**Group Project 1**

**PART 2**

**Dataset**

The dataset we choose to explore for this project is the Employee Salary Dataset from <https://www.kaggle.com/varungitboi/employee-salary-dataset/download#employee_data.csv>

It has a size of 1000 rows and 6 features.

The features include:

* *id*: interviewee ID
  + Type: quantitative (primary key)
* *groups*: blood groups
  + Type: qualitative/categorical (A, B, AB, O)
* *age*: interviewee’s age
  + Type: quantitative (from 18-64)
* *healthy\_eating*: scale 1-10 from Non-healthy Eating to Extremely Healthy Eating
  + Type: quantitative (from 1-10)
* *active\_lifestyle*: scale 1-10 from Non-active At All to Extremely Active
  + Type: quantitative (from 1-10)
* *salary*: monthly salary
  + Type: quantitative (from 553-5550)

**Visualization**

*All the code can be found at Lu’s repository at GitHub:*

[*https://github.com/techill/data-vis\_fall\_2019*](https://github.com/techill/data-vis_fall_2019)

**GRAPH #1**

**What kind of plot?**

* Scatter Plot Matrix, Heatmap

**What attribute in this plot?**

* age
  + Quantitative
* healthy\_eating
  + Quantitative
* active\_lifestyle
  + Quantitative
* salary
  + Quantitative

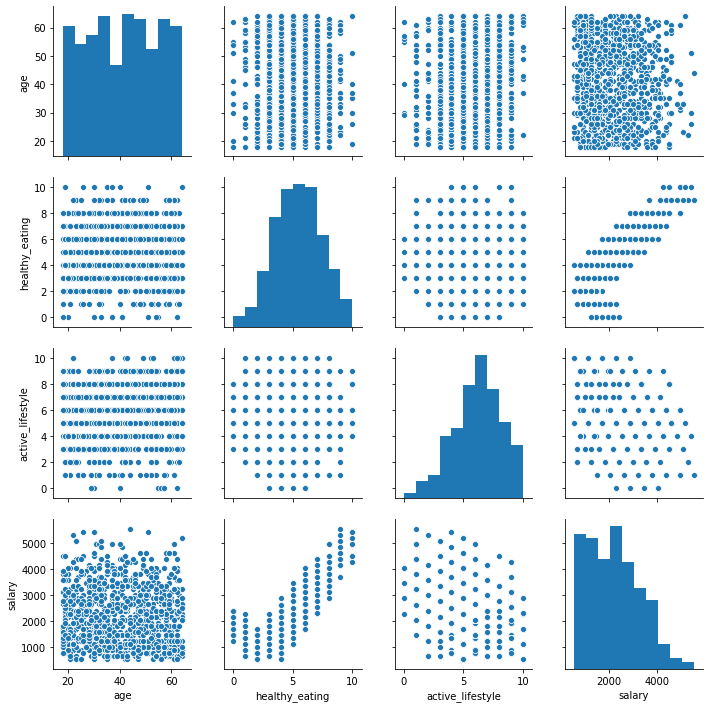
**What language did you use in this plot?**

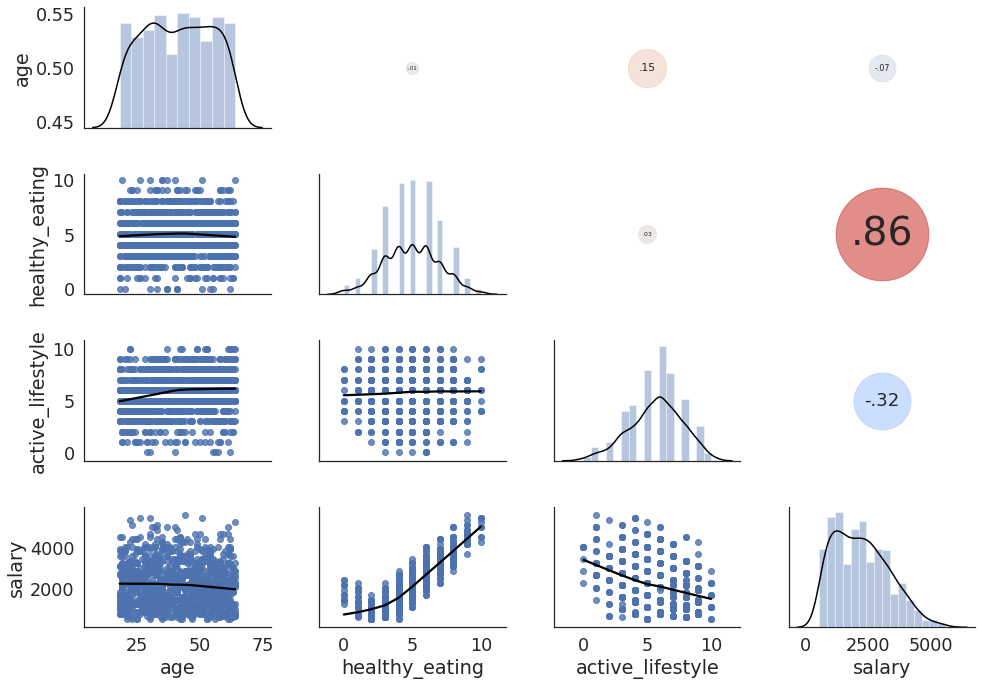
* Python

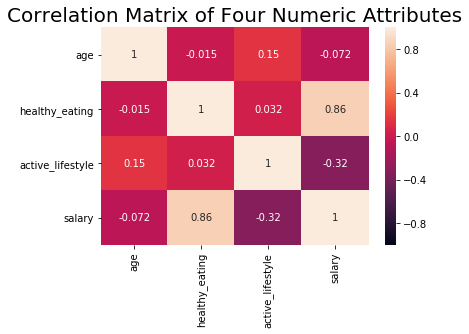
**What tools/packages/libraries did you use in your plot?**

* Pandas
* Numpy
* Scikit-learn
* Matplotlib
* Seaborn

**Screenshots:**







**Code:**

# Import libraries for data preprocessing:

from pandas import DataFrame

import pandas as pd

import numpy as np

from sklearn.preprocessing import LabelEncoder

# Import library for visualization:

import matplotlib.pyplot as plt

%matplotlib inline

import seaborn as sns

# Load dataset from the source link into Pandas Dataframe:

df = pd.read\_csv("https://raw.githubusercontent.com/techill/data-vis\_fall\_2019/master/employee\_data.csv", header = 0)

# Print the data type of the variable:

print(type(df))

# Print the first n rows of the dataframe:

df

df.shape

# Count the number of missing values in each column:

df.isnull().sum()

df.describe()

# # Remove columns from a dataframe by iloc:

# df.drop(df.iloc[:, 0:2], inplace = True, axis = 1) # 0:2 indicates removing columns from from Index 0 to 1.

# Remove columns from a dataframe by column index:

df.drop(df.columns[[0, 1]], axis = 1, inplace = True)

df

# import seaborn as sns

# Plot correlation matrices between every two numeric features:

sns.pairplot(df)

plt.show()

# import seaborn as sns

# Heatmap of Correlations Matrix:

df.corr()

sns.heatmap(df.corr(),

vmin = -1, # Set the minimun value of the color scale as -1.

annot = True) # Add annotation of coefficients to the heatmap.

