

**CSE303: Statistics for Data Science**

**Spring 2023**

**Project Report**

**Submitted by:**

|  |  |  |  |
| --- | --- | --- | --- |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |

**Colab Links:**

**Great Customer:** <https://colab.research.google.com/drive/1u4Q6XpKHt_yvrSPfH0Ko0qt3DZnx1kb5?usp=sharing>

**Mobile Price:** <https://colab.research.google.com/drive/1SWtHvZFs5gI8molIcpc39OywyZtwwacu?usp=sharing>

**Heart Disease:** <https://colab.research.google.com/drive/1Mu0GuzWgzeHuJuSHLQobYI4X9ciDVXo5?usp=sharing>

**Introduction:**

In this project, we worked with three datasets: Great Customer Class, Mobile Price Range, and Heart Disease, with the objective of identifying the relevant features to predict their respective target variables.

Data Preprocessing: We began by addressing null values in the datasets by employing suitable methods. Next, we examined the categorical features to determine if they contained any string values, which we converted to numerical representations. We then handled outliers by applying appropriate methods to effectively manage them.

Hypothesis Testing: To identify the relevant independent features for predicting the target variables, we used hypothesis testing. Since the target variables were categorical, we employed the ANOVA (Analysis of Variance) test to calculate p-values between numerical and categorical features. Furthermore, we used the Chi-square test of independence for the association test to determine the p-values. We selected the features with p-values less than 0.5 (at a significance level) as significant predictors.

Mobile Price Range Predictive Modeling: For the Mobile Price Range dataset, we created a predictive model using the decision tree algorithm. This model aimed to predict the price range of mobile phones based on the relevant independent features identified in the previous steps.

In summary, our project involved data preprocessing techniques, hypothesis testing, and predictive modeling to identify relevant features in the three datasets. This approach can be applied to similar datasets for effective predictive modeling.

**Data Preprocessing:**

**Great Customer**

**Handling Missing Values:**

The dataset contains several columns with null values. The percentage of missing values varies across columns, with some columns having only a small percentage of missing values, such as age and salary, while others have a much higher percentage of missing values, such as tea\_per\_year and coffee\_per\_year.

age: 3.095816%

workclass: 3.992941%

salary: 3.103169%

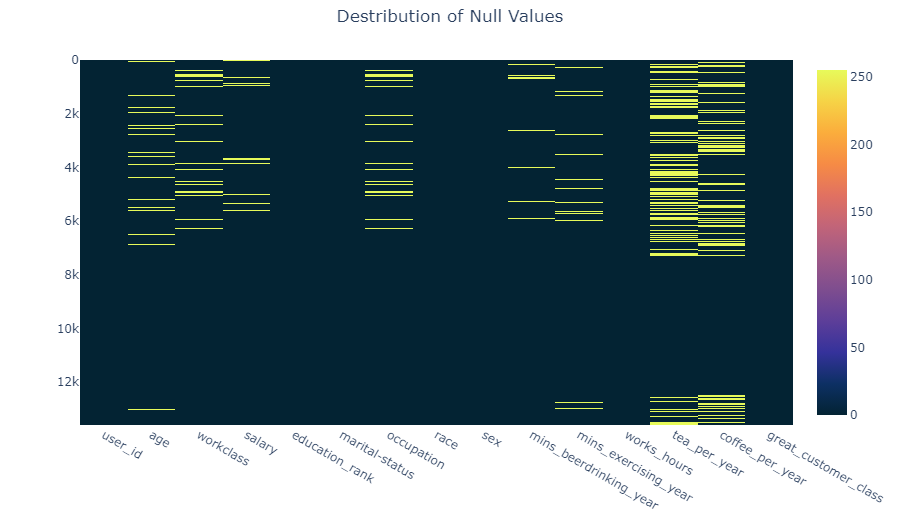
occupation: 3.992941%

mins\_beerdrinking\_year: 3.117876%

mins\_exercising\_year: 3.095816%

tea\_per\_year: 17.861607%

coffee\_per\_year: 17.729245%

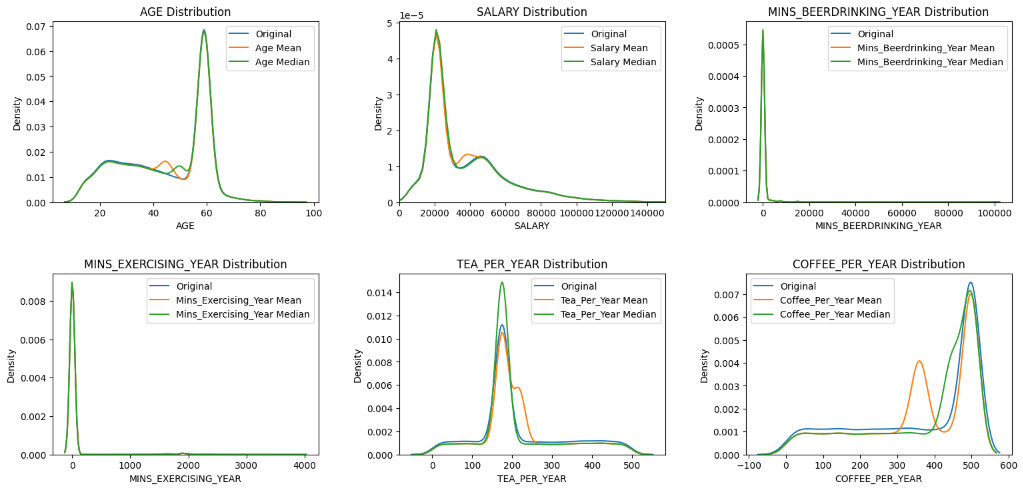


Visualizing the Null Values

This missing data can lead to biased or inaccurate results if not handled appropriately.

**Handling Numerical Features:**

Here age, salary, mins\_beerdrinking\_year, mins\_exercising\_year, tea\_per\_year, coffee\_per\_year this column are numeric columns. We filled up these columns with mean or median. We checked the variance for both mean and median compared it with the main variance.

****

Visualizing the Variance Difference

age Variance: 257.62 (0.00% Difference)

age\_mean Variance: 249.64 (**3.10%** Difference)

age\_median Variance: 250.44 (2.79% Difference)

salary Variance: 591275399.77 (0.00% Difference)

salary\_mean Variance: 572925773.45 (3.10% Difference)

salary\_median Variance: 576722530.07 (2.46% Difference)

mins\_beerdrinking\_year Variance: 22868806.57 (0.00% Difference)

mins\_beerdrinking\_year\_mean Variance: 22155733.03 (3.12% Difference)

mins\_beerdrinking\_year\_median Variance: 22161793.92 (3.09% Difference)

mins\_exercising\_year Variance: 86573.72 (0.00% Difference)

mins\_exercising\_year\_mean Variance: 83893.36 (3.10% Difference)

mins\_exercising\_year\_median Variance: 83956.94 (3.02% Difference)

tea\_per\_year Variance: 12983.33 (0.00% Difference)

tea\_per\_year\_mean Variance: 10664.13 (17.86% Difference)

tea\_per\_year\_median Variance: 10926.37 (15.84% Difference)

coffee\_per\_year Variance: 27119.69 (0.00% Difference)

coffee\_per\_year\_mean Variance: 22311.22 (17.73% Difference)

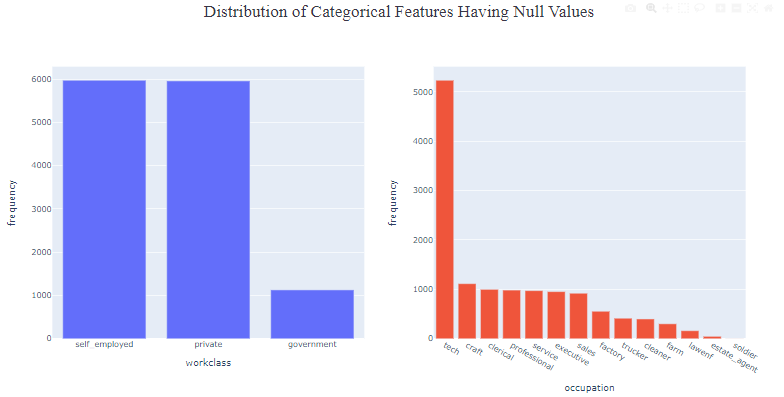
coffee\_per\_year\_median Variance: 23422.74 (13.63% Difference)

Based on the distribution analysis and the % of change in variance it is observed that there are certain columns that demonstrate less variation in distribution after filling the missing values with the mean. These columns include Mins\_BeerDrinking\_Year, Mins\_Exercising\_Year, and Coffee\_Per\_Year.

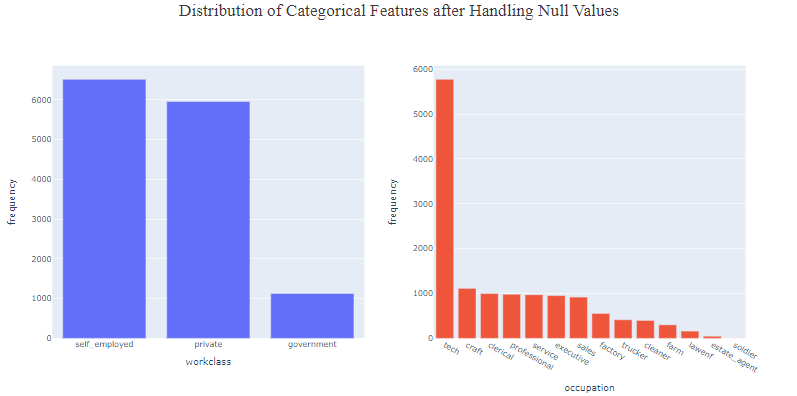
On the other hand, there are other columns such as Age, Salary, and Tea\_Per\_Year, where the distribution shows significantly less variation after filling the missing values with the median. Therefore, we have to use the median to fill the missing values in these columns.

**Handling Categorical Features:**

For categorical features with null values which are workclass and occupation, we used mode to fill up this column.

****

Distribution Difference before Handling Null Values

****

Distribution Difference after Handling Null Values

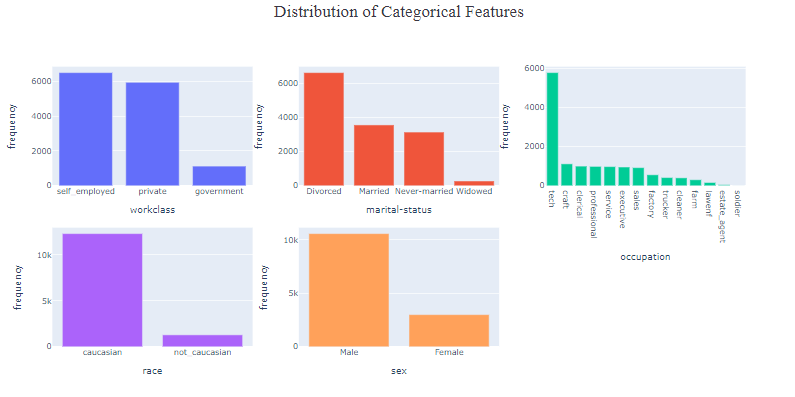
**Encoding:**

In the dataset, categorical columns such as workclass, marital-status, occupation, race, and sex contain string values that represent different categories or classes. In order to use this data for statistical analyses, it is necessary to encode these categorical string columns into numerical values first.

As our string columns are nominal, we used one hot encoding. In this method each category is represented by a binary vector, with a value of 1 indicating the presence of that category and a value of 0 indicating the absence of that category.

As an example,

There are four categories under marital-status feature. So, after encoding this feature convert into 4 new boolean features, which are 'Never-married' 'Divorced' 'Married' 'Widowed'. But we will consider any 3 from these 4 columns. Because it is possible to derive the fourth category from the other three, as it represents individuals who were previously married but are now widowed. By creating separate boolean columns for the remaining three categories, we can better understand the relationships between marital status and the target variable, "great\_customer\_class," in our analysis.



Distribution of Categorical Features

From the above plot we can see there are too many categories in occupation feature. If we use one hot encoding method directly in this feature it will generate a lot of new features resulting in a curse of dimensionality and increase computational complexity. As a result, we can’t use normal one hot encoding method directly to occupation.

To handle occupation feature we grouped the categories in 6 new features. Then we used binary encoding for this feature.

Grouping the categories for the occupation feature:

**Office jobs:** clerical, professional, executive, lawenf, estate\_agent

**Manual jobs:** farm, craft, factory, cleaner, soldier

**Sales jobs:** sales

**Service jobs:** service

**Driver jobs:** trucker

**Tech jobs**: tech

After encoding the dataset will have these new features shown below,

**work\_class:** workclass\_private, workclass\_self\_employed, workclass\_government,

**marital\_status:** marital-status\_Never-married, marital-status\_Divorced, marital-status\_Married, marital-status\_Widowed,

**race:** race\_not\_caucasian, race\_caucasian,

**sex:** sex\_Male, sex\_Female

As mentioned earlier we have to remove one new feature from each category that we encoded. So after this process the dataset will have these following features:

'user\_id', 'age', 'office\_jobs', 'manual\_jobs', 'sales\_jobs', 'service\_jobs', 'Tech\_jobs', 'workclass\_private',

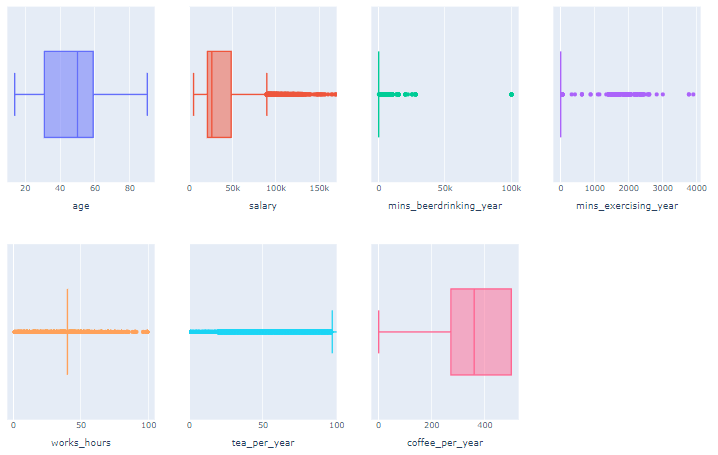
'workclass\_government', 'salary', 'education\_rank', 'marital-status\_Divorced', 'marital-status\_Married',

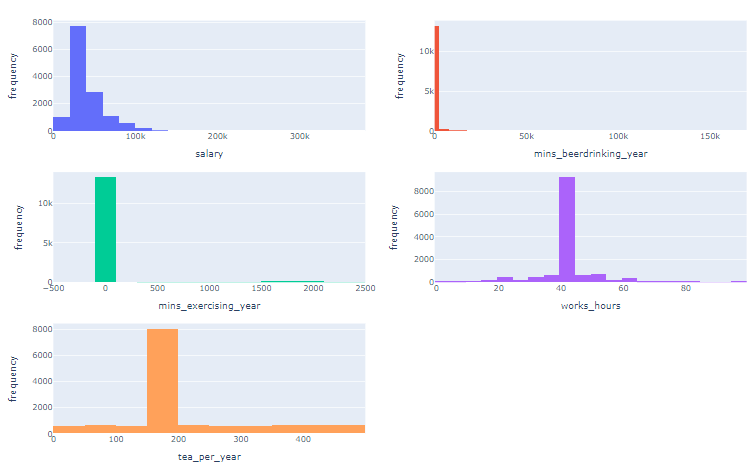
'marital-status\_Widowed', 'race\_caucasian', 'sex\_Male', 'mins\_beerdrinking\_year',

'mins\_exercising\_year', 'works\_hours', 'tea\_per\_year', 'coffee\_per\_year', 'great\_customer\_class'

**Outlier Handling:**

After handling missing values, we handled outliers.



Visualization of the outliers of numerical features

Numerical Columns Distribution before Handling Outliers

**Summary Statistics of features with outliers**:

salary Percentage: 3.75%

salary Mean: 111321.91

salary Median: 102429.62

salary Std: 28982.65

mins\_beerdrinking\_year Percentage: 7.09%

mins\_beerdrinking\_year Mean: 6318.75

mins\_beerdrinking\_year Median: 2174.00

mins\_beerdrinking\_year Std: 16596.03

mins\_exercising\_year Percentage: 5.51%

mins\_exercising\_year Mean: 835.81

mins\_exercising\_year Median: 46.03

mins\_exercising\_year Std: 928.96

works\_hours Percentage: 33.33%

works\_hours Mean: 39.75

works\_hours Median: 42.00

works\_hours Std: 17.04

tea\_per\_year Percentage: 26.83%

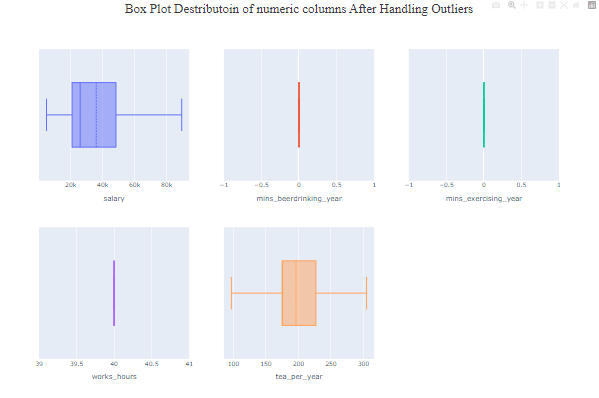
tea\_per\_year Mean: 286.60

tea\_per\_year Median: 356.50

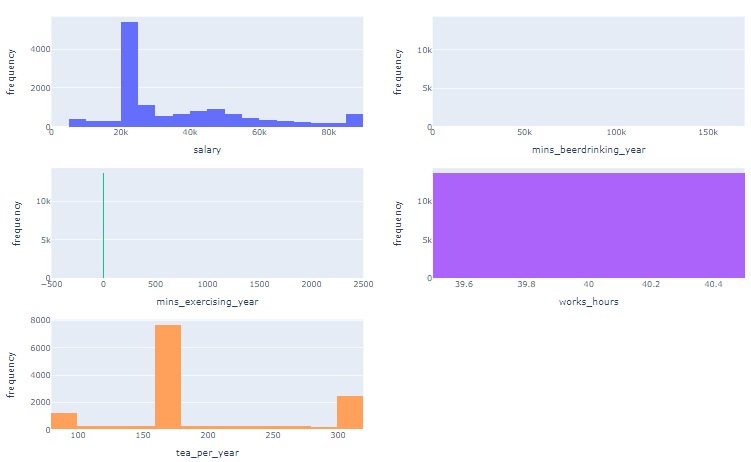
tea\_per\_year Std: 172.66

From the above analysis it is clear that there are a huge number of outliers in salary, mins\_beerdrinking\_year, mins\_exercising\_year, works\_hours, tea\_per\_year columns.

To handle outliers, we used a capping method.



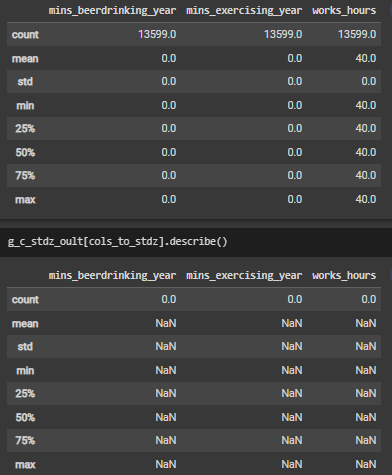
Box Plot Distribution of Numerical Features after Handling Outliers



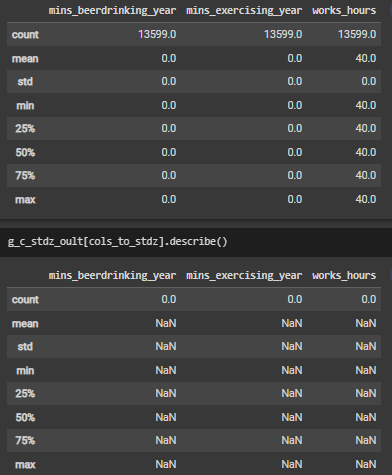
Hist Plot Distribution of Numerical Features after Handling Outliers

As we can see there is extreme change in mins\_beerdrinking\_year, mins\_exercising\_year and works\_hour column. So, it is clear that we have to use other methods to determine if these features are important.

There comes our next choice standardizing the features which have extreme change. Standardizing is a data preprocessing method that scales the data to have a mean of 0 and standard deviation of 1. And the value range is -1 to 1.



Before standardization



After standardization

After standardization we can see that all the values are NaN which means data are missing due to division by standard deviation which was 0.00. So, it’s clear that we can’t use the standardization method.

After using capping method,

Variance of mins\_beerdrinking\_year : 0.0

Variance of mins\_exercising\_year : 0.0

Variance of works\_hours : 0.0

For these conditions we have excluded these three features from the dataset for further analysis.

**Mobile Price**

**Handling Null Values:**

battery\_power 0.0

blue 0.0

clock\_speed 0.0

dual\_sim 0.0

fc 0.0

four\_g 0.0

int\_memory 0.0

m\_dep 0.0

mobile\_wt 0.0

n\_cores 0.0

pc 0.0

px\_height 0.0

px\_width 0.0

ram 0.0

sc\_h 0.0

sc\_w 0.0

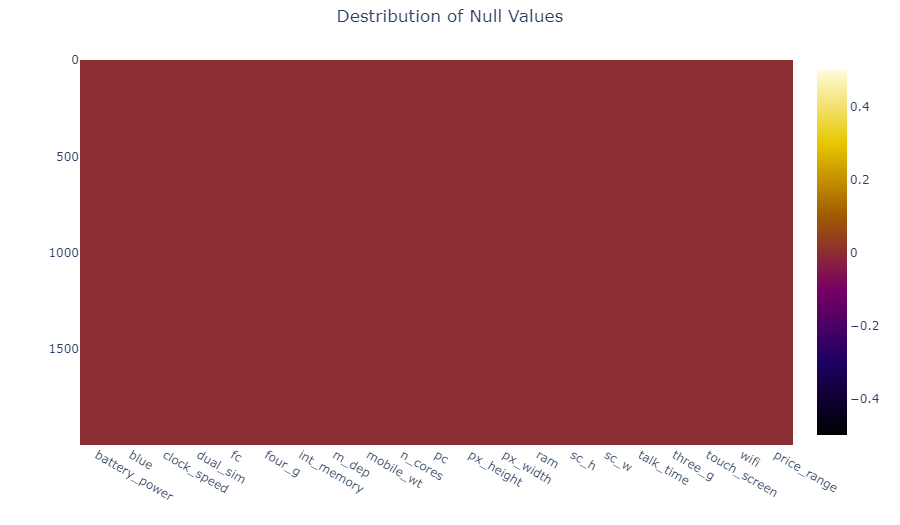
talk\_time 0.0

three\_g 0.0

touch\_screen 0.0

wifi 0.0

price\_range 0.0

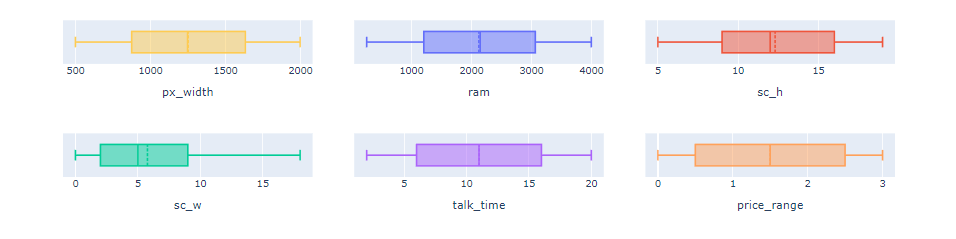


Distribution of Null Values

From the above heatmap we can conclude that, in this dataset there is no null values in any feature.

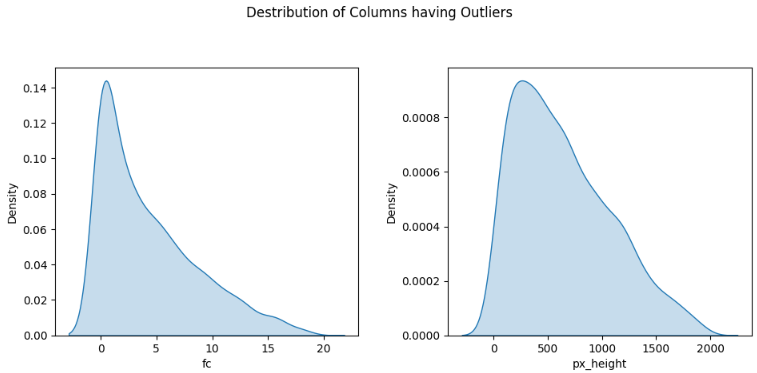
**Handling Outlier:**





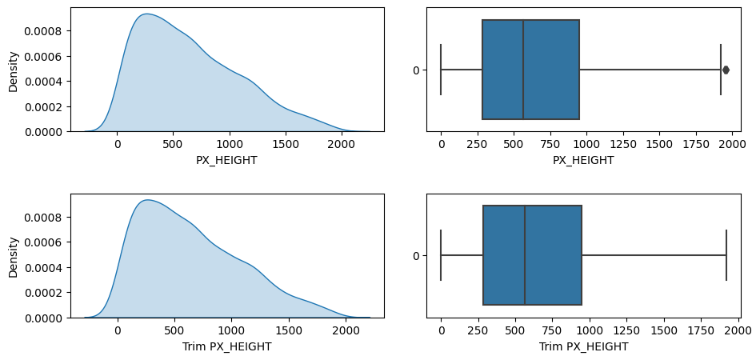
Relationship Between Relevant Features & Price Range

From the graph we can see only two columns has outliers. [fc, px\_height] And they both are rightly skewed. So, we have to use IQR method to handle the outliers in these two columns.



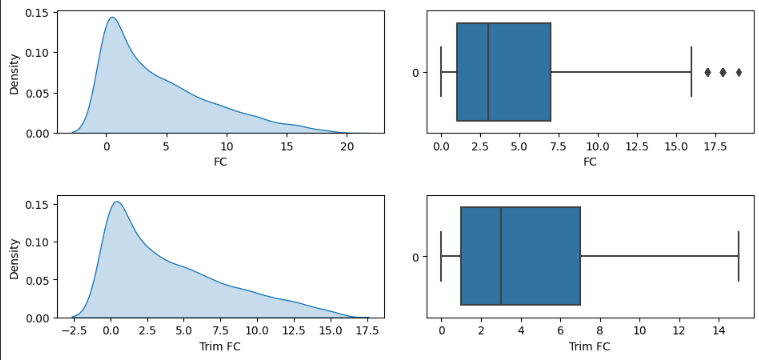
Distribution of Columns Having Outlier

**Using trimming method to handle the outlier in px\_height feature:**



From the box plots we can see that the distribution didn’t change too much.

