USC Viterbi
School of Engineering

# PATIENT READMISSION PREDICTION
# ISE-543 GCP PROJECT

USC Viterbi School of Engineering

# READMISSION PREDICTION

## OUTLINE

- Exploratory Data Analysis Report
- Data Preparation Plan
- Model Pipelines
- Summary Discussion

USC Viterbi
School of Engineering

# EDA REPORT

## OUTLINE

- Dataset overview
- Data quality summary
- Statistical summary of dataset
- Univariate analysis
- Bivariate analysis

USC Viterbi School of Engineering

# EDA REPORT

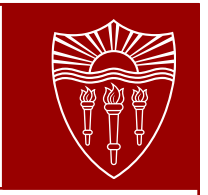
## DATASET OVERVIEW

- The Dataset ‘healthcare_readmissions_dataset_train’ contains the following dimensions - (8038, 19)
- The Dataset contains columns related to the patient readmission, with target variable as readmission within 30 days, a binary classification problem.
-

| A | B | C | D | E | F | G | H | I | J | K | L | M | N | O | P | Q | R | S | T |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| PatientID | Age | Gender | Ethnicity | Hospital ID | Height (m) | Smoker | BMI | Weight (kg) | Adjusted Wei | Has Diabetes | Has Hyperter | Exercise Fre | Diet Type | Number of Pr | Medications | Length of Sta | Type of Treat | Readmission within 30 Days | |
| 1000000 | 23 | Female | African Amer | Hosp2 | 1.6 | FALSE | 25 | 64 | 63.2833464 | 0 | 0 | Regular | High-fat | 3 | 3 | 0 | None | 0 | |
| 1000002 | 56 | Female | Hispanic | Hosp3 | 1.8 | TRUE | 27 | 87.5 | 87.6788588 | 0 | 0 | Regular | High-fat | 2 | | 2 | None | 0 | |
| 1000003 | 28 | Male | African Amer | Hosp1 | 1.8 | FALSE | 35 | 113.4 | 113.497844 | 0 | 1 | None | Other | | 2 | 5 | None | 0 | |
| 1000004 | 70 | Female | Caucasian | Hosp2 | 1.8 | FALSE | 27.7 | 89.7 | 89.7176939 | 0 | 0 | None | Other | 3 | | 0 | Major Surger | 0 | |
| 1000005 | 48 | Female | Hispanic | Hosp1 | 1.9 | FALSE | 22.4 | 80.9 | 80.5289275 | 0 | 0 | Occasional | High-fat | 7 | 5 | 7 | Major Surger | 1 | |
| 1000006 | 51 | Female | African Amer | Hosp1 | 1.8 | FALSE | 29.1 | 94.3 | 94.4678508 | 0 | 0 | Regular | Vegetarian | 3 | | 2 | Major Surger | 0 | |
| 1000008 | 63 | Female | Hispanic | Hosp3 | 1.8 | TRUE | 32.8 | 106.3 | 106.416939 | 0 | 0 | None | High-fat | 2 | 6 | 2 | None | 0 | |
| 1000009 | 64 | Female | African Amer | Hosp1 | 1.8 | FALSE | 35.3 | 114.4 | 115.135124 | 0 | 0 | Occasional | High-fat | | 4 | 4 | Minor Surger | 0 | |
| 1000010 | 56 | Female | Caucasian | Hosp1 | 1.9 | FALSE | 18.1 | 65.3 | 65.406647 | 0 | 0 | Regular | Balanced | 4 | 4 | 0 | Minor Surger | 0 | |
| 1000013 | 59 | Male | African Amer | Hosp3 | 1.7 | FALSE | 32.5 | 93.9 | 93.490119 | 0 | 0 | None | High-fat | 4 | 6 | 0 | Other Treatm | 0 | |
| 1000014 | 43 | Male | African Amer | Hosp1 | 1.7 | TRUE | 27.5 | 79.5 | 79.6010148 | 0 | 0 | Occasional | Other | 6 | 4 | 1 | None | 0 | |
| 1000015 | 50 | Female | Hispanic | Hosp3 | 1.6 | FALSE | 40.7 | 104.2 | 104.21197 | 0 | 0 | Occasional | High-fat | 2 | 5 | 3 | Minor Surger | 0 | |

# EDA

## DATA, VARIABLE EXPLORATION



› The Dataset has a total of 8038 unique patient, which means one row per patientID in the dataset.

```
print("Unique patients count : ", len(data['PatientID'].unique()))
```

› 8038 Unique Patients.

# EDA REPORT

## VARIABLE EXPLORATION

Exploring the variables and their types before the analysis

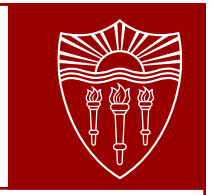
```
PatientID                      int64
Age                            int64
Gender                        object
Ethnicity                     object
Hospital ID                   object
Height (m)                   float64
Smoker                          bool
BMI                          float64
Weight (kg)                  float64
Adjusted Weight (kg)         float64
Has Diabetes                   int64
Has Hypertension               int64
Exercise Frequency            object
Diet Type                     object
Number of Prior Visits       float64
Medications Prescribed       float64
Length of Stay                 int64
Type of Treatment             object
Readmission within 30 Days     int64
dtype: object
```

There are integers, float64, bool and object data types in the columns

```
data.dtypes.value_counts()

int64      6
object     6
float64    6
bool       1
Name: count, dtype: int64
```

# EDA REPORT

## VARIABLE EXPLORATION



› Checking for missing values in the Columns

```
data.isnull().sum()
```

```
16]
```

| | |
|---|---|
| PatientID | 0 |
| Age | 0 |
| Gender | 0 |
| Ethnicity | 0 |
| Hospital ID | 0 |
| Height (m) | 0 |
| Smoker | 0 |
| BMI | 0 |
| Weight (kg) | 0 |
| Adjusted Weight (kg) | 0 |
| Has Diabetes | 0 |
| Has Hypertension | 0 |
| Exercise Frequency | 2350 |
| Diet Type | 0 |
| Number of Prior Visits | 314 |
| Medications Prescribed | 657 |
| Length of Stay | 0 |
| Type of Treatment | 2486 |
| Readmission within 30 Days | 0 |

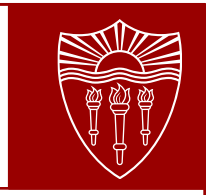
dtype: int64

The Exercise Frequency, Number of Prior Visits and type of Treatment columns has missing values.

USC Viterbi School of Engineering

# EDA REPORT

## VARIABLE EXPLORATION



› Getting an overall summary to get a quick view of the dataset would be good to proceed with further analysis.

›

| | PatientID | Age | Height (m) | BMI | Weight (kg) | Adjusted Weight (kg) | Has Diabetes | Has Hypertension | Number of Prior Visits | N |
|---|---|---|---|---|---|---|---|---|---|---|
| count | 8.038000e+03 | 8038.000000 | 8038.000000 | 8038.000000 | 8038.000000 | 8038.000000 | 8038.000000 | 8038.000000 | 7724.000000 | 7 |
| mean | 1.005018e+06 | 51.123787 | 1.700983 | 26.258335 | 77.145366 | 76.269064 | 0.132620 | 0.172431 | 3.044795 | |
| std | 2.888400e+03 | 20.040198 | 0.104152 | 4.753106 | 18.961059 | 16.701363 | 0.339185 | 0.377778 | 1.740236 | |
| min | 1.000000e+06 | 18.000000 | 1.300000 | 8.300000 | 23.300000 | 23.126324 | 0.000000 | 0.000000 | 0.000000 | |
| 25% | 1.002520e+06 | 38.000000 | 1.600000 | 23.100000 | 64.800000 | 64.720543 | 0.000000 | 0.000000 | 2.000000 | |
| 50% | 1.005022e+06 | 50.000000 | 1.700000 | 26.200000 | 75.400000 | 75.127095 | 0.000000 | 0.000000 | 3.000000 | |
| 75% | 1.007533e+06 | 62.000000 | 1.800000 | 29.300000 | 87.475000 | 86.904890 | 0.000000 | 0.000000 | 4.000000 | |
| max | 1.009999e+06 | 195.000000 | 2.000000 | 44.000000 | 236.300000 | 159.051116 | 1.000000 | 1.000000 | 11.000000 | |

We might not need the PatientID in the model as it doesn't signify meaningful value to the readmission variable

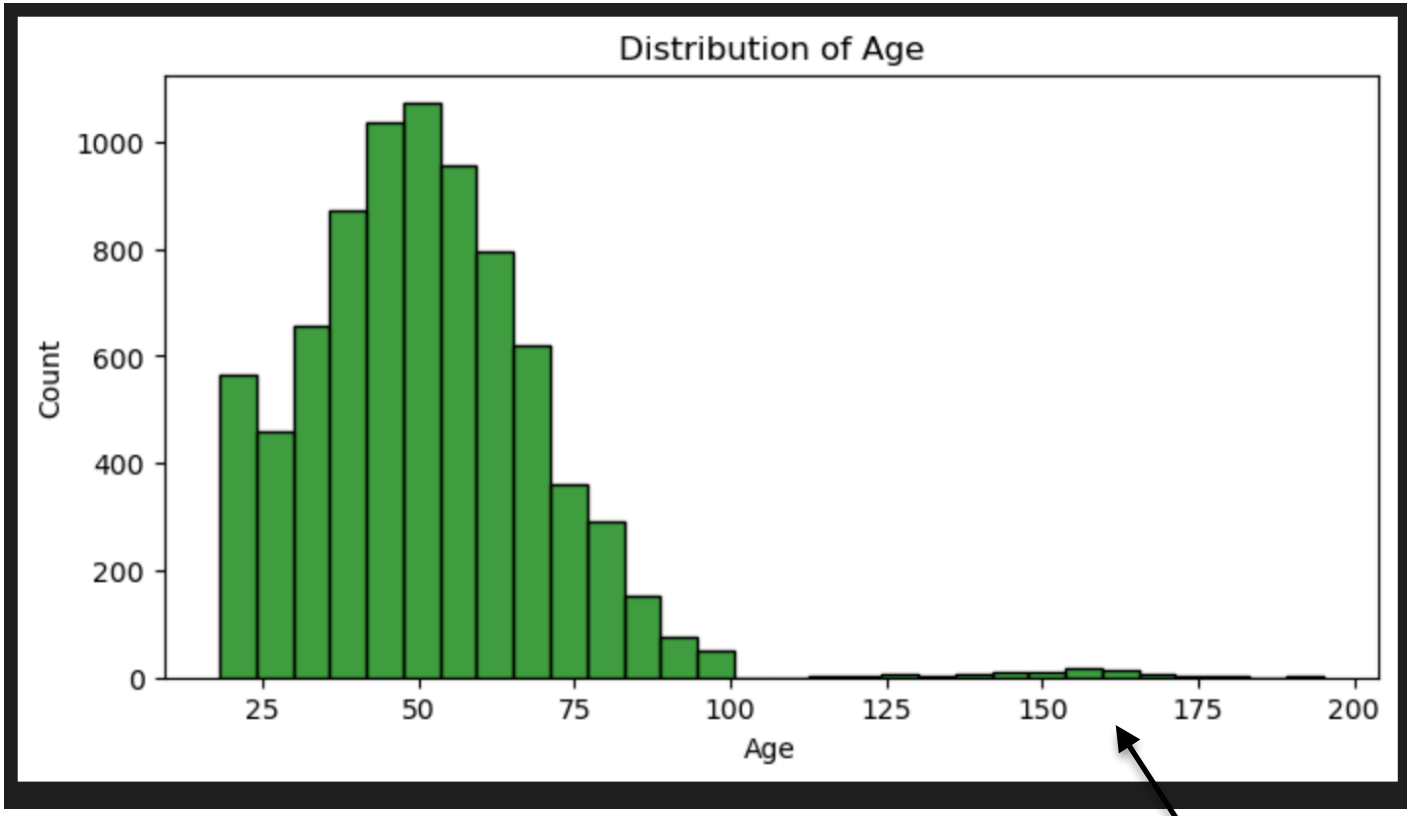
There might be skewness in the 'Age' category

We see the range for all variables are so wide and we might have to bring up to a same scale after choosing them for the model

# EDA

## UNIVARIATE ANALYSIS

› Checking how age is distributed

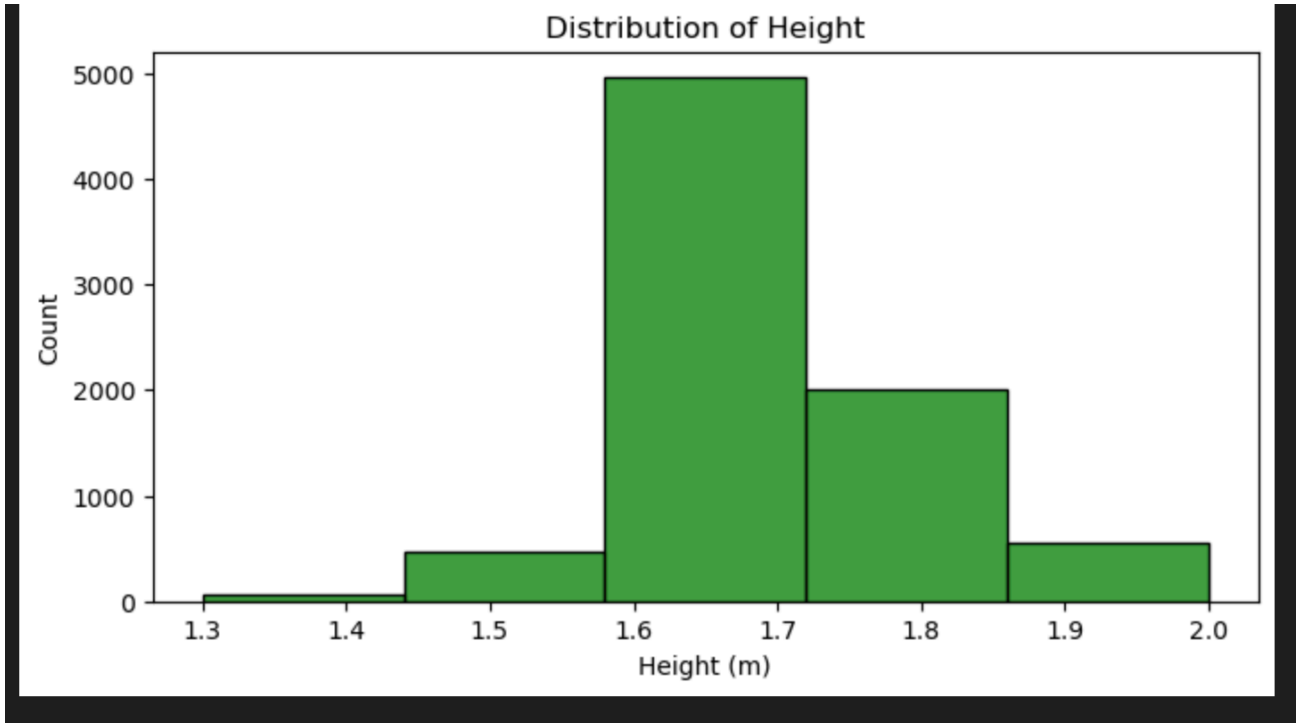



There is a longer tail on the right- hence it is **positively skewed** - with few people with higher age or might be faulty data

# EDA

## UNIVARIATE ANALYSIS

› Height appears to be closely normally distributed. But would height have an impact on the readmission?



