUKwbEGD9eRb4lD
KianYOvu1GQmU/Z4eilFJKqdtQSbaEtQGOGGOOmOuAEuBghOR9AYW

GoudQFURqV2CMRUrx/oIc60c4L4XnRmKUkoppW5jJZmE1QHyzpqXaN13
vWVK1bkzuwCoXLEBVKjszFCUUuqW5u3tXWjfnDlzWLhwYYnf+4svvqBly5
YEBQURGBjIt99+y/z58wut13j27Fl8fX25fPkyWVlZjBs3joCAAAIDA2nTpg3f
f/+93ev369ePo0eP2rb37duHiLBu3Trbvvj4+EKTlk6cOJHp06fbtqdPn07
JDAwkODj4prw3CxYsICAggICAABYsWGC3TEJCAuHh4dxzzzz0EBQXx3Xf
f5Tt+/vx56tSpw6hRo2z7nnzySQ4fPnzD8am/lOTTkfYWvio4KZkjZRCRZ7F0
V1KvXr0bj+waQroPICNlHoHd/k+J3kcppVzR888/X6LXN8Zw4sQJ3nnnHfb
u3YuPjw/p6emcOXOG6tWrM2bMGDIyMqhY0bIG8PLly+nVqxcVKlRg3Lhx
nD59mpiYGCpUqEBSUhJbtmwpdI8DBw5w9epV7rrrLtu+iIgIgIHnjgASIiIujSp
YtDsc6ZM4cNGzawa9cuqlSpQlpaGitXrryh15+SksKkSZPYvXs3IkKrVq3o1as
X1apVy1du8uTJ9O/fnxEjRnDw4EG6d+9OfHy87fiECRPo2LFFjvnNGjBjBe+
+9x9y5c28oRvWXkkzCEoG6ebb9gYIrejpSBmPMZ8BnYJms9eaGmd+dde+k
3+uFZz9WSqlb1davDnH2RPpNvWaNut506H/3dZ83ceJEvL29GTNmDJ06d
aJt27Zs2rSJc+fOMW/ePDp06MDVq1cZN24cmzdv5vLly4wcOZLnnnuO9PR
0evfuTWpqKllZWUyePXJnevXsTHx9Pt27dCA8PZ8eOHcyYMYPKlSvbWuJy1
z4Ey0z1q1evZsCAAQAsXbqUN954g4yMDObOncuxY8dsa0r6+fnRv3//Qq9h8
eLF+Zb5McawfPlyNmzYQIcOHcjMzMTT07PY92ThnwcsSzvlXTTtyz549JCUl0
bVrV/JOjt6hQweGDBlCdnY27u46w9w9XNUJLdkVFAgIg0FBEP4ElgVYEyq4
BnrE9JtgPSjDGnSzAmpZRSZUh2dja7du1ixowZT0CYB58+bh4+NDVFQ
UUVFRtuTI09OTFStWsHfvXjZt2sQrr7xiW28xjZi6xiW28xLi6OZ555hn379vHAAw/g5+
dHw4YNGTp0KKtXr7bdb+DAgSxatAiA5OTkmHPuHOvbGfkajfrFgmavNkdBJjxuOp
du306pVq3zbDRs2pFGjRnTq1MlfgSxsfACa3YHNcyd8Xo68CcD2rrNakzudmzutmll
wqVPXnyJHXr/tW24e/vz8mTJwuVmzKrHpl2p5l1hs97zGUmA3Kn/7oApjs/.
vK0bNjQK1WqUKlSJY4ePZpvJk906ZNY/Hix5HzZ7/P392bx5s207MTGRT
p06FTp33rx5rF27FoD27duTmZnJ2bNn2bFjB1u3bmX27Nmkp6dz5coVvL29
effddwHIzMzEy8vrmvEqx+mM+UoppcqULl268Omnn5KVlQXAoUOHujxI
mlpadSsWZPy5cuzadMmjh8/bvf8U6dOsXfvXtt2dHQ09evXt20PDz74gK
SkJFvrWcWKFRk2bBgvvfQSV65cssLSWXosWlLSp0/WbNmnHkyBEANm7cS
HBwMCdOnCA+Pp7jx4/Tt29fVq5cSibe3N7Vr17YtZp2SksLatWt54IEHAj99