**Using trimming method to handle the outlier in fc feature:**



Here, there is a noticeable difference between the plots. So, we used capping method to compare and find out the best possible outcome.

**Using capping method to handle the outlier in fc feature:**

Chart

Description automatically generated

After capping we can see that the difference is much less between the trimmed and capped plot.

Based on the above analysis conducted, it can be concluded that the capping method is useful for the FC column, as this feature contains a greater number of outliers. Conversely, trimming may be more appropriate for the Px\_height feature, as it only contains two outliers.



Box Plot After Handling Outliers

**Heart Disease**

**Null / Missing Values Handling:**

Many features in this dataset contains null / missing values.

Percentage of Null values in each feature:

male 0.000000%

age 0.000000%

education 2.477584%

currentSmoker 0.000000%

cigsPerDay 0.684285%

BPMeds 1.250590%

prevalentStroke 0.000000%

prevalentHyp 0.000000%

diabetes 0.000000%

totChol 1.179802%

sysBP 0.000000%

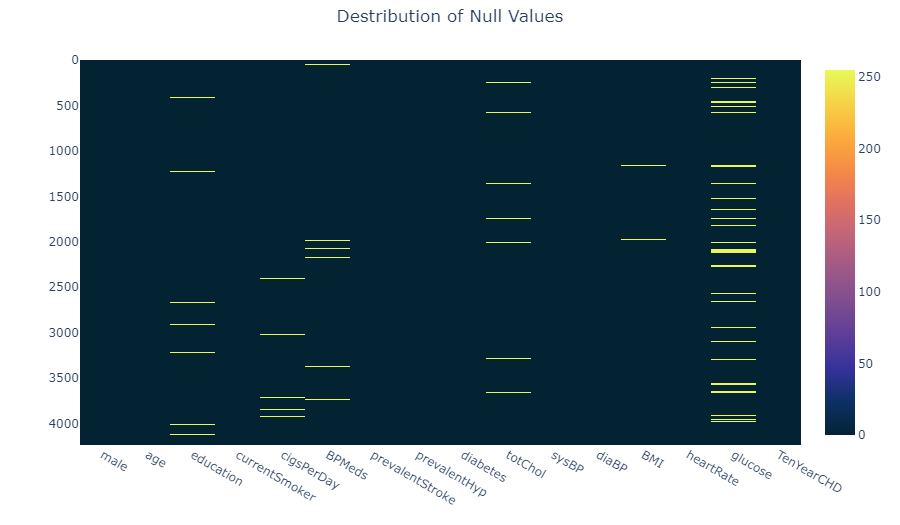
diaBP 0.000000%

BMI 0.448325%

eartrate 0.023596%

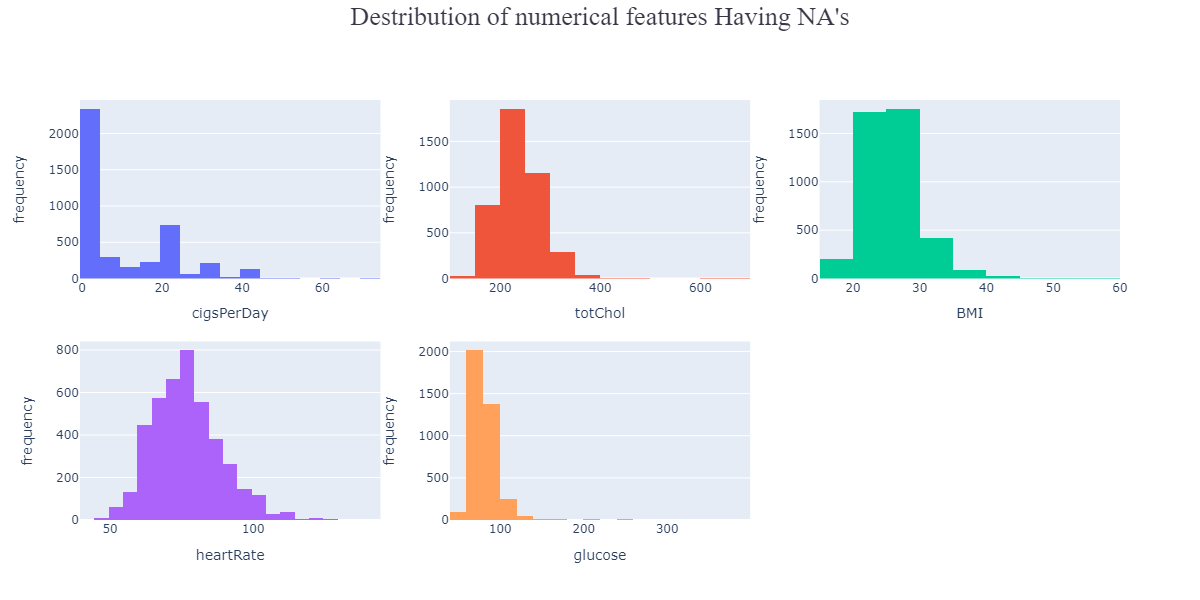
glucose 9.155262%

TenYearCHD 0.000000%



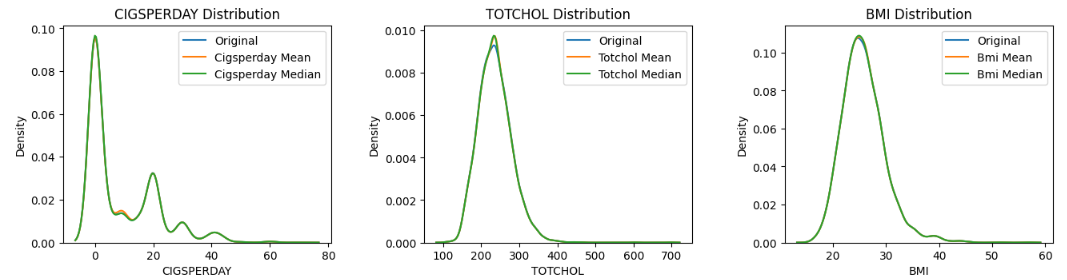
Plotting the Missing Values

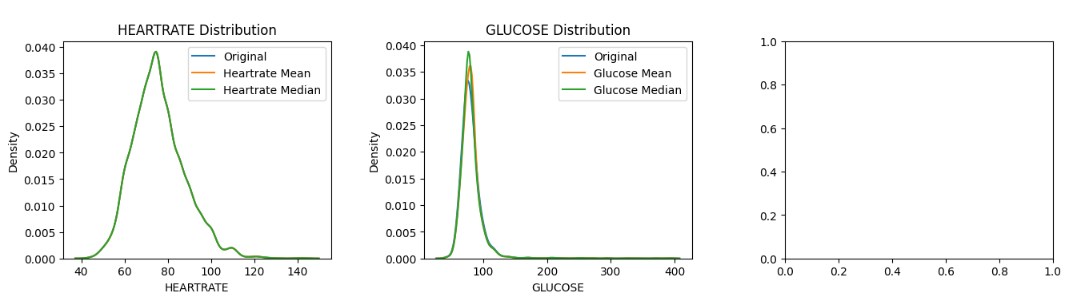
Here we can see education, cigsPerDay, BPMeds, totChol, BMI, eartrate, glucose these feature contains null values.



Distribution of Numerical Features Having NA’s

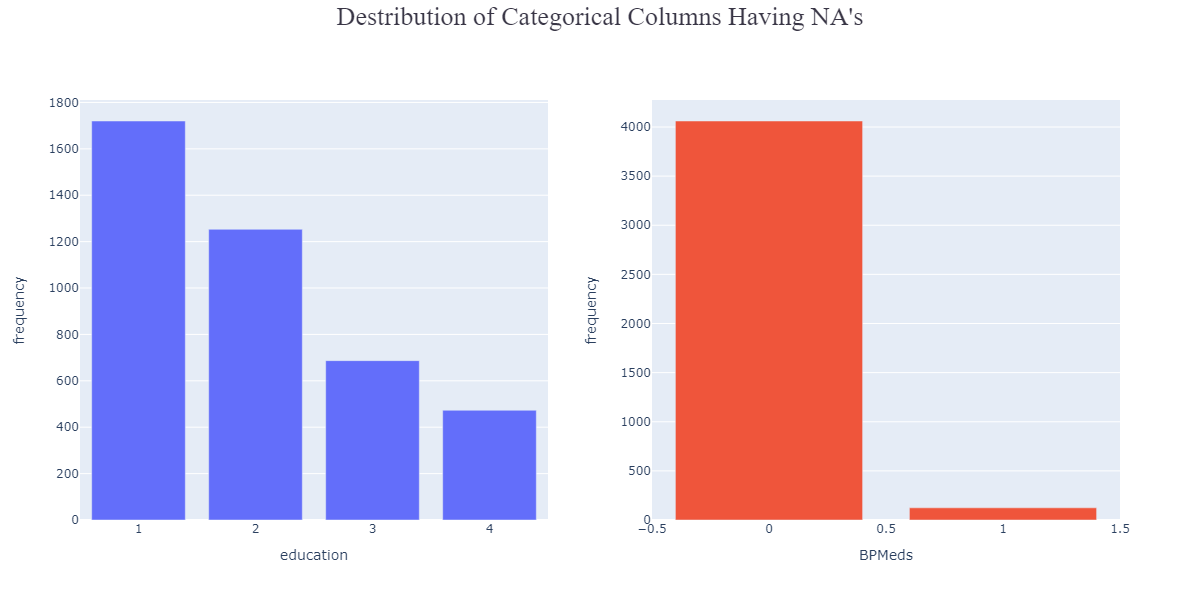
To handle the null values in numerical features we used both mean and median, and compared them to find out the best possible outcomes.





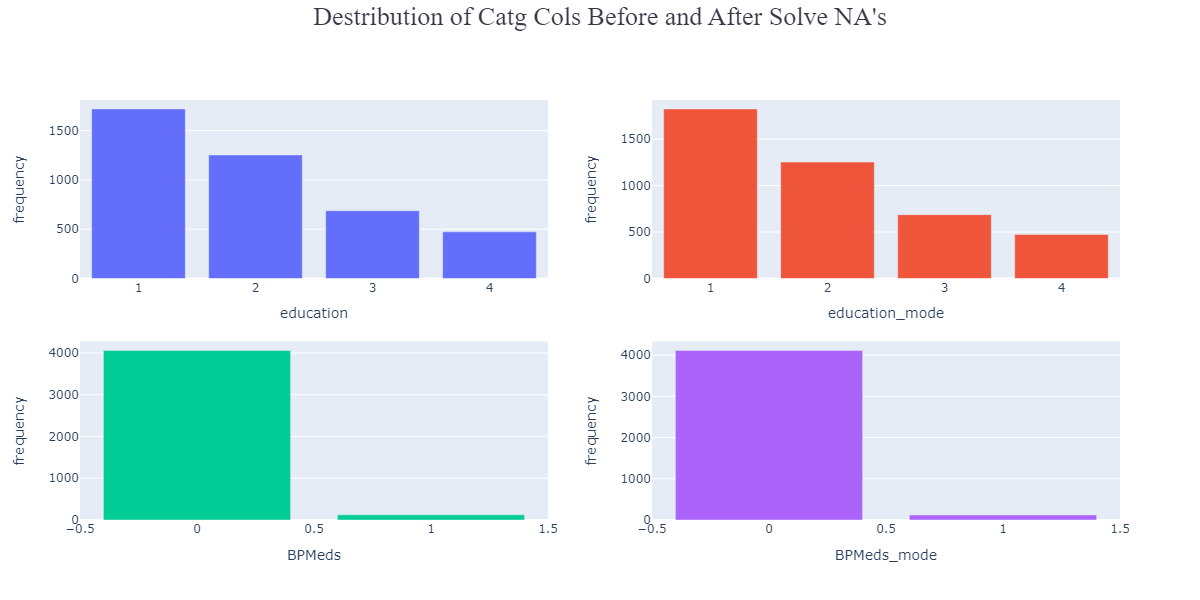
Based on the distribution analysis, it is observed that there are certain columns that demonstrate less variation in distribution after filling the missing values with the mean. These columns include totchol, bmi, heartrate, glucose.

On the other hand, in cigsperday feature the distribution shows significantly less variation after filling the missing values with the median. Therefore, we have to use the median to fill the missing values in these columns.



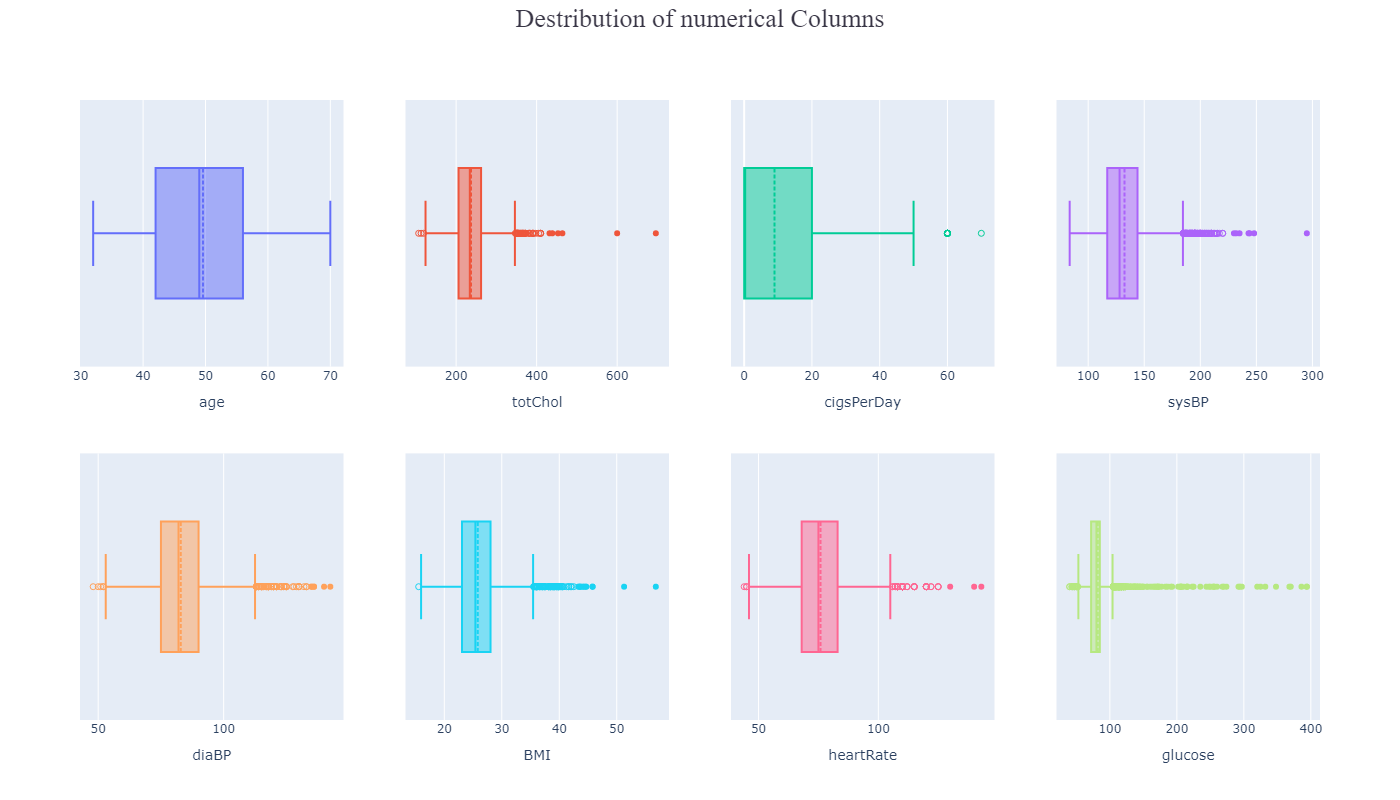
Distribution of Categorical Features Having NA’s

To handle the null values in categorical feature we used mode to fill those cells up.



Distribution of Categorical Features Before & After Solving NA’s

**Handling Outliers:**



Distribution of Numerical Features

**Summery Statistics of outliers in each feature:**

Age Percentage: 0.00%

TotChol Percentage: 1.34%

TotChol Mean: 371.23

TotChol Median: 366.00

TotChol Std: 83.46

CigsPerDay Percentage: 0.28%

CigsPerDay Mean: 60.83

CigsPerDay Median: 60.00

CigsPerDay Std: 2.76

SysBP Percentage: 2.97%

SysBP Mean: 199.27

SysBP Median: 195.50

SysBP Std: 15.21

DiaBP Percentage: 1.91%

DiaBP Mean: 116.90

DiaBP Median: 119.00

DiaBP Std: 18.42

BMI Percentage: 2.29%

BMI Mean: 38.97

BMI Median: 38.54

BMI Std: 4.10

HeartRate Percentage: 1.79%

HeartRate Mean: 110.28

HeartRate Median: 110.00

HeartRate Std: 15.00

Glucose Percentage: 6.18%

Glucose Mean: 135.73

Glucose Median: 116.00

Glucose Std: 62.25

From the above analysis we can see feature that have outliers are 'totChol', 'cigsPerDay', 'totChol', 'sysBP', 'diaBP', 'BMI', 'heartRate', 'glucose'.

To handle the outlier we used capping method.



Distribution of Numerical Features After Solving Outliers

**Hypothesis Test:**

**Great Customer**

Hypothesis testing can be performed on this dataset to determine the statistical significance of the relationship between the target variable 'great\_customer\_class' and other features. One possible hypothesis test is to perform a chi-square test of the association for the categorical features and an ANOVA test for the numerical features to determine whether there is a significant difference in the mean target variable across different levels of the categorical features such as 'workclass', 'marital-status', 'occupation', 'race', and 'sex'.

The hypothesis test can be formulated as follows:

Null Hypothesis: There is no significant difference in the mean 'great\_customer\_class' across different levels of the categorical features in the dataset.

Alternative Hypothesis: There is a significant difference in the mean 'great\_customer\_class' across different levels of the categorical features in the dataset.

Anova test analyzes the variance between different categories or groups to check if the differences between the category means are significant. It is done by calculating the ratio of the variance between groups to the variance within groups, which is called F-statistic. If the F-statistic is greater than the critical value, then we can reject the null hypothesis, which means at least one group is significantly different than other groups.

**Mobile Price Range**

On this dataset hypothesis testing can be performed to determine the statistical significance of the relationship between the target feature prize\_range. By performing Anova(for numerical columns) and Chi-square (for categorical columns) tests it is possible to test hypothesis for this dataset. If there is no significant difference in the mean ‘prize\_range’ across different levels of categorical features in the data then it is Null Hypothesis, and if there is a significant difference then we call it alternative hypothesis.

We can do Anova test for following features: 'battery\_power', 'clock\_speed', 'fc', 'int\_memory', 'm\_dep', 'mobile\_wt', 'n\_cores', 'pc', 'px\_height', 'px\_width', 'ram', 'sc\_h', 'sc\_w', 'talk\_time', 'price\_range'.

**Heart Disease**

On this dataset hypothesis testing can be performed to determine the statistical significance of the relationship between the target feature TenYearCHD(risk of developing Coronary Heart Disease within the Next Ten Years). By performing Anova(for numerical columns) and Chi-square (for categorical columns) tests it is possible to test hypotheses for this dataset.

If there is no significant difference in the mean ‘TenYearCHD’ across different levels of categorical features in the data then it is Null Hypothesis, and if there is a significant difference then we call it alternative hypothesis.

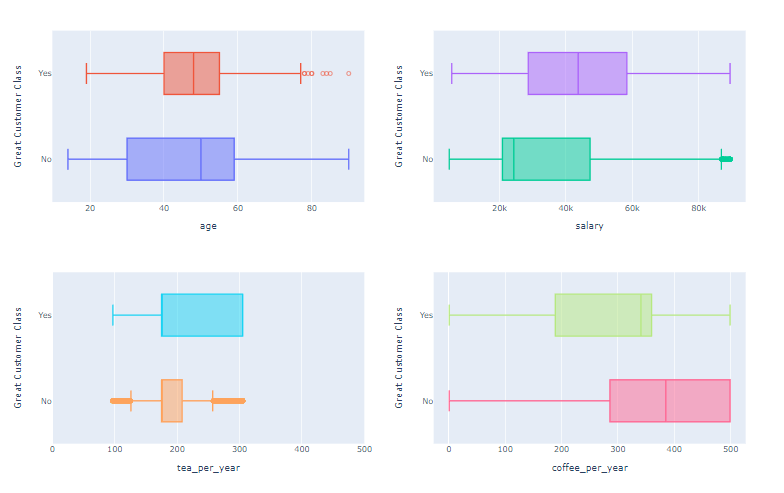
**P-value, Test of Association:**

**Great Customer**

A test of association means a test of independence or a chi-square test. It is a statistical test used to determine whether two categorical features are associated with each other or not. In chi-square the null hypothesis is that there is no association between the two variables. And the alternate hypothesis is that there is a significant association between the two variables.

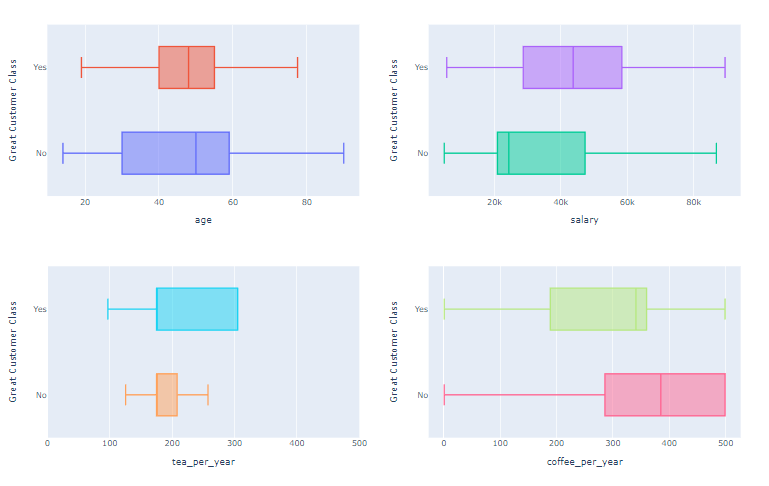
Chi-square test calculates a p-value, which helps to observe the association between the variables if the null hypothesis is true. If the p-values are low it means the null hypothesis is rejected.

In the dataset the target feature is "great\_customer\_class" and the categorical predictor features are ‘office\_jobs', 'manual\_jobs', 'sales\_jobs', 'service\_jobs', 'Tech\_jobs', 'workclass\_private', 'workclass\_government', 'education\_rank', 'marital-status\_Divorced', 'marital-status\_Married', 'marital-status\_Widowed', 'race\_caucasian', 'sex\_Male', 'great\_customer\_class'.



**Relationship between Numerical Features and Great Customer Class**

**After handling the outlier using capping method:**

****

**Relationship between Numerical Features and Great Customer Class After handling Outlier**

****

**Relationship between Categorical Features and Great Customer Class**

**Comparing the result of our Anova Test and Chi Square test to build in functions:**

**Anova:**

**age**

stat = 41.07733519698571, p\_value = 1.511501226582168e-10

my\_stat = 41.07733519698485, my\_p\_value = 1.511501226582168e-10

**salary**

stat = 287.8010578033135, p\_value = 6.799046308534331e-64

my\_stat = 287.80105780331127, my\_p\_value = 6.79904630853974e-64

**tea\_per\_year**

stat = 257.4107076172952, p\_value = 2.115958342152212e-57

my\_stat = 257.4107076172933, my\_p\_value = 2.115958342153896e-57

**Chi Square:**

**office\_jobs**

stat = 947.1691670031346, p\_value = 5.4713896157168446e-208

my\_stat = 947.1691670031346, my\_p\_value = 0.0

**manual\_jobs**

stat = 1.2597585965627767, p\_value = 0.2616967899077225

my\_stat = 1.2597585965627764, my\_p\_value = 0.2616967899077257

**sales\_jobs**

stat = 43.271700834306515, p\_value = 4.764243869439441e-11

my\_stat = 43.27170083430652, my\_p\_value = 4.764244554422703e-11

**service\_jobs**

stat = 62.48075234087651, p\_value = 2.690614379798907e-15

my\_stat = 62.480752340876506, my\_p\_value = 2.6645352591003757e-15

**Tech\_jobs**

stat = 696.202521882976, p\_value = 2.0021375665855697e-153

my\_stat = 696.202521882976, my\_p\_value = 0.0

**workclass\_private**

stat = 181.42132534949388, p\_value = 2.3718866118252305e-41

my\_stat = 181.42132534949386, my\_p\_value = 0.0

**workclass\_government**

stat = 126.21262757737627, p\_value = 2.762418448957248e-29

my\_stat = 126.21262757737627, my\_p\_value = 0.0

**education\_rank**

stat = 2028.2817524297695, p\_value = 0.0

my\_stat = 2028.2817524297698, my\_p\_value = 0.0

**marital-status\_Divorced**

stat = 881.8808654707948, p\_value = 8.526037551126789e-194

my\_stat = 881.8808654707948, my\_p\_value = 0.0

**marital-status\_Married**

stat = 2377.4212220155305, p\_value = 0.0

my\_stat = 2377.4212220155305, my\_p\_value = 0.0

**marital-status\_Widowed**

stat = 8.689087533219803, p\_value = 0.003201208453255956

my\_stat = 8.689087533219803, my\_p\_value = 0.0032012084532559992

**race\_caucasian**

stat = 1.1951838710199714, p\_value = 0.27428639646303465

my\_stat = 1.1951838710199716, my\_p\_value = 0.27428639646303776

**sex\_Male**

stat = 50.00942946582381, p\_value = 1.5300891364406532e-12

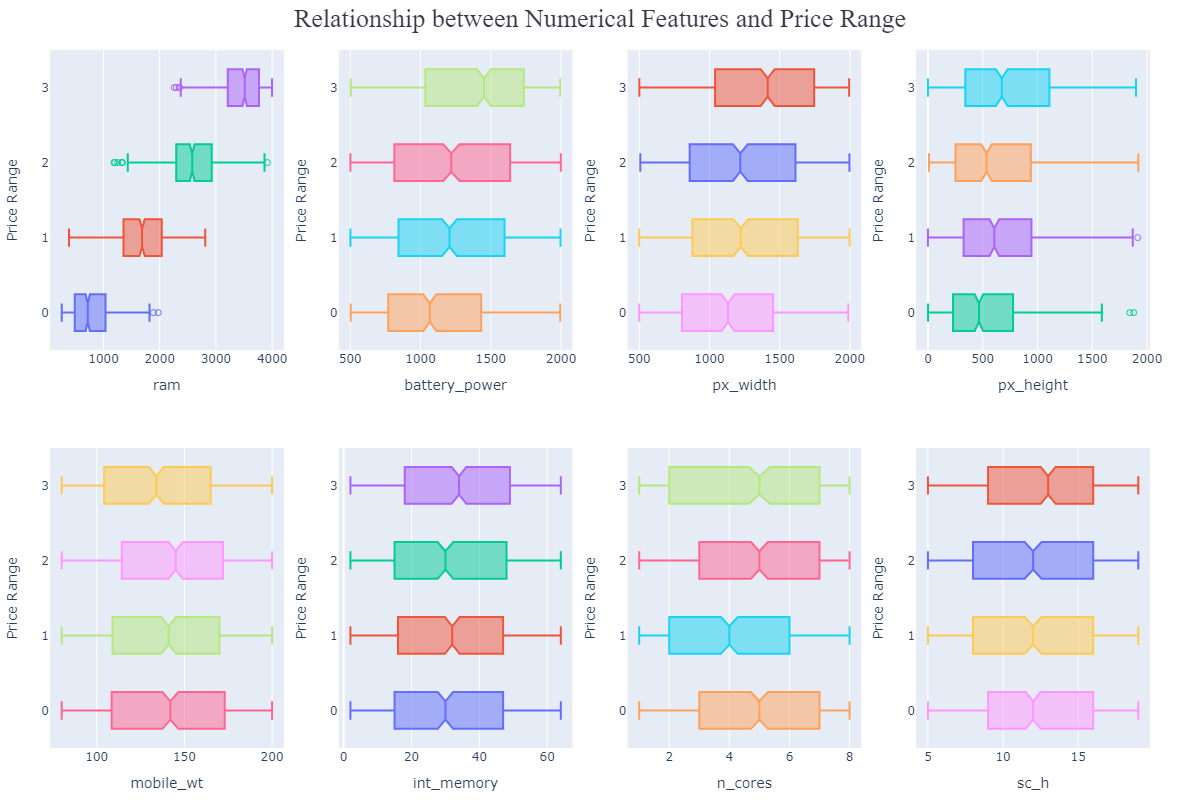
my\_stat = 50.00942946582381, my\_p\_value = 1.5301093725383907e-12

**Mobile Price Range**

For Chi-square test in this dataset we have following features:

blue, dual\_sim, four\_g, three\_g, touch\_screen, wifi.