# EDA

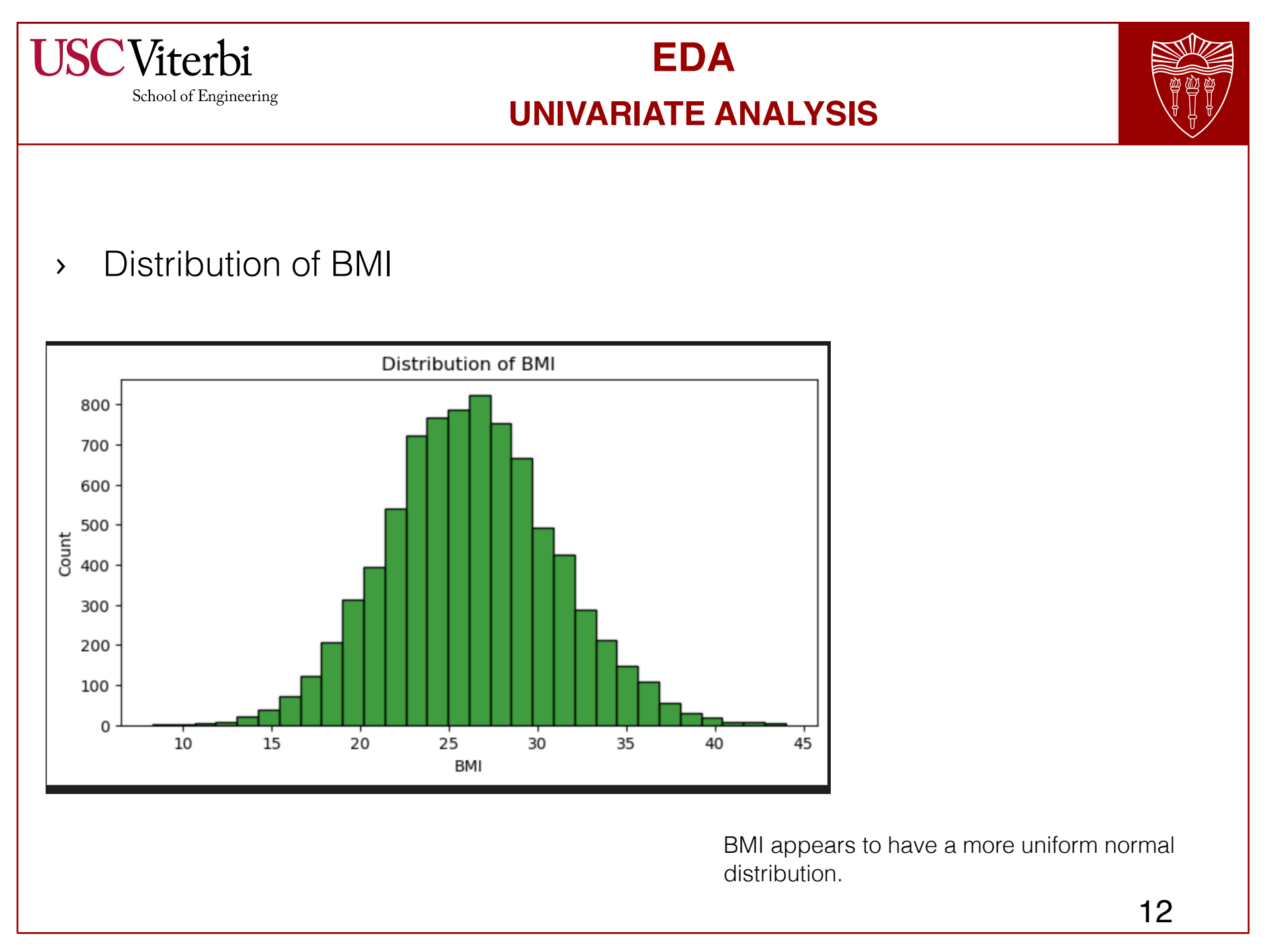
## UNIVARIATE ANALYSIS

› Checking if height has a correlation with readmission -

```
It has a correlation of 0.004827098515140524 – which could be nearly insignificant.
```

# EDA

## UNIVARIATE ANALYSIS



› Distribution of BMI

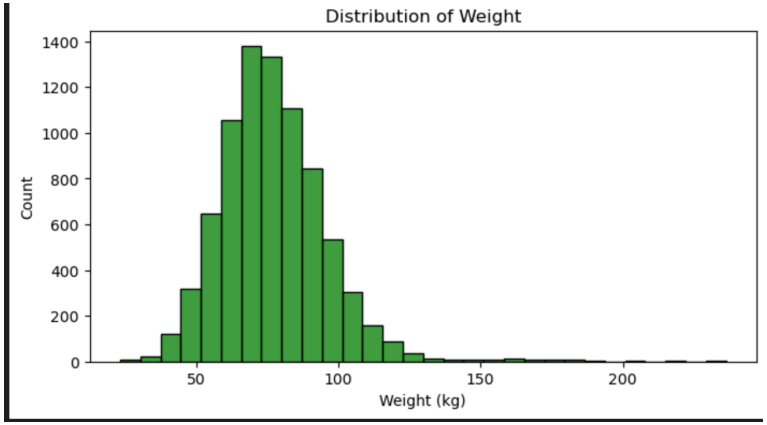


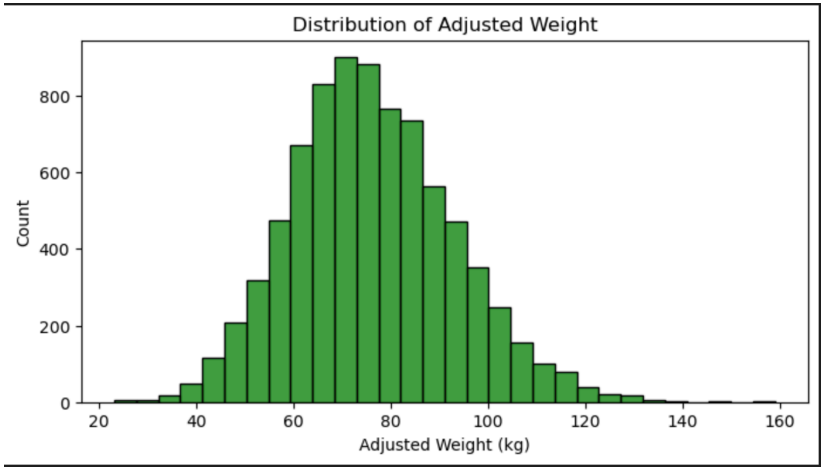

BMI appears to have a more uniform normal distribution.

# EDA

## UNIVARIATE ANALYSIS

- But BMI is related to Weight and Adjusted Weight - which are other components/variables in the dataset.
- Let's see the distribution of Weight and adjusted Weight.



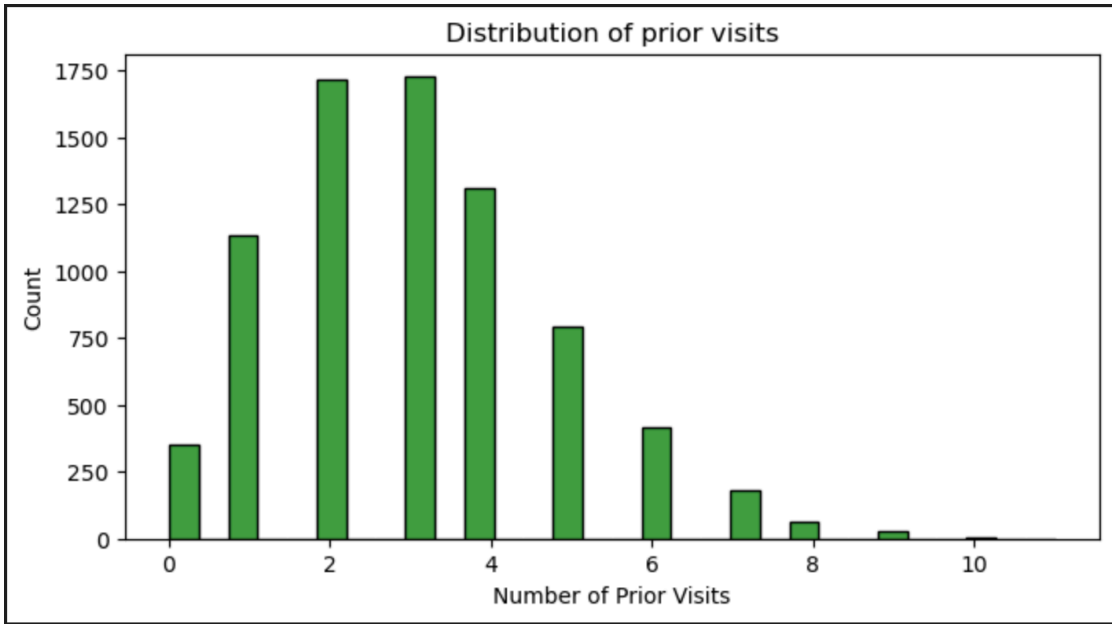




Adjusted Weight has a more wider distribution - since they are adjusted

# EDA

## UNIVARIATE ANALYSIS

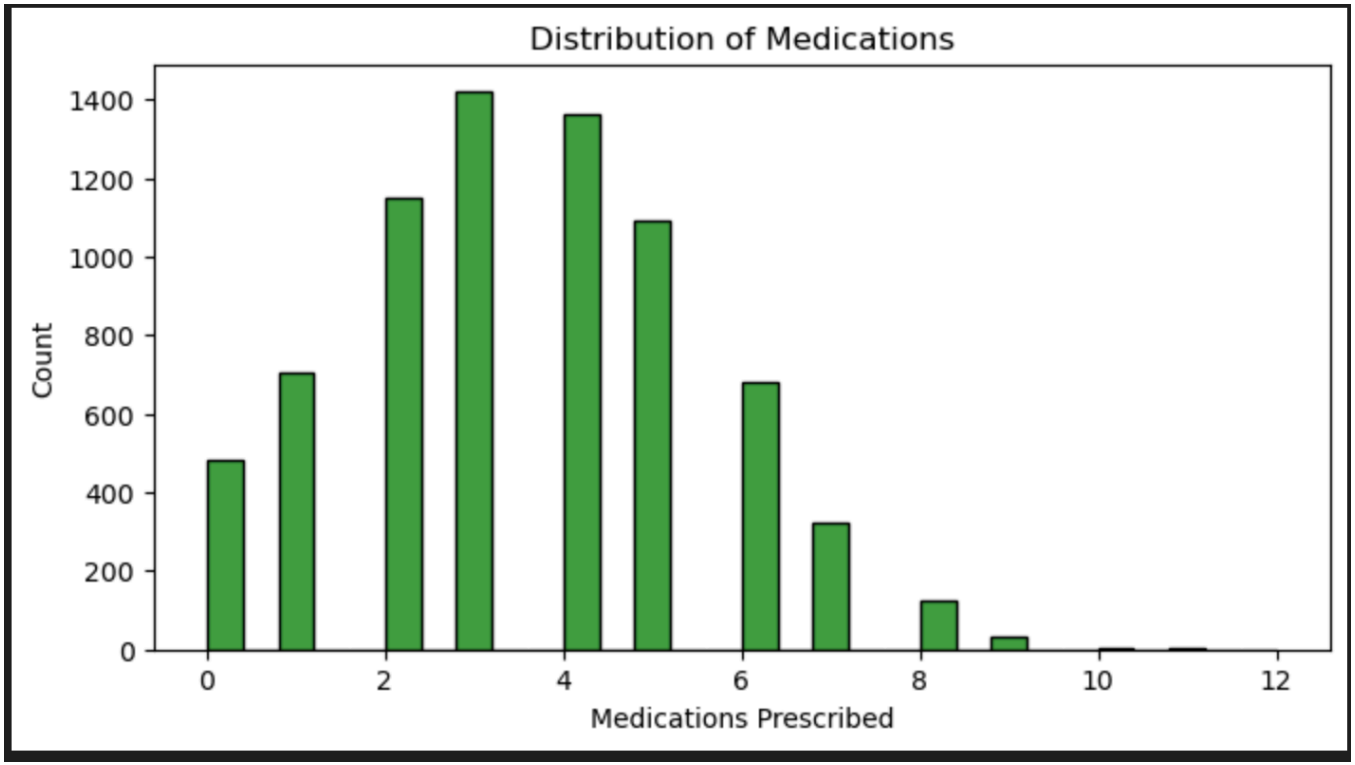
- The number of prior visits seems to have a slightly right skewed distribution with few people visiting more frequently than the majority.



# EDA

## UNIVARIATE ANALYSIS

- Number of Medications Prescribed

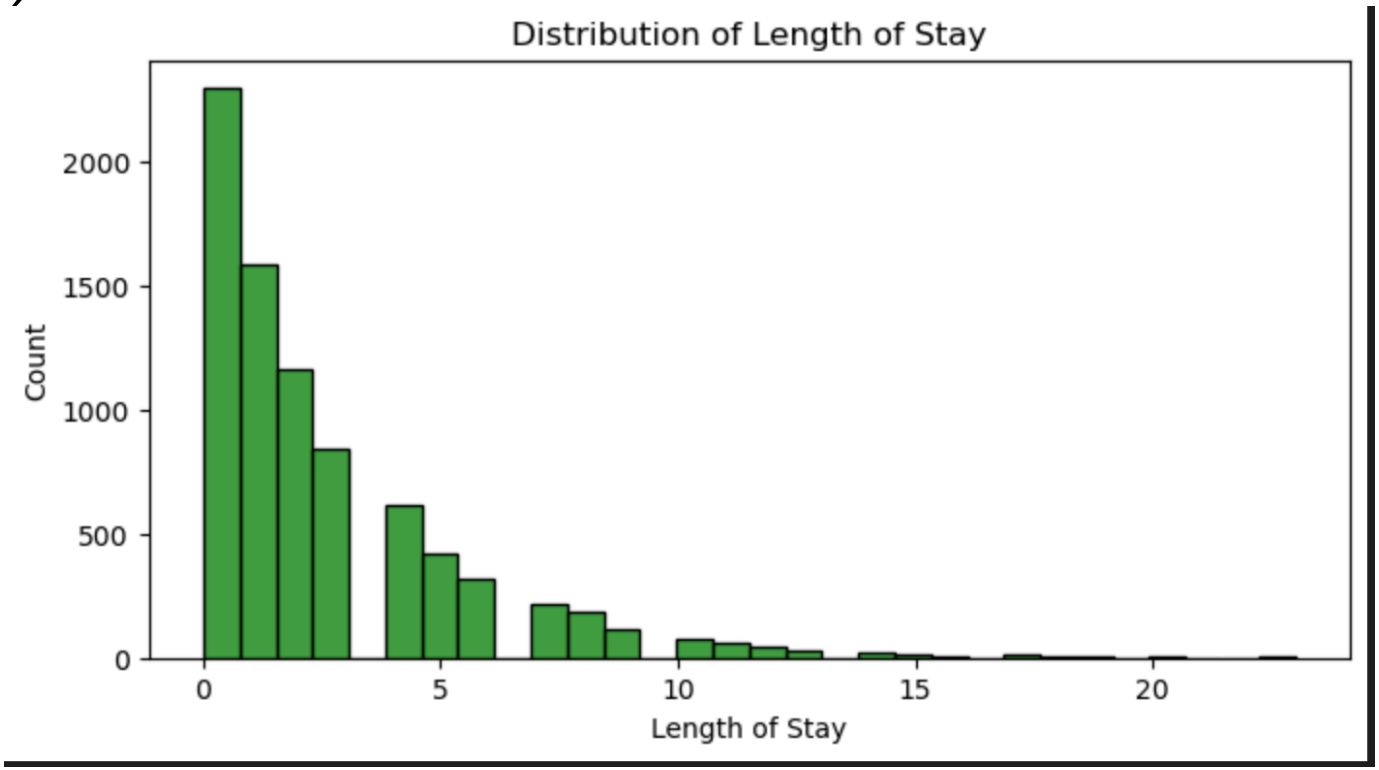



This appears to be a almost symmetrical normal distribution - but it has a slight skewness in the right.