dcZOXKkbWzW+fPn+eyzzwrdb+zYsURHRxf6KZiA5b5369evJzU1ldTUVNav
X2/3QYF69erZ4oqNjSUzMxNfX18WL15MQkIC8fHxTJ8+nWeeecaWgOXW
RYsWLey+7+r6aRKmlFKqRGRkZODv72/7+eCDDxw6b/jw4TRv3px7772Xw
MBAnnvuObKzs3n66afZvXs3oaGhLF68mKZNm9o9PysrizFjxtC0aVNCQkJ
YtmwZH330ke34I488wqlTpxgwYEC+rrrJkyfj6+tL8+bNCQwM5LHHHsvX7
ZirR48ettamiIgI+vTpk+943759WbJkCQALFy5k8uTJhISE8OCDD/LWW2/Z
xoGNGDGC8PBwWrduTWBgIB07drQ9tfl33XHHUyYMIHWrVvTunVr3nz
zTdsg/TfffJNVqyxDs99//33mp1LcHAwAwcOZP78+cV2jyYlJeHl5UXt2k6dzv
O2Ivb6j8uy0NBQk/dpDaWUUoXFxsbSrFkzZ4dxuW7p06useT3
co+/PBDqlSpwjBdyq9I9j53IrLHGBNqr7y2hCmllFLXwcvLi0mTJtl96vB2VrV
q1RueQkPlpxN9KKWUUtfJ0QlZbydDh+o+PEBjebtoQppdRt6lYbbqLUrezvfN40
CVNKqduQp6cnycnJmogpVQpypyZxZKWEvLQ7UimlbkP+/v4kJiZy5swZZ4e
ilEvw9PTE39//us7RJEwppW5D5cuXp2HDhs4OQyl1DdodqZRSinlBJqEKa
WUUko5gSZhSimllFJOcMvNmC8iZwD7C4bdXDWAs6V+U4iwtLEyR4+ukb
NJ6KXu0Tsqm0qiX+saYwutfcQsmYaVFRHYXtcyAcg6tk7JH66Rs0nope7RO
yiZn14t2RyqllFJKOYEmYUoppZRSTqBJWNE+c3YAqhCtk7JH66Rs0nope7
ROyian1ouOCVNKKaWUcgJtCVNKKaWUcgJNwpRSSimlnMClkzAR6SoicS
JyRETG2TkuIjLTeny/iNzrjDhdjQP18rS1PvaLyE8iEuyMOF1JcXWSp1xrEb
kqIv1KMz5X5Ui9iEgnEYkWkQMisqW0Y3Q1Dvz/5SMiq0XkF2udDHVGnK5
ERL4QkT9FJKaI4077rnfZJExE3IBPgG5Ac2CgiDQvUKwbEGD9eRb4tFSD
dEEO1ssxoKMxJgj4FzrgtUQ5WCe55aYC60o3QtfkSL2ISFVgNtDLGNMCe
KLUA3UhDn5WRgIHjTHBQCfgfRHxKNVAXc98oOs1jjvtu95lkzCgDXDEG
HPUGHMFWAr0LlCmN7DQWOwEqopI7dIO1MUUWy/GmJ+MManWzZ2
AfynH6Goc+awAvAh8A/xZmsG5MEfq5Sngv8aYBABjjNZNyXKkTgxQWUQ
E8AZSgOzSDdO1GGMisbzPRXHad70rJ2F1gBN5thOt+663jLq5rvc9HwZ8X
6IRqWLrRETqAH2AOaUYl6tz5LNyN1BNRDaLyB4ReaeZc
Ar4FXjZGJNTOuGpIjjtu969NG5SRomdfQW/U+tIGXVzOfyei0hnLEnYQyU
GpBypkxnAa8aYq5Zf8FUpcKRe3IFWQDjgBeTogPAZtNdUKNHLHx0YygYz2bWlt
MKBEXPhLeMTARZUr4h7VaddfKqLmLq/bg/Qv7qUsTsY0zEBfrBGxkWOXqL
GZYUxDkCA4tXpRVtu0DHGGM0M02NKVk7Cfw+rBhh+QcK9ENnwvUaRqHngHWMTmdGM99NrCd
E7P3ndQ9brK3RDhZaUfH/QJVyiQqWM05JZtjtRMTSnZo15bBnmMeZbtJLJBWl2K
cWqwmvS7wAQZ2FKe6RMlSHZJ6KI8ZNoIdqXQiu72XfJ+6nqQnUUrc8w6tE
nCuQNRr9n9x6GfI+RKeGapC5mbR/O8k7nvzqSgHKLKfHXOKbq2EnqMP7iB
0gBUT+3BajhGsRCvejeajJCsbFxLuHaU3qm8Sv8ykBxuXqkVNnhLjI98TXEZ
j7KcaQxXKFXMTZGhQ+qPtIJ+ZA7IhiqzSHJhw+K/98qvcWWnzM4W0Mrcp2KSF
XDfXZiDZiEyAfXR92DVPoH+4Ahm2lvQWE8VQ8P5PfMOaWjwDv4Im6

```
cdU3HVFZAFwyBjzzjXKDwFCjTGjbnYsSinn05YwpVSJExFvEfnB2kr1q4j0t
lOmtohE5mkp6mDd/4iI7LCe+7WIFJccRQKNref+j/VaMSLyT+u+SiLyvyLyi
3X/AOv+zSISKiLvAl7WOBZbj6Vb/1yWt2XK2gLXV0TcRGSaiESJyH4Rec6B
t2UH1kWCRaSNiPwkIvusfzaxzrj+NjDAGssAa+xfWO+zz977qJS6dbjsjPlKqR
LlJSLR1r8fA54A+hhjzotIDWCniKwy+ZvinwLWGWPeERE3oKK17BvAw8a
YiyLyGvA/WJKTovQEfhWRVlhmvm6LZWbyn0VkC3AXcMoY0wNARHzyn
myMGScio4wxIXauvRQYAHxnTZIeAkZgWUg+zRjTWkQaANtFZL0x5pi9A
K2v7yFgnnXXb0CYdcb1h4Epxpi+IvImeVrCRGQQKK8KMx5h/WrsxdIrLRujak
UuoWo0mYUqokXMqbxIhIeWCKiIRhWT6nDuAH/JHnnCjgC2vZlcaYaBHp
CDTHktQAeGBpQbJnmoi8AZzBkhQ9BKzITVBE5L9AB2AtMF1EpmLpwtx
6Ha/re2CmNdHqCkQaYy5Zu0CDRKSftZwPEIAlAc0rNzltAOwBNuQpv0BE
AgCDdUkVOx4BeonIGOu2J1APXQ9SqVuSJmFKqdLwNOALtDLGZIlIPJY
EwsYYE2lN0noAX4rINCAV2GCMGejAPcYaY5bnblhblAoxxhyytpJ1B/5tbbG
6Vsta3nMzRWQz0AVLi1hE7u2AF40x64q5xCVjTIi19W0NMBKYiWU9wU3
GmD7Whxg2F3G+AH2NMXGOxKuUKtt0TJhSqjT4AH9aE7BwoH7BAiJS3
1pmLpZuunuBncD9IpI7xquiiNzt4D0jgces51QC+gTMKVUaVgMhIrIbiytYr/ZKdMJiBa
RfUBf4CNjzBksSUmEiOzHkpQ1deSGxpi9wHxgF/Az8LkxZh/QEstYqmjg/w
KT7Zz+GbA/d2B+AeuBMGCjMeaKdd/nwEFgr4jEAP+PYnoarLH8AjwJvIel
VW474Jan2Cagee7AfCwtZuWtscVYt5VStyidokIppZRSygm0JUwppZRSygk0
CVNKKaWUcgJNwpRSSimlnECTMKWUUkopJ9AkTCmllFLKCTQJU0opp
ZRyAk3ClFJKKaWc4P8Dio83mopjxTkAAAAASUVORK5CYII=\n",
      "text/plain": [
       "<Figure size 720x504 with 1 Axes>"
      ]
     },
     "metadata": {
      "needs_background": "light"
     },
     "output_type": "display_data"
    }
   ],
   "source": [
    "plt.rcParams[\"figure.figsize\"] = [10,7]\n",
    "disp = plot_roc_curve(lr,X_test,y_test)\n",
    "plt.title(' Area Under Curve Analysis\\n')\n",
    "plot_roc_curve(dt,X_test,y_test,ax=disp.ax_)\n",
    "plot_roc_curve(rfc,X_test,y_test,ax=disp.ax_)\n",
    "plot_roc_curve(gnb,X_test,y_test,ax=disp.ax_)\n",
    "plot_roc_curve(svr,X_test,y_test,ax=disp.ax_)"
   ]
  },
  {
   "cell_type": "markdown",
   "metadata": {},
   "source": [
    "\n",
    "### Discussions:\n",
    "\n",
    "As LogisticRegression is having high AUC values among all the algorithms, we'll work with LR to make the predictions for the COVID-19 Infection Probability."
   ]
  },
  {
   "cell_type": "code",
   "execution_count": null,
   "metadata": {},
   "outputs": [],
   "source": []
  }
```

