Course Code: ANL252

Title: ECA01

PI No.: H2310678

Name: Muhammad Bin Osman

Date of submission: 03/11/23

Group: T03

1)

Code:

import pandas as pd

df\_medical = pd.read\_csv("ECA.csv")

#removal of missing data (1)

df\_medical.isnull().sum()

df\_medical.dropna(axis = "rows", inplace = True) #remove rows with missing

#standardise gender entries (2)

df\_medical["sex"] = df\_medical["sex"].replace({"M":"male", "F":"female"})

#identifying and removal of outliers (medical costs) (3)

q1 = df\_medical["charges"].quantile(q = 0.25)

q3 = df\_medical["charges"].quantile(q = 0.75)

iqr = q3 - q1 #interquartile range

llimit = q1 - (1.5\*iqr) #upper limit

ulimit = q3 + (1.5\*iqr) #lower limit

df\_med\_new = df\_medical[~((df\_medical["charges"] < llimit) | (df\_medical["charges"] > ulimit))] #display data without outliers

#review updated data

df\_med\_new

Firstly, we would remove any form of missing data from the dataset. With regards to missing data, we can either replace or remove it. In this case, removing it would be feasible as the dataset is large enough. Leaving missing data in may result in making bias or inaccurate analysis. As seen in (1) in the code, pandas’ .isnull().sum()allows me to locate and calculate the sum of missing inputs. Next, we use .dropna to remove data with missing values. We would then specify it with (axis = "rows", inplace = True) for Python to remove rows within the actual dataset with missing values.

Secondly, we would standardise the data entries. Within the data, in the column “sex”, we see discrepencies where some inputs are labeled with “M”/”F” instead of male/female. We would standardise it by replacing the wrong inputs with the proper variants with the command .replace({"M":"male", "F":"female"}). With this, all “M”/”F” will be converted to male/female. This is important as it affects the data quality. Leaving misinputs may result in ambiguity and even worse, errors or inaccuracy in data visualisation via charts etc. as they rely on standardised labels. Correcting it would also make the data more readable for other users.

Lastly, we would get rid of outliers from the data. As the data relates to the medical costs, we would use it as the basis for outliers. We calculate q1 and q3 via the command quantile(q = 0.25) and quantile(q = 0.75). Next, we would find the interquartile range and the upper/lower limits and use it in df\_medical[~((df\_medical["charges"] < llimit) | (df\_medical["charges"] > ulimit))] to get an output of the data without outliers. I have assigned it to df\_med\_new. Ridding of outliers is important as outliers can skew analytical results, making visualisation more inaccurate of the actual dataset. Distortion of data may occur as they exist as noise, making the data quality not as good as it can be.

(331 words)

2)

Chart 1 code:

import matplotlib.pyplot as plt

from sklearn.linear\_model import LinearRegression

#scatterplot between charges and age

plt.scatter(df\_med\_new["age"],df\_med\_new["charges"],

color = "blue",

marker = "o",

s = 25,

alpha = 0.5

)

#graph labels

plt.title("Correlation between age and charges", fontsize = 12)

plt.xlabel("Age of beneficiaries", fontsize = 12)

plt.ylabel("Medical costs", fontsize = 12)

#graph style

plt.style.use("fivethirtyeight")

#tick sizing

plt.xticks(fontsize=8)

plt.yticks(fontsize=8)

#line of best fit

x = df\_med\_new[["age"]]

y = df\_med\_new["charges"]

bfline = LinearRegression()

bfline.fit(x,y)

y\_predict = bfline.predict(x)

