USC Viterbi School of Engineering

SAMPLE FINAL REPORT

ARCHITA DHANDE
4844466596

USC Viterbi School of Engineering

SAMPLE FINAL REPORT

OUTLINE

- Exploratory Data Analysis Report
- Data Preparation Plan
- Model Pipeline
- Inference Pipeline
- Summary Discussion

USC Viterbi School of Engineering

EDA REPORT
OUTLINE

- Business objective
- Dataset summary
- Data quality summary
- Univariate analysis
- Bivariate analysis

The model leverages the TenYearCHD variable as the target variable for prediction.

EDA REPORT

BUSINESS OBJECTIVE

- Developing a predictive model to identify 3,816 paitents who are at high risk of developing coronary heart disease (CHD) within ten years.
  - Utilize patient demographic information (male, age, education, income), medical history (currentSmoker, cigsPerDay, BPMeds, prevalentStroke, prevalentHyp, diabetes), and health measurements (totChol, sysBP, diaBP, BMI, heartRate, glucose, a1c) to improve the effectiveness of preventive interventions and healthcare management strategies.
  - The model leverages the TenYearCHD variable as the target variable for prediction.
- This objective aims to develop a model that enhances patient care and promotes proactive health management practices.

EDA REPORT

DATASET SUMMARY

- “Final Project Dataset.csv” – dataset with 19 variables and 3,816 observations
- Dataset contains one potential response variables:

Name	Label	Description
TenYearCHD	Ten Year of Coronary Heart Disease(CHD)	A binary target variable. This column represents whether a patient is at risk of developing coronary heart disease (CHD) within ten years. The values in this column are binary, where 1 indicates that the patient is at risk and 0 indicates that the patient is not at risk.

USC Viterbi School of Engineering

DATASET SUMMARY
DEMOGRAPHIC VARIABLES

The dataset is breakdown into patient demographic information, medical history and health measurements.

1. Patient demographic information

Name	Label	Description
patientID	Patient Identification number	This variable is used to uniquely identify each individual in the dataset.
male	Gender of the patient	A binary categorical variable where 0 represents female and 1 represents male.
age	Age of the patient	A continuous numerical variable that represents the age of the patient.
education	Education of the patient	A categorical variable that is education level of the patient (1 = some high school, 2 = high school or GED, 3 = some college or vocational school, 4 = college).
income	Income of the patient	A continuous variable that is total income of the patient.

2. Medical History

Name	Label	Description
currentSmoker	Whether the patient is smoker	A binary variable that represents 1 as yes the patient is a smoker and 0 if the patient is not a smoker
cigsPerDay	Number of cigarettes smoked per day	A continuous variable that represent number of cigarettes smoked per day.
BPMeds	Whether the patient is on blood pressure medications	A binary variable that represents 1 as yes the patient is on blood pressure medications and 0 if the patient is not on blood pressure medications
prevalentStroke	Whether the patient has a history of stroke	A binary variable that represents 1 as the patient has a history is stroke and 0 if the patient does not have a history of stroke
prevalentHyp	Whether the patient has prevalent hypertension	A binary variable that represents 1 as the patient has a prevalent hypertension and 0 if the patient does not have a prevalent hypertension
diabetes	Whether the patient has diabetes	A binary variable that represents 1 as yes the patient has diabetes and 0 if the patient is not have diabetes

3. Health Measurements

Name	Label	Description
totChol	Total cholesterol level	A continuous variable that represents the total cholesterol level of the patient
sysBP	Systolic blood pressure	A continuous variable that represents the systolic blood pressure of the patient
diaBP	Diastolic blood pressure	A continuous variable that represents the diastolic blood pressure of the patient
BMI	Body mass index (BMI)	A continuous variable that represents the body mass index of the patient
heartRate	Heart rate	A continuous variable that represents the heart rate of the patient
glucose	Glucose level	A continuous variable that represents the glucose level of the patient
a1c	A1 test results	A continuous variable that represents the average blood glucose over past 2-3 month of the patient

EDA REPORT

DATA QUALITY SUMMARY

Missing values:

```
train_df.isnull().sum()
```

```
patientID           0
male                0
age                 0
education          72
currentSmoker       0
cigsPerDay       1549
BPMeds             39
prevalentStroke     0
prevalentHyp        0
diabetes            0
totChol            38
sysBP               0
diaBP               0
BMI                15
heartRate           1
glucose           292
TenYearCHD          0
a1c               292
income              0
dtype: int64
```

In the dataset, there are 8 variables with missing values:

- Education: 72 missing values
- CigsPerDay: 1549 missing values
- BPMeds: 39 missing values
- TotChol: 38 missing values
- BMI: 15 missing values
- HeartRate: 1 missing value
- Glucose: 292 missing values
- A1c: 292 missing values

EDA REPORT

DATA QUALITY SUMMARY

Data types:

```
train_df.dtypes

patientID            int64
male                 int64
age                  int64
education          float64
currentSmoker        int64
cigsPerDay         float64
BPMeds             float64
prevalentStroke      int64
prevalentHyp         int64
diabetes             int64
totChol            float64
sysBP              float64
diaBP              float64
BMI                float64
heartRate          float64
glucose            float64
TenYearCHD           int64
a1c                float64
income             float64
dtype: object
```

Integer Variables:

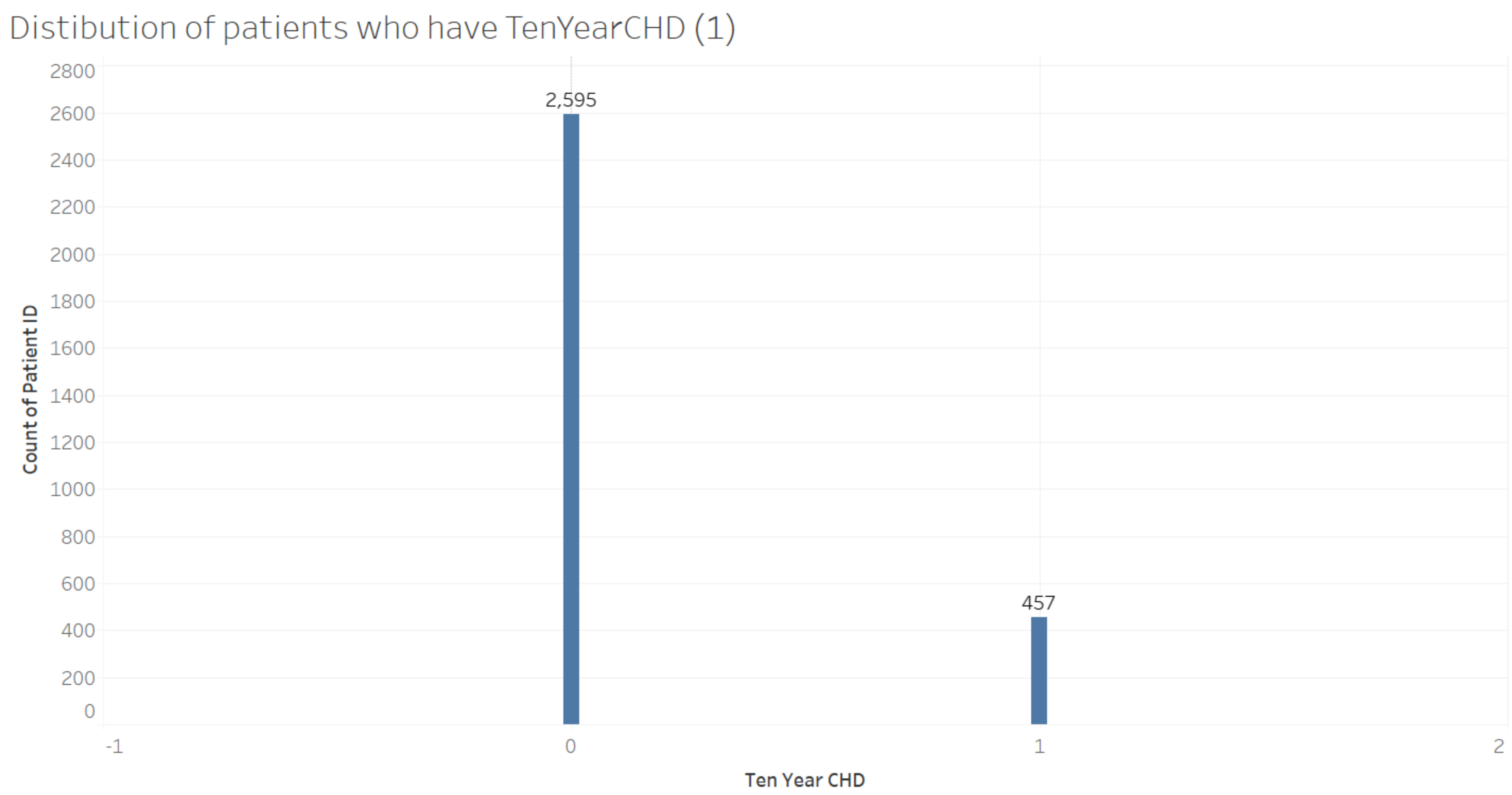
- patientID
- male
- age
- currentSmoker
- prevalentStroke
- prevalentHyp
- diabetes
- TenYearCHD

Float Variables:

- education
- cigsPerDay
- BPMeds
- totChol
- sysBP
- diaBP
- BMI
- heartRate
- glucose
- a1c
- income

EDA REPORT – UNIVARIATE ANALYSIS
RESPONSE VARIABLES

The bar chart illustrates the distribution of patients for a 10-year coronary heart disease risk factor, showing a significant majority (2,595 patients) without the condition (0) compared to 457 patients who have the condition (1).

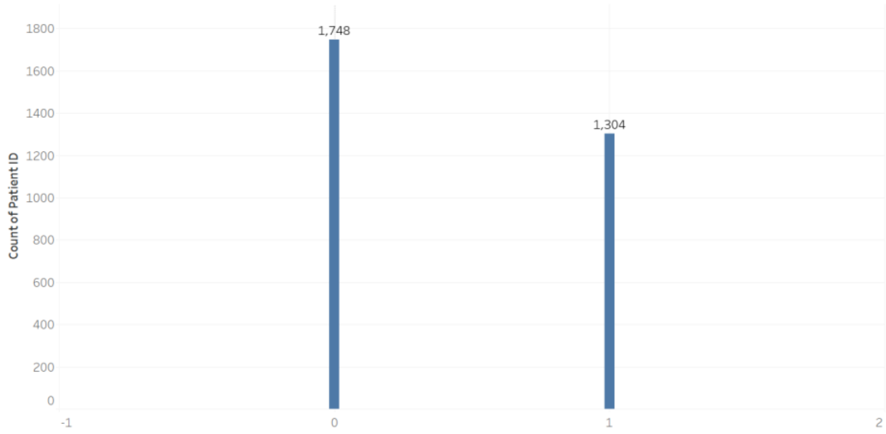


![USC Viterbi School of Engineering]

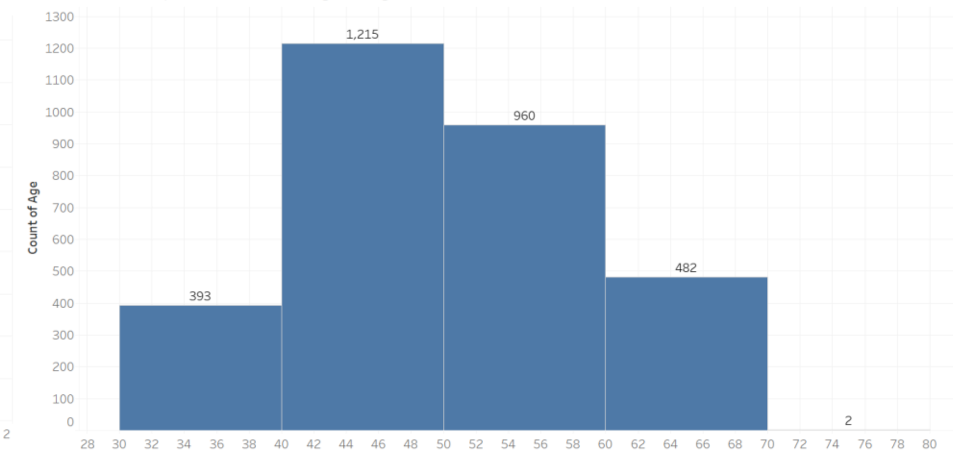
EDA REPORT – UNIVARIATE ANALYSIS

PATIENT DEMOGRAPHICS HISTORY

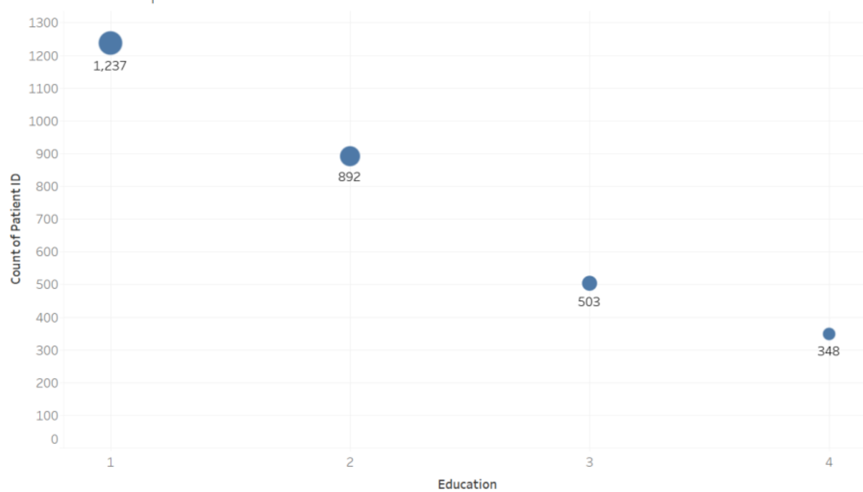
Gender distribution for the patients (1=Male, 0=Female)



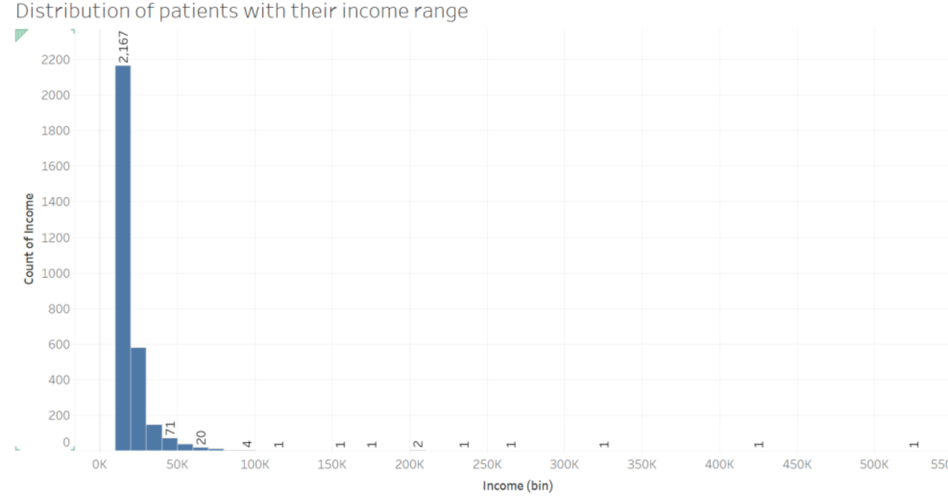
Distribution of patients in the Age range



Distribution of patients wrt their Education



Distribution of patients with their income range

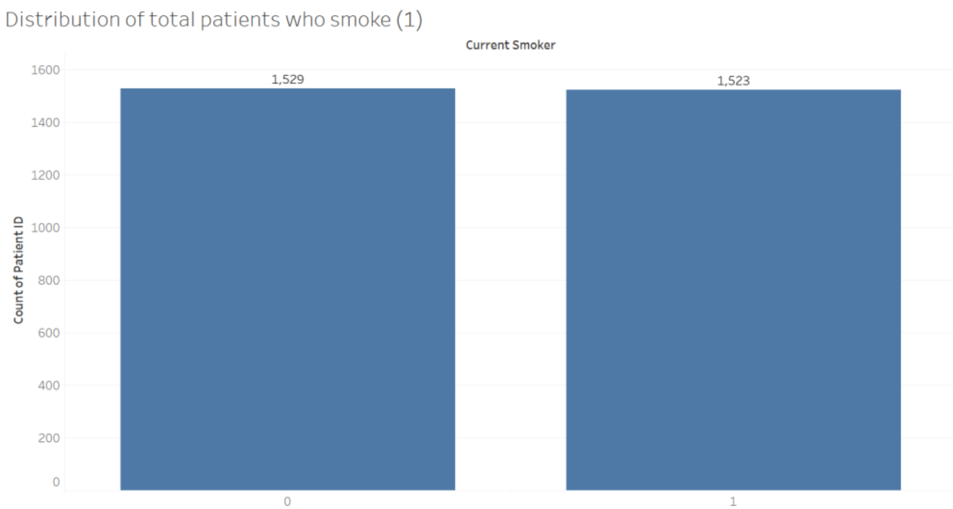


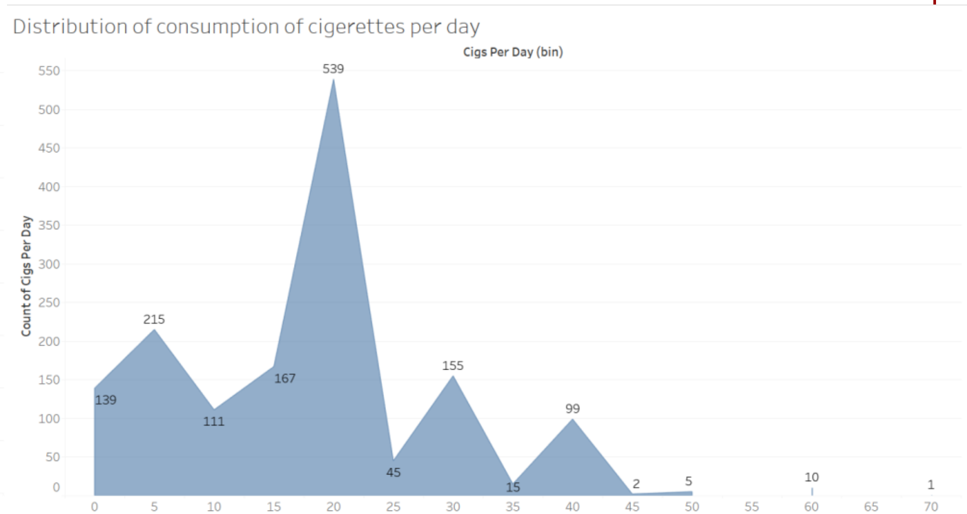
The income is strong skewed

The charts show a balanced gender distribution among patients, peak ages at 46-54, highest education at some college level, and most incomes concentrated in the lower brackets.

EDA REPORT – UNIVARIATE ANALYSIS

MEDICAL HISTORY: SMOKING HISTORY





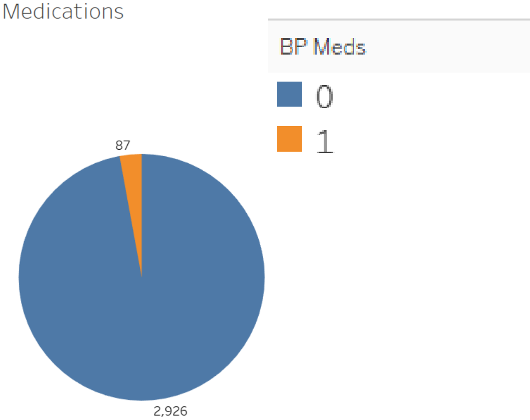
The charts show an equal number of patients who smoke (1,529 each for smokers and non-smokers) and a varied distribution of cigarette consumption per day, peaking at 20 cigarettes.

USC Viterbi School of Engineering

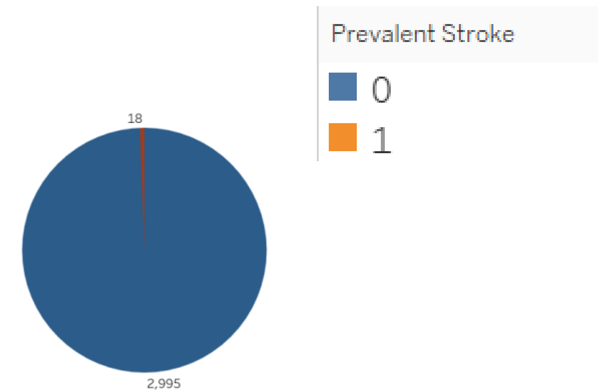
EDA REPORT – UNIVARIATE ANALYSIS

MEDICAL HISTORY

Distribution of Patients on BP Medications

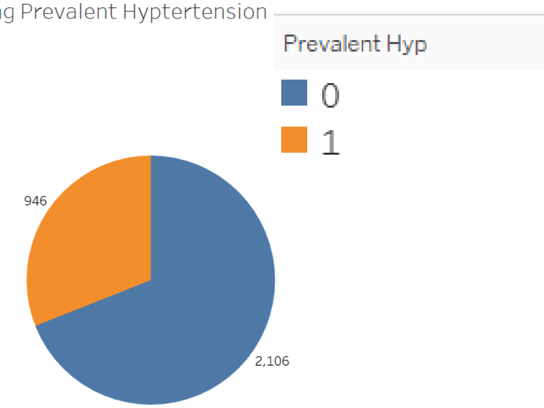


Distribution of Patients on BP Medications

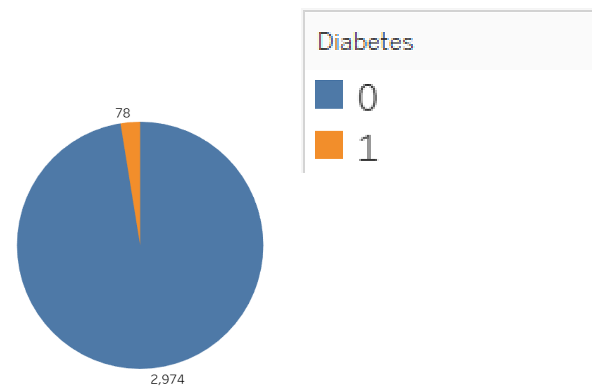


‘1’ is yes the patient has diabetes

Distribution of patients having Prevalent Hyptertension



Distribution of number of patients having Diabetes



‘1’ is yes the patient has prevalent hypertension

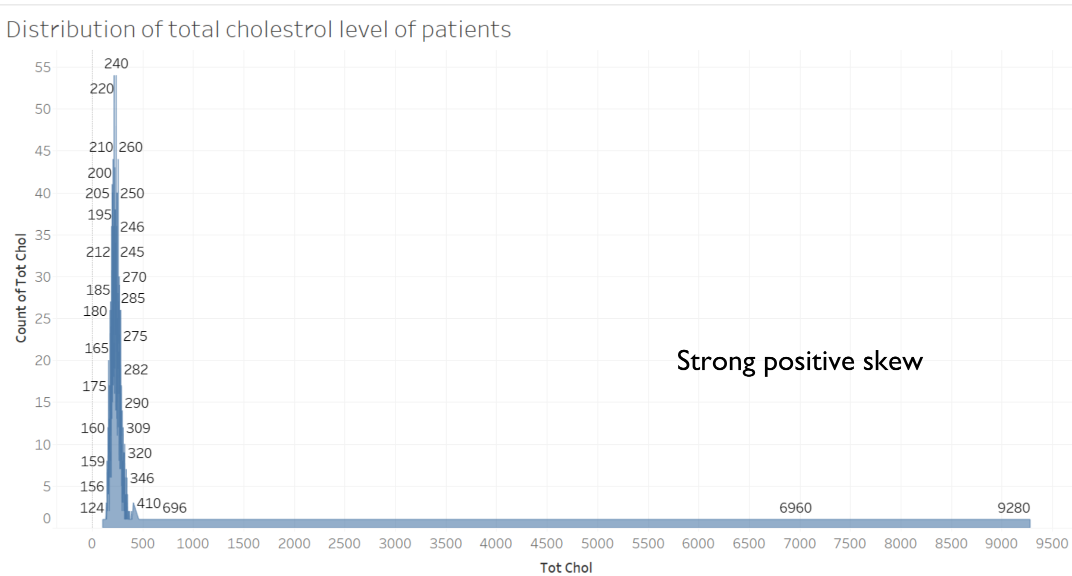
‘1’ is yes the patient has a history of stroke

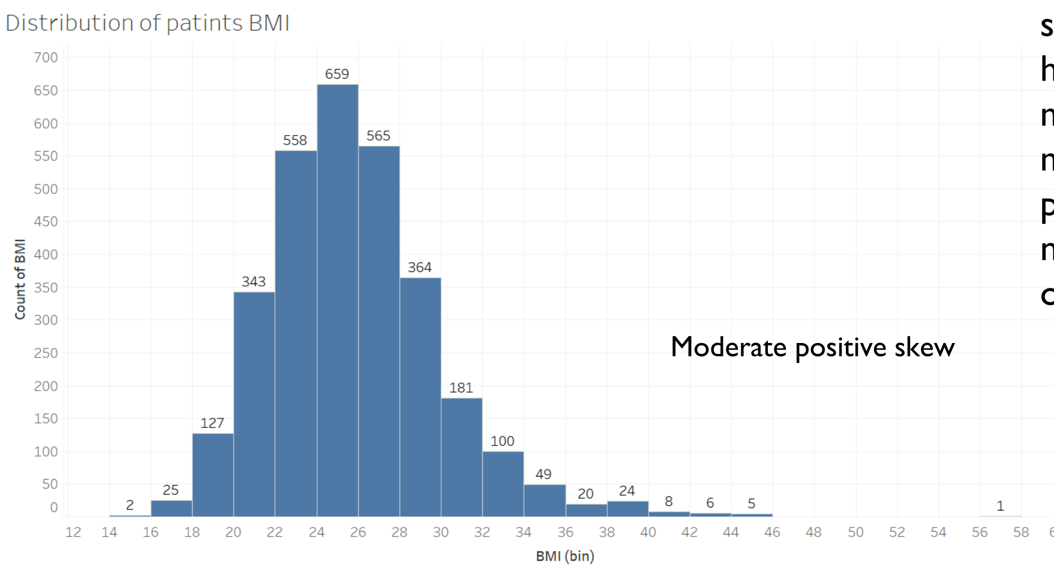
The charts highlight that a small percentage of the patient population use BP medications (3.2%), have prevalent hypertension (31.0%), diabetes (7.6%), or a history of stroke (0.9%).

