Data Geeks

Final Project

Cardiovasvular Disease Prediction

Background

Sehat Selalu, a Health Tech company specializing in cardiovascular health, is planning to develop an innovative application aimed at enabling the public to conduct early screening tests for **cardiovascular diseases** (**CVD**).

In pursuit of this project, Sehat Selalu has partnered with Data Geeks, a group of Data Scientists renowned for their expertise in building machine learning models.

Dataset Description

Dataset is Obtain from Kaggle consisted of 70000 rows and 12 features. The data is balanced and already encoded for detailed information check here

Dataset Description

Features:

Age | Objective Feature | age | int (days)

Height | Objective Feature | height | int (cm) |

Weight | Objective Feature | weight | float (kg) |

Gender | Objective Feature | gender | categorical code | 0 Female: 1 Male

Systolic blood pressure | Examination Feature | ap_hi | int |

Diastolic blood pressure | Examination Feature | ap_lo | int |

Cholesterol | Examination Feature | cholesterol | 1: normal, 2: above normal, 3: well above normal |

Glucose | Examination Feature | gluc | 1: normal, 2: above normal, 3: well above normal |

Smoking | Subjective Feature | smoke | binary |

Alcohol intake | Subjective Feature | alco | binary |

Physical activity | Subjective Feature | active | binary |

Presence or absence of cardiovascular disease | Target Variable | cardio | binary |

Source Code

Data Science Project 4-6 Digital Skola + ChatGPT

Objectives

01. Building a machine learning model to predict CVD risk

02. Determine key factor of Cardiovascular Disease

03. Giving personal recommendations based on risk status of each person

Workflow

Data Cleaning

- Check Missing Value
- Drop irrelevant value
- Renaming few columns
- Outlier treatment

EDA

- Categorical Column
- Numerical Column
- Correlation

Feature Engineering

- Age Column
- BMI Column

Workflow

Modeling

- Feature Scaling
- Train Test Split
- Model Evaluation

Modeling II

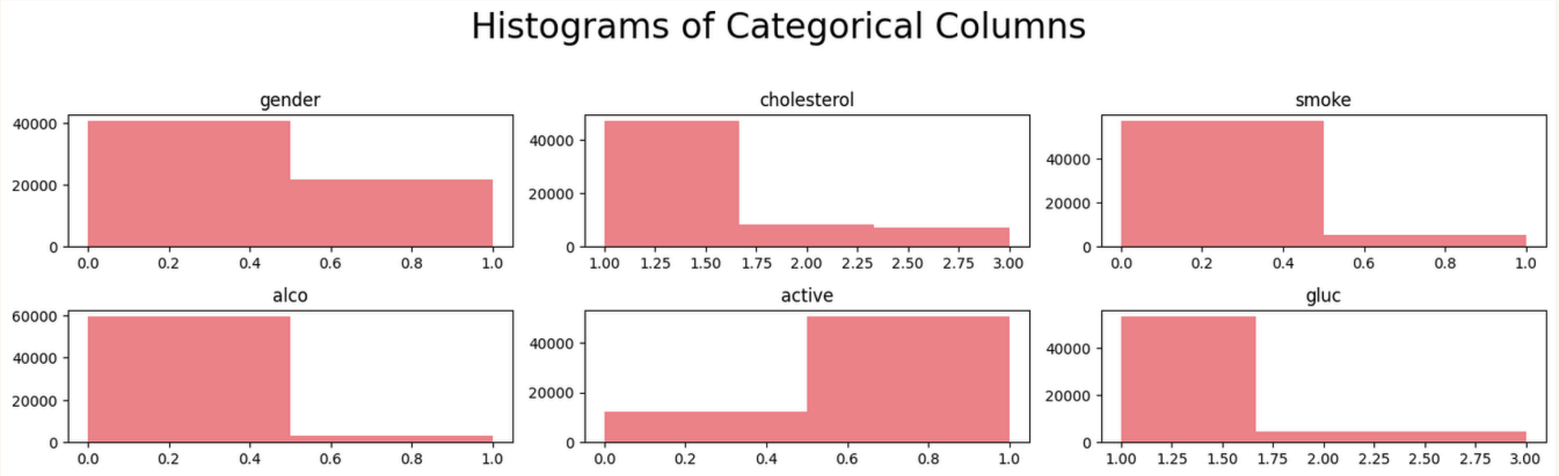
- GridSearch CV
- Select Model
- Feature Importance
- Repredict Cleaned Data