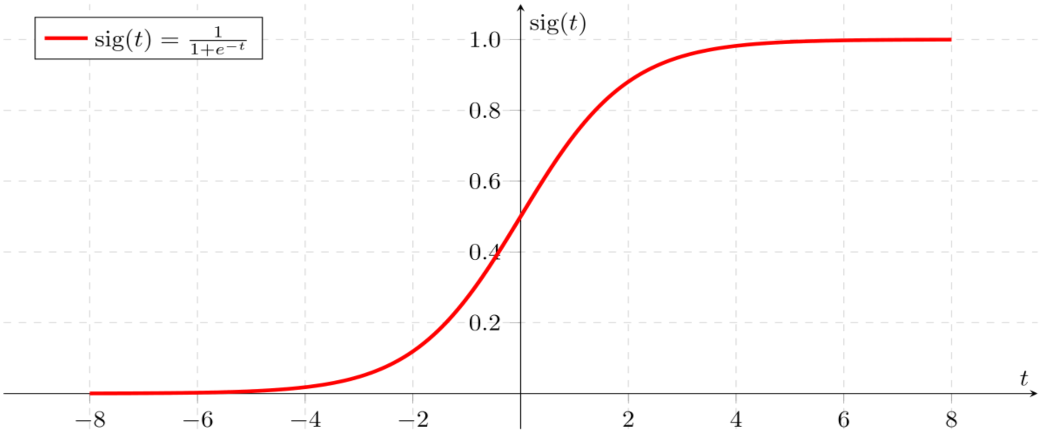
```
],
"metadata": {
 "kernelspec": {
  "display_name": "Python 3",
  "language": "python",
  "name": "python3"
 },
 "language_info": {
  "codemirror_mode": {
   "name": "ipython",
   "version": 3
  },
  "file_extension": ".py",
  "mimetype": "text/x-python",
  "name": "python",
  "nbconvert_exporter": "python",
  "pygments_lexer": "ipython3",
  "version": "3.7.3"
 }
},
"nbformat": 4,
"nbformat_minor": 4
}
```

Algorithms Used

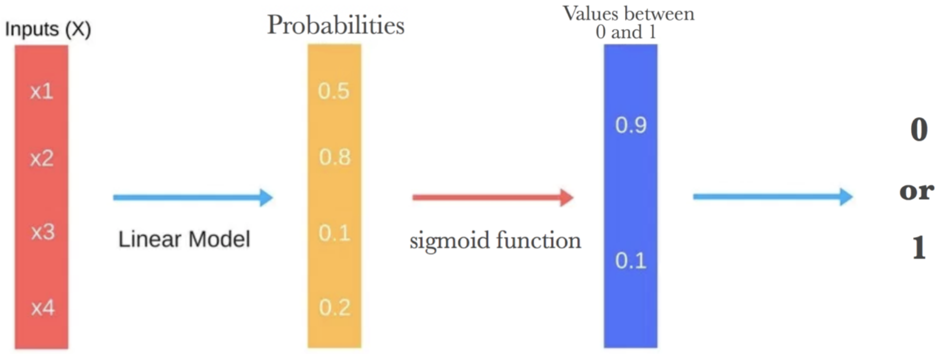
1. *Logistic Regression*

This method of regression is a famous method after Linear Regression. Though, both of them are regression algorithms the underlying difference is that the

Logistic Regression is used for classification tasks while the other is used for predicting or forecasting of data provided to the model. This algorithm also uses a linear equation to predict the value but the output range lies between negative and positive infinity. To have a classification output we will be having binary data for which 1 means Yes and 0 means No which sums it up to have the range 0 to 1. For squeezing the output data to be in this range, the sigmoid function is used.