EDA REPORT – UNIVARIATE ANALYSIS

HEALTH MEASUREMENTS

The calculated skewness values are approximately 1.65 for cholesterol (strong positive skew) and 0.82 for BMI (moderate positive skew).



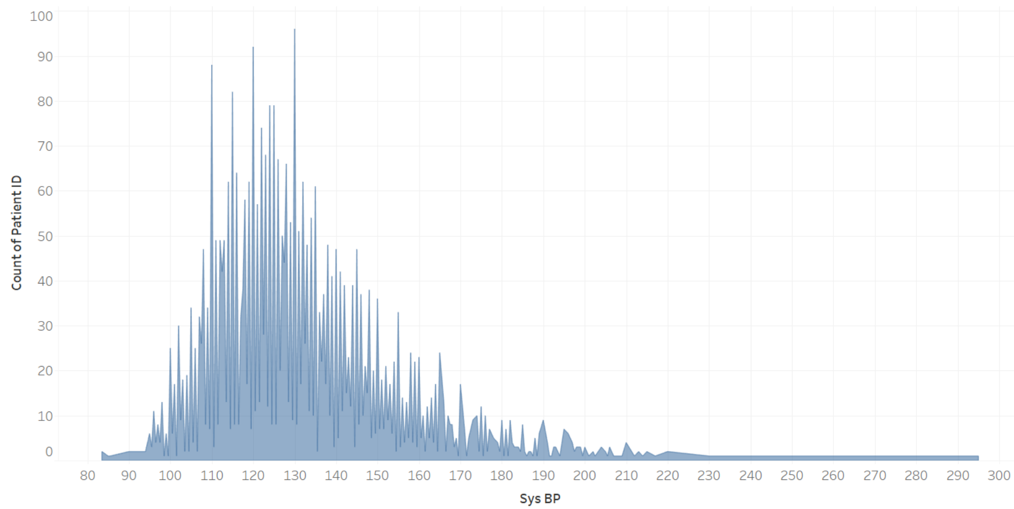


Both cholesterol and BMI distributions are positively skewed, with cholesterol showing a stronger skew, suggesting more extreme high values relative to the mean, while BMI's skew is moderate, implying a less pronounced tail but still a notable number of high outliers.

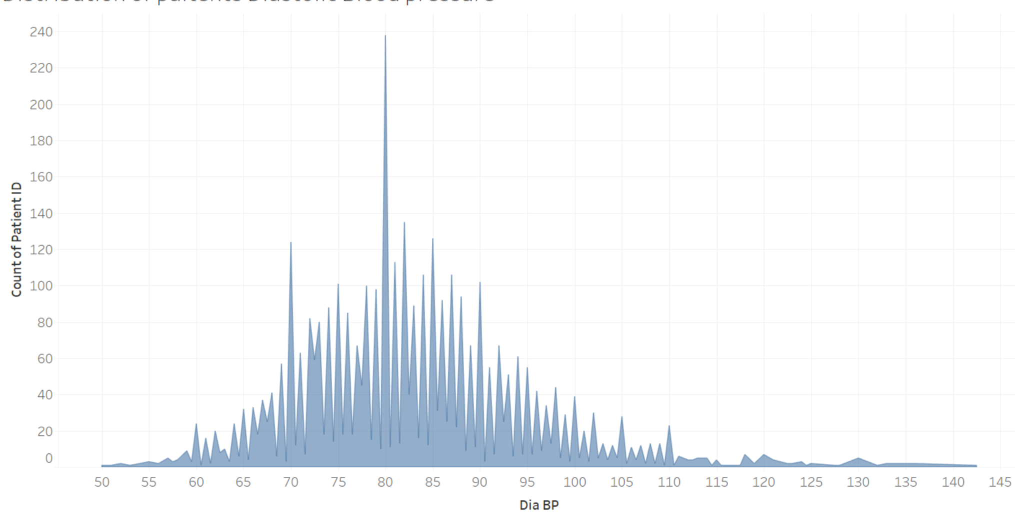
EDA REPORT – UNIVARIATE ANALYSIS

HEALTH MEASUREMENTS

Distribution of paitents Systolic Blood pressure



Distribution of paitents Diastolic Blood pressure

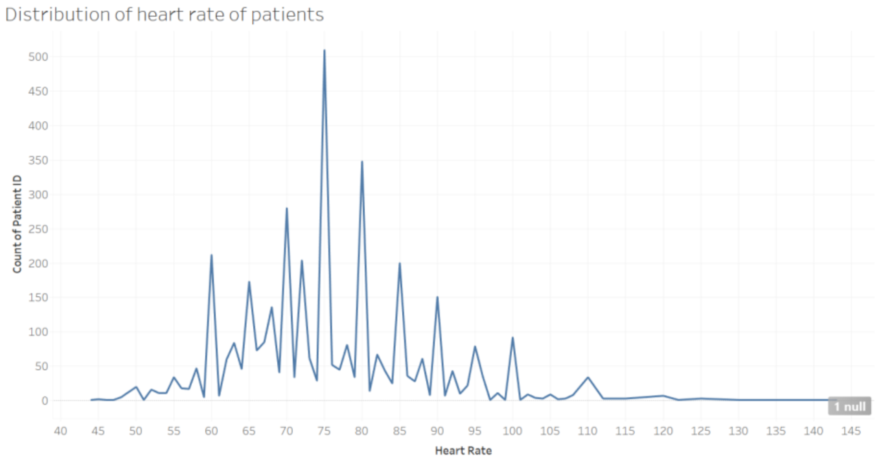


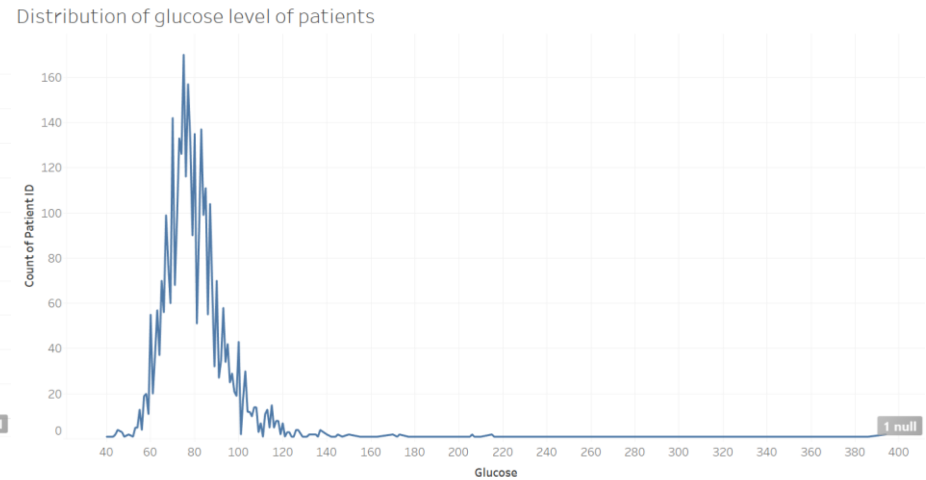
The distributions of systolic and diastolic blood pressure both exhibit right-skewed patterns, indicating that while most patients have blood pressure within a normal range, a significant minority exhibit abnormally high values.

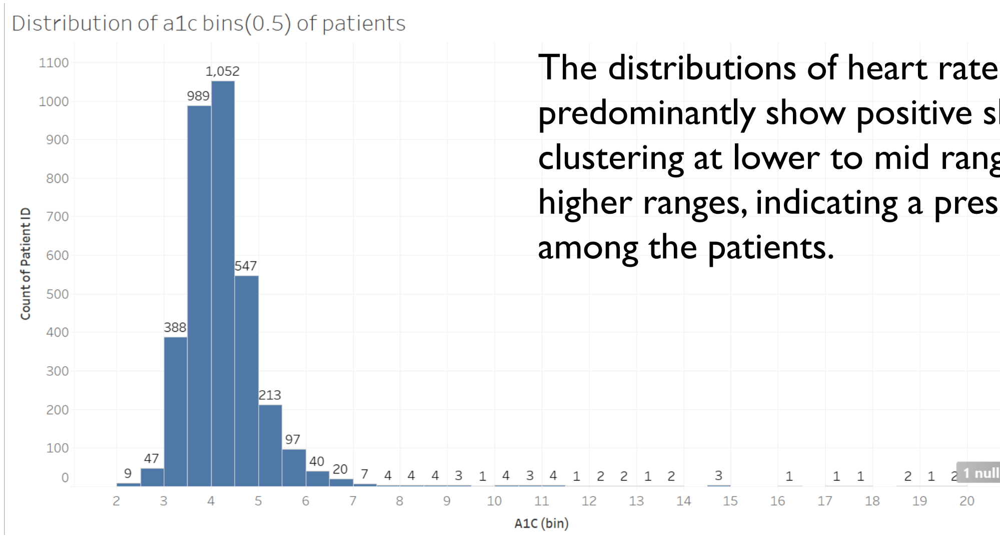
EDA REPORT – UNIVARIATE ANALYSIS

HEALTH MEASUREMENTS

USC Viterbi School of Engineering





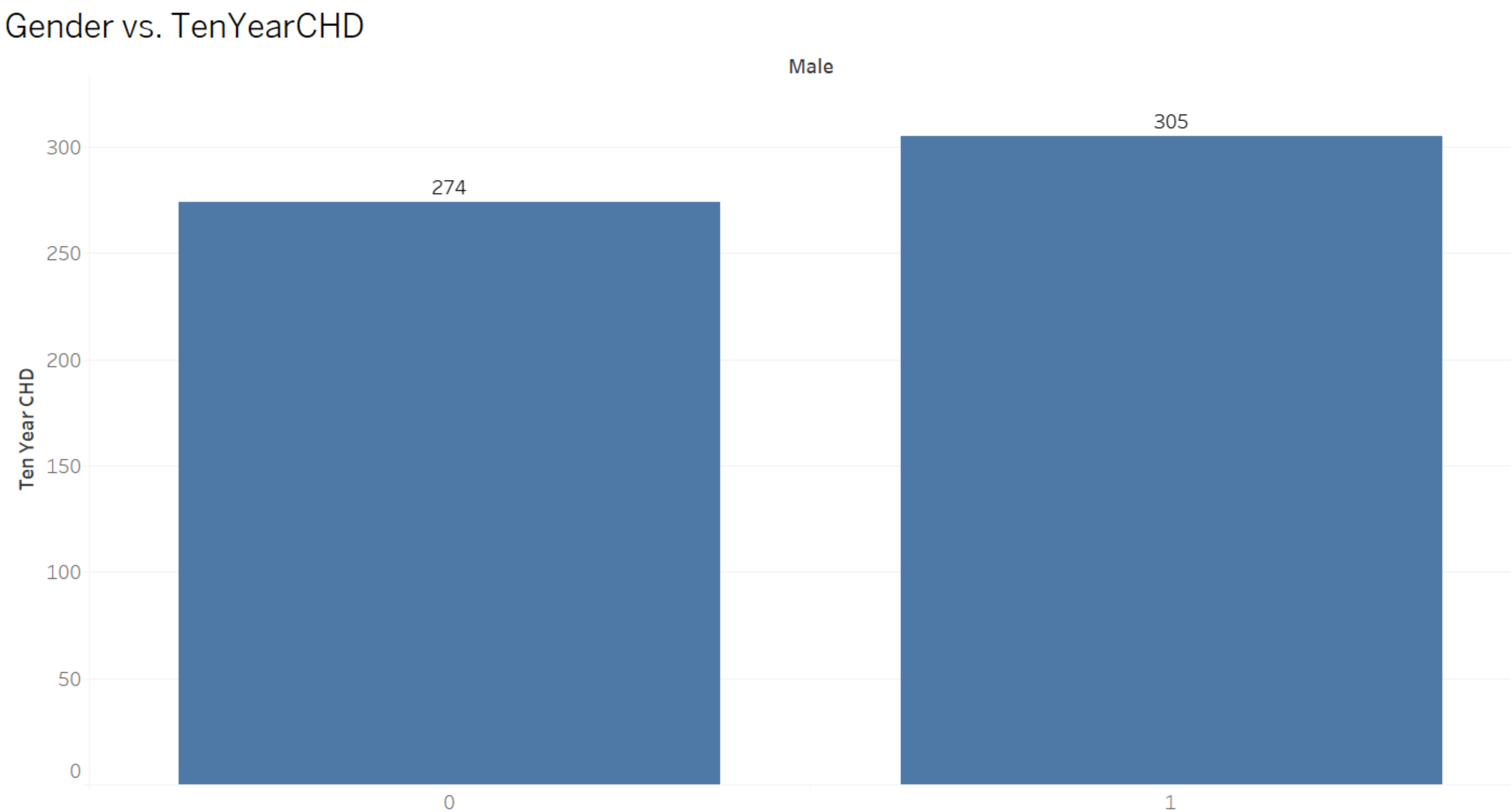


The distributions of heart rate, glucose, and A1C scores predominantly show positive skewness, with most values clustering at lower to mid ranges but extending towards higher ranges, indicating a presence of outlier conditions among the patients.

All the distributions here are positive skews

BIVARIATE ANALYSIS
PATIENT DEMOGRAPHICS VS RESPONSE

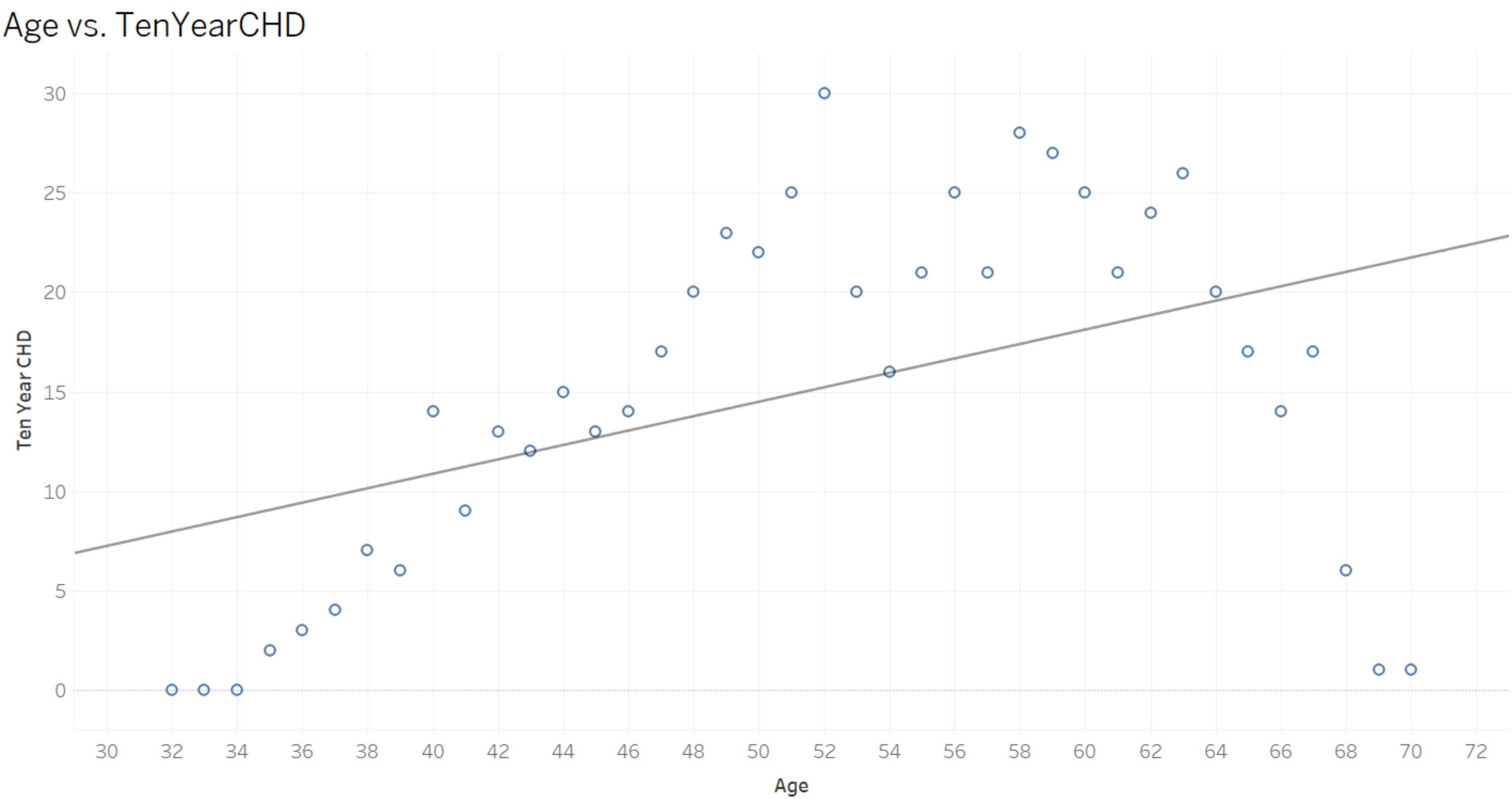
As per the given data, we can see that males (1) have higher chance of getting CHD than Female.



BIVARIATE ANALYSIS

PATIENT DEMOGRAPHICS VS RESPONSE

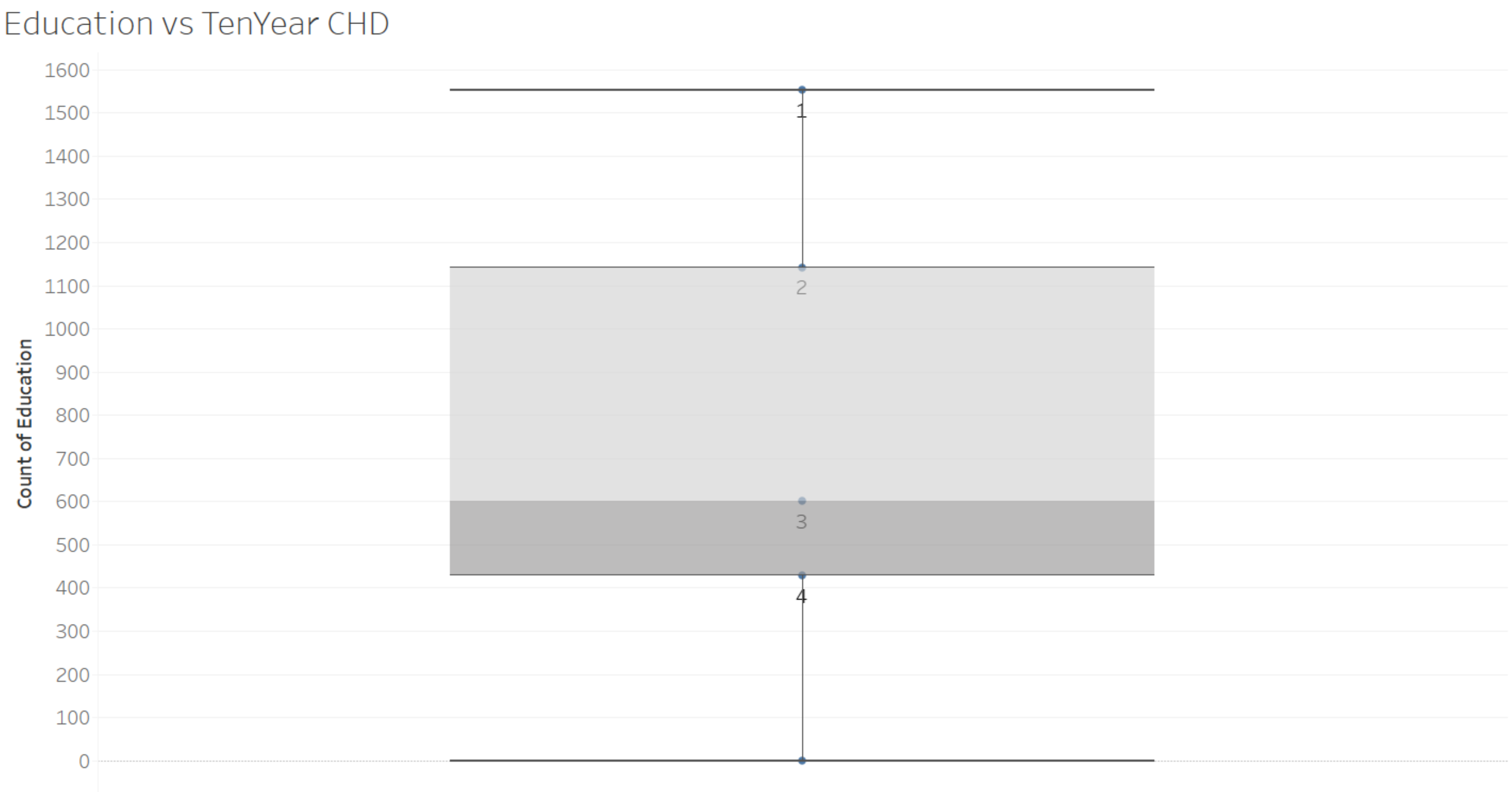
The positive slope of the trend line suggests a positive correlation between age and the risk of developing CHD. In other words, as age increases, the likelihood of developing CHD also increases.



