**Youth Tobacco Survey Analysis**

**Arshita Dixit - 305832363**

**Roshik Ganesan - 305641224**

**California State University Los Angeles**

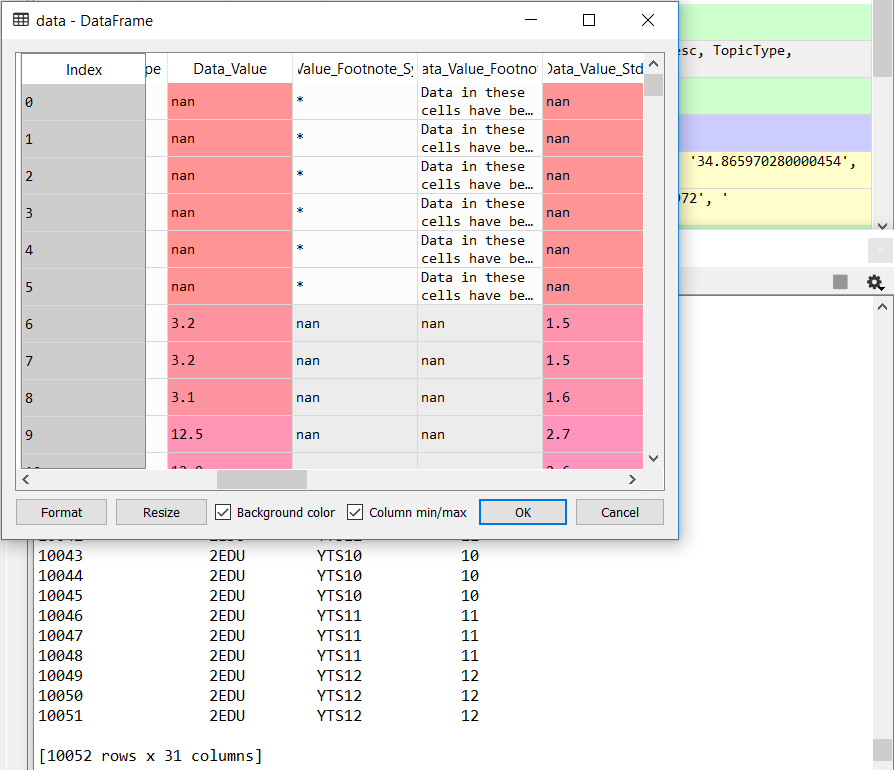
**Dataset URL:**

<https://catalog.data.gov/dataset/youth-tobacco-survey-yts-data>

**Dataset Description:**

**Data Cleaning 1:**

Pre-Implementation:



"""

Data Cleaning 1

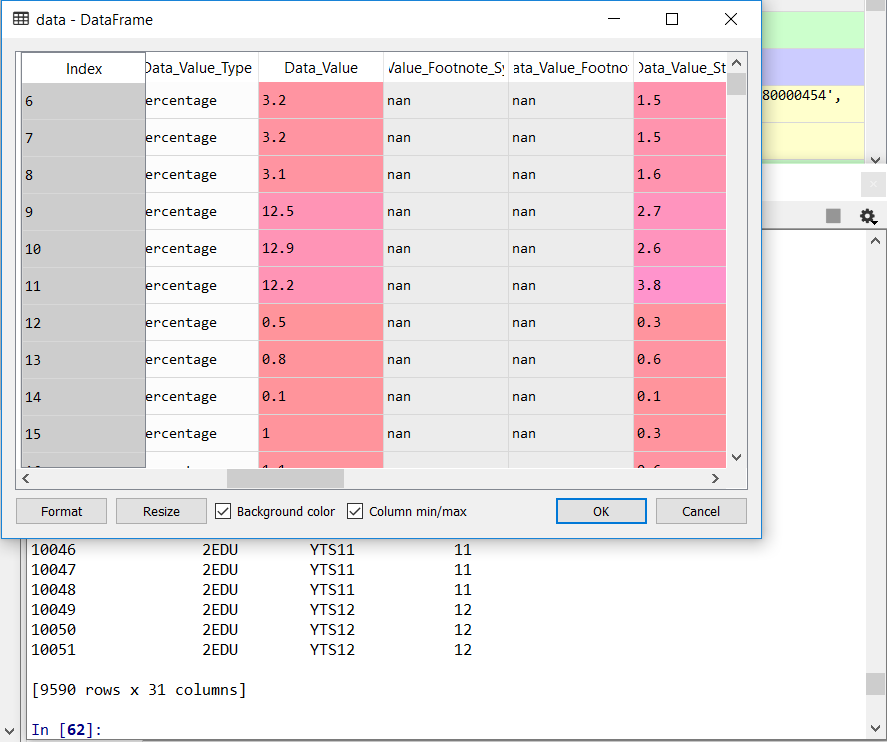
Misfed Rows cleanup.

Removes rows where there is improper data value

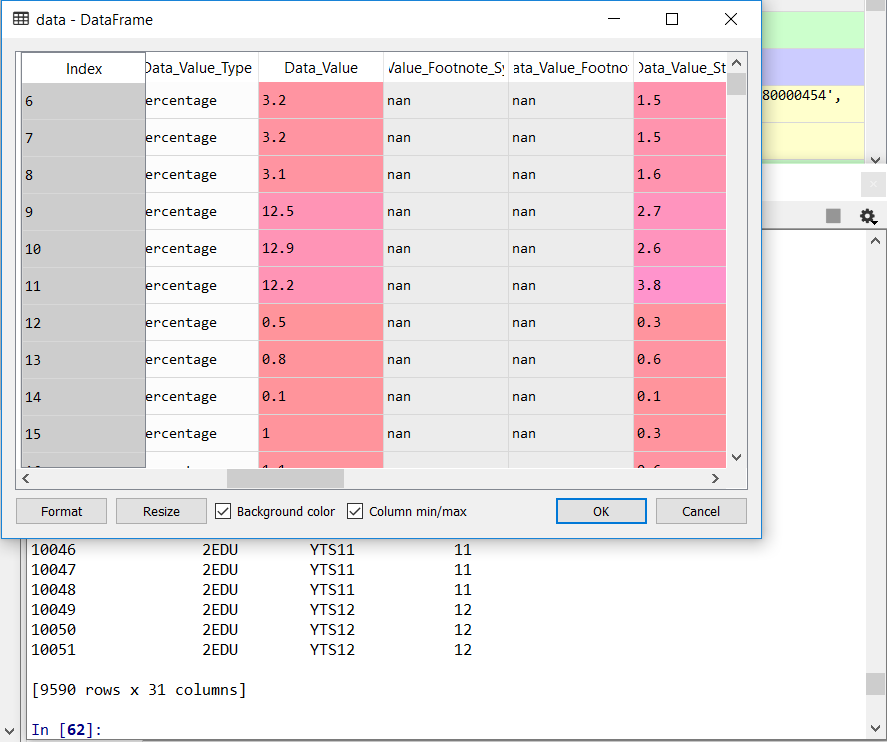
"""

data = data[data.Data\_Value\_Footnote\_Symbol != "\*"]