# EDA

## UNIVARIATE ANALYSIS

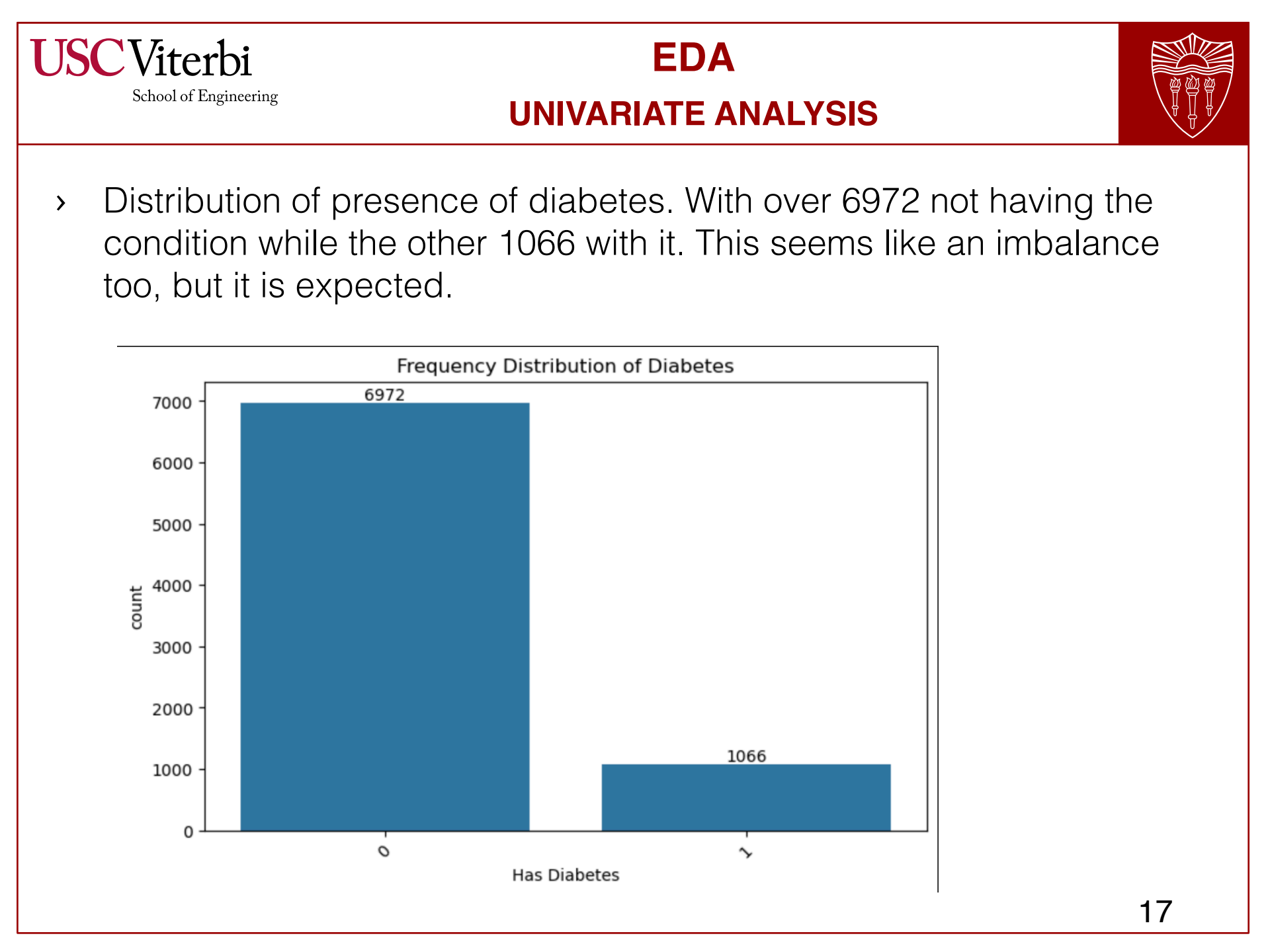
- Distribution of length of Stay




Length of Stay variable has highly right or positively skewed data distribution. They might have to be treated if being used in the model

# EDA

## UNIVARIATE ANALYSIS

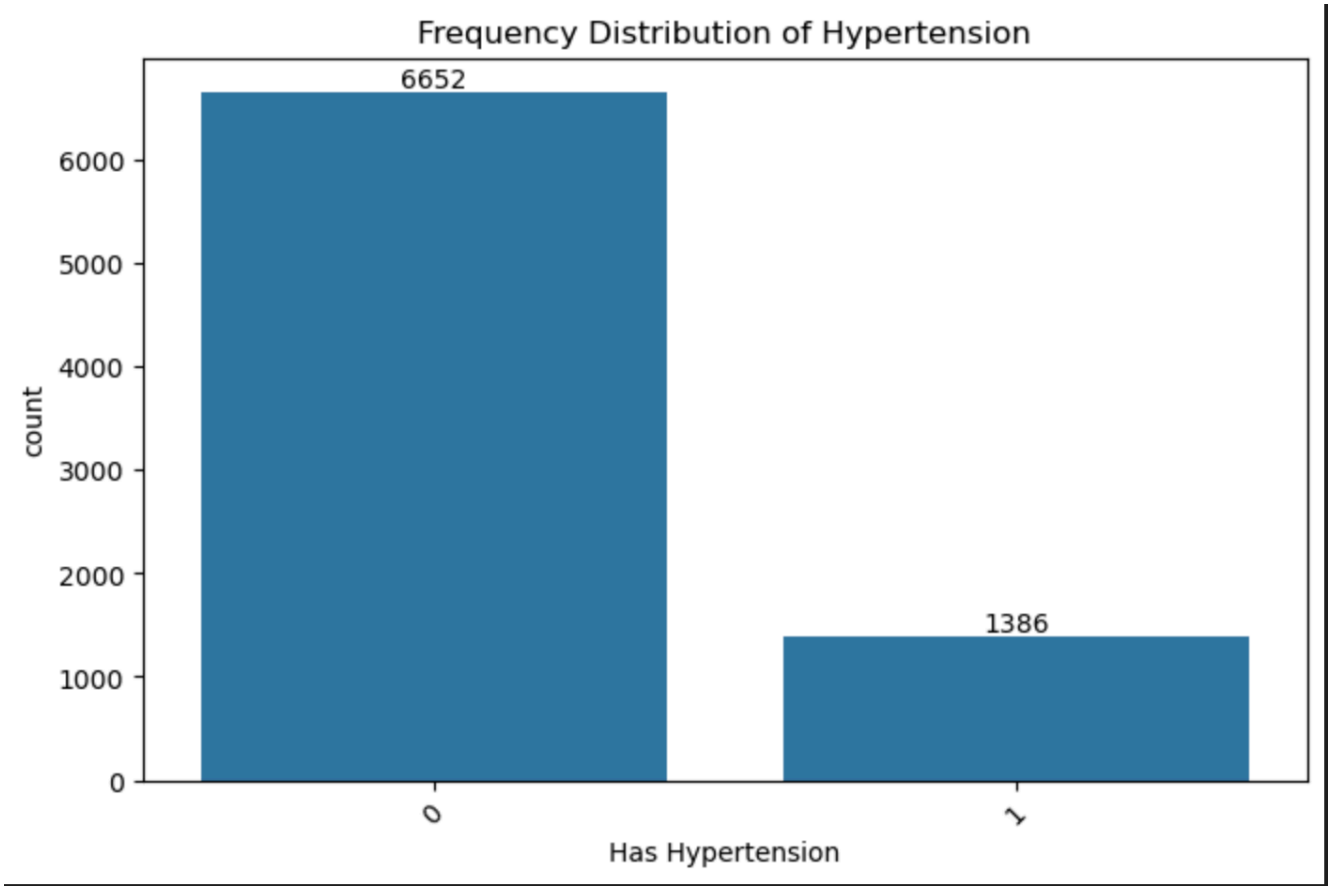


- Distribution of presence of diabetes. With over 6972 not having the condition while the other 1066 with it. This seems like an imbalance too, but it is expected.

# EDA

## UNIVARIATE ANALYSIS

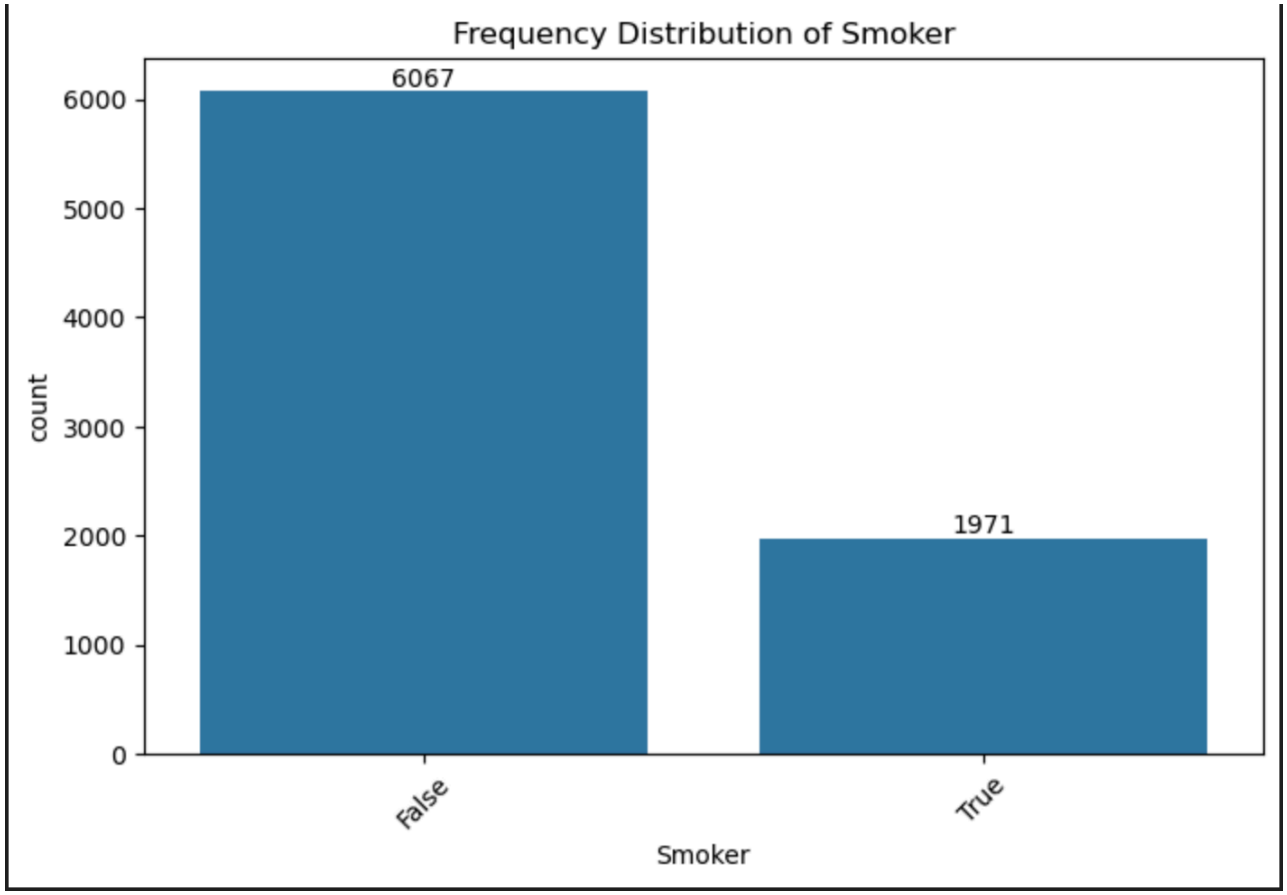
› Likewise, for hypertension, there's a similar huge difference with the people who have and don't have the condition.



# EDA

## UNIVARIATE ANALYSIS

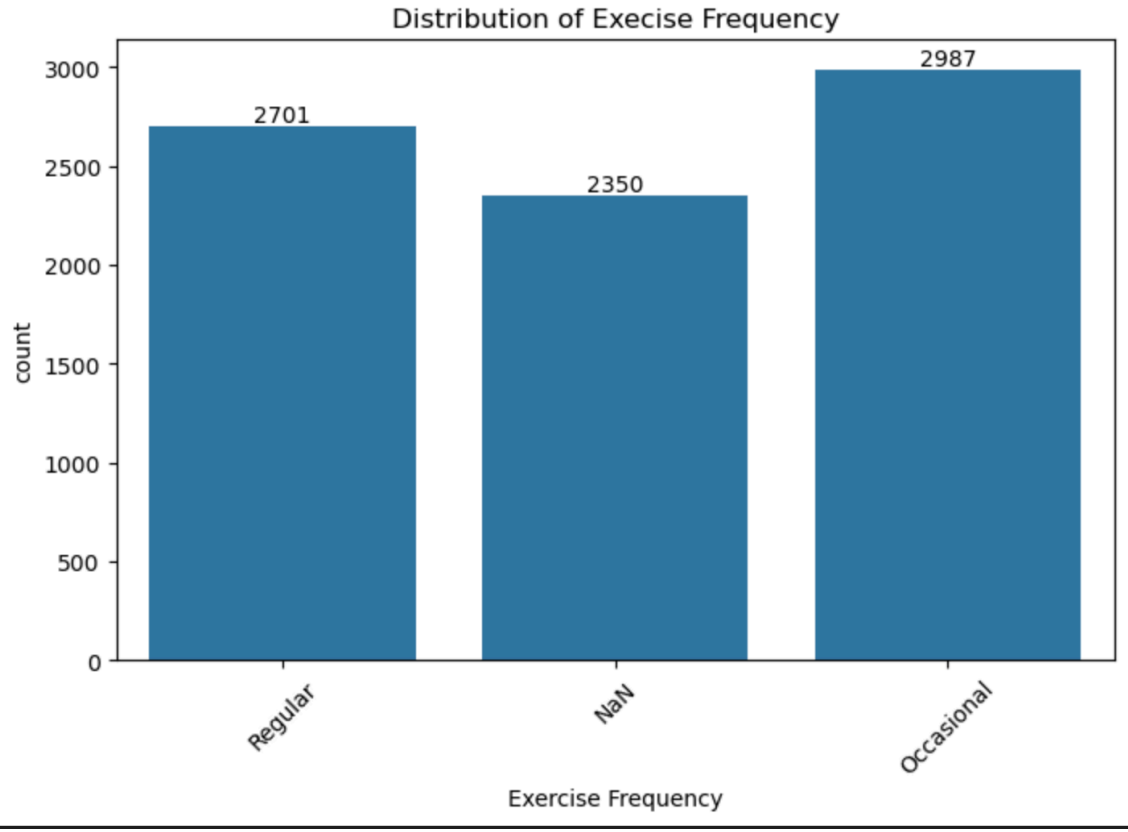
- But, compared to diabetes and hypertension, the number of people who smoke is higher, than those with presence of conditions