Below is the graphical representation of Logistic Regression input and output scenario.



2. *Decision Tree Classifier*

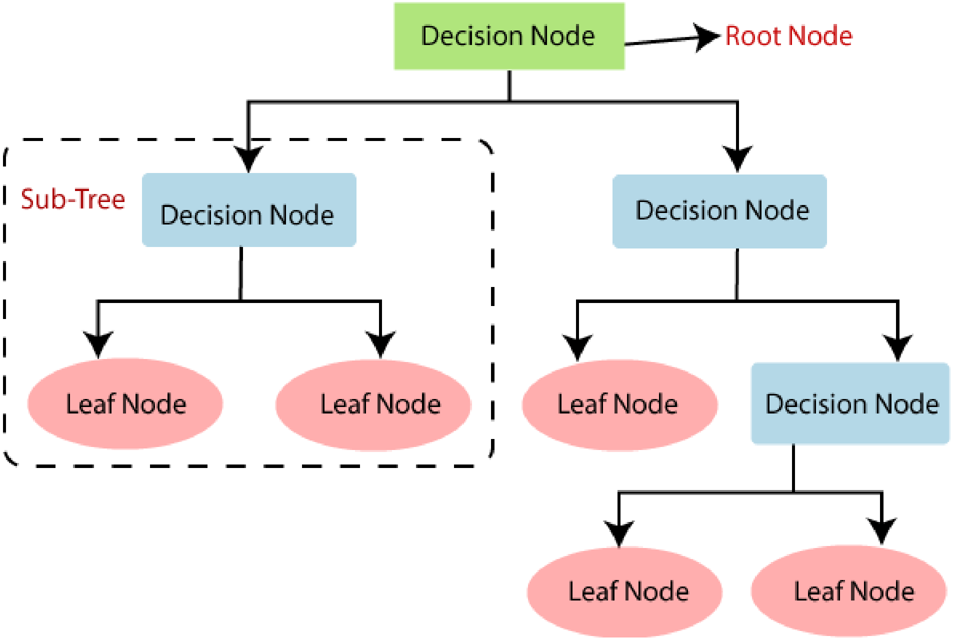
A Decision Tree is a simple representation for classifying examples. It is a Supervised Machine Learning where the data is continuously split according to a certain parameter.

Decision Tree consists of:

- Nodes: Test for the value of a certain attribute.

- Edges/ Branch: Correspond to the outcome of a test and connect to the next node or leaf.
- Leaf nodes: Terminal nodes that predict the outcome (represent class labels or class distribution).

Below is the graphical representation: -



There are two main types of Decision Trees:

- Classification Trees.
- Regression Trees.

2.1. Classification trees (Yes/No types):

What we've seen above is an example of classification tree, where the outcome was a variable like 'fit' or 'unfit'. Here the decision variable is Categorical/ discrete.

Such a tree is built through a process known as binary recursive partitioning. This is an iterative process of splitting the data into partitions, and then splitting it up further on each of the branches.

2.2. Regression trees (Continuous data types):

Decision trees where the target variable can take continuous values (typically real numbers) are called regression trees. (e.g., the price of a house, or a patient's length of stay in a hospital).

Creation of Decision Tree:

In this method a set of training examples is broken down into smaller and smaller subsets while at the same time an associated decision tree gets incrementally developed. At the end of the learning process, a decision tree covering the training set is returned.

The key idea is to use a decision tree to partition the data space into cluster (or dense) regions and empty (or sparse) regions.

In Decision Tree Classification a new example is classified by submitting it to a series of tests that determine the class label of the example. These tests are organized in a hierarchical structure called a decision tree. Decision Trees follow Divide-and-Conquer Algorithm.

Decision Tree Classifier

Using the decision algorithm, we start at the tree root and split the data on the feature that results in the largest information gain (IG) (reduction in uncertainty towards the final decision).

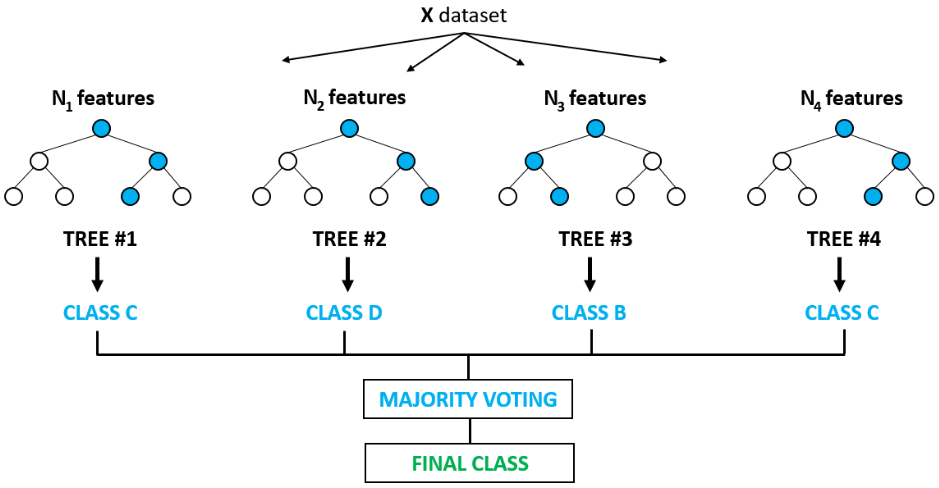
In an iterative process, we can then repeat this splitting procedure at each child node until the leaves are pure. This means that the samples at each leaf node all belong to the same class.

