EX1

In [16]:
```
df = pd.read_excel('Rice_Cammeo_Osmancik.xlsx')
df_quant = df.select_dtypes(['int64', 'float64'])
scaler = StandardScaler()
scaler.fit(df_quant)
df_quant_scaled = scaler.transform(df_quant)
df[df_quant.columns] = df_quant_scaled
```

In [17]:
```
df
```

Out[17]:

	Area	Perimeter	Major_Axis_Length	Minor_Axis_Length	Eccentricity	Convex_Area	Extent	Class
0	1.479830	2.004354	2.348547	-0.212943	2.018337	1.499659	-1.152921	Cammeo
1	1.147870	1.125853	0.988390	0.945568	0.410018	1.192918	-0.602079	Cammeo
2	1.135169	1.317214	1.451908	0.253887	1.212956	1.126504	0.405611	Cammeo
3	0.293436	0.115300	0.261439	0.198051	0.239751	0.233857	-0.275351	Cammeo
4	1.166345	1.487053	1.316442	0.523419	0.952221	1.299855	-0.206013	Cammeo
...	...	...	...	...	...	...	...	...
3805	-0.708215	-1.078353	-1.048323	-0.097251	-1.085282	-0.745465	0.247031	Osmancik
3806	-0.601988	-0.922926	-1.207208	0.549622	-1.970731	-0.590124	0.418815	Osmancik
3807	-0.133204	-0.329851	-0.298245	0.085220	-0.275099	-0.173068	-0.455731	Osmancik
3808	-1.608257	-1.740320	-1.580971	-1.414414	-0.598821	-1.607156	-0.037168	Osmancik
3809	-0.712256	-1.391566	-1.587546	0.794972	-2.939160	-0.766290	1.825947	Osmancik

3810 rows × 8 columns

Data is normalized.

EX2

https://www.kaggle.com/datasets/nareshbhat/health-care-data-set-on-heart-attack-possibility

My dataset is about heart disease and causative factors. It includes one dependent variable and 13 independent variables. I intend to perform the following three analyses on the dataset. First descriptive analysis. The characteristics of the sample are analyzed to understand the composition structure of the sample. Bar charts, pie charts, box plots, etc. will be well suited for presenting the sample characteristics. Second, inferential analysis, uses statistical tests to determine if there is a significant relationship between the dependent and independent variables in the data set. Finally, a multivariate analysis can be performed. This may be able to find which factors are most strongly associated with heart disease and how they interact with each other. By understanding the factors that contribute to heart disease and how they interact, it can help develop strategies to prevent and treat heart disease and ultimately improve the health of individuals and populations.

I plan to include a chart as an interactive interface in the web page. It will ask the user to enter age and gender, and after submitting these two parameters, the chart will show the average values of heart disease causative parameters for a given age and gender, such as typical blood pressure, heart rate, fasting glucose, etc. for a 63-year-old female.

Charts will include bar charts, box charts, line charts, scatter charts, and heat maps

I wanted to allow users to explore the data and see how different factors relate to heart disease. The visualization is used to show the relationship between the different variables. Users can then select different variables and see how the data changes in the

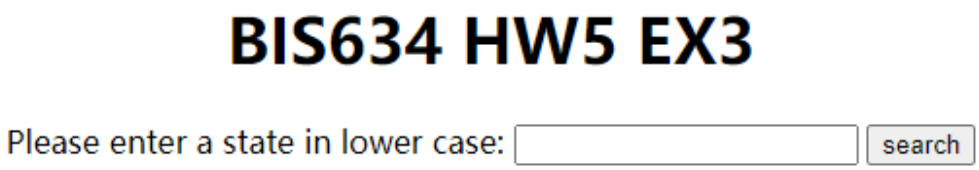
visualization. I plan to include interactive features that allow users to enter their own values for age and gender and see how it affects the likelihood of developing heart disease. This can be done using the slider.

The main challenges include how to make the charts more readable and interesting? How to reduce bias when the total sample set is small? It will be helpful to read tutorials and examples related to data visualization. I also plan to explore statistical methods suitable for smaller sample sets.