# EDA

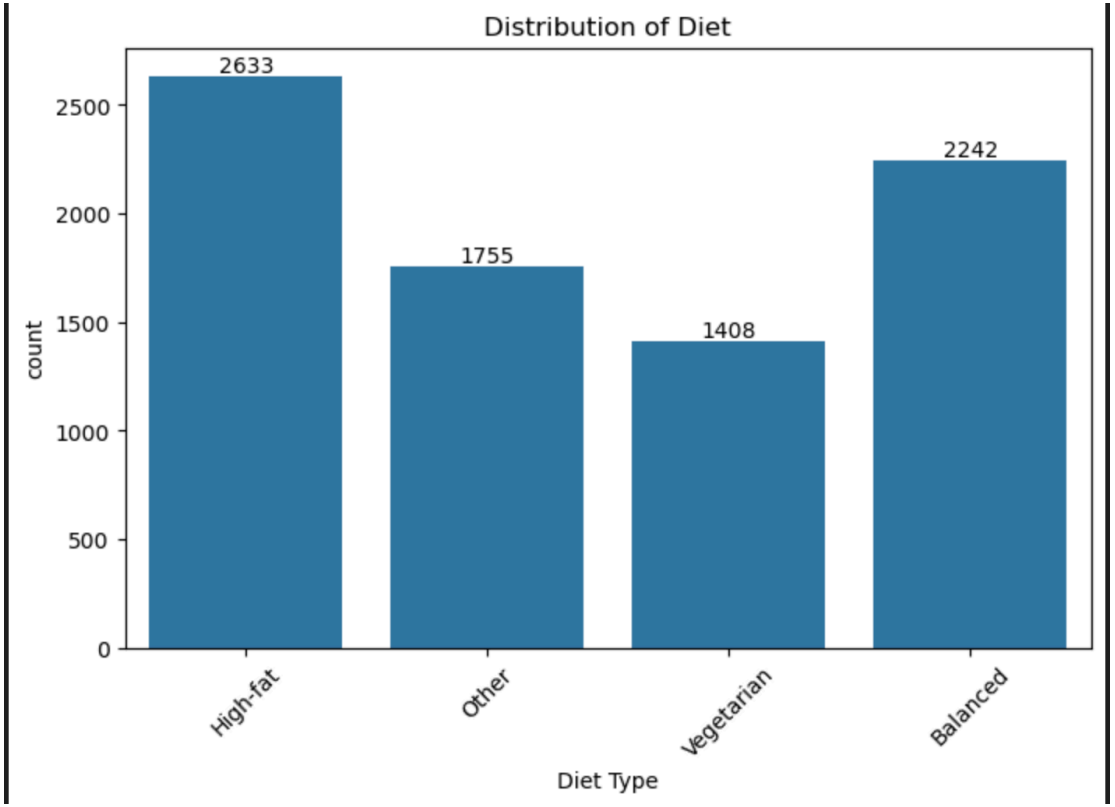
## UNIVARIATE ANALYSIS

- Most people tend to do some occasional exercise, the second higher proportion having a regular exercise routine.
- The rest cohort, doesn't perform any exercise



# EDA
## UNIVARIATE ANALYSIS

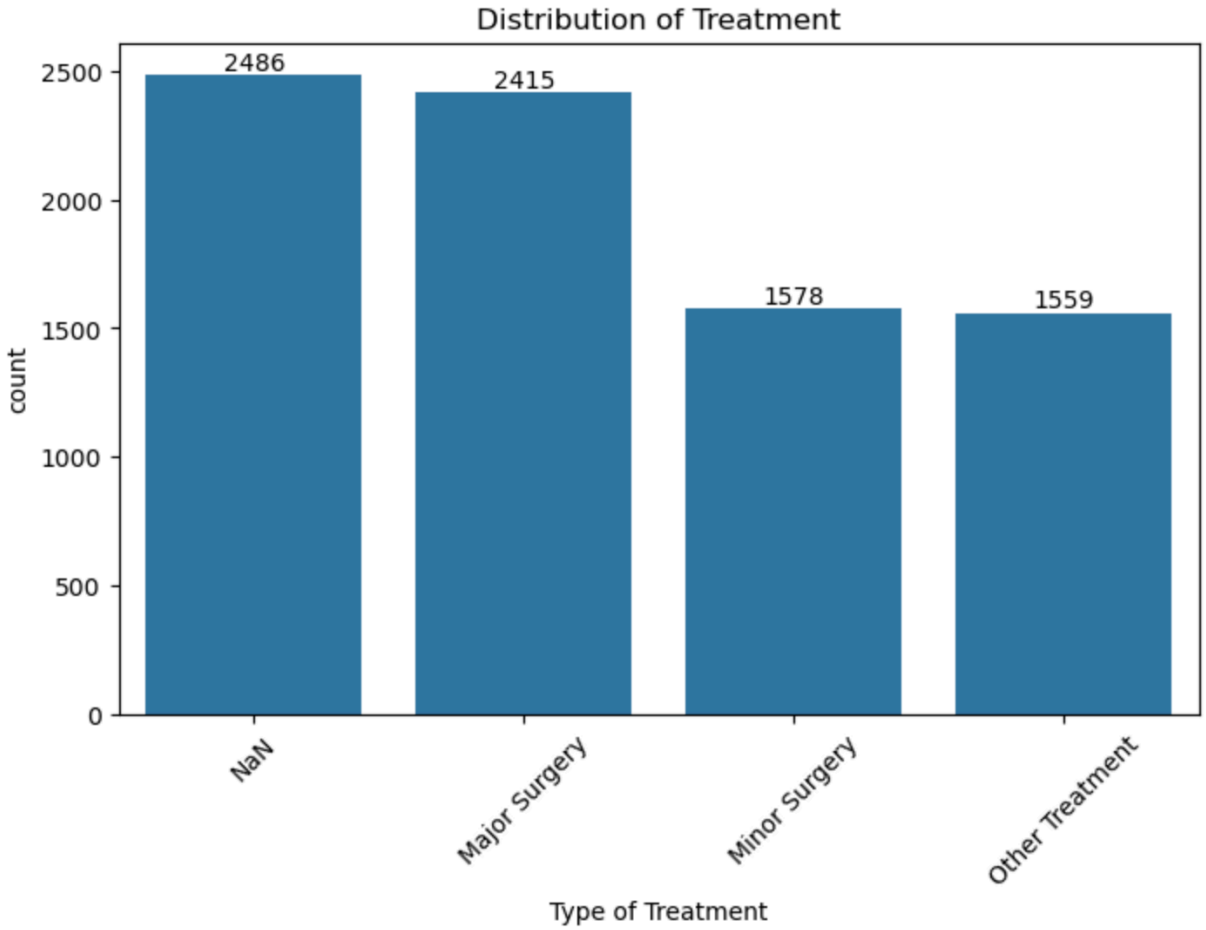
- Most people tend to have a high-fat diet. With vegetarians being the least proportion



# EDA

## UNIVARIATE ANALYSIS

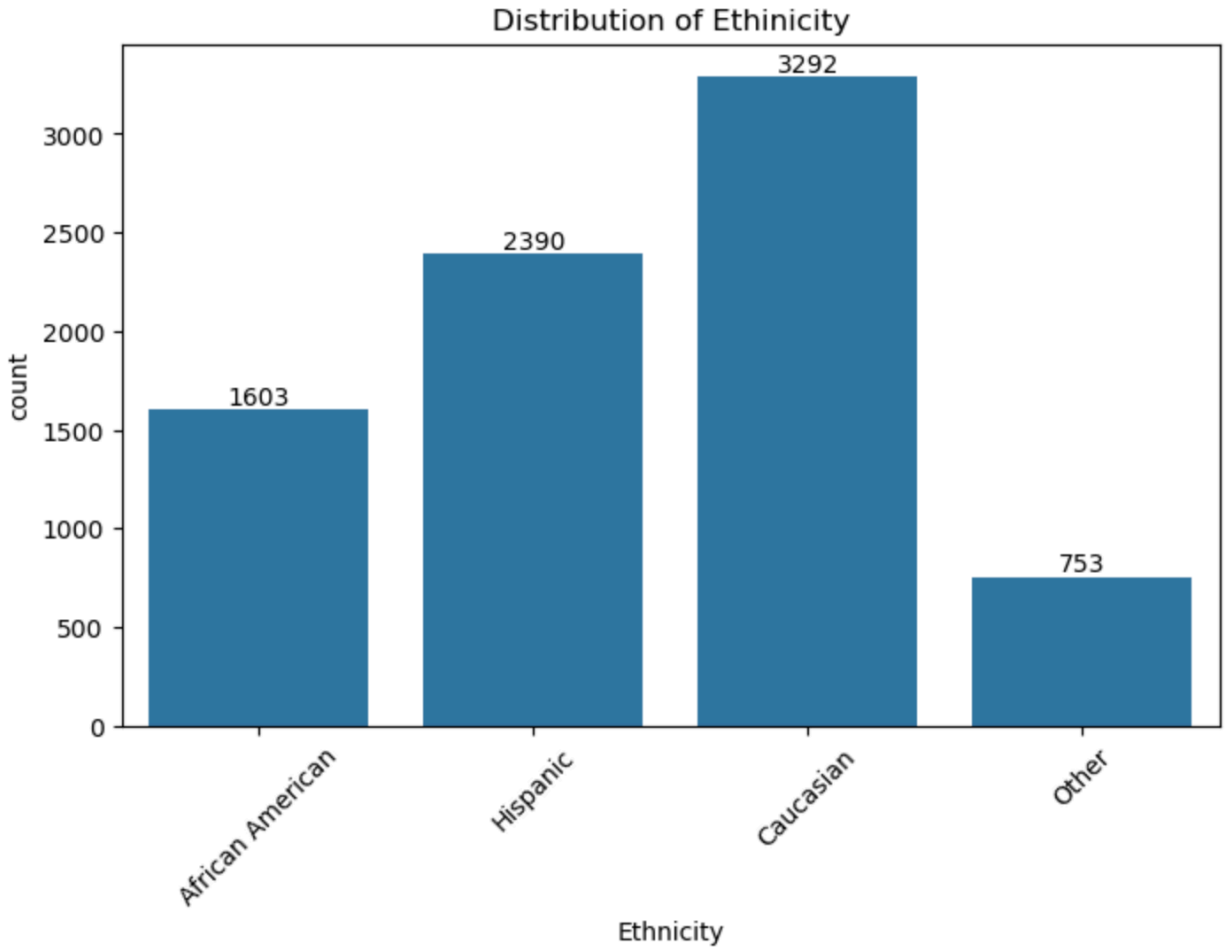
- Most people do not have a treatment. But do treatments affect readmission? A bivariate plot will reveal that information - later in the slides.



# EDA

## UNIVARIATE ANALYSIS

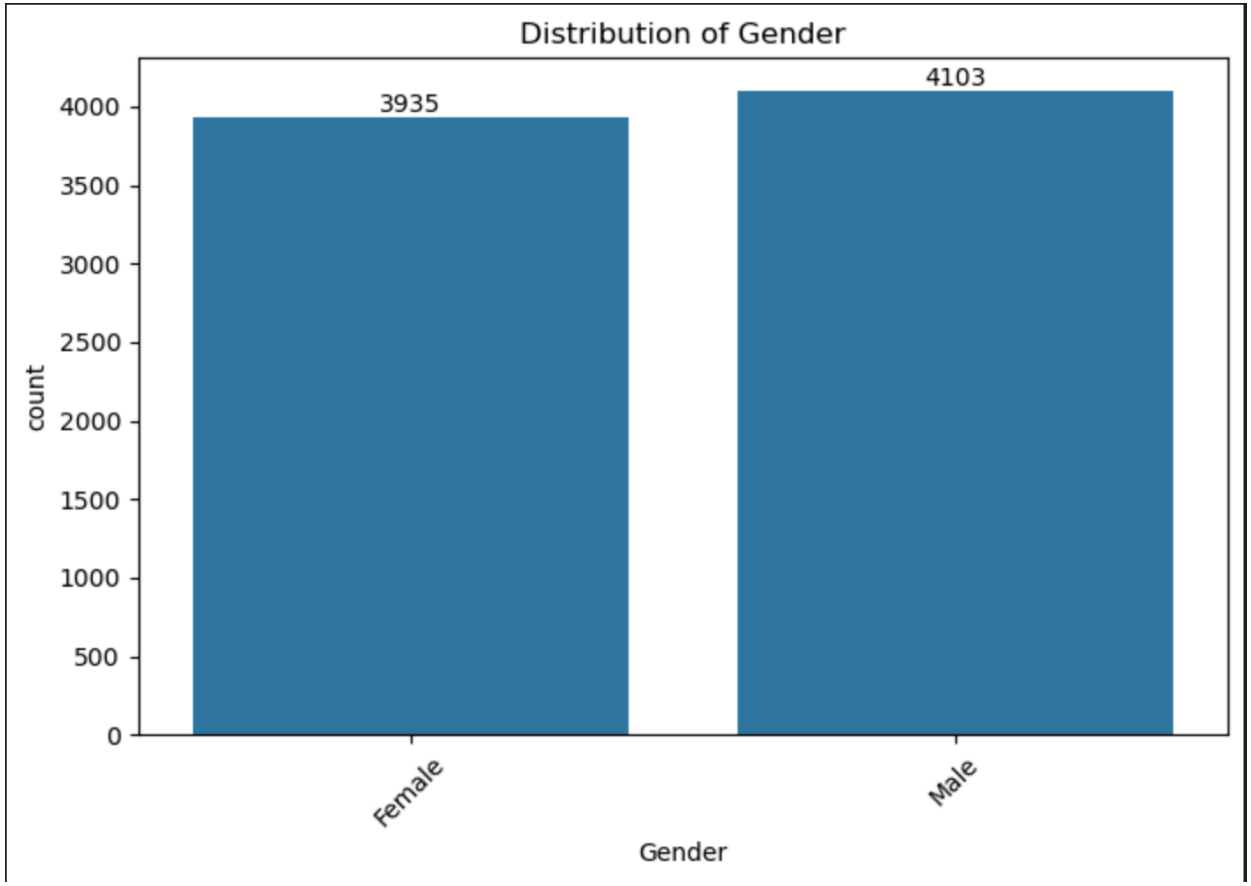
› Checking the type of ethnicity the patients belong. The majority appear to be a Caucasian.