In practice, we may set a limit on the depth of the tree to prevent overfitting. We compromise on purity here somewhat as the final leaves may still have some impurity.

3. Random Forest Classifier

Random forests are a supervised learning algorithm. It can be used both for classification and regression. It is also the most flexible and easy to use algorithm. A forest is comprised of trees. It is said that the more trees it has, the more robust a forest is. Random forests create decision trees on randomly selected data samples, gets prediction from each tree and selects the best solution by means of voting. It also provides a pretty good indicator of the feature importance.

Random forests have a variety of applications, such as recommendation engines, image classification and feature selection. It can be used to classify loyal loan applicants, identify fraudulent activity and predict diseases. It lies at the base of the Boruta algorithm, which selects important features in a dataset.



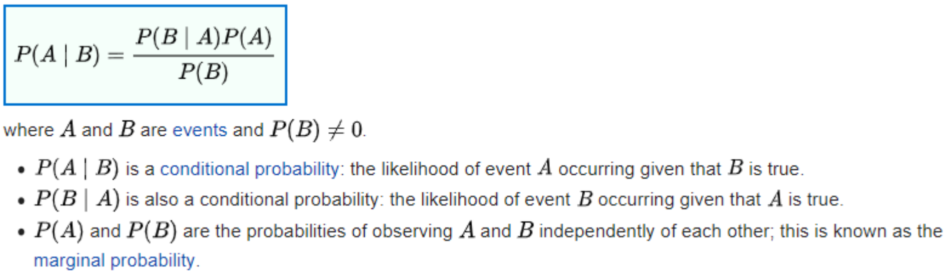
It works in four steps:

1. Select random samples from a given dataset.
2. Construct a decision tree for each sample and get a prediction result from each decision tree.
3. Perform a vote for each predicted result.
4. Select the prediction result with the most votes as the final prediction.

4. *Naive Bayes*

Naive Bayes is the simplest algorithm that you can apply to your data. As the name suggests, here this algorithm makes an assumption as all the variables in the dataset is "Naive" i.e not correlated to each other.

Naive Bayes is a very popular classification algorithm that is mostly used to get the base accuracy of the dataset.



$$\boxed{P(A \mid B) = \frac{P(B \mid A)P(A)}{P(B)}}$$

where A and B are events and $P(B) \neq 0$.

- $P(A \mid B)$ is a conditional probability: the likelihood of event A occurring given that B is true.
- $P(B \mid A)$ is also a conditional probability: the likelihood of event B occurring given that A is true.
- $P(A)$ and $P(B)$ are the probabilities of observing A and B independently of each other; this is known as the marginal probability.

Advantages

- It is easy and fast to predict the class of the test data set. It also performs well in multi-class prediction.

- When assumption of independence holds, a Naive Bayes classifier performs better compare to other models like logistic regression and you need less training data.
- It performs well in case of categorical input variables compared to numerical variable(s). For numerical variable, normal distribution is assumed (bell curve, which is a strong assumption).