# Fix the bug that cuts off top/bottom of Seaborn visualization:

b, t = plt.ylim() # Find out the values for bottom and top.

b += 0.5 # Add 0.5 to the bottom.

t -= 0.5 # Subtract 0.5 from the top.

plt.ylim(b, t) # Update the ylim(bottom, top) values.

plt.title('Correlation Matrix of Four Numeric Attributes', fontsize = 20)

plt.show()

# import seaborn as sns

def corrdot(\*args, \*\*kwargs):

corr\_r = args[0].corr(args[1], 'pearson')

corr\_text = f"{corr\_r:2.2f}".replace("0.", ".")

ax = plt.gca()

ax.set\_axis\_off()

marker\_size = abs(corr\_r) \* 10000

ax.scatter([.5], [.5], marker\_size, [corr\_r], alpha = 0.6, cmap = "coolwarm",

vmin = -1, vmax = 1, transform = ax.transAxes)

font\_size = abs(corr\_r) \* 40 + 5

ax.annotate(corr\_text, [.5, .5,], xycoords = "axes fraction",

ha = 'center', va = 'center', fontsize = font\_size)

sns.set(style = 'white', font\_scale = 1.6)

cmc = sns.PairGrid(df, aspect = 1.4, diag\_sharey = False)

cmc.map\_lower(sns.regplot, lowess = True, ci = False, line\_kws = {'color': 'black'})

cmc.map\_diag(sns.distplot, kde\_kws = {'color': 'black'})

cmc.map\_upper(corrdot)

plt.show()

**GRAPH #2**

**What kind of plot?**

* Parallel Coordinates (Static & Dynamic)

**What attribute in this plot?**

* age
  + Quantitative
* healthy\_eating
  + Quantitative
* active\_lifestyle
  + Quantitative
* salary
  + Quantitative
* groups
  + Categorical
* blood\_group\_code (The numeric representation of the “groups” attribute)
  + Quantitative

**What language did you use in this plot?**

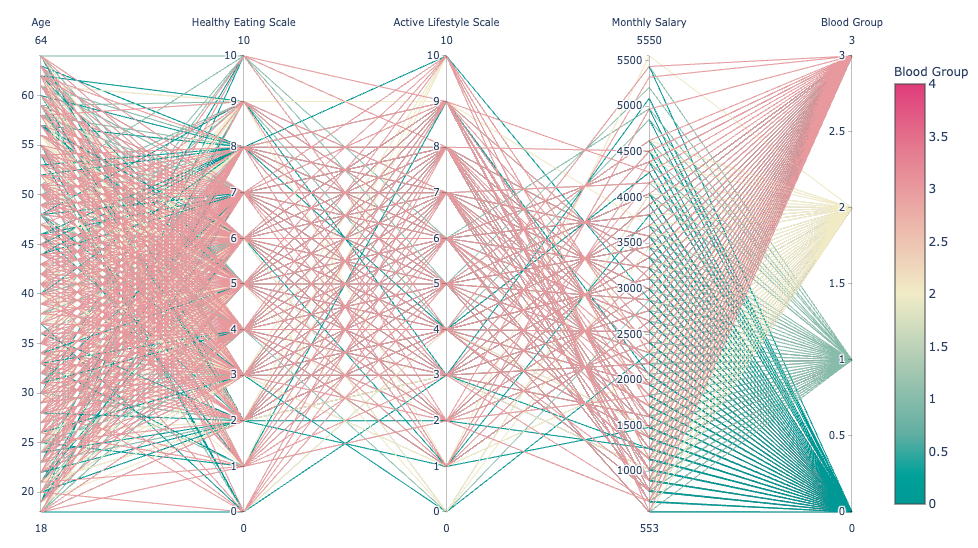
* Python

**What tools/packages/libraries did you use in your plot?**

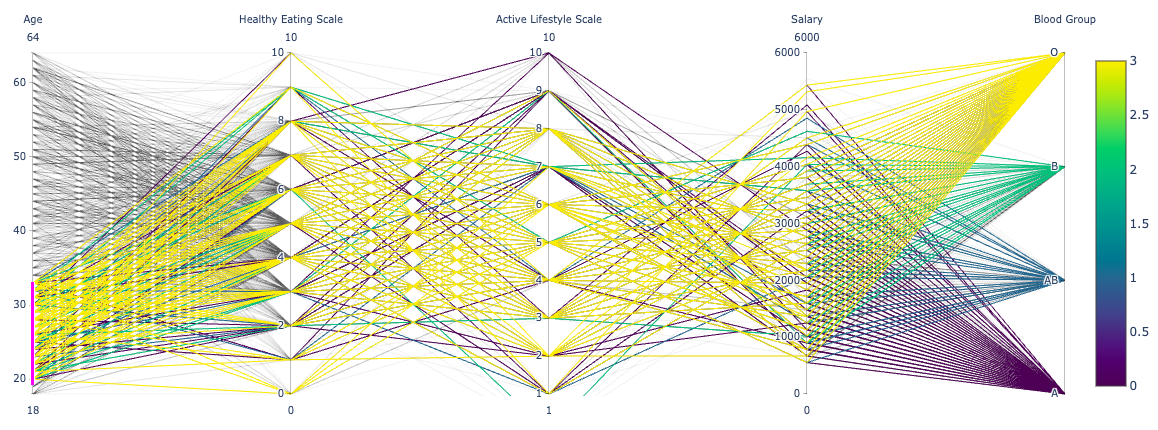
* Pandas
* Numpy
* Scikit-learn
* Matplotlib
* Seaborn
* Plotly

**Screenshots:**

Static:



Dynamic:



**Code:**

# Import libraries for data preprocessing:

from pandas import DataFrame

import pandas as pd

import numpy as np

from sklearn.preprocessing import LabelEncoder

# Import library for visualization:

import matplotlib.pyplot as plt

%matplotlib inline

import seaborn as sns

# Implot Plotly library for static parallel coordinates plot:

import plotly.express as px

# Implot Plotly library for interactive parallel coordinates plot:

import plotly.graph\_objects as go

# from sklearn.preprocessing import LabelEncoder

# Initialize the label encoder:

lbEncoder = LabelEncoder()