Model Deployment

Deploy Model using Streamlit

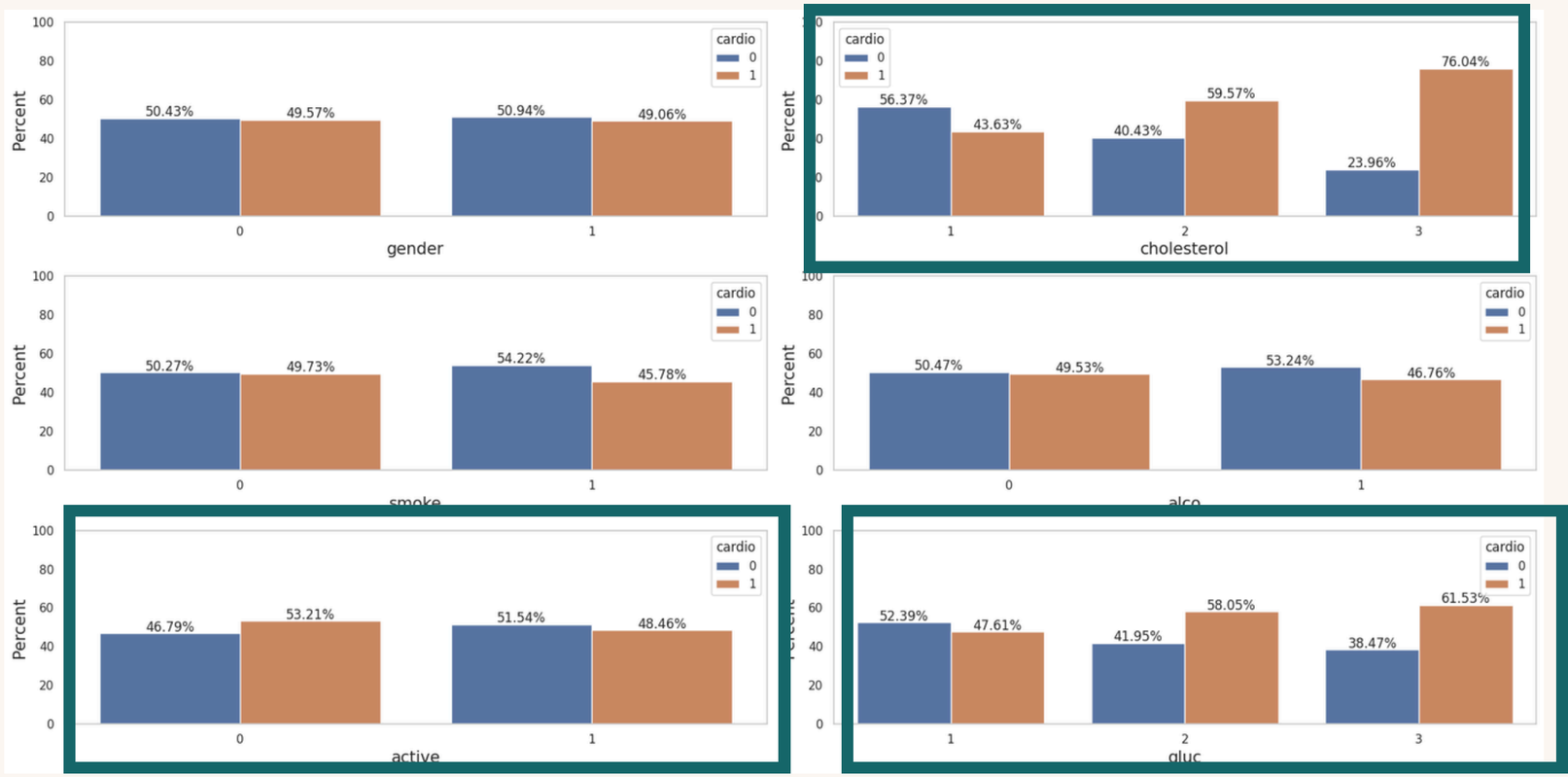
Patient Profile

- Our patient population overall exhibits a **healthy lifestyle**, marked by **high levels of physical activity** and good habits.
- **Regular physical activity** has been proven to have **numerous benefits**, including **improved cardiovascular health, weight management, and mental well-being.**