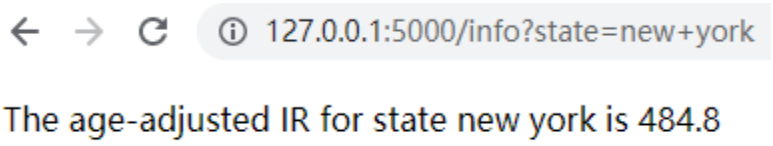
EX3

Data is from https://statecancerprofiles.cancer.gov/incidencerates/index.php

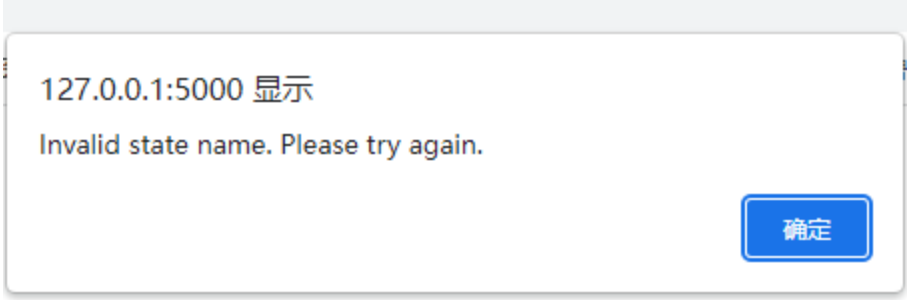
I found that the first eight rows of data and the last thirty-one rows were irrelevant information, so I skipped them when reading CSV. In addition, I removed the parentheses after the state name and the number of sources, and made the state names all lowercase. Finally, I kept only the state names and age-adjusted incidence rates and output them as a new CSV file.



The index page as above.



When the correct state name is entered, the result is displayed with the URL as above.



When an invalid state name is entered, I designed a pop-up window to increase interactivity. The pop-up will ask the user to try again and click on confirm to return to the index page. This is my extension.

Only lowercase, space-correct state name input is valid, so I require lowercase input on the index page.

Code

EX1

In [7]:
```
import pandas as pd
import sklearn
import numpy as np
import matplotlib.pyplot as plt
from sklearn.preprocessing import StandardScaler
```

In [15]:
```
df = pd.read_excel('Rice_Cammeo_Osmancik.xlsx')
df
```

Out[15]:

	Area	Perimeter	Major_Axis_Length	Minor_Axis_Length	Eccentricity	Convex_Area	Extent	Class
0	15231	525.578979	229.749878	85.093788	0.928882	15617	0.572896	Cammeo
1	14656	494.311005	206.020065	91.730972	0.895405	15072	0.615436	Cammeo
2	14634	501.122009	214.106781	87.768288	0.912118	14954	0.693259	Cammeo
3	13176	458.342987	193.337387	87.448395	0.891861	13368	0.640669	Cammeo
4	14688	507.166992	211.743378	89.312454	0.906691	15262	0.646024	Cammeo
...	...	...	...	...	...	...	...	...
3805	11441	415.858002	170.486771	85.756592	0.864280	11628	0.681012	Osmancik
3806	11625	421.390015	167.714798	89.462570	0.845850	11904	0.694279	Osmancik
3807	12437	442.498993	183.572922	86.801979	0.881144	12645	0.626739	Osmancik
3808	9882	392.296997	161.193985	78.210480	0.874406	10097	0.659064	Osmancik
3809	11434	404.709991	161.079269	90.868195	0.825692	11591	0.802949	Osmancik

3810 rows × 8 columns

In [16]:
```
df = pd.read_excel('Rice_Cammeo_Osmancik.xlsx')
df_quant = df.select_dtypes(['int64', 'float64'])
scaler = StandardScaler()
scaler.fit(df_quant)
df_quant_scaled = scaler.transform(df_quant)
df[df_quant.columns] = df_quant_scaled
```

In [17]:
```
df
```

Out[17]:

	Area	Perimeter	Major_Axis_Length	Minor_Axis_Length	Eccentricity	Convex_Area	Extent	Class
0	1.479830	2.004354	2.348547	-0.212943	2.018337	1.499659	-1.152921	Cammeo
1	1.147870	1.125853	0.988390	0.945568	0.410018	1.192918	-0.602079	Cammeo
2	1.135169	1.317214	1.451908	0.253887	1.212956	1.126504	0.405611	Cammeo
3	0.293436	0.115300	0.261439	0.198051	0.239751	0.233857	-0.275351	Cammeo
4	1.166345	1.487053	1.316442	0.523419	0.952221	1.299855	-0.206013	Cammeo
...	...	...	...	...	...	...	...	...
3805	-0.708215	-1.078353	-1.048323	-0.097251	-1.085282	-0.745465	0.247031	Osmancik
3806	-0.601988	-0.922926	-1.207208	0.549622	-1.970731	-0.590124	0.418815	Osmancik
3807	-0.133204	-0.329851	-0.298245	0.085220	-0.275099	-0.173068	-0.455731	Osmancik
3808	-1.608257	-1.740320	-1.580971	-1.414414	-0.598821	-1.607156	-0.037168	Osmancik
3809	-0.712256	-1.391566	-1.587546	0.794972	-2.939160	-0.766290	1.825947	Osmancik