# Covert the categorical data of groups into numeric codes:

df["blood\_group\_code"] = lbEncoder.fit\_transform(df["groups"])

df.head()

# import plotly.express as px

figPCstatic = px.parallel\_coordinates(df, color = "blood\_group\_code",

labels = {"age": "Age",

"healthy\_eating": "Healthy Eating Scale",

"active\_lifestyle": "Active Lifestyle Scale",

"salary": "Monthly Salary",

"blood\_group\_code": "Blood Group"},

color\_continuous\_scale = px.colors.diverging.Tealrose,

color\_continuous\_midpoint = 2)

figPCstatic.show()

# import plotly.graph\_objects as go

figPCinteractive = go.Figure(data =

go.Parcoords(

line = dict(color = df["blood\_group\_code"],

colorscale = "Viridis",

showscale = True),

dimensions = list([

dict(range = [18, 64],

label = "Age", values = df["age"]),

dict(range = [0, 10],

label = "Healthy Eating Scale", values = df['healthy\_eating']),

dict(range = [1, 10],

label = "Active Lifestyle Scale", values = df['active\_lifestyle']),

dict(range = [0, 6000],

label = "Salary", values = df['salary']),

dict(tickvals = [0, 1, 2, 3],

ticktext = ["A", "AB", "B", "O"],

label = "Blood Group", values = df['blood\_group\_code'])

])

))

figPCinteractive.show()

**GRAPH #3**

**What kind of plot?**

* Bar Chart, Pie Chart, Scatter Plot

**What attribute in this plot?**

* groups
  + Categorical
* age
  + Quantitative
* healthy\_eating
  + Quantitative
* active\_lifestyle
  + Quantitative
* salary
  + Quantitative

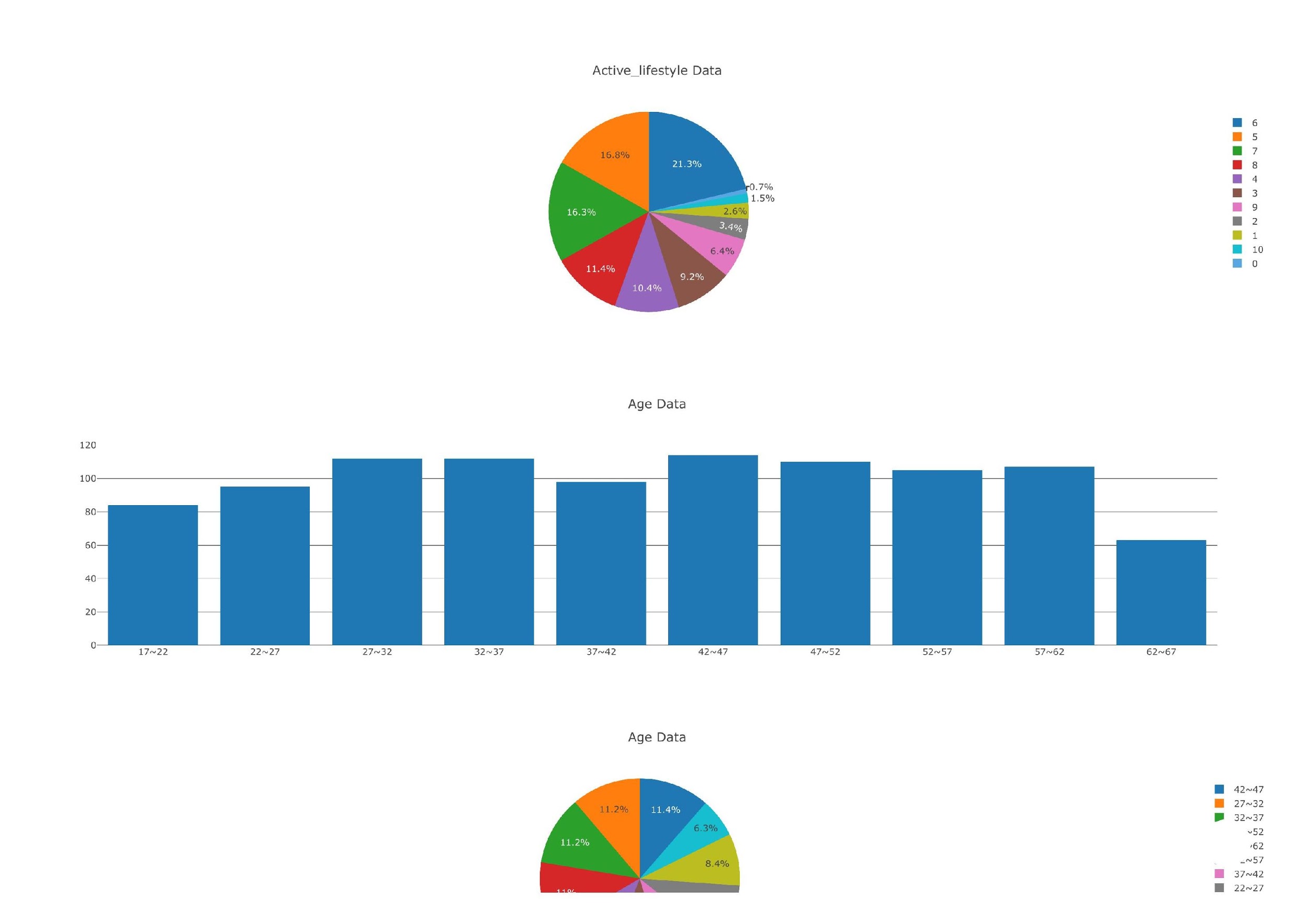
**What language did you use in this plot?**