BIVARIATE ANALYSIS

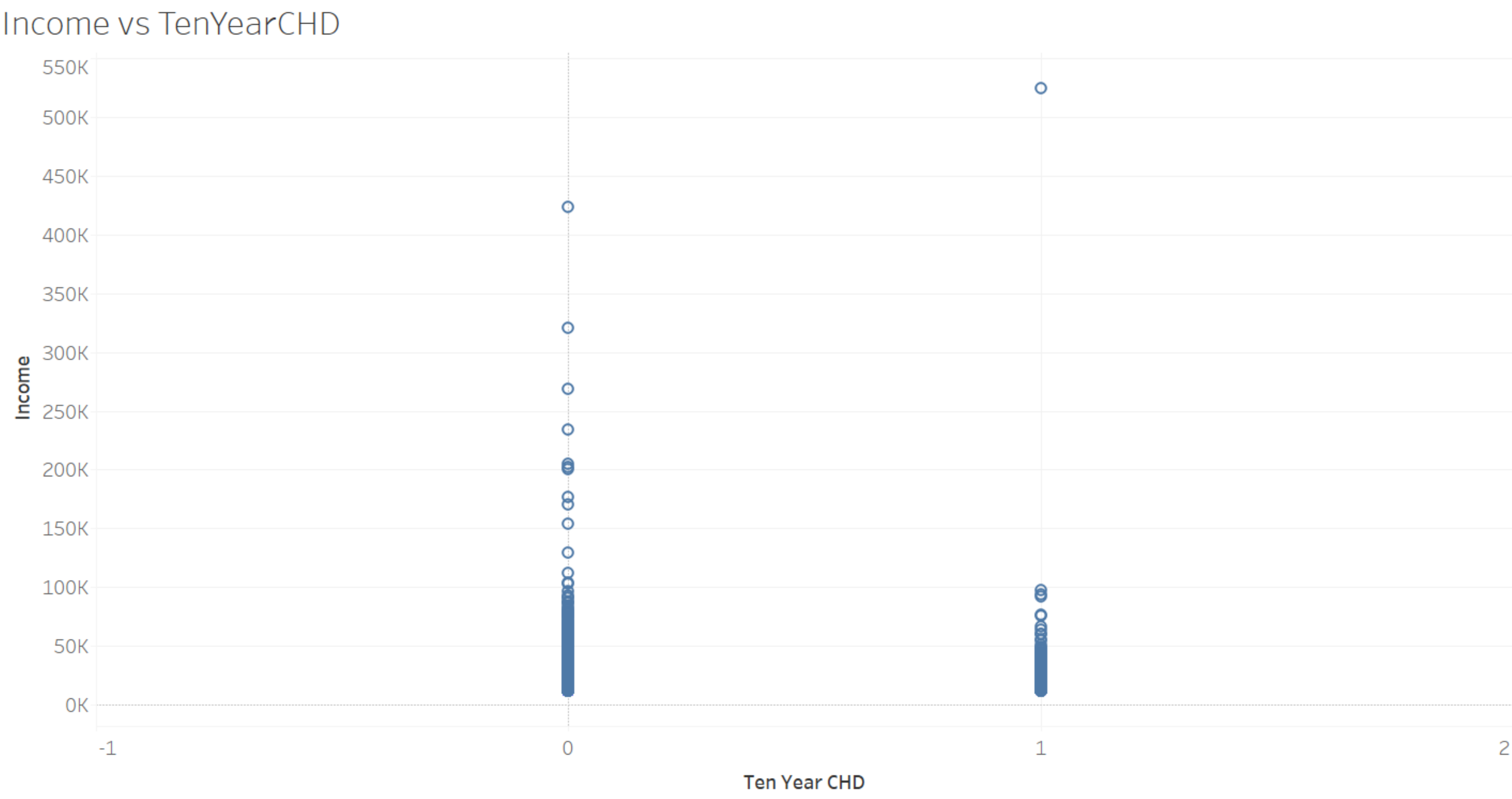
PATIENT DEMOGRAPHICS VS RESPONSE

Higher education levels (3 and 4) are associated with a lower risk of CHD, contrasting with <u>higher risk observed in individuals with lower education levels (1 and 2).</u>



BIVARIATE ANALYSIS
PATIENT DEMOGRAPHICS VS RESPONSE

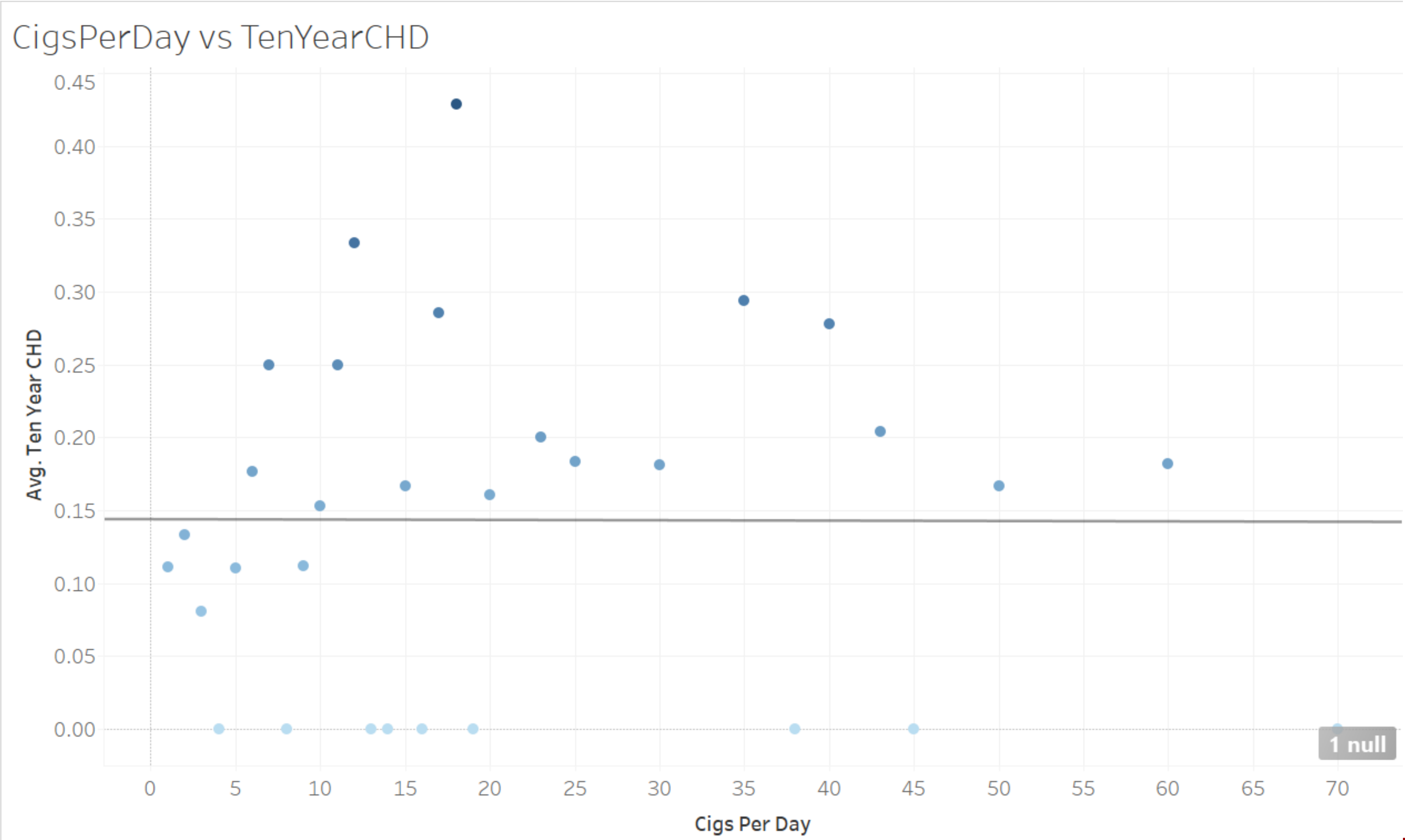
Individuals with TenYearCHD have a higher likelihood of having an income between 10k-100k, while those without CHD tend to have incomes ranging from 10k to 425k.



BIVARIATE ANALYSIS

PATIENT DEMOGRAPHICS VS RESPONSE

The scatter plot with trend line suggests that while higher numbers of cigarettes per day may indicate a higher average risk of CHD over ten years, this relationship is not fixed, indicating that CHD risk is likely influenced by factors other than just smoking habits.

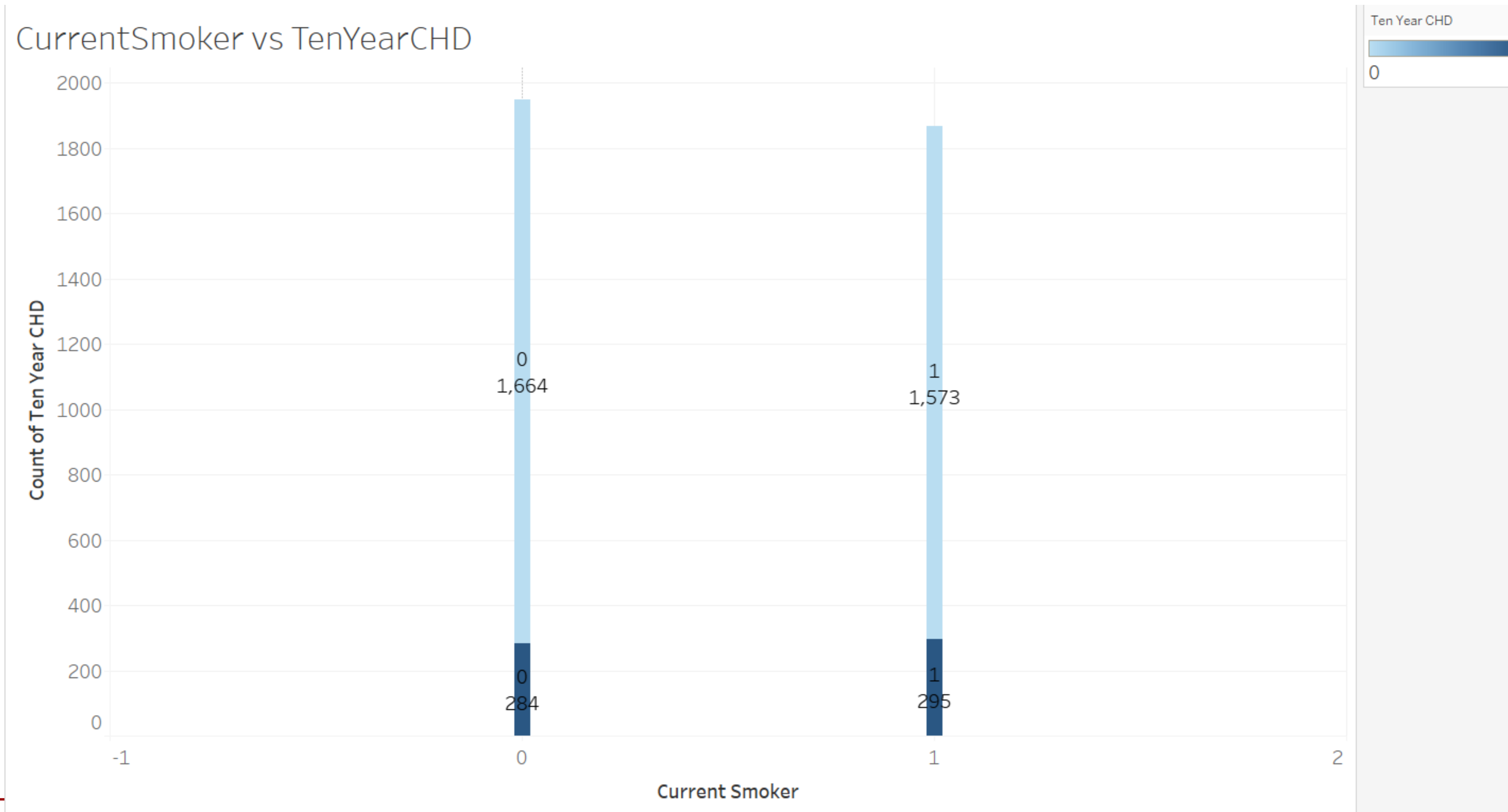


BIVARIATE ANALYSIS

PATIENT DEMOGRAPHICS VS RESPONSE

- Non-smokers (currentSmoker=0) have a higher count of individuals who did not develop CHD (1665) compared to those who developed CHD (284)
- Smokers (currentSmoker=1) also have a higher count of individuals who did not develop CHD (1573) compared to those who developed CHD (265)

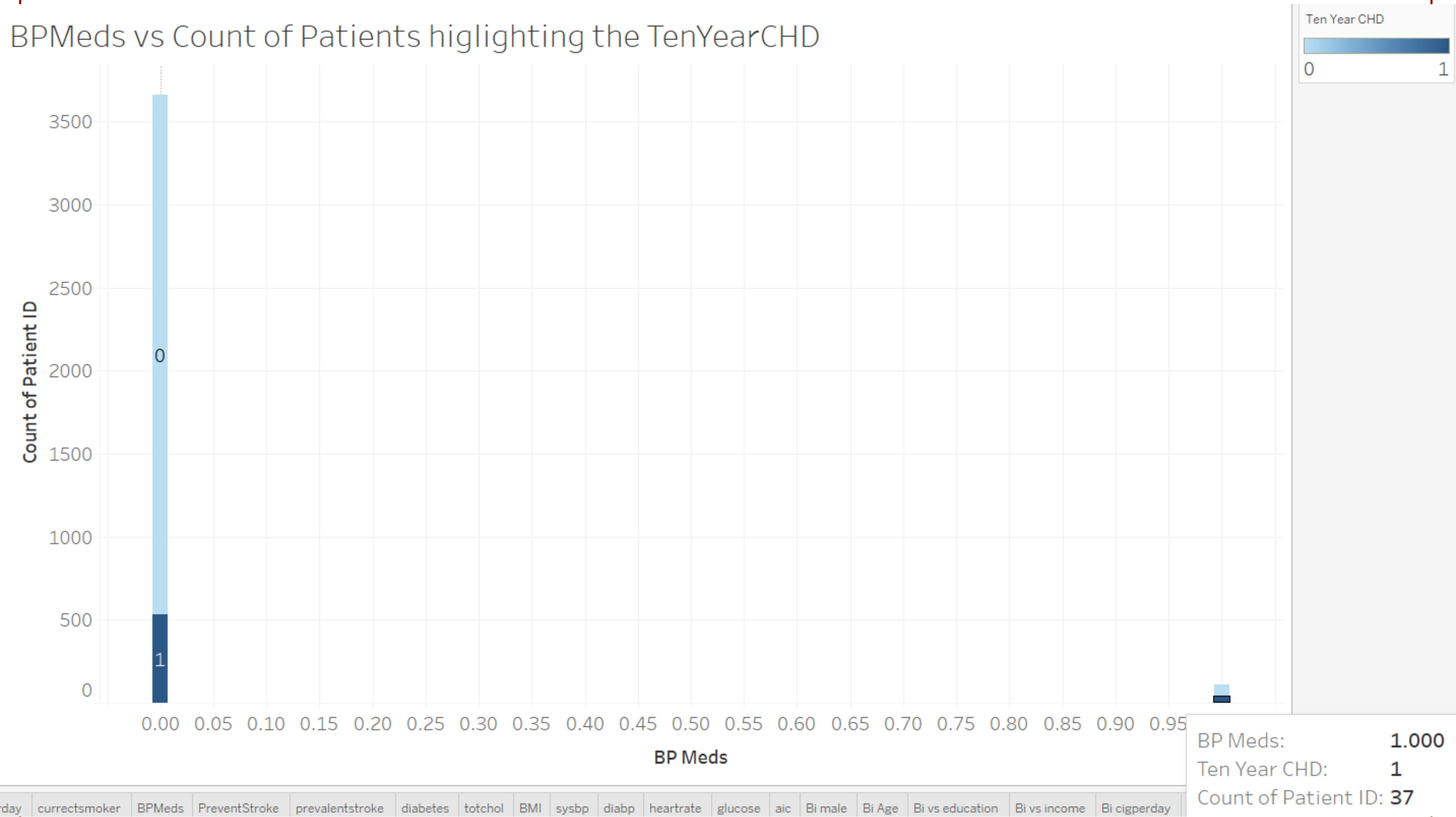
Both smokers and non-smokers have a higher count of individuals who did not develop CHD, suggesting that smoking alone may not be the sole factor influencing the development of CHD.



BIVARIATE ANALYSIS

PATIENT DEMOGRAPHICS VS RESPONSE

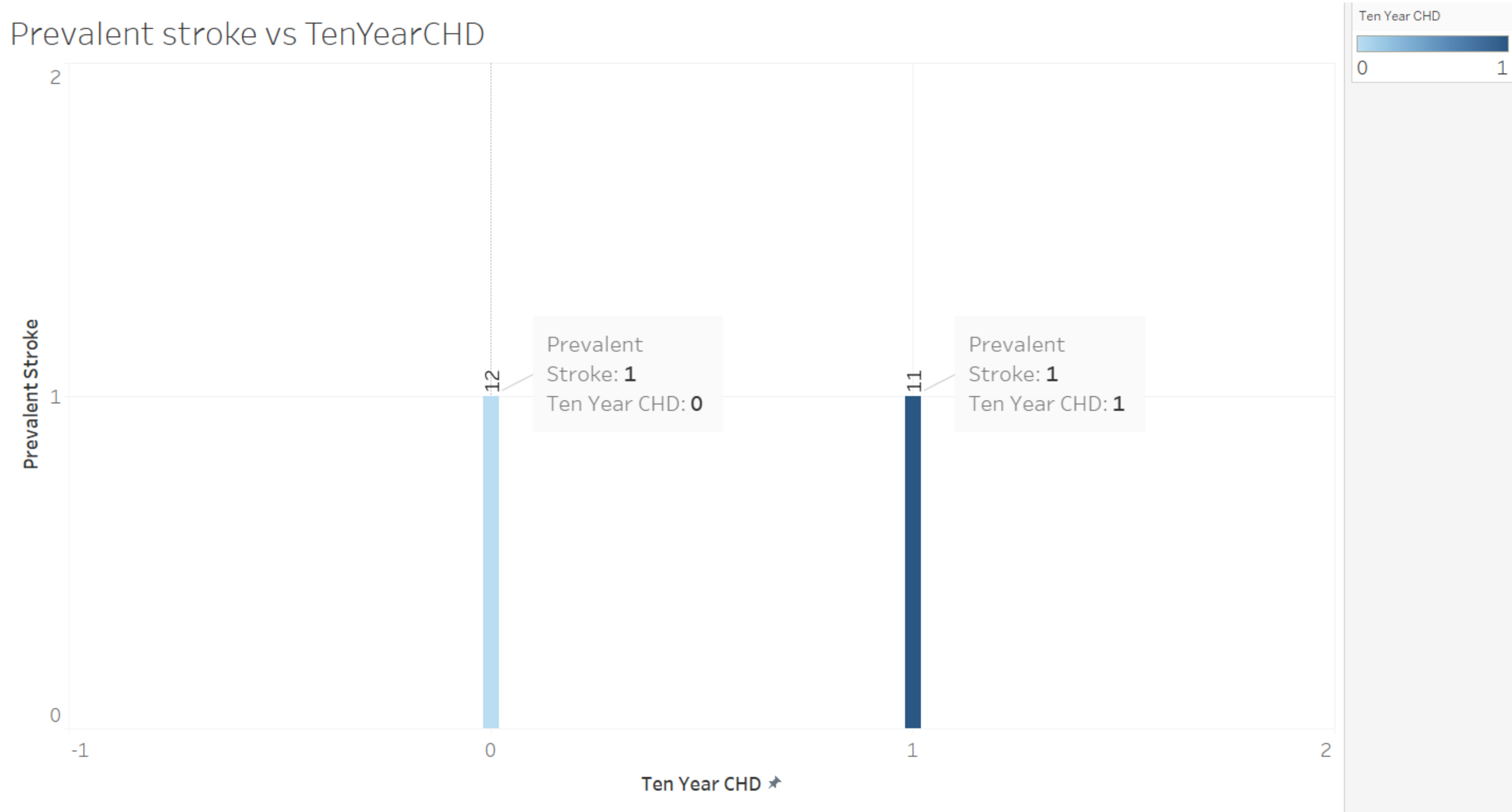
The graph shows that a higher proportion of patients not on blood pressure medication (BPMeds: 0) have a ten-year risk of developing coronary heart disease (TenYearCHD: 1) compared to those on medication, indicating a potential association between lack of BP medication and increased CHD risk.



BIVARIATE ANALYSIS

PATIENT DEMOGRAPHICS VS RESPONSE

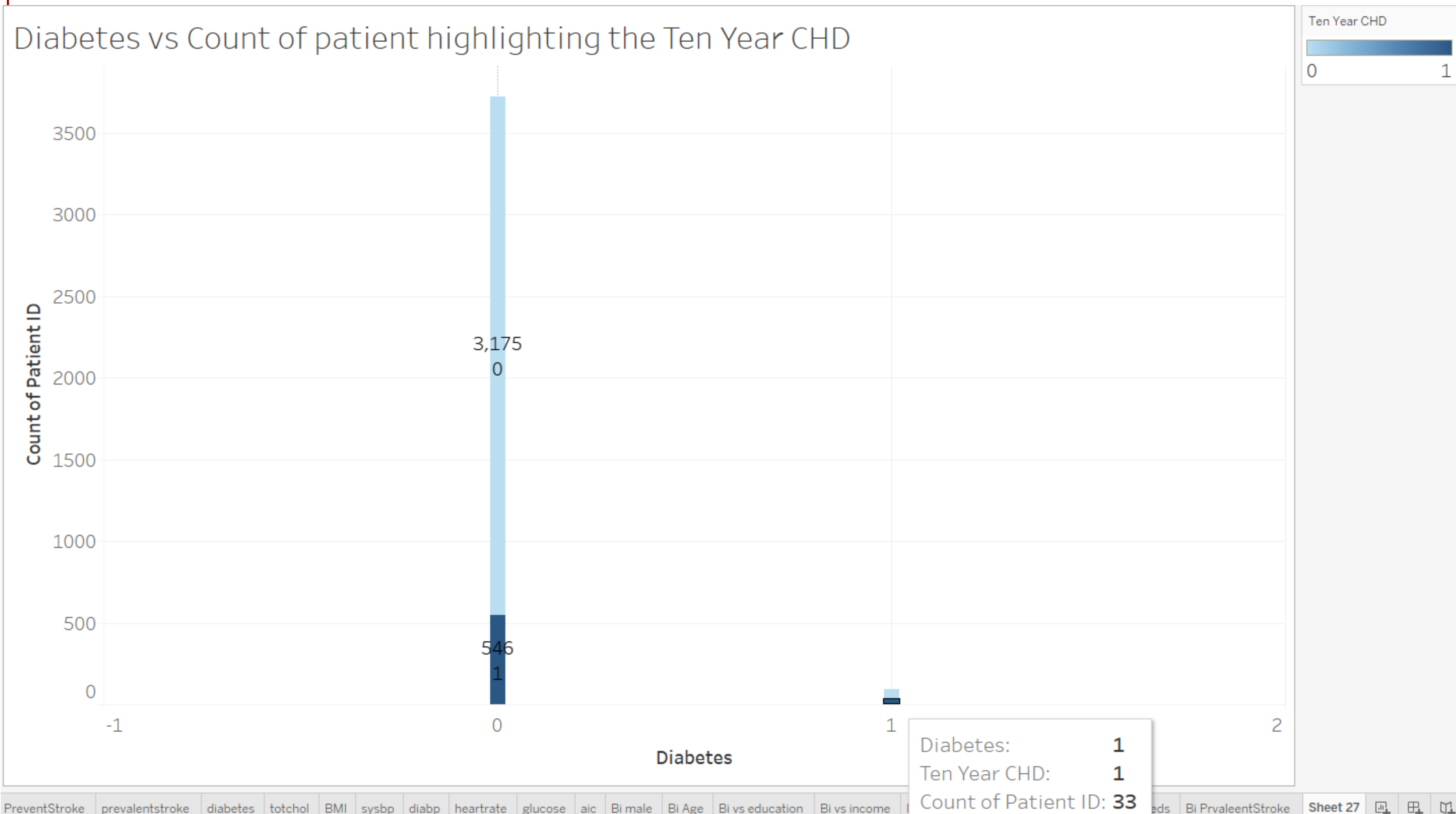
The results is that there were 12 patients with prevalent stroke (prevalentStroke=1) and no TenYearCHD (TenYearCHD=0), and 11 patients with prevalent stroke and TenYearCHD (TenYearCHD=1).