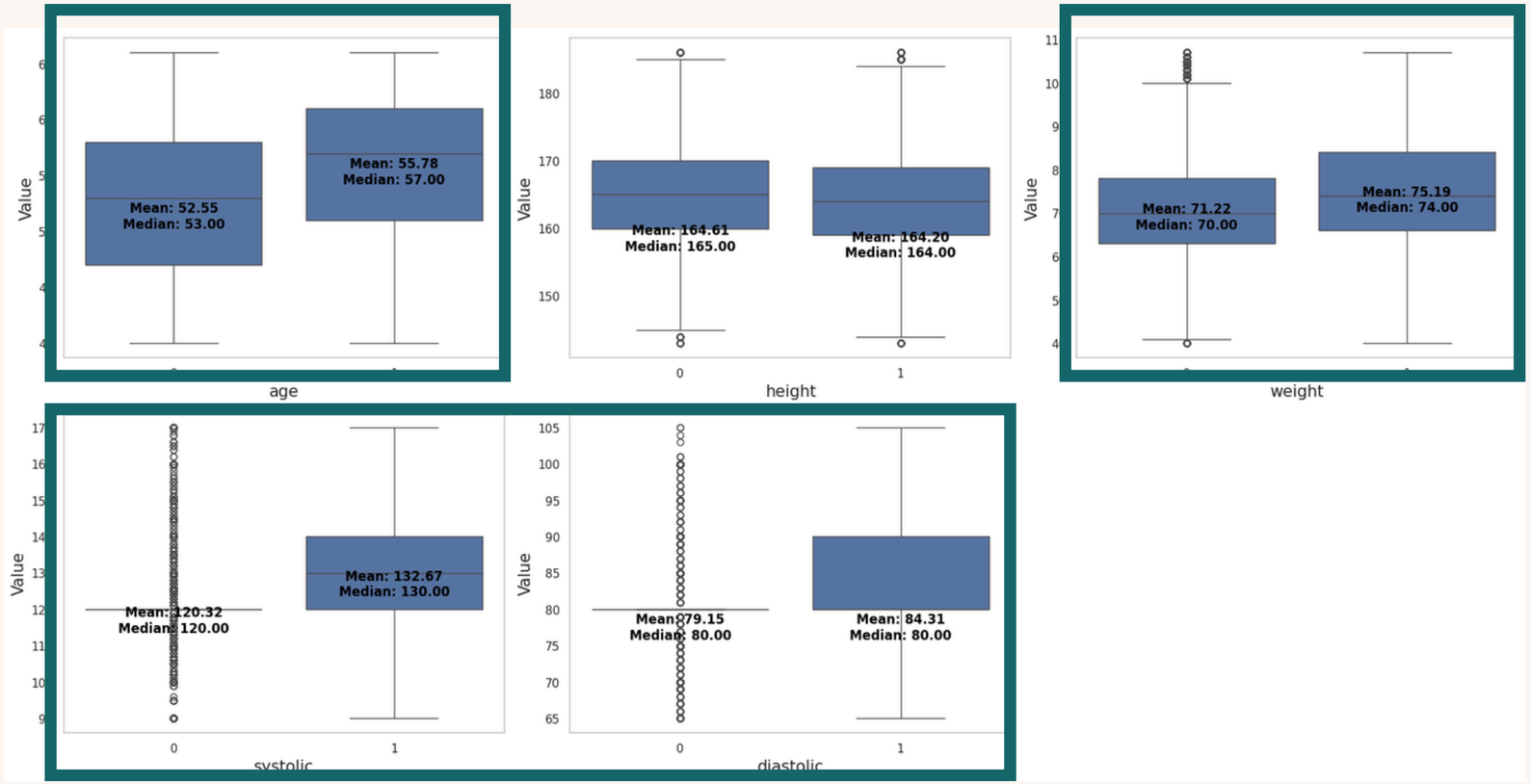
Patient Profile

- People who have **higher cholesterol** and **glucose levels** are prone to cardiovascular disease
- **Physically active people** are **less likely** to have cardiovascular problems



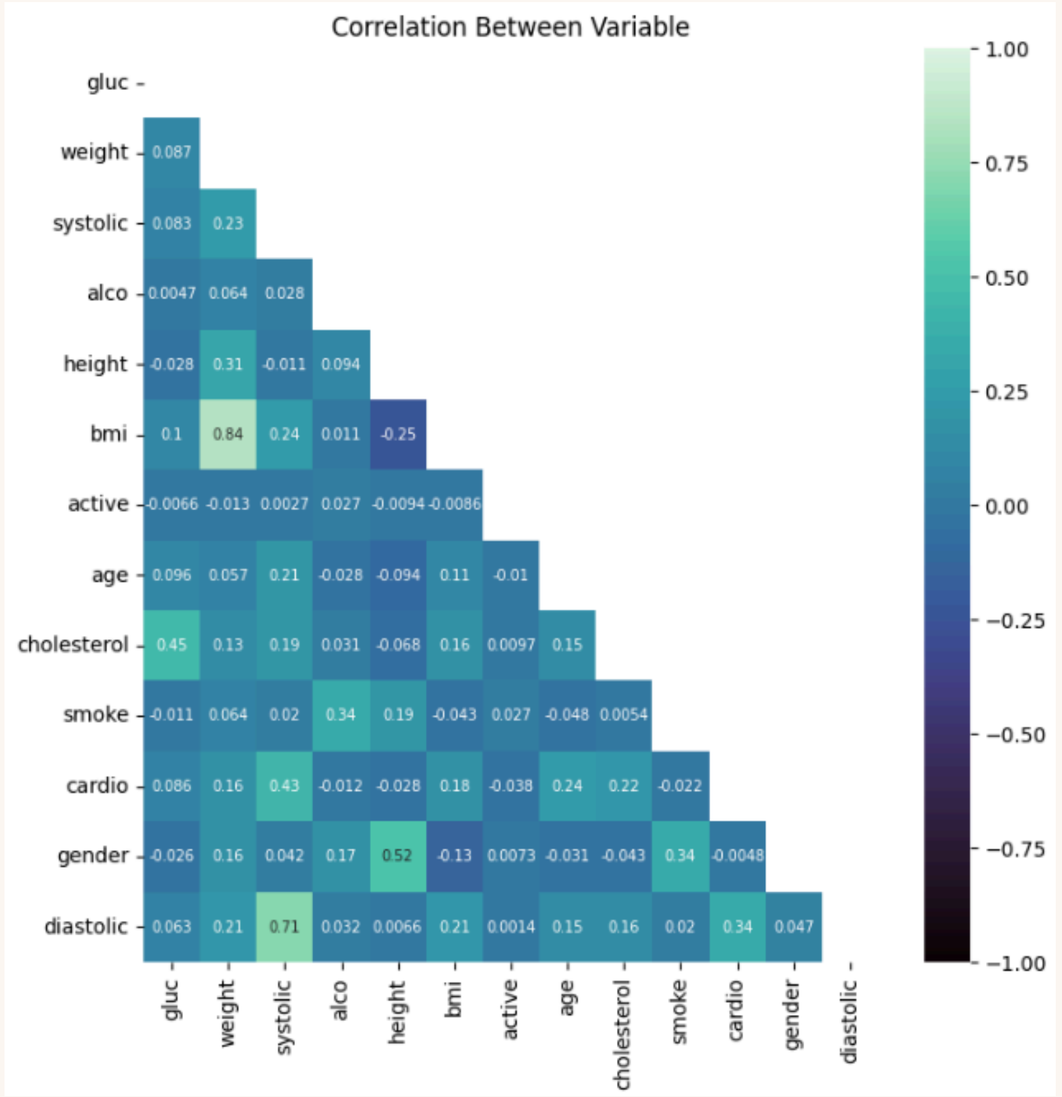
Patient Profile

- People who have **higher weight** and **blood pressure** (**systolic & diastolic**) **levels** are prone to cardiovascular disease
- **Older people** are prone to **cardiovascular disease**