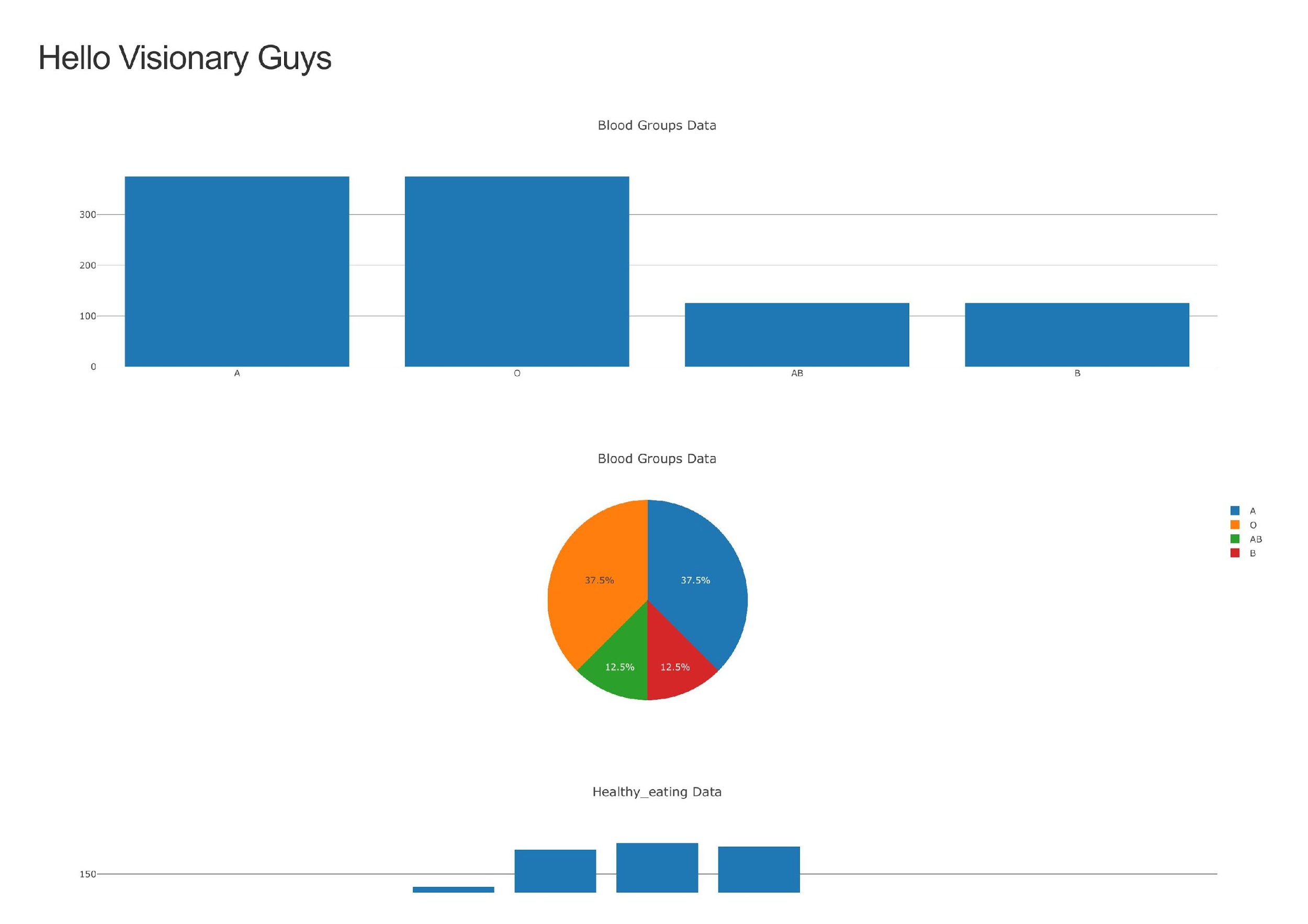
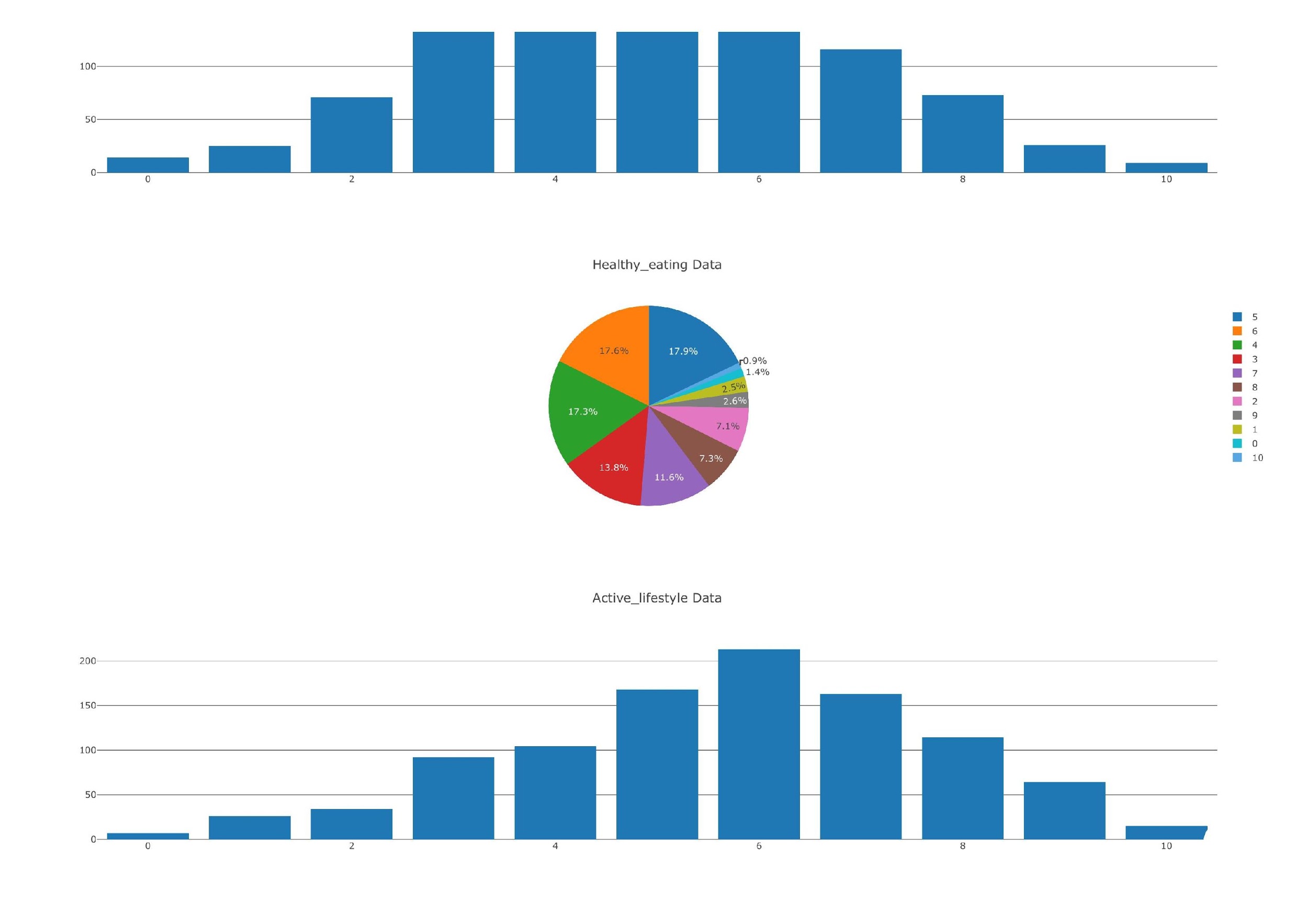
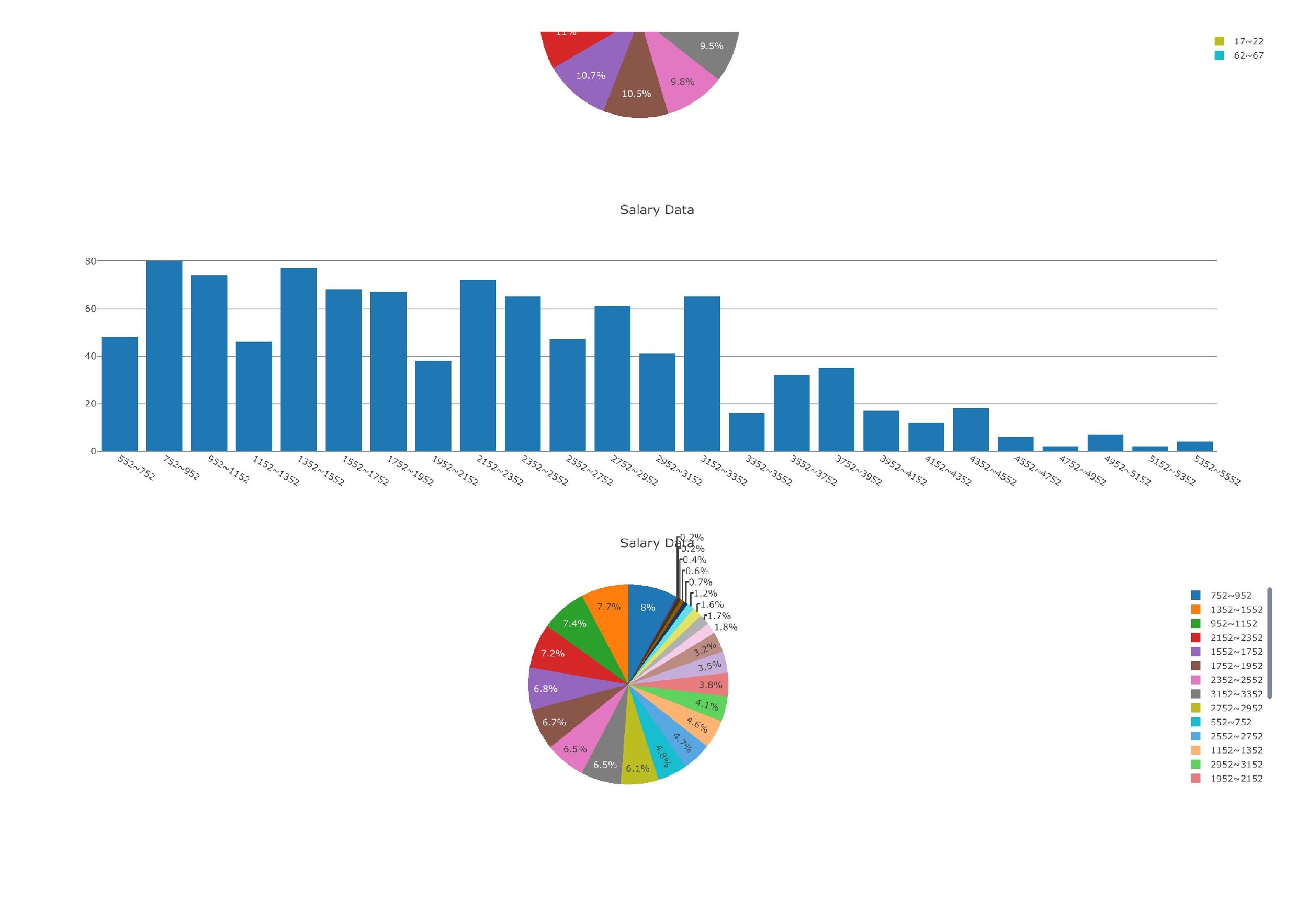
* Python