BIVARIATE ANALYSIS

PATIENT DEMOGRAPHICS VS RESPONSE

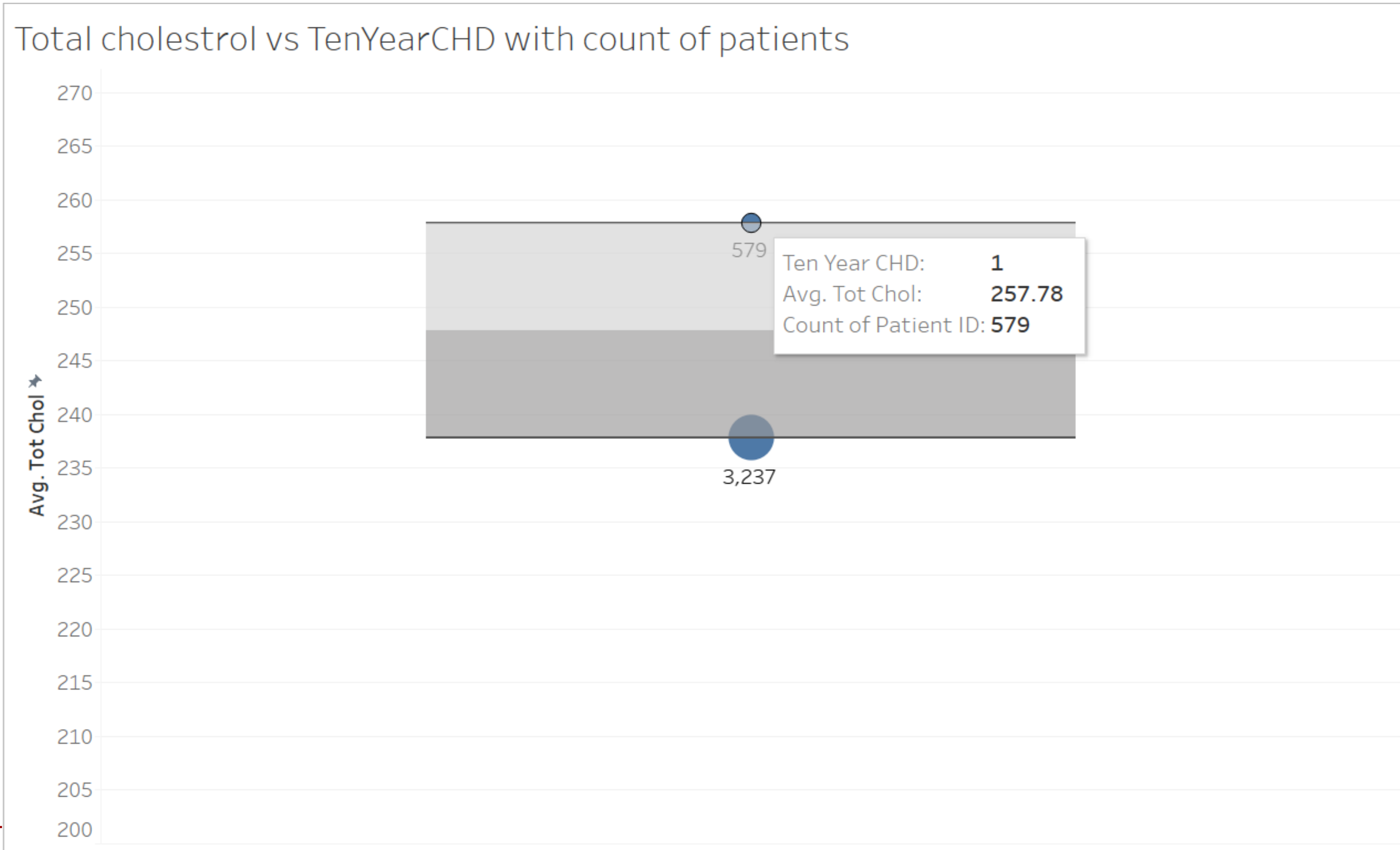
The graph shows that among patients without diabetes (diabetes = 0), 546 patients had a TenYearCHD of 1, and 175 patients had a TenYearCHD of 0. Among patients with diabetes (diabetes = 1), 62 patients had a TenYearCHD of 0, and only 3 patients had a TenYearCHD of 1.



BIVARIATE ANALYSIS

PATIENT DEMOGRAPHICS VS RESPONSE

Higher average total cholesterol levels (257.78) are observed in patients with a ten-year CHD risk (579 patients) compared to those without CHD (237.84 average and 3237 patients), suggesting a potential association between higher cholesterol levels and CHD risk.



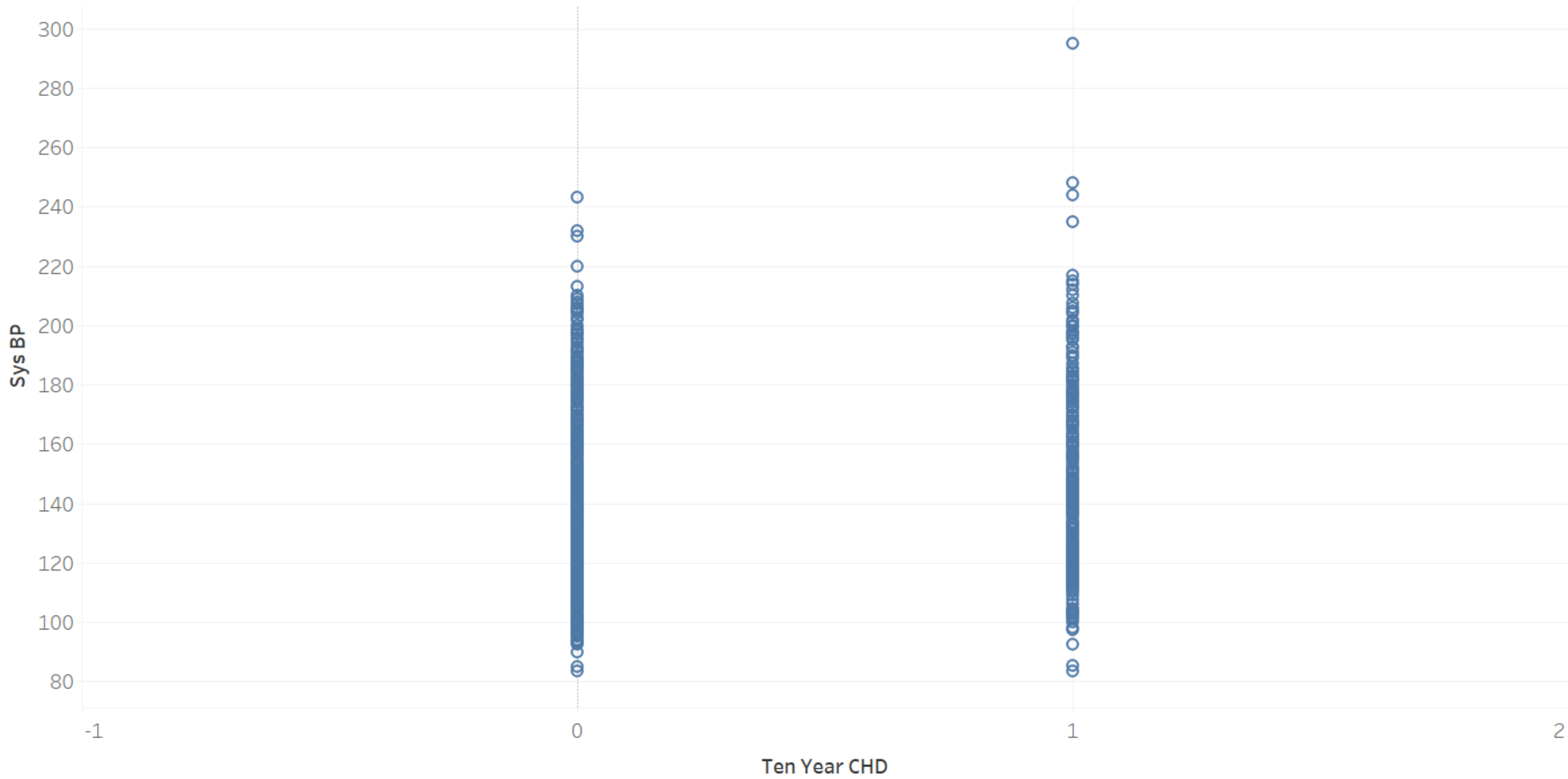
USC Viterbi School of Engineering

BIVARIATE ANALYSIS

PATIENT DEMOGRAPHICS VS RESPONSE

A fairly even distribution of systolic blood pressure across both CHD positive and negative cases suggests that <u>systolic blood pressure alone may not be a strong predictor for CHD risk</u> in this dataset.

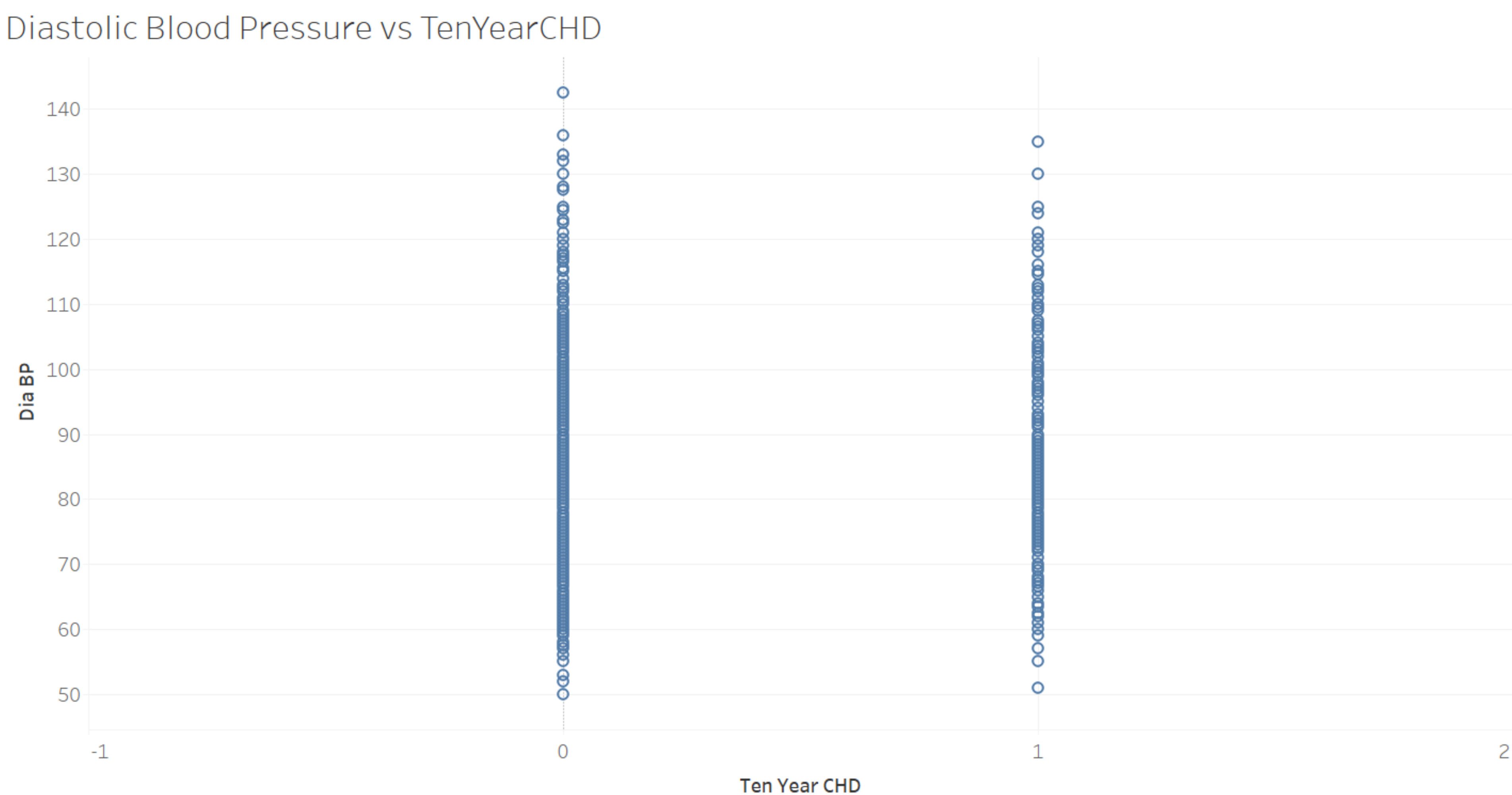
Systolic Blood Pressure vs TenYearCHD



BIVARIATE ANALYSIS

PATIENT DEMOGRAPHICS VS RESPONSE

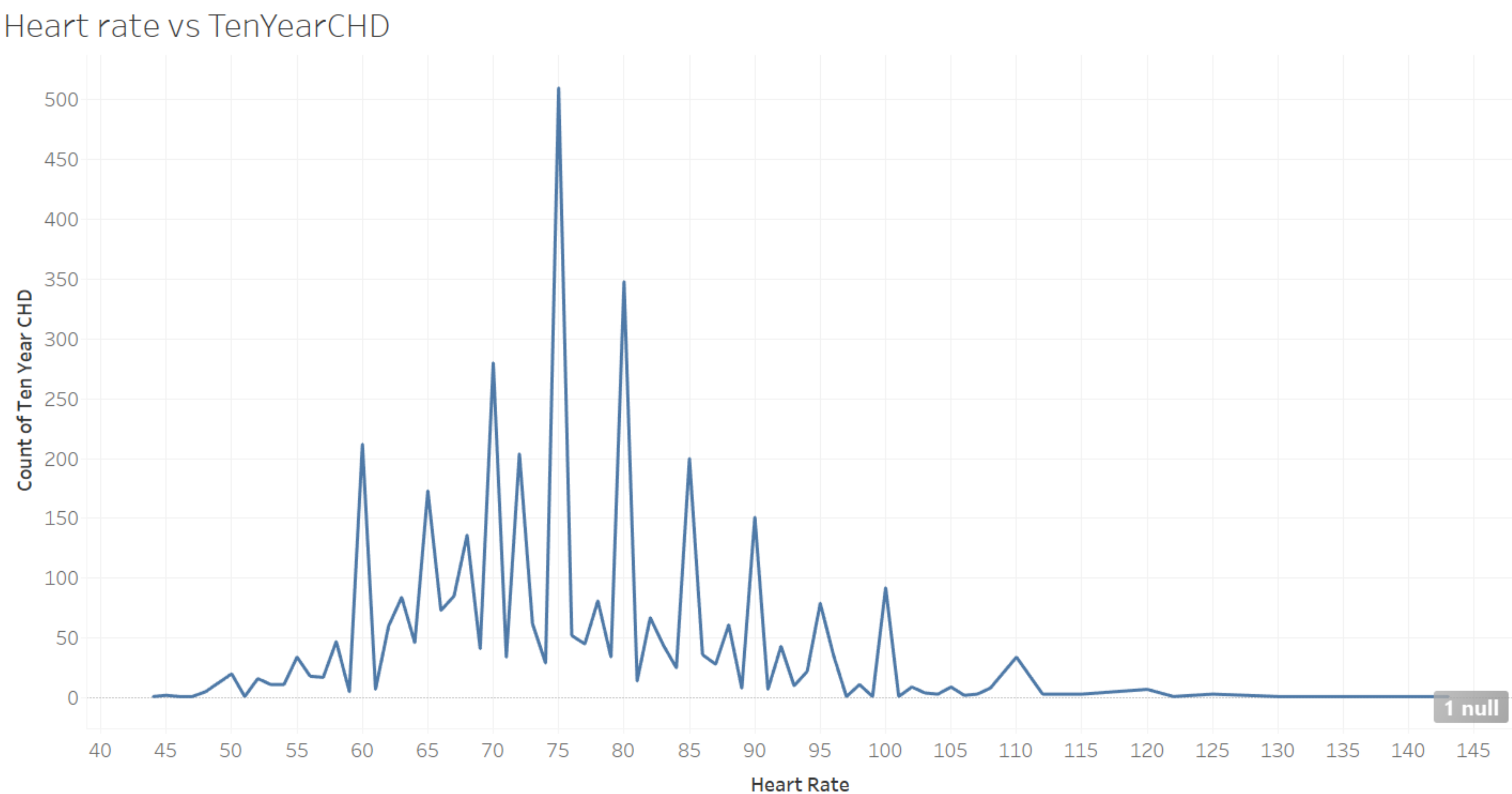
A almost even distribution of diastolic blood pressure across both CHD positive and negative cases suggests that diastolic blood pressure alone may also not be a strong predictor for CHD risk in this dataset.



BIVARIATE ANALYSIS

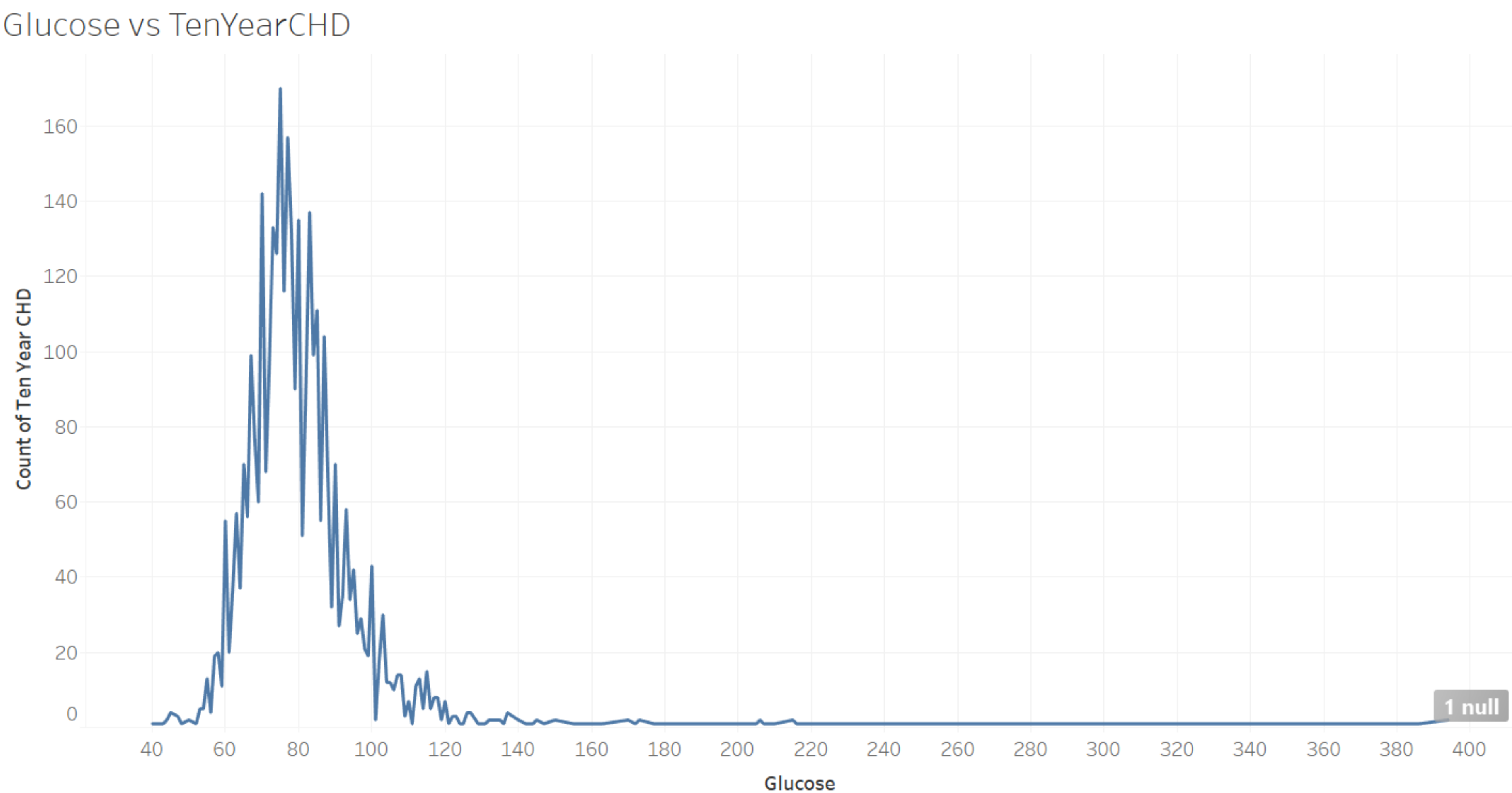
PATIENT DEMOGRAPHICS VS RESPONSE

Based on the peak around 75 to 80, the evaluation of the line graph suggests that there may be a higher frequency of TenYearCHD cases in patients with heart rates in this range. However, this <u>does not necessarily imply that heart rate is a dependable variable for predicting TenYearCHD</u>.



BIVARIATE ANALYSIS
PATIENT DEMOGRAPHICS VS RESPONSE

The line graph indicates a potential link between heart rates in the 70-83 range and higher counts of TenYearCHD cases, but it does not establish heart rate as a reliable predictor of TenYearCHD.