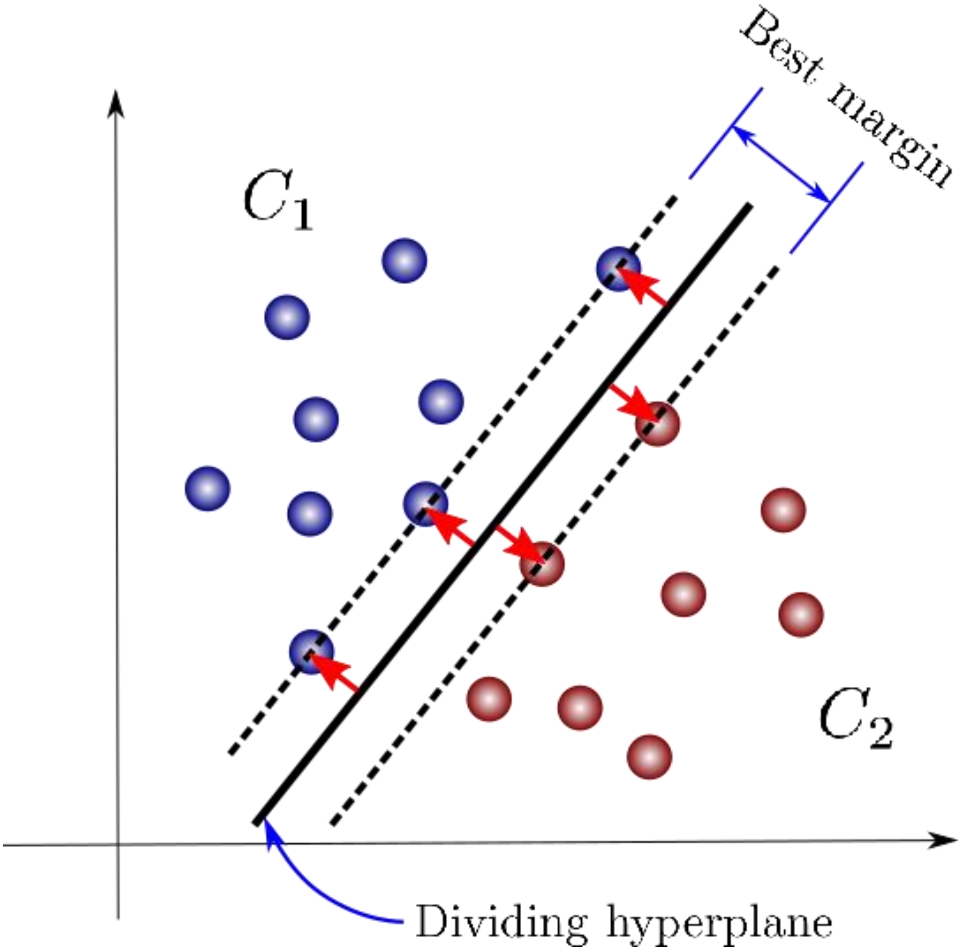
Disadvantages

- If categorical variable has a category (in test data set), which was not observed in training data set, then model will assign a 0 (zero) probability and will be unable to make a prediction. This is often known as Zero Frequency. To solve this, we can use the smoothing technique. One of the simplest smoothing techniques is called Laplace estimation.
- On the other side naive Bayes is also known as a bad estimator, so the probability outputs are not to be taken too seriously.
- Another limitation of Naive Bayes is the assumption of independent predictors. In real life, it is almost impossible that we get a set of predictors which are completely independent.

5. *Support Vector Machine*

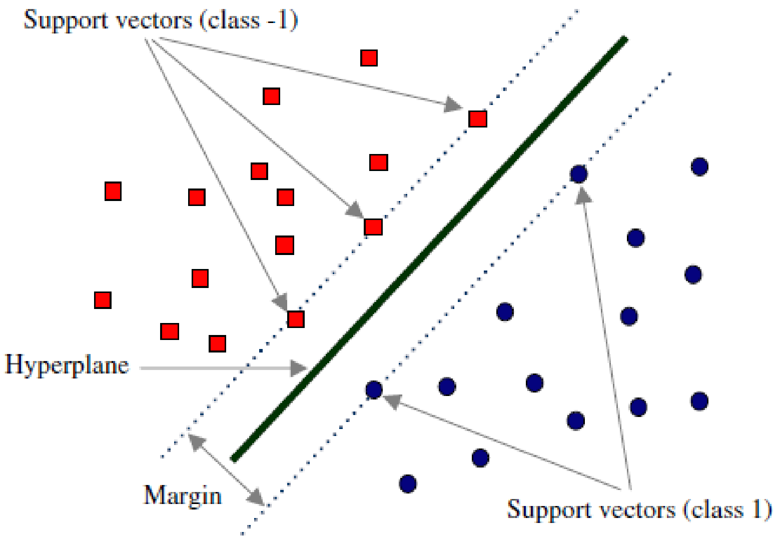
The objective of the support vector machine algorithm is to find a hyperplane in an N-dimensional space (N — the number of features) that distinctly classifies the data points.

Following is the visual representation:

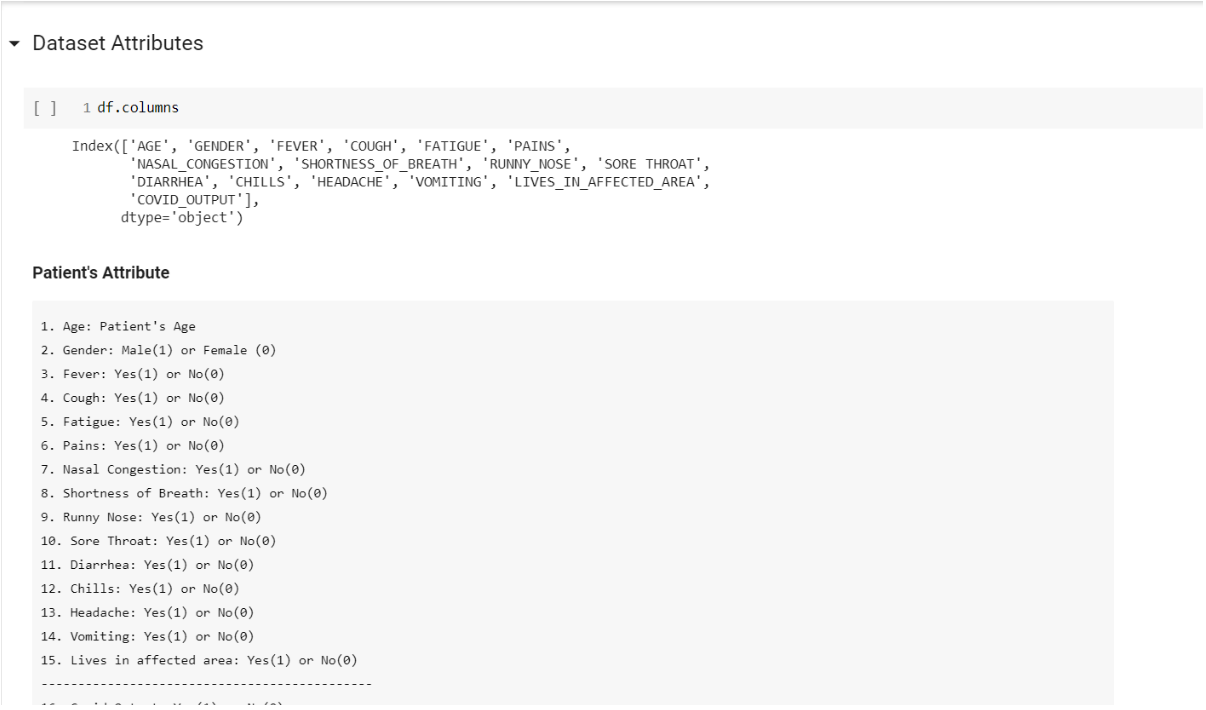


To separate the two classes of data points, there are many possible hyperplanes that could be chosen. Our objective is to find a plane that has the maximum margin, i.e the maximum distance between data points of both classes. Maximizing the margin distance provides some reinforcement so that future data points can be classified with more confidence.