plt.plot(x, y\_predict,

color = "red",

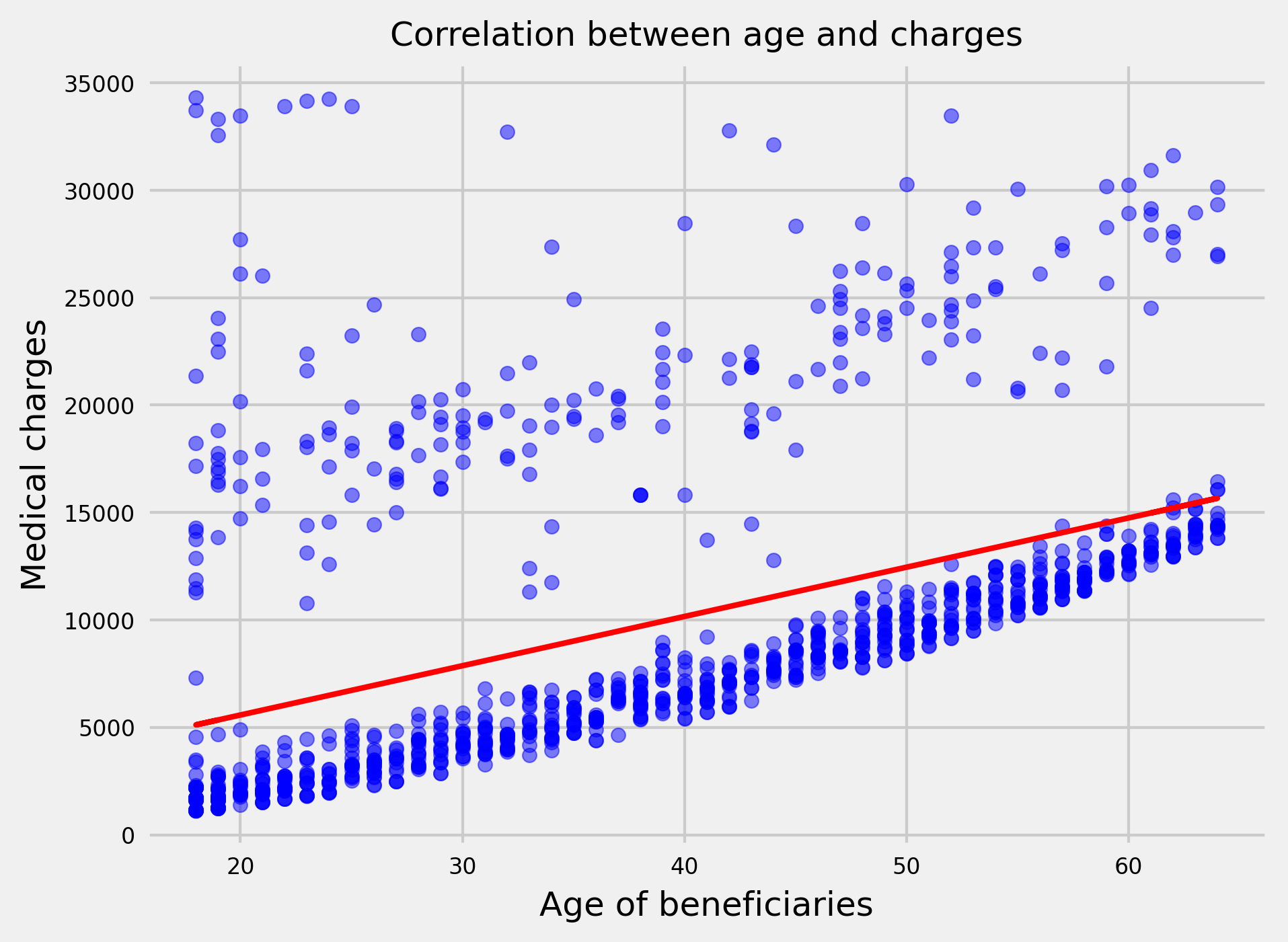
linewidth = 2) #draws the line of best fit

# Save as image

plt.savefig("scatterplot\_age\_charges", dpi=300)

plt.show()

Chart 1:



This data shows a positive correlation between age and charges, where there is a generally increasing trend. The data spread is relatively tight, with outliers present. These outliers are spread evenly and hence do not negatively impact the quality of the analysis. We can also note a small cluster between the ages of 18 and 25 within the outliers. We can infer from this data that older beneficiaries tend to be charged higher as compared to younger beneficiaries. This could be likely due to older beneficiaries having underlying medical conditions that develop as they age. Having more medical complications would drive up medical costs due to additional medication and care needed. As mentioned, there is a small cluster between 18 to 25 years old. This anomaly could be due to injuries sustained from high-risk activities such as sports. Beneficiaries in that age group are at prone to high medical bills due to their active nature.