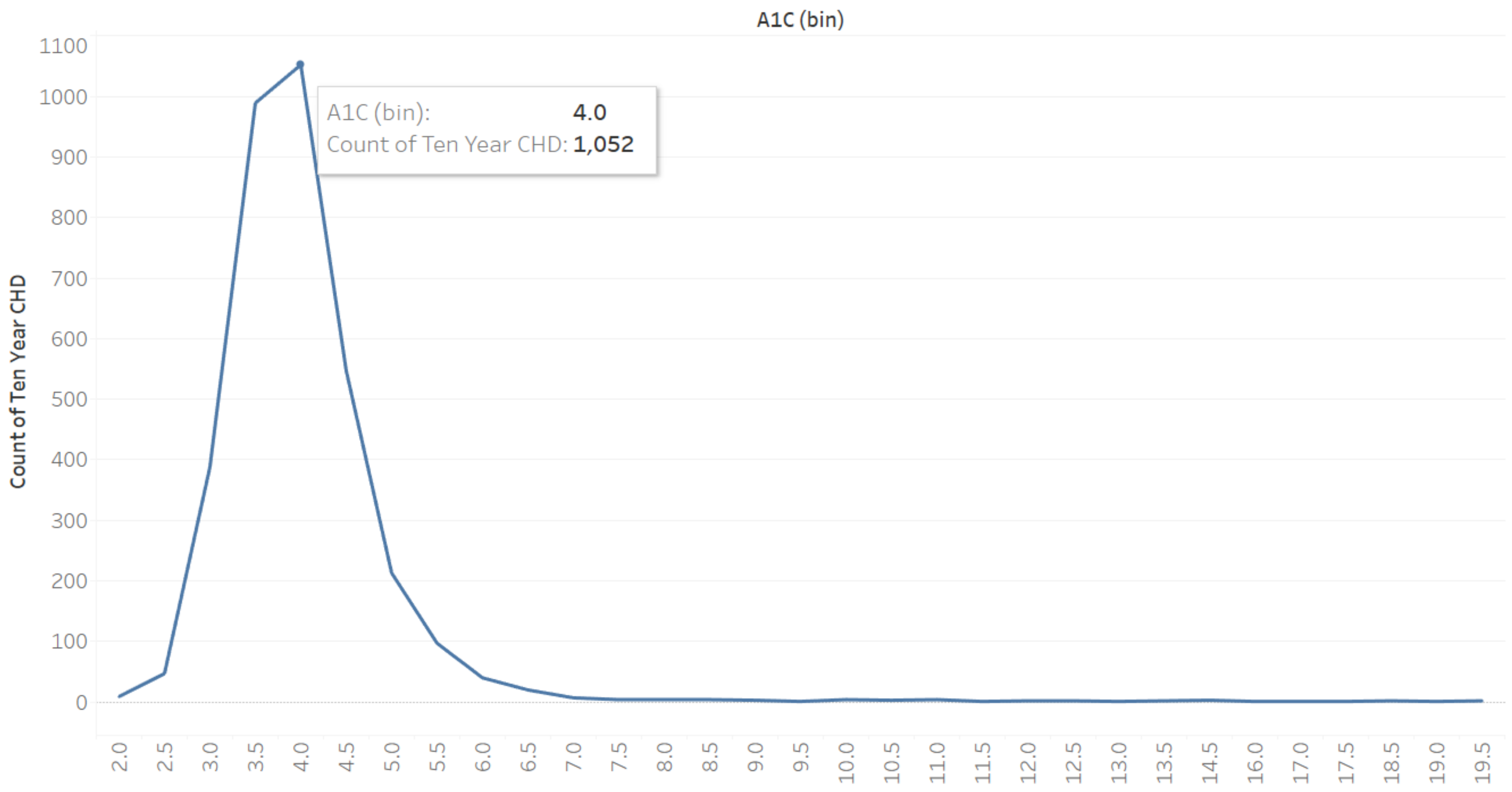


BIVARIATE ANALYSIS

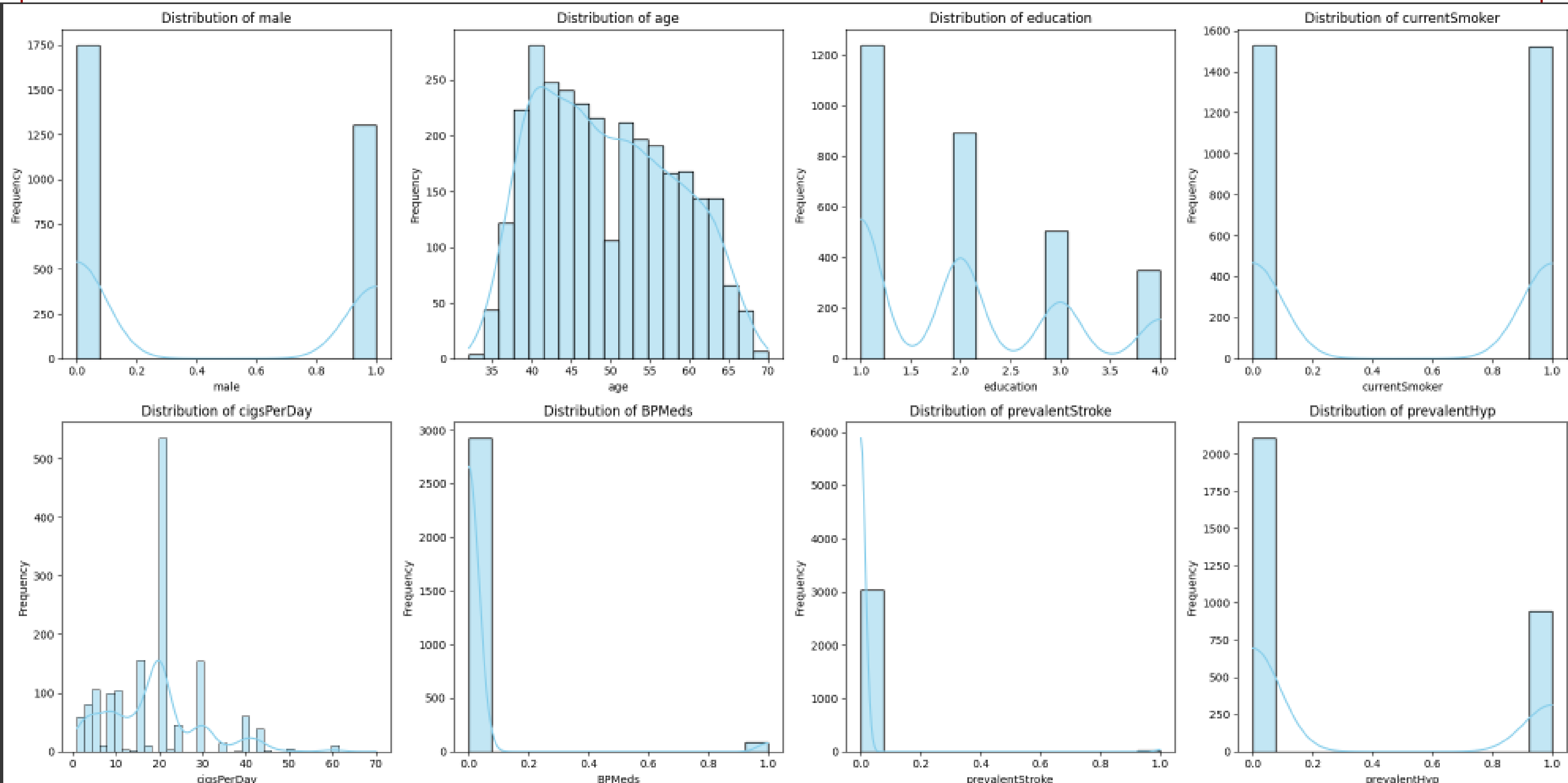
PATIENT DEMOGRAPHICS VS RESPONSE

Based on the peak around A1c level 4.0 with the highest count of 1052, it suggests that there is a significant number of patients with this A1c level. However, this alone <u>does not indicate the dependency of A1c on TenYearCHD</u>. Further analysis, such as statistical tests or deeper exploration, would be needed to determine if A1c is a dependable predictor of TenYearCHD.

A1c vs TenYearCHD

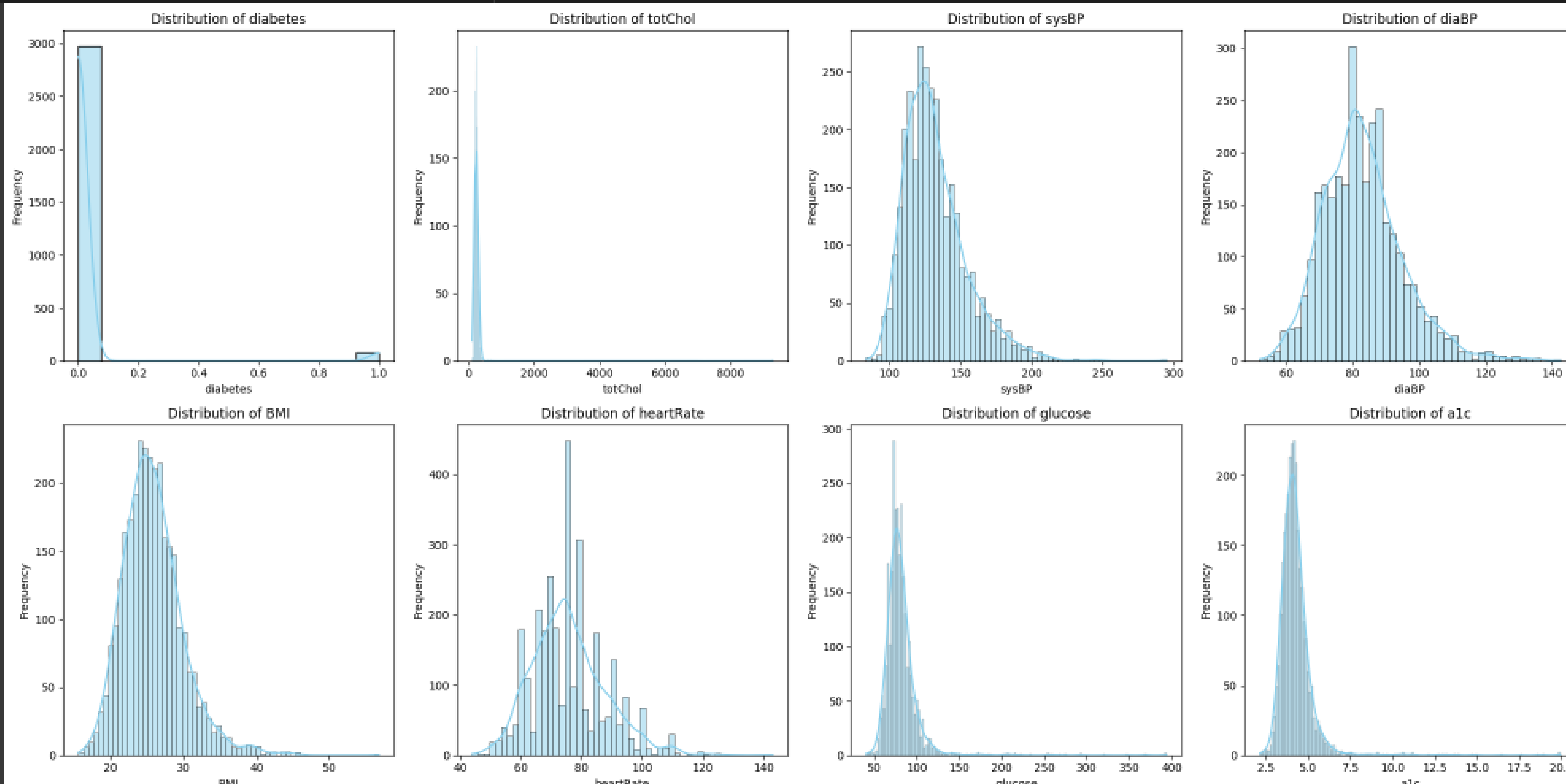


UNIVARIATE



Distribution of male
Distribution of age
Distribution of education
Distribution of currentSmoker
Distribution of cigsPerDay
Distribution of BPMeds
Distribution of prevalentStroke
Distribution of prevalentHyp
Frequency
male
age
education
currentSmoker
cigsPerDay
BPMeds
prevalentStroke
prevalentHyp

UNIVARIATE


Distribution of diabetes
Frequency
diabetes
Distribution of totChol
Frequency
totChol
Distribution of sysBP
Frequency
sysBP
Distribution of diaBP
Frequency
diaBP
Distribution of BMI
Frequency
BMI
Distribution of heartRate
Frequency
heartRate
Distribution of glucose
Frequency
glucose
Distribution of a1c
Frequency
a1c

BIVARIATE ANALYSIS

PATIENT DEMOGRAPHICS VS RESPONSE

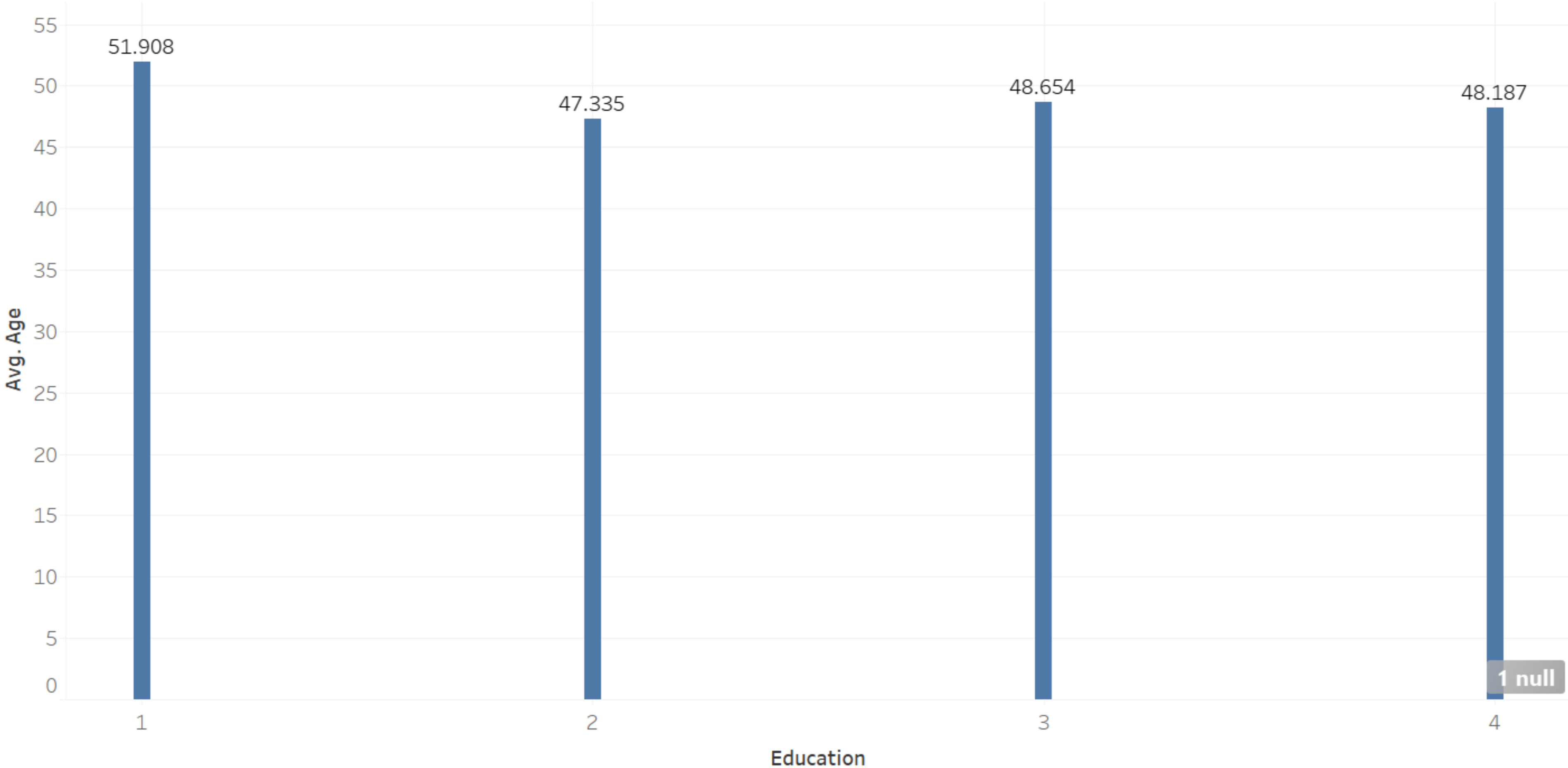
After analyzing the variables in relation to the response variable, I found that age, education, income, BP meds, and total cholesterol are related to the response variable. To further analyze this dataset, we can explore the following bivariate relationships:

1. Age vs. Education: Explore how education level varies with age.
2. Age vs. Income: Investigate the relationship between age and income.
3. Age vs. BP Meds: See if there's a correlation between age and the use of blood pressure medications.
4. Age vs. TotChol: Examine how age relates to total cholesterol levels.

These comparisons could provide insights into how these variables are distributed and potentially related within your dataset.

BIVARIATE ANALYSIS
OTHER PAIRS OF VARIABLES

Age vs Education

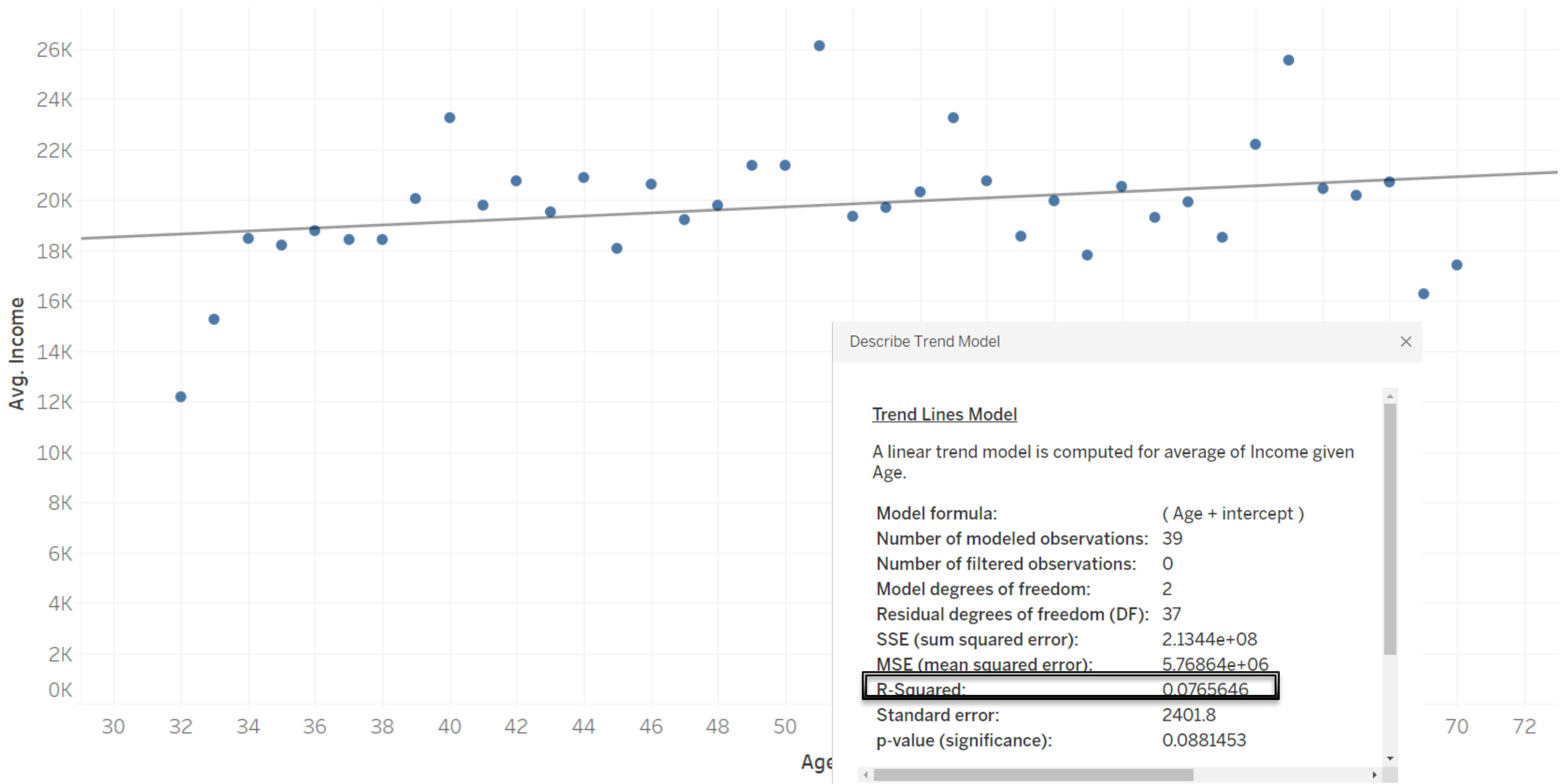


From the bar graph comparing age across different education levels, we can see that the average age for individuals with education level 1 is 51.90, for level 2 it is 47.33, for level 3 it is 48.65, and for level 4 it is 48.18. This suggests that, on average, individuals with higher education levels tend to be younger than those with lower education levels.

BIVARIATE ANALYSIS

OTHER PAIRS OF VARIABLES

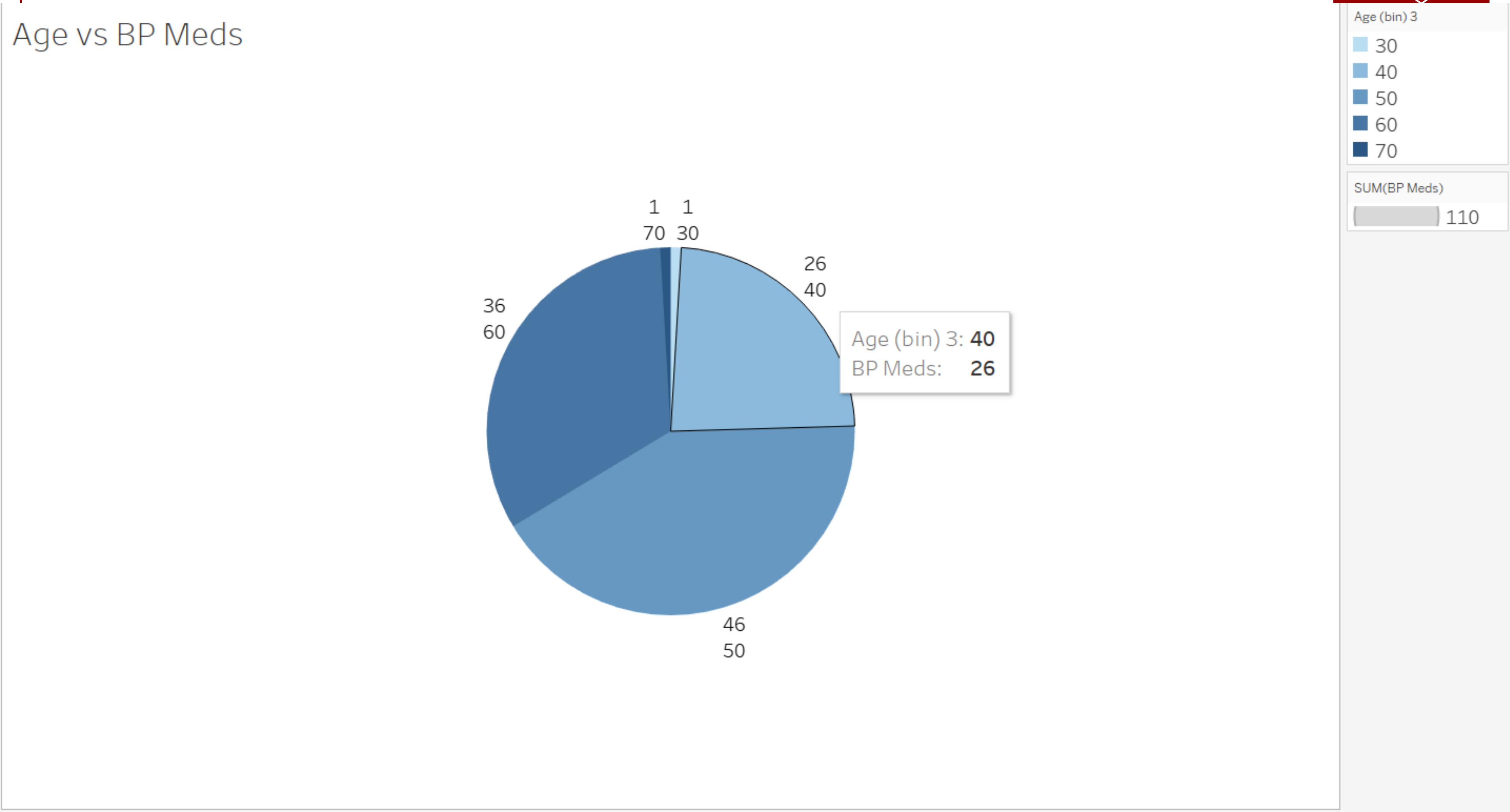
Age vs Income



The R-squared value of 0.07656 indicates that only about 7.7% of the variability in income can be explained by age in this dataset. The p-value of 0.088 is higher than the conventional significance level of 0.05, suggesting that <u>the relationship between age and income may not be statistically significant.</u>

BIVARIATE ANALYSIS

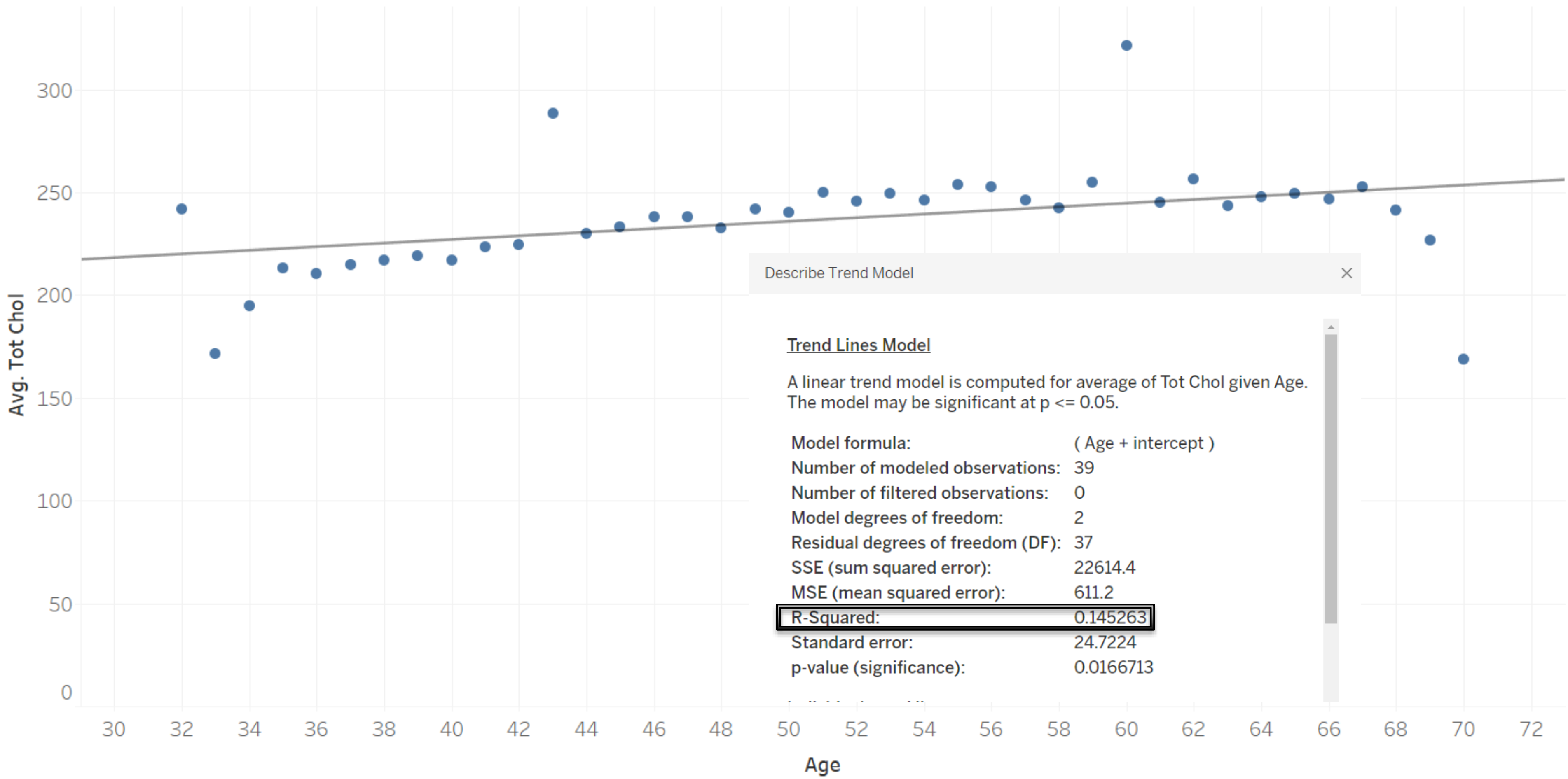
OTHER PAIRS OF VARIABLES



The distribution of BP Meds across age groups suggests that <u>BP Meds usage tends to be more common among individuals aged 50 and 60</u>, with a noticeable decline in usage among younger and older age groups.