And for Anova test we have following features:

ram, battery\_power, px\_width, px\_height, mobile\_wt, int\_memory, n\_cores, sc\_h

Relationship Between Numerical Features & Price Range



Relationship Between Categorical Features & Price Range

**Comparing Our Anova Function and Chi-square Function with Build-in Anova Function:**

**Anova:**

**battery\_power**

stat = 31.513495036295836, p\_value = 6.713993985856411e-20

my\_stat = 31.513495036295783, my\_p\_value = 6.713993985856411e-20

**clock\_speed**

stat = 0.491978773388441, p\_value = 0.6878760540629911

my\_stat = 0.49197877338861495, my\_p\_value = 0.6878760540628887

**fc**

stat = 0.8412347075208211, p\_value = 0.47119535676382607

my\_stat = 0.8412347075208505, my\_p\_value = 0.47119535676382607

**int\_memory**

stat = 2.9131423673657735, p\_value = 0.03321699701567197

my\_stat = 2.913142367365822, my\_p\_value = 0.033216997015667736

**m\_dep**

stat = 1.5181696567539429, p\_value = 0.20783366859699934

my\_stat = 1.5181696567539407, my\_p\_value = 0.20783366859699934

**mobile\_wt**

stat = 3.57796808786005, p\_value = 0.013415544766487946

my\_stat = 3.5779680878599414, my\_p\_value = 0.013415544766489661

**n\_cores**

stat = 2.576356312579159, p\_value = 0.052259228034057296

my\_stat = 2.5763563125790894, my\_p\_value = 0.05225922803406589

**pc**

stat = 0.8581096270002767, p\_value = 0.4621872664942891

my\_stat = 0.8581096270003586, my\_p\_value = 0.4621872664942498

**px\_height**

stat = 19.256805801002372, p\_value = 2.6162526915481838e-12

my\_stat = 19.25680580100237, my\_p\_value = 2.6162526915481838e-12

**px\_width**

stat = 22.405610912881148, p\_value = 2.881567837380335e-14

my\_stat = 22.405610912881055, my\_p\_value = 2.881567837380949e-14

**ram**

stat = 3527.774583085812, p\_value = 0.0

my\_stat = 3527.774583085812, my\_p\_value = 0.0

**sc\_h**

stat = 2.2300733023906982, p\_value = 0.0828023294418198

my\_stat = 2.2300733023906734, my\_p\_value = 0.08280232944182207

**sc\_w**

stat = 1.673070617080054, p\_value = 0.17076307782254166

my\_stat = 1.6730706170800698, my\_p\_value = 0.17076307782254166

**talk\_time**

stat = 1.6254402003698731, p\_value = 0.18144519465488843

my\_stat = 1.6254402003699502, my\_p\_value = 0.18144519465487718

**Chi-square:**

**blue**

stat = 1.379985190664331, p\_value = 0.7102329471672442

my\_stat = 1.379985190664331, my\_p\_value = 0.7102329471672442

**dual\_sim**

stat = 1.1650734607420992, p\_value = 0.7613927398347171

my\_stat = 1.1650734607420992, my\_p\_value = 0.7613927398347171

**four\_g**

stat = 3.1839157849182875, p\_value = 0.3641289090432397

my\_stat = 3.183915784918287, my\_p\_value = 0.3641289090432396

**three\_g**

stat = 1.3194817939999668, p\_value = 0.7245122123530843

my\_stat = 1.3194817939999668, my\_p\_value = 0.7245122123530843

**touch\_screen**

stat = 3.9919597018810995, p\_value = 0.26233364677575377

my\_stat = 3.991959701881099, my\_p\_value = 0.262333646775754

**wifi**

stat = 0.8039541146769342, p\_value = 0.8485210293812833

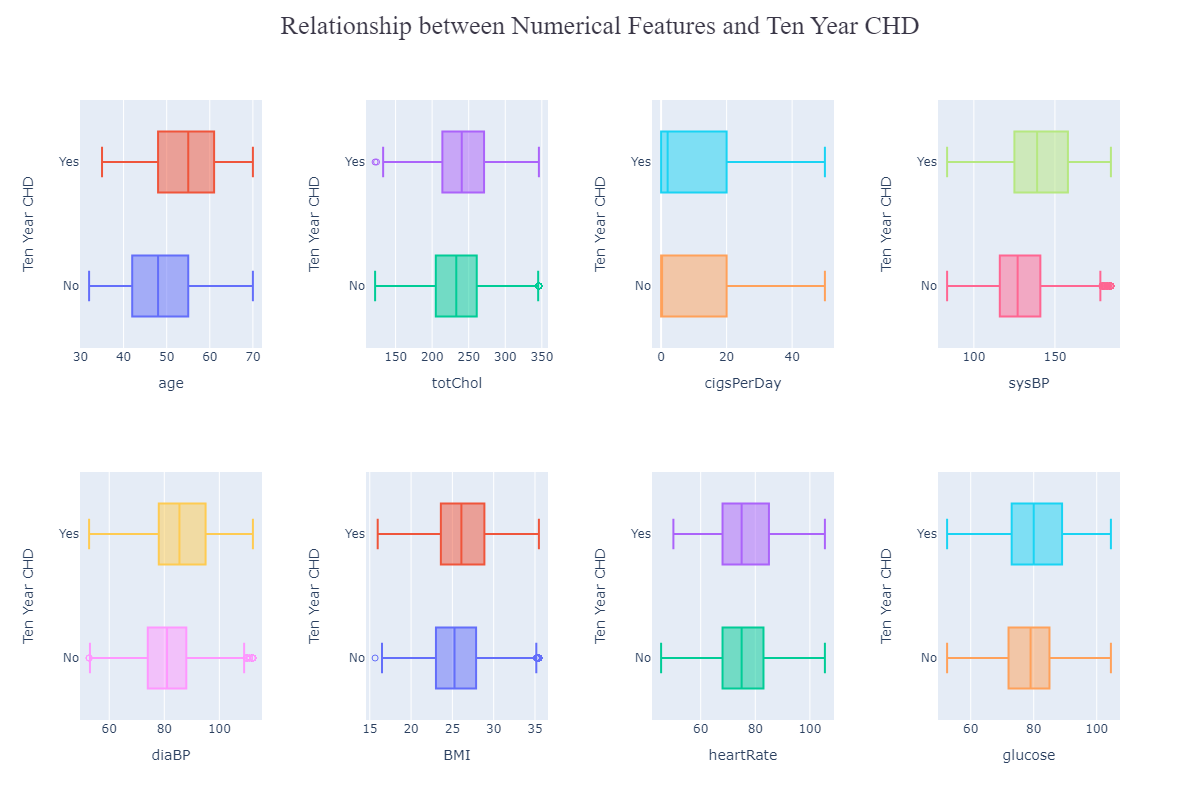
my\_stat = 0.8039541146769342, my\_p\_value = 0.8485210293812833

**Heart Disease**

For Chi-square test we have following features:

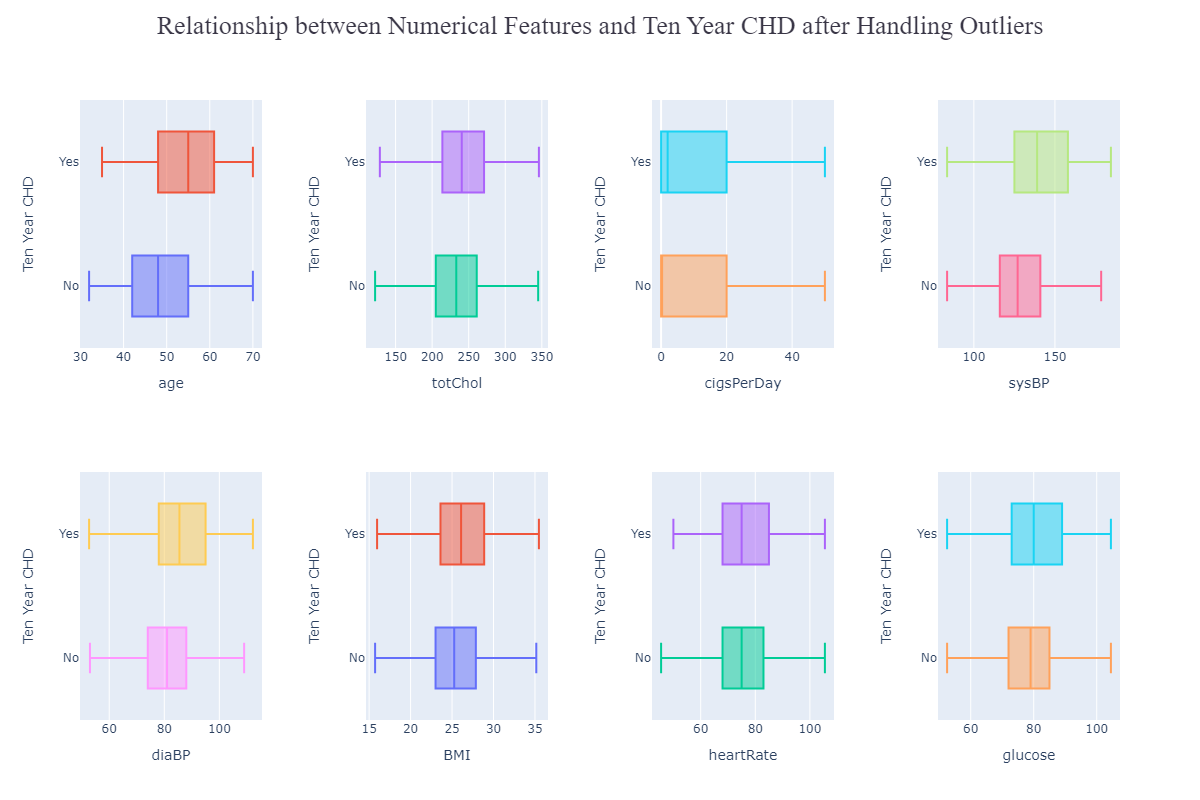
male, education, currentSmoker, BPMeds, prevalentStroke, prevalentHyp, diabetes.

For the Anova test we have following features:

age, totChol, cigsPerDay, sysBP, diaBP, BMI, heartRate, glucose

Relationship Between Numerical Featured & Ten Year CHD

From the above distribution we see that feature having outlier are 'totChol’,'sysBP', 'diaBP', 'BMI'. As a result, we used capping method to fill up the outlier with respective upper and lower limit.



Relationship Between Numerical Features & Ten Year CHD after Handling Outliers



Relationship Between Categorical Features & Ten CHD

**Comparing Our Anova Test function wth Built-in Anova Fuction:**

**age**

stat = 226.42482952300466, p\_value = 6.8450085874350565e-50

my\_stat = 226.42482952300432, my\_p\_value = 6.8450085874350565e-50

**totChol**

stat = 26.896520053510475, p\_value = 2.246671581802369e-07

my\_stat = 26.89652005351028, my\_p\_value = 2.246671581802369e-07

**cigsPerDay**

stat = 15.025334270599298, p\_value = 0.0001076833974413811

my\_stat = 15.025334270599334, my\_p\_value = 0.0001076833974413811

**sysBP**

stat = 206.41830554811094, p\_value = 9.716367662190807e-46

my\_stat = 206.41830554811006, my\_p\_value = 9.716367662195639e-46

**diaBP**

stat = 88.71014175059128, p\_value = 7.301881020945315e-21

my\_stat = 88.71014175059173, my\_p\_value = 7.301881020941683e-21

**BMI**

stat = 22.486967523452538, p\_value = 2.185287570160557e-06

my\_stat = 22.48696752345222, my\_p\_value = 2.185287570160557e-06

**heartRate**

stat = 2.216695482358314, p\_value = 0.13659944260506415

my\_stat = 2.2166954823592198, my\_p\_value = 0.13659944260509999

**Comparing Our Chi-square Function with Built-in Chi-square Function:**

**male**

stat = 33.13876412178594, p\_value = 8.581080129462392e-09

my\_stat = 33.13876412178594, my\_p\_value = 8.581080179759226e-09

**education**

stat = 30.93552881168504, p\_value = 8.770368701361081e-07

my\_stat = 30.935528811685035, my\_p\_value = 8.770368701283004e-07

**currentSmoker**

stat = 1.6042792098599596, p\_value = 0.20529783701529147

my\_stat = 1.6042792098599596, my\_p\_value = 0.20529783701529503

**BPMeds**

stat = 31.649053344312318, p\_value = 1.8470461189725918e-08

my\_stat = 31.649053344312318, my\_p\_value = 1.8470461160546847e-08

**prevalentStroke**

stat = 16.191149461946694, p\_value = 5.726102177501766e-05

my\_stat = 16.191149461946694, my\_p\_value = 5.7261021775056875e-05

**prevalentHyp**

stat = 133.6780899180762, p\_value = 6.425269735903649e-31

my\_stat = 133.6780899180762, my\_p\_value = 0.0

**diabetes**

stat = 40.135995000258504, p\_value = 2.368836764784068e-10

my\_stat = 40.135995000258504, my\_p\_value = 2.368836238275662e-10

**Feature Selection:**

**Great Customer**

After performing Anova Test between each numerical feature and target feature (great\_customer\_class) and chi square test of association between each categorical feature and target feature we get the following p values in order (low - high):

office\_jobs: 0.0

Tech\_jobs: 0.0

workclass\_private: 0.0

workclass\_government: 0.0

education\_rank: 0.0

marital-status\_Divorced: 0.0

marital-status\_Married: 0.0

salary: 6.799046308534331e-64

tea\_per\_year: 2.115958342152212e-57

service\_jobs: 2.6645352591003757e-15

sex\_Male: 1.5301093725383907e-12

sales\_jobs: 4.764244554422703e-11

age: 1.511501226582168e-10

marital-status\_Widowed: 0.0032012084532559992

manual\_jobs: 0.2616967899077257

race\_caucasian: 0.27428639646303776

Based on the 0.5 significance level the analysis revealed that several features are highly significant predictors of the 'great\_customer\_class', including 'office\_jobs', 'Tech\_jobs', 'workclass\_private', 'workclass\_government', 'education\_rank', 'marital-status\_Divorced', and 'marital-status\_Married'.

Additionally, features such as 'salary', 'tea\_per\_year', 'service\_jobs', 'sex\_Male', 'sales\_jobs', and 'age' showed moderate significance.

**Mobile Price**

ram: 0.0

battery\_power: 6.713993985856411e-20

px\_width: 2.881567837380335e-14

px\_height: 2.6162526915481838e-12

mobile\_wt: 0.013415544766487946

int\_memory: 0.03321699701567197

n\_cores: 0.052259228034057296

sc\_h: 0.0828023294418198

sc\_w: 0.17076307782254166

talk\_time: 0.18144519465488843

m\_dep: 0.20783366859699934

touch\_screen: 0.262333646775754

four\_g: 0.3641289090432396

pc: 0.4621872664942891

fc: 0.47119535676382607

clock\_speed: 0.6878760540629911

blue: 0.7102329471672442

three\_g: 0.7245122123530843

dual\_sim: 0.7613927398347171

wifi: 0.8485210293812833

We Have to select the top features with the smallest p-values for training the model.

Ram and battery\_power have the smallest p-values, indicating a strong relationship with the

target variable.

Px\_width and px\_height also have small p-values and are statistically significant, but to a

slightly lesser extent.

Other features have larger p-values and may not have a strong relationship with the target

variable.

As a result we select ram, battery\_power, px\_width, px\_height, mobile\_wt, int\_memory ,

n\_cores and sc\_h as our top features for training the model.

**Heart Disease**

prevalentHyp: 0.0

age: 6.8450085874350565e-50

sysBP: 9.716367662190807e-46

diaBP: 7.301881020945315e-21

diabetes: 2.368836238275662e-10

male: 8.581080179759226e-09

BPMeds: 1.8470461160546847e-08

totChol: 2.246671581802369e-07

education: 8.770368701283004e-07

BMI: 2.185287570160557e-06

prevalentStroke: 5.7261021775056875e-05

cigsPerDay: 0.0001076833974413811

heartRate: 0.13659944260506415

currentSmoker: 0.20529783701529503

The null hypothesis for each p-value is that there is no relationship between the feature and

TenYearCHD. Therefore, if the p-value is less than the significance level (usually 0.05), we can

reject the null hypothesis and conclude that there is a significant relationship between the feature

and TenYearCHD.

From the above p-values, we can see that all the features except &#39;heartRate&#39; and &#39;currentSmoker&#39;

have p-values less than 0.05. This means that these features are statistically significant and have

a significant relationship with TenYearCHD.

Therefore, the significant features based on the given p-values are:

prevalentHyp, age, sysBP, diaBP, diabetes, male, BPMeds, totChol, education, BMI,

prevalentStroke, cigsPerDay.

We can formally state that there is a statistically significant relationship between TenYearCHD

and the above mentioned features at a significance level of 0.05.

**Relevant Exploratory Data Analysis:**

**Great Customer**

1. Distribution of age among the great customers

A picture containing diagram, screenshot, plot, pixel

Description automatically generated

2. Is there a correlation between an individual's age and salary?

A black squares with white text

Description automatically generated with low confidence

A picture containing screenshot, text, colorfulness, line

Description automatically generated

From the above correlation matrix and the plot, we can conclude that there is a very week positive correlation between age and salary. As a result, we can say both features are not similar and each provide unique stats

3. Is there a significant difference in the average salary between males and females.

Ans: (23.657253883780122, 7.330899247195383e-117)

The t-test performed indicates a statistically significant difference between the average salaries of males and females. The obtained t-statistic of 23.66 and p-value of 7.33e-117 suggest that the difference in salary between males and females is not likely due to chance. Therefore, we reject the null hypothesis and conclude that there is a significant difference in the average salary between males and females.

4. What is the most common occupation among individuals who are self-employed

A picture containing text, screenshot, diagram, number

Description automatically generated

Based on the bar plot, it appears that Tech\_jobs is the most common occupation among self-employed individuals. This may suggest that there is a growing trend towards self-employment in the technology industry.

5. Are individuals who work in government jobs more likely to consume coffee or tea, and how does this vary by race?

A picture containing text, screenshot, diagram, font

Description automatically generated

From above plot we can see that, it appears that both non-Caucasian and Caucasian individuals working in government jobs consume coffee more frequently than tea. However, the average coffee consumption among Caucasians appears to be slightly higher than non-Caucasians, with a mean value of 283.9708 compared to 260.54. On the other hand, non-Caucasians appear to consume slightly more tea on average, with a mean value of 204.3896 compared to 208.8661 for Caucasians. However, the difference in tea consumption between the two racial groups is much smaller than the difference in coffee consumption. Overall, it seems that both coffee and tea are popular beverages among government workers, regardless of their race.

6. Are individuals who are married more likely to have a higher education rank than those who are unmarried, and does this differ by race?

A picture containing text, screenshot, font

Description automatically generated

From the plot, it is difficult to draw any conclusions about the relationship between marital status and education rank, as all four data points are exactly the same. All four data points have a race value of either Caucasian or non-Caucasian, a marital status value of either married or unmarried, and an education rank value of 9.

Therefore, we can conclude that there is no observed difference in education rank based on marital status within each racial group in the data being presented.

7. Average salary across different job types (office, manual, sales, service, tech)

A picture containing text, screenshot, colorfulness, line

Description automatically generated

From above plot, we can see that the average salary varies slightly across different job types. Among the job types listed, the highest average salary is for manual jobs at 45118.59, followed closely by sales jobs at 45296.86 and office jobs at 44820.61. Service jobs have a slightly lower average salary at 43531.55.

The average salary for tech jobs is much lower than the other job types listed, at 24007.83. This could indicate that the tech jobs in this dataset are lower-skilled or entry-level positions with lower salaries, or that the dataset only includes a small number of tech jobs.

8. How does the great\_customer\_class vary across different categories such as race and sex?

A picture containing text, screenshot, plot, software

Description automatically generated

From the plot, we can see that the great\_customer\_class appears to vary across different categories of sex and race. Among non-Caucasian customers, the sum of the great\_customer\_class is 24 for females (sex\_Male=0) and 16 for males (sex\_Male=1). This suggests that female non-Caucasian customers may be more likely to have a higher great\_customer\_class than male non-Caucasian customers. However, it's important to note that the difference in great\_customer\_class between the two groups is relatively small.

Among Caucasian customers, the sum of the great\_customer\_class is much higher overall, with a sum of 930 for males (sex\_Male=1) and 138 for females (sex\_Male=0). This suggests that Caucasian male customers may be more likely to have a higher great\_customer\_class than female Caucasian customers.

9. Distribution of office\_jobs, manual\_jobs, sales\_jobs, service\_jobs, and Tech\_jobs across different categories such as race and sex?

A picture containing text, screenshot, diagram, line

Description automatically generated

A picture containing text, screenshot, diagram, line

Description automatically generated

A picture containing text, screenshot, diagram, line

Description automatically generated

A picture containing text, screenshot, rectangle, diagram

Description automatically generated

A picture containing text, screenshot, diagram, line

Description automatically generated

From the above plot, we can see the distribution of office\_jobs, manual\_jobs, sales\_jobs, service\_jobs, and Tech\_jobs across different categories such as race and sex.

For office\_jobs, we can see that the sum of office\_jobs is higher for males (sex\_Male=1) than females (sex\_Male=0) for both non-Caucasian and Caucasian groups, with a particularly large difference for the non-Caucasian group.

For manual\_jobs, we can see that the sum of manual\_jobs is also higher for males (sex\_Male=1) than females (sex\_Male=0) for both non-Caucasian and Caucasian groups, although the difference is not as large as for office\_jobs.

For sales\_jobs, we can see that the sum of sales\_jobs is higher for males (sex\_Male=1) than females (sex\_Male=0) for both non-Caucasian and Caucasian groups, although again the difference is not as large as for office\_jobs.

For service\_jobs, we do not have any information in the provided plot.

For Tech\_jobs, we can see that the sum of Tech\_jobs is higher for males (sex\_Male=1) than females (sex\_Male=0) for both non-Caucasian and Caucasian groups, although again the difference is not as large as for office\_jobs.

10. The proportion of great customers in each marital status category

A picture containing text, screenshot, plot, number

Description automatically generated

The plot shows the proportion of great customers in each marital status category. Among the four categories, married customers have the highest proportion of great customers with 86.13%, followed by divorced customers with 7.19%. Unmarried customers have a proportion of 5.82%, while windowed customers have the lowest proportion with 0.86%. This information could be useful for businesses that target customers based on their marital status to adjust their marketing strategies to attract more customers who are more likely to be great customers. However, it's important to keep in mind that this data is based on a specific sample and may not be representative of the population as a whole.

11. The proportion of not great customers in each marital status category

A picture containing text, screenshot, diagram, rectangle

Description automatically generated

The plot shows the proportion of not great customers in each marital status category. Among the four categories, divorced customers have the highest proportion of not great customers with 52.62%, followed by unmarried customers with 24.72%. Married customers have a proportion of 20.54% of not great customers, while widowed customers have the lowest proportion with 2.12%. This information could be useful for businesses that target customers based on their marital status to adjust their marketing strategies to avoid attracting customers who are less likely to be great customers. However, it's important to keep in mind that this data is based on a specific sample and may not be representative of the population as a whole.

**Heart Disease**

1. What is the relationship between total cholesterol levels and systolic blood pressure, and how does this impact the risk of developing heart disease?

A picture containing text, screenshot, colorfulness, majorelle blue

Description automatically generated

The correlation value of 0.2132 indicates a positive correlation between total cholesterol levels and systolic blood pressure. This means that as the levels of total cholesterol increase, the systolic blood pressure tends to increase as well. However, the strength of this correlation is only moderate, suggesting that other factors may also contribute to variations in blood pressure and cholesterol levels. Furthermore, it is important to note that correlation does not imply causation, and further investigation would be needed to establish a causal relationship between these variables and the risk of developing heart disease.

2. Is there a correlation between BMI and heart rate, and does this vary between smokers and non-smokers?

A picture containing text, screenshot, colorfulness, line

Description automatically generated

The correlation coefficient of 0.06 suggests a weak positive correlation between BMI and heart rate. However, this correlation is not statistically significant or practically meaningful.