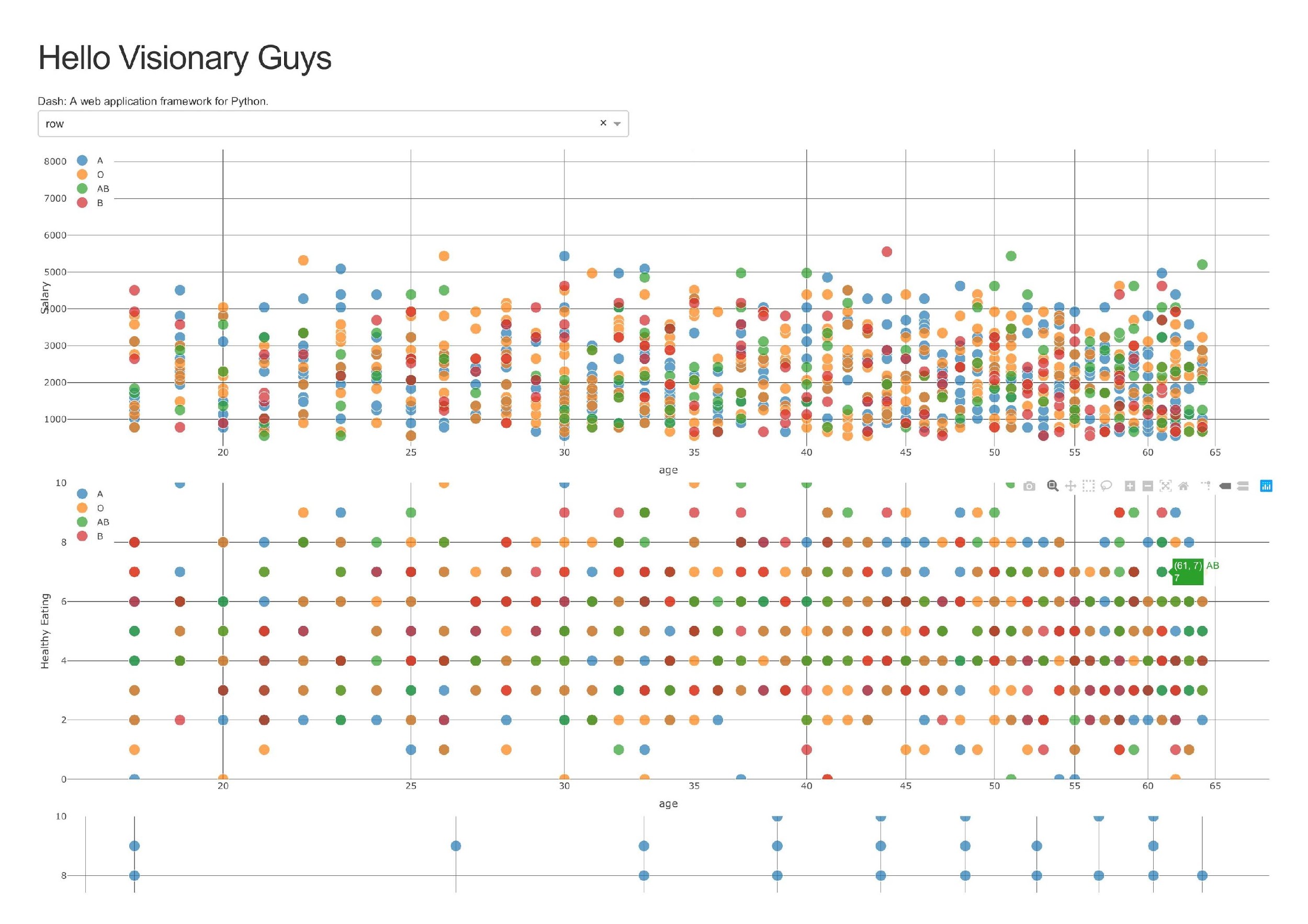
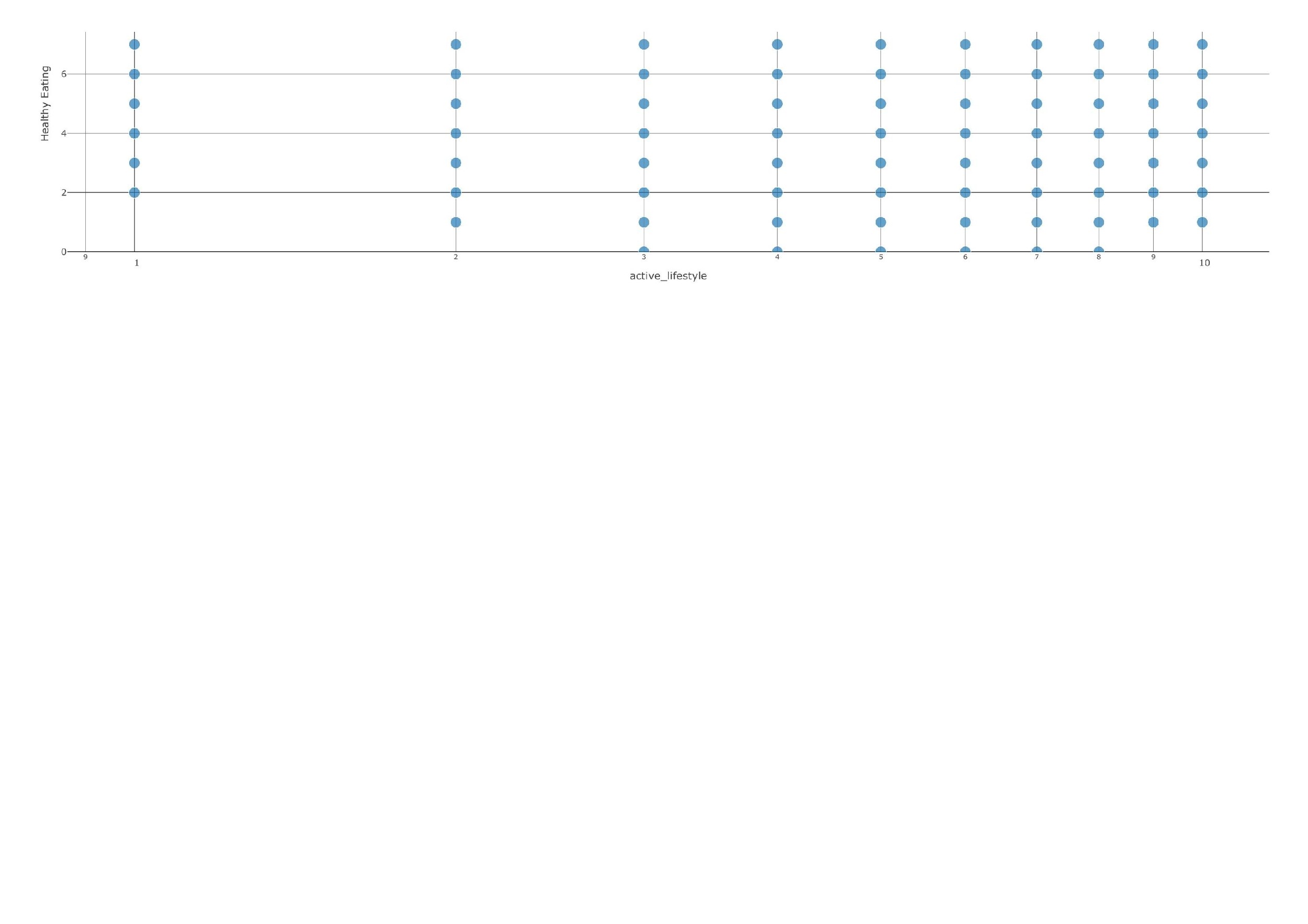
**What tools/packages/libraries did you use in your plot?**