Chart 2 code:

#bar graph

#grouping via smoking status + mean costs

smokerstatus = df\_med\_new.groupby(by = ["smoker"])["charges"].mean()

#axis defining

status = smokerstatus.index #x-axis yes/no

medcharge = smokerstatus.values #y-axis mean

#plotting bar graph

plt.bar(status,medcharge,

color = ["blue","red"],

width = 0.25

)

#adjustments

plt.title("Average medical charge by smoking status" , fontsize = 12)

plt.xlabel("Smoking status", fontsize = 12)

plt.ylabel("Mean medical charge", fontsize = 12)

plt.xticks(fontsize=8)

plt.yticks(fontsize=8)

#mean on each bar

for i in range(len(status)):

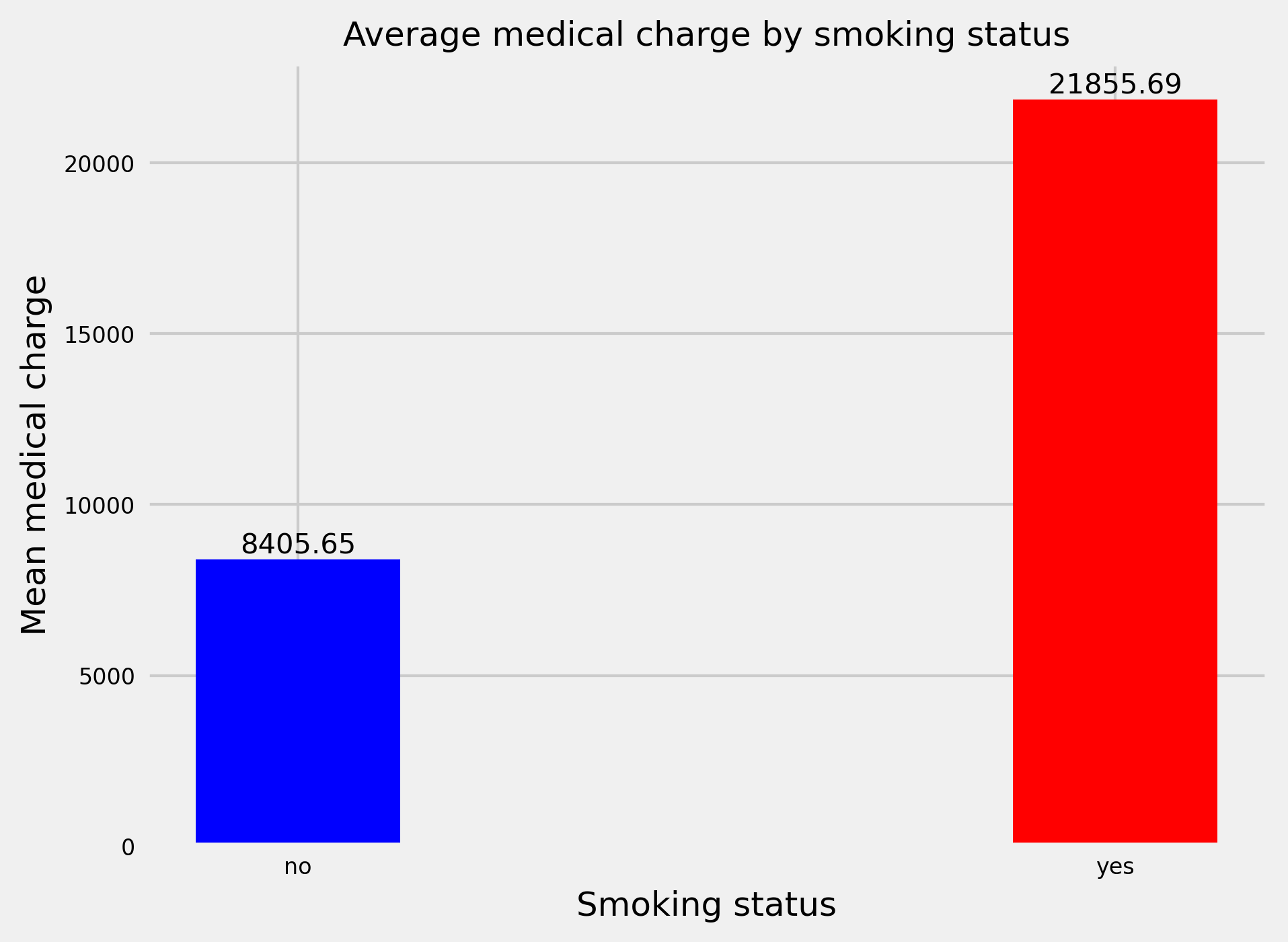
plt.text(status[i], medcharge[i], f"{medcharge[i]:.2f}",ha="center", va="bottom",fontsize=10)