3. What is the distribution of glucose levels among patients with and without diabetes, and does this have any impact on the risk of developing heart disease?

Chart, histogram

Description automatically generated

The distribution of glucose levels among patients with and without diabetes reveals distinct patterns. Patients with diabetes exhibit higher glucose levels, with the majority exceeding 100. In contrast, patients without diabetes show a relatively normal distribution of glucose levels, centered around 80 or above. This shows that those who have diabetes are more likely to have increased glucose levels, which may also be a sign of the disease. Additionally, the finding that people without diabetes also have a considerable percentage of blood sugar levels that are above 80 suggests that there may be a link between blood sugar levels and an increased risk of developing heart disease.

4. How does the average number of cigarettes smoked per day vary across different age groups in the dataset?

A picture containing text, diagram, screenshot, plot

Description automatically generated

The distribution of the number of cigarettes smoked per day among different age groups reveals interesting patterns. Among individuals aged up to 50 years, there is a notable presence of heavy smokers, with some individuals reporting up to 50 cigarettes per day. However, as the age increases to 60 years and above, the number of heavy smokers decreases significantly. This suggests that older individuals tend to smoke fewer cigarettes per day on average. Furthermore, the distribution of cigarette consumption among individuals below 40 years appears to follow a relatively normal pattern.

5. Is there a significant difference in the average BMI between patients who do and do not take blood pressure medication?

A screenshot of a graph

Description automatically generated with low confidence

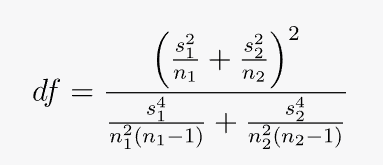
The t-statistic is 5.59, which indicates that there is a significant difference in the mean BMI between patients who take blood pressure medication and those who do not. The p-value is very small (1.29e-07), which indicates strong evidence against the null hypothesis that there is no difference in mean BMI between the two groups. Therefore, we can reject the null hypothesis and conclude that there is a significant difference in the average BMI between patients who take blood pressure medication and those who do not. Specifically, it appears that patients who take blood pressure medication have a higher mean BMI compared to those who do not.

Build-in function output: (5.602745021573313, 1.2157005860832093e-07)

Our function output: (5.624928797454224, 1.975463281844725e-08)

In build in function, they used different formula to calculate degree of freedom as a result the p value is different from our function.

**Build-in Formula:**



**Our Formula:** (n1 + n2) - 2

6. What is the relationship between age and the total cholesterol levels of patients?

A picture containing text, screenshot, line, diagram

Description automatically generated

Since the Correlation coefficient between age and total cholesterol levels is 0.27 so we can say that there is no relation between age and total cholesterol levels.

7. Is there a significant difference in systolic blood pressure between male and female patients?

After using student t-test we found the following answer,

t-statistic: -1.81, p-value: 0.07089

The t-test was conducted to determine if there is a significant difference in systolic blood pressure between male and female patients. The negative t-statistic value (-1.81) suggests that the mean systolic blood pressure is lower in female patients than in male patients. The p-value of 0.07089 is greater then the significance level of 0.05, indicating no statistically significant difference between male and female patients.

8. What is the average heart rate of patients with diabetes compared to those without diabetes? Is there a significant difference between them?

Average Heart rate patients having diabetes: 79.28

Average Heart rate patients not having diabetes: 75.66

After using student t-test we found the following answer,

t-statistic: 3.21, p-value: 0.00131

The p-value of 0.00131 is less than the common threshold for statistical significance of 0.05, indicating that we can reject the null hypothesis that there is no difference in the average heart rate between the two groups. Therefore, we can conclude that patients with diabetes have a significantly different average heart rate compared to those without diabetes.

9. Is there a relationship between the number of cigarettes smoked per day and systolic blood pressure?

A picture containing text, screenshot, line, diagram

Description automatically generated

The correlation coefficient between the number of cigarettes smoked per day and systolic blood pressure was found to be -0.09. This suggests a weak negative correlation between the two variables, indicating that as the number of cigarettes smoked per day increases, systolic blood pressure tends to slightly decrease. However, the correlation coefficient value is relatively small, indicating that the relationship between the two variables is not very strong. Therefore, while there is some association between cigarette smoking and systolic blood pressure, the effect size is weak.

10.Destribution of age in the dataset in terms of Gender

A picture containing screenshot, text, diagram, plot

Description automatically generated

From the above plot we can see that the destribution is rightly skewed and the majority of the people are between the age 38 - 43. Also based on the group we can analyze that majority number of people are female.

11. Distribution of education levels Inerms of Gender

A picture containing text, screenshot, rectangle, diagram

Description automatically generated

The distribution of education levels indicates that the majority of individuals in the dataset have an education rank of 1. Additionally, among all the education categories, category 4 stands out as having a higher proportion of males compared to females.

This observation suggests that there is an imbalance in the distribution of education levels, with a significant number of individuals falling into the education rank 1 category.

12. Distribution of smokers interms of diabetes

A picture containing text, screenshot, rectangle, diagram

Description automatically generated

This count distribution indicates a gender disparity in smoking behavior within the dataset, with a higher proportion of females abstaining from smoking compared to males. However, among smokers, the difference in the count between males and females is less pronounced, suggesting a relatively comparable proportion of male and female smokers.

13. Is there a significant difference in the proportion of smokers between males and females?

Built in : 165.47002460557053 7.221626137546635e-38

Ours : 165.47002460557053 0.0

The chi-square test for association yielded a test statistic of 165.47 and an extremely low p-value of 7.22e-38. Based on these results, we can conclude that there is a significant association between gender and smoking status in the dataset.

The low p-value indicates strong evidence against the null hypothesis, suggesting that the proportion of smokers differs significantly between males and females. Therefore, we reject the null hypothesis and accept the alternative hypothesis, which suggests a notable difference in smoking prevalence based on gender.

14. distribution of the target variable (TenYearCHD)

A picture containing diagram, circle, screenshot, font

Description automatically generated

The distribution of gender among individuals with TenYearCHD (Ten-Year Risk of Coronary Heart Disease) shows that approximately 46.7% are female and 53.3% are male. On the other hand, the distribution of gender among individuals without TenYearCHD indicates that approximately 58.9% are female and 41.1% are male.

This suggests that there may be a difference in the gender distribution between those with and without TenYearCHD. Females seem to be more prevalent in both groups, but the difference is more pronounced among individuals without TenYearCHD.

15. Relationship between age and cigsperday.

A picture containing text, screenshot, line, font

Description automatically generated

From the above distribution we can conclude that as the age increases, there tends to be a slight decrease in the number of cigarettes smoked per day. However, the correlation is weak, suggesting that the relationship between age and cigarette consumption is not strongly linear.

16. Variation Heart Rate across AGE groups

A picture containing text, screenshot, line, colorfulness

Description automatically generated

The correlation coefficient between age and heart rate in the dataset is -0.01. This indicates a very weak or negligible linear relationship between these variables. The correlation coefficient being close to zero suggests that there is no substantial linear association or dependence between age and heart rate.

**Mobile Price Range**

1. How does the battery power vary across different price ranges?

A picture containing text, screenshot, diagram, line

Description automatically generated

2. What is the relationship between pixel height (px\_height) and pixel width (px\_width)

A picture containing text, screenshot, electric blue, design

Description automatically generated

From the above distribution we can say that there is a positive correlation between pixel height and pixel width. Though the relation is moderately strong. It's not highly linear.

3.Distribution of price ranges across different categorical features

A picture containing text, screenshot, colorfulness, parallel

Description automatically generated

From the above distribution we can conclude that the majority of the phones in price range 3 have dual sim installed in them also there is not that much difference between the count of touch screen phone and button phone.

One thing also noticeable is that there is more touch screen phones in price range 0 and 1 compare to price range 2 and 3. Also the majority of phones without touch screen are in price range 2.

Overall, the numbers are almost similar in each category.

4. What is the frequency of 4G connectivity for different price range categories

A picture containing text, screenshot, rectangle, design

Description automatically generated

From the above destribution we can conclude that higher-priced mobile phones (Category 3) tend to have a higher frequency of 4G connectivity compared to other price range categories. On the other hand, lower-priced mobile phones (Category 2) have a higher frequency of 3G connectivity.

This information can be useful for understanding the relationship between price range and network connectivity options in the dataset. It suggests that as the price range increases, there is a higher likelihood of mobile phones having 4G connectivity

The highest frequency count of 274 in Price Range Category 3 suggests that mobile phones in this higher-priced category are more likely to offer 4G connectivity.

5. How does the weight of the mobile device ('mobile\_wt') differ across price ranges?

A picture containing text, screenshot, colorfulness, diagram

Description automatically generated

The distribution of phone weights across different price range categories suggests a potential negative correlation between price range and weight. The majority of mobile phones in the higher price range (Category 3) have relatively lower weights, typically falling within the 80-90 range. Conversely, the majority of mobile phones with weights ranging from 140 to 170 are found in the mid-price range (Category 2).

This observation indicates that as the price range increases, there is a tendency for mobile phones to have lighter weights. Similarly, within the mid-price range, there is a higher concentration of mobile phones with heavier weights.

6. Distribution of the front camera megapixels ('fc') for different price range categories

A picture containing text, screenshot, diagram, plot

Description automatically generated

The analysis indicates the following observations regarding front camera specifications and their relationship with price range categories: Majority of phones lack a front camera. Higher front camera pixel count is associated with a decrease in phone count. The highest front camera resolution (16 megapixels) is mostly found in price range 1. These findings highlight the limited prevalence of front cameras in the dataset, the inverse relationship between front camera pixel count and phone count, and the concentration of high-resolution front cameras in price range 1.

7. Visualizing the relationship between the internal memory (int\_memory) and the price range using a violin plot

A picture containing text, screenshot, plot, diagram

Description automatically generated

The box plot analysis reveals that the distribution of internal memory is similar across different price range categories. There are no outliers, indicating a consistent pattern. The internal memory ranges from 2 to 64 megapixels in each category, providing a variety of options for consumers.

8. Variation of clock speed across different price ranges

A picture containing text, diagram, screenshot, rectangle

Description automatically generated

The analysis of clock speed distribution across different price range categories indicates a consistent pattern. The distribution of clock speed appears to be similar across all price range categories, suggesting that there is no significant variation in clock speed based on the price range.

9. Distribution of talk time for mobile devices

A picture containing text, screenshot, diagram, colorfulness

Description automatically generated

Based on the distribution analysis, a notable observation is that the talk time for mobile phones in price range 0 is consistently greater compared to other price range categories, irrespective of the duration of talk time. This conclusion suggests that, within the given dataset, mobile phones falling within price range 0 tend to offer longer talk time capabilities compared to devices in other price range categories.

10. Distribution of mobile device weights across different touch screen types(0, 1)

A picture containing text, diagram, screenshot, rectangle

Description automatically generated

The analysis of Mobile Device Weights in relation to Touch Screen Types reveals that there is minimal difference observed between touch screen and non-touch screen devices. The distribution of mobile weights for both touch screen and non-touch screen devices exhibits similarities. Specifically, both touch screen and non-touch screen devices have a maximum weight of 200 and a minimum weight of 80. Additionally, the median weight for both types of devices is found to be the same.

11. Correlation between battery power and clock speed using scatter plot

A picture containing text, screenshot, font, number

Description automatically generated

The scatter plot analysis reveals a correlation coefficient of 0.012 between battery power and clock speed. This correlation coefficient indicates a very weak and nearly negligible positive correlation between the two variables.

The value of 0.012 suggests that there is almost no linear relationship between battery power and clock speed within the given dataset. It implies that changes in battery power do not consistently correspond to changes in clock speed, and vice versa. Therefore, based on this analysis, it can be concluded that battery power and clock speed have a very weak or insignificant linear relationship.

12. Correlation between the amount of internal memory and the screen size (sc\_h \* sc\_w) using scatter plot.

A picture containing text, screenshot, majorelle blue, electric blue

Description automatically generated

The analysis of the relationship between the internal memory and screen size (sc\_h \* sc\_w) of mobile devices showed a very weak correlation (r = 0.021). This means that there is almost no connection between the size of the screen and the amount of internal memory in mobile devices.

**Model:**

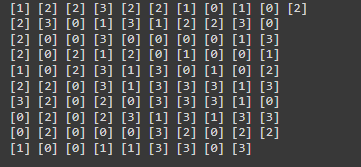
We created a model using the decision tree algorithm to predict the price range of mobile phones using relevant independent features.

**Here is the accuracy of our model:**

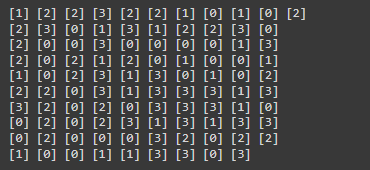
**Text

Description automatically generated**

**Sampel of Y\_Test:**

****

**Sample of Y Predict:**

****

**Discussion:**

**Great Customer:**

Based on our analysis using a significance level of 0.5, we have identified several highly significant predictors of the 'great\_customer\_class' variable. These predictors include 'office\_jobs', 'Tech\_jobs', 'workclass\_private', 'workclass\_government', 'education\_rank', 'marital-status\_Divorced', and 'marital-status\_Married'.

Furthermore, we found that certain features, such as 'salary', 'tea\_per\_year', 'service\_jobs', 'sex\_Male', 'sales\_jobs', and 'age', exhibited moderate significance in predicting the 'great\_customer\_class' variable.

These results suggest that the identified features play a crucial role in determining the classification of a customer as a "great\_customer\_class."

**Mobile Price Range:**

Following a comprehensive analysis of the dataset and the determination of p-values, we have reached the following conclusions. The features "Ram" and "battery\_power" demonstrate the smallest p-values, indicating a robust relationship with the target variable. Similarly, "Px\_width" and "px\_height" exhibit statistically significant p-values, albeit to a slightly lesser extent. Conversely, the remaining features have larger p-values, suggesting a weaker association with the target variable.

Consequently, we have selected "ram," "battery\_power," "px\_width," "px\_height," "mobile\_wt," "int\_memory," "n\_cores," and "sc\_h" as the top features for training our model. These features are deemed to hold the most predictive power in relation to the target variable.

**Heart Disease:**

After conducting an analysis on this dataset, we have concluded that, with the exception of 'heartRate' and 'currentSmoker', all the features have p-values less than 0.05. This indicates that these features are statistically significant and possess a substantial relationship with the target variable, TenYearCHD.

Consequently, the features identified as statistically significant, based on the provided p-values, are as follows: prevalentHyp, age, sysBP, diaBP, diabetes, male, BPMeds, totChol, education, BMI, prevalentStroke, and cigsPerDay.

**Conclusion:**

This project has been a valuable learning experience for us in the field of data analysis. We have gained knowledge and skills in handling missing values, outliers, encoding techniques, exploratory data analysis, hypothesis testing, and model creation.

Throughout the project, we encountered challenges and obstacles, but with the help of online documentation and resources like Stack Overflow, we were able to overcome them and find solutions.

Overall, this project has increased our familiarity and comfort with various data analysis techniques. We have gained practical experience and a better understanding of how to apply these techniques effectively. We feel more confident in our abilities and are excited to apply what we have learned in future projects.

**Appendix:**

**Comparison of Mean, Median & Mode Functions:**

Our Mean: 1238.4354354354355

Built in Mean: 1238.4354354354355

Our Median: 1225

Built in Median: 1225.0

Our Std: 439.51077382231654

Built in Std: 439.5107738223165

Our Variance: 193169.72030589147

Built in Variane: 193169.72030589145

**Module We Used:**

import pandas as pd

import numpy as np

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

from sklearn.metrics import accuracy\_score

import matplotlib.pyplot as plt

import seaborn as sns

import plotly.graph\_objs as go

import plotly.express as px

import plotly.subplots as sp

from plotly.subplots import make\_subplots

import plotly.figure\_factory as ff

from scipy.stats import f\_oneway

from scipy.stats import f

from scipy.stats import chi2

from scipy.stats import chisquare

from scipy.stats import chi2\_contingency

from scipy.stats import norm

from scipy.stats import ttest\_ind

from typing import List

**Hard Code:**

def mean(col):

col = [v for v in col if v is not None]

if not col:

return None

return np.sum(col) / len(col)

def median(col):

col = [v for v in col if v is not None]

if not col:

return None

sorted\_values = sorted(col)

n = len(sorted\_values)

middle = n // 2

if n % 2 == 0:

return (sorted\_values[middle - 1] + sorted\_values[middle]) // 2

else:

return sorted\_values[middle]

def std\_var(col, type = "std"):

col = [v for v in col if v is not None]

if not col:

return None

mean = np.sum(col) / len(col)

squared\_diffs = [(v - mean)\*\*2 for v in col]

variance = sum(squared\_diffs) / len(squared\_diffs)

if type == "var":

return variance

return np.sqrt(variance)

#### \*\*Creating my own Anova Function\*\*

def anova\_one(\*categories) -> List[float]:

  # Flatten all the categories into one 1D array

  # grand = np.array(categories)

  # grand = grand.flatten()

  grand = []

  for outer in categories:

    for elm in outer:

      grand.append(elm)

  num\_of\_condition = len(categories)

  total\_num\_of\_data = len(grand)

  # for catg in categories:

  #   total\_num\_of\_data += len(catg)

  df\_bw = num\_of\_condition - 1

  df\_wtn = total\_num\_of\_data - num\_of\_condition

  # Critical Value

  f\_crit = f.ppf(q=1-.05, dfn=df\_bw, dfd=df\_wtn)

  grand\_mean = np.mean(grand)

  mean\_arr = [np.mean(x) for x in categories]

  SST\_numerator = grand

  # Sum of Squares Total Σ(x - x̄)^2

  SST = np.sum(np.apply\_along\_axis(lambda x: np.power(x - grand\_mean, 2), 0, SST\_numerator))

  # Sum of Squares Withing Σ(x1 - x̄1)^2 + Σ(x2 - x̄2)^2 + ... ... ... + Σ(xn - x̄n)^2

  SSW = 0

  temp = categories

  for catg in temp:

    catg\_mean = np.mean(catg)

    SSW += np.sum(np.apply\_along\_axis(lambda x: np.power(x - catg\_mean, 2), 0, catg))

  SSW = SSW

# Sum of Squares Between

  SSBW = SST - SSW

  # Mean Squares Between

  MSBW =SSBW / df\_bw

  # Mean Squares Within

  MSW = SSW / df\_wtn

  f\_stat = MSBW / MSW

  p\_value = f.sf(f\_stat, df\_bw, df\_wtn)

  return f\_crit, f\_stat, p\_value

### \*\*Doing Chi Square Test to categorical columns to determine the significance of predicting price\_range\*\*

#### \*\*Creating our Own Chi Square Function\*\*

def cross\_tab(ind, dep):

  # Fetching the rows and columns for the table

  new\_df = pd.DataFrame({

      'ind' : ind,

      'dep' : dep

  })

  rows = ind.unique()

  cols = dep.unique()

  cols.sort()

  cols

  table = []

  for row in rows:

    c = []

    for col in cols:

      slices = len(new\_df[(new\_df['ind'] == row) & (new\_df['dep'] == col)])

      c.append(slices)

    table.append(c.copy())

    c = c.clear()

  table = pd.DataFrame(table, columns = cols, index = rows)

  return table

def to\_expected(observe):

  cols = observe.columns

  rows = observe.index

  grand\_ttl = np.sum(np.sum(observe))

  expected = observe.copy()

  for row in rows:

    for col in cols:

      row\_ttl = np.sum(observe.loc[row])

      col\_ttl = np.sum(observe[col])

      expected.loc[row, col] = (row\_ttl \* col\_ttl) / grand\_ttl

  return expected

# print(to\_expected(observe))

def chi\_2\_t(observed, expected):

  chi\_square = np.sum(np.sum(np.power(observe - expected, 2) / expected))

  row\_count = len(observed.index) - 1

  col\_count = len(observed.columns) - 1

  df = row\_count \* col\_count

  p\_value = 1 - chi2.cdf(chi\_square, df)

  return chi\_square, p\_value

#student ttest

def student\_t\_test(sample1, sample2):

    n1 = len(sample1)

    n2 = len(sample2)

    x1 = np.mean(sample1)

    x2 = np.mean(sample2)

    s1 = np.std(sample1)

    s2 = np.std(sample2)

    t\_stat = abs(x1-x2)/np.sqrt((np.power(s1,2)/n1)+(np.power(s2,2)/n2))

    df = n1 + n2 - 2

    p\_value = 2 \* (1 - t.cdf(np.abs(t\_stat), df=df))

    return t\_stat, p\_value

**Decision Tree Model:**

class Node():

def \_\_init\_\_(self, f\_idx = None, threshold = None, left = None, right = None, IG = None, value = None,):

# Decision Node

self.f\_idx = f\_idx

self.threshold = threshold

self.IG = IG

self.left = left

self.right = right

# Leaf Node

self.value = value

class Decision\_tree():

def \_\_init\_\_(self, min\_sample\_split = 2, max\_depth = 2):

self.root = None

self.min\_sample\_split = min\_sample\_split

self.max\_depth = max\_depth

def make\_tree(self, df, cur\_depth = 0):

X, Y = df[:, : -1], df[:, -1]

n\_samp, n\_feature = np.shape(X)

if n\_samp >= self.min\_sample\_split and cur\_depth <= self.max\_depth:

best\_split = self.get\_require\_split(df, n\_samp, n\_feature)

if best\_split['IG'] > 0:

left = self.make\_tree(best\_split['left\_df'], cur\_depth + 1)

right = self.make\_tree(best\_split['right\_df'], cur\_depth + 1)

return Node(best\_split['f\_idx'], best\_split['threshold'], left, right, best\_split['IG'])

leaf\_node = self.get\_leaf\_val(Y)

return Node(value = leaf\_node)

def get\_require\_split(self, df, n\_samp, n\_feature):

best\_split = {}

max\_IG = -float("inf")

for f\_idx in range(n\_feature):

f\_values = df[:, f\_idx]

n\_threshholds = np.unique(f\_values)

for threshold in n\_threshholds:

left\_df = np.array([r for r in df if r[f\_idx] <= threshold])

right\_df = np.array([r for r in df if r[f\_idx] > threshold])

if len(left\_df) > 0 and len(right\_df) > 0:

parent\_y, l\_child\_y, r\_child\_y = df[:, -1], left\_df[:, -1], right\_df[:, -1]

#

cur\_IG = self.information\_gain(parent\_y, l\_child\_y, r\_child\_y)

if cur\_IG > max\_IG:

max\_IG = cur\_IG

best\_split['f\_idx'] = f\_idx

best\_split['threshold'] = threshold

best\_split['left\_df'] = left\_df

best\_split['right\_df'] = right\_df

best\_split['IG'] = cur\_IG

return best\_split

def information\_gain(self, parent, l\_child, r\_child):

l\_weight = len(l\_child) / len(parent)

r\_weight = len(r\_child) / len(parent)

G\_parent = self.gini\_index(parent)

G\_l\_child = self.gini\_index(l\_child)

G\_r\_child = self.gini\_index(r\_child)

IG = G\_parent - (l\_weight \* G\_l\_child + r\_weight \* G\_r\_child)

return IG

def gini\_index(self, target):

categories = np.unique(target)

gini\_index = 0

for catg in categories:

p = len(target[target == catg]) / len(target)

gini\_index += p\*\*2

return 1 - gini\_index

def get\_leaf\_val(self, Y):

Y = list(Y)

return max(Y, key = Y.count)

def fit(self, X, Y):

df = np.concatenate((X, Y), axis = 1)

self.root = self.make\_tree(df)

def predict(self, X):

pred = [self.get\_pred(x, self.root) for x in X]

return pred

def get\_pred(self, x, root):

if root.value != None:

return root.value

f\_val = x[root.f\_idx]

if f\_val <= root.threshold:

return self.get\_pred(x, root.left)

else:

return self.get\_pred(x, root.right)

**Great Customer**

# from google.colab import files

# uploaded = files.upload()

pd.set\_option('display.max\_columns', None)

## \*\*Reading Datasets\*\*

customer = pd.read\_csv('great\_customers.csv')

g\_c = customer.copy()

## \*\*Checking the Datasets\*\*

g\_c.shape

g\_c.info()

g\_c.head()

g\_c['great\_customer\_class'].unique()

g\_c.occupation.value\_counts().sort\_values(ascending = False)

## \*\*Null Value Percentage\*\*

g\_c\_null\_colms = g\_c.isnull().mean() \* 100

g\_c\_null\_colms

## \*\*Visualizing the Null Values\*\*

fig = px.imshow(g\_c.isnull(), color\_continuous\_scale='thermal')

fig.update\_layout(

    title = {

        'text' : "Destribution of Null Values",

        'x' : 0.5,

        'y' : 0.98

    },

    width = 900,

)

fig.show()

## \*\*Handling missing values of numerical columns\*\*

### \*\*Columns Having Nulls\*\*

num\_cols\_with\_nulls = []

for col in g\_c.columns:

  if g\_c\_null\_colms[col] > 0:

    if g\_c[col].dtype == 'int64' or g\_c[col].dtype == 'float64':

      num\_cols\_with\_nulls.append(col)

num\_cols\_with\_nulls

#### \*\*Plotting the Distribution\*\*

def num\_col\_dest(rows, cols, df, cols\_list, title, break\_point = -1, height = 600, width = 1200):

  fig = sp.make\_subplots(rows=rows, cols=cols)

  idx  = 0

  for r in range(1, rows + 1):

    for c in range(1, cols + 1):

      col = cols\_list[idx]

      idx += 1

      data = df[col]

      trace = go.Histogram(x=data, nbinsx=20)

      fig.add\_trace(trace, row = r, col = c)

      fig.update\_xaxes(title\_text=col, row=r, col=c)

      fig.update\_yaxes(title\_text="frequency", row=r, col=c)

      if idx == break\_point: break

  fig.update\_layout(

      title={

          'text': title,

          'y': 0.989,

          'x': 0.5,

          'xanchor': 'center',

          'yanchor': 'top',

          'font' : {

              'color' : '#393646',

              'family' : 'Bold',

              'size' : 26

          }

      },

      showlegend = False,

      height = height,

      width = width

  )

  fig.update\_xaxes(range = [0, 170000], row = 1, col = 2)

  fig.update\_xaxes(range = [-1200, 12000], row = 1, col = 3)

  fig.update\_xaxes(range = [-500, 2500], row = 2, col = 1)

  fig.show()

num\_col\_dest(2, 3, g\_c, num\_cols\_with\_nulls,  "Destribution of numericsl features having NA's")