* Dash
* Pandas

**Screenshots:**







**Code:**

Program1: Test\_scatter.py

# -\*- coding: utf-8 -\*-  
import dash  
import dash\_core\_components as dcc  
import dash\_html\_components as html  
import pandas as pd  
import plotly.graph\_objs as go  
  
external\_stylesheets = ['https://codepen.io/chriddyp/pen/bWLwgP.css']  
  
app = dash.Dash(\_\_name\_\_, external\_stylesheets=external\_stylesheets)  
  
df = pd.read\_csv('./employee\_data.csv')  
print(df.shape)  
print(df.columns.values)  
  
  
def get\_figure\_1():  
 filtered\_df = df  
 traces = []  
 for i in filtered\_df.groups.unique():  
 df\_by\_groups = filtered\_df[filtered\_df['groups'] == i]  
 traces.append(go.Scatter(  
 x=df\_by\_groups['age'],  
 y=df\_by\_groups['salary'],  
 text=df\_by\_groups['healthy\_eating'],  
 mode='markers',  
 opacity=0.7,  
 marker={  
 'size': 15,  
 'line': {'width': 0.5, 'color': 'white'}  
 },  
 name=i  
 ))  
  
 return {  
 'data': traces,  
 'layout': go.Layout(  
 xaxis={'type': 'log', 'title': 'age '},  
 yaxis={'title': 'Salary', 'range': [df['salary'].min()\*0.5, df['salary'].max()\*1.5]},  
 margin={'l': 40, 'b': 40, 't': 10, 'r': 10},  
 legend={'x': 0, 'y': 1},  
 hovermode='closest'  
 )  
 }  
  
  
def get\_figure\_2():  
 filtered\_df = df  
 traces = []  
 for i in filtered\_df.groups.unique():  
 df\_by\_groups = filtered\_df[filtered\_df['groups'] == i]  
 traces.append(go.Scatter(  
 x=df\_by\_groups['age'],  
 y=df\_by\_groups['healthy\_eating'],  
 text=df\_by\_groups['healthy\_eating'],  
 mode='markers',  
 opacity=0.7,  
 marker={  
 'size': 15,  
 'line': {'width': 0.5, 'color': 'white'}  
 },  
 name=i  
 ))  
  