Hyperplanes are decision boundaries that help classify the data points. Data points falling on either side of the hyperplane can be attributed to different classes. Also, the dimension of the hyperplane depends upon the number of features. If the number of input features is 2, then the hyperplane is just a line. If the number of input features is 3, then the hyperplane becomes a two-dimensional plane. It becomes difficult to imagine when the number of features exceeds 3.



In logistic regression, we take the output of the linear function and squash the value within the range of [0,1] using the sigmoid function. If the squashed value is greater than a threshold value(0.5) we assign it a label 1, else we assign it a label 0. In SVM, we take the output of the linear function and if that output is greater than 1, we identify it with one class and if the output is -1, we identify is with another class. Since the threshold values are changed to 1 and -1 in SVM, we obtain this reinforcement range of values([-1,1]) which acts as margin.

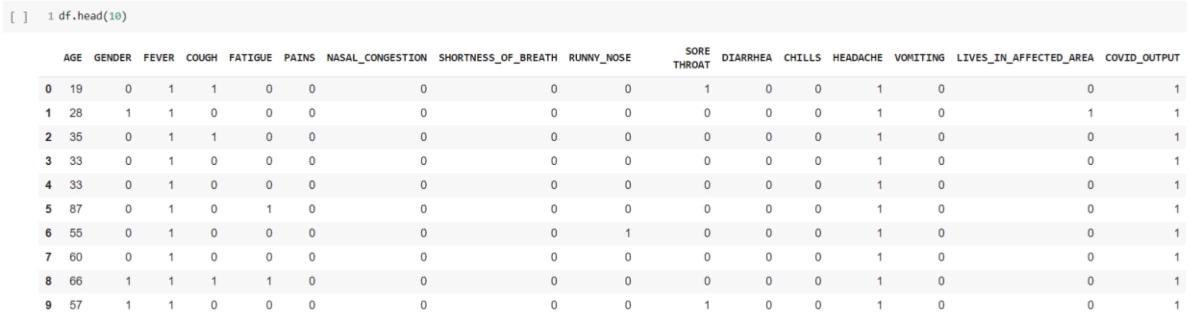


```
df.columns
```

```
Index(['AGE', 'GENDER', 'FEVER', 'COUGH', 'FATIGUE', 'PAINS',
       'NASAL_CONGESTION', 'SHORTNESS_OF_BREATH', 'RUNNY_NOSE', 'SORE THROAT',
       'DIARRHEA', 'CHILLS', 'HEADACHE', 'VOMITING', 'LIVES_IN_AFFECTED_AREA',
       'COVID_OUTPUT'],
      dtype='object')
```

Patient's Attribute

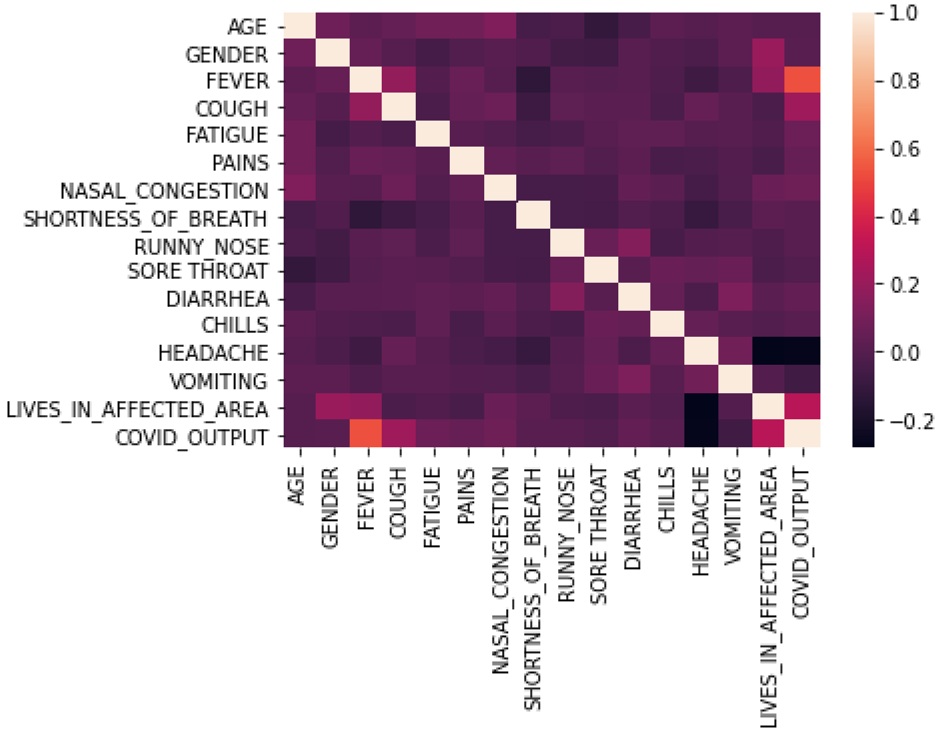
```
1. Age: Patient's Age
2. Gender: Male(1) or Female (0)
3. Fever: Yes(1) or No(0)
4. Cough: Yes(1) or No(0)
5. Fatigue: Yes(1) or No(0)
6. Pains: Yes(1) or No(0)
7. Nasal Congestion: Yes(1) or No(0)
8. Shortness of Breath: Yes(1) or No(0)
9. Runny Nose: Yes(1) or No(0)
10. Sore Throat: Yes(1) or No(0)
11. Diarrhea: Yes(1) or No(0)
12. Chills: Yes(1) or No(0)
13. Headache: Yes(1) or No(0)
14. Vomiting: Yes(1) or No(0)
15. Lives in affected area: Yes(1) or No(0)
---------------------------------------------
16. Covid Output: Yes(1) or No(0)
```



```
df.head(10)
```