#### \*\*Using Mean and Median to fill up NA's\*\*

# Numerical Columns Having Null values

# ['age', 'salary', 'mins\_beerdrinking\_year', 'mins\_exercising\_year', 'tea\_per\_year', 'coffee\_per\_year']

solve\_col = []

g\_c\_no\_null = g\_c.copy()

for col in num\_cols\_with\_nulls:

  mean = g\_c\_no\_null[col].mean()

  median = g\_c\_no\_null[col].median()

  # Create new columns with the original column name and "\_mean" or "\_median" suffix

  # The new columns contain the original values with missing values replaced by the mean or median

  g\_c\_no\_null[col + "\_mean"] = g\_c\_no\_null[col].fillna(mean)

  g\_c\_no\_null[col + "\_median"] = g\_c\_no\_null[col].fillna(median)

  solve\_col.append(col)

  solve\_col.append(col + "\_mean")

  solve\_col.append(col + "\_median")

#### \*\*Checking the Basic Statistical Difference\*\*

g\_c\_no\_null[solve\_col].describe()

#### \*\*Checking the Variance Difference\*\*

idx = 1

original = g\_c\_no\_null[solve\_col[0]].var()

for col in solve\_col:

  change = 0

  variance = g\_c\_no\_null[col].var()

  prct = 100 - (variance / original) \* 100

  print(f"{col} Variance: {variance:.2f} ({prct:.2f}% Difference)")

  if idx % 3 == 0:

    change = 1

    print(" ")

  idx += 1

  if change and idx < len(solve\_col):

    original = g\_c\_no\_null[solve\_col[idx - 1]].var()

#### \*\*Plotting the Distribution Difference\*\*

fig, axs = plt.subplots(2, 3, figsize=(16, 8))

idx = 0

for r in range(2):

  for c in range(3):

    col = num\_cols\_with\_nulls[idx]

    idx+=1

    sns.kdeplot(data=g\_c\_no\_null[col], ax=axs[r, c], label="Original")

    sns.kdeplot(data=g\_c\_no\_null[col + "\_mean"], ax=axs[r, c], label=col.title() + " Mean")

    sns.kdeplot(data=g\_c\_no\_null[col + "\_median"], ax=axs[r, c], label=col.title() + " Median")

    axs[r, c].set\_title(col.upper() + " Distribution")

    axs[r, c].set\_xlabel(col.upper())

    axs[r, c].legend()

axs[0, 1].set\_xlim(0, 150000)

plt.tight\_layout(pad = 3.0)

plt.show()

mean\_fill = ['mins\_beerdrinking\_year', 'mins\_exercising\_year', 'coffee\_per\_year']

median\_fill = ['age', 'salary', 'tea\_per\_year']

g\_c\_no\_null[mean\_fill] = g\_c\_no\_null[['mins\_beerdrinking\_year\_mean', 'mins\_exercising\_year\_mean', 'coffee\_per\_year\_mean']]

g\_c\_no\_null[median\_fill] = g\_c\_no\_null[['age\_median', 'salary\_median', 'tea\_per\_year\_median']]

cols = g\_c.columns

g\_c\_no\_null = g\_c\_no\_null[cols]

g\_c\_no\_null.isnull().mean() \* 100

## \*\*Handling the missing values of Categorical columns\*\*

#### \*\*Null Value Percentage\*\*

null\_catg\_cols = []

# Extracting the categorical cols having null values

for col in g\_c.columns:

  if g\_c\_null\_colms[col] > 0:

    if g\_c[col].dtype != 'int64' and g\_c[col].dtype != 'float64':

      print(f"{col} : {g\_c\_null\_colms[col]:.2f}%")

      null\_catg\_cols.append(col)

#### \*\*Plotting the Destribution\*\*

def catg\_col\_dest(rows, cols, catg\_cols, break\_point = -1, title = "Distribution of Categorical Features Having Null Values"):

  fig = sp.make\_subplots(rows=rows, cols=cols)

  idx  = 0

  for r in range(1, rows + 1):

    for c in range(1, cols + 1):

      col = catg\_cols[idx]

      idx += 1

      data = g\_c\_no\_null[col].value\_counts().sort\_values(ascending = False).reset\_index()

      data.rename(columns = {'index': col, col : 'frequency'}, inplace = True)

      trace = go.Bar(x = data[col], y = data['frequency'])

      fig.add\_trace(trace, row = r, col = c)

      fig.update\_xaxes(title\_text=col, row=r, col=c)

      fig.update\_yaxes(title\_text="frequency", row=r, col=c)

      if idx == break\_point: break

  fig.update\_layout(

      title={

          'text': title,

          'y': 0.989,

          'x': 0.5,

          'xanchor': 'center',

          'yanchor': 'top',

          'font' : {

              'color' : '#393646',

              'family' : 'Bold',

              'size' : 26

          }

      },

      showlegend = False,

      height = 600,

      width = 1200

  )

  fig.show()

catg\_col\_dest(1, 2, null\_catg\_cols)

### \*\*Using Mode to fill NA's\*\*

catg\_null\_cols = ['workclass', 'occupation']

for col in catg\_null\_cols:

  mode = g\_c\_no\_null[col].mode()[0]

  g\_c\_no\_null[col] = g\_c\_no\_null[col].fillna(mode)

catg\_col\_dest(1, 2, null\_catg\_cols, title = "Distribution of Categorical Features after Handling Null Values")

g\_c\_no\_null.isnull().mean() \* 100

## \*\*Using Encoding to convert the string of categorical features to numeric\*\*

cols\_to\_transforme = []

for col in g\_c.columns:

  if g\_c[col].dtype != 'int64' and g\_c[col].dtype != 'float64':

    cols\_to\_transforme.append(col)

cols\_to\_transforme

#### \*\*Number of unique category in each feature\*\*

for col in cols\_to\_transforme:

  print(col)

  print(g\_c\_no\_null[col].unique())

  print()

catg\_col\_dest(2, 3, cols\_to\_transforme, break\_point = 5, title = "Distribution of Categorical Features")

def one\_hot\_encode(df):

    feature\_len = len(df.columns)

    for feature in df.columns:

#### \*\*Performing One Hot Encoding Technique\*\*

g\_c.head(1)

cols\_to\_one\_hot\_encode = ['workclass', 'marital-status', 'race', 'sex']

g\_c\_one\_encode = g\_c\_no\_null[cols\_to\_one\_hot\_encode].copy()

g\_c\_one\_encode = one\_hot\_encode(g\_c\_one\_encode)

cols\_to\_add = ['user\_id', 'age', 'salary', 'education\_rank', 'occupation', 'mins\_beerdrinking\_year',

               'mins\_exercising\_year', 'works\_hours', 'tea\_per\_year', 'coffee\_per\_year', 'great\_customer\_class']

g\_c\_one\_encode = pd.concat([g\_c\_one\_encode, g\_c\_no\_null[cols\_to\_add]], axis = 1)

g\_c\_one\_encode

### \*\*Handling the occupation feature\*\*

g\_c\_one\_encode['occupation'].unique()

data = g\_c\_one\_encode['occupation'].value\_counts().reset\_index()

# plot the histogram using plotly.express

fig = px.bar(data, x='index', y='occupation', labels={'index':'Occupation', 'occupation':'Count'})

fig.update\_layout(

    title = {

        'text': "Occupation",

        'y': 0.989,

        'x': 0.5,

        'xanchor': 'center',

        'yanchor': 'top',

        'font' : {

            'color' : '#393646',

            'family' : 'Bold',

            'size' : 26

        },

    },

    height = 500,

    width  = 1200

)

fig.show()

job\_catgories = {

    'office\_jobs': ['clerical', 'professional', 'executive', 'lawenf', 'estate\_agent'],

    'manual\_jobs': ['farm', 'craft', 'factory', 'cleaner', 'soldier', 'trucker'],

    'sales\_jobs' : ['sales'],

    'service\_jobs' : ['service'],

    'Tech\_jobs' : ['tech']

}

# 'office\_jobs', 'manual\_jobs', 'sales\_jobs', 'service\_jobs', 'Tech\_jobs'

g\_c\_encode = g\_c\_one\_encode.copy()

for catg in job\_catgories:

  g\_c\_encode[catg] = g\_c\_encode['occupation'].isin(job\_catgories[catg]).astype('int32')

g\_c\_encode.sample(5)

re\_index\_cols = ['user\_id', 'age','office\_jobs', 'manual\_jobs', 'sales\_jobs', 'service\_jobs', 'Tech\_jobs',

                 'workclass\_private', 'workclass\_government', 'salary', 'education\_rank', 'marital-status\_Divorced',

                 'marital-status\_Married', 'marital-status\_Widowed', 'race\_caucasian', 'sex\_Male',

                 'mins\_beerdrinking\_year', 'mins\_exercising\_year', 'works\_hours', 'tea\_per\_year', 'coffee\_per\_year',

                 'great\_customer\_class']

g\_c\_encode = g\_c\_encode.reindex(re\_index\_cols, axis = 1)

g\_c\_encode.head()

g\_c\_encode.shape

g\_c\_encode.columns

g\_c\_encode.head()

## \*\*Handling Outliers\*\*

### \*\*Plotting the destribution to identify any outliers\*\*

num\_colms = ['age', 'salary', 'mins\_beerdrinking\_year', 'mins\_exercising\_year', 'works\_hours', 'tea\_per\_year', 'coffee\_per\_year']

# , vertical\_spacing = 0.14, horizontal\_spacing = 0.05

fig = sp.make\_subplots(rows=2, cols=4)

idx = 0

for r in range(1, 3):

  for c in range(1, 5):

    col = num\_colms[idx]

    idx+=1

    trace = go.Box(x=g\_c\_no\_null[col], orientation='h', boxpoints='suspectedoutliers')

    fig.add\_trace(trace, row=r, col=c)

    fig.update\_yaxes(showticklabels=False)

    fig.update\_xaxes(title\_text=col, row=r, col=c)

    if idx == 7: break

fig.update\_layout(

    title={

        'text': 'Destribution of Numeric Columns',

        'y': 0.989,

        'x': 0.5,

        'xanchor': 'center',

        'yanchor': 'top',

        'font' : {

            'color' : '#393646',

            'family' : 'Bold',

            'size' : 26

        }

    },

    showlegend = False,

    height = 800,

    width = 1200

)

fig.update\_xaxes(range=[0, 170000], row=1, col=2)

fig.update\_xaxes(range=[0, 100], row=2, col=2)

fig.show()

### \*\*Summery Statistics of outliers in each columns\*\*

def get\_limit(col):

  perct\_25 = col.quantile(0.25)

  perct\_75 = col.quantile(0.75)

  IQR = perct\_75 - perct\_25

  upper\_limit = perct\_75 + (1.5 \* IQR)

  lower\_limit = perct\_25 - (1.5 \* IQR)

  return upper\_limit, lower\_limit

cols\_wth\_outl = []

for col in num\_colms:

  data = g\_c\_no\_null[col]

  upper\_bound, lower\_bound = get\_limit(data)

  outliers = data[(data < lower\_bound) | (data > upper\_bound)]

  outlier\_count = len(outliers)

  total\_count = len(data)

  outlier\_percent = outlier\_count/total\_count \* 100

  print(f"{col} Percentage : {outlier\_percent : .2f}%")

  if outlier\_percent != 0:

    cols\_wth\_outl.append(col)

    mean\_outliers = np.mean(outliers)

    median\_outliers = np.median(outliers)

    std\_outliers = np.std(outliers)

    print(f"{col} Mean       : {mean\_outliers : .2f}")

    print(f"{col} Median     : {median\_outliers : .2f}")

    print(f"{col} Std        : {std\_outliers : .2f}")

  print()

#### \*\*Columns Having Outliers\*\*

cols\_wth\_outl

g\_c\_no\_null[cols\_wth\_outl].describe()

#### \*\*Destribution of outlier columns:\*\*

def outlier\_dist\_plt(df, cols\_wth\_outl, title, rows = 2, cols = 3, break\_point = -1):

  fig = sp.make\_subplots(rows=rows, cols=cols)

  idx = 0

  for r in range(1, 3):

    for c in range(1, 4):

      col = cols\_wth\_outl[idx]

      idx += 1

      trace = go.Box(x = df[col], orientation='h', boxpoints='suspectedoutliers', boxmean=True)

      fig.add\_trace(trace, row = r, col = c)

      fig.update\_xaxes(title = col, row = r, col = c)

      if idx == break\_point: break

  # fig.update\_xaxes(range = [0, 100000], row = 1, col = 1)

  # fig.update\_xaxes(range = [0, 1000], row = 1, col = 3)

  # fig.update\_xaxes(range = [0, 500], row = 2, col = 1)

  fig.update\_yaxes(showticklabels=False)

  # fig.update\_yaxes(range = [0, 100], row = 2, col = 3)

  fig.update\_layout(

      title={

          'text': title,

          'y': 0.989,

          'x': 0.5,

          'xanchor': 'center',

          'yanchor': 'top',

          'font' : {

              'color' : '#393646',

              'family' : 'Bold',

              'size' : 26

          }

      },

      showlegend = False,

      height = 800,

      width = 1200

  )

  fig.show()

outlier\_dist\_plt(g\_c\_no\_null, cols\_wth\_outl, 'Destributoin of numeric columns Having Outliers', break\_point = 5)

num\_col\_dest(3, 2, g\_c\_no\_null, cols\_wth\_outl, "Destribution of Numerical Columns Before handling Outliers", break\_point = 5, height = 800,width= 1200)

#### \*\*Using Capping Method to handle outliers\*\*

def get\_limit(col):

  perct\_25 = col.quantile(0.25)

  perct\_75 = col.quantile(0.75)

  IQR = perct\_75 - perct\_25

  upper\_limit = perct\_75 + (1.5 \* IQR)

  lower\_limit = perct\_25 - (1.5 \* IQR)

  return upper\_limit, lower\_limit

g\_c\_no\_outl = g\_c\_encode.copy()

# Iterate through columns with outliers

for col in cols\_wth\_outl:

  upper\_limit, lower\_limit = get\_limit(g\_c\_no\_outl[col])

  # Replace values outside of the upper and lower limits with the respective limit

  g\_c\_no\_outl[col] = np.where(

      g\_c\_no\_outl[col] > upper\_limit,

      upper\_limit,

      np.where(

          g\_c\_no\_outl[col] < lower\_limit,

          lower\_limit,

          g\_c\_no\_outl[col]

      )

  )

g\_c\_no\_null[num\_colms].describe()

g\_c\_no\_outl[num\_colms].describe()

#### \*\*Plotting the destribution after handling outliers\*\*

outlier\_dist\_plt(g\_c\_no\_outl, cols\_wth\_outl, 'Box Plot Destributoin of numeric columns After Handling Outliers', break\_point = 5)

num\_col\_dest(3, 2, g\_c\_no\_outl, cols\_wth\_outl, "Destribution of Numerical Columns After handling Outliers", break\_point = 5, height = 800,width= 1200)

### \*\*Standardizing the features with extreme change after handling the outliers\*\*

cols\_to\_stdz = ['mins\_beerdrinking\_year', 'mins\_exercising\_year', 'works\_hours']

g\_c\_stdz\_oult = g\_c\_no\_outl.copy()

for col in cols\_to\_stdz:

  g\_c\_stdz\_oult[col] = (g\_c\_stdz\_oult[col] - g\_c\_stdz\_oult[col].mean()) / g\_c\_stdz\_oult[col].std()

g\_c\_no\_outl[cols\_to\_stdz].describe()

g\_c\_stdz\_oult[cols\_to\_stdz].describe()

### \*\*Variance of these features Before Standardized\*\*

for col in cols\_to\_stdz:

  var = g\_c\_no\_outl[col].var()

  print(f"Variance of {col} : {var}")

\*\*Top 10 Values counts of unique numerical value of these features before handling outliers\*\*

for col in cols\_to\_stdz:

  print(g\_c\_no\_null[col].value\_counts().iloc[:10])

  print()

### \*\*Final DataFrame After Preprocessing\*\*

feature\_to\_select = ['user\_id', 'age', 'office\_jobs', 'manual\_jobs', 'sales\_jobs',

                     'service\_jobs', 'Tech\_jobs', 'workclass\_private',

                     'workclass\_government', 'salary', 'education\_rank',

                     'marital-status\_Divorced', 'marital-status\_Married',

                     'marital-status\_Widowed', 'race\_caucasian', 'sex\_Male',

                     'tea\_per\_year', 'coffee\_per\_year', 'great\_customer\_class']

g\_c\_pre\_process = g\_c\_no\_outl[feature\_to\_select].copy()

num\_colms = ['age', 'salary', 'tea\_per\_year', 'coffee\_per\_year']

catg\_colms = ['office\_jobs', 'manual\_jobs', 'sales\_jobs', 'service\_jobs', 'Tech\_jobs', 'workclass\_private',

              'workclass\_government', 'education\_rank', 'marital-status\_Divorced', 'marital-status\_Married', 'marital-status\_Widowed',

              'race\_caucasian', 'sex\_Male', 'great\_customer\_class']

g\_c\_pre\_process.shape

g\_c\_pre\_process.head()

## \*\*Destribution of numerical features with respect to great\_customer\_class\*\*

def box\_plot(df, relevent\_colms, rows = 2, cols = 2) -> None:

  fig = sp.make\_subplots(rows=rows, cols=cols, vertical\_spacing = 0.17, horizontal\_spacing = 0.1)

  idx = 0

  for r in range(1, 3):

    for c in range(1, 3):

      col = relevent\_colms[idx]

      idx+=1

      trace1 = go.Box(x=df[df.great\_customer\_class == 0][col], name='No', orientation='h', boxpoints='suspectedoutliers')

      trace2 = go.Box(x=df[df.great\_customer\_class == 1][col], name='Yes', orientation='h', boxpoints='suspectedoutliers')

      fig.add\_trace(trace1, row=r, col=c)

      fig.add\_trace(trace2, row=r, col=c)

      fig.update\_yaxes(title\_text='Great Customer Class', row=r, col=c)

      fig.update\_xaxes(title\_text=col, row=r, col=c)

  fig.update\_xaxes(range = [0, 500], row=2, col=1)

  fig.update\_layout(

      title={

          'text': 'Relationship between Numerical Features and Great Customer Class',

          'y': 0.99,

          'x': 0.5,

          'xanchor': 'center',

          'yanchor': 'top',

          'font' : {

              'color' : '#393646',

              'family' : 'Bold',

              'size' : 26

          }

      },

      showlegend = False,

      height = 800,

      width = 1200

  )

  fig.show()

box\_plot(g\_c\_pre\_process, num\_colms)

### \*\*Handling the outliers of the numerical features with respect to great\_customer\_class\*\*

categories = [0, 1]

for col in num\_colms:

    for cat in categories:

        subset = g\_c\_pre\_process[g\_c\_pre\_process['great\_customer\_class'] == cat][col]

        q1 = subset.quantile(0.25)

        q3 = subset.quantile(0.75)

        iqr = q3 - q1

        lower\_bound = q1 - (1.5 \* iqr)

        upper\_bound = q3 + (1.5 \* iqr)

        subset[subset < lower\_bound] = lower\_bound

        subset[subset > upper\_bound] = upper\_bound

        g\_c\_pre\_process.loc[g\_c\_pre\_process['great\_customer\_class'] == cat, col] = subset

catg\_colms

g\_c\_pre\_process[catg\_colms].head()

### \*\*Destribution of these features after handling the outliers\*\*

box\_plot(g\_c\_pre\_process, num\_colms)

def target\_category\_hist(df, title = "Relationship between Categorical Features and Great Customer Class", break\_point = 13, rows = 3, cols = 5,  ):

  fig = make\_subplots(rows=rows, cols=5, subplot\_titles=catg\_colms[:-1], horizontal\_spacing = 0.1)

  idx = 0

  great\_customer = df[df['great\_customer\_class'] == 1]

  not\_great\_customer = df[df['great\_customer\_class'] == 0]

  for r in range(1, rows + 1):

    for c in range(1, cols + 1):

      col = catg\_colms[idx]

      idx += 1

      fig.add\_trace(

          go.Histogram(x=df[col], y=great\_customer['great\_customer\_class']),

          row=r, col=c

      )

      fig.add\_trace(

          go.Histogram(x=df[col], y=not\_great\_customer['great\_customer\_class']),

          row=r, col=c

      )

      sorted\_values = sorted(df[col].unique())

      fig.update\_xaxes(type='category',categoryorder='array', categoryarray=sorted\_values, row=r, col=c)

      fig.update\_yaxes(title = "Great Customer Class", row=r, col=c)

      if idx == 13:

        break

  fig.update\_layout(

      showlegend=False,

      height=800,

      width=1600,

      title\_text=title,

      title\_x = 0.5,

      title\_y = 0.98,

      title\_font\_size = 22

  )

  fig.show()

target\_category\_hist(g\_c\_pre\_process)

## \*\*Statistical test to select feature\*\*

#### \*\*Doing Anova Test to numeric columns to determine the significance of predicting if a customer is great\_customer or not\*\*

#### \*\*Implementing Anova test with both our and built in functions and comparing it\*\*

# Extracting the numerical columns

HC\_anova = []

anova = []

for col in num\_colms[ :- 1]:

  col\_0 = g\_c\_pre\_process[g\_c\_pre\_process["great\_customer\_class"] == 0][col]

  col\_1 = g\_c\_pre\_process[g\_c\_pre\_process["great\_customer\_class"] == 1][col]

  f\_stat, p\_value = f\_oneway(col\_0, col\_1)

  anova.append([col, f\_stat, p\_value])

  \_, f\_stat, p\_value = anova\_one(col\_0, col\_1)

  HC\_anova.append([col, f\_stat, p\_value])

# col = 'mins\_beerdrinking\_year'

# col\_0 = g\_c\_no\_outl[g\_c\_no\_outl["great\_customer\_class"] == 0]

# col\_1 = g\_c\_no\_outl[g\_c\_no\_outl["great\_customer\_class"] == 1][col]

# g\_c\_no\_outl.head()

#### \*\*Comparing my anova function to original anova\*\*

for i in range(len(anova)):

  print(anova[i][0])

  print(f"stat = {anova[i][1]}, p\_value = {anova[i][2]}")

  print(f"my\_stat = {HC\_anova[i][1]}, my\_p\_value = {HC\_anova[i][2]}")

  print()

### \*\*Comparing my Chi2 function with built in functions\*\*

observe = cross\_tab(g\_c\_pre\_process['office\_jobs'], g\_c\_pre\_process['great\_customer\_class'])

expected = to\_expected(observe)

chi\_square,p\_value,\_,\_ = chi2\_contingency(observe, correction=False)

HC\_chi\_square, HC\_p\_values = chi\_2\_t(observe, expected)

print(chi\_square,p\_value)

print(HC\_chi\_square, HC\_p\_values)

# Fetching the categorical columns

n = len(catg\_colms)

HC\_chi\_2 = []

chi\_2 = []

for col in catg\_colms[:-1]:

  observe = cross\_tab(g\_c\_pre\_process[col], g\_c\_pre\_process['great\_customer\_class'])

  expected = to\_expected(observe)

  chi\_square,p\_value,\_,\_ = chi2\_contingency(observe, correction=False)

  HC\_chi\_square, HC\_p\_values = chi\_2\_t(observe, expected)

  chi\_2.append([col, chi\_square, p\_value])

  HC\_chi\_2.append([col, HC\_chi\_square, HC\_p\_values])

for i in range(len(chi\_2)):

  print(HC\_chi\_2[i][0])

  print(f"stat    = {chi\_2[i][1]},  p\_value    = {chi\_2[i][2]}")

  print(f"my\_stat = {HC\_chi\_2[i][1]}, my\_p\_value = {HC\_chi\_2[i][2]}")

  print()

p\_values\_num = {anova[i][0] : anova[i][2] for i in range(len(anova))}

p\_values\_catg = {HC\_chi\_2[i][0] : HC\_chi\_2[i][2] for i in range(len(HC\_chi\_2))}

p\_values  = dict(p\_values\_num)

p\_values.update(p\_values\_catg)

sort\_p\_values = sorted(p\_values.items(), key = lambda x: x[1])

sort\_p\_values

df = g\_c\_no\_outl.copy()

df.isnull().mean() \* 100

df.head()

df[num\_colms].describe()

df.info()

#### \*\*1. Distribution of age among the great customers\*\*

great\_customer = df[df['great\_customer\_class'] == 1]

fig = px.histogram(great\_customer, x='age', nbins=20, color = "sex\_Male")

fig.update\_layout(

    xaxis\_title='Age', yaxis\_title='Count',

    title = {

        'text' : 'Distribution of Age Among Great Customers',

        'x' : 0.5,

        'y' : 0.98,

        'font\_family' : 'bold',

        'font\_size' : 22

    },

    height = 600,

    width = 1200

)

fig.show()

#### \*\*2. Is there a correlation between an individual's age and salary?\*\*

fig, axes = plt.subplots(1, 1, figsize=(10, 5))

corr\_matrix = df[['age', 'salary']].corr()

sns.heatmap(corr\_matrix, ax= axes, annot=True, cmap=plt.cm.CMRmap\_r)

plt.show()

fig = px.scatter(df, x = 'age', y = 'salary', color = 'education\_rank')

corr\_coef = df['age'].corr(df['education\_rank'])

fig.update\_layout(

    title={

        'text' : f'Relation Between Individuals Age and Salary {corr\_coef:.2f}',

        'x'    : 0.5,

        'y'    : 0.93,

        'font\_family' : 'bold',

        'font\_size' : 22

    },

    height = 500,

    width = 1000

)

fig.show()

#### \*\*3. Is there a significant difference in the average salary between males and females\*\*

male\_salary = df[df['sex\_Male'] == 1]['salary']

female\_salary = df[df['sex\_Male'] == 0]['salary']

t\_stat, p\_value = ttest\_ind(female\_salary, male\_salary, equal\_var=False)

t\_stat, p\_value

#### \*\*4. What is the most common occupation among individuals who are self-employed\*\*

self\_employed = df[(df['workclass\_private'] == 0) & (df['workclass\_government'] == 0)]

office\_jobs = self\_employed[self\_employed.office\_jobs == 1]

sales\_jobs = self\_employed[self\_employed.sales\_jobs == 1]

manual\_jobs = self\_employed[self\_employed.manual\_jobs == 1]

service\_jobs = self\_employed[self\_employed.service\_jobs == 1]