BIVARIATE ANALYSIS
OTHER PAIRS OF VARIABLES

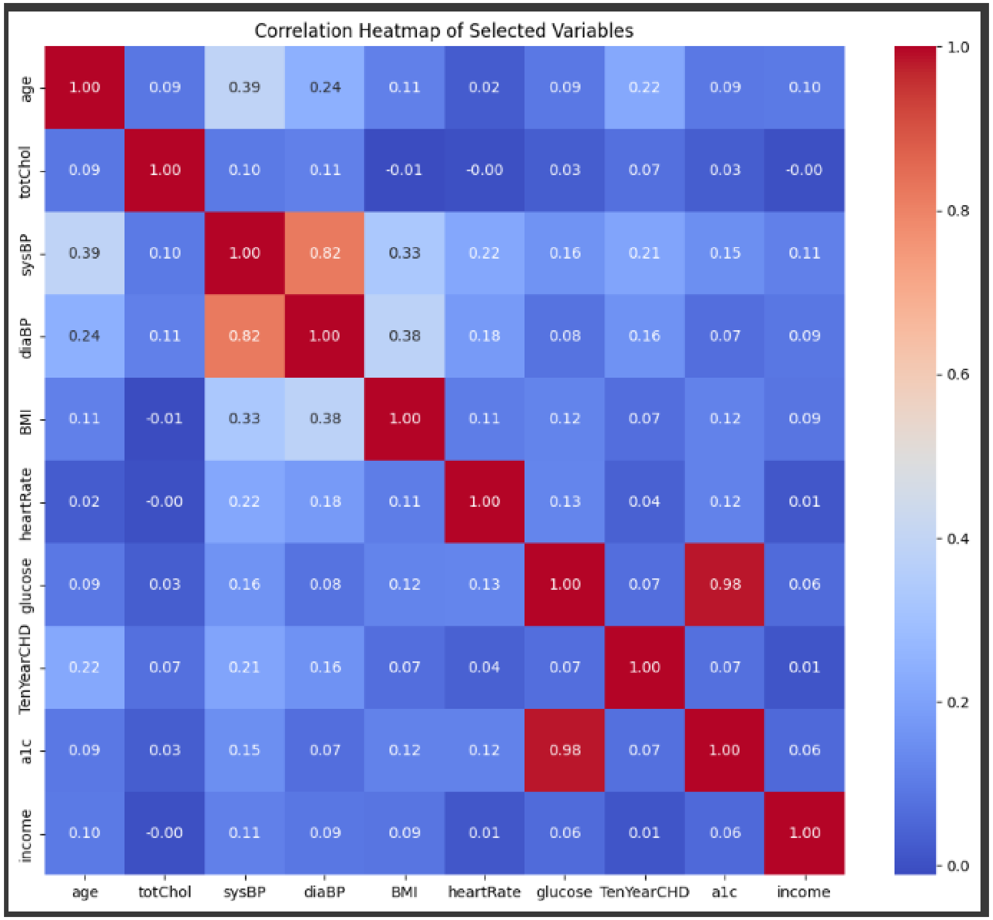
Age vs TotChol



The scatter plot shows a weak positive correlation between age and total cholesterol levels, indicated by the low R-square value of 0.1452. The p-value of 0.0166 suggests that this relationship is statistically significant, but the correlation is not very strong. This indicates that <u>as age increases, there is a slight tendency for total cholesterol levels to also increase</u>.

MULTIVARIATE ANALYSIS

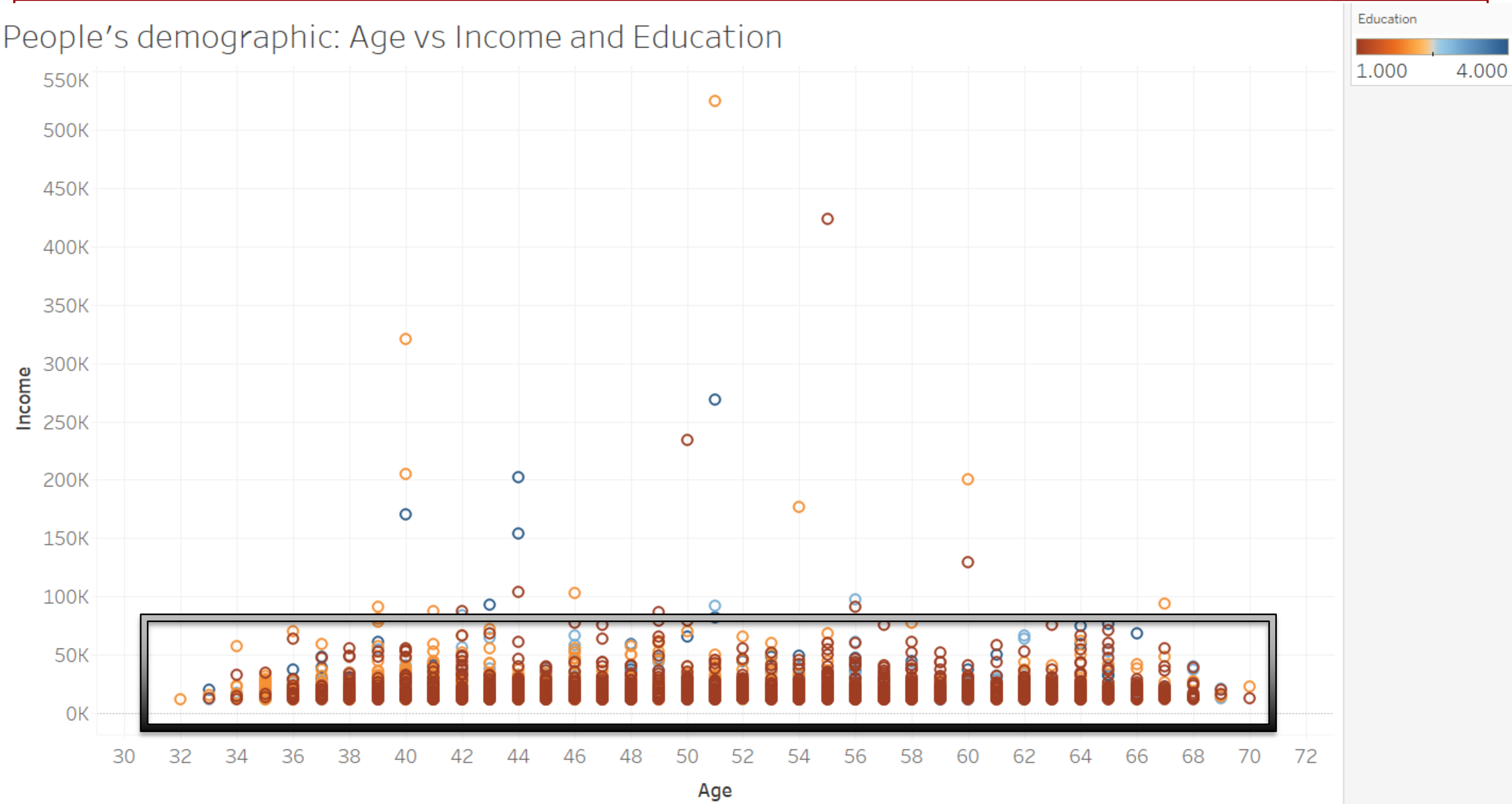
OTHER PAIRS OF VARIABLES



- Glucose-A1C levels have a high correlation and both systolic and diastolic blood pressure, indicating a significant relationship between these variables.
- Other variables may also exhibit correlations, but the strongest correlations seem to involve Glucose-A1C and blood pressure measures.

MULTIVARIATE ANALYSIS

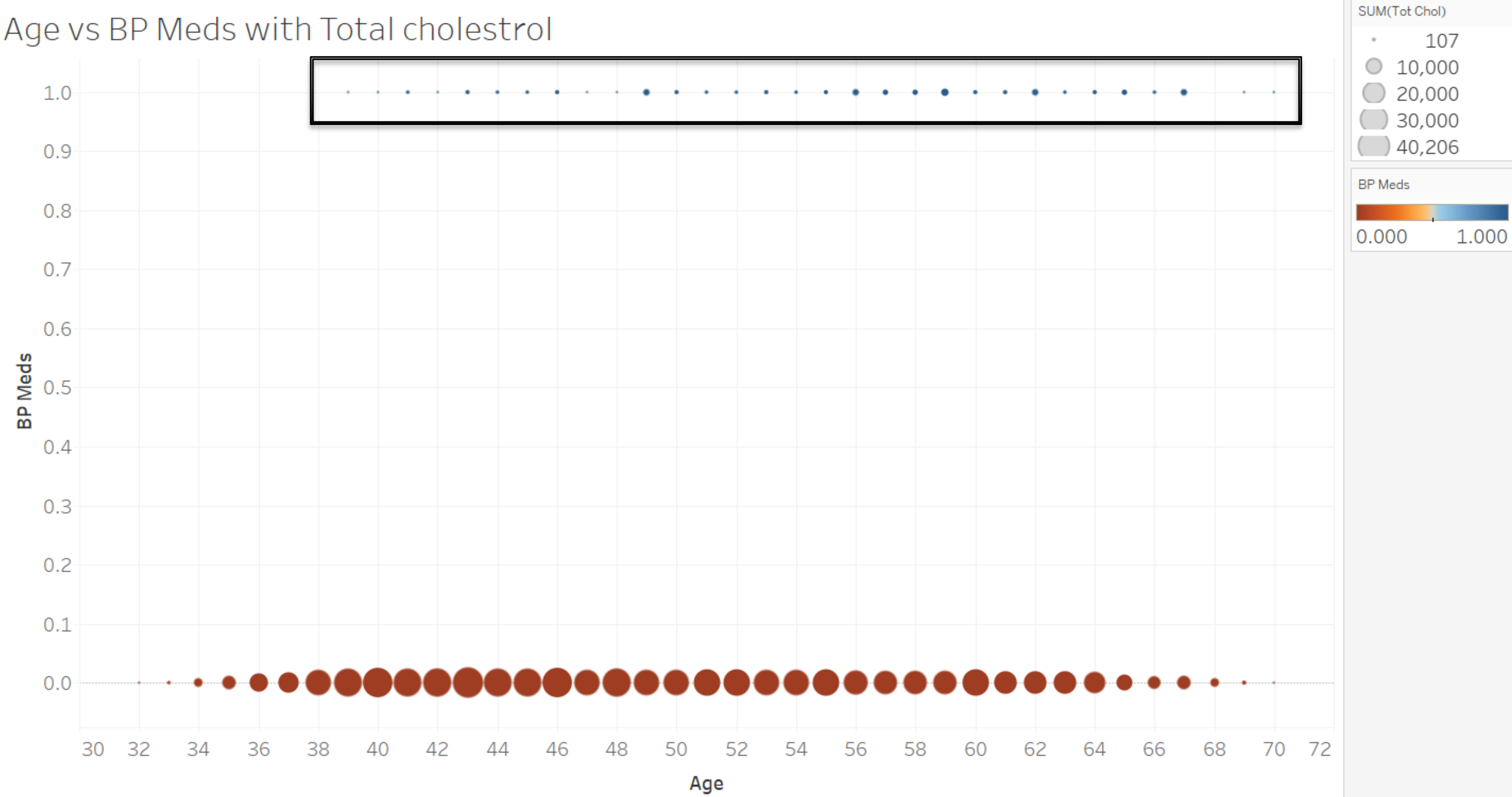
OTHER PAIRS OF VARIABLES



The scatter plot matrix reveals that the dataset's audience is diverse in age, with a range from 32 to 70 years old. Income is also varied, but most individuals fall within the 10,000 to 50,000 income range. Education levels are primarily at level 1, likely indicating some high school education. This demographic profile suggests a diverse audience with a significant portion in the lower to moderate income brackets and varying levels of education.

MULTIVARIATE ANALYSIS

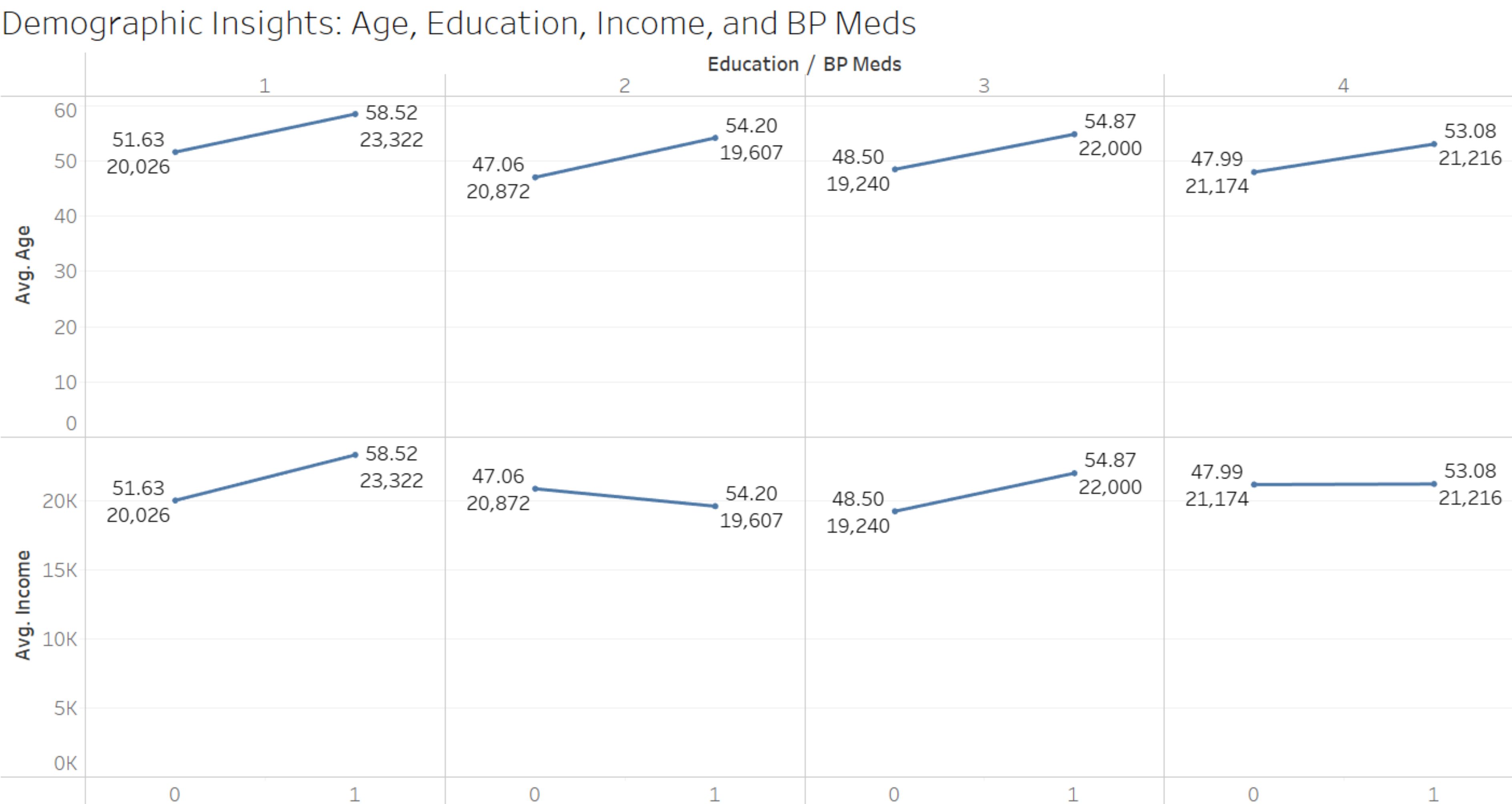
OTHER PAIRS OF VARIABLES



In the bubble chart, individuals aged 32-70 taking BP meds generally have lower total cholesterol (230-2800) compared to those not taking BP meds, where higher cholesterol levels (up to 40000) are observed. Total cholesterol does not seem to depend solely on BP meds usage in this dataset.

MULTIVARIATE ANALYSIS

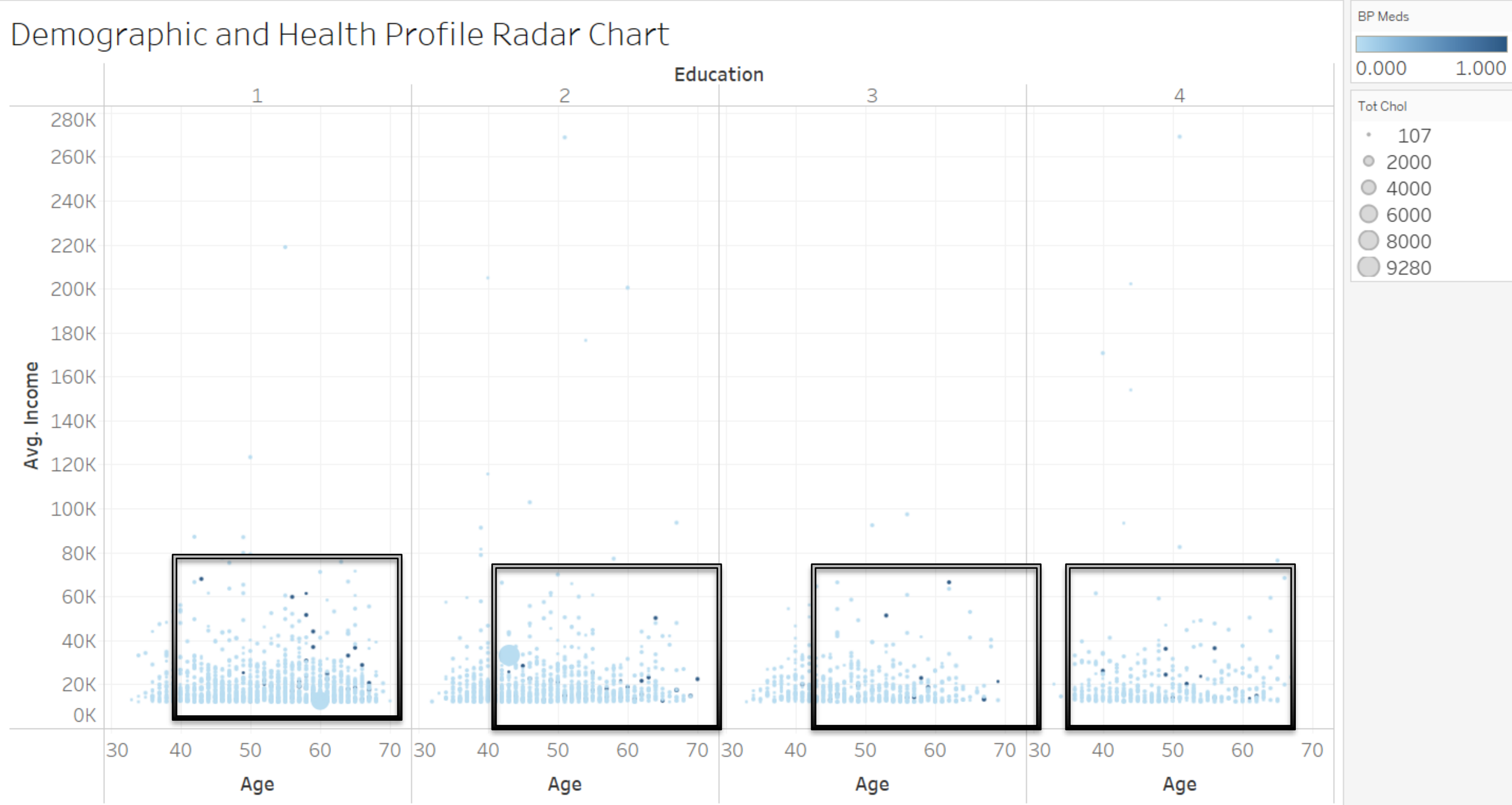
OTHER PAIRS OF VARIABLES



We can see that while there's a general trend of higher education correlating with higher income, there are exceptions, particularly among individuals on blood pressure medications. This suggests that factors beyond education and income, such as health status, may influence medication use.

MULTIVARIATE ANALYSIS

OTHER PAIRS OF VARIABLES



The radar chart provides a concise overview of the demographic and health profile of the sample, highlighting potential relationships such as higher education correlating with higher income, and age possibly influencing BP medication usage.

DATA PREPARATION PLAN OVERVIEW

- Binning Age
  - age (Variable to categorize into age groups)
- Categorizing Cigarette Consumption
  - cigsPerDay (Variable to categorize based on cigarette consumption)
- Log Transformation
  - income (Variable for which logarithmic transformation is applied to normalize data)
  - sysBP (Systolic blood pressure variable for log transformation)
  - diaBP (Diastolic blood pressure variable for log transformation)
- One-Hot Encoding
  - Categorical variables converted to binary

DATA PREPARATION PLAN

DATA QUALITY ISSUES AND ACTIONS

Creating artifacts for below varibales during imputation

- Data Cleaning Patient ID: Drop 'patientID' column.
- Imputation cigsPerDay: Missing values filled with median.
- Imputation BPMeds: Missing values filled with median.
- Imputation Education: Missing values filled with median.
- Imputation totChol: Missing values filled with mean.
- Imputation BMI: Missing values filled with mean.
- Imputation Glucose: Missing values filled with mean.
- Imputation AIC: Missing values filled with mean.
- Imputation HeartRate: Missing values filled with mean.

DATA PREPARATION PLAN

FEATURE ENGINEERING DECISIONS

- Binning Age: Grouping into 'Young', 'Middle-aged', 'Senior'.
- Classifying Cigarettes: Categorizing 'cigsPerDay' into smoker types.
- Log Transformations: Stabilizing 'income', 'sysBP'.
- One-Hot Encoding: Converting categorical variables into binary vectors.
- Removing Features: Dropping unnecessary variables.
- Standardizing Data: Normalizing 'BMI', 'Glucose', 'totChol'.
- Creating Interactions: Generating terms from combinations like age and cholesterol.