# EDA

## UNIVARIATE ANALYSIS

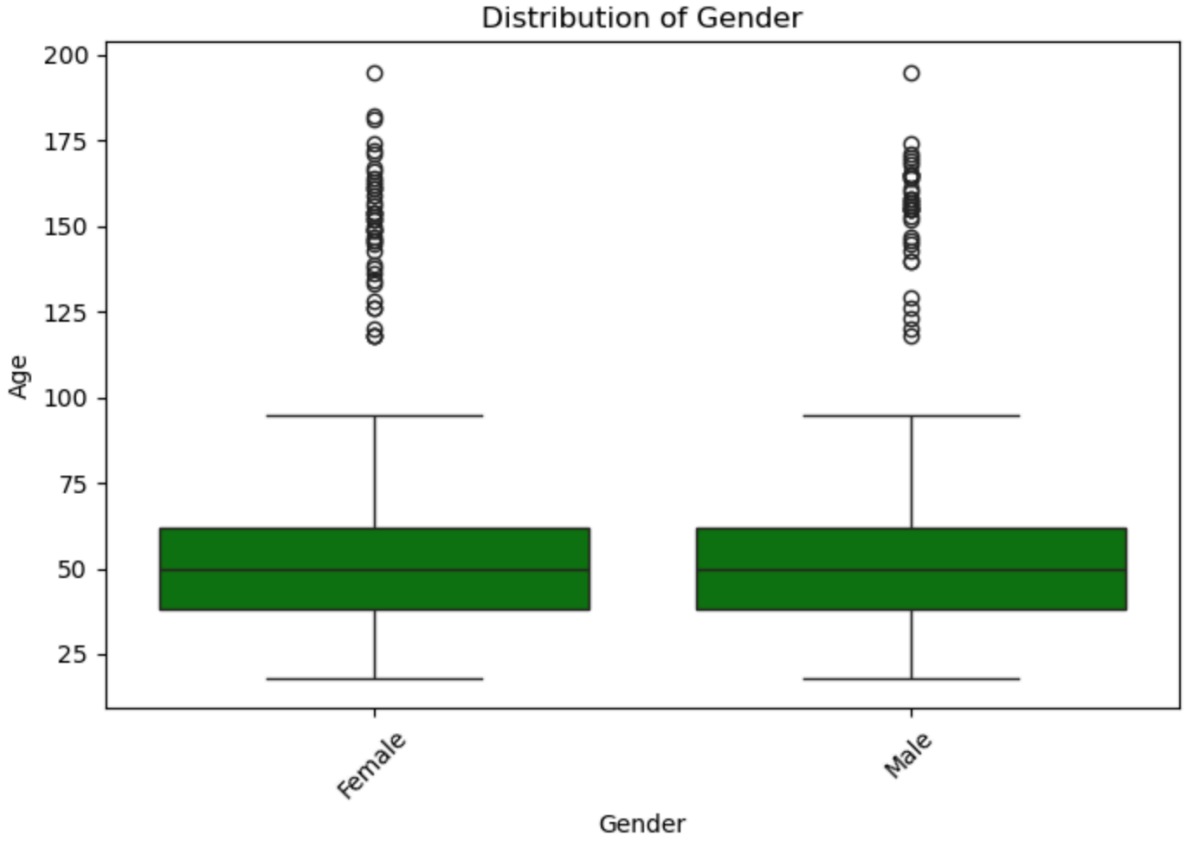
- Finally, the count of gender of patients - with Male being the most proportion



# EDA

## BI-VARIATE ANALYSIS

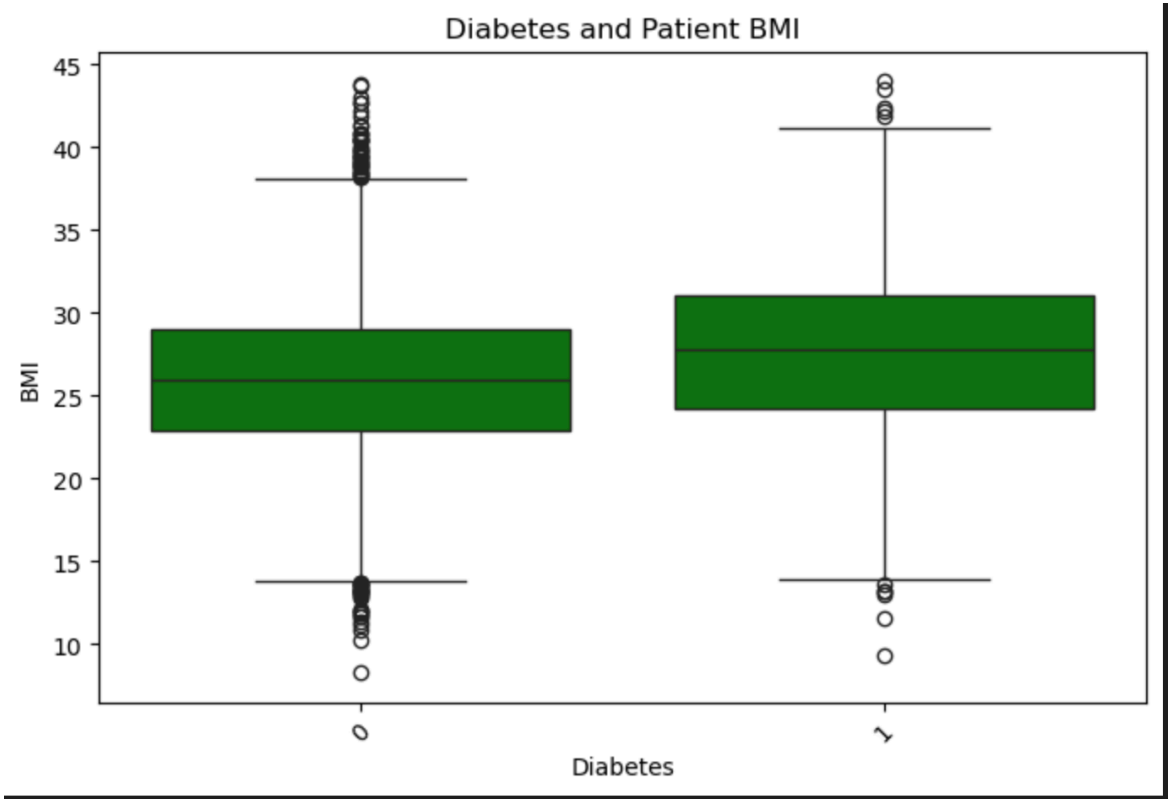
- Male and female have almost the same average age of patients around 50.
- The plot shows there are outliers with patients age above 125.



# EDA

## BI-VARIATE ANALYSIS

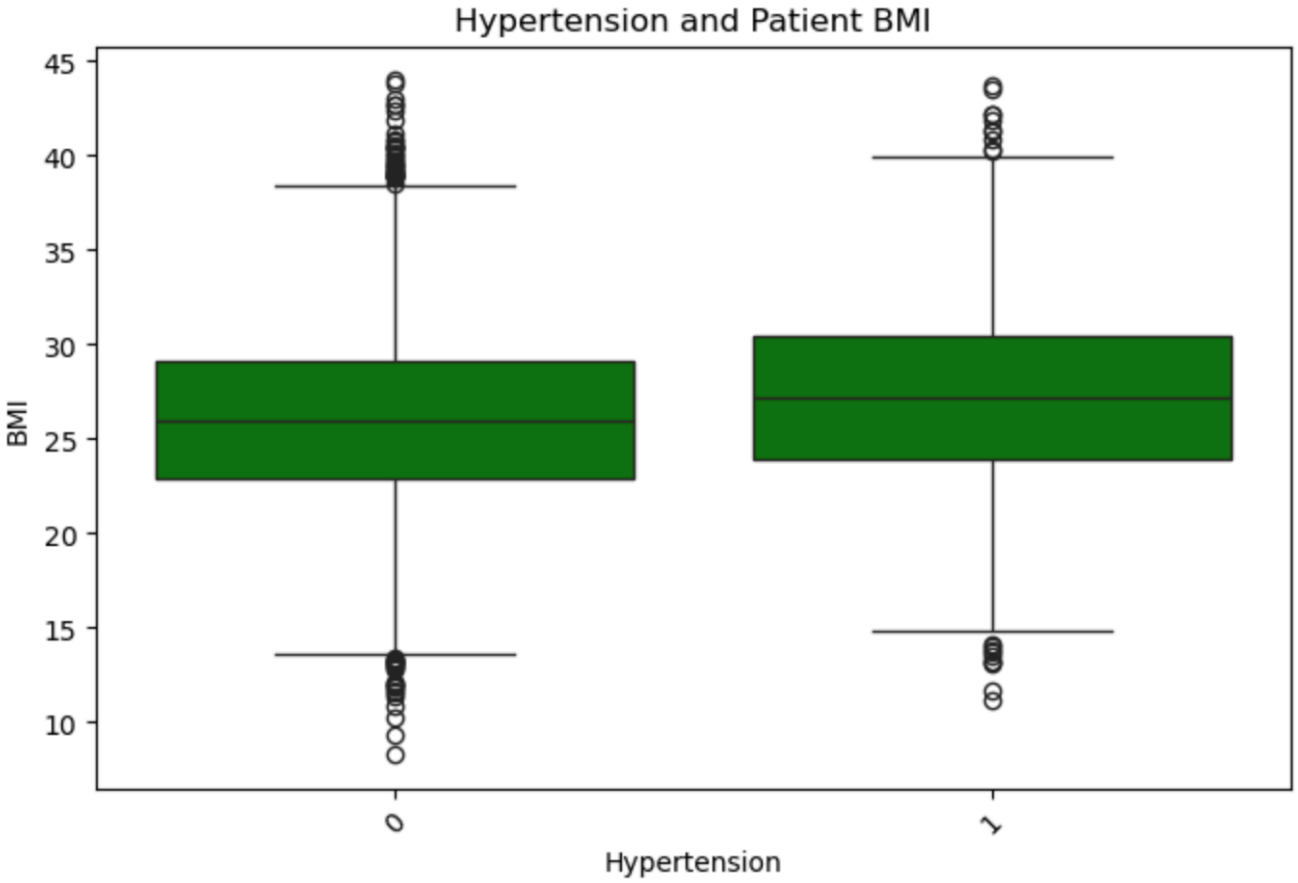
- Patients with Diabetes show a higher BMI (Body mass index)



# EDA

## BI-VARIATE ANALYSIS

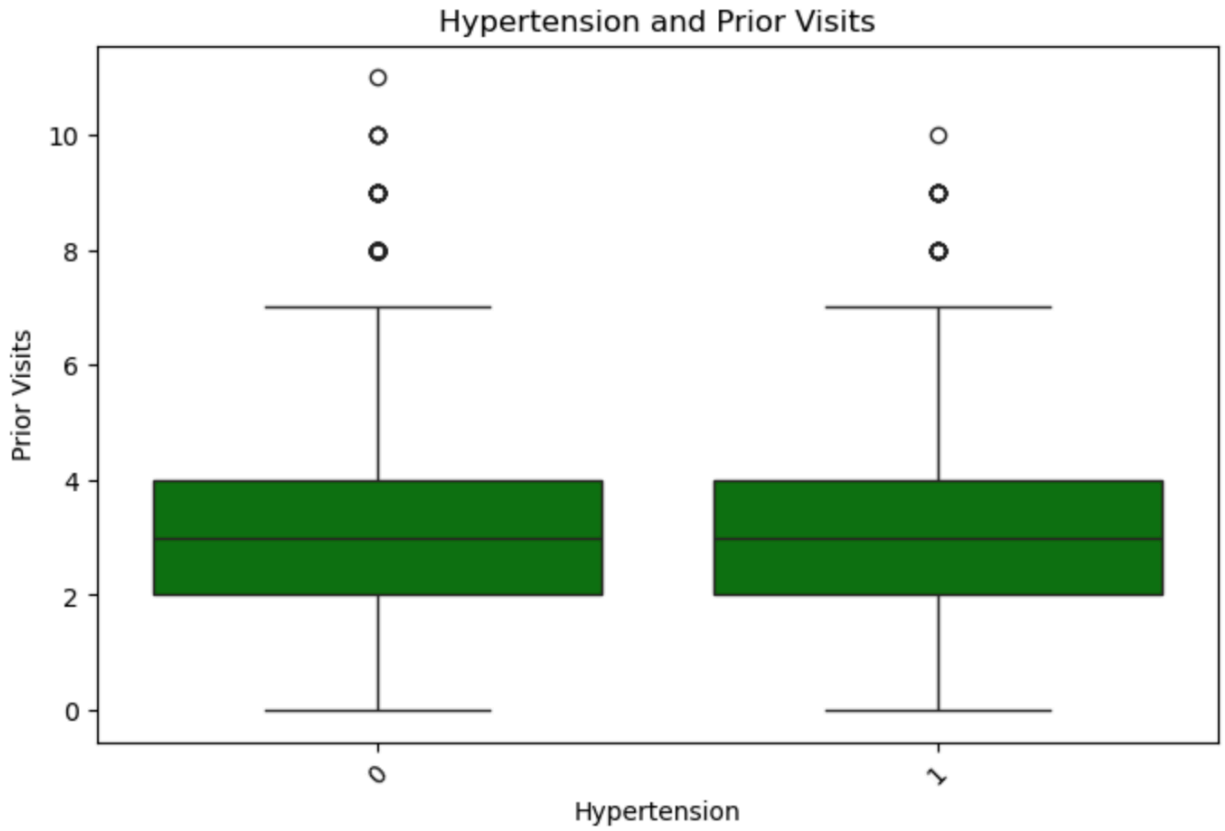
- Patients with hypertension show a slightly higher average BMI



# EDA

## BI-VARIATE ANALYSIS

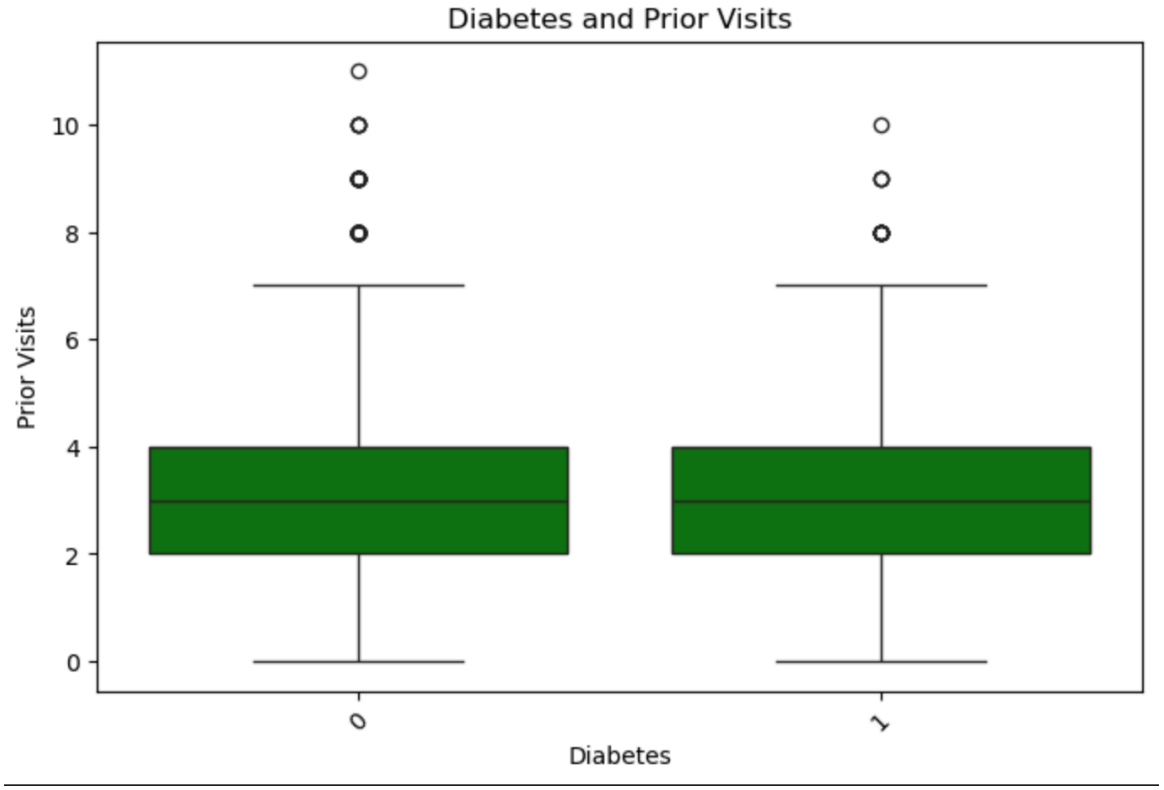
- Hypertension and prior visits doesn't seem to have a significant different in the number of prior visits based on hypertension