Tech\_jobs = self\_employed[self\_employed.Tech\_jobs == 1]

data = pd.DataFrame({

    'Occupation' : [len(office\_jobs),len(sales\_jobs),len(manual\_jobs),len(service\_jobs), len(Tech\_jobs)]

}, index = ['office\_jobs', 'manual\_jobs', 'sales\_jobs', 'service\_jobs', 'Tech\_jobs']).sort\_values(by = 'Occupation', ascending = False)

trace = go.Bar(x=data.index, y=data['Occupation'], texttemplate='%{y:.2s}', textposition='auto')

fig = go.Figure(data=[trace])

fig.update\_xaxes()

fig.update\_layout(

    title = {

        'text' : 'Occupations of Self-Employed Individuals',

        'x' :  0.5,

        'y' : 0.95,

        'font\_family' : 'bold',

        'font\_size' : 22

    }

)

fig.show()

#### \*\*5. Are individuals who work in government jobs more likely to consume coffee or tea, and how does this vary by race?\*\*

government\_df = df[df['workclass\_government'] == 1]

race\_coffee\_tea = government\_df.groupby('race\_caucasian')[['coffee\_per\_year', 'tea\_per\_year']].mean()

trace1 = go.Bar(x=race\_coffee\_tea.index, y=race\_coffee\_tea['coffee\_per\_year'], name='Coffee')

trace2 = go.Bar(x=race\_coffee\_tea.index, y=race\_coffee\_tea['tea\_per\_year'], name='Tea')

data = [trace1, trace2]

fig = go.Figure(data=data)

fig.update\_layout(

    title={

        'text' : 'Average Coffee and Tea Consumption of Government Workers by Race',

        'x'    : 0.5,

        'y'    : 0.93,

        'font\_family' : 'bold',

        'font\_size' : 22

    },

    xaxis={

        'title':'Race',

        'tickvals':[0, 1],

        'ticktext':['Non-Caucasian', 'Caucasian']

    },

    yaxis={

        'title':'Average Consumption'

    }

)

fig.show()

#### \*\*6. Are individuals who are married more likely to have a higher education rank than those who are unmarried, and does this differ by race?\*\*

married = df[df['marital-status\_Married'] == 1]

unmarried = df[(df['marital-status\_Married'] == 0) & (df['marital-status\_Divorced'] == 0) & (df['marital-status\_Widowed'] == 0)]

married\_edu\_rank = married.groupby('race\_caucasian')['education\_rank'].apply(lambda x: x.mode()[0]).reset\_index()

unmarried\_edu\_rank = unmarried.groupby('race\_caucasian')['education\_rank'].apply(lambda x: x.mode()[0]).reset\_index()

trace1 = go.Bar(x=married\_edu\_rank['race\_caucasian'], y=married\_edu\_rank['education\_rank'], name='Married')

trace2 = go.Bar(x=unmarried\_edu\_rank['race\_caucasian'], y=unmarried\_edu\_rank['education\_rank'], name='Unmarried')

data = [trace1, trace2]

fig = go.Figure(data=data)

fig.update\_layout(

    title={

        'text' : ' Most Common Education Rank of Married and Unmarried Individuals by Race',

        'x'    : 0.5,

        'y'    : 0.93,

        'font\_family' : 'bold',

        'font\_size' : 22

    },

    xaxis={

        'title'   :'Race',

        'tickvals':[0, 1],

        'ticktext':['Non-Caucasian', 'Caucasian']

    },

    yaxis={

        'title':'Mode Education Rank'

    }

)

fig.show()

#### \*\*7. Average salary across different job types (office, manual, sales, service, tech)\*\*

jobs = ['office\_jobs', 'manual\_jobs', 'sales\_jobs', 'service\_jobs', 'Tech\_jobs']

avg\_salary = []

for job in jobs:

  avg\_sal = np.mean(df[df[job] == 1]['salary'])

  avg\_salary.append(float(f"{avg\_sal:.2f}"))

salary\_accross\_jobs = pd.DataFrame({

    'AVG\_salary' : avg\_salary

}, index = jobs)

salary\_accross\_jobs

colors = ['#088395', '#FF6969', '#0A4D68', '#41644A', '#FF8400']

fig = px.bar(salary\_accross\_jobs, x='AVG\_salary', text\_auto = True, color = colors)

fig.update\_layout(

    title={

        'text' : 'Average Salary by Job Type',

        'x'    : 0.5,

        'y'    : 0.97,

        'font\_family' : 'bold',

        'font\_size' : 22

    },

    showlegend = False,

    yaxis={

        'title':'Jobs'

    },

    height = 500,

    width = 1200,

    xaxis\_tickformat = '.2f'

)

fig.show()

#### \*\*8. How does the great\_customer\_class vary across different categories such as race and sex?\*\*

great\_customer = df[df['great\_customer\_class'] == 1]

group\_df = great\_customer.groupby(['race\_caucasian', 'sex\_Male'])['great\_customer\_class'].count().reset\_index()

fig = px.histogram(great\_customer, x="race\_caucasian", y="great\_customer\_class",

             color='sex\_Male',

             barmode='group',

             histfunc='sum',

             height=400,

             text\_auto=True)

fig.update\_layout(

     title={

        'text' : 'Great Customer Class Count by Race and Gender',

        'x'    : 0.5,

        'y'    : 0.97,

        'font\_family' : 'bold',

        'font\_size' : 22

    },

    xaxis={

        'title':'Race',

        'tickvals':[0, 1],

        'ticktext':['Non-Caucasian', 'Caucasian']

    },

    yaxis = {

        'title' : 'Count of Great Customer Class'

    },

    height = 500,

    width = 1200

)

fig.show()

#### \*\*9. Distribution of office\_jobs, manual\_jobs, sales\_jobs, service\_jobs, and Tech\_jobs across different categories such as race and sex?\*\*

job\_categories = ["office\_jobs", "manual\_jobs", "sales\_jobs", "service\_jobs", "Tech\_jobs"]

office = great\_customer[["race\_caucasian", "sex\_Male", "office\_jobs"]]

fig = px.histogram(office, x=["race\_caucasian", "sex\_Male"], y="office\_jobs",

             color='sex\_Male',

             barmode='group',

             histfunc='sum',

             height=400,

             text\_auto=True)

fig.update\_xaxes(title = "office\_jobs")

fig.update\_xaxes(title = "Race")

fig.update\_layout(

    title = {

        'text' : 'Destributoin of difference jobs across race and gender',

        'font\_family' : 'bold',

        'font\_size' : 22,

        'x' : 0.5,

        'y' : 0.98

    },

    height = 300

)

fig.show()

manual = great\_customer[["race\_caucasian", "sex\_Male", "manual\_jobs"]]

fig = px.histogram(manual, x=["race\_caucasian", "sex\_Male"], y="manual\_jobs",

             color='sex\_Male',

             barmode='group',

             histfunc='sum',

             height=400,

             text\_auto=True)

fig.update\_xaxes(title = "Race")

fig.update\_xaxes(title = "manual\_jobs")

fig.update\_layout(height = 300)

fig.show()

sales = great\_customer[["race\_caucasian", "sex\_Male", "sales\_jobs"]]

fig = px.histogram(sales, x=["race\_caucasian", "sex\_Male"], y="sales\_jobs",

             color='sex\_Male',

             barmode='group',

             histfunc='sum',

             height=400,

             text\_auto=True)

fig.update\_xaxes(title = "Race")

fig.update\_xaxes(title = "sales\_jobs")

fig.update\_layout(height = 300)

fig.show()

service = great\_customer[["race\_caucasian", "sex\_Male", "service\_jobs"]]

fig = px.histogram(service, x=["race\_caucasian", "sex\_Male"], y="service\_jobs",

             color='sex\_Male',

             barmode='group',

             histfunc='sum',

             height=400,

             text\_auto=True)

fig.update\_xaxes(title = "Race")

fig.update\_xaxes(title = "service\_jobs")

fig.update\_layout(height = 300)

fig.show()

Tech = great\_customer[["race\_caucasian", "sex\_Male", "Tech\_jobs"]]

fig = px.histogram(Tech, x=["race\_caucasian", "sex\_Male"], y="Tech\_jobs",

             color='sex\_Male',

             barmode='group',

             histfunc='sum',

             height=400,

             text\_auto=True)

fig.update\_xaxes(title = "Race")

fig.update\_xaxes(title = "Tech\_jobs")

fig.update\_layout(height = 300)

fig.show()

#### \*\*10. The proportion of great customers in each marital status category\*\*

def marital\_status\_proportion(df):

  unmarried = df[(df['marital-status\_Married'] == 0) & (df['marital-status\_Divorced'] == 0) & (df['marital-status\_Widowed'] == 0)]

  married = df[df['marital-status\_Married'] == 1]

  widowed = df[df['marital-status\_Widowed'] == 1]

  divorce = df[df['marital-status\_Divorced'] == 1]

  total\_len = df.shape[0]

  unmarried\_proportion = float(f"{(len(unmarried) / total\_len) \* 100 : .2f}")

  married\_proportion   = float(f"{(len(married) / total\_len)   \* 100: .2f}")

  widowed\_proportion   = float(f"{(len(widowed) / total\_len)   \* 100: .2f}")

  divorce\_proportion   = float(f"{(len(divorce) / total\_len)   \* 100: .2f}")

  labels = ['Unmarried', 'Married', 'Widowed', 'Divorced']

  proportions = [unmarried\_proportion, married\_proportion, widowed\_proportion, divorce\_proportion]

  colors = ['#F15A59', '#393646', '#4287f5', '#FFA500']

  fig = px.bar(

      x=labels,

      y=proportions,

      labels={'x': 'Marital Status', 'y': 'Poportions (%)'},

      color = colors,

      text\_auto = True

  )

  fig.update\_layout(

      title={

          'text' : 'Proportion of Great Customers by Marital Status',

          'x'    : 0.5,

          'y'    : 0.97,

          'font\_family' : 'bold',

          'font\_size' : 22

      },

      yaxis\_tickformat='.2f',

      showlegend = False,

      height = 500,

      width = 1200

  )

  fig.show()

marital\_status\_proportion(great\_customer)

#### \*\*11. The proportion of not great customers in each marital status category\*\*

not\_great\_customer = df[df['great\_customer\_class'] == 0]

marital\_status\_proportion(not\_great\_customer)

**Mobile Price**

pd.set\_option('display.max\_columns', None)

mobile = pd.read\_csv('mobile\_price\_train.csv')

m\_p\_t = mobile.copy()

m\_p\_t.head()

m\_p\_t.isnull().mean() \* 100

m\_p\_t.info()

fig = px.imshow(m\_p\_t.isnull(), color\_continuous\_scale='electric')

fig.update\_layout(

    title = {

        'text' : "Distribution of Null Values",

        'x' : 0.5,

        'y' : 0.98

    },

    width = 900,

)

fig.show()

"""## \*\*Handling Outliers\*\*

### \*\*Plotting the destribution to identify any outliers\*\*

#### \*\*Box Plot\*\*

"""

num\_colms = [col for col in m\_p\_t.columns if m\_p\_t[col].nunique() > 2]

fig = sp.make\_subplots(rows=5, cols=3, vertical\_spacing = 0.14, horizontal\_spacing = 0.05)

idx = 0

for r in range(1, 6):

  for c in range(1, 4):

    col = num\_colms[idx]

    idx+=1

    trace = go.Box(x=m\_p\_t[col], orientation='h', boxpoints='suspectedoutliers', boxmean=True)

    fig.add\_trace(trace, row=r, col=c)

    fig.update\_yaxes(showticklabels=False)

    fig.update\_xaxes(title\_text=col, row=r, col=c)

fig.update\_layout(

    title={

        'text': 'Relationship between Relevant Features and Price Range',

        'y': 0.989,

        'x': 0.5,

        'xanchor': 'center',

        'yanchor': 'top',

        'font' : {

            'color' : '#393646',

            'family' : 'Bold',

            'size' : 26

        }

    },

    showlegend = False,

    height = 800,

    width = 1200

)

fig.show()

"""#### \*\*Violin Plot\*\*"""

num\_colms = [col for col in m\_p\_t.columns if m\_p\_t[col].nunique() > 2]

fig = sp.make\_subplots(rows=5, cols=3, vertical\_spacing = 0.14, horizontal\_spacing = 0.05)

idx = 0

for r in range(1, 6):

  for c in range(1, 4):

    col = num\_colms[idx]

    idx+=1

    trace = go.Violin(x=m\_p\_t[col], meanline\_visible=True, orientation='h', points='suspectedoutliers', box\_visible=True)

    fig.add\_trace(trace, row=r, col=c)

    fig.update\_yaxes(showticklabels=False)

    fig.update\_xaxes(title\_text=col, row=r, col=c)

fig.update\_layout(

    title={

        'text': 'Relationship between Relevant Features and Price Range',

        'y': 0.989,

        'x': 0.5,

        'xanchor': 'center',

        'yanchor': 'top',

        'font' : {

            'color' : '#393646',

            'family' : 'Bold',

            'size' : 26

        }

    },

    showlegend = False,

    height = 1000,

    width = 1200

)

fig.show()

fig, axes = plt.subplots(1, 2, figsize = (10, 5))

fig.suptitle("Distribution of Columns having Outliers")

sns.kdeplot(data=m\_p\_t['fc'],ax = axes[0], label="FC", fill=True)

sns.kdeplot(data=m\_p\_t['px\_height'],ax = axes[1], label="PX Height", fill=True)

plt.tight\_layout(pad = 2.5)

plt.show()

"""### \*\*Using Trimming Method\*\*"""

print(f"FC \n {m\_p\_t['fc'].describe()} \n\n PX Height \n{m\_p\_t['px\_height'].describe()}")

def iqr\_lim(col):

  perct\_25 = col.quantile(0.25)

  perct\_75 = col.quantile(0.75)

  IQR = perct\_75 - perct\_25

  upper\_limit = perct\_75 + (1.5 \* IQR)

  lower\_limit = perct\_25 - (1.5 \* IQR)

  return upper\_limit, lower\_limit

upper\_fc, lower\_fc               = iqr\_lim(m\_p\_t['fc'])

upper\_px\_height, lower\_px\_height = iqr\_lim(m\_p\_t['px\_height'])

print(f"FC Upper Limit        : {upper\_fc}")

print(f"FC Lower Limit        : {lower\_fc}")

print(f"PX Height Upper Limit : {upper\_px\_height}")

print(f"PX Height Lower Limit : {lower\_px\_height}")

"""#### \*\*Showing the outliers rows\*\*"""

m\_p\_t[m\_p\_t.fc > upper\_fc]

m\_p\_t[m\_p\_t.fc < lower\_fc]

m\_p\_t[m\_p\_t['px\_height'] > upper\_px\_height]

m\_p\_t[m\_p\_t['px\_height'] < lower\_px\_height]

"""##### \*\*Trimming FC\*\*"""

def trim\_columns\_plot(col, trim\_col, name, method = "Trim"):

  plt.figure(figsize = (10, 5))

  plt.subplot(2, 2, 1)

  sns.kdeplot(data=col, label=name.upper(), fill=True)

  plt.xlabel(name.upper())

  plt.subplot(2, 2, 2)

  sns.boxplot(col,  orient = 'h')

  plt.xlabel(name.upper())

  plt.subplot(2, 2, 3)

  sns.kdeplot(data=trim\_col, label=name.upper(), fill=True)

  plt.xlabel(f'{method} {name.upper()}')

  plt.subplot(2, 2, 4)

  ax = sns.boxplot(trim\_col, orient = 'h')

  plt.xlabel(f'{method} {name.upper()}')

  plt.tight\_layout(pad = 2.5)

  plt.show()

trim\_m\_p\_t\_fc = m\_p\_t[m\_p\_t['fc'] < upper\_fc]

trim\_columns\_plot(m\_p\_t.fc, trim\_m\_p\_t\_fc.fc, 'fc')

"""##### \*\*Trimming PX Height\*\*"""

trim\_m\_p\_t\_px\_height = m\_p\_t[m\_p\_t['px\_height'] < upper\_px\_height]

trim\_columns\_plot(m\_p\_t['px\_height'], trim\_m\_p\_t\_px\_height['px\_height'], 'px\_height')

"""### \*\*Using Capping Method\*\*"""

capp\_m\_p\_t\_fc = m\_p\_t.copy()

capp\_m\_p\_t\_fc['fc'] = np.where(

    capp\_m\_p\_t\_fc['fc'] > upper\_fc,

    upper\_fc,

    np.where(

        capp\_m\_p\_t\_fc['fc'] < lower\_fc,

        lower\_fc,

        capp\_m\_p\_t\_fc['fc']

    )

)

trim\_columns\_plot(m\_p\_t['fc'], capp\_m\_p\_t\_fc['fc'], 'fc', method = "Capping")

"""#### \*\*Based on the above analysis conducted, it can be concluded that the capping method is useful for the FC column, as this feature contains a greater number of outliers. Conversely, trimming may be more appropriate for the Px\_height feature, as it only contains two outliers.\*\*"""

# Capping the FC columns

m\_p\_t['fc'] = np.where(

    m\_p\_t['fc'] > upper\_fc,

    upper\_fc,

    np.where(

        m\_p\_t['fc'] < lower\_fc,

        lower\_fc,

        m\_p\_t['fc']

    )

)

# Trimming the Px\_height column

m\_p\_t\_no\_outlr = m\_p\_t[m\_p\_t['px\_height'] < upper\_px\_height]

fig = sp.make\_subplots(rows=1, cols=2, vertical\_spacing = 0.14, horizontal\_spacing = 0.05)

trace1 = go.Box(x=m\_p\_t\_no\_outlr['fc'], orientation='h', boxpoints='suspectedoutliers', boxmean=True)

trace2 = go.Box(x=m\_p\_t\_no\_outlr['px\_height'], orientation='h', boxpoints='suspectedoutliers', boxmean=True)

fig.add\_trace(trace1, row=1, col=1)

fig.update\_xaxes(title\_text="FC", row=1, col=1)

fig.add\_trace(trace2, row=1, col=2)

fig.update\_xaxes(title\_text="PX Height", row=1, col=2)

fig.update\_yaxes(showticklabels=False)

fig.update\_layout(

  title={

    'text': 'After Outliers Removal',

    'y': 0.96,

    'x': 0.5,

    'xanchor': 'center',

    'yanchor': 'top',

    'font' : {

        'color' : '#393646',

        'family' : 'Bold',

        'size' : 26

  }

},

showlegend = False,

height = 500,

width = 800

)

fig.show()

catg\_colms = [col for col in m\_p\_t.columns if col not in num\_colms]

catg\_colms

m\_p\_t\_no\_outlr.head(5)

def target\_category\_hist(df, title = "Relationship between Categorical Features and Price Range", break\_point = 13, rows = 2, cols = 3):

  fig = make\_subplots(rows=rows, cols=cols, subplot\_titles=catg\_colms)

  idx = 0

  range\_1 = df[df['price\_range'] == 0]

  range\_2 = df[df['price\_range'] == 1]

  range\_3 = df[df['price\_range'] == 2]

  range\_4 = df[df['price\_range'] == 3]

  for r in range(1, rows + 1):

    for c in range(1, cols + 1):

      col = catg\_colms[idx]

      idx += 1

      fig.add\_trace(

          go.Histogram(x=df[col], y=range\_1['price\_range']),

          row=r, col=c

      )

      fig.add\_trace(

          go.Histogram(x=df[col], y=range\_2['price\_range']),

          row=r, col=c

      )

      fig.add\_trace(

        go.Histogram(x=df[col], y=range\_3['price\_range']),

        row=r, col=c

      )

      fig.add\_trace(

        go.Histogram(x=df[col], y=range\_4['price\_range']),

        row=r, col=c

      )

      sorted\_values = sorted(df[col].unique())

      fig.update\_xaxes(type='category',categoryorder='array', categoryarray=sorted\_values, row=r, col=c)

      fig.update\_yaxes(title = "Price Range", row=r, col=c)

      if idx == 13:

        break

  fig.update\_layout(

      showlegend=False,

      height=800,

      width=1600,

      title\_text=title,

      title\_x = 0.5,

      title\_y = 0.98,

      title\_font\_size = 22,

  )

  fig.show()

target\_category\_hist(m\_p\_t\_no\_outlr)

"""## \*\*Statistical Test to Select Features\*\*

#### \*\*Doing Anova Test to numeric columns to determine the significance of predicting price\_rang\*\*

"""#### \*\*Testing the Accuracy of my function with states.f\_oneway\*\*"""

stdz\_m\_p\_t = m\_p\_t\_no\_outlr.copy()

feature = 'battery\_power'

col\_0 = stdz\_m\_p\_t[m\_p\_t["price\_range"] == 0][feature]

col\_1 = stdz\_m\_p\_t[m\_p\_t["price\_range"] == 1][feature]

col\_2 = stdz\_m\_p\_t[m\_p\_t["price\_range"] == 2][feature]

col\_3 = stdz\_m\_p\_t[m\_p\_t["price\_range"] == 3][feature]

f\_oneway(col\_0, col\_1, col\_2, col\_3)

anova\_one(col\_0, col\_1, col\_2, col\_3)

"""#### \*\*Implementing Anova test with both our and built in functions and comparing it\*\*"""

# Extracting the numerical columns

n = len(num\_colms)

HC\_anova = []

anova = []

for col in num\_colms[ :n - 1]:

  col\_0 = stdz\_m\_p\_t[m\_p\_t["price\_range"] == 0][col]

  col\_1 = stdz\_m\_p\_t[m\_p\_t["price\_range"] == 1][col]

  col\_2 = stdz\_m\_p\_t[m\_p\_t["price\_range"] == 2][col]

  col\_3 = stdz\_m\_p\_t[m\_p\_t["price\_range"] == 3][col]

  f\_stat, p\_value = f\_oneway(col\_0, col\_1, col\_2, col\_3)

  anova.append([col, f\_stat, p\_value])

  \_, f\_stat, p\_value = anova\_one(col\_0, col\_1, col\_2, col\_3)

  HC\_anova.append([col, f\_stat, p\_value])

"""#### \*\*Comparing my anova function to original anova\*\*

"""

for i in range(len(anova)):

  print(anova[i][0])

  print(f"stat = {anova[i][1]}, p\_value = {anova[i][2]}")

  print(f"my\_stat = {HC\_anova[i][1]}, my\_p\_value = {HC\_anova[i][2]}")

  print()

"""### \*\*Doing Chi Square Test to categorical columns to determine the significance of predicting price\_range\*\*

"""### \*\*Comparing my Chi2 function with built in functions\*\*

#### Table Compare

"""

# My functions

HC\_observe = cross\_tab(stdz\_m\_p\_t['dual\_sim'], stdz\_m\_p\_t['price\_range'])

HC\_expected = to\_expected(HC\_observe)

# Built in functions

observe = pd.crosstab(stdz\_m\_p\_t['dual\_sim'], stdz\_m\_p\_t['price\_range'])

expected = chi2\_contingency(observe)[3]

expected = pd.DataFrame(expected, columns = observe.columns, index = observe.index)

print("Ovserve:\n ")

print(f"Hard Code:\n{HC\_observe}")

print(f"\nBuilt IN:\n{observe}")

print(f"\nExpected: ")

print(f"\nHard Code:\n{HC\_expected}")

print(f"\nBuilt IN:\n{expected}")

"""#### \*\*Chi Square value and p value compare\*\*"""

chi\_square,p\_values, \_, \_ = chi2\_contingency(HC\_observe)

HC\_chi\_square, HC\_p\_values = chi\_2\_t(HC\_observe, HC\_expected)

print(f"Built in chi\_2  = {chi\_square},   Built in p value  = {p\_values}")

print(f"Hard Code chi 2 = {HC\_chi\_square},   Hard Code p value = {HC\_p\_values}")

"""#### \*\*Implementing Chi square test with both our and built in functions and comparing it\*\*

"""

# Fetching the categorical columns

catg\_colms = [col for col in stdz\_m\_p\_t.columns if stdz\_m\_p\_t[col].nunique() == 2]

n = len(catg\_colms)

HC\_chi\_2 = []

chi\_2 = []

for col in catg\_colms:

  observe = cross\_tab(stdz\_m\_p\_t[col], stdz\_m\_p\_t['price\_range'])

  expected = to\_expected(observe)

  chi\_square,p\_value, \_, \_ = chi2\_contingency(observe)

  HC\_chi\_square, HC\_p\_values = chi\_2\_t(observe, expected)

  chi\_2.append([col, chi\_square, p\_value])

  HC\_chi\_2.append([col, HC\_chi\_square, HC\_p\_values])

for i in range(len(chi\_2)):

  print(HC\_chi\_2[i][0])

  print(f"stat    = {chi\_2[i][1]},  p\_value    = {chi\_2[i][2]}")

  print(f"my\_stat = {HC\_chi\_2[i][1]}, my\_p\_value = {HC\_chi\_2[i][2]}")

  print()

m\_p\_t.head()

"""## \*\*Feature Selection Based on P Value\*\*

"""

p\_values\_num = {anova[i][0] : anova[i][2] for i in range(len(anova))}

p\_values\_catg = {HC\_chi\_2[i][0] : HC\_chi\_2[i][2] for i in range(len(HC\_chi\_2))}

p\_values  = dict(p\_values\_num)

p\_values.update(p\_values\_catg)

sort\_p\_values = sorted(p\_values.items(), key = lambda x: x[1])

sort\_p\_values

"""

### \*\*Features to work with\*\*

"""

feature\_to\_work\_with = [f[0] for f in sort\_p\_values[:4]]

feature\_to\_work\_with

"""#### \*\*Creating Box Plot to find any outliers (comparing with price\_range)\*\*

"""

relevent\_colms = ['ram' , 'battery\_power' , 'px\_width' ,'px\_height' , 'mobile\_wt' , 'int\_memory' , 'n\_cores', 'sc\_h']

def box\_plot(m\_p\_t, relevent\_colms, title='Relationship between Numerical Features and Price Range') -> None:

  fig = sp.make\_subplots(rows=2, cols=4, vertical\_spacing = 0.14, horizontal\_spacing = 0.05)

  idx = 0

  for r in range(1, 3):

    for c in range(1, 5):

      col = relevent\_colms[idx]

      idx+=1

      trace1 = go.Box(x=m\_p\_t[m\_p\_t.price\_range == 0][col], name='0', notched=True, orientation='h', boxpoints='suspectedoutliers')

      trace2 = go.Box(x=m\_p\_t[m\_p\_t.price\_range == 1][col], name='1', notched=True, orientation='h', boxpoints='suspectedoutliers')

      trace3 = go.Box(x=m\_p\_t[m\_p\_t.price\_range == 2][col], name='2', notched=True, orientation='h', boxpoints='suspectedoutliers')

      trace4 = go.Box(x=m\_p\_t[m\_p\_t.price\_range == 3][col], name='3', notched=True, orientation='h', boxpoints='suspectedoutliers')

      fig.add\_trace(trace1, row=r, col=c)

      fig.add\_trace(trace2, row=r, col=c)

      fig.add\_trace(trace3, row=r, col=c)

      fig.add\_trace(trace4, row=r, col=c)

      fig.update\_yaxes(title\_text='Price Range', row=r, col=c)

      fig.update\_xaxes(title\_text=col, row=r, col=c)

  fig.update\_layout(

      title={

          'text': title,

          'y': 0.989,

          'x': 0.5,

          'xanchor': 'center',

          'yanchor': 'top',

          'font' : {

              'color' : '#393646',

              'family' : 'Bold',

              'size' : 26

          }

      },

      margin=dict(l=50, r=50, t=50, b=50),

      showlegend = False,

      height = 800,

      width = 1200

  )

  fig.show()

box\_plot(stdz\_m\_p\_t, relevent\_colms)

