Cardiovascular Disease Risk Prediction Model