Post Implementation:



**Data Cleaning 2:**



"""

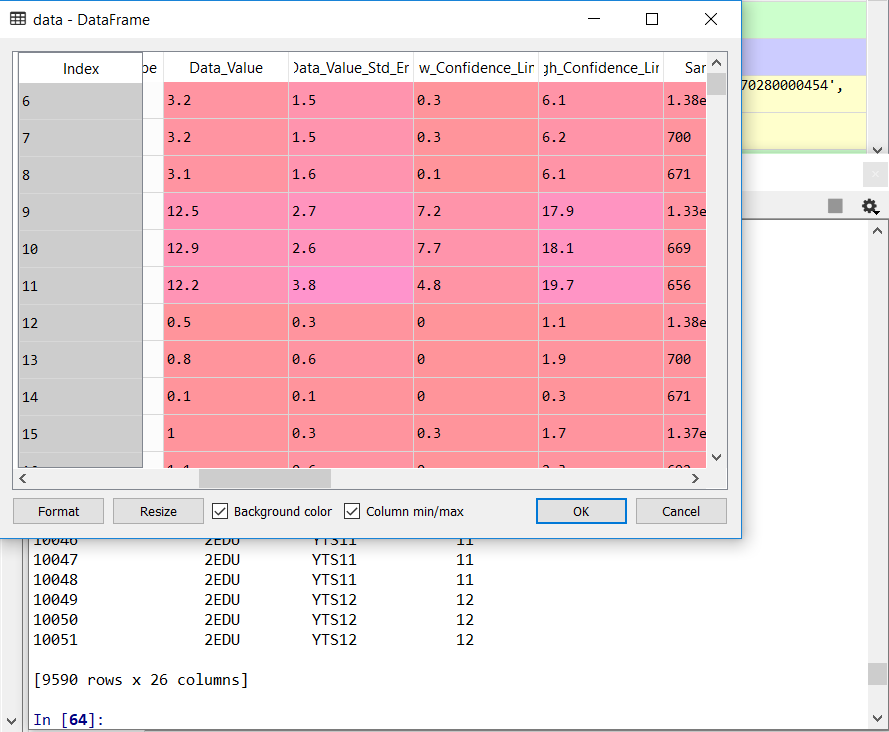
Data Cleaning 2

Removing Unwanted Columns.

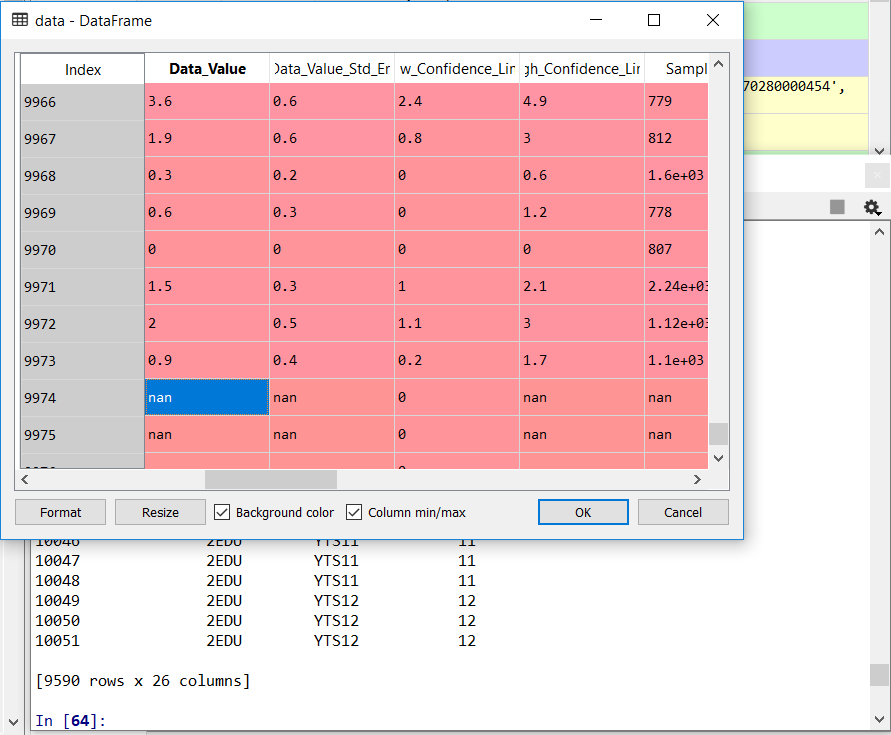
"""

data = data.drop(['Data\_Value\_Footnote','DataSource', 'Data\_Value\_Unit','TopicType','Data\_Value\_Footnote','Data\_Value\_Footnote\_Symbol'],axis = 1)

Post Implementation:



**Data Cleaning 3:**



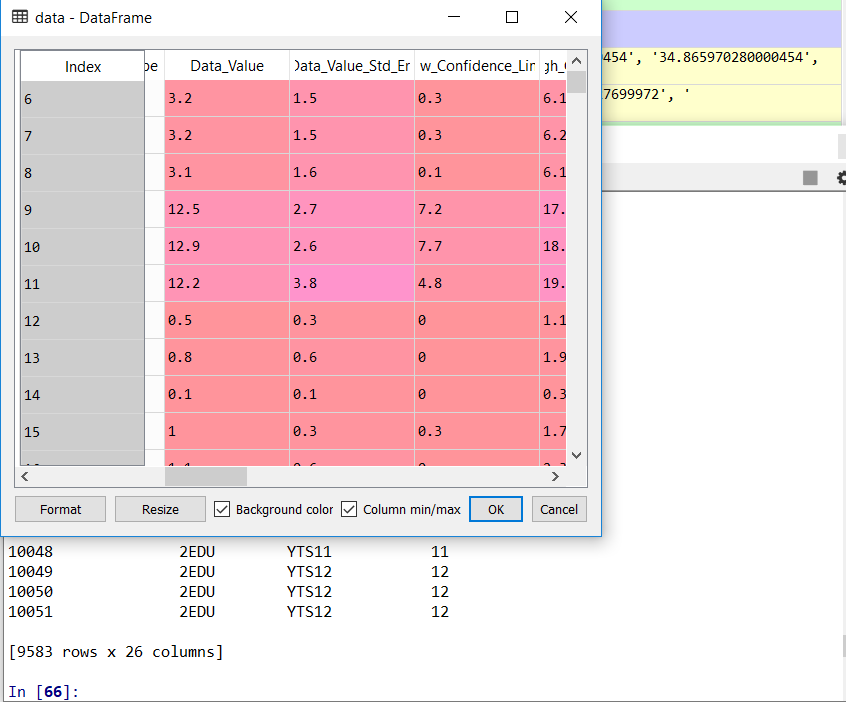
“””