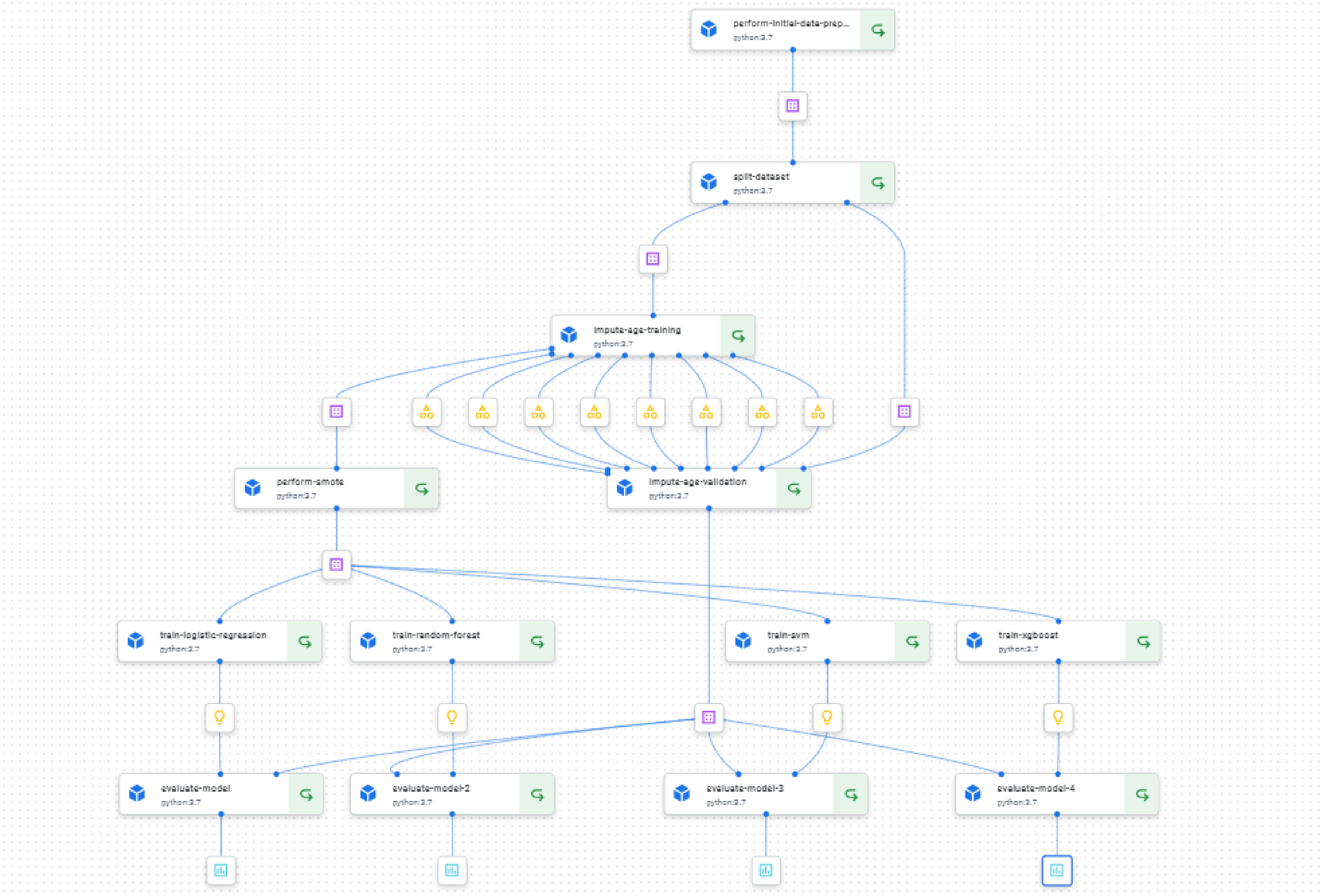
DATA PREPARATION PLAN

DATASET PARTITIONING DECISIONS

- Perform SMOTE oversampling due to unbalanced dataset
  - Load Data: Read dataset from input_df_path.
  - Feature-Target Split: Separate 'TenYearCHD' as target.
  - Apply SMOTE: Balance class distribution.
  - DataFrame Conversion: Reform features and target into DataFrames.

MODEL PIPELINE

OVERALL PIPELINE



MODEL PIPELINE

MODEL EVALUATION RESULTS

Name	metrics
Type	system.Metrics
URI	gs://architafinalproject/812826359571/chd-prediction-pipeline-20240501014111/evaluate-model-4_5082690411148755968/metrics

Metrics

Scalar metrics produced by this step.

accuracy	0.8403141361256544
f1_score	0.7940784171378298
false_negatives	109
false_positives	13
model_type	XGBoost
true_negatives	629
true_positives	13

MODEL PIPELINE

PIPELINE DEFINITION CODE

```
[32] from kfp.v2.dsl import pipeline, Output, Dataset, component, Model

@pipeline(name='chd-prediction-pipeline')
def chd_prediction_pipeline(training_dataset_path: str):

    # Process training dataset - initial data preparation
    data_preparation = perform_initial_data_preparation(
        input_dataset_path=training_dataset_path)
    split_result = split_dataset(input_dataset_path=data_preparation.outputs['output_dataset_path'])

    # Process training dataset - impute age and other features
    imputed_training_data = impute_age_training(
        training_dataset_path=split_result.outputs['train_data_path'])

    # Impute age and other features in "validation" dataset using the same means/medians from training data
    # Here, using the same training_dataset_path for validation due to lack of a separate test dataset
    imputed_validation_data = impute_age_validation(
        validation_dataset_path=split_result.outputs['validation_data_path'],
        cig_mean=imputed_training_data.outputs['cig_mean'],
        BP_mean=imputed_training_data.outputs['BP_mean'],
        EDU_mean=imputed_training_data.outputs['EDU_mean'],
        Chol_mean=imputed_training_data.outputs['Chol_mean'],
        BMI_mean=imputed_training_data.outputs['BMI_mean'],
        glucose_mean=imputed_training_data.outputs['glucose_mean'],
        a1c_mean=imputed_training_data.outputs['a1c_mean'],
        heartRate_mean=imputed_training_data.outputs['heartRate_mean'])

    # Perform SMOTE oversampling on the imputed training dataset
    oversampled_training_data = perform_SMOTE(
        input_df_path=imputed_training_data.outputs['imputed_dataset_path'])
```

MODEL PIPELINE

PIPELINE DEFINITION CODE

```
# Perform SMOTE oversampling on the imputed training dataset
oversampled_training_data = perform_SMOTE(
    input_df_path=imputed_training_data.outputs['imputed_dataset_path'])

# Train models using oversampled data
trained_lr_model = train_logistic_regression(
    training_dataset_path=oversampled_training_data.outputs['output_df_path'])
trained_rf_model = train_random_forest(
    training_dataset_path=oversampled_training_data.outputs['output_df_path'])
trained_svm_model = train_svm(
    training_dataset_path=oversampled_training_data.outputs['output_df_path'])
trained_xgb_model = train_xgboost(
    training_dataset_path=oversampled_training_data.outputs['output_df_path'])

# Evaluate all models using the imputed validation dataset
# Reusing the 'training_dataset_path' for validation purpose
evaluate_model(
    test_dataset_path=imputed_validation_data.outputs['imputed_dataset_path'],
    model=trained_lr_model.outputs['trained_model_artifact'],
    model_type="Logistic Regression")
evaluate_model(
    test_dataset_path=imputed_validation_data.outputs['imputed_dataset_path'],
    model=trained_rf_model.outputs['trained_model_artifact'],
    model_type="Random Forest")
evaluate_model(
    test_dataset_path=imputed_validation_data.outputs['imputed_dataset_path'],
    model=trained_svm_model.outputs['trained_model_artifact'],
    model_type="SVM")
evaluate_model(
    test_dataset_path=imputed_validation_data.outputs['imputed_dataset_path'],
    model=trained_xgb_model.outputs['trained_model_artifact'],
    model_type="XGBoost")
```

COMPONENT DEFINITION PERFORM_INITIAL_DATA_PREPARATION

```python
from kfp.v2.dsl import InputPath, OutputPath, Dataset

@component(packages_to_install=["pandas", "numpy", "fsspec", "gcsfs"])
def perform_initial_data_preparation(input_dataset_path: str, output_dataset_path: OutputPath(Dataset)):
    import pandas as pd
    import numpy as np

    data = pd.read_csv(input_dataset_path)

    # Binning age into categories
    age_bins = [0, 35, 55, 100]  # Define age bins
    age_labels = ['Young', 'Middle-aged', 'Senior']
    data['age_group'] = pd.cut(data['age'], bins=age_bins, labels=age_labels, right=False)

    #  Binning cigarettes per day into smoker categories
    cig_bins = [-1, 0, 10, 20, float('inf')]  # Define cigarette bins
    cig_labels = ['Non-smoker', 'Light smoker', 'Moderate smoker', 'Heavy smoker']
    data['smoker_type'] = pd.cut(data['cigsPerDay'], bins=cig_bins, labels=cig_labels, right=True)

    # Log transformation of income and blood pressure, handling cases where value might be zero
    data['log_income'] = np.log(data['income'] + 1)  # Adding 1 to avoid log(0)
    data['log_sysBP'] = np.log(data['sysBP'])
    data['log_diaBP'] = np.log(data['diaBP'])

    # Perform one-hot encoding on categorical variables
    data = pd.get_dummies(data, drop_first=True)
```

COMPONENT DEFINITION

SPLIT_DATASET

```
[22] from kfp.v2.dsl import component, InputPath, OutputPath


     @component(packages_to_install=["scikit-learn", "pandas"])
     def split_dataset(input_dataset_path: InputPath('Dataset'),
                       train_data_path: OutputPath('Dataset'),
                       validation_data_path: OutputPath('Dataset')):
         from sklearn.model_selection import train_test_split
         import pandas as pd
         df = pd.read_csv(input_dataset_path)

         train_data, validation_data = train_test_split(df, test_size=0.20, random_state=42)

         train_data.to_csv(train_data_path, index=False)

         validation_data.to_csv(validation_data_path, index=False)
```

USC Viterbi
School of Engineering

COMPONENT DEFINITION
IMPUTE_AGE_TRAINING

```
from kfp.v2.dsl import Output
from kfp.v2.dsl import Artifact

@component(packages_to_install=["pandas"])
def impute_age_training(training_dataset_path: InputPath('Dataset'),
                   imputed_dataset_path: OutputPath('Dataset'),
                   cig_mean: Output[Artifact],
                        BP_mean: Output[Artifact],
                        EDU_mean: Output[Artifact],
                        Chol_mean: Output[Artifact],
                        BMI_mean:Output[Artifact],
                        glucose_mean: Output[Artifact],
                        a1c_mean: Output[Artifact],
                        heartRate_mean: Output[Artifact]):
    # Load the training dataset
    import pandas as pd
    df = pd.read_csv(training_dataset_path)
    # Replace missing values with the median of the column
    df = df.drop(['patientID'], axis=1)
    cig_value = df['cigsPerDay'].mean()
    df['cigsPerDay'].fillna(cig_value, inplace=True)
    BP_value = df['BPMeds'].median()
    df['BPMeds'].fillna(BP_value, inplace=True)
    EDU_value = df['education'].median()
    df['education'].fillna(EDU_value, inplace=True)
    Chol = df['totChol'].mean()
    df['totChol'].fillna(Chol, inplace=True)
    BMI = df['BMI'].mean()
    df['BMI'].fillna(BMI, inplace=True)
    glucose = df['glucose'].mean()
    df['glucose'].fillna(glucose, inplace=True)
    a1c = df['a1c'].mean()
```

COMPONENT DEFINITION
IMPUTE_AGE_TRAINING

```python
df['BPMeds'].fillna(BP_value, inplace=True)
EDU_value = df['education'].median()
df['education'].fillna(EDU_value, inplace=True)
Chol = df['totChol'].mean()
df['totChol'].fillna(Chol, inplace=True)
BMI = df['BMI'].mean()
df['BMI'].fillna(BMI, inplace=True)
glucose = df['glucose'].mean()
df['glucose'].fillna(glucose, inplace=True)
a1c = df['a1c'].mean()
df['a1c'].fillna(a1c, inplace=True)
heartRate = df['heartRate'].mean()
df['heartRate'].fillna(heartRate, inplace=True)

# Save the imputed dataframe to the output path
df.to_csv(imputed_dataset_path, index=False)

# Output the median value
cig_mean.metadata['value'] = cig_value
BP_mean.metadata['value'] = BP_value
EDU_mean.metadata['value'] = EDU_value
Chol_mean.metadata['value'] = Chol
BMI_mean.metadata['value'] = BMI
glucose_mean.metadata['value'] = glucose
a1c_mean.metadata['value'] = a1c
heartRate_mean.metadata['value'] = heartRate
```

COMPONENT DEFINITION

IMPUTE_AGE_VALIDATION

```
[24] from kfp.v2.dsl import Input
     from kfp.v2.dsl import Model

     @component(packages_to_install=["pandas"])
     def impute_age_validation(validation_dataset_path: InputPath('Dataset'),
                               imputed_dataset_path: OutputPath('Dataset'),
                               cig_mean: Input[Artifact],
                               BP_mean: Input[Artifact],
                               EDU_mean: Input[Artifact],
                               Chol_mean: Input[Artifact],
                               BMI_mean:Input[Artifact],
                               glucose_mean: Input[Artifact],
                               a1c_mean: Input[Artifact],
                               heartRate_mean: Input[Artifact]):
         import pandas as pd
         # Load the test dataset
         df = pd.read_csv(validation_dataset_path)

         # Impute missing values in the 'Glucose' column with the provided median value
         df = df.drop(['patientID'], axis=1)
         df['cigsPerDay'].fillna(cig_mean.metadata['value'], inplace=True)
         df['BPMeds'].fillna(BP_mean.metadata['value'], inplace=True)
         df['education'].fillna(EDU_mean.metadata['value'], inplace=True)
         df['totChol'].fillna(Chol_mean.metadata['value'], inplace=True)
         df['BMI'].fillna(BMI_mean.metadata['value'], inplace=True)
         df['glucose'].fillna(glucose_mean.metadata['value'], inplace=True)
         df['a1c'].fillna(a1c_mean.metadata['value'], inplace=True)
         df['heartRate'].fillna(heartRate_mean.metadata['value'], inplace=True)
         # Save the imputed dataframe to the output path
         df.to_csv(imputed_dataset_path, index=False)
```

COMPONENT DEFINITION
PERFORM_SMOTE

[25]
```python
@component(packages_to_install=["pandas", "numpy", "scikit-learn", "imbalanced-learn==0.11.0"])
def perform_SMOTE(input_df_path:  InputPath('Dataset'),
                  output_df_path: OutputPath('Dataset')):
    import pandas as pd
    import numpy as np
    from imblearn.over_sampling import SMOTE

    # Load the input dataset
    df = pd.read_csv(input_df_path)

    X = df.drop('TenYearCHD', axis = 1)
    y = df['TenYearCHD']

    # Perform SMOTE oversampling
    smote = SMOTE()
    X_smote, y_smote = smote.fit_resample(X, y)

    # Convert the oversampled feature set and target vector back into a DataFrame
    X_smote_df = pd.DataFrame(X_smote, columns=X.columns)
    y_smote_df = pd.DataFrame(y_smote, columns=['TenYearCHD'])

    # Re-join the features and the target into a single DataFrame
    oversampled_df = pd.concat([X_smote_df, y_smote_df], axis=1)

    # Save the re-joined, oversampled dataset to the specified OutputPath
    oversampled_df.to_csv(output_df_path, index=False)
```

COMPONENT DEFINITION
TRAIN_LOGISTIC_REGRESSION

```
[26] @component(packages_to_install=["pandas", "scikit-learn", "joblib"])
     def train_logistic_regression(training_dataset_path: InputPath('Dataset'),
                                   trained_model_artifact: Output[Model]):

         import pandas as pd
         from sklearn.linear_model import LogisticRegression
         import joblib
         import os

         # Load the training data
         train_df = pd.read_csv(training_dataset_path)

         X_train = train_df.drop('TenYearCHD', axis=1)
         y_train = train_df['TenYearCHD']

         trained_model = LogisticRegression(max_iter=1000)
         trained_model.fit(X_train, y_train)

         # Save the model to the designated gcs output path
         os.makedirs(trained_model_artifact.path, exist_ok=True)
         joblib.dump(trained_model, os.path.join(trained_model_artifact.path, "model.joblib"))
```

COMPONENT DEFINITION
TRAIN_RANDOM_FOREST

```
@component(packages_to_install=["pandas", "scikit-learn", "joblib"])
def train_random_forest(training_dataset_path: InputPath('Dataset'),
                        trained_model_artifact: Output[Model],
                        n_estimators: int = 100,
                        max_depth: int = None):
    import pandas as pd
    from sklearn.ensemble import RandomForestClassifier
    import joblib
    import os

    train_df = pd.read_csv(training_dataset_path)
    X_train = train_df.drop('TenYearCHD', axis=1)
    y_train = train_df['TenYearCHD']

    model = RandomForestClassifier(n_estimators=n_estimators, max_depth=max_depth, random_state=42)
    model.fit(X_train, y_train)

    os.makedirs(trained_model_artifact.path, exist_ok=True)
    joblib.dump(model, os.path.join(trained_model_artifact.path, "model.joblib"))
```

COMPONENT DEFINITION

TRAIN_SVM

```
@component(packages_to_install=["pandas", "scikit-learn", "joblib"])
def train_svm(training_dataset_path: InputPath('Dataset'),
              trained_model_artifact: Output[Model],
              C: float = 1.0,
              kernel: str = 'rbf'):
    import pandas as pd
    from sklearn.svm import SVC
    import joblib
    import os

    train_df = pd.read_csv(training_dataset_path)
    X_train = train_df.drop('TenYearCHD', axis=1)
    y_train = train_df['TenYearCHD']

    model = SVC(C=C, kernel=kernel, probability=True, random_state=42)
    model.fit(X_train, y_train)

    os.makedirs(trained_model_artifact.path, exist_ok=True)
    joblib.dump(model, os.path.join(trained_model_artifact.path, "model.joblib"))
```

COMPONENT DEFINITION
TRAIN_XGBOOST

```
@component(packages_to_install=["pandas", "scikit-learn", "joblib", "xgboost"])
def train_xgboost(training_dataset_path: InputPath('Dataset'),
                  trained_model_artifact: Output[Model],
                  n_estimators: int = 100,
                  max_depth: int = 3,
                  learning_rate: float = 0.1):
    import pandas as pd
    from xgboost import XGBClassifier
    import joblib
    import os

    train_df = pd.read_csv(training_dataset_path)
    X_train = train_df.drop('TenYearCHD', axis=1)
    y_train = train_df['TenYearCHD']

    model = XGBClassifier(n_estimators=n_estimators, max_depth=max_depth, learning_rate=learning_rate, random_state=42)
    model.fit(X_train, y_train)

    os.makedirs(trained_model_artifact.path, exist_ok=True)
    joblib.dump(model, os.path.join(trained_model_artifact.path, "model.joblib"))
```

COMPONENT DEFINITION

EVALUATE_MODEL

```
from kfp.v2.dsl import Metrics

@component(packages_to_install=["pandas", "scikit-learn", "joblib", "xgboost"])
def evaluate_model(test_dataset_path: InputPath('Dataset'),
                   model: Input[Model],
                   model_type: str,  # Add model type to customize evaluation messages
                   metrics: Output[Metrics]):
    import pandas as pd
    import joblib
    from sklearn.metrics import confusion_matrix, accuracy_score, f1_score

    test_df = pd.read_csv(test_dataset_path)
    X_test = test_df.drop(columns=['TenYearCHD'])
    y_test = test_df['TenYearCHD']

    model_file_path = model.path + "/model.joblib"
    trained_model = joblib.load(model_file_path)

    y_pred = trained_model.predict(X_test)
    tn, fp, fn, tp = confusion_matrix(y_test, y_pred).ravel()
    accuracy = accuracy_score(y_test, y_pred)
    f1 = f1_score(y_test, y_pred, average='weighted')

    metrics.log_metric("model_type", model_type)
    metrics.log_metric("accuracy", accuracy)
    metrics.log_metric("f1_score", f1)
    metrics.log_metric("true_negatives", int(tn))
    metrics.log_metric("false_positives", int(fp))
    metrics.log_metric("false_negatives", int(fn))
    metrics.log_metric("true_positives", int(tp))
```

SUMMARY DISCUSSION ABOUT PIPELINE

› Data Processing:

The pipeline begins with perform_initial_data_prep for initial data cleaning, followed by split_dataset to separate the data into training and validation datasets. Subsequent steps include impute-age-training and impute-age-validation for filling missing values in both datasets using calculated statistics.

› Model Training and Validation:

Post-imputation, perform_smote is applied to balance the training dataset. The pipeline utilizes 8 artifacts representing means or medians calculated during the imputation steps and model artifacts from the training steps. Several models are then trained (train-logistic-regression, train-random-forest, train-svm, train-xgboost) using the oversampled data. Each model is evaluated using its respective evaluate-model component.

SUMMARY DISCUSSION

1. DATA PREPROCESSING COMPONENT IN DETAIL

- Data Loading:

Reads data from the provided CSV file using pandas.

- Feature Engineering:
  - Age Binning: Classifies age into categories ('Young', 'Middle-aged', 'Senior') based on defined age bins ([0, 35, 55, 100]).
  - Variable: age_group:
  - Smoking Status Binning: Categorizes smoking habits into ('Non-smoker', 'Light smoker', 'Moderate smoker', 'Heavy smoker') using defined cigarette bins ([-1, 0, 10, 20, inf]).
  - Variable: smoker_type:
- Data Transformation:
  - Log Transformation: log_income, log_sysBP, log_diaBP

  Applies logarithmic transformation to 'income', 'sysBP', and 'diaBP' to normalize the distribution. Adds 1 to income to handle zero values and ensure non-negative input for the logarithm.
- One-hot Encoding: Converts categorical variables into a format that can be provided to ML algorithms to better predict the result. Uses pd.get_dummies for encoding, dropping the first category to avoid dummy variable trap.

SUMMARY DISCUSSION

OTHER COMPONENTS

- Dataset Splitting (split_dataset): Splits the cleaned dataset into training and validation datasets.
  - Variables: input_dataset_path: Input path for the dataset, train_data_path: Output path for the training dataset, validation_data_path: Output path for the validation dataset.
- Imputation on Training Data (impute_age_training): Imputes missing values in the training dataset using statistical methods (mean, median).
  - Variables: imputed_training_data: Imputed training dataset.
  - Artifacts like BMI_mean, glucose_mean.
- Imputation on Validation Data (impute_age_validation): Applies training data statistics to impute missing values in the validation dataset.
  - Variables: imputed_validation_data (uses the same artifacts from the training imputation).
- SMOTE Oversampling (perform_smote): Applies SMOTE to the training data to balance class distribution.
  - Variables:imputed_dataset_path: Input path for imputed data, oversampled_training_data: Output path for balanced training data.
- Model Training (e.g., train_logistic_regression, train_random_forest, train_svm, train_xgboost): Trains various models on the oversampled training data.
  - Variables: Outputs like trained_lr_model, trained_rf_model for each model type.
- Model Evaluation (e.g., evaluate_model1, evaluate_model2, evaluate_model3, evaluate_model4): Evaluates each trained model on the validation dataset using performance metrics.
  - Variables:Each model artifact. imputed_validation_dataset: Path for validation data.