	AGE	GENDER	FEVER	COUGH	FATIGUE	PAINS	NASAL_CONGESTION	SHORTNESS_OF_BREATH	RUNNY_NOSE	SORE THROAT	DIARRHEA	CHILLS	HEADACHE	VOMITING	LIVES_IN_AFFECTED_AREA	COVID_OUTPUT
0	19	0	1	1	0	0	0	0	0	1	0	0	1	0	0	1
1	28	1	1	0	0	0	0	0	0	0	0	0	1	0	1	1
2	35	0	1	1	0	0	0	0	0	0	0	0	1	0	0	1
3	33	0	1	0	0	0	0	0	0	0	0	0	1	0	0	1
4	33	0	1	0	0	0	0	0	0	0	0	0	1	0	0	1
5	87	0	1	0	1	0	0	0	0	0	0	0	1	0	0	1
6	55	0	1	0	0	0	0	0	1	0	0	0	1	0	0	1
7	60	0	1	0	0	0	0	0	0	0	0	0	1	0	0	1
8	66	1	1	1	1	0	0	0	0	0	0	0	1	0	0	1
9	57	1	1	0	0	0	0	0	0	1	0	0	1	0	0	1

Correlation Matrix



Results and Discussion

In machine learning, the performance of individual model which is tested is measured by the following factors:

AUC Score: From the prediction scores, the Receiver Operating Characteristic Curve (ROC) is plotted and the area under the curve defines the following metric.

Recall: It is defined as the number of true positives divided by the total count of false negatives and true positives.

Accuracy: It is one of the metrics which is used to compare the correct predictions with all the data points of the classifier.

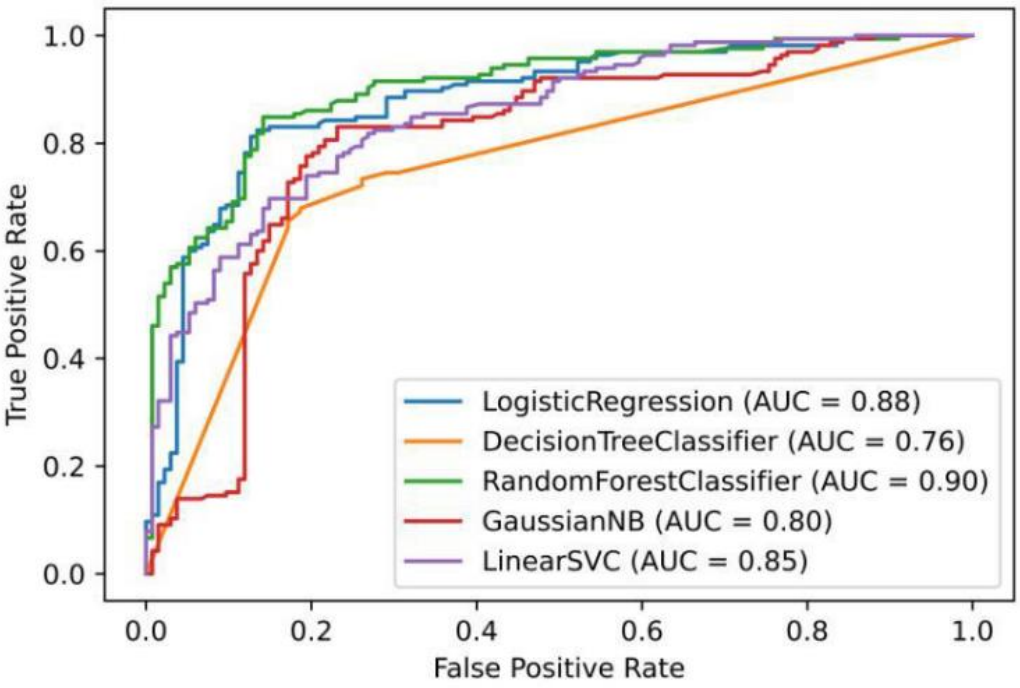
Precision: It is defined as the number of true positives divided by the total count of false positives and true positives.

F1 Score: It is defined as the weighted average of the precision and recall.

Various classification algorithms are applied and the COVID19 Symptoms dataset is provided for the performance analysis of different models.

Conclusion

At present the information about the symptoms to COVID-19 infected outcome is present at a very small scale while the future work will be able to draw a better understanding between the symptoms and the infection in a greater detail as the contribution of adding new data will be done.



This paper gives an insight of the different symptoms linked to COVID-19 and proposed a probabilistic classification of getting infected. The work concludes that men have a higher tendency to get infected having common symptoms as fever and cough. To choose the best algorithm to train the dataset, statistical analysis is carried out. Random Forest Classifier Algorithm is superior to other models in comparison to various performance factors such as accuracy, AUC

Score, F1 Score and Recall. The approximate mean accuracy score using these algorithms was found to be 78%.