Data Cleaning 3

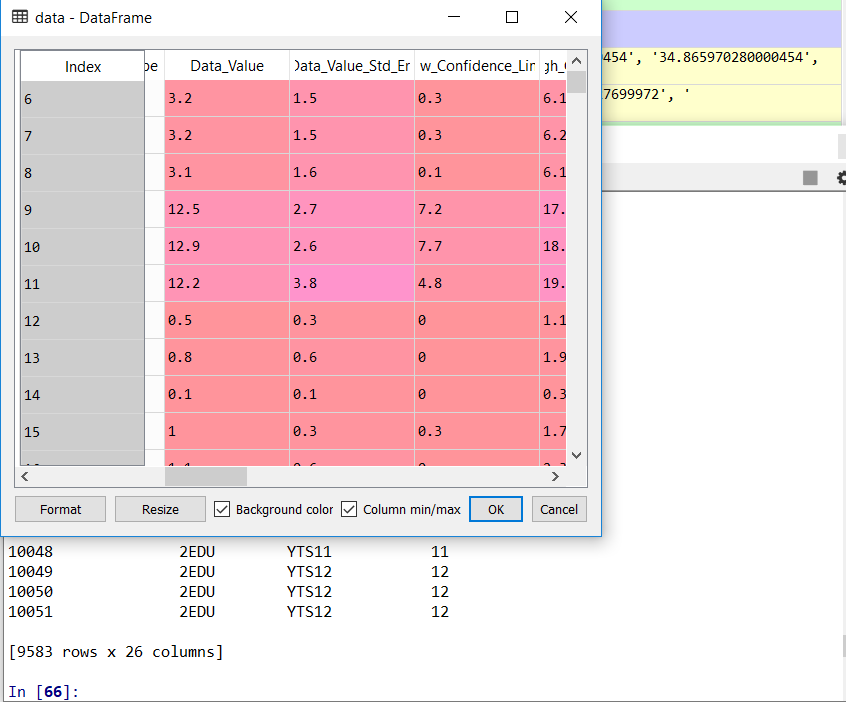
Checking and Removing Null values

"""

data = data.dropna(subset=['Data\_Value','GeoLocation','Sample\_Size'])



**Data Cleaning 4:**



"""

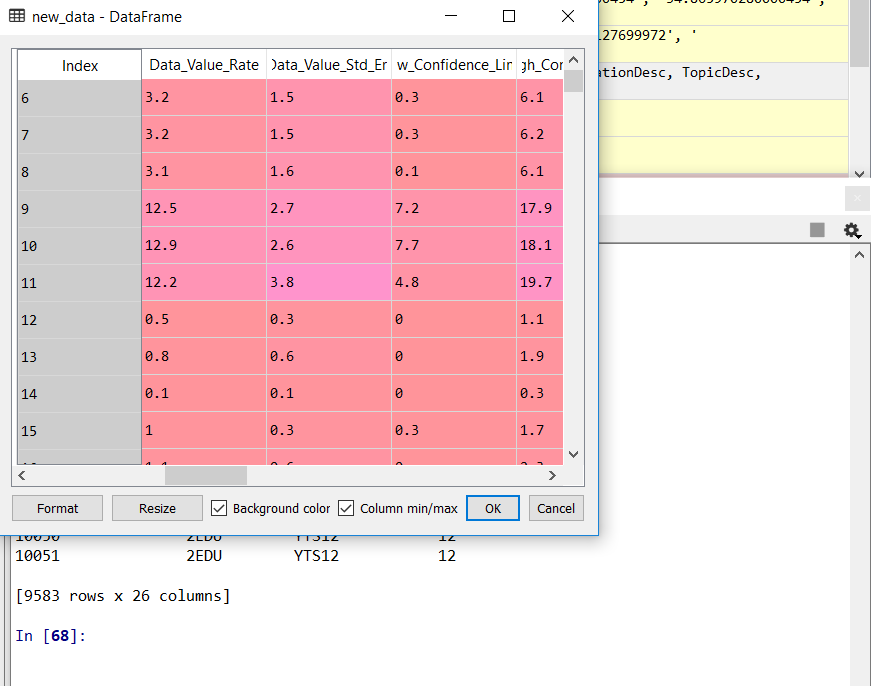
Data Cleaning 4

Changing names of Improperly Named columns.

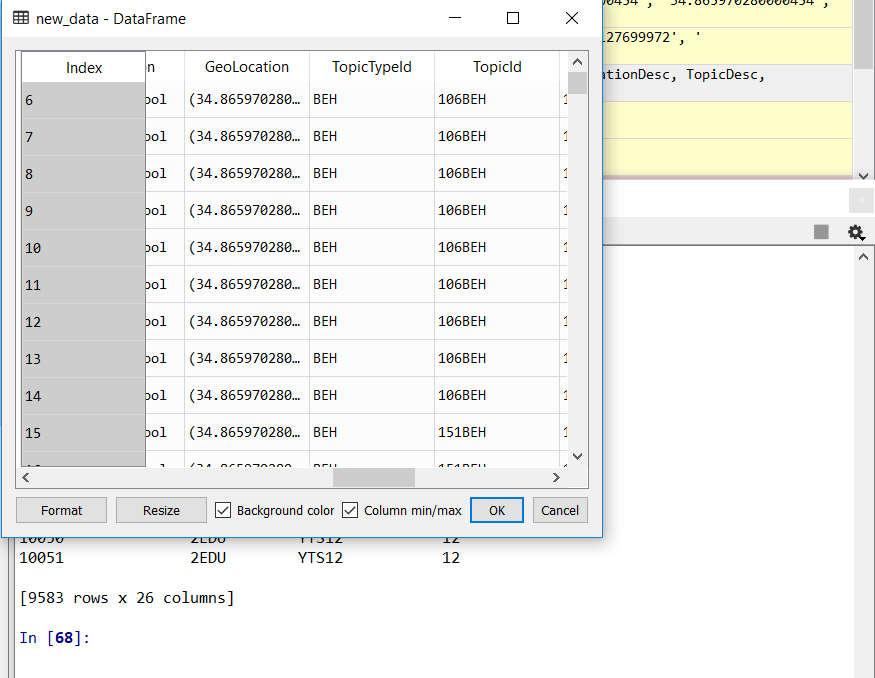
"""

new\_data = data.rename(columns = {'Data\_Value':'Data\_Value\_Rate'})

Post Implementation:



**Data Cleaning 5: Operations Used – List String Split, Strip, NumPy and File Operation Used**



"""

Data Cleaning 5

Splitting Latitude and Longitute into 2 columns.