# Save as image

plt.savefig("bar\_smoker", dpi=300)

plt.show()

Chart 2:



From the graph we see a large difference in mean medical charge between smokers and non-smokers. Smokers experience around 2.5 times the mean medical charge against non-smokers. This large difference could be a result of health risks associated with smoking, leading to additional medical care in treatments for diseases like bronchitis. With that, the graph also highlights a strong positive correlation between medical charges and smoking status.

Chart 3 code:

#heatmap

import seaborn as sns

#binning of bmi

bin\_bmi = [0, 18.5, 24.9, 29.9, 100] #ranges between each category

bin\_bmi\_labels = ["Underweight","Normal", "Overweight", "Obese"]

#binning of age

bin\_age = [18, 30, 40, 50, 60, 65]

bin\_age\_labels = ["18-29", "30-39", "40-49", "50-59", "60-65"]

#add coloumn for binned bmi

df\_med\_new["bmi\_category"] = pd.cut(df\_med\_new["bmi"], bins = bin\_bmi , labels = bin\_bmi\_labels)

#add coloumn for binned age

df\_med\_new["age\_range"] = pd.cut(df\_med\_new["age"], bins = bin\_age, labels = bin\_age\_labels)

#pivot table

heatmap\_data = df\_med\_new.pivot\_table(index=“age\_range”,

columns=“bmi\_category”,

values=“charges”,

aggfunc=“mean”,

)

#heatmap plot

plt.figure(figsize=(10, 7))

heatmap = sns.heatmap(heatmap\_data,

cmap=“coolwarm”,

annot=True,

fmt=“.2f”,

cbar=True,

linewidth = 0.5,

annot\_kws={"size": 10,},

xticklabels = True,

yticklabels = True

)

#adjustments

plt.title("Mean charges by Age and BMI" , fontsize = 15)

plt.xlabel("bmi\_category", fontsize= 13)

plt.ylabel("age\_range", fontsize = 13)

plt.xticks(fontsize = 10)