 return {  
 'data': traces,  
 'layout': go.Layout(  
 xaxis={'type': 'log', 'title': 'age '},  
 yaxis={'title': 'Healthy Eating', 'range': [df['healthy\_eating'].min(), df['healthy\_eating'].max()]},  
 margin={'l': 40, 'b': 40, 't': 10, 'r': 10},  
 legend={'x': 0, 'y': 1},  
 hovermode='closest'  
 )  
 }  
  
  
def get\_figure\_3():  
 filtered\_df = df  
 traces = []  
 df\_by\_groups = filtered\_df  
 traces.append(go.Scatter(  
 x=df\_by\_groups['active\_lifestyle'],  
 y=df\_by\_groups['healthy\_eating'],  
 text=df\_by\_groups['healthy\_eating'],  
 mode='markers',  
 opacity=0.7,  
 marker={  
 'size': 15,  
 'line': {'width': 0.5, 'color': 'white'}  
 },  
 name='all group'  
 ))  
 return {  
 'data': traces,  
 'layout': go.Layout(  
 xaxis={'type': 'log', 'title': 'active\_lifestyle ', },  
 yaxis={'title': 'Healthy Eating', 'range': [df['healthy\_eating'].min(), df['healthy\_eating'].max()]},  
 margin={'l': 40, 'b': 40, 't': 10, 'r': 10},  
 legend={'x': 0, 'y': 1},  
 hovermode='closest'  
 )  
 }  
  
  
app.layout = html.Div(children=[  
 html.H1(children='Hello Visionary Guys'),  
  
 html.Div(children='''  
 Dash: A web application framework for Python.  
 '''),  
 html.Div([  
 dcc.Dropdown(  
 id='attributes',  
 options=[{'label': i, 'value': i} for i in df.columns.values],  
 value=df.columns.values[0]  
 )  
 ], style={'width': '48%', 'display': 'inline-block'}),  
 dcc.Graph(  
 id='graph-scatter-1',  
 figure=get\_figure\_1()  
 ),  
 dcc.Graph(  
 id='graph-scatter-2',  
 figure=get\_figure\_2()  
 ),  
 dcc.Graph(  
 id='graph-scatter-3',  
 figure=get\_figure\_3()  
 )  
 ])  
  
  
  
if \_\_name\_\_ == '\_\_main\_\_':  
 app.run\_server(debug=True)

2.program2 : my\_test\_barchart\_piechart.py

# -\*- coding: utf-8 -\*-  
import math  
  
import dash  
import dash\_core\_components as dcc  
import dash\_html\_components as html  
import pandas as pd  
  
external\_stylesheets = ['https://codepen.io/chriddyp/pen/bWLwgP.css']  
  
app = dash.Dash(\_\_name\_\_, external\_stylesheets=external\_stylesheets)  
  
df = pd.read\_csv('./employee\_data.csv')  
print(df.shape)  
print(df.columns.values)  
  
def get\_barchart\_data\_cat(col\_name):  
 x = df[col\_name].unique()  
 # print(x.shape)  
 y = []  
 for i in x:  
 y.append(df[df[col\_name] == i].shape[0])  
 return {'x': x, 'y': y, 'type': 'bar', 'name': col\_name}  
  
  
def get\_piechart\_data\_cat(col\_name):  
 x = df[col\_name].unique()  
 # print(x.shape)  
 y = []  
 for i in x:  
 y.append(df[df[col\_name] == i].shape[0])  
 return {'labels': x, 'values': y, 'type': 'pie', 'name': col\_name}  
  
  
def get\_barchart\_data\_real(col\_name,span):  
 min = df[col\_name].min() - 1  
 max = df[col\_name].max() + 1  
 spans = math.ceil((max - min)/span)  
 x = []  
 interval\_left = min  
 interval\_right = min + span  
 y = []  
 for i in range(0, spans):  
 x.append(str(interval\_left) + "~" + str(interval\_right))  
 y.append(df[ (df[col\_name] >= interval\_left) & (df[col\_name] < interval\_right)].shape[0])  
 interval\_left = interval\_left + span  
 interval\_right = interval\_right + span  
 return {'x': x, 'y': y, 'type': 'bar', 'name': col\_name}  
  
  
def get\_piechart\_data\_real(col\_name,span):  
 min = df[col\_name].min() - 1  
 max = df[col\_name].max() + 1  
 spans = math.ceil((max - min)/span)  
 x = []  
 interval\_left = min  
 interval\_right = min + span  
 y = []  
 for i in range(0, spans):  
 x.append(str(interval\_left) + "~" + str(interval\_right))  
 y.append(df[ (df[col\_name] >= interval\_left) & (df[col\_name] < interval\_right)].shape[0])  
 interval\_left = interval\_left + span  
 interval\_right = interval\_right + span  
 return {'labels': x, 'values': y, 'type': 'pie', 'name': col\_name}  
  
  
app.layout = html.Div(children=[  
 html.H1(children='Hello Visionary Guys'),  
  
 dcc.Graph(  
 id='barchar-1',  
 figure={  
 'data': [  
 get\_barchart\_data\_cat('groups')  
 ],  
 'layout': {  
 'title': 'Blood Groups Data'  
 }  
 }  
 ),  
 dcc.Graph(  
 id='piechart-1',  
 figure={  
 'data': [  
 get\_piechart\_data\_cat('groups')  
 ],  
 'layout': {  
 'title': 'Blood Groups Data'  
 }  
 }  
 ),  
 dcc.Graph(  
 id='barchar-2',  
 figure={  
 'data': [  
 get\_barchart\_data\_cat('healthy\_eating')  
 ],  
 'layout': {  
 'title': 'Healthy\_eating Data'  
 }  
 }  
 ),  
 dcc.Graph(  
 id='piechart-2',  
 figure={  
 'data': [  
 get\_piechart\_data\_cat('healthy\_eating')  
 ],  
 'layout': {  
 'title': 'Healthy\_eating Data'  
 }  
 }  
 ),  
 dcc.Graph(  
 id='barchar-3',  
 figure={  
 'data': [  
 get\_barchart\_data\_cat('active\_lifestyle')  
 ],  
 'layout': {  
 'title': 'Active\_lifestyle Data'  
 }  
 }  
 ),  
 dcc.Graph(  
 id='piechar-3',  
 figure={  
 'data': [  
 get\_piechart\_data\_cat('active\_lifestyle')  
 ],  
 'layout': {  
 'title': 'Active\_lifestyle Data'  
 }  
 }  
 ),  
 dcc.Graph(  
 id='barchar-4',  
 figure={  
 'data': [  
 get\_barchart\_data\_real('age', 5)  
 ],  
 'layout': {  
 'title': 'Age Data'  
 }  
 }  
 ),  
 dcc.Graph(  
 id='piechar-4',  
 figure={  
 'data': [  
 get\_piechart\_data\_real('age', 5)  
 ],  
 'layout': {  
 'title': 'Age Data'  
 }  
 }  
 ),  
 dcc.Graph(  
 id='barchar-5',  
 figure={  
 'data': [  
 get\_barchart\_data\_real('salary', 200)  
 ],  
 'layout': {  
 'title': 'Salary Data'  
 }  
 }  
 ),  
 dcc.Graph(  
 id='piechar-5',  
 figure={  
 'data': [  
 get\_piechart\_data\_real('salary', 200)  
 ],  
 'layout': {  
 'title': 'Salary Data'  
 }  
 }  
 )  
])  
  
if \_\_name\_\_ == '\_\_main\_\_':  
 app.run\_server(debug=True)