String Operations Used to Strip

List Operations Used

"""

lat = []

lon = []

new\_data['GeoLocation'] = new\_data['GeoLocation'].str.strip('()')

for row in new\_data['GeoLocation']:

# Try to,

try:

# Split the row by comma and append

# everything before the comma to lat

lat.append(row.split(',')[0])

# Split the row by comma and append

# everything after the comma to lon

lon.append(row.split(',')[1])

# But if you get an error

except:

# append a missing value to lat

lat.append(np.NaN)

# append a missing value to lon

lon.append(np.NaN)

new\_data['latitude'] = lat

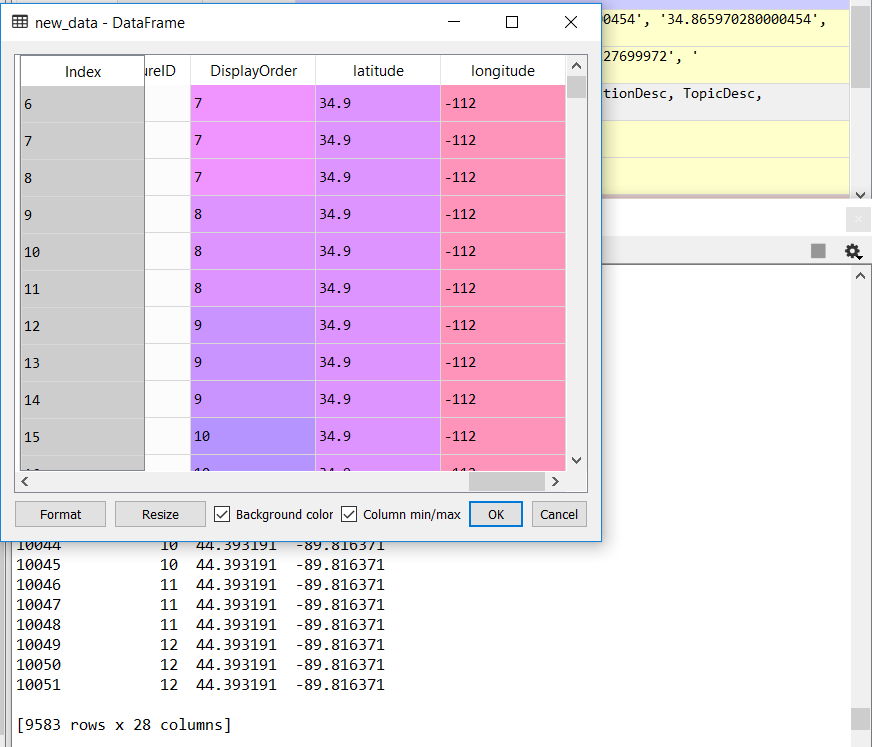
new\_data['latitude'] = new\_data['latitude'].astype(float)

new\_data['longitude'] = lon

new\_data['longitude'] = new\_data['longitude'].astype(float)

new\_data.to\_csv('./data/youth\_tobacco\_analysis.csv', sep = ',', encoding='utf-8')

Post Implementation:



**Question 1: How does Smoking status compare based on states?**

**Operation Used - List Operation - Functions - PandasDataFrame - Dict Used Operations Implemented**

Code:

import pandas as pd

import matplotlib.pyplot as plt

import dash

import dash\_core\_components as dcc

import dash\_html\_components as html

import plotly.graph\_objs as go

from app import app

data = pd.read\_csv('./data/youth\_tobacco\_analysis.csv')

"""Pandas DataFrame Implemented"""

final\_data = pd.DataFrame(data.groupby(['LocationDesc','MeasureDesc']).count())

final\_data.to\_csv('./data/question1.csv', sep = ',', encoding='utf-8')

qn1data = pd.read\_csv('./data/question1.csv')

state\_names = list(qn1data['LocationDesc'].unique())

layout = html.Div(children=[

html.Div([

dcc.Dropdown(

id='state\_names',

options=[{'label': i, 'value': i} for i in state\_names],

value='Arizona'

),

dcc.Dropdown(

id='state\_names2',

options=[{'label': i, 'value': i} for i in state\_names],

value='Connecticut'

),

],

style={'width': '30%', 'display': 'inline-block'}),

html.Div([

dcc.Graph(id='simple-bar'),

],style={ 'width': '49%'}),

])

@app.callback(

dash.dependencies.Output('simple-bar', 'figure'),

[dash.dependencies.Input('state\_names', 'value'),

dash.dependencies.Input('state\_names2', 'value')])

def update\_bar\_chart(statename1,statename2):

"""

Forms a Stacked Bar Chart

Keyword Arguments:

statename1 -- Gets the first state name to compare

statename2 -- Gets the second state name to compare

The values of the states are fetched and compared using a stacked Bar chart

"""

value\_list = list(qn1data['YEAR'][(qn1data['LocationDesc'] == statename1)])

name\_list = list(qn1data['MeasureDesc'][(qn1data['LocationDesc'] == statename1)])

value\_list2 = list(qn1data['YEAR'][(qn1data['LocationDesc'] == statename2)])

name\_list2 = list(qn1data['MeasureDesc'][(qn1data['LocationDesc'] == statename2)])