Correlation

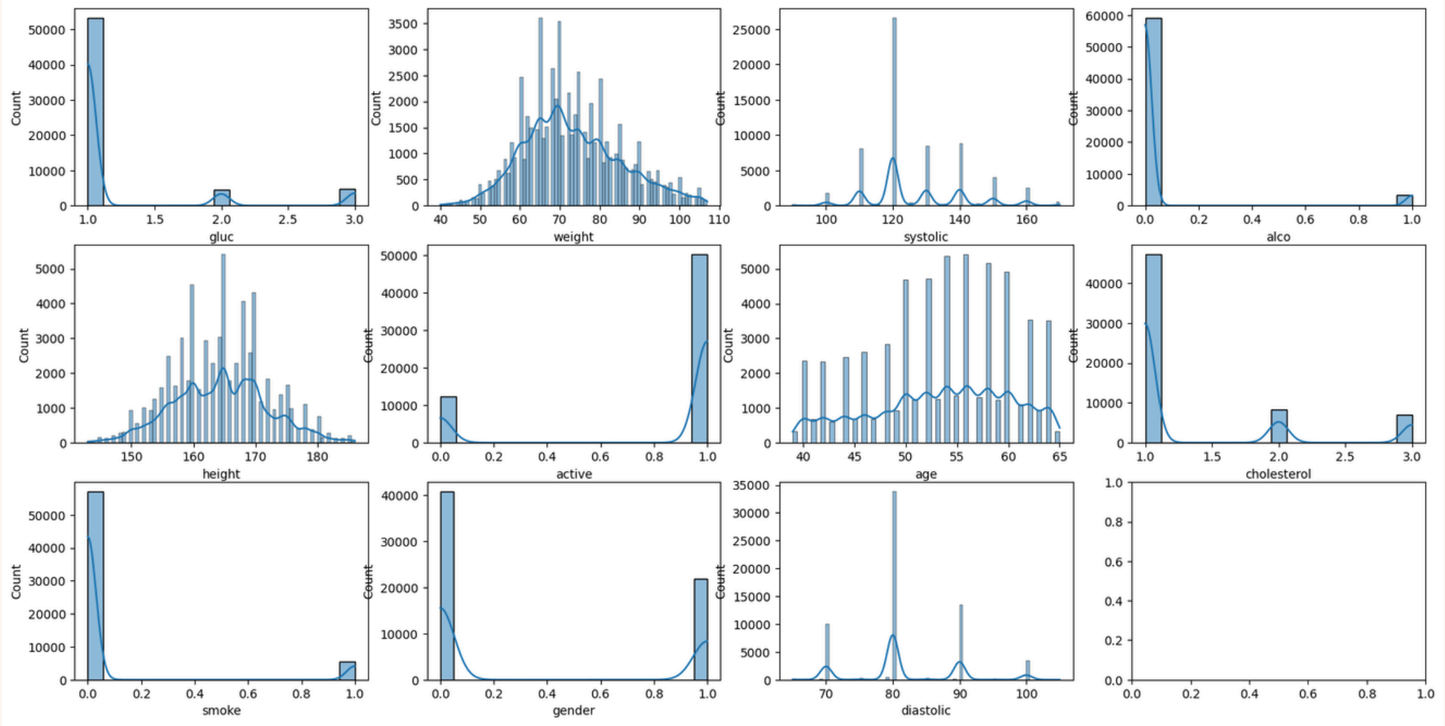
it's shown that **BMI and weight**, are **highly correlated**, also **systolic and diastolic blood pressure. glucose level indirectly correlated to cardiovascular disease** chance since it's correlated wih cholesterol



Feature Scaling - Data Distribution

The Data are relatively having normal distribution. From the picture we decided to scale the data that have range greater than 2 points. using **standard scaler**

(**age, weight, height, systolic, diastolic**)



Modeling

Scaler

```
[66] scaler = StandardScaler()
```

```
[67] to_scale = list(set(corr_column_new)-set(catcol)-set(mvcol))
     to_scale
```

```
['height', 'age', 'diastolic', 'weight', 'systolic']
```

```
[68] scaler.fit(df_new[to_scale] )
```

```
StandardScaler
StandardScaler()
```

```
[69] df_new[to_scale] = scaler.transform(df_new[to_scale])
```