3810 rows × 8 columns

In [10]:

```
class QuadTree:
    def __init__(self, data, bounding_box=None, max_leaf_data=3):
        if bounding_box is None:
            xs, ys, classes = zip(*data)
            self.xlo = min(xs)
            self.ylo = min(ys)
            self.xhi = max(xs)
            self.yhi = max(ys)
        else:
            self.xlo = bounding_box['xlo']
            self.xhi = bounding_box['xhi']
            self.ylo = bounding_box['ylo']
            self.yhi = bounding_box['yhi']
        if len(data) <= max_leaf_data:
            self._data = data
            self.children = []
        else:
            self._data = None
            self.children = []
            xsplit = (self.xlo + self.xhi) / 2
            ysplit = (self.ylo + self.yhi) / 2
            bbox = [
                {'xlo': self.xlo, 'xhi': xsplit, 'ylo': self.ylo, 'yhi': ysplit},
                {'xlo': self.xlo, 'xhi': xsplit, 'ylo': ysplit, 'yhi': self.yhi},
                {'xlo': xsplit, 'xhi': self.xhi, 'ylo': self.ylo, 'yhi': ysplit},
                {'xlo': xsplit, 'xhi': self.xhi, 'ylo': ysplit, 'yhi': self.yhi}
            ]
            self.children = [
                QuadTree(get_data_in_range(data, my_bbox), my_bbox, max_leaf_data)
                for my_bbox in bbox
            ]

    def get_descendant_count(self):
        if not self.children:
            return len(self._data)
        else:
            return sum(child.get_descendant_count() for child in self.children)

    def __repr__(self):
        return f'<QuadTree xlo={self.xlo} ylo={self.ylo} xhi={self.xhi} yhi={self.yhi} #desc={self.get_descendant_count()}>'


    def get_data_in_range(data, bbox):
        result = []
        for x, y, condition in data:
            if bbox['xlo'] <= x <= bbox['xhi'] and bbox['ylo'] <= y <= bbox['yhi']:
                result.append([x, y, condition])
        return result
```

EX3

In [1]:
```
from flask import Flask, render_template, request
import pandas as pd
import numpy as np
import csv
import json
```

In [2]:
```
df=pd.read_csv('incd.csv',skiprows=8,skipfooter=31,engine="python")
```

In [3]:
```
df
```

Out[3]:

	State	FIPS	Age-Adjusted Incidence Rate([rate note]) - cases per 100,000	Lower 95% Confidence Interval	Upper 95% Confidence Interval	CI*Rank([rank note])	Lower CI (CI*Rank)	Upper CI (CI*Rank)	Average Annual Count	Recent Trend	Recent 5-Year Trend ([trend note]) in Incidence Rates	Lower 95% Confidence Interval.1	Upper 95% Confidence Interval.1
0	US (SEER+NPCR) (1)	0	449.4	449.1	449.7	N/A	N/A	N/A	1728431	stable	-0.9	-2.0	0.2
1	Kentucky(7)	21000	516	513.2	518.8	1	1	1	27998	falling	-0.9	-1.8	-0.1
2	Iowa(7)	19000	490.7	487.5	494	2	2	5	19110	rising	0.8	0.4	1.2
3	New Jersey(7)	34000	488.9	487	490.8	3	2	5	53473	falling	-0.6	-0.7	-0.5
4	West Virginia(6)	54000	487.4	483.3	491.4	4	2	8	12216	falling	-0.2	-0.4	-0.1
5	New York(7)	36000	484.8	483.6	486.1	5	4	8	116044	falling	-0.6	-0.8	-0.4
6	Louisiana(7)	22000	484.3	481.7	487	6	3	9	26426	stable	0.5	-0.3	1.3

In [4]:
```
df['State']=df['State'].str.extract(r'.*?([A-Za-z ]+).*?', expand=True)
```

In [5]:
```
df
```

Out[5]:

	State	FIPS	Age-Adjusted Incidence Rate([rate note]) - cases per 100,000	Lower 95% Confidence Interval	Upper 95% Confidence Interval	CI*Rank([rank note])	Lower CI (CI*Rank)	Upper CI (CI*Rank)	Average Annual Count	Recent Trend	Recent 5-Year Trend ([trend note]) in Incidence Rates	Lower 95% Confidence Interval.1	Upper 95% Confidence Interval.1
0	US	0	449.4	449.1	449.7	N/A	N/A	N/A	1728431	stable	-0.9	-2.0	0.2
1	Kentucky	21000	516	513.2	518.8	1	1	1	27998	falling	-0.9	-1.8	-0.1
2	Iowa	19000	490.7	487.5	494	2	2	5	19110	rising	0.8	0.4	1.2
3	New Jersey	34000	488.9	487	490.8	3	2	5	53473	falling	-0.6	-0.7	-0.5
4	West Virginia	54000	487.4	483.3	491.4	4	2	8	12216	falling	-0.2	-0.4	-0.1
5	New York	36000	484.8	483.6	486.1	5	4	8	116044	falling	-0.6	-0.8	-0.4
6	Louisiana	22000	484.3	481.7	487	6	3	9	26426	stable	0.5	-0.3	1.3
7	Arkansas	5000	483.6	480.4	486.9	7	3	9	17906	stable	0.4	-0.5	1.4

In [6]:
```
df.to_csv(r'cleaned_incd.csv', index = False)
```

In [7]:
```
df=pd.read_csv('cleaned_incd.csv',usecols=['State','Age-Adjusted Incidence Rate([rate note]) - cases per 100,000'])
df.columns=['State','Age-adjusted IR']
df['State']=df['State'].str.lower()
df.to_csv(r'incd_webdata.csv', index = False)
df
```

Out[7]:

	State	Age-adjusted IR
0	us	449.4
1	kentucky	516
2	iowa	490.7
3	new jersey	488.9
4	west virginia	487.4
5	new york	484.8
6	louisiana	484.3
7	arkansas	483.6
8	new hampshire	482.9
9	pennsylvania	476.8
10	maine	476.7
11	rhode island	476.2

In [8]:
```
app = Flask(__name__)

with open("incd_webdata.csv") as csvfile:
    reader = csv.reader(csvfile)
    # create a dictionary for state names
    state_values = {row[0]: (row[1]) for row in reader}

@app.route('/')
def index():
    #index page with input box and button
    return """
    <center>
    <h1> BIS634 HW5 EX3 </h1>
    <p>
    <form method="get" action="/info">
    Please enter a state: <input type="text" name="state"> </input> <input type="submit" value="search"> </input>
    </p>
    </center>
    """

@app.route("/info", methods=["GET"])
def info():
    # get the name of the state
    name = request.args.get("state", None)

    if name in state_values:
        return f"The age-adjusted IR for state {name} is {state_values[name]}"

    else:
        return """
        <center>
        <h1> State name is not valid </h1>
        <p> Please input a valid state name. </p>
        <script>
        window.setTimeout(function() {
            alert("Invalid state name. Please try again.");
            window.location.href = "/";
        }, 1000);
        </script>
        </center>
        """

@app.route("/state/<string:name>")
def state(name):
    if name in state_values:
        return jsonify(state=name, age_adjusted_incidence_rate=state_values[name])
    else:
        return jsonify(error=f"{name} is not a valid state name.")

app.run()
```

```
 * Serving Flask app "__main__" (lazy loading)
 * Environment: production
   WARNING: This is a development server. Do not use it in a production deployment.
   Use a production WSGI server instead.
 * Debug mode: off
 * Running on http://127.0.0.1:5000/ (Press CTRL+C to quit)
127.0.0.1 - - [09/Dec/2022 22:54:59] "GET / HTTP/1.1" 200 -
127.0.0.1 - - [09/Dec/2022 22:55:02] "GET /info?state=sdf HTTP/1.1" 200 -
127.0.0.1 - - [09/Dec/2022 22:55:08] "GET / HTTP/1.1" 200 -
127.0.0.1 - - [09/Dec/2022 22:55:12] "GET /info?state=ohio HTTP/1.1" 200 -
127.0.0.1 - - [09/Dec/2022 22:55:19] "GET /info?state=washignton HTTP/1.1" 200 -
127.0.0.1 - - [09/Dec/2022 22:55:21] "GET / HTTP/1.1" 200 -
127.0.0.1 - - [09/Dec/2022 22:55:26] "GET /info?state=washington HTTP/1.1" 200 -
```