return {

'data': ([

"""Dict Implemented""",

{'x': name\_list, 'y': value\_list, 'type': 'bar', 'name': statename1},

{'x': name\_list2, 'y': value\_list2, 'type': 'bar', 'name': statename2},

]),

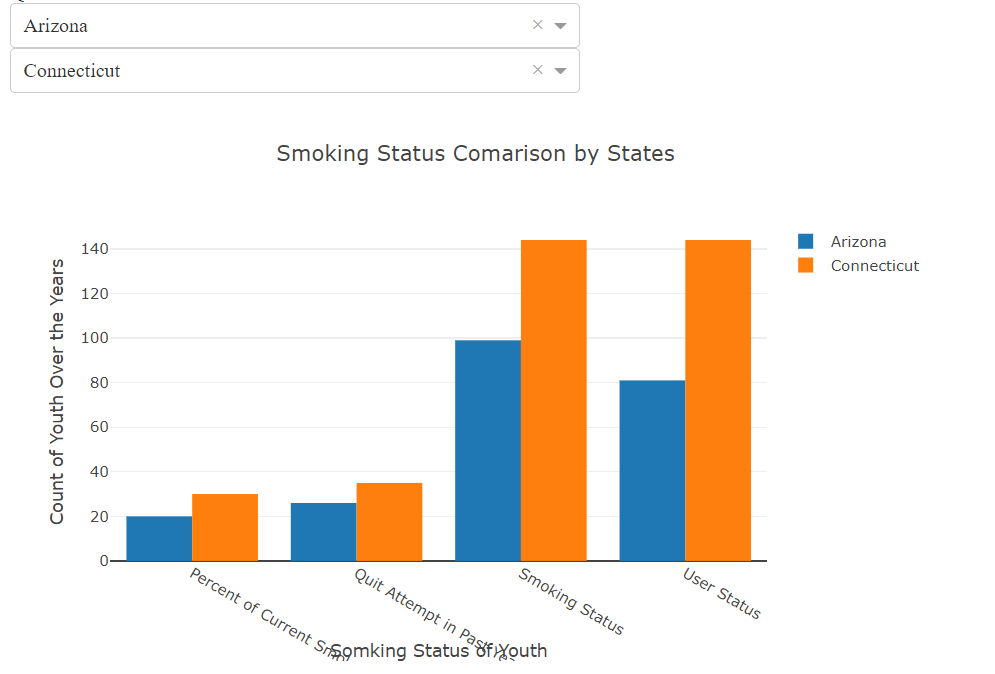
'layout': go.Layout(

title = "Smoking Status Comarison by States",

xaxis={'title': 'Somking Status of Youth'},

yaxis={'title': 'Count of Youth Over the Years'}),

}



**Question 2: How is the smoking trend of each states over the years?**

**Operation Used - Strings**

import pandas as pd

import matplotlib.pyplot as plt

import dash

import dash\_core\_components as dcc

import dash\_html\_components as html

import plotly.graph\_objs as go

from app import app

data = pd.read\_csv('./data/youth\_tobacco\_analysis.csv')

"""Pandas DataFrame Implemented"""

final\_data = pd.DataFrame(data.groupby(['YEAR','LocationDesc']).count())

final\_data.to\_csv('./data/question2.csv', sep = ',', encoding='utf-8')

qn2data = pd.read\_csv('./data/question2.csv')

qn2data['LocationDesc'] = qn2data['LocationDesc'].str.upper()

x=0

state\_names = list(qn2data['LocationDesc'].unique())

"""

String Operation to Convert the string in each column to upper case is used.

It is used as the columns names are inconsistent with the casing

"""

for i in state\_names:

state\_names[x] = state\_names[x].upper()

x=x+1

years = list(qn2data['YEAR'].unique())

layout = html.Div(children=[

html.Div([

dcc.Dropdown(

id='state\_names',

options=[{'label': i, 'value': i} for i in state\_names],

value='ARIZONA'

),

],

style={'width': '30%', 'display': 'inline-block'}),

html.Div([

dcc.Graph(id='line-chart'),

],style={'width': '49%'}),

])

@app.callback(

dash.dependencies.Output('line-chart', 'figure'),

[dash.dependencies.Input('state\_names', 'value')])

def update\_bar\_chart(statename1):

"""

Forms a Line Chart

Keyword Arguments:

statename1 -- Gets the first state name to compare

The values of the states are fetched and compared using a line chart for the trend analysis

Functions - PandasDataFrame Operations Implemented

"""

value\_list = list(qn2data['LocationAbbr'][(qn2data['LocationDesc'] == statename1)])

xvalues = list(qn2data['YEAR'][(qn2data['LocationDesc'] == statename1)])

return {

'data': ([

{'x':xvalues , 'y': value\_list, 'type': 'line', 'name': 'NB'},

]),

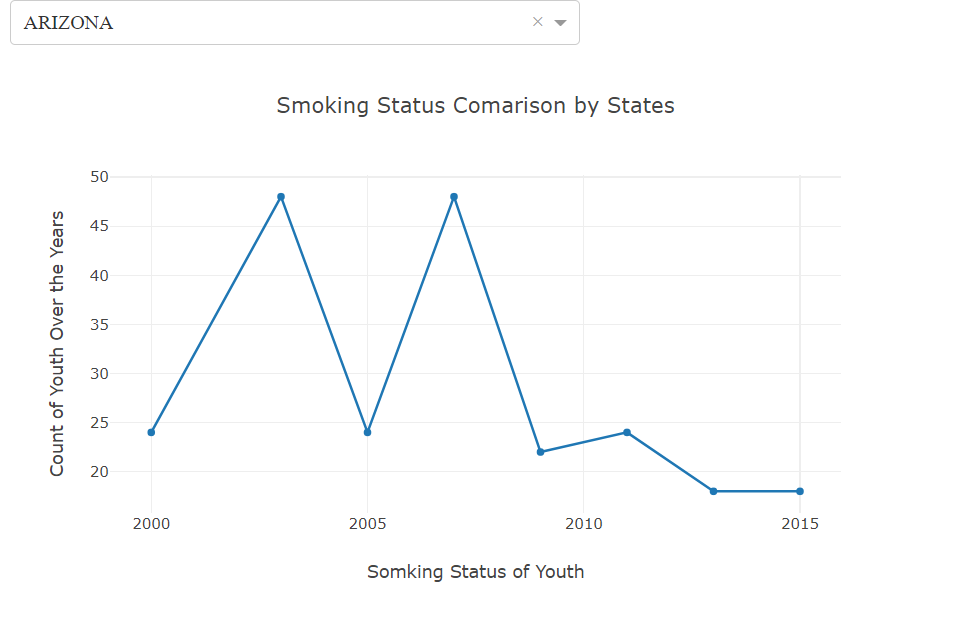
'layout': go.Layout(

title = "Smoking Status Comarison by States",

xaxis={'title': 'Somking Status of Youth'},

yaxis={'title': 'Count of Youth Over the Years'}),

}



**Question 3: Which school level has attracted the most number of youth smokers?**

**Operation Used - Tuples**

import pandas as pd

import matplotlib.pyplot as plt

import dash

import dash\_core\_components as dcc

import dash\_html\_components as html

import plotly.graph\_objs as go

from app import app

data = pd.read\_csv('./data/youth\_tobacco\_analysis.csv')

"""Pandas DataFrame Implemented"""

final\_data = pd.DataFrame(data.groupby(['YEAR','Education']).count())

final\_data.to\_csv('./data/question3.csv', sep = ',', encoding='utf-8')

qn3data = pd.read\_csv('./data/question3.csv')

years = list(qn3data['YEAR'].unique())

layout = html.Div(children=[

html.Div([

dcc.Dropdown(

id='years',

options=[{'label': i, 'value': i} for i in years],

value=2000

),

],

style={'width': '30%', 'display': 'inline-block'}),

html.Div([

dcc.Graph(id='pie-chart'),

],style={'width': '49%'}),

])