Train Test Split

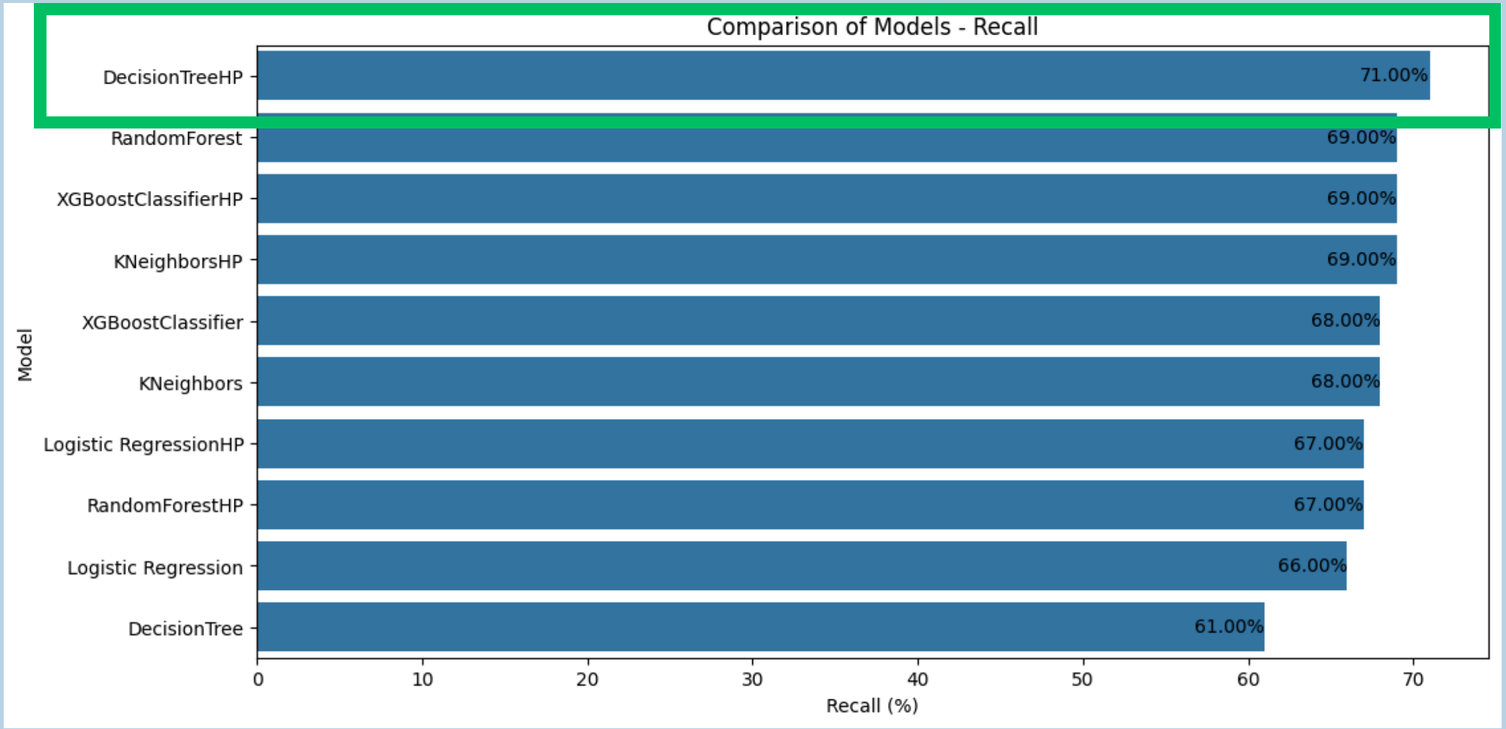
```
[70] # Define the whole data
     X = df_new[corr_column_new]
     y = df_new['cardio']  # Extracting the 'cardio' column as a Series, not a DataFrame

     X_train, X_test, y_train, y_test = train_test_split(X, y,
                                                         test_size=0.3,
                                                         random_state=42)

     y_train = y_train.ravel()
```

Train Model

```
2] models = []

   models.append(('Logistic Regression', LogisticRegression()))

   # Other models
   models.append(('Random Forest', RandomForestClassifier()))

   models.append(('DecisionTree', DecisionTreeClassifier()))

   models.append(('KNearestNeighbor', KNeighborsClassifier()))

   models.append(('XGBoost', XGBClassifier()))

   #Evaluating Model Results:
   acc_results = []
   auc_results = []
   names = []

   col = ['Algorithm', 'ROC AUC Mean', 'ROC AUC STD', 'Accuracy Mean', 'Accuracy STD']
   model_results = pd.DataFrame(columns=col)

   from sklearn import model_selection

   i=0
   #Evaluate model dengan K-Fold Cross Validation
   for name, model in models:
     kfold = model_selection.KFold(n_splits=10) #10 Fold Cross Validation
     cv_acc_results = model_selection.cross_val_score(model, X_train, y_train, cv=kfold, scoring='accuracy')
     cv_auc_results = model_selection.cross_val_score(model, X_train, y_train, cv=kfold, scoring='roc_auc')

     acc_results.append(cv_acc_results)
     auc_results.append(cv_auc_results)
     names.append(name)
     model_results.loc[i] = [name, round(cv_auc_results.mean()*100,2),
                             round(cv_auc_results.std()*100,2),
                             round(cv_acc_results.mean()*100,2),
                             round(cv_acc_results.std()*100,2)]
     i+=1
   model_results.sort_values(by=['ROC AUC Mean'], ascending=False)
```