"""### \*\*Handling the Outliers in ['ram', 'px\_height'] Using Capping Method\*\*"""

categories = [0, 1, 2, 3]

for col in ['ram', 'px\_height']:

    for cat in categories:

        # Subset of the data for the current category and column

        subset = stdz\_m\_p\_t[stdz\_m\_p\_t['price\_range'] == cat][col]

        q1 = subset.quantile(0.25)

        q3 = subset.quantile(0.75)

        iqr = q3 - q1

        lower\_bound = q1 - (1.5 \* iqr)

        upper\_bound = q3 + (1.5 \* iqr)

        # Using Capping Method

        subset[subset < lower\_bound] = lower\_bound

        subset[subset > upper\_bound] = upper\_bound

        stdz\_m\_p\_t.loc[m\_p\_t\_no\_outlr['price\_range'] == cat, col] = subset

box\_plot(stdz\_m\_p\_t, relevent\_colms, title = 'Relationship between Numerical Features and Price Range after handing outliers')

"""# \*\*Creating a model to predict the price\_range based on ram, battery\_power, px\_width, px\_height using Decision Tree\*\*"""

"""## \*\*Coding my model from scratch\*\*

#### \*\*Node Class\*\*

"""

class Node():

  def \_\_init\_\_(self, f\_idx = None, threshold = None, left = None, right = None, IG = None,  value = None,):

    # Decision Node

    self.f\_idx = f\_idx

    self.threshold = threshold

    self.IG = IG

    self.left = left

    self.right = right

    # Leaf Node

    self.value = value

"""#### \*\*Decision Tree Class\*\*"""

class Decision\_tree():

  def \_\_init\_\_(self, min\_sample\_split = 2, max\_depth = 2):

    self.root = None

    self.min\_sample\_split = min\_sample\_split

    self.max\_depth = max\_depth

  def make\_tree(self, df, cur\_depth = 0):

    X, Y = df[:, : -1], df[:, -1]

    n\_samp, n\_feature = np.shape(X)

    if n\_samp >= self.min\_sample\_split and cur\_depth <= self.max\_depth:

      best\_split = self.get\_require\_split(df, n\_samp, n\_feature)

      if best\_split['IG'] > 0:

        left = self.make\_tree(best\_split['left\_df'], cur\_depth + 1)

        right = self.make\_tree(best\_split['right\_df'], cur\_depth + 1)

        return Node(best\_split['f\_idx'], best\_split['threshold'], left, right, best\_split['IG'])

    leaf\_node = self.get\_leaf\_val(Y)

    return Node(value = leaf\_node)

  def get\_require\_split(self, df, n\_samp, n\_feature):

    best\_split = {}

    max\_IG = -float("inf")

    for f\_idx in range(n\_feature):

      f\_values = df[:, f\_idx]

      n\_threshholds = np.unique(f\_values)

      for threshold in n\_threshholds:

        left\_df = np.array([r for r in df if r[f\_idx] <= threshold])

        right\_df = np.array([r for r in df if r[f\_idx] > threshold])

        if len(left\_df) > 0 and len(right\_df) > 0:

          parent\_y, l\_child\_y, r\_child\_y = df[:, -1], left\_df[:, -1], right\_df[:, -1]

          cur\_IG = self.information\_gain(parent\_y, l\_child\_y, r\_child\_y)

          if cur\_IG > max\_IG:

            max\_IG = cur\_IG

            best\_split['f\_idx'] = f\_idx

            best\_split['threshold'] = threshold

            best\_split['left\_df'] = left\_df

            best\_split['right\_df'] = right\_df

            best\_split['IG'] = cur\_IG

    return best\_split

  def information\_gain(self, parent, l\_child, r\_child):

    l\_weight = len(l\_child) / len(parent)

    r\_weight = len(r\_child) / len(parent)

    G\_parent = self.gini\_index(parent)

    G\_l\_child = self.gini\_index(l\_child)

    G\_r\_child = self.gini\_index(r\_child)

    IG = G\_parent - (l\_weight \* G\_l\_child + r\_weight \* G\_r\_child)

    return IG

  def gini\_index(self, target):

    categories = np.unique(target)

    gini\_index = 0

    for catg in categories:

      p = len(target[target == catg]) / len(target)

      gini\_index += p\*\*2

    return 1 - gini\_index

  def get\_leaf\_val(self, Y):

    Y = list(Y)

    return max(Y, key = Y.count)

  def fit(self, X, Y):

    df = np.concatenate((X, Y), axis = 1)

    self.root = self.make\_tree(df)

  def predict(self, X):

    pred = [self.get\_pred(x, self.root) for x in X]

    return pred

  def get\_pred(self, x, root):

    if root.value != None:

      return root.value

    f\_val = x[root.f\_idx]

    if f\_val <= root.threshold:

      return self.get\_pred(x, root.left)

    else:

      return self.get\_pred(x, root.right)

data = stdz\_m\_p\_t[['ram', 'battery\_power', 'px\_width', 'px\_height', 'price\_range']]

X = data.iloc[:, :-1].values

Y = data.iloc[:, -1].values.reshape(-1,1)

X\_train, X\_test, Y\_train, Y\_test = train\_test\_split(X, Y, test\_size=.2, random\_state=41)

model = Decision\_tree(min\_sample\_split=2, max\_depth=5)

model.fit(X\_train,Y\_train)

Y\_pred = model.predict(X\_test)

for x in range(len(Y\_test[:100])):

  print(Y\_test[x], end = ' ')

  if x % 10 == 0 and x != 0:

    print()

for x in range(len(Y\_pred[:100])):

  print(Y\_test[x], end = ' ')

  if x % 10 == 0 and x != 0:

    print()

X\_test[:10, :]

"""### \*\*Comparing the Accuracy of my model vs sklearn model\*\*"""

from sklearn.tree import DecisionTreeClassifier

clf = DecisionTreeClassifier(max\_depth=5, min\_samples\_split=2)

clf.fit(X\_train, Y\_train)

y\_pred2 = clf.predict(X\_test)

"""#### \*\*Sklearn\*\*"""

accuracy\_score(Y\_test, y\_pred2)

"""#### \*\*Ours\*\*"""

accuracy\_score(Y\_test, Y\_pred)

"""# \*\*Hypothesis Test\*\*"""

stdz\_m\_p\_t.head()

"""## \*\*Is the mean battery power of mobile phones with dual SIM slots significantly different from the mean battery power of mobile phones without dual SIM slots?\*\*"""

bp\_dual\_sim = stdz\_m\_p\_t[stdz\_m\_p\_t.dual\_sim == 1]['battery\_power']

bp\_sing\_sim = stdz\_m\_p\_t[stdz\_m\_p\_t.dual\_sim == 0]['battery\_power']

"""#### \*\*Doing two t test\*\*"""

def two\_samp\_t\_test(catg1, catg2, tail = 2):

  n1 = len(catg1)

  n2 = len(catg2)

  x1 = np.mean(catg1)

  x2 = np.mean(catg2)

  s1 = np.std(catg1)

  s2 = np.std(catg2)

  numerator = x1 - x2

  dnumarator = np.sqrt((np.power(s1, 2) / n1) + (np.power(s2, 2) / n2))

  z = numerator / dnumarator

  p\_value = 1 - norm.cdf(abs(z))

  if tail == 2: p\_value \*= 2

  return z, p\_value

"""### \*\*Comparing my function to built in function\*\*"""

test\_stat, p\_val = two\_samp\_t\_test(bp\_dual\_sim, bp\_sing\_sim)

print(f"Ttest\_indResult(statistic={test\_stat}, pvalue={p\_val})")

ttest\_ind(a = bp\_dual\_sim, b = bp\_sing\_sim)

"""## \*\*Is the mean weight of mobile phones with touch screens significantly different from the mean weight of mobile phones without touch screens?\*\*

"""

wth\_touch\_scrn =  stdz\_m\_p\_t[stdz\_m\_p\_t.touch\_screen == 1]['mobile\_wt']

wthout\_touch\_scrn =  stdz\_m\_p\_t[stdz\_m\_p\_t.touch\_screen == 0]['mobile\_wt']

"""### \*\*Comparing my function to built in function\*\*"""

test\_stat, p\_val =  two\_samp\_t\_test(wth\_touch\_scrn, wthout\_touch\_scrn)

print(f"Ttest\_indResult(statistic={test\_stat}, pvalue={p\_val})")

ttest\_ind(a = wth\_touch\_scrn, b = wthout\_touch\_scrn)

"""# \*\*My functions\*\*"""

def mean(col):

  col = [v for v in col if v is not None]

  if not col:

      return None

  return np.sum(col) / len(col)

def median(col):

  col = [v for v in col if v is not None]

  if not col:

      return None

  sorted\_values = sorted(col)

  n = len(sorted\_values)

  middle = n // 2

  if n % 2 == 0:

      return (sorted\_values[middle - 1] + sorted\_values[middle]) // 2

  else:

      return sorted\_values[middle]

def std\_var(col, type = "std"):

  col = [v for v in col if v is not None]

  if not col:

      return None

  mean = np.sum(col) / len(col)

  squared\_diffs = [(v - mean)\*\*2 for v in col]

  variance = sum(squared\_diffs) / len(squared\_diffs)

  if type == "var":

    return variance

  return np.sqrt(variance)

print(f"Our Mean           : {mean(df['battery\_power'])}")

print(f"Built in Mean      : {np.mean(df['battery\_power'])}")

print()

print(f"Our Median         : {median(df['battery\_power'])}")

print(f"Built in Median    : {np.median(df['battery\_power'])}")

print()

print(f"Our Std            : {std\_var(df['battery\_power'])}")

print(f"Built in Std       : {np.std(df['battery\_power'])}")

print()

variance = std\_var(df['battery\_power'], "var")

print(f"Our Variance       : {variance}")

print(f"Built in Variane   : {np.var(df['battery\_power'])}")

arr = np.array([3, 3, 4, 6, 5, 6, 9])

np.median(arr)

print(median(arr))

np.std(wth\_touch\_scrn)

print(std\_var(wth\_touch\_scrn))

"""# \*\*Relevant Exploratory Data Analysis(previous):\*\*"""

"""#### \*\*1. How does the battery power vary across different price ranges?\*\*"""

fig = sp.make\_subplots(rows=1, cols=1, vertical\_spacing = 0.14, horizontal\_spacing = 0.05)

trace1 = go.Box(x=df[df.price\_range == 0][col], name='0', notched=True, orientation='h', boxpoints='suspectedoutliers')

trace2 = go.Box(x=df[df.price\_range == 1][col], name='1', notched=True, orientation='h', boxpoints='suspectedoutliers')

trace3 = go.Box(x=df[df.price\_range == 2][col], name='2', notched=True, orientation='h', boxpoints='suspectedoutliers')

trace4 = go.Box(x=df[df.price\_range == 3][col], name='3', notched=True, orientation='h', boxpoints='suspectedoutliers')

fig.add\_trace(trace1, row=1, col=1)

fig.add\_trace(trace2, row=1, col=1)

fig.add\_trace(trace3, row=1, col=1)

fig.add\_trace(trace4, row=1, col=1)

fig.update\_yaxes(title\_text='Price Range', row=r, col=c)

fig.update\_xaxes(title\_text=col, row=r, col=c)

fig.update\_layout(

  title={

      'text': 'Variation of battery power across different price-range',

      'y': 0.989,

      'x': 0.5,

      'xanchor': 'center',

      'yanchor': 'top',

      'font' : {

          'color' : '#393646',

          'family' : 'Bold',

          'size' : 22

      }

  },

  margin=dict(l=50, r=50, t=50, b=50),

  showlegend = False,

  height = 500,

  width = 700

)

fig.show()

# box\_plot(stdz\_m\_p\_t, relevent\_colms)

"""#### \*\*2. What is the relationship between pixel height (px\_height) and pixel width (px\_width)\*\*"""

px\_height = df['px\_height']

px\_width = df['px\_width']

fig = go.Figure()

fig.add\_trace(go.Scatter(

    x=px\_width,

    y=px\_height,

    mode='markers',

    marker={

        'size':5,

        'color':'blue'

    }

))

corr = px\_height.corr(px\_width)

fig.update\_layout(

    title = {

        'text' : f'Correlatoin between px\_height and px\_width: {corr:.2f}',

        'x' : 0.5,

        'y' : 0.96,

        'font' : {

            'family' : 'Bold',

            'size' : 22

        }

    },

    xaxis\_title='Pixel Width',

    yaxis\_title='Pixel Height'

)

fig.show()

"""3.Distribution of price ranges across different categorical features\*\*

"""

categorical\_columns = ['dual\_sim', 'touch\_screen', 'wifi']

colors = ['#F15A59', '#393646', '#4287f5', '#FFA500']

fig = sp.make\_subplots(rows=1, cols=3)

for i, column in enumerate(categorical\_columns, 1):

    counts = df.groupby([column, 'price\_range']).size().unstack()

    for j, category in enumerate(counts.columns):

        fig.add\_trace(go.Bar(

            x=counts.index,

            y=counts[category],

            name=f'Price Range {category}',

            legendgroup=f'Price Range {category}',

            marker\_color=colors[j],

            text=counts[category],

            showlegend=(i == 1)

        ), row=1, col=i)

    fig.update\_xaxes(title\_text=column, row=1, col=i)

    fig.update\_yaxes(title\_text='Count', row=1, col=i)

fig.update\_layout(

    title = {

        'text' : 'Price Range Distribution by Categorical Features',

        'x' : 0.5,

        'y' : 0.96,

        'font' : {

            'family' : 'Bold',

            'size' : 22

        }

    }

)

fig.show()

"""\*4. What is the frequency of 4G connectivity for different price range categories\*\*

"""

four\_g = df['four\_g']

price\_range = df['price\_range']

four\_g\_counts = df.groupby(['four\_g', 'price\_range']).size().unstack()

category\_colors = ['#60424C' , '#984A59','#FF5959' , '#FF8F56']

fig = go.Figure()

for j, category in enumerate(four\_g\_counts.columns):

    fig.add\_trace(go.Bar(

        x=four\_g\_counts.index,

        y=four\_g\_counts[category],

        name=f'Price Range {category}',

        marker\_color=category\_colors[j % len(category\_colors)],

        text=four\_g\_counts[category],

        textposition='auto'

    ))

fig.update\_yaxes(title\_text='Count')

fig.update\_layout(

    title = {

        'text' : 'Frequency of 4G Connectivity by Price Range',

        'x' : 0.5,

        'y' : 0.96,

        'font' : {

            'family' : 'Bold',

            'size' : 22

        }

    },

    xaxis={

        'tickvals': [0, 1],

        'ticktext': ['3G', '4G'],

    }

)

fig.show()

"""\*5. How does the weight of the mobile device ('mobile\_wt') differ across price ranges?\*\*

"""

fig = px.histogram(

    df,

    x='mobile\_wt',

    color='price\_range',

    nbins=20,

    labels={'mobile\_wt': 'Mobile Device Weight', 'count': 'Count', 'price\_range': 'Price Range'},

)

fig.update\_layout(

    bargap=0.1,

    title = {

        'text' : 'Distribution of Mobile Device Weight by Price Range',

        'x' : 0.5,

        'y' : 0.96,

        'font' : {

            'family' : 'Bold',

            'size' : 22

        }

    }

)

fig.show()

"""\*6. Distribution of the front camera megapixels ('fc') for different price range categories\*\*

"""

price\_range\_categories = np.sort(df['price\_range'].unique())

colors = ['#FF8400', '#FC2947', '#4287f5', '#000000']

fig = go.Figure()

for category, color in zip(price\_range\_categories, colors):

    data = df[df['price\_range'] == category]['fc']

    fig.add\_trace(go.Histogram(

        x=data,

        name=f'Price Range {category}',

        marker\_color=color,

        opacity=0.7

    ))

fig.update\_layout(

    title = {

        'text' : 'Distribution of Front Camera Megapixels by Price Range',

        'x' : 0.5,

        'y' : 0.96,

        'font' : {

            'family' : 'Bold',

            'size' : 22

        }

    },

    xaxis\_title='Front Camera Megapixels',

    yaxis\_title='Count',

    barmode='overlay',

    legend={

        'title':'Price Range',

        'x' : 0.85,

        'y' : 0.9

    }

)

fig.show()

"""7. Visualizing the relationship between the internal memory (int\_memory) and the price range using a violin plot\*\*

"""

fig = go.Figure(

colors\_7 = ['#FFC600', '#F55353', '#9254C8', '#77D970']

for category in sorted(df['price\_range'].unique()):

    data = df[df['price\_range'] == category]['int\_memory']

    fig.add\_trace(go.Box(

        y=data,

        name=f"{category}",

        fillcolor=colors\_7[category],

        line\_color='black'

    ))

fig.update\_layout(

    title={

        'text': 'Relationship between Internal Memory and Price Range',

        'x': 0.5,

        'y': 0.96,

        'font': {

            'family': 'Bold',

            'size': 22

        }

    },

    xaxis\_title='Price Range',

    yaxis\_title='Internal Memory',

    boxmode='group',

    legend={

        'title': 'Price Range',

        'x': 0.75,

        'y': 1

    },

    width=1000

)

fig.show()

"""8. Variation of clock speed across different price ranges\*\*

"""

fig = px.box(df, x="price\_range", y="clock\_speed", color="price\_range",

             labels={"price\_range": "Price Range", "clock\_speed": "Clock Speed"})

fig.update\_layout(

    title = {

        'text' : 'Variation of Clock Speed across Price Ranges',

        'x' : 0.5,

        'y' : 0.96,

        'font' : {

            'family' : 'Bold',

            'size' : 22

        }

    },

    showlegend=False

)

fig.show()

"""\*9. Distribution of talk time for mobile devices\*\*

"""

talk\_time = df['talk\_time']

fig = px.histogram(df, x='talk\_time',color = 'price\_range', nbins=30)

fig.update\_layout(

    title = {

        'text' : 'Distribution of Talk Time for Mobile Devices across different price range',

        'x' : 0.5,

        'y' : 0.96,

        'font' : {

            'family' : 'Bold',

            'size' : 22

        }

    },

    xaxis\_title='Talk Time',

    yaxis\_title='Count'

)

fig.show()

"""10. Distribution of mobile device weights across different touch screen types(0, 1)\*\*

"""

weights = df['mobile\_wt']

touch\_screen = df['touch\_screen']

fig = px.box(df, x=touch\_screen, y=weights, color=touch\_screen)

fig.update\_layout(

    title = {

        'text' : 'Distribution of Mobile Device Weights across Touch Screen Types',

        'x' : 0.5,

        'y' : 0.96,

        'font' : {

            'family' : 'Bold',

            'size' : 22

        }

    },

    xaxis\_title  = 'Touch Screen',

    yaxis\_title = 'Mobile Device Weight',

    showlegend=False

)

fig.show()

"""\*11. Correlation between battery power and clock speed using scatter plot\*\*

"""

battery\_power = df['battery\_power']

clock\_speed = df['clock\_speed'

correlation = battery\_power.corr(clock\_speed)

fig = px.scatter(df, x=battery\_power, y=clock\_speed)

fig.update\_layout(

    title = {

        'text' : f'Correlation between Battery Power and Clock Speed {correlation : .3f}',

        'x' : 0.5,

        'y' : 0.96,

        'font' : {

            'family' : 'Bold',

            'size' : 22

        }

    },

    xaxis\_title='Battery Power',

    yaxis\_title='Clock Speed'

)

fig.show()

"""12. Correlation between the amount of internal memory and the screen size (sc\_h \* sc\_w) using scatter plot\*\*

"""

df\_12 = df.copy()

df\_12['screen\_size'] = df\_12['sc\_h'] \* df\_12['sc\_w']

correlation = np.corrcoef(df\_12['int\_memory'], df\_12['screen\_size'])[0][1]

fig = px.scatter(df\_12, x='int\_memory', y='screen\_size')

fig.update\_layout(

    title={

        'text': f'Relationship between Internal Memory and Screen Size {correlation : .3f}',

        'x': 0.5,

        'y': 0.96,

        'font': {'size': 22, 'family': 'Bold'}

    },

    xaxis\_title='Internal Memory (GB)',

    yaxis\_title='Screen Size (pixels)'

)

fig.show()

**Heart Disease**

"""## \*\*Reading Dataset\*\*"""

disease = pd.read\_csv('heart\_disease.csv')

h\_d = disease.copy()

"""## \*\*Checking The Dataset\*\*"""

h\_d.head(10)

h\_d.prevalentHyp.unique()

h\_d.info()

h\_d.describe()

"""## \*\*Checking the percentage of Null values in each feature\*\*"""

h\_d\_nulls = h\_d.isnull().mean() \* 100

h\_d\_nulls

h\_d.isnull().sum()

"""## \*\*Plotting the Missing Values\*\*"""

fig = px.imshow(h\_d.isnull(), color\_continuous\_scale='thermal')

fig.update\_layout(

    title = {

        'text' : "Distribution of Null Values",

        'x' : 0.5,

        'y' : 0.98

    },

    width = 900,

)

fig.show()

"""## \*\*Destribution of Numerical columns with nulls\*\*"""

num\_col\_w\_nulls = ['cigsPerDay' , 'totChol' ,'BMI' ,'heartRate', 'glucose']

def null\_num\_col\_dest(rows, cols, catg\_cols):

  fig = sp.make\_subplots(rows=rows, cols=cols)

  idx  = 0

  for r in range(1, rows + 1):

    for c in range(1, cols + 1):

      col = catg\_cols[idx]

      idx += 1

      data = h\_d[col]

      # data.rename(columns = {'index': col, col : 'frequency'}, inplace = True)

      trace = go.Histogram(x=data, nbinsx=20)

      fig.add\_trace(trace, row = r, col = c)

      fig.update\_xaxes(title\_text=col, row=r, col=c)

      fig.update\_yaxes(title\_text="frequency", row=r, col=c)

      if idx == 5 : break

  fig.update\_layout(

      title={

          'text': "Distribution of numerical features Having NA's ",

          'y': 0.989,

          'x': 0.5,

          'xanchor': 'center',

          'yanchor': 'top',

          'font' : {

              'color' : '#393646',

              'family' : 'Bold',

              'size' : 26

          }

      },

      showlegend = False,

      height = 600,

      width = 1200

  )

  fig.show()

null\_num\_col\_dest(2, 3, num\_col\_w\_nulls)

"""### \*\*Filling the NA's using Mean and Median\*\*"""

num\_col\_w\_nulls

# ['cigsPerDay' , 'totChol' ,'BMI' ,'heartRate', 'glucose']

h\_d\_no\_null = h\_d.copy()

solve\_col = []

for col in num\_col\_w\_nulls:

  mean = h\_d[col].mean()

  median = h\_d[col].median()

  h\_d\_no\_null[col + "\_mean"] = h\_d[col].fillna(mean)

  h\_d\_no\_null[col + "\_median"] = h\_d[col].fillna(median)

  solve\_col.append(col)

  solve\_col.append(col + "\_mean")

  solve\_col.append(col + "\_median")

len(solve\_col)

"""### Checking the basic statistical difference"""

h\_d\_no\_null[solve\_col].describe()

"""### Checking the Variance Difference"""

idx = 1

original = h\_d\_no\_null[solve\_col[0]].var()

for col in solve\_col:

  change = 0

  variance = h\_d\_no\_null[col].var()

  prct = 100 - (variance / original) \* 100

  print(f"{col} Variance: {variance:.2f} ({prct:.2f}% Difference)")

  if idx % 3 == 0:

    change = 1

    print(" ")

  idx += 1

  if change and idx < len(solve\_col):

    original = h\_d\_no\_null[solve\_col[idx - 1]].var()

"""#### \*\*Plotting the Distribution Difference\*\*"""

fig, axs = plt.subplots(2, 3, figsize=(14, 8))

idx = 0

for r in range(2):

  for c in range(3):

    col = num\_col\_w\_nulls[idx]

    sns.kdeplot(data=h\_d\_no\_null[col], ax=axs[r, c], label="Original")

    sns.kdeplot(data=h\_d\_no\_null[col + "\_mean"], ax=axs[r, c], label=col.title() + " Mean")

    sns.kdeplot(data=h\_d\_no\_null[col + "\_median"], ax=axs[r, c], label=col.title() + " Median")

    axs[r, c].set\_title(col.upper() + " Distribution")

    axs[r, c].set\_xlabel(col.upper())

    axs[r, c].legend()

    idx += 1

    if idx == 5: break

fig.suptitle("Distribution difference after fillup the NA's using mean and median", fontsize=16, fontweight="bold")

plt.tight\_layout(pad = 3.0)

plt.show()

mean\_fill = ['totChol', 'BMI', 'heartRate', 'glucose']

median\_fill = 'cigsPerDay'

h\_d\_no\_null[mean\_fill] = h\_d\_no\_null[['totChol\_mean', 'BMI\_mean', 'heartRate\_mean', 'glucose\_mean']]

h\_d\_no\_null[median\_fill] = h\_d\_no\_null['cigsPerDay\_median']

"""## \*\*Handling the missing values of Categorical columns\*\*"""

catg\_col\_w\_nulls = ['education', 'BPMeds']