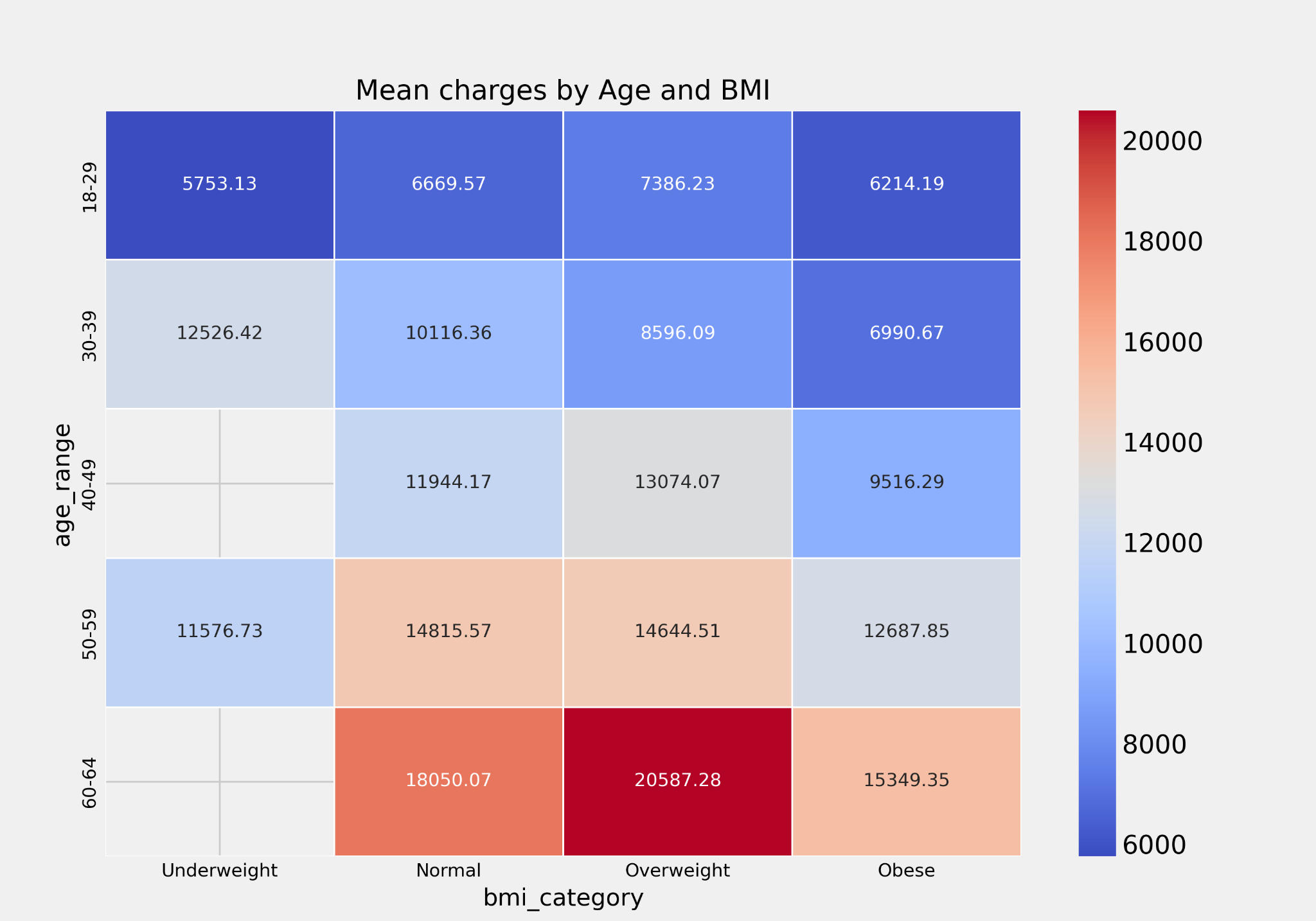
plt.yticks(fontsize = 10)

# Save as image

plt.savefig("heatmap\_agebmicharges", dpi=300)

plt.show()

Chart 3:



From the heatmap, we notice a clear trend. Medical costs are more expensive for older beneficiaries as compared to younger ones. This aligns with the correlation graph plotted earlier, highlighting the inference that developing and existing medical conditions contribute to the necessity for additional treatment, driving costs up. We see the mean spiking at the 60-64 age range, more notably in the overweight category. Interestingly, we notice that overweight beneficiaries have higher mean costs compared to the obese. Likewise, we notice further anomalies in the 30-39 age range where underweights are charged the highest. We can deduce that the usual expectation where the more “unhealthy” one is, the higher the medical charge may not be true all the time. Having seeing this trend, medical charges go beyond bmi range alone. Chronic illnesses, type of treament etc. influences medical charges, potentially skewing the distribution resulting in results as seen above. Hence, being obese does not necessarily mean higher charges than their counterparts.

(384 words)

3)

Code:

#data exploration for decision tree

#imports

import sklearn

from sklearn.model\_selection import train\_test\_split

from sklearn import tree

#remove unecessary columns

df\_med\_new.drop(columns = ["bmi\_category", "age\_range"], inplace = True)

#dummies

df\_dummy = pd.get\_dummies(df\_med\_new, columns = ["sex","region"])

#independent/dependent variables

y = df\_dummy["smoker"]

X = df\_dummy[["age",

"bmi",

"children",

"charges",

"sex\_female",

"sex\_male",

"region\_northeast",

"region\_northwest",

"region\_southeast",

"region\_southwest"]]