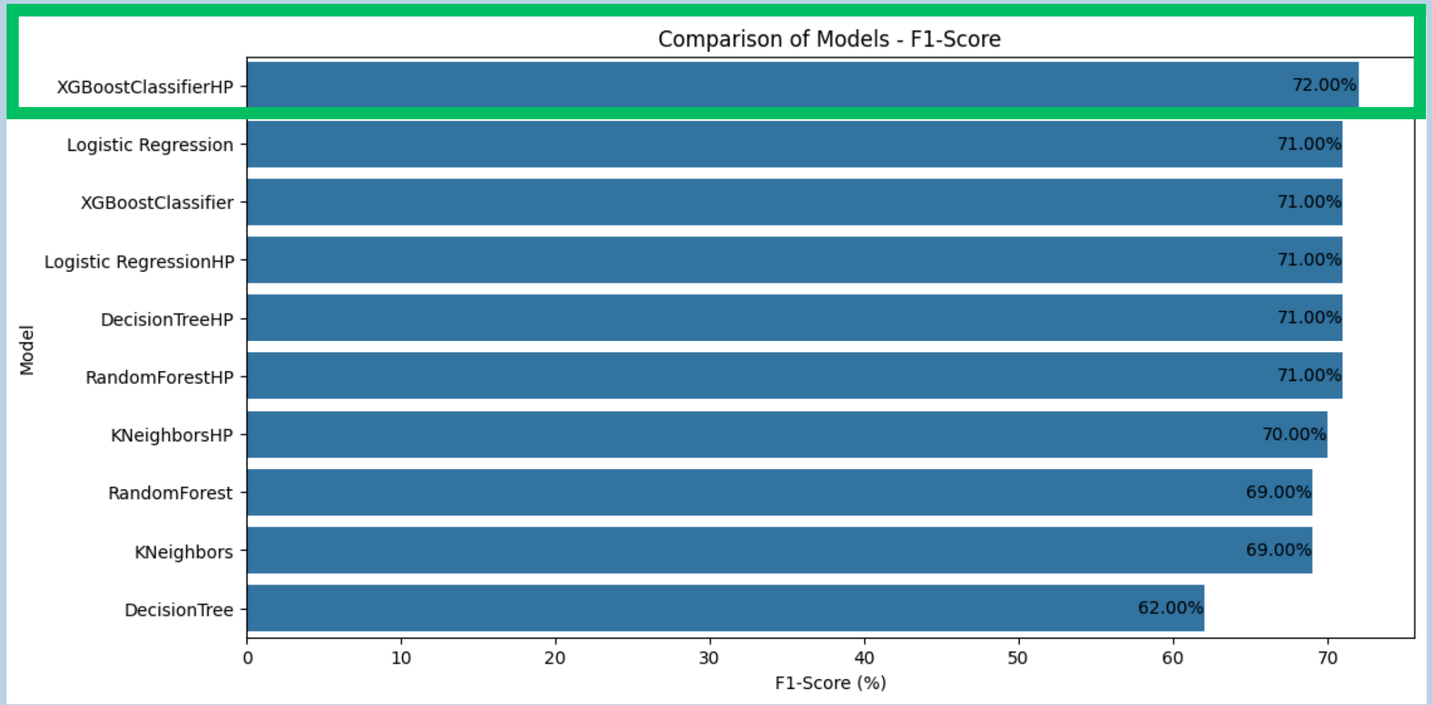
Model Evaluation Recall

Based on recall score,
Parameter-Tuned Decision Tree
have the highest score



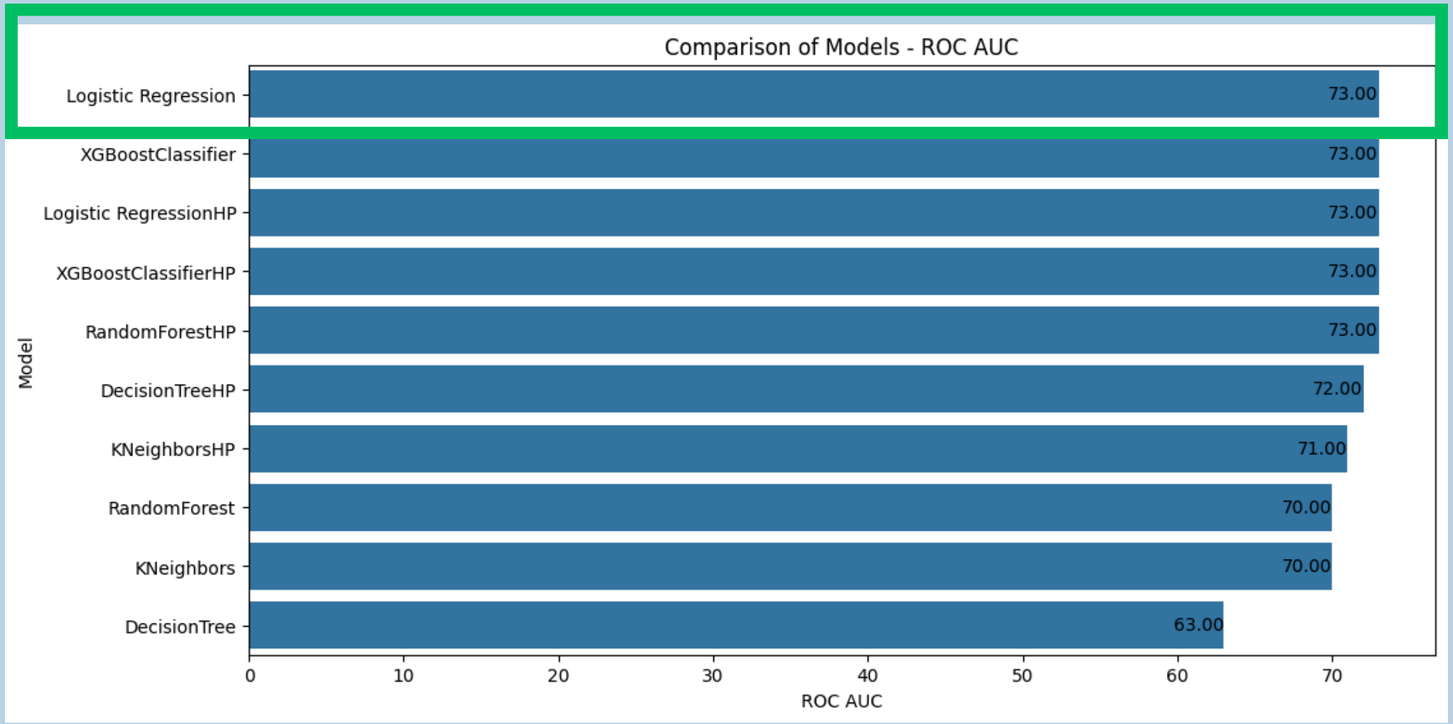
Model Evaluation F1-Score

Based on F1-Score,
Parameter-Tuned XGB
have the highest score



Model Evaluation ROC AUC

Based on **ROC AUC** Score,
Logistic Regression
have the highest score



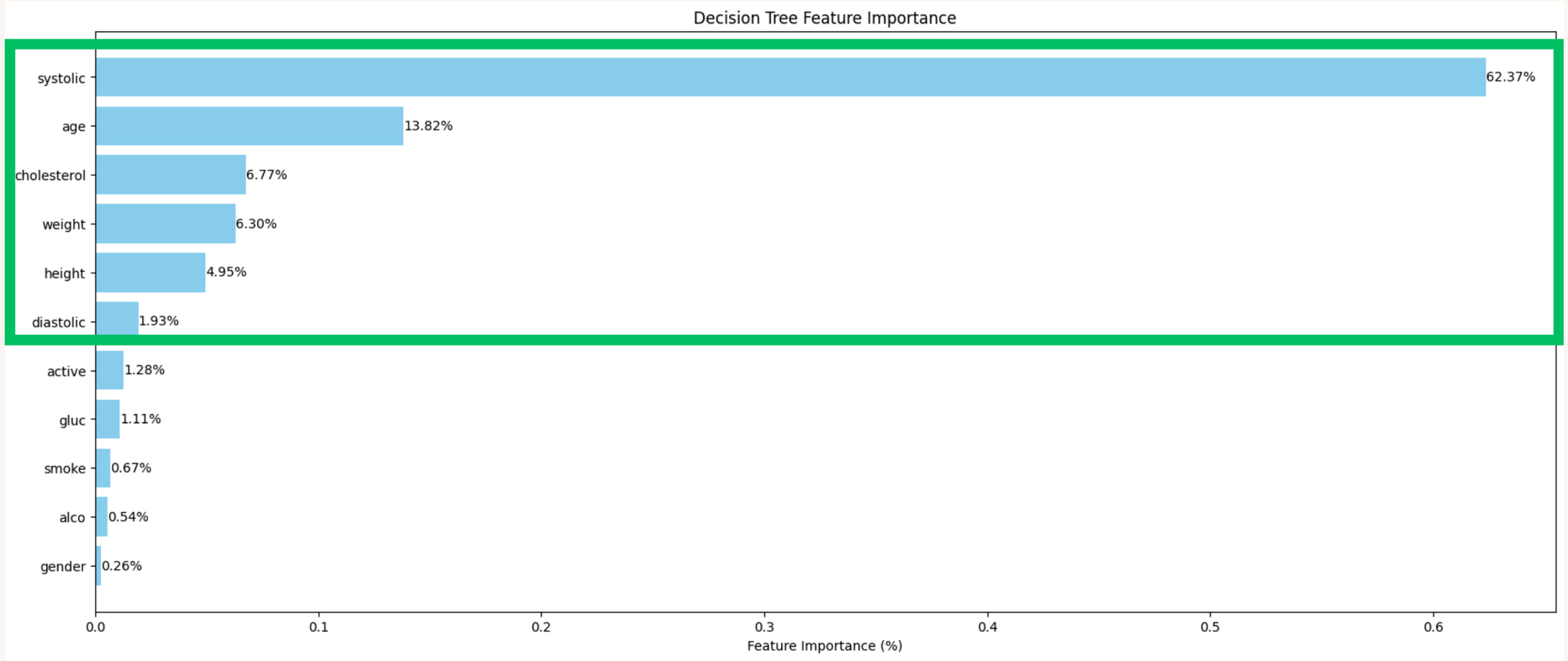
Model Evaluation Overall

Based on the **Overall Score**, **Decision Tree** have the **best performance** among other models.

Since we are predicting **health-related** issue, we'll be **considering Recall and F1-Score**, also **ROC AUC score** in addition.
so we'll select **Parameter-Tuned Decision Tree**

Model	Accuracy	Precision	Recall	F1-Score	ROC AUC
DecisionTreeHP	72.0	72.0	71.0	71.0	72.0
XGBoostClassifierHP	73.0	75.0	69.0	72.0	73.0
KNeighborsHP	71.0	71.0	69.0	70.0	71.0
RandomForest	70.0	70.0	69.0	69.0	70.0
XGBoostClassifier	73.0	74.0	68.0	71.0	73.0
KNeighbors	70.0	69.0	68.0	69.0	70.0
Logistic RegressionHP	73.0	75.0	67.0	71.0	73.0
RandomForestHP	73.0	75.0	67.0	71.0	73.0
Logistic Regression	73.0	75.0	66.0	71.0	73.0
DecisionTree	63.0	63.0	61.0	62.0	63.0

Feature Importance



Age, cholesterol levels, weight, and height (essentially BMI), and blood pressure (systolic & diastolic) are key factors that can influence one's risk status

Model Deployment - Preparation

Before deploying the model, there are **several preprocessing steps** to **convert numerical columns to string column for the UI**