# EDA

## BI-VARIATE ANALYSIS

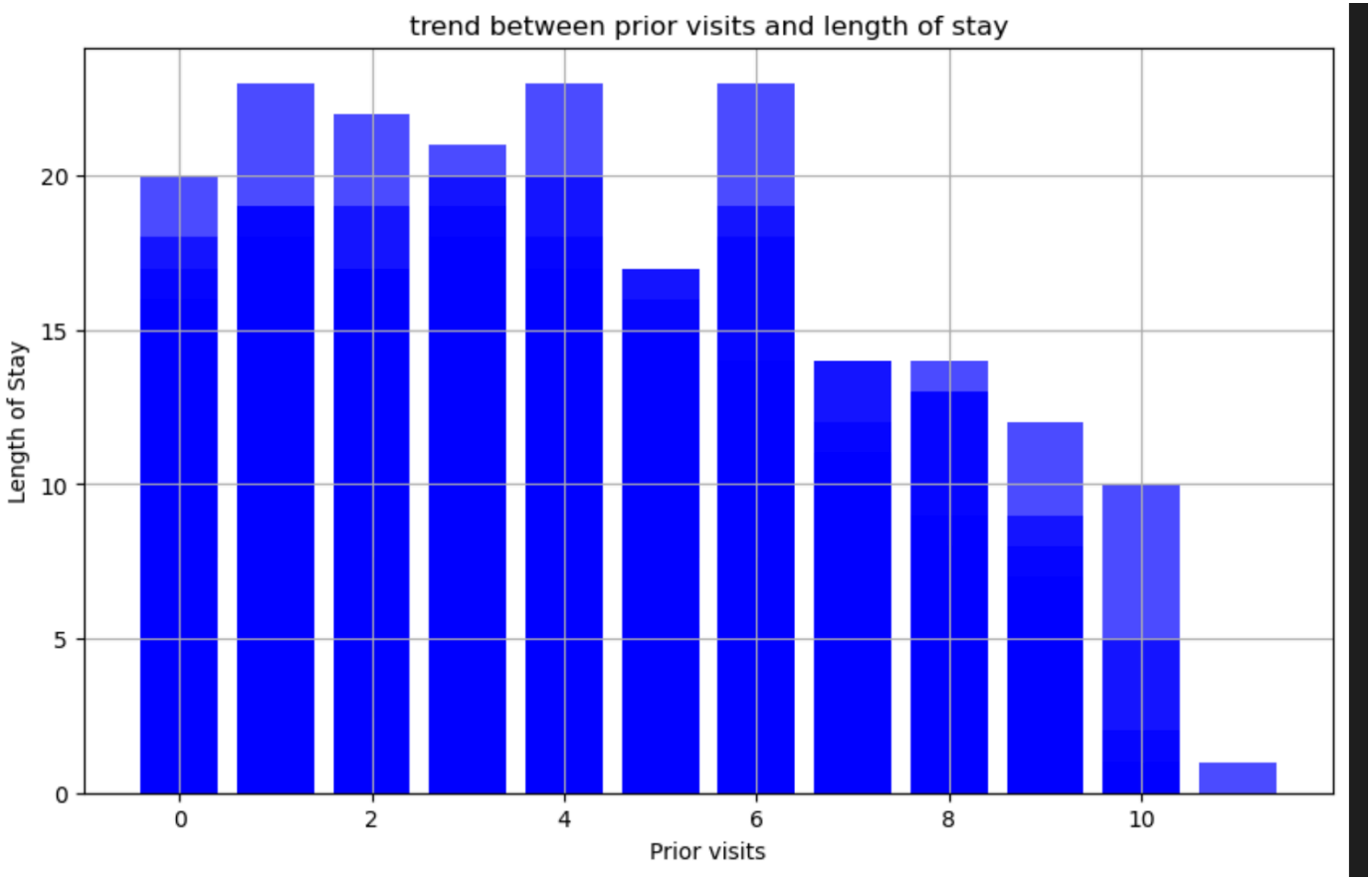
› Likewise, Diabetes and prior visits doesn't seem to have a significant different in the number of prior visits based on Diabetes



# EDA

## BI-VARIATE ANALYSIS

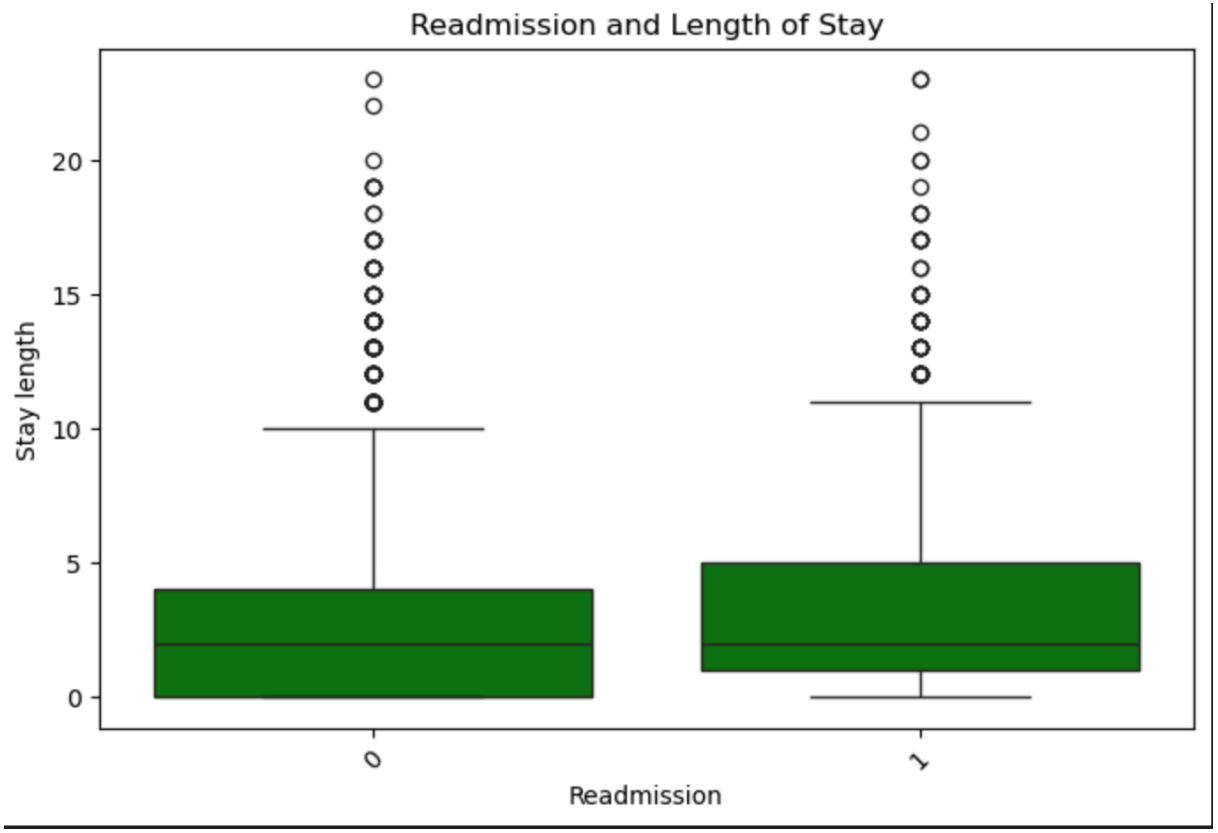
› The higher the number of prior visits, the lower the length of stay, so they don't seem to be aligned



# EDA

## BI-VARIATE ANALYSIS

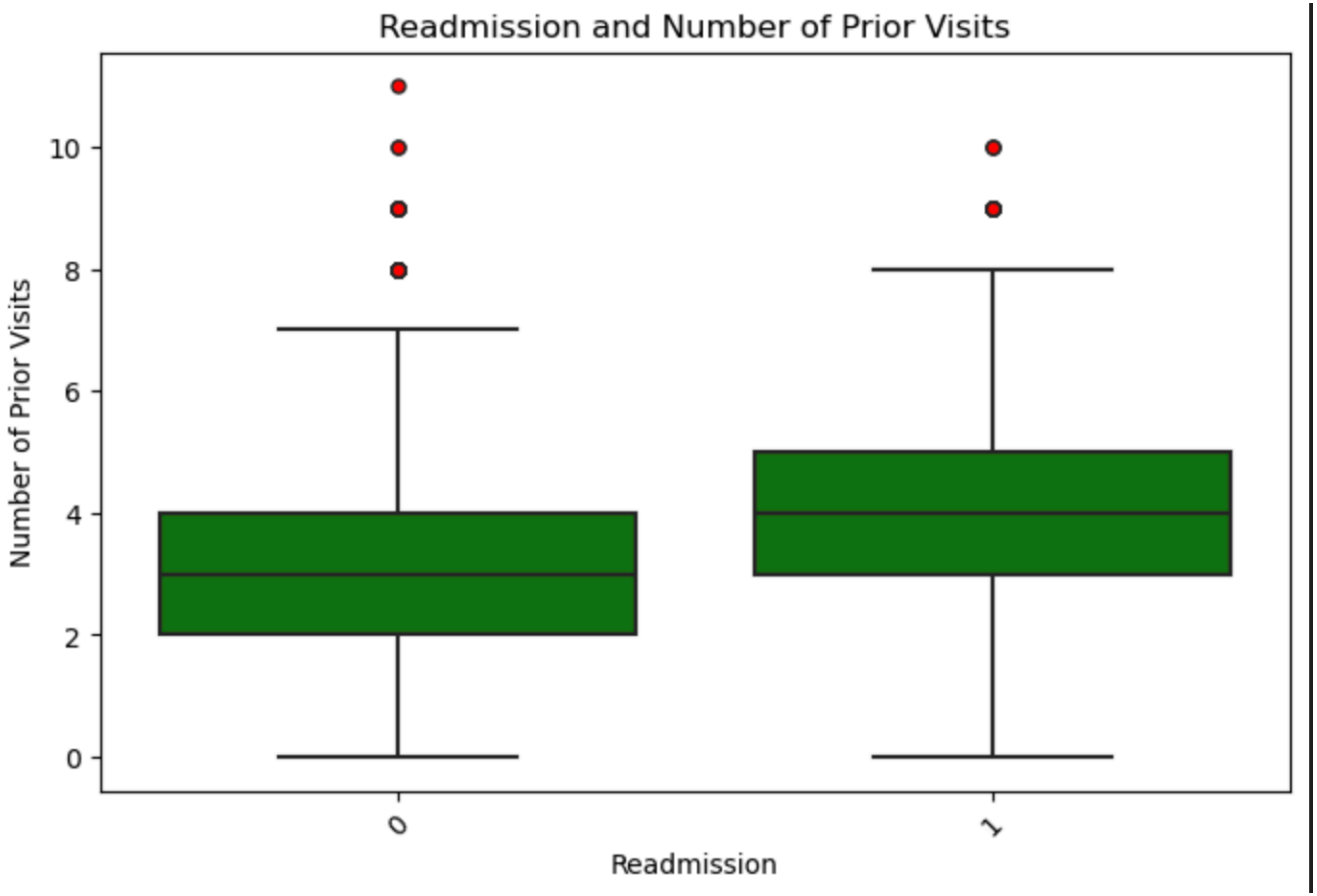
› Patients getting readmitted, show a higher maximum length of stay



# EDA

## BI-VARIATE ANALYSIS

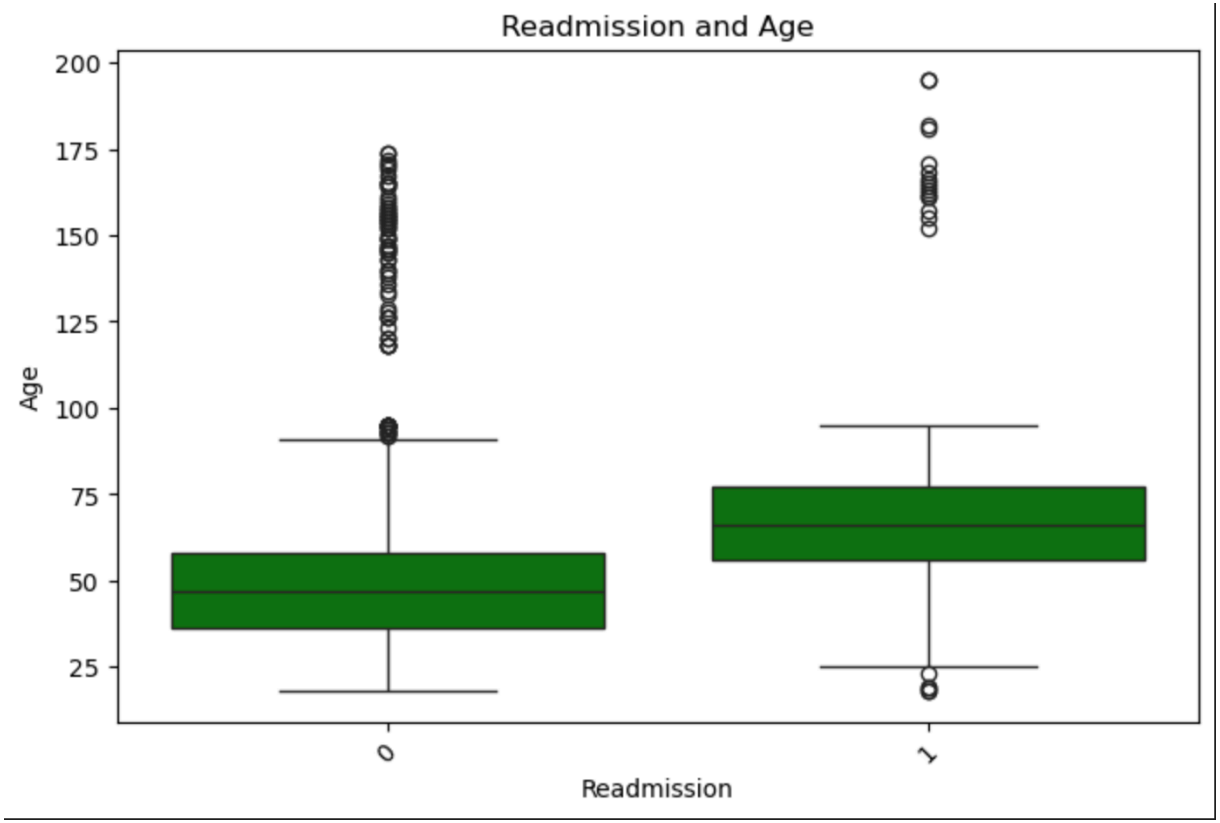
› Readmitted patients tend to have higher average number of prior visits.



# EDA

## BI-VARIATE ANALYSIS

› Form this plot, it is seen that older the patient, higher the chance of readmission



USC Viterbi School of Engineering

# EDA

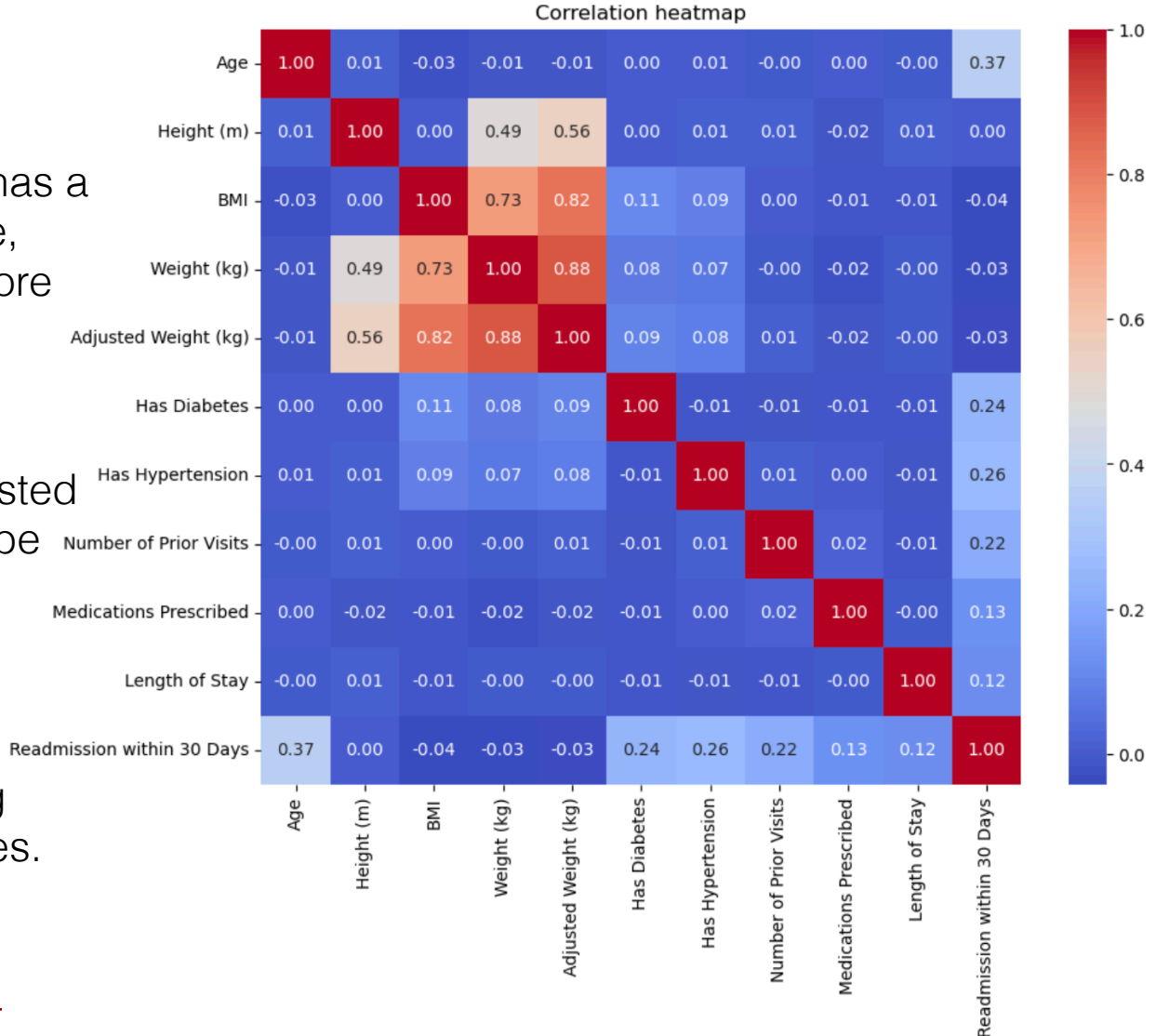
## BI-VARIATE ANALYSIS

› Correlation matrix - Shows that the readmission variable is closely related to

The readmission variable has a higher correlation with Age, Diabetes, Hypertension more compared to others.