@app.callback(

dash.dependencies.Output('pie-chart', 'figure'),

[dash.dependencies.Input('years', 'value')])

def update\_bar\_chart(yearvalue):

"""

Forms a Pie Chart

Keyword Arguments:

yearvalue -- Gets the value of year from the user

A pie chart to display the ratio between high school and middle school smokers are displayed

Functions - Tuples Implemented

"""

title\_pie\_Chart = ("Middle School - High School Comparison")

labels\_pie\_chart = ("High School","Middle School")

value\_1 = qn3data['LocationAbbr'][(qn3data['YEAR'] == yearvalue) & (qn3data['Education'] == "High School") ]

value1 = int(value\_1)

value\_2 = qn3data['LocationAbbr'][(qn3data['YEAR'] == yearvalue) & (qn3data['Education'] == "Middle School") ]

value2 = int(value\_2)

return {

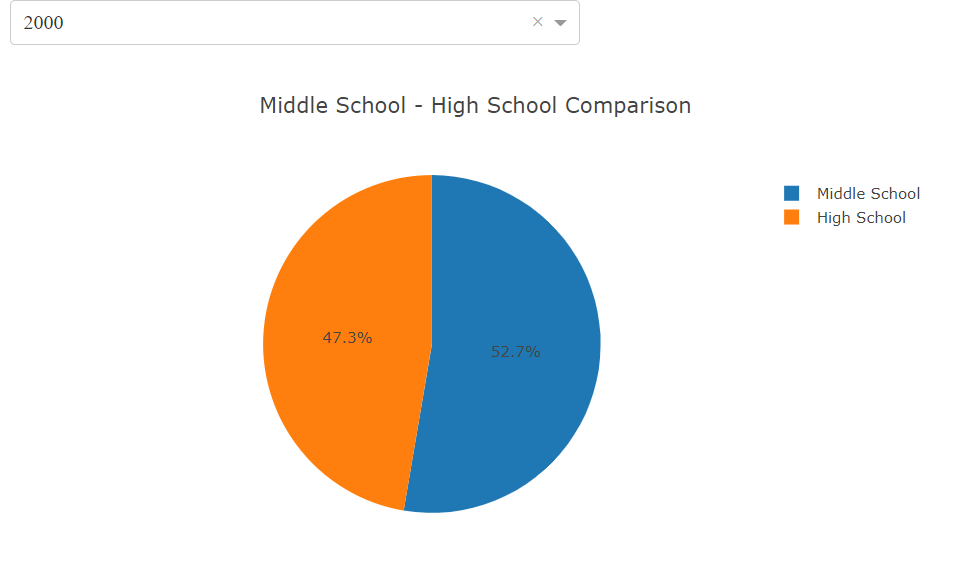
'data': [go.Pie(labels=labels\_pie\_chart, values=[value1,value2])],

'layout': {

'title': title\_pie\_Chart

}

}



**Question 4: How does each gender group compare to the overall smoking count?**

import pandas as pd

import matplotlib.pyplot as plt

import dash

import dash\_core\_components as dcc

import dash\_html\_components as html

import plotly.graph\_objs as go

from app import app

data = pd.read\_csv('./data/youth\_tobacco\_analysis.csv')

"""Pandas DataFrame Implemented"""

final\_data = (data.groupby(['YEAR','Gender'])['Sample\_Size'].agg(['sum','count']))

final\_data.to\_csv('./data/question4.csv', sep = ',', encoding='utf-8')

qn1data = pd.read\_csv('./data/question4.csv')

layout = html.Div(children=[

html.Div([

dcc.RadioItems(

id='gender-type',

options=[{'label': i, 'value': i} for i in ['Male', 'Female']],

value='Male',

labelStyle={'display': 'inline-block'}

)

],

style={'width': '100%','align':'centre'}),

html.Div([

dcc.Graph(id='bar-line-chart'),

],style={'display': 'full', 'width': '100%'}),

])

@app.callback(

dash.dependencies.Output('bar-line-chart', 'figure'),

[dash.dependencies.Input('gender-type', 'value')])

def update\_bar\_chart(gendername):

"""

Forms a Bar titled Line Chart

Keyword Arguments:

gendername = to fetch input of the gender to be compared from the user

A Line chart in comparison to the bar chart with overall comparison is displayed

List Operation - Functions - PandasDataFrame Operations Implemented

"""

value\_list = list(qn1data['sum'][(qn1data['Gender'] == gendername)])

name\_list = list(qn1data['YEAR'][(qn1data['Gender'] == gendername)])

overall\_value = list(qn1data['sum'][(qn1data['Gender'] == "Overall")])