#splitting data

X\_train, X\_test, y\_train, y\_test = train\_test\_split(X, y, test\_size = 0.3, random\_state = 0)

#training model

dtc = tree.DecisionTreeClassifier(max\_depth = 5)

dtc.fit(X\_train, y\_train)

Firstly, we remove columns that are irrelevant when making the model by using .drop syntax. In this case, “bmi\_category” and “age\_range” that was used in the heatmap earlier would be removed. Next, we would need to carry out one-hot encoding for categorical values in the dataset as the CART algorithm relies on numerical inputs. By using pd.get\_dummies, python would convert the specified columns into separate binary columns, represented by 0 or 1, depending on the status of the object in question. We store the dataframe in a new variable df\_dummy to separate the datasets with and without the dummy values. With that, we can start to define our independent and dependent variables. X represents the independent variables (features) and y represents the dependent variable (target). In context, y would be “smoker” and X would be the other categories in df\_dummy, except for “PersonID”. We then proceed to split the data into 2 categories: train and test in order to both train the model with a portion of the data and use it to test it on the unseen data. In using train\_test\_split, we would also specify the test\_size to be 0.3 to split it into 70% for training and 30% for testing. Also, random\_state is set to 0 so the model runs on that specific seed only. Finally, we would build the model by using tree.DecisionTreeClassifier and specifying the max\_depth to be at 5 for easier interpretation and prevention of overfitting. The training data is then fit into the model via .fit before the tree is plotted.

4)

Code:

#feature labels

feature\_label = ["age",

"bmi",

"children",

"charges",

"sex\_female",

"sex\_male",

"region\_northeast",

"region\_northwest",

"region\_southeast",

"region\_southwest"]

#plotting the tree

plt.figure(figsize=(15, 8))

tree.plot\_tree(dtc,

filled = True,

rounded = True,

class\_names = ["yes smoke", "no smoke"],

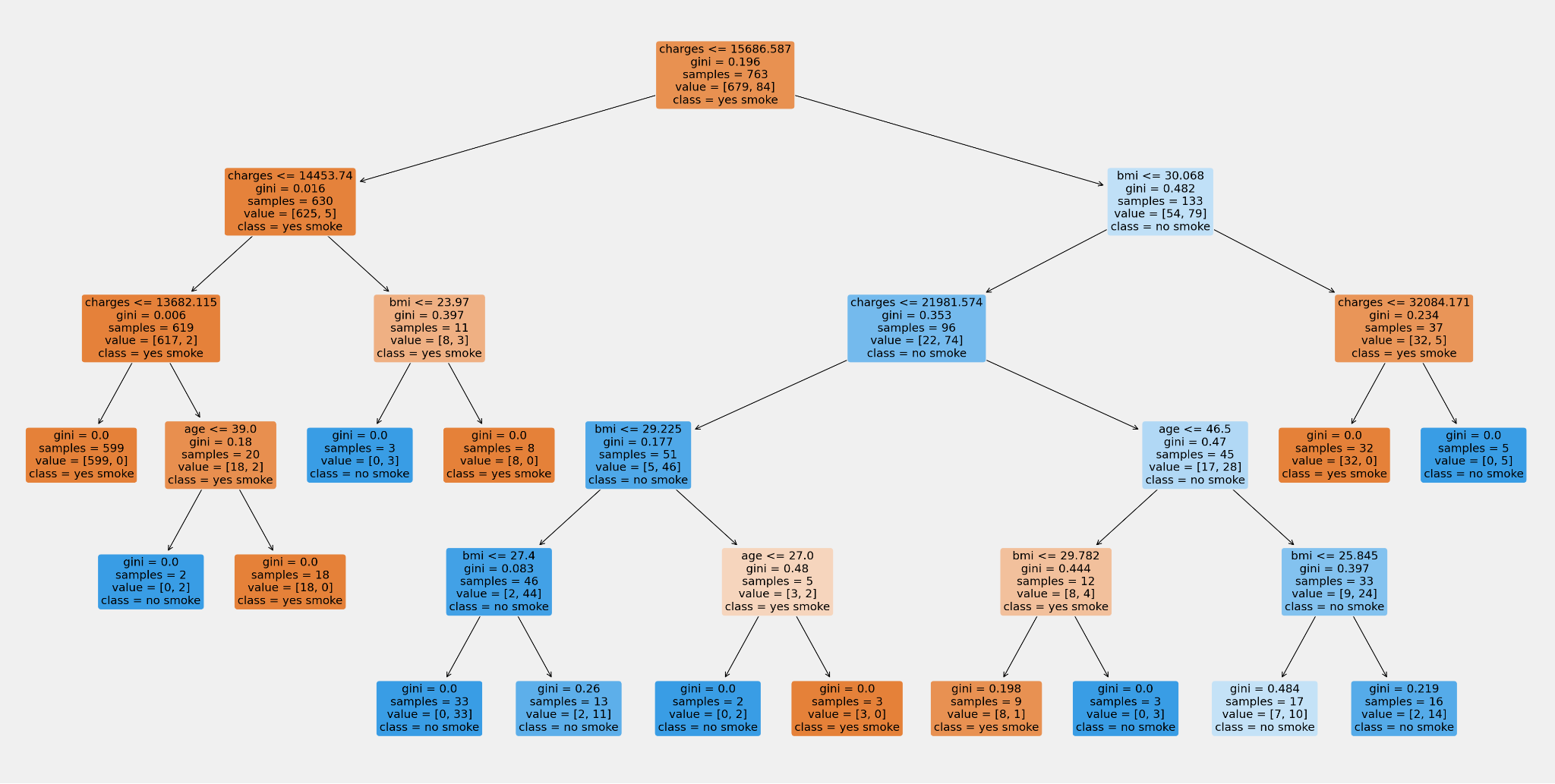
feature\_names = feature\_label)