The variables weight, adjusted weight and BMI seems to be related and causing **multicollinearity.**

We can consider removing either one or unrelated ones.



## › Feature Engineering Steps

**Dropped Irrelevant Columns:**

- Removed *PatientID*, *Hospital ID*, *Weight (kg)*, *Adjusted Weight (kg)*, and *Height (m)* as they were unrelated or redundant.

**Imputation:**

- Filled missing values in:
  - *Number of Prior Visits* and *Medications* Prescribed with the **median**.
  - *Exercise Frequency* and *Type of Treatment* with **default categorical values** (No exercise, No Treatment).

**Categorical Encoding:**

- One-hot encoded Diet Type, Type of Treatment, Ethnicity, and Exercise Frequency.

## › Feature Engineering Steps

**Binary Mapping:**

- Converted Gender to binary (Male: 1, Female: 0).
- Converted Smoker to binary (True: 1, False: 0).

**Target Encoding:**

- Extracted Readmission within 30 Days as the target variable (y).

**Data Resampling:**

- Balanced the training data using **SMOTE** to address class imbalance.

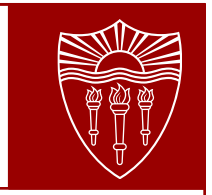
**Feature Scaling:**

- Applied **StandardScaler** to normalize features in both training and testing datasets.

USC Viterbi School of Engineering

# MODEL PIPELINE

## TRAINING PIPELINE - VERTEX AI



› Data preprocessing and feature engineering

Vertex_AI_Pipeline_Readmission.ipynb ☆

File Edit View Insert Runtime Tools Help All changes saved

Code + Text

```
from google.colab import auth
auth.authenticate_user()

from kfp.v2 import compiler
from kfp.v2.dsl import pipeline, component
from kfp.v2.dsl import InputPath, OutputPath, Input, Output

from ast import Str
import pandas as pd
import numpy as np
from imblearn.over_sampling import SMOTE
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import accuracy_score, classification_report
from google.cloud import aiplatform
from kfp.v2 import compiler
from kfp.v2.dsl import pipeline


# Initialize AI platform
project_id = 'long-state-444907-d0'
location = 'us-central1'
aiplatform.init(project=project_id, location=location)

# Component 1: Data Preprocessing

@component(packages_to_install=["pandas", "numpy","fsspec","gcsfs"])
def Data_Preparation_and_processing(
    input_dataset_path: str,
    prepared_dataset_path: OutputPath()
):
    import pandas as pd
    import numpy as np
    data = pd.read_csv(input_dataset_path)

    # Error handling for missing critical columns
```

```
# Error handling for missing critical columns
required_columns = ['PatientID', 'Number of Prior Visits', 'Medications Prescribed',
                    'Exercise Frequency', 'Type of Treatment', 'Weight (kg)',
                    'Adjusted Weight (kg)', 'Gender', 'Smoker', 'Hospital ID',
                    'Diet Type', 'Ethnicity', 'Height (m)']
missing_columns = [col for col in required_columns if col not in data.columns]
if missing_columns:
    raise ValueError(f"Missing columns in dataset: {missing_columns}")

# Fill missing values with median or default values
data['Number of Prior Visits'].fillna(data['Number of Prior Visits'].median(), inplace=True)
data['Medications Prescribed'].fillna(data['Medications Prescribed'].median(), inplace=True)
data['Exercise Frequency'].fillna('No exercise', inplace=True)
data['Type of Treatment'].fillna('No Treatment', inplace=True)

# Drop unnecessary or redundant columns
drop_columns = ['PatientID', 'Weight (kg)', 'Adjusted Weight (kg)', 'Hospital ID', 'Height (m)']
data.drop(columns=drop_columns, inplace=True)

# Encode binary categorical features
binary_mappings = {'Gender': {'Male': 1, 'Female': 0}, 'Smoker': {True: 1, False: 0}}
for col, mapping in binary_mappings.items():
    if col in data.columns:
        data[col] = data[col].map(mapping)

# One-hot encode categorical features
categorical_columns = ['Diet Type', 'Type of Treatment', 'Ethnicity', 'Exercise Frequency']
data = pd.get_dummies(data, columns=categorical_columns, drop_first=True)

# Save the processed data to output path
data.to_csv(prepared_dataset_path, index=False)
```

USC Viterbi School of Engineering

# MODEL PIPELINE

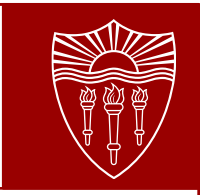
## MODEL TRAINING

› Model training trains the model and saves the path

```python
# Component 5: Model Training
@component(packages_to_install=["scikit-learn", "pandas", "numpy", "joblib", "fsspec", "gcsfs"])
def Model_Training( training_dataset_path: InputPath(),  # Input: Processed training dataset
    trained_model_path: OutputPath("Artifact")):

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


    df_train = pd.read_csv(training_dataset_path)


    X_train = df_train.drop('Readmission within 30 Days', axis=1)
    y_train = df_train['Readmission within 30 Days']

    # Train a logistic regression model
    model = LogisticRegression(random_state=42)
    model.fit(X_train, y_train)

    # Save the trained model
    import joblib
    joblib.dump(model, trained_model_path)
    print(f"Model trained and saved to {trained_model_path}")
```

# MODEL PIPELINE

## DATA SPLIT AND SMOTE

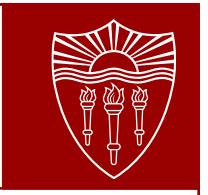


› Data split and resampling function splits the data and performs the SMOTE resampling to fix the class imbalance issue

```python
def Data_Split_and_Resampling(prepared_dataset_path: InputPath(),
    training_dataset_path: OutputPath(),
    validation_dataset_path: OutputPath(),
    scaler_path: OutputPath("Artifact")):

    import pandas as pd
    from sklearn.model_selection import train_test_split
    from sklearn.preprocessing import StandardScaler
    from imblearn.over_sampling import SMOTE
    import joblib
    data = pd.read_csv(prepared_dataset_path)

    # Split into features and target
    data = data.dropna(subset=['Readmission within 30 Days'])

    X = data.drop('Readmission within 30 Days', axis=1, errors='ignore')  # Features
    y = data['Readmission within 30 Days']
    from sklearn.model_selection import train_test_split

    # Split into train and test sets
    X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)

    smote = SMOTE(random_state=42)

    try:
        X_train_resampled, y_train_resampled = smote.fit_resample(X_train, y_train)
    except ValueError as e:
        raise ValueError(f"SMOTE failed due to invalid input data: {e}")

    scaler = StandardScaler()

    try:
        X_train_resampled = scaler.fit_transform(X_train_resampled)
        X_test = scaler.transform(X_test)
    except ValueError as e:
        raise ValueError(f"Scaling failed due to invalid input data: {e}")
```

USC Viterbi School of Engineering

# MODEL PIPELINE



› Model Evaluation - evaluates the model on several classifcation metrics

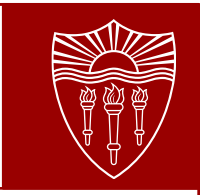
```python
@component(packages_to_install=["pandas", "scikit-learn", "joblib", "fsspec", "gcsfs"])
def Evaluation(
    validation_dataset_path: InputPath(),
    trained_model_path: InputPath("Artifact"),
    metrics: Output[Metrics]
):

    import pandas as pd
    from sklearn.linear_model import LogisticRegression
    import joblib
    from sklearn.metrics import (
        accuracy_score,
        f1_score,
        auc,
        precision_recall_curve,
        confusion_matrix
    )
    import joblib


    val_df = pd.read_csv(validation_dataset_path)

    print(val_df.head(5))

    X_val = val_df.drop('Readmission within 30 Days', axis=1)
    y_val = val_df['Readmission within 30 Days']

    print(y_val.head(5))

    model = joblib.load(trained_model_path)

    y_pred = model.predict(X_val)
    y_pred_proba = model.predict_proba(X_val)[:, 1]

    accuracy = accuracy_score(y_val, y_pred)
```

```python
accuracy = accuracy_score(y_val, y_pred)
f1 = f1_score(y_val, y_pred)

# Calculate area under precision-recall curve (AU-PRC)
precision_curve, recall_curve, _ = precision_recall_curve(y_val, y_pred_proba)
auprc = auc(recall_curve, precision_curve)

# Confusion matrix for calculating TP, TN, FP, FN
tn, fp, fn, tp = confusion_matrix(y_val, y_pred).ravel()

# Log required metrics
   # Convert metrics to Python native types
accuracy = float(accuracy)
f1 = float(f1)
auprc = float(auprc)
tn, fp, fn, tp = map(int, [tn, fp, fn, tp])  # Convert to native int

# Log required metrics
metrics.log_metric("accuracy", accuracy)
metrics.log_metric("f1_score", f1)
metrics.log_metric("area_under_precision_recall_curve", auprc)
metrics.log_metric("true_positives", tp)
metrics.log_metric("true_negatives", tn)
metrics.log_metric("false_positives", fp)
metrics.log_metric("false_negatives", fn)


# Optional: Print evaluation metrics for debugging
print(f"Evaluation Metrics:\nAccuracy: {accuracy}\nF1 Score: {f1}\nArea Under Precision-Recall Curve (AU-PRC): {auprc}")
print(f"Confusion Matrix: TP={tp}, TN={tn}, FP={fp}, FN={fn}")
```

USC Viterbi School of Engineering

# PIPELINE COMPILER

## MODEL TRAINING

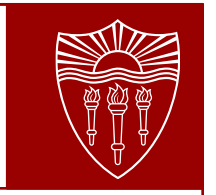


› Compiler sets the parameters and paths for running the pipeline

```
from kfp.v2 import compiler
# Compile the pipeline
compiler.Compiler().compile(
    pipeline_func=healthcare_readmissions_pipeline,
    package_path='healthcare_readmissions_pipeline.json'
)


pipeline_job = aiplatform.PipelineJob(
    display_name='healthcare-readmissions-training-pipeline',
    template_path='healthcare_readmissions_pipeline.json',
    pipeline_root='gs://healthcare_readmissions_30days',
    parameter_values={
        'input_dataset_path': 'gs://healthcare_readmissions_30days/healthcare_readmissions_dataset_train.csv'
    },


    enable_caching=True
)

pipeline_job.run(sync=False)
```

USC Viterbi School of Engineering

# TRAINING PIPELINE



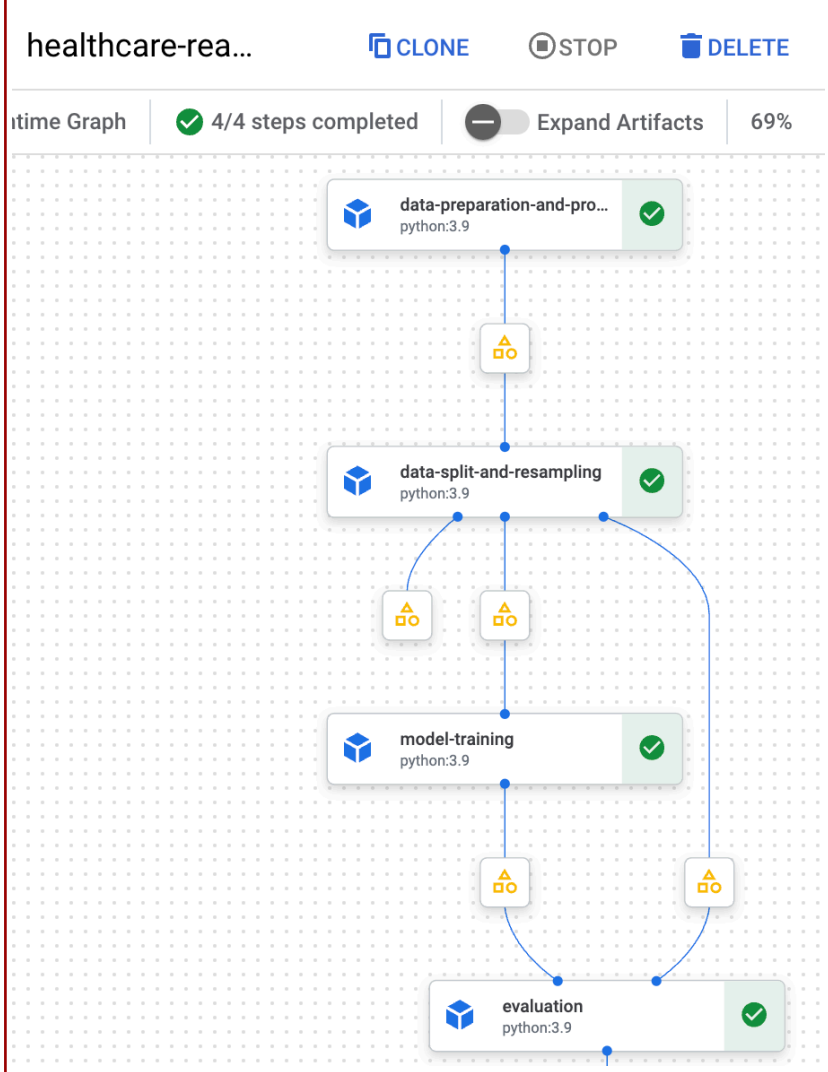
› The pipeline calls the related functions for the training.