SUMMARY DISCUSSION

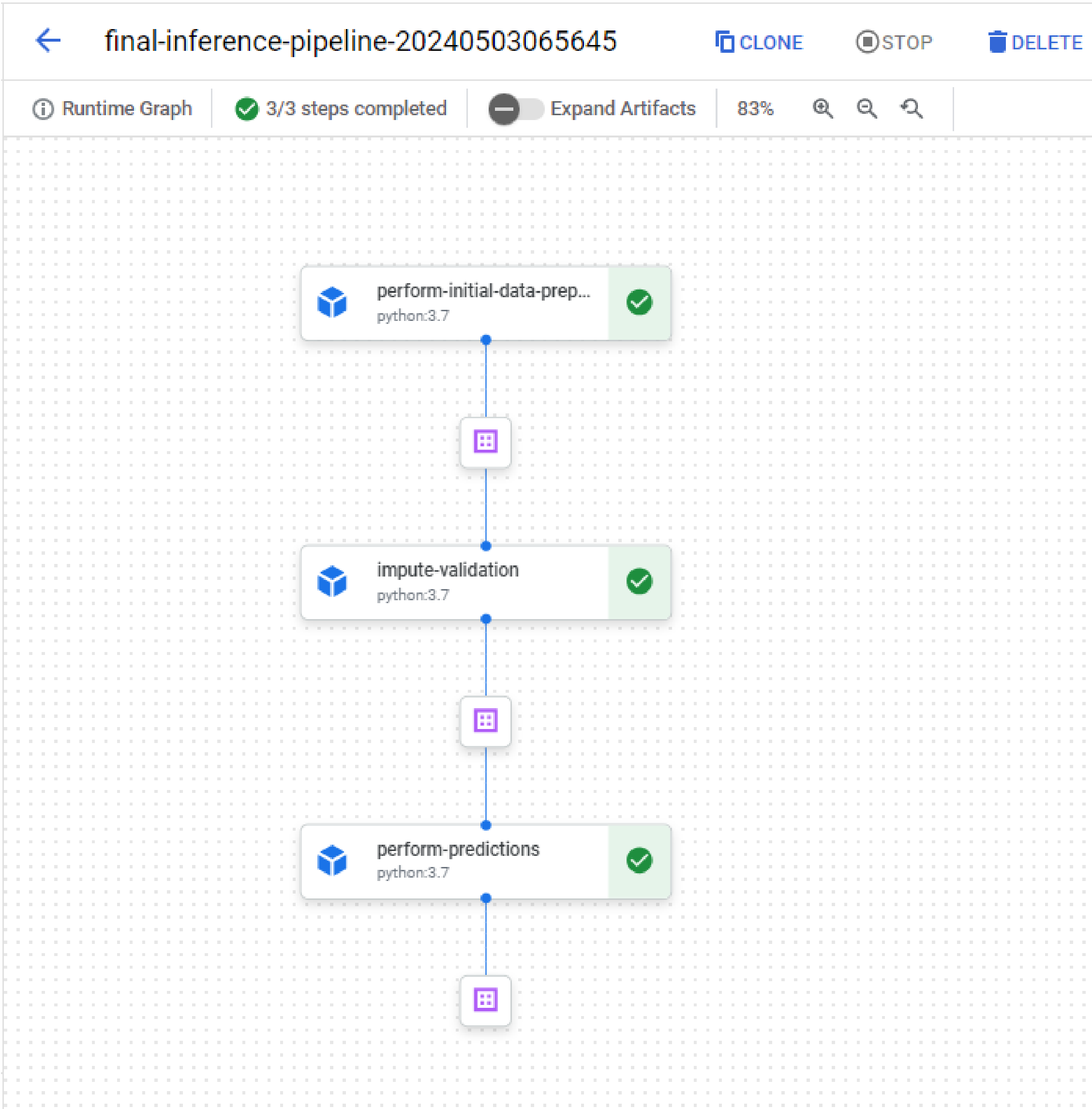
RESULT

- We deployed four different machine learning models in parallel within our predictive analytics pipeline to identify the most effective approach for our dataset. The models used included Logistic Regression, Random Forest, Support Vector Machine (SVM), and XGBoost.
- Performance Insights:
  - Among the models evaluated, XGBoost demonstrated superior performance.
  - Metrics Achieved:
    - Accuracy: 84.03%
    - F1 Score: 79.41%
    - These metrics were calculated based on a balanced assessment of both precision (minimizing false positives) and recall (minimizing false negatives), with the model yielding 13 true positives and 629 true negatives, while maintaining low false positives (13) and higher false negatives (109).

Conclusion:
The XGBoost model outperformed its counterparts in both accuracy and F1 score, marking it as the most suitable model for our project's needs based on the current dataset.

INFERENCE PIPELINE
PIPELINE VISUALIZATION

- To be add



INFERENCE PIPELINE ARTIFACT

```
artifact_paths = {
    "BMI_mean": "gs://architafinalproject/812826359571/chd-prediction-pipeline-20240501010201/impute-age-training_-5571590455801413632/BMI_mean ",
    "BP_mean": "gs://architafinalproject/812826359571/chd-prediction-pipeline-20240501010201/impute-age-training_-5571590455801413632/BP_mean json",
    "Chol_mean": "gs://architafinalproject/812826359571/chd-prediction-pipeline-20240501010201/impute-age-training_-5571590455801413632/Chol_mean",
    "EDU_mean": "gs://architafinalproject/812826359571/chd-prediction-pipeline-20240501010201/impute-age-training_-5571590455801413632/EDU_mean",
    "a1c_mean": "gs://architafinalproject/812826359571/chd-prediction-pipeline-20240501010201/impute-age-training_-5571590455801413632/a1c_mean ",
    "cig_mean": "gs://architafinalproject/812826359571/chd-prediction-pipeline-20240501010201/impute-age-training_-5571590455801413632/cig_mean",
    "glucose_mean": "gs://architafinalproject/812826359571/chd-prediction-pipeline-20240501010201/impute-age-training_-5571590455801413632/glucose_mean ",
    "heartRate_mean": "gs://architafinalproject/812826359571/chd-prediction-pipeline-20240501010201/impute-age-training_-5571590455801413632/heartRate_mean"
}
```

```
imputed_age_artifact_path ="gs://architafinalproject/812826359571/chd-prediction-pipeline-20240501010201/impute-age-training_-5571590455801413632/executor_output.json"
```

```
[49] imputed_age_artifact_path = pd.read_json(imputed_age_artifact_path).to_dict()
     imputed_age_artifact_path
```


```
{'artifacts': {'BMI_mean': {'artifacts': [{'name': 'projects/812826359571/locations/us-west2/metadataStores/default/artifacts/13142662851457072687',
    'uri': 'gs://architafinalproject/812826359571/chd-prediction-pipeline-20240501010201/impute-age-training_-5571590455801413632/BMI_mean',
    'metadata': {'value': 25.785610800131707}}]},
  'BP_mean': {'artifacts': [{'name': 'projects/812826359571/locations/us-west2/metadataStores/default/artifacts/13110617755604810145',
    'uri': 'gs://architafinalproject/812826359571/chd-prediction-pipeline-20240501010201/impute-age-training_-5571590455801413632/BP_mean',
    'metadata': {'value': 0.0}}]},
  'Chol_mean': {'artifacts': [{'name': 'projects/812826359571/locations/us-west2/metadataStores/default/artifacts/16311907782319626649',
    'uri': 'gs://architafinalproject/812826359571/chd-prediction-pipeline-20240501010201/impute-age-training_-5571590455801413632/Chol_mean',
    'metadata': {'value': 241.6118115461181}}]},
  'EDU_mean': {'artifacts': [{'name': 'projects/812826359571/locations/us-west2/metadataStores/default/artifacts/1213062247966963069',
    'uri': 'gs://architafinalproject/812826359571/chd-prediction-pipeline-20240501010201/impute-age-training_-5571590455801413632/EDU_mean',
    'metadata': {'value': 2.0}}]},
  'a1c_mean': {'artifacts': [{'name': 'projects/812826359571/locations/us-west2/metadataStores/default/artifacts/8799601154233068976',
    'uri': 'gs://architafinalproject/812826359571/chd-prediction-pipeline-20240501010201/impute-age-training_-5571590455801413632/a1c_mean',
    'metadata': {'value': 4.281971983139468}}]},
  'cig_mean': {'artifacts': [{'name': 'projects/812826359571/locations/us-west2/metadataStores/default/artifacts/18093737387197017579',
    'uri': 'gs://architafinalproject/812826359571/chd-prediction-pipeline-20240501010201/impute-age-training_-5571590455801413632/cig_mean',
    'metadata': {'value': 18.404524284763806}}]},
  'glucose_mean': {'artifacts': [{'name': 'projects/812826359571/locations/us-west2/metadataStores/default/artifacts/11091348671394712524',
    'uri': 'gs://architafinalproject/812826359571/chd-prediction-pipeline-20240501010201/impute-age-training_-5571590455801413632/glucose_mean',
    'metadata': {'value': 81.63514492753623}}]},
  'heartRate_mean': {'artifacts': [{'name': 'projects/812826359571/locations/us-west2/metadataStores/default/artifacts/12562446418026050530',
    'uri': 'gs://architafinalproject/812826359571/chd-prediction-pipeline-20240501010201/impute-age-training_-5571590455801413632/heartRate_mean',
    'metadata': {'value': 75.75385119632907}}]},
  'imputed_dataset_path': {'artifacts': [{'name': 'projects/812826359571/locations/us-west2/metadataStores/default/artifacts/9960838621201295682',
    'uri': 'gs://architafinalproject/812826359571/chd-prediction-pipeline-20240501010201/impute-age-training_-5571590455801413632/imputed_dataset_path',
    'metadata': {}}]}}}
```

INFERENCE PIPELINE
PERFORM_INITIAL_DATA_PREPARATION

› To b

```python
from kfp.v2.dsl import InputPath, OutputPath, Dataset

@component(packages_to_install=["pandas", "numpy", "fsspec", "gcsfs", "xgboost"])
def perform_initial_data_preparation(input_dataset_path: str, output_dataset_path: OutputPath(Dataset)):
    import pandas as pd
    import numpy as np

    data = pd.read_csv(input_dataset_path)

    # Binning age into categories
    age_bins = [0, 35, 55, 100]  # Define age bins
    age_labels = ['Young', 'Middle-aged', 'Senior']
    data['age_group'] = pd.cut(data['age'], bins=age_bins, labels=age_labels, right=False)

    #  Binning cigarettes per day into smoker categories
    cig_bins = [-1, 0, 10, 20, float('inf')]  # Define cigarette bins
    cig_labels = ['Non-smoker', 'Light smoker', 'Moderate smoker', 'Heavy smoker']
    data['smoker_type'] = pd.cut(data['cigsPerDay'], bins=cig_bins, labels=cig_labels, right=True)

    # Log transformation of income and blood pressure, handling cases where value might be zero
    data['log_income'] = np.log(data['income'] + 1)  # Adding 1 to avoid log(0)
    data['log_sysBP'] = np.log(data['sysBP'])
    data['log_diaBP'] = np.log(data['diaBP'])

    # Perform one-hot encoding on categorical variables
    data = pd.get_dummies(data, drop_first=True)

    # Convert 'demog Customer Age' to an integer
    # df["demog Customer Age"] = df["demog Customer Age"].astype(int)

    data.to_csv(output_dataset_path, index=False)
```

```
/usr/local/lib/python3.10/dist-packages/kfp/dsl/component_decorator.py:119: FutureWarning: Python 3.7 has reached
  return component_factory.create_component_from_func(
```

INFERENCE PIPELINE
IMPUTE_VALIDATION

› To k

```
from kfp.v2.dsl import Input
from kfp.v2.dsl import Model

@component(packages_to_install=["pandas"])
def impute_validation(validation_dataset_path: InputPath('Dataset'),
                      imputed_dataset_path: OutputPath('Dataset'),
                      average_cig: float,
                      median_BP: float,
                      median_education: float,
                      average_chol: float,
                      average_BMI:float,
                      average_glucose: float,
                      average_a1c: float,
                      average_heart_rate: float):
    import pandas as pd
    # Load the test dataset
    df = pd.read_csv(validation_dataset_path)

    # Impute missing values in the 'Glucose' column with the provided median value
    df['cigsPerDay'].fillna(average_cig, inplace=True)
    df['BPMeds'].fillna(median_BP, inplace=True)
    df['education'].fillna(median_education, inplace=True)
    df['totChol'].fillna(average_chol, inplace=True)
    df['BMI'].fillna(average_BMI, inplace=True)
    df['glucose'].fillna(average_glucose, inplace=True)
    df['a1c'].fillna(average_a1c, inplace=True)
    df['heartRate'].fillna(average_heart_rate, inplace=True)
    # Save the imputed dataframe to the output path
    df.to_csv(imputed_dataset_path, index=False)
```

INFERENCE PIPELINE
PERFORM_PREDICTION

```
@component(packages_to_install=["pandas", "numpy", "scikit-learn", "joblib", "fsspec", "gcsfs", "xgboost"])
def perform_predictions(dataset_for_prediction_path: InputPath('Dataset'),
                        model_path: str,
                        predictions_path: OutputPath('Dataset')):

    import pandas as pd
    import joblib
    import gcsfs

    # Create a GCS file system object
    import gcsfs
    import joblib

    fs = gcsfs.GCSFileSystem()

    with fs.open(model_path, 'rb') as f:
        trained_model = joblib.load(f)
      # best_estimator_ = trained_model

    # Access the individual base estimators of the BaggingClassifier
    # Load the test dataset
    pred_df = pd.read_csv(dataset_for_prediction_path)

    # Make predictions
    y_pred = trained_model.predict(pred_df.drop(['patientID'], axis=1))

    # Convert the predictions to a dataframe
    pred_df = pd.DataFrame(pred_df['patientID'])
    pred_df['pred'] = y_pred
    pred_df = pred_df[['patientID', 'pred']]

    # Save the predictions
    pred_df.to_csv(predictions_path, index=False)
```

INFERENCE PIPELINE

PIPELINE DEFINITION

```
[53] from kfp.v2.dsl import pipeline, Output, Dataset
imputed_artifact_path = "gs://architafinalproject/812826359571/chd-prediction-pipeline-20240501010201/perform-initial-data-preparation_-9599044373740257 28/executor_output.json"
model_path = 'gs://architafinalproject/812826359571/chd-prediction-pipeline-20240501013404/train-xgboost_-6580396772332404736/trained_model_artifact/model.joblib'
training_bmi_median = imputed_age_artifact_path['artifacts']['BMI_mean']['artifacts'][0]['metadata']['value']
imputed_bp_dictionary=imputed_age_artifact_path['artifacts']['BP_mean']['artifacts'][0]['metadata']['value']
imputed_chol_dictionary=imputed_age_artifact_path['artifacts']['Chol_mean']['artifacts'][0]['metadata']['value']
imputed_edu_dictionary=imputed_age_artifact_path['artifacts']['EDU_mean']['artifacts'][0]['metadata']['value']
imputed_a1c_dictionary=imputed_age_artifact_path['artifacts']['a1c_mean']['artifacts'][0]['metadata']['value']
imputed_cig_dictionary=imputed_age_artifact_path['artifacts']['cig_mean']['artifacts'][0]['metadata']['value']
imputed_glucose_dictionary=imputed_age_artifact_path['artifacts']['glucose_mean']['artifacts'][0]['metadata']['value']
imputed_hr_dictionary=imputed_age_artifact_path['artifacts']['heartRate_mean']['artifacts'][0]['metadata']['value']
@pipeline(name='final-inference-pipeline')
def final_inference_pipeline(dataset_for_predictions_path: str,
                             training_bmi_median: float = training_bmi_median,
                             imputed_bp_dictionary: float = imputed_bp_dictionary,
                             imputed_chol_dictionary: float = imputed_chol_dictionary,
                             imputed_edu_dictionary: float = imputed_edu_dictionary,
                             imputed_a1c_dictionary: float = imputed_a1c_dictionary,
                             imputed_cig_dictionary: float = imputed_cig_dictionary,
                             imputed_glucose_dictionary: float = imputed_glucose_dictionary,
                             imputed_hr_dictionary: float = imputed_hr_dictionary,
                             model_uri: str = model_path):

    # Process dataset - initial data preparation
    initial_prepared_dataset = perform_initial_data_preparation(input_dataset_path=dataset_for_predictions_path)

    # Impute age
    imputed_dataset = impute_validation(
       validation_dataset_path=initial_prepared_dataset.outputs['output_dataset_path'],
                        average_cig = imputed_cig_dictionary,
                        median_BP = imputed_bp_dictionary,
                        median_education = imputed_edu_dictionary,
                        average_chol = imputed_chol_dictionary,
                        average_BMI = training_bmi_median,
                        average_glucose =  imputed_glucose_dictionary,
                        average_a1c = imputed_a1c_dictionary,
                        average_heart_rate = imputed_hr_dictionary
    )

    perform_predictions(
        dataset_for_prediction_path=imputed_dataset.outputs['imputed_dataset_path'],
        model_path=model_uri
    )
```

INFERENCE PIPELINE

PIPELINE RUN

- To be added

```
from kfp.v2 import compiler

compiler.Compiler().compile(
    pipeline_func=final_inference_pipeline,
    package_path = 'final_inference_pipeline.json'
)

pipeline_job = aiplatform.PipelineJob(
    display_name='final_inference_pipeline',
    template_path='final_inference_pipeline.json',
    pipeline_root='gs://architafinalproject',
    parameter_values={
        'dataset_for_predictions_path': 'gs://architafinalproject/Final Project Evaluation Dataset - Student.csv'
    },
    enable_caching=True
)
```

```
pipeline_job.run()
```

```
INFO:google.cloud.aiplatform.pipeline_jobs:Creating PipelineJob
INFO:google.cloud.aiplatform.pipeline_jobs:PipelineJob created. Resource name: projects/812826359571/locations/us-west2/pipelineJobs/final-inference-pipeline-20240503065645
INFO:google.cloud.aiplatform.pipeline_jobs:To use this PipelineJob in another session:
INFO:google.cloud.aiplatform.pipeline_jobs:pipeline_job = aiplatform.PipelineJob.get('projects/812826359571/locations/us-west2/pipelineJobs/final-inference-pipeline-20240503065645')
INFO:google.cloud.aiplatform.pipeline_jobs:View Pipeline Job:
https://console.cloud.google.com/vertex-ai/locations/us-west2/pipelines/runs/final-inference-pipeline-20240503065645?project=812826359571
INFO:google.cloud.aiplatform.pipeline_jobs:PipelineJob projects/812826359571/locations/us-west2/pipelineJobs/final-inference-pipeline-20240503065645 current state:
PipelineState.PIPELINE_STATE_RUNNING
INFO:google.cloud.aiplatform.pipeline_jobs:PipelineJob projects/812826359571/locations/us-west2/pipelineJobs/final-inference-pipeline-20240503065645 current state:
PipelineState.PIPELINE_STATE_RUNNING
INFO:google.cloud.aiplatform.pipeline_jobs:PipelineJob projects/812826359571/locations/us-west2/pipelineJobs/final-inference-pipeline-20240503065645 current state:
PipelineState.PIPELINE_STATE_RUNNING
INFO:google.cloud.aiplatform.pipeline_jobs:PipelineJob projects/812826359571/locations/us-west2/pipelineJobs/final-inference-pipeline-20240503065645 current state:
PipelineState.PIPELINE_STATE_RUNNING
INFO:google.cloud.aiplatform.pipeline_jobs:PipelineJob projects/812826359571/locations/us-west2/pipelineJobs/final-inference-pipeline-20240503065645 current state:
PipelineState.PIPELINE_STATE_RUNNING
INFO:google.cloud.aiplatform.pipeline_jobs:PipelineJob run completed. Resource name: projects/812826359571/locations/us-west2/pipelineJobs/final-inference-pipeline-20240503065645
```

SUMMARY DISCUSSION

INFERENCE_PIPELINE

- Perform Initial Data Preparation (perform-initial-data-prep): Prepares the incoming data for subsequent analysis and predictions.
  - Processes:
    - Binning age into categories with age_bins and age_labels.
    - Categorizing smoking habits using cig_bins and cig_labels.
    - Log transformation of income (log_income) and blood pressure (log_sysBP, log_diaBP).
    - One-hot encoding on categorical variables to enhance model input.
- Impute Validation Data (impute-validation): Ensures the validation dataset is complete by filling missing values using pre-determined statistics.
  - Processes:
    - Applies statistical imputation for features like BMI (average_BMI), cholesterol (average_chol), and others based on training data calculations.
- Perform Predictions (perform-predictions): Uses the prepared and imputed data to generate predictions from a trained model.
  - Processes:
    - Loads a trained model from model_path.
    - Predicts outcomes on the validation set and organizes the results for evaluation.
- This pipeline is specifically designed for inference, using a well-defined sequence of data preparation, imputation, and prediction to handle new or existing data effectively for real-time decision-making.

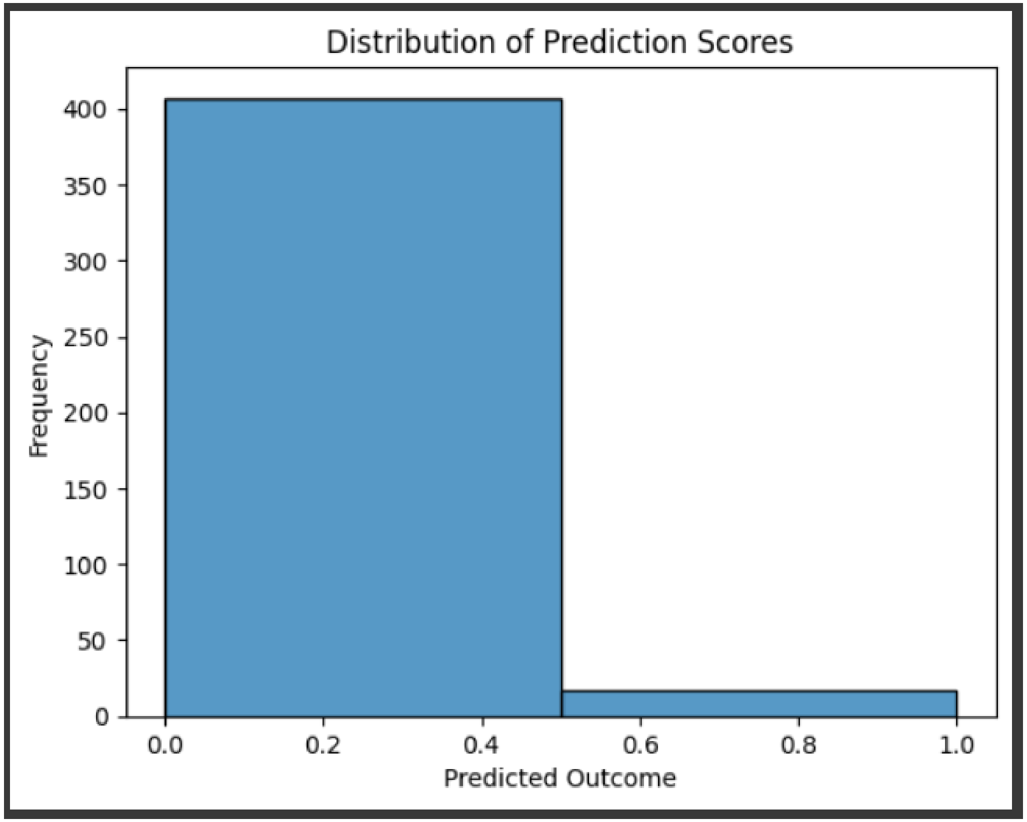
PREDICTIONS

CSV

	A	B	C	D
1	patientID	pred		
2	110399	0		
3	189047	0		
4	957019	0		
5	208967	0		
6	230935	0		
7	216024	0		
8	368834	1		
9	135175	0		
10	294070	0		
11	595710	0		
12	425597	0		
13	650137	0		
14	590019	0		
15	925626	0		
16	276518	0		
17	342284	0		
18	469306	0		
19	197764	0		
20	416488	0		
21	208652	0		

PREDICTIONS

- High Frequency of Non-CHD Predictions: The graph predominantly shows that a large number of individuals are predicted to have no risk of coronary heart disease, as indicated by the scores clustered at 0.0.
- Few CHD Predictions: There is a minimal count of cases predicted as having a risk of coronary heart disease, shown by the few occurrences at score 1.0.



THANK YOU NOTE

Dear Prof. Bruce & CPs Ruixin Deng, Sanath Sridhar, Tian Cui, Xizhu Lin, and Zimin Zhu,

Thank you for guiding me through an enlightening Business Intelligence course journey! Your efforts are deeply appreciated.