"""### \*\*Plotting the Destribution of catg columns having null values\*\*"""

def solve\_catg\_col\_dest(catg\_cols, title, rows = 1, cols = 2):

  fig = sp.make\_subplots(rows=rows, cols=cols)

  idx  = 0

  for r in range(1, rows + 1):

    for c in range(1, cols + 1):

      col = catg\_cols[idx]

      idx += 1

      data = h\_d\_no\_null[col].value\_counts().sort\_values(ascending = False).reset\_index()

      data.rename(columns = {'index': col, col : 'frequency'}, inplace = True)

      trace = go.Bar(x = data[col], y = data['frequency'])

      fig.add\_trace(trace, row = r, col = c)

      fig.update\_xaxes(title\_text=col, row=r, col=c)

      fig.update\_yaxes(title\_text="frequency", row=r, col=c)

  fig.update\_layout(

      title={

          'text': title,

          'y': 0.989,

          'x': 0.5,

          'xanchor': 'center',

          'yanchor': 'top',

          'font' : {

              'color' : '#393646',

              'family' : 'Bold',

              'size' : 26

          }

      },

      showlegend = False,

      height = 600,

      width = 1200

  )

  fig.show()

solve\_catg\_col\_dest(catg\_col\_w\_nulls, title = "Distribution of Categorical Columns Having NA's")

h\_d\_no\_null["BPMeds"].value\_counts().sort\_values(ascending = False).reset\_index()

"""### \*\*Checking the Percentage of null values in these columns\*\*"""

h\_d\_no\_null[catg\_col\_w\_nulls].isnull().mean() \* 100

"""### \*\*Handling Occupation Columns (Using Mode to fill NA's)\*\*

"""

solve\_catg\_col = []

for col in catg\_col\_w\_nulls:

  mode = h\_d\_no\_null[col].mode()[0]

  h\_d\_no\_null[col + "\_mode"] = h\_d[col].fillna(mode)

  solve\_catg\_col.append(col)

  solve\_catg\_col.append(col + "\_mode")

solve\_catg\_col

solve\_catg\_col\_dest(solve\_catg\_col, "Distribution of Categorical Cols Before and After Solving NA's", 2, 2)

h\_d\_no\_null[['education', 'BPMeds']] = h\_d\_no\_null[['education\_mode', 'BPMeds\_mode']]

col = h\_d.columns

h\_d\_no\_null = h\_d\_no\_null[col]

"""## \*\*Handling Outliers\*\*

### \*\*Plotting the distribution to identify any outliers\*\*

"""

h\_d\_no\_null.head(3)

num\_colms = ['age', 'totChol', 'cigsPerDay', 'sysBP', 'diaBP', 'BMI', 'heartRate', 'glucose']

fig = sp.make\_subplots(rows=2, cols=4, vertical\_spacing = 0.14, horizontal\_spacing = 0.05)

idx = 0

for r in range(1, 3):

  for c in range(1, 5):

    col = num\_colms[idx]

    idx+=1

    trace = go.Box(x=h\_d\_no\_null[col], orientation='h', boxpoints='suspectedoutliers', boxmean=True)

    fig.add\_trace(trace, row=r, col=c)

    fig.update\_yaxes(showticklabels=False)

    fig.update\_xaxes(title\_text=col, row=r, col=c)

fig.update\_layout(

    title={

        'text': 'Distribution of numerical Columns',

        'y': 0.989,

        'x': 0.5,

        'xanchor': 'center',

        'yanchor': 'top',

        'font' : {

            'color' : '#393646',

            'family' : 'Bold',

            'size' : 26

        }

    },

    showlegend = False,

    height = 800,

    width = 1400

)

fig.show()

"""### \*\*Summery Statistics of outliers in each columns\*\*"""

def get\_limit(col):

  perct\_25 = col.quantile(0.25)

  perct\_75 = col.quantile(0.75)

  IQR = perct\_75 - perct\_25

  upper\_limit = perct\_75 + (1.5 \* IQR)

  lower\_limit = perct\_25 - (1.5 \* IQR)

  return upper\_limit, lower\_limit

cols\_wth\_outl = []

for col in num\_colms:

  data = h\_d\_no\_null[col]

  upper\_bound, lower\_bound = get\_limit(data)

  outliers = data[(data < lower\_bound) | (data > upper\_bound)]

  outlier\_count = len(outliers)

  total\_count = len(data)

  outlier\_percent = outlier\_count/total\_count \* 100

  print(f"{col.title()} Percentage : {outlier\_percent : .2f}%")

  if outlier\_percent != 0:

    cols\_wth\_outl.append(col)

    mean\_outliers = np.mean(outliers)

    median\_outliers = np.median(outliers)

    std\_outliers = np.std(outliers)

    print(f"{col.title()} Mean       : {mean\_outliers : .2f}")

    print(f"{col.title()} Median     : {median\_outliers : .2f}")

    print(f"{col.title()} Std        : {std\_outliers : .2f}")

  print()

cols\_wth\_outl = ['totChol', 'cigsPerDay', 'totChol', 'sysBP', 'diaBP', 'BMI', 'heartRate', 'glucose']

cols\_wth\_outl

h\_d\_no\_null[cols\_wth\_outl].describe()

"""#### \*\*Using Capping Method to handle outliers\*\*"""

def get\_limit(col):

  perct\_25 = col.quantile(0.25)

  perct\_75 = col.quantile(0.75)

  IQR = perct\_75 - perct\_25

  upper\_limit = perct\_75 + (1.5 \* IQR)

  lower\_limit = perct\_25 - (1.5 \* IQR)

  return upper\_limit, lower\_limit

h\_d\_no\_outl = h\_d\_no\_null.copy()

# Iterate through columns with outliers

for col in cols\_wth\_outl:

  upper\_limit, lower\_limit = get\_limit(h\_d\_no\_outl[col])

  # Replace values outside of the upper and lower limits with the respective limit

  h\_d\_no\_outl[col] = np.where(

      h\_d\_no\_outl[col] > upper\_limit,

      upper\_limit,

      np.where(

          h\_d\_no\_outl[col] < lower\_limit,

          lower\_limit,

          h\_d\_no\_outl[col]

      )

  )

h\_d\_no\_outl['cigsPerDay'].unique()

col = h\_d\_no\_outl['cigsPerDay']

print(get\_limit(col))

"""#### \*\*Distribution of columns after solving the outliers\*\* """

def outlier\_dist\_plt(df, cols\_wth\_outl, title, rows = 2, cols = 4):

  fig = sp.make\_subplots(rows=rows, cols=cols)

  idx = 0

  for r in range(1, rows + 1):

    for c in range(1, cols + 1):

      col = cols\_wth\_outl[idx]

      idx += 1

      # trace = go.Box(x  = df[col], histnorm='density')

      trace = go.Box(x = df[col], orientation='h', boxpoints='suspectedoutliers', boxmean=True)

      fig.add\_trace(trace, row = r, col = c)

      fig.update\_xaxes(title = col, row = r, col = c)

  # fig.update\_xaxes(range = [0, 100000], row = 1, col = 1)

  # fig.update\_xaxes(range = [0, 1000], row = 1, col = 3)

  # fig.update\_xaxes(range = [0, 500], row = 2, col = 1)

  fig.update\_xaxes(range = [-50, 50], row = 1, col = 2)

  fig.update\_yaxes(showticklabels=False)

  fig.update\_layout(

      title={

          'text': title,

          'y': 0.989,

          'x': 0.5,

          'xanchor': 'center',

          'yanchor': 'top',

          'font' : {

              'color' : '#393646',

              'family' : 'Bold',

              'size' : 26

          }

      },

      showlegend = False,

      height = 800,

      width = 1200

  )

  fig.show()

outlier\_dist\_plt(h\_d\_no\_outl, cols\_wth\_outl, 'Distributoin of numeric columns After Solving Outliers')

"""## \*\*Distribution of numerical features with respect to TenYearCHD\*\*"""

catg\_colms = [col for col in h\_d\_no\_outl.columns if col not in num\_colms]

num\_colms

h\_d\_no\_outl.head()

def box\_plot(df, relevent\_colms,title = 'Relationship between Numerical Features and Ten Year CHD', rows = 2, cols = 4) -> None:

  fig = sp.make\_subplots(rows=rows, cols=cols, vertical\_spacing = 0.20, horizontal\_spacing = 0.1)

  idx = 0

  for r in range(1, rows + 1):

    for c in range(1, cols + 1):

      col = relevent\_colms[idx]

      idx+=1

      trace1 = go.Box(x=df[df.TenYearCHD == 0][col], name='No', orientation='h', boxpoints='suspectedoutliers')

      trace2 = go.Box(x=df[df.TenYearCHD == 1][col], name='Yes', orientation='h', boxpoints='suspectedoutliers')

      fig.add\_trace(trace1, row=r, col=c)

      fig.add\_trace(trace2, row=r, col=c)

      fig.update\_yaxes(title\_text='Ten Year CHD', row=r, col=c)

      fig.update\_xaxes(title\_text=col, row=r, col=c)

  # fig.update\_xaxes(range = [0, 500], row=2, col=1)

  fig.update\_layout(

      title={

          'text': title,

          'y': 0.98,

          'x': 0.5,

          'xanchor': 'center',

          'yanchor': 'top',

          'font' : {

              'color' : '#393646',

              'family' : 'Bold',

              'size' : 26

          }

      },

      showlegend = False,

      height = 800,

      width = 1200

  )

  fig.show()

box\_plot(h\_d\_no\_outl, num\_colms)

"""### \*\*Handling the outliers of the numerical features with respect to Ten\_Year\_CHD\*\*"""

categories = [0, 1]

cols\_wth\_out = ['totChol' ,'sysBP', 'diaBP', 'BMI']

for col in cols\_wth\_out:

    for cat in categories:

        subset = h\_d\_no\_outl[h\_d\_no\_outl['TenYearCHD'] == cat][col]

        q1 = subset.quantile(0.25)

        q3 = subset.quantile(0.75)

        iqr = q3 - q1

        lower\_bound = q1 - (1.5 \* iqr)

        upper\_bound = q3 + (1.5 \* iqr)

        subset[subset < lower\_bound] = lower\_bound

        subset[subset > upper\_bound] = upper\_bound

        h\_d\_no\_outl.loc[h\_d\_no\_outl['TenYearCHD'] == cat, col] = subset

"""### \*\*Destribution of these features after handling the outliers\*\*"""

box\_plot(h\_d\_no\_outl, num\_colms, title = 'Relationship between Numerical Features and Ten Year CHD after Handling Outliers')

def target\_category\_hist(df, title = "Relationship between Categorical Features and Ten Year CHD", break\_point = 13, rows = 2, cols = 4 ):

  fig = make\_subplots(rows=rows, cols=cols, subplot\_titles=catg\_colms[:-1], horizontal\_spacing = 0.1)

  idx = 0

  great\_customer = df[df['TenYearCHD'] == 1]

  not\_great\_customer = df[df['TenYearCHD'] == 0]

  for r in range(1, rows + 1):

    for c in range(1, cols + 1):

      col = catg\_colms[idx]

      idx += 1

      fig.add\_trace(

          go.Histogram(x=df[col], y=great\_customer['TenYearCHD']),

          row=r, col=c

      )

      fig.add\_trace(

          go.Histogram(x=df[col], y=not\_great\_customer['TenYearCHD']),

          row=r, col=c

      )

      sorted\_values = sorted(df[col].unique())

      fig.update\_xaxes(type='category',categoryorder='array', categoryarray=sorted\_values, row=r, col=c)

      fig.update\_yaxes(title = "Ten Year CHD", row=r, col=c)

      if idx == 13:

        break

  fig.update\_layout(

      showlegend=False,

      height=800,

      width=1600,

      title\_text=title,

      title\_x = 0.5,

      title\_y = 0.98,

      title\_font\_size = 22

  )

  fig.show()

target\_category\_hist(h\_d\_no\_outl)

"""## \*\*Statistical Test to Select Features\*\*

#### \*\*Doing Anova Test to numeric columns to determine the significance of predicting Ten Year CHD\*\*

"""#### \*\*Implementing Anova test with both our and built in functions and comparing it\*\*"""

# Extracting the numerical columns

n = len(num\_colms)

HC\_anova = []

anova = []

for col in num\_colms[ :n - 1]:

  col\_0 = h\_d\_no\_outl[h\_d\_no\_outl["TenYearCHD"] == 0][col]

  col\_1 = h\_d\_no\_outl[h\_d\_no\_outl["TenYearCHD"] == 1][col]

  f\_stat, p\_value = f\_oneway(col\_0, col\_1)

  anova.append([col, f\_stat, p\_value])

  \_, f\_stat, p\_value = anova\_one(col\_0, col\_1)

  HC\_anova.append([col, f\_stat, p\_value])

"""#### \*\*Comparing my anova function to original anova\*\*"""

for i in range(len(anova)):

  print(anova[i][0])

  print(f"stat = {anova[i][1]}, p\_value = {anova[i][2]}")

  print(f"my\_stat = {HC\_anova[i][1]}, my\_p\_value = {HC\_anova[i][2]}")

  print()

"""### \*\*Doing Chi Square Test to categorical columns to determine the significance of predicting Ten Year CHD\*\*

"""### \*\*Comparing my Chi2 functions with built in functions\*\*

"""

chi2\_test = []

HC\_chi2 = []

for col in catg\_colms[:-1]:

  HC\_observe = cross\_tab(h\_d\_no\_outl[col], h\_d\_no\_outl['TenYearCHD'])

  HC\_expected = to\_expected(HC\_observe)

  chi\_square,p\_values, \_, \_ = chi2\_contingency(HC\_observe, correction = False)

  HC\_chi\_square, HC\_p\_values = chi\_2\_t(HC\_observe, HC\_expected)

  chi2\_test.append([chi\_square,p\_values])

  HC\_chi2.append([col, HC\_chi\_square, HC\_p\_values])

for i in range(len(HC\_chi2)):

  print(HC\_chi2[i][0])

  print(f"stat = {chi2\_test[i][0]}, p\_value = {chi2\_test[i][1]}")

  print(f"my\_stat = {HC\_chi2[i][1]}, my\_p\_value = {HC\_chi2[i][2]}") ; print()

## \*\*Feature Selection Based on P value\*\*

"""

p\_values\_num = {anova[i][0] : anova[i][2] for i in range(len(anova))}

p\_values\_catg = {HC\_chi2[i][0] : HC\_chi2[i][2] for i in range(len(HC\_chi2))}

p\_values  = dict(p\_values\_num)

p\_values.update(p\_values\_catg)

sort\_p\_values = sorted(p\_values.items(), key = lambda x: x[1])

sort\_p\_values

"""

# \*\*Relevant Exploratory Data Analysis\*\*

"""

df = h\_d\_no\_outl.copy()

"""\*\*1. What is the relationship between total cholesterol levels and systolic blood pressure, and how does this impact the risk of developing heart disease?\*\*"""

fig = px.scatter(df, x="totChol", y="sysBP", color="TenYearCHD",

                 hover\_name="TenYearCHD", trendline="ols")

corr\_coef = df['totChol'].corr(df['sysBP'])

fig.update\_layout(

    title = {

        'text' : f"Relationship between Total Cholesterol Levels and Systolic Blood Pressure {corr\_coef:.2f}",

        'x' : 0.5,

        'y' : 0.98

    },

    xaxis\_title="Total Cholesterol Levels",

    yaxis\_title="Systolic Blood Pressure"

)

fig.show()

cholesterol\_bp = df[['totChol', 'sysBP']]

corr = cholesterol\_bp.corr(method='pearson').iloc[0,1]

print(corr)

"""

\*\*2. Is there a correlation between BMI and heart rate, and does this vary between smokers and non-smokers?\*\*

"""

fig = px.scatter(df, x='BMI', y='heartRate', color='currentSmoker')

corr\_coef  = df['BMI'].corr(df['heartRate'])

fig.update\_layout(

    title = {

        'text' : f"Relationship between BMI and Heart Rate {corr\_coef:.2f}",

        'x' : 0.5,

        'y' : 0.97,

        'font\_size' : 18

    },

    xaxis\_title='BMI',

    yaxis\_title='Heart Rate'

)

fig.show()

"""3. What is the distribution of glucose levels among patients with and without diabetes, and does this have any impact on the risk of developing heart disease?\*\*

"""

fig = make\_subplots(rows=1, cols=2, subplot\_titles=("Glucose levels among patients with diabetes", "Glucose levels among patients without diabetes"))

fig.add\_trace(go.Histogram(x=h\_d\_no\_outl[h\_d\_no\_outl["diabetes"]==1]["glucose"], nbinsx=20, name="Diabetes"), row=1, col=1)

fig.add\_trace(go.Histogram(x=h\_d\_no\_outl[h\_d\_no\_outl["diabetes"]==0]["glucose"], nbinsx=20, name="No Diabetes"), row=1, col=2)

fig.update\_xaxes(title\_text="Glucose level", row=1, col=1)

fig.update\_xaxes(title\_text="Glucose level", row=1, col=2)

fig.update\_yaxes(title\_text="Count", row=1, col=1)

fig.update\_yaxes(title\_text="Count", row=1, col=2)

"""4. How does the average number of cigarettes smoked per day vary across different age groups in the dataset?\*\*

"""

df = h\_d\_no\_outl.copy()

df["age\_group"] = pd.cut(df["age"], bins=[0,40,50,60,70], labels=["<40","40-49","50-59", "60-69"])

# Create a box plot of cigsPerDay distribution for each age group

fig = px.box(df, x="age\_group", y="cigsPerDay", color="age\_group")

# Set the plot title and axis titles

fig.update\_layout(title = {

                    "text" : "Distribution of cigarettes smoked per day among different age groups",

                    'y'    : 0.98,

                    'x'    : 0.5

                  },

                  xaxis\_title="Age group",

                  yaxis\_title="Number of cigarettes smoked per day",

                  height = 600,

                  width = 1200

)

# Show the plot

fig.show()

"""5. Is there a significant difference in the average BMI between patients who do and do not take blood pressure medication?\*\*

"""

meds = df[df["BPMeds"] == 1]["BMI"]

no\_meds = df[df["BPMeds"] == 0]["BMI"]

fig = make\_subplots(rows = 1, cols = 2)

fig.add\_trace(go.Histogram(x=meds, nbinsx=20, name="Meds"), row=1, col=1)

fig.add\_trace(go.Histogram(x=no\_meds, nbinsx=20, name="No Meds"), row=1, col=2)

fig.update\_xaxes(title\_text="Meds level", row=1, col=1)

fig.update\_xaxes(title\_text="No Meds level", row=1, col=2)

fig.update\_yaxes(title\_text="Count", row=1, col=1)

fig.update\_yaxes(title\_text="Count", row=1, col=2)

fig.update\_layout(

    title = {

        'text' : f"Destribution of Average BMI of patients who do and do not take BP Meds",

        'x' : 0.5,

        'y' : 0.97,

        'font\_size' : 18

    },

)

# The two histograms are drawn on top of another

fig.show()

ttest, p = ttest\_ind(meds, no\_meds, equal\_var=False)

ttest, p

student\_t\_test(meds, no\_meds)

"""6. What is the relationship between age and the total cholesterol levels of patients?\*\*

"""

fig = px.scatter(h\_d\_no\_outl, x="age", y="totChol", title="Age vs Total Cholesterol")

fig.show()

age = df['age']

totChol = df['totChol']

n = len(age)

mean\_age = np.mean(age)

mean\_totChol = np.mean(totChol)

sum\_of\_products = np.sum((age - mean\_age) \* (totChol - mean\_totChol))

sum\_of\_squares\_age = np.sum((age - mean\_age)\*\*2)

sum\_of\_squares\_totChol = np.sum((totChol - mean\_totChol)\*\*2)

correlation = sum\_of\_products / np.sqrt(sum\_of\_squares\_age \* sum\_of\_squares\_totChol)

print(f"Correlation coefficient between age and total cholesterol levels: {correlation:.2f}")

"""7. Is there a significant difference in systolic blood pressure between male and female patients?\*\*

"""

male\_bp = df[df['male'] == 1]['sysBP']

female\_bp = df[df['male'] == 0]['sysBP']

t\_stat, p\_value = ttest\_ind(male\_bp, female\_bp)

print(f"t-statistic: {t\_stat:.2f}, p-value: {p\_value:.5f}")

"""8. What is the average heart rate of patients with diabetes compared to those without diabetes? Is there a significance difference between them?\*\*

"""

diabetes\_hr = h\_d\_no\_outl[h\_d\_no\_outl['diabetes'] == 1]['heartRate']

no\_diabetes\_hr = h\_d\_no\_outl[h\_d\_no\_outl['diabetes'] == 0]['heartRate']

mean\_diabetes\_hr = diabetes\_hr.mean()

std\_diabetes\_hr = diabetes\_hr.std()

n\_diabetes\_hr = len(diabetes\_hr)

mean\_no\_diabetes\_hr = no\_diabetes\_hr.mean()

std\_no\_diabetes\_hr = no\_diabetes\_hr.std()

n\_no\_diabetes\_hr = len(no\_diabetes\_hr)

t\_stat, p\_value = ttest\_ind(diabetes\_hr, no\_diabetes\_hr)

print(f"Average Heart rate patients having diabetes    : {mean\_diabetes\_hr}")

print(f"Average Heart rate patients not having diabetes: {mean\_no\_diabetes\_hr}")

print(f"t-statistic: {t\_stat:.2f}, p-value: {p\_value:.5f}")

"""9. Is there a relationship between the number of cigarettes smoked per day and systolic blood pressure?\*\*

"""

cigs\_per\_day = h\_d\_no\_outl['cigsPerDay']

systolic\_bp = h\_d\_no\_outl['sysBP']

corr\_coef = np.corrcoef(cigs\_per\_day, systolic\_bp)[0, 1]

fig = px.scatter(h\_d\_no\_outl, x="cigsPerDay", y="sysBP")

fig.update\_xaxes(title="Cigarettes per day")

fig.update\_yaxes(title="Systolic blood pressure")

fig.update\_layout(

    title = {

        'text' : f"Correlation coefficient: {corr\_coef:.2f}",

        'x' : 0.5,

        'y' : 0.98

    },

    height = 400, width = 900

    )

fig.show()

"""10.Destribution of age in the dataset in terms of Gender\*\*

"""

fig = px.histogram(df, x='age', nbins=30, color = "male",  title='Age Distribution')

fig.update\_layout(

    title = {

        'text' : "Destribution of Age",

        'x'    : 0.5,

        'y'    : 0.96,

        'font\_size' : 22

    },

    width = 1200

)

fig.show()

"""11. Distribution of education levels Inerms of Gender\*\*

"""

fig = px.histogram(df, x='education', color = "male")

sort\_category = np.sort(df['education'].unique())

fig.update\_layout(

    title = {

        'text' : "Distribution of Education Levels",

        'x'    : 0.5,

        'y'    : 0.96,

        'font\_size' : 22

    },

    width = 1200,

    xaxis\_title\_font = {'size' : 16},

    yaxis\_title\_font = {'size' : 16}

)

fig.update\_xaxes(type='category', categoryorder='array', categoryarray=sort\_category)

fig.show()

"""12. Destribution of smokers interms of diabetes\*\*

"""

fig = px.histogram(df, x='currentSmoker', color = "male", text\_auto = True)

sort\_category = np.sort(df['currentSmoker'].unique())

fig.update\_layout(

    title = {

        'text' : "Destribution of smokers",

        'x'    : 0.5,

        'y'    : 0.96,

        'font\_size' : 22

    },

    width = 1200,

    xaxis={

        'tickvals': [0, 1],

        'ticktext': ['Non Smoker', 'Smoker'],

        'title': '',

        'tickfont' : {'size': 16}

    },

    yaxis\_title\_font = {'size' : 16}

)

fig.update\_xaxes(type='category', categoryorder='array', categoryarray=sort\_category)

fig.show()

"""13. Is there a significant difference in the proportion of smokers between males and females?\*\*

"""

my\_observe = cross\_tab(df['currentSmoker'], df['male'])

my\_expected = to\_expected(my\_observe)

observe = pd.crosstab(df['currentSmoker'], df['male'])

chi\_square,p\_value,\_,\_ = chi2\_contingency(observe, correction=False)

HC\_chi\_square, HC\_p\_values = chi\_2\_t(my\_observe, my\_expected)

print("Built in : ", chi\_square,p\_value)

print("Ours     : ", HC\_chi\_square, HC\_p\_values)

"""14. distribution of the target variable (TenYearCHD)\*\*

"""

have\_TenYearCHD = df[df.TenYearCHD == 1]

not\_have\_TenYearCHD = df[df.TenYearCHD == 0]

male\_chd\_count = have\_TenYearCHD[have\_TenYearCHD['male'] == 1].shape[0]

female\_chd\_count = have\_TenYearCHD[have\_TenYearCHD['male'] == 0].shape[0]

male\_not\_chd\_count = not\_have\_TenYearCHD[not\_have\_TenYearCHD['male'] == 1].shape[0]

female\_not\_chd\_count = not\_have\_TenYearCHD[not\_have\_TenYearCHD['male'] == 0].shape[0]

labels\_chd = ['Male', 'Female']

values\_chd = [male\_chd\_count, female\_chd\_count]

labels\_not\_chd = ['Male', 'Female']

values\_not\_chd = [male\_not\_chd\_count, female\_not\_chd\_count]

fig = make\_subplots(rows=1, cols=2,

                    subplot\_titles=['TenYearCHD - Gender Distribution', 'No TenYearCHD - Gender Distribution'],

                    specs=[[{"type": "pie"}, {"type": "pie"}]])

fig.add\_trace(go.Pie(labels=labels\_chd, values=values\_chd, textinfo='label+percent'), row=1, col=1)

fig.add\_trace(go.Pie(labels=labels\_not\_chd, values=values\_not\_chd, textinfo='label+percent'), row=1, col=2)

fig.update\_annotations(yshift=20)

fig.update\_layout(

    width = 1000

)

fig.show()

"""15.  relationship between age and cigsperday.\*\*

"""

correlation = df['age'].corr(df['cigsPerDay'])

fig = px.scatter(df, x='age', y='cigsPerDay', trendline='ols', color = 'male')

fig.update\_layout(

    title = {

        'text' : f"Relationship between Age and Cigarettes Per Day {correlation : .2f}",

        'x'    : 0.5,

        'y'    : 0.96,

        'font\_size' : 18

    },

    xaxis\_title='Age',

    yaxis\_title='Cigarettes Per Day',

    width = 1000,

    height = 500

)

fig.show()

"""16. Variation Heart Rate across AGE groups\*\*

"""

correlation = df['age'].corr(df['heartRate'])

fig = px.scatter(df, x='age', y='heartRate', trendline='ols', color = 'male')

fig.update\_layout(

    title = {

        'text' : f"Relationship between Age and heartRate {correlation : .2f}",

        'x'    : 0.5,

        'y'    : 0.96,

        'font\_size' : 18

    },

    xaxis\_title='Age',

    yaxis\_title='heartRate',

    width = 1000,

    height = 500

)

fig.show()