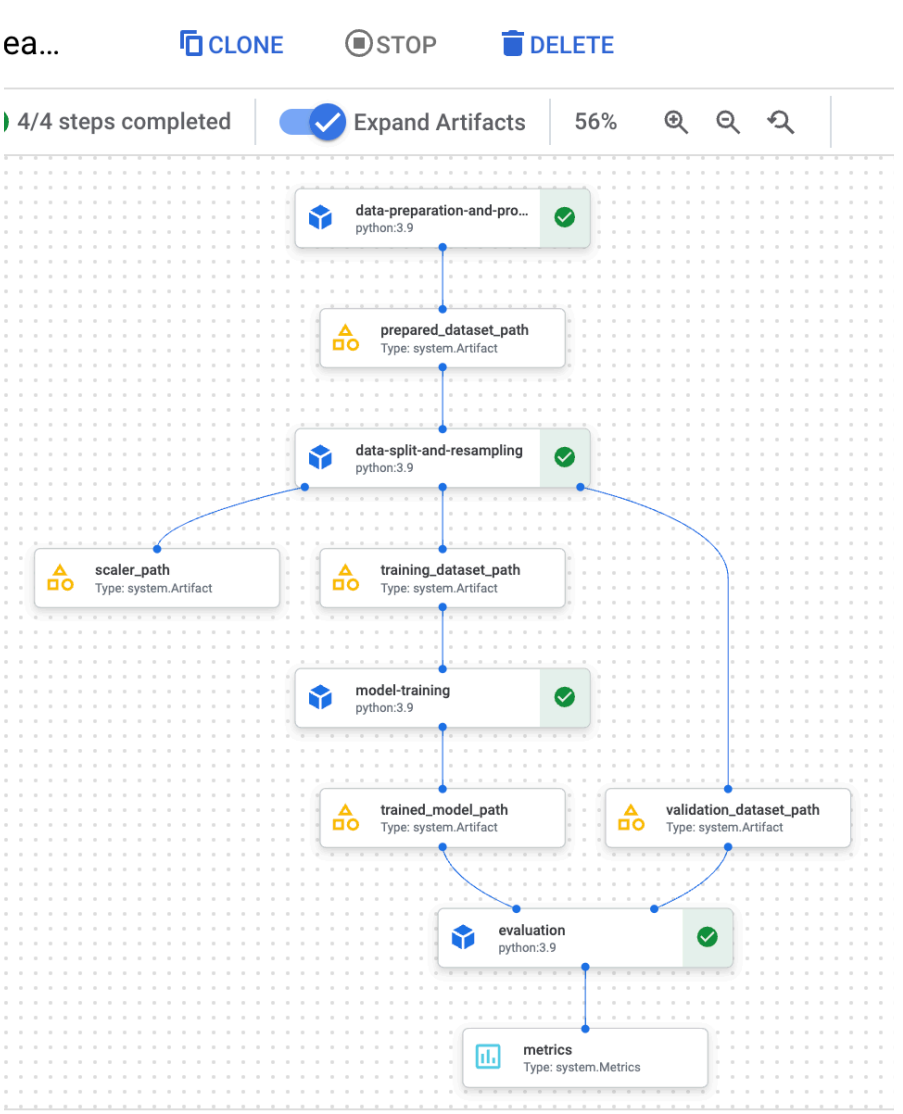
```python
# Define the pipeline
@pipeline(name="healthcare-readmissions-training-pipeline2")
def healthcare_readmissions_pipeline(input_dataset_path: str):
    # Preprocessing the data
    prepared_data = Data_Preparation_and_processing(input_dataset_path=input_dataset_path)

    # Step 2: Split, encode, and preprocess
    processed_data = Data_Split_and_Resampling(
        prepared_dataset_path=prepared_data.outputs['prepared_dataset_path']
    )

    # Step 3: Train the model/
    trained_model = Model_Training(
        training_dataset_path=processed_data.outputs['training_dataset_path']
    )

    # Step 4: Evaluate the model
    Evaluation(
        validation_dataset_path=processed_data.outputs['validation_dataset_path'],
        trained_model_path=trained_model.outputs['trained_model_path']
    )
```

USC Viterbi School of Engineering

# MODEL VERTEX PIPELINE

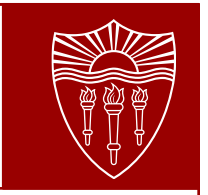
## TRAINING PIPELINE



USC Viterbi School of Engineering

# MODEL TRAINING PERFORMANCE



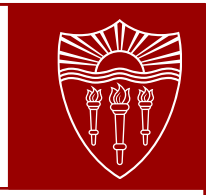
- The training model performs with an
- Accuracy of **0.9359** and
- F1 score of **0.7867**.

```
Accuracy: 0.9359
Classification Report:
              precision    recall  f1-score   support

           0       0.96      0.97      0.96      1357
           1       0.82      0.76      0.79       251

    accuracy                           0.94      1608
   macro avg       0.89      0.86      0.87      1608
weighted avg       0.93      0.94      0.93      1608

F1 Score: 0.7867
```

USC Viterbi School of Engineering

# VALIDATION PIPELINE CODE

## VERTEX AI



› Data preparation component

```python
from kfp.v2.dsl import pipeline, component, InputPath, OutputPath

# Step 1: Initial Data Preparation (Reuse the same component as in training)
@component(packages_to_install=["pandas", "numpy", "fsspec", "gcsfs"])
def Testing_Data_Preperation(
    input_dataset_path: str,  # Input: Raw dataset path
    prepared_dataset_path: OutputPath("Dataset")  # Output: Cleaned and prepared dataset pa
):
    import pandas as pd
    import numpy as np
    data = pd.read_csv(input_dataset_path)

    # Error handling for missing critical columns
    required_columns = [ 'PatientID','Number of Prior Visits', 'Medications Prescribed',
                        'Exercise Frequency', 'Type of Treatment', 'Weight (kg)',
                        'Adjusted Weight (kg)', 'Gender', 'Smoker', 'Hospital ID',
                        'Diet Type', 'Ethnicity', 'Height (m)']
    missing_columns = [col for col in required_columns if col not in data.columns]
    if missing_columns:
        raise ValueError(f"Missing columns in dataset: {missing_columns}")

    # Fill missing values with median or default values
    data['Number of Prior Visits'].fillna(data['Number of Prior Visits'].median(), inplace=
    data['Medications Prescribed'].fillna(data['Medications Prescribed'].median(), inplace=
    data['Exercise Frequency'].fillna('No exercise', inplace=True)
    data['Type of Treatment'].fillna('No Treatment', inplace=True)

    # Drop unnecessary or redundant columns
    drop_columns = ['Weight (kg)', 'Adjusted Weight (kg)', 'Hospital ID', 'Height (m)']
    data.drop(columns=drop_columns, inplace=True)

    # Encode binary categorical features
    binary_mappings = {'Gender': {'Male': 1, 'Female': 0}, 'Smoker': {True: 1, False: 0}}
    for col, mapping in binary_mappings.items():
        if col in data.columns:
            data[col] = data[col].map(mapping)
```

```python
    # Fill missing values with median or default values
    data['Number of Prior Visits'].fillna(data['Number of Prior Visits'].median(), inplace=True)
    data['Medications Prescribed'].fillna(data['Medications Prescribed'].median(), inplace=True)
    data['Exercise Frequency'].fillna('No exercise', inplace=True)
    data['Type of Treatment'].fillna('No Treatment', inplace=True)

    # Drop unnecessary or redundant columns
    drop_columns = ['Weight (kg)', 'Adjusted Weight (kg)', 'Hospital ID', 'Height (m)']
    data.drop(columns=drop_columns, inplace=True)

    # Encode binary categorical features
    binary_mappings = {'Gender': {'Male': 1, 'Female': 0}, 'Smoker': {True: 1, False: 0}}
    for col, mapping in binary_mappings.items():
        if col in data.columns:
            data[col] = data[col].map(mapping)

    # One-hot encode categorical features
    categorical_columns = ['Diet Type', 'Type of Treatment', 'Ethnicity', 'Exercise Frequency']
    data = pd.get_dummies(data, columns=categorical_columns, drop_first=True)

    # Save the processed data to output path
    data.to_csv(prepared_dataset_path, index=False)
```

# VALIDATION PIPELINE

## VERTEX AI

```
# Step 2: Perform validation (Normalize the test dataset using the scaler from training pipeline)
@component(packages_to_install=["pandas", "numpy", "joblib", "gcsfs", "scikit-learn"])
def Scaling_data(
    validation_dataset_path: InputPath("Dataset"),  # Input: Raw test dataset
    scaler_path: str,                               # Input: Path to saved scaler
    normalized_validation_dataset_path: OutputPath("Dataset")  # Output: Processed test dataset
):
    import pandas as pd
    import joblib
    import gcsfs

    # Load validation dataset
    df_validate = pd.read_csv(validation_dataset_path)

    # Extract PatientID for later use
    if 'PatientID' in df_validate.columns:
        patient_ids = df_validate['PatientID']
    else:
        raise ValueError("PatientID column is missing in the input dataset.")

    # Drop PatientID temporarily for transformations
    df_validate = df_validate.drop(columns=['PatientID'])

    # Load the scaler
    fs = gcsfs.GCSFileSystem()
    with fs.open(scaler_path, 'rb') as f:
        scaler = joblib.load(f)


    # Apply scaling to numerical features
    df_validate = scaler.transform(df_validate)


    df_validate = pd.DataFrame(df_validate)

    # Reattach the PatientID column at the start
    df_validate.insert(0, 'PatientID', patient_ids)
```

USC Viterbi School of Engineering

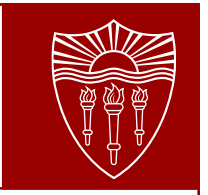
# VALIDATION PIPELINE

## VERTEX AI

```python
# Step 3: Perform Predictions
@component(packages_to_install=["pandas", "joblib", "gcsfs", "scikit-learn", "xgboost"])
def Perform_Predictions(
    dataset_for_prediction_path: InputPath("Dataset"),
    model_path: str,
    predictions_path: OutputPath("Dataset"),
    gcs_dump_path: str
):
    import pandas as pd
    import joblib
    import gcsfs

    # Load the dataset for predictions
    df = pd.read_csv(dataset_for_prediction_path)

    # Extract the PatientID column for output
    if 'PatientID' in df.columns:
        patient_ids = df['PatientID']
        features = df.drop(columns=['PatientID'])
    else:
        raise ValueError("PatientID column is missing in the input dataset.")

    # Align columns to match the model's training features
    fs = gcsfs.GCSFileSystem()
    with fs.open(model_path, 'rb') as f:
        model = joblib.load(f)
    expected_columns = model.feature_names_in_
    features = features.reindex(columns=expected_columns, fill_value=0)

    # Dump the testing data to GCS for review
    with fs.open(f"{gcs_dump_path}/logistic_test_data.csv", 'w') as f:
        features.to_csv(f, index=False)
    print("Testing DataFrame dumped to GCS.")


     # Print a preview of the data
    print("Testing DataFrame (features) Preview:")
```

# VALIDATION PIPELINE

## VERTEX AI



› Pipeline

```python
# Step 4: Define the Testing Pipeline
@pipeline(name="healthcare-readmissions-testing-inference-pipeline")
def healthcare_testing_inference_pipeline(
    input_test_dataset_path: str,
    scaler_uri: str,
    model_uri: str
):
    # Step 1: Prepare the test dataset
    prepared_test_data = Testing_Data_Preperation(
        input_dataset_path=input_test_dataset_path
    )

    # Step 2: Normalize the test dataset
    normalized_test_data = Scaling_data(
        validation_dataset_path=prepared_test_data.outputs["prepared_dataset_path"],
        scaler_path=scaler_uri
    )

    # Step 3: Perform predictions
    Perform_Predictions(
        dataset_for_prediction_path=normalized_test_data.outputs["normalized_validation_dataset_path"],
        model_path=model_uri,
        gcs_dump_path="gs://healthcare_readmissions_30days/debugdatasets"
    )
```

# VALIDATION PIPELINE

## VERTEX AI

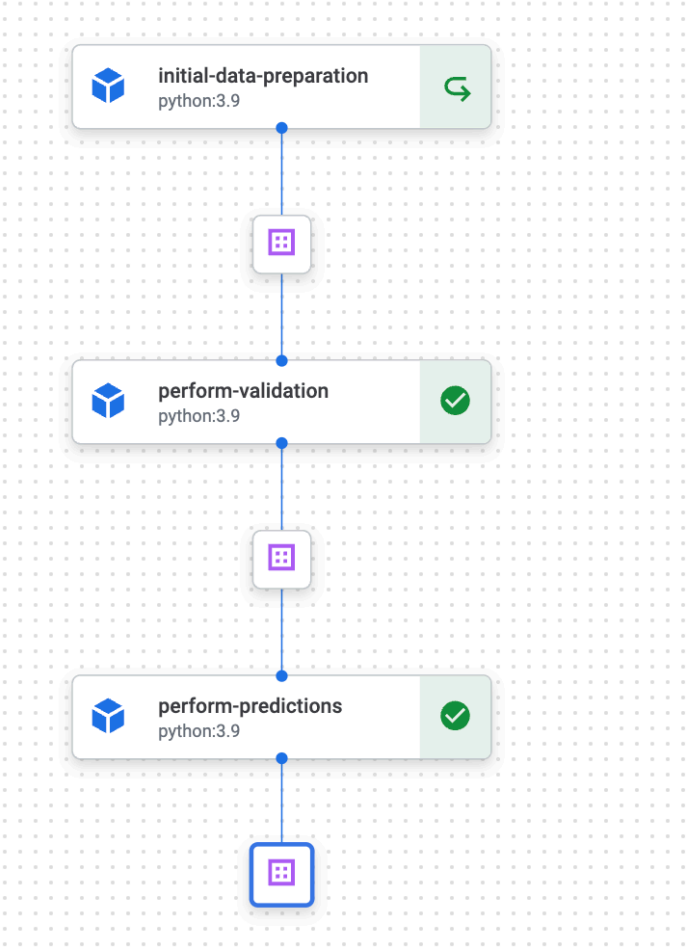
- Compiler

```python
# Compile and Run the Pipeline
from kfp.v2 import compiler
from google.cloud import aiplatform

compiler.Compiler().compile(
    pipeline_func=healthcare_testing_inference_pipeline,
    package_path="healthcare_testing_inference_pipeline.json"
)

pipeline_job = aiplatform.PipelineJob(
    display_name="healthcare-readmissions-testing-inference-pipeline",
    template_path="healthcare_testing_inference_pipeline.json",
    pipeline_root="gs://healthcare_readmissions_30days",
    parameter_values={
        "input_test_dataset_path": "gs://healthcare_readmissions_30days/healthcare_readmissions_dataset_test.csv",
        "scaler_uri": "gs://healthcare_readmissions_30days/237661141851/healthcare-readmissions-training-pipeline2-20241218200924/data-split-and-resampling_135
        "model_uri": "gs://healthcare_readmissions_30days/237661141851/healthcare-readmissions-training-pipeline2-20241218200924/model-training_474676215577103
    },
    enable_caching=True
)

pipeline_job.run()
```

# MODEL PIPELINE

## TESTING VALIDATION PIPELINE

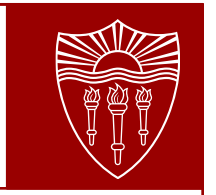
- validation inference pipeline is created and we will test the test dataset on the trained model



USC Viterbi School of Engineering

# MODEL PIPELINE

## MODEL PERFORMANCE - INFERENCE



- The model performance with the test data has an accuracy F1 Score of **0.7814**

- Snapshot of Prediction table,
- Column 1 : Patient Id
- Column 2: Readmission prediction

**predictions.csv**

```
1  PatientID,Readmission Prediction
2  1000001,0
3  1000007,0
4  1000011,0
5  1000012,0
6  1000033,0
7  1000034,0
8  1000035,0
9  1000043,0
10 1000050,0
11 1000052,0
12 1000053,0
13 1000055,1
14 1000062,0
15 1000067,0
16 1000069,0
17 1000073,0
18 1000080,0
19 1000088,0
20 1000104,1
21 1000112,1
22 1000113,0
23 1000115,0
24 1000116,0
25 1000118,0
26 1000120,0
27 1000121,0
28 1000126,0
```

# SUMMARY

## MODEL PERFORMANCE AND CONCLUSION

- The model overall performs with an accuracy of 0.78 tested with the test data.
- Logistic Regression shows a prediction of 0.78 with both test and train, which can be considered a not so bad score considering the uncertainty involved in patient readmission prediction in healthcare domain
- Features mostly contributing to the decision includes - Age, Hypertension and Diabetes.