#save as image

plt.savefig("dt\_smoker", dpi=300, bbox\_inches = "tight")

plt.show()

Chart:



Based on the tree, the most important feature would be “charges” as it is the root node. With the highest sample size, it is also indicative of its importance in making the initial split. The depth of the tree is feasible, encompassing both major and specific levels of predictions. Comparing the features, “bmi” occurs the most frequently as a basis as compared to “age” which makes it a more significant feature in prediction. In the leaf nodes, the impurity is relatively low, signalling more confident predictions. We also see an overwhelming difference between the classes within the node which highlights the confidence in assigning the appropriate class. However, we should note that majority of the leaf nodes have the same class: “no smoke”. This could highlight that the dataset used may have a disproportionate amount of non-smokers compared to smokers, potentially skewing the predictions for smokers.

(148 words)

5)

Decision trees are traditionally used to make predictions by using the input features as a basis to carry out decision-making. The algorithm visualises complex decision making into a more interpretable model, consisting of parent and child nodes. In exploratory data analysis (EDA), the main intention is to gain deeper insights and understanding of the given data. While they both serve conventionally different purposes, we will be discussing if decision trees are feasible in the realm of EDA below.

I am of the opinion that decision trees can be used in EDA, but not as the main driver. Decision trees provide valuable insights such as the relationships between variables and identifying important features. Additionally, decision trees are visualised in a user-friendly way where the model is easily interpreted. It also handles both categorical and numerical data. Hence, we can see that decision trees have a place in carrying out EDA.

However, we do need to consider the drawbacks as well. Decision trees may risk overfitting and are sensitive to changes within the data. Overfitting would hampen EDA as it overly caters to the training data, which makes insights potentially biased. This would be misleading for EDA processes. Likewise, EDA relies on visualisation that is consistent. As decision trees are very sensitive, slight data changes would result in multiple variations of trees which is counterproductive.

In conclusion, decision trees are best used as a supplement to EDA. The potential shines when combined. For instance, in healthcare, decision trees can help identify prominent features that contribute to specific outcomes like ilnesses (McArdle, 2023). Coupling that with EDA methods like histograms, professionals can both understand the data and gain insight on certain variables to place priority on when developing treatments or identifying potential patients. Thus, decision trees would be useful combined with traditional EDA.

(300 words)

References:

McArdle, J. J. (2023). *Exploratory data mining using decision trees in the behavioral sciences.* American Psychological Association. https://psycnet.apa.org/record/2013-32933-001