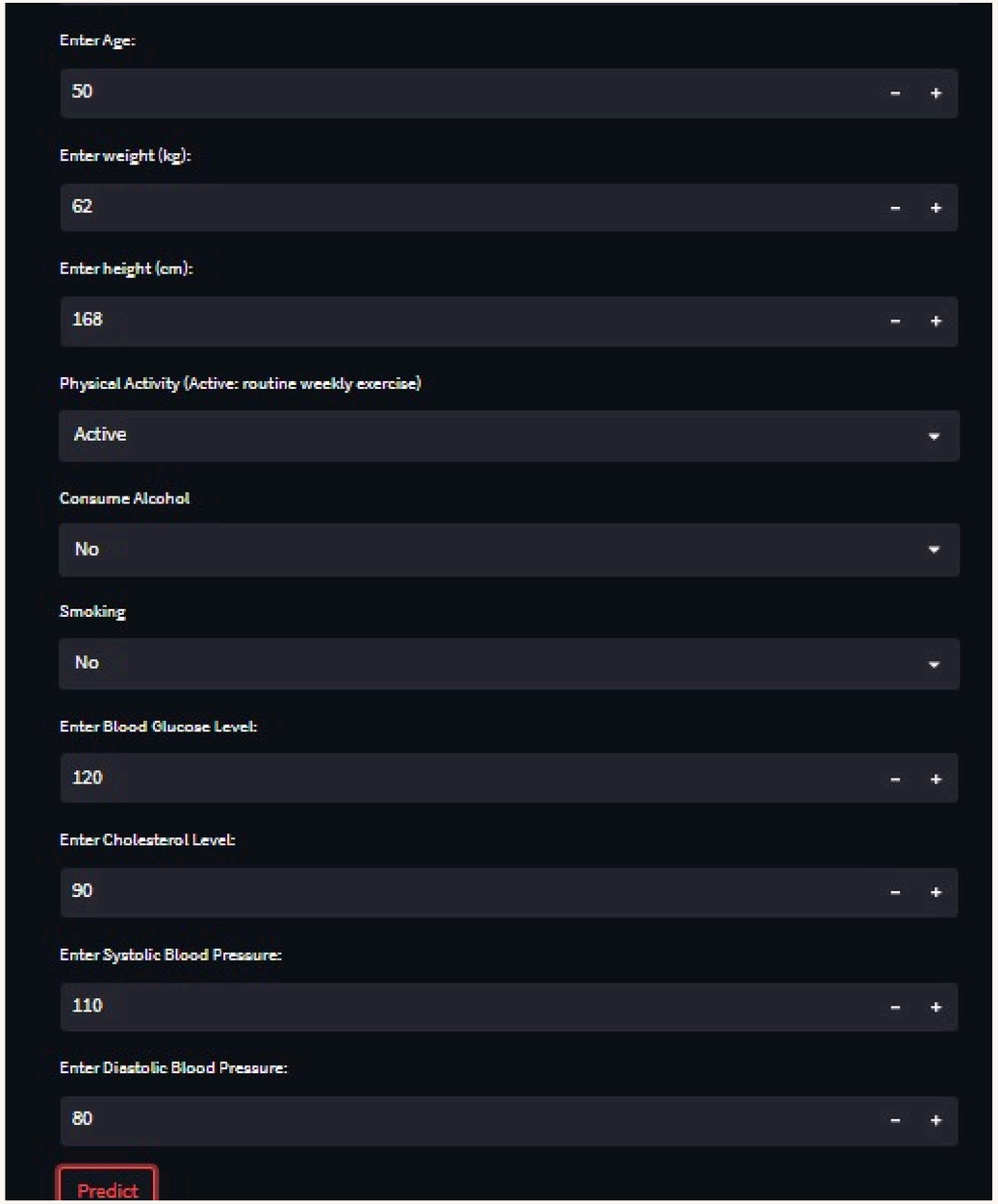
Also, **we create some conditions regarding to the prediction result**,

```
to_scale = ['diastolic', 'systolic', 'height', 'weight', 'age']


def preprocess(df, scaler):
    # impute data with 0 value with median from training set
    df[to_scale] = scaler.transform(df[to_scale])

    #convert_cholesterol
    def categorize_cholesterol(value):
        value = float(value)
        if value < 200:
            return 1
        elif 200 <= value <= 239:
            return 2
        else:
            return 3
    def categorize_glucose(value):
        value = float(value)
        if value < 140:
            return 1
        elif 140 <= value <= 199:
            return 2
        else:
            return 3

    df['cholesterol'] = df['cholesterol'].apply(categorize_cholesterol)
    df['gluc'] = df['gluc'].apply(categorize_glucose)
    df['gender'] = df['gender'].apply(lambda x: 1 if x == 'Male' else 0)
    df['active'] = df['active'].apply(lambda x: 1 if x == 'Active' else 0)
    df['smoke'] = df['smoke'].apply(lambda x: 1 if x == 'Yes' else 0)
    df['alco'] = df['alco'].apply(lambda x: 1 if x == 'Yes' else 0)


    return df
```

Model Deployment

Model is deployed by using Streamlit locally, and we'll give a recommendation based on the risk status, by looking at **model.predict_proba class 1 score**

- Low Risk/Healthy (< 20 %)
  - "Schedule routine check-ups."
- Moderate Risk (20 % to 55 %)
  - "Seek Medical Advice for Preventive Measures."
- High Risk (55 % >)
  - "Urgently consult with a healthcare professional."



Data Geeks

Thank you very much!

[Colab Link]

[Dashboard]

[Model Link]