return {

'data': ([

{'x': name\_list, 'y': overall\_value, 'type': 'bar', 'name': 'Overall'},

{'x': name\_list, 'y': value\_list, 'type': 'line', 'name': gendername},

]),

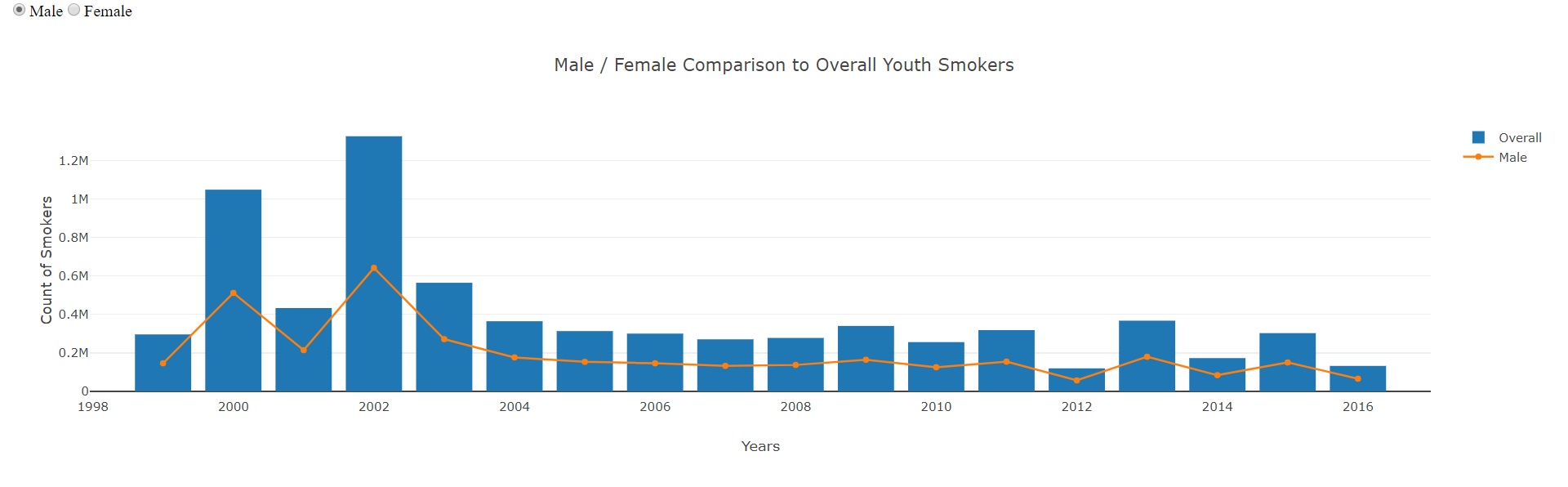
'layout': go.Layout(

title = "Male / Female Comparison to Overall Youth Smokers",

xaxis={'title': 'Years'},

yaxis={'title': 'Count of Smokers'}),

}



**Question 5: What is the trend of smoking comparison between years?**

import pandas as pd

import matplotlib.pyplot as plt

import dash

import dash\_core\_components as dcc

import dash\_html\_components as html

import plotly.graph\_objs as go

from app import app

data = pd.read\_csv('./data/youth\_tobacco\_analysis.csv')

"""Pandas DataFrame Implemented"""

final\_data = (data.groupby(['YEAR','LocationDesc'])['Sample\_Size'].agg(['sum']))

final\_data.to\_csv('./data/question5.csv', sep = ',', encoding='utf-8')

qn5data = pd.read\_csv('./data/question5.csv')

years = list(qn5data['YEAR'].unique())

state\_names = list(qn5data['LocationDesc'].unique())

layout = html.Div(children=[

html.Div([

dcc.Dropdown(

id='years1',

options=[{'label': i, 'value': i} for i in years],

value=1999

),

dcc.Dropdown(

id='years2',

options=[{'label': i, 'value': i} for i in years],

value=2013

),

],

style={'margin-top': '20','width':'70%'}

),

html.Div([

dcc.Graph(id='last-line-chart'),

],style={'display': 'full', 'width': '100%'}),

])

@app.callback(

dash.dependencies.Output('last-line-chart', 'figure'),

[dash.dependencies.Input('years1', 'value'),

dash.dependencies.Input('years2', 'value')])

def update\_bar\_chart(yearvalue1,yearvalue2):

"""

Forms a Stacked Bar Chart

Keyword Arguments:

Yearvalue1 -- Gets the first year to compare

Yearvalue2 -- Gets the second second to compare

The values of the years are fetched from the user and then traced into line chart for the tend comparison between years

List Operation - Functions - PandasDataFrame Operations Implemented

"""

# new\_list = []

# x = 0

# for i in qn5data['LocationDesc']:

# if i == "Alabama":

# new\_list.append(qn5data['sum'])

value\_list1 = list(qn5data['sum'][(qn5data['YEAR'] == yearvalue1)])

print(value\_list1)

value\_list2 = list(qn5data['sum'][(qn5data['YEAR'] == yearvalue2)])

name\_list1 = list(qn5data['LocationDesc'][(qn5data['YEAR'] == yearvalue1)])

name\_list2 = list(qn5data['LocationDesc'][(qn5data['YEAR'] == yearvalue2)])

return {

'data': ([

{'x': name\_list1, 'y': value\_list1, 'type': 'line', 'name': yearvalue1},

{'x': name\_list1, 'y': value\_list2, 'type': 'line', 'name': yearvalue2},

]),

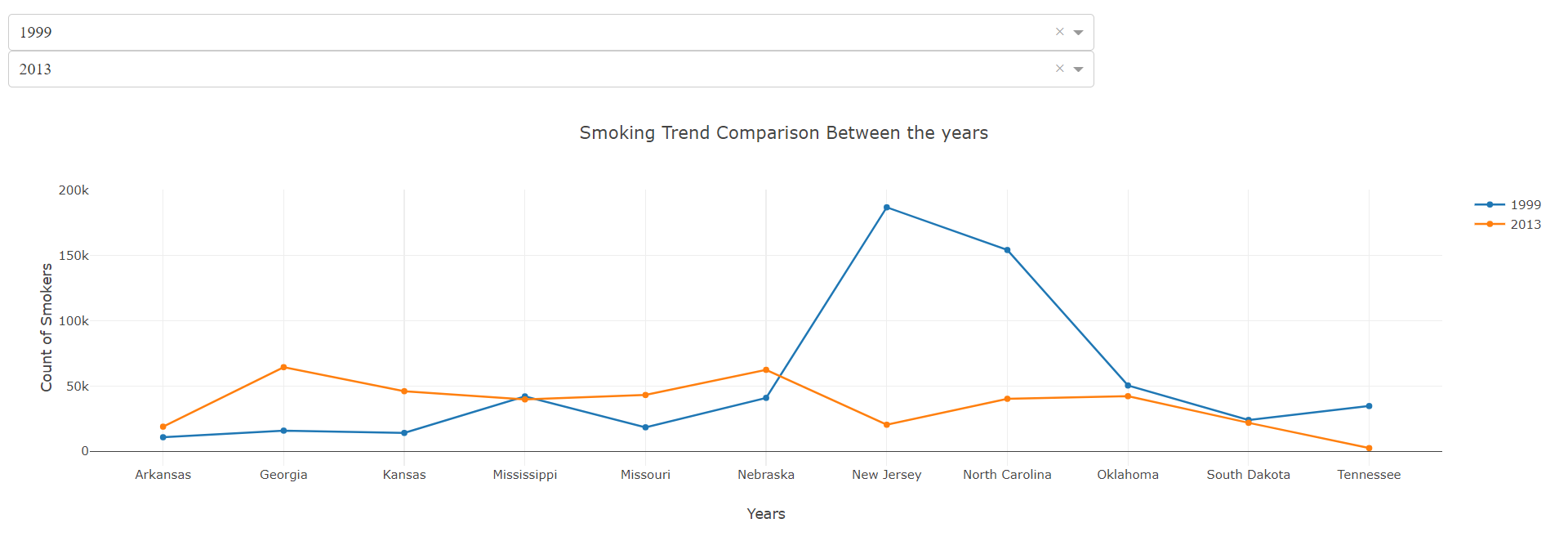
'layout': go.Layout(

title = "Smoking Trend Comparison Between the years",

xaxis={'title': 'Years'},

yaxis={'title': 'Count of Smokers'}),

}



**Question 6: How is the low confidence and high confidence value regressed?**

import pandas as pd

import matplotlib.pyplot as plt

import dash

import dash\_core\_components as dcc

import dash\_html\_components as html

import plotly.graph\_objs as go

from app import app

data = pd.read\_csv('./data/youth\_tobacco\_analysis.csv')