**GRAPH #4**

**What kind of plot?**

* Scatter Plot, Bar Chart

**What attribute in this plot?**

* Using PCA variance ratio as attributes for bar plot. Using the first 2 PCA as attributes for the scatter plot.

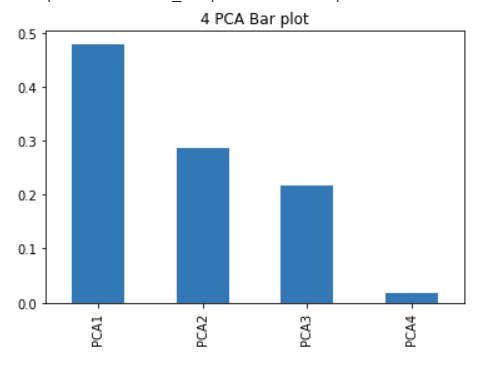
**What language did you use in this plot?**

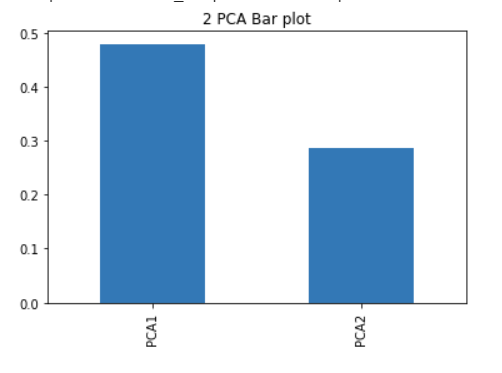
* Python

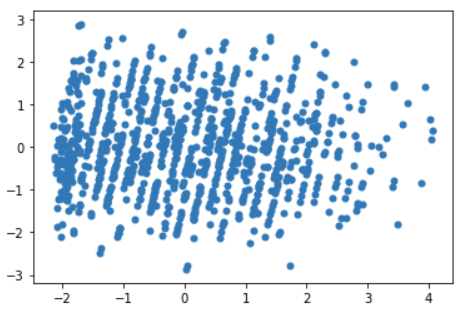
**What tools/packages/libraries did you use in your plot?**

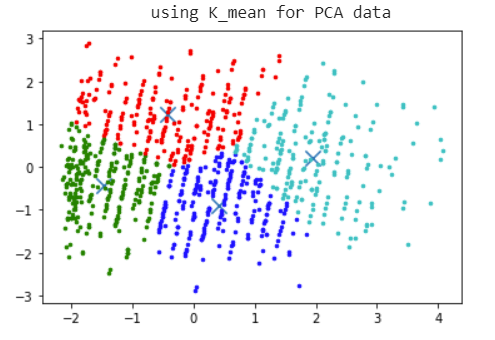
* Pandas
* Numpy
* Scikit-learn
* Matplotlib

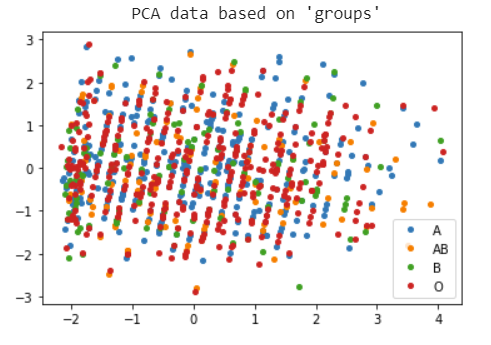
**Screenshots:**











**Code:**

import pandas as pd

from pandas import DataFrame

import numpy as np

import matplotlib.pyplot as plt

data = pd.read\_csv('employee\_data.csv',header=0)

data1=data[['age','healthy\_eating','active\_lifestyle','salary']]

data1

from sklearn.preprocessing import StandardScaler

features = ['age','healthy\_eating','active\_lifestyle','salary']

# Separating out the features

x = data1.loc[:, features].values

x = StandardScaler().fit\_transform(x)

print(x)

from sklearn.decomposition import PCA

pca = PCA(n\_components=4)

pca.fit(x)

#4D explained value ratio

data3=pca.explained\_variance\_ratio\_

from pandas import Series

s3 = Series(data3,

index = ['PCA1','PCA2','PCA3','PCA4'])

s3.plot(kind='bar', title='4 PCA Bar plot')

pca1 = PCA(n\_components=2)

pca1.fit(x)

#2D explained value ratio

data4=pca1.explained\_variance\_ratio\_

from pandas import Series

s1 = Series(data4,

index = ['PCA1', 'PCA2'])

s1.plot(kind='bar', title='2 PCA Bar plot')

X\_new = pca1.transform(x)

plt.scatter(X\_new[:, 0], X\_new[:, 1],marker='o',linewidths=0.01)

plt.show()

from sklearn.cluster import KMeans

from sklearn.externals import joblib

import numpy

import matplotlib.pyplot as plt

clf = KMeans(n\_clusters=4)

clf.fit(X\_new)

centroids = clf.cluster\_centers\_

labels = clf.labels\_

colors = ['g.','r.','c.','b.']

print(" using K\_mean for PCA data")

for i in range(len(X\_new)):

plt.plot(X\_new[i][0], X\_new[i][1], colors[labels[i]], markersize = 5)

plt.scatter(centroids[:,0], centroids[:,1], marker='x', s=150, linewidths=6)

plt.show()

import matplotlib.pyplot as plt

import numpy as np

import pandas as pd

labels = data["groups"]

df = pd.DataFrame(dict(x=X\_new[:, 0], y=X\_new[:, 1], label=labels))

groups = df.groupby('label')

# Plot

fig, ax = plt.subplots()

ax.margins(0.05) # Optional, just adds 5% padding to the autoscaling

print(" PCA data based on 'groups'")

for name, group in groups:

ax.plot(group.x, group.y, marker='o', linestyle='', ms=4, label=name)

ax.legend()

plt.show()