"""Pandas DataFrame Implemented"""

years = list(data['YEAR'].unique())

state\_names = list(data['LocationDesc'].unique())

#app = dash.Dash()

layout = html.Div(

[

html.Div([

dcc.Dropdown(

id='years1',

options=[{'label': i, 'value': i} for i in years],

value=2000

),

],style={'width': '20%', 'display': 'inline-block'}),

html.Div([

dcc.Dropdown(

id='state\_names',

value= "Arizona",

)],style={'width': '20%', 'display': 'inline-block'}),

html.Div(id='table\_chart'),

html.Div([

dcc.Graph(id='scatter-chart'),

],style={'display': 'full', 'width': '49%'}),

])

@app.callback(

dash.dependencies.Output('state\_names', 'options'),

[dash.dependencies.Input('years1', 'value')])

def update\_date\_dropdown(yearvalue):

"""

Populates the second dropdown

Keyword Arguments:

yearvalue -- Gets the value of the year to populate the child dropdown

The specific values of each year is populated based on the year selected by the user

"""

xvalue\_list = data['LocationDesc'][(data['YEAR'] == yearvalue)]

xvalue\_list = xvalue\_list.unique()

return [{'label': i, 'value': i} for i in xvalue\_list]

@app.callback(

dash.dependencies.Output('scatter-chart', 'figure'),

[dash.dependencies.Input('years1', 'value'),

dash.dependencies.Input('state\_names', 'value')])

def update\_figure(yearvalue,statenames):

"""

Forms a Scatter plot

Keyword Arguments:

yearvalue -- Gets the value of the year to be analyzed

statenames -- Gets the value state to be analyzed

The values of the states are fetched and compared using a stacked Bar chart

List Operation - Functions - PandasDataFrame Operations Implemented

"""

xvalue\_list = data['Low\_Confidence\_Limit'][(data['YEAR'] == yearvalue) & (data['LocationDesc'] == statenames)]

yvalue\_list = data['High\_Confidence\_Limit'][(data['YEAR'] == yearvalue) & (data['LocationDesc'] == statenames)]

name\_list = data['LocationDesc'][(data['YEAR'] == yearvalue) & (data['LocationDesc'] == statenames)]

trace = go.Scatter(

x=xvalue\_list,

y=yvalue\_list,

text = name\_list,

mode='markers',

opacity=0.7,

marker={

'size': 10,

'line': {'width': 0.4, 'color': 'white'}

},

)

return {

'data': [trace],

'layout': go.Layout(dict(

title = 'High Confidence and Low Confidence Comparison',

xaxis={'title': 'Low Confidence Limit'},

yaxis={'title': 'High Confidence Limit'}),

legend={'x': 0, 'y': 1},

hovermode='closest'

)

}

@app.callback(

dash.dependencies.Output('table\_chart', 'children'),

[dash.dependencies.Input('years1', 'value'),

dash.dependencies.Input('state\_names', 'value')])

def update\_table\_chart(yearvalue,statenames):

"""

Forms a table with Mean Standard Deviation, Minimumm and Maximum values

yearvalue -- Gets the value of the year to be analyzed

statenames -- Gets the value state to be analyzed.

A table displaying the statistical data is obtained

"""

xvalue\_list = data['Low\_Confidence\_Limit'][(data['YEAR'] == yearvalue) & (data['LocationDesc'] == statenames)]

yvalue\_list = data['High\_Confidence\_Limit'][(data['YEAR'] == yearvalue) & (data['LocationDesc'] == statenames)]

df = pd.DataFrame({'Low\_Confidence': xvalue\_list,'High\_Confidence' : yvalue\_list})

lcmin = df['Low\_Confidence'].min()

lcmax = df['Low\_Confidence'].max()

hcmin = df['High\_Confidence'].min()

hcmax = df['High\_Confidence'].max()

lcstd = df['Low\_Confidence'].std()

hcstd = df['High\_Confidence'].std()

lcmean = df['Low\_Confidence'].mean()

hcmean = df['High\_Confidence'].mean()

lclist = [lcmin,lcmax,lcstd,lcmean]

hclist = [hcmin,hcmax,hcstd,hcmean]

new\_df\_name = ['Minimum','Maximum','Standard Deviation','Mean']

new\_df = pd.DataFrame({'Name':new\_df\_name,'LowConfidence':lclist,'High Confidence':hclist})

print(new\_df)

max\_rows = 4

return html.Table(

# Header

[html.Tr([html.Th(col) for col in new\_df.columns])] +

# Body

[html.Tr([

html.Td(new\_df.iloc[i][col]) for col in new\_df.columns

]) for i in range(min(len(new\_df), max\_rows))]

)



**Question 7: Which Region in United States has the most number of youth smokers?**

import xlrd

import dash

import dash\_core\_components as dcc

import dash\_html\_components as html

import pandas as pd

import plotly.graph\_objs as go

import plotly as py

import numpy as np

import folium

from folium import plugins

from IPython.display import HTML, display

from folium.plugins import HeatMap

file\_location = pd.read\_csv("./youth\_tobacco\_analysis.csv")

data = pd.DataFrame(file\_location)

data = pd.DataFrame(data.groupby(['GeoLocation'])['Sample\_Size'].agg(['sum']))

data.to\_csv('./question6.csv', sep = ',', encoding='utf-8')

data = pd.read\_csv('./question6.csv')

lat = []

lon = []

data['GeoLocation'] = data['GeoLocation'].str.strip('()')

for row in data['GeoLocation']:

# Try to,

try:

# Split the row by comma and append

# everything before the comma to lat

lat.append(row.split(',')[0])

# Split the row by comma and append

# everything after the comma to lon

lon.append(row.split(',')[1])

# But if you get an error

except:

# append a missing value to lat

lat.append(np.NaN)

# append a missing value to lon

lon.append(np.NaN)

# Create two new columns from lat and lon

data['latitude'] = lat

data['latitude'] = data['latitude'].astype(float)

data['longitude'] = lon

data['longitude'] = data['longitude'].astype(float)

max\_amount = data['sum'].max()

values = []

for i in data['sum']:

values.append(i)

url\_base = 'http://server.arcgisonline.com/ArcGIS/rest/services/'

service = 'NatGeo\_World\_Map/MapServer/tile/{z}/{y}/{x}'

tileset = url\_base + service

colors = {'A' : 'red', 'B' : 'blue'}

hmap = folium.Map(location=[data.latitude[10], data.longitude[10]], zoom\_start=3,control\_scale = True, tiles=tileset, attr='USGS style' )

hm\_wide = HeatMap( list(zip(data.latitude, data.longitude, values)),

min\_opacity=0.2,

max\_val=max\_amount,

radius=17, blur=10,

max\_zoom=1,

)

hmap.add\_child(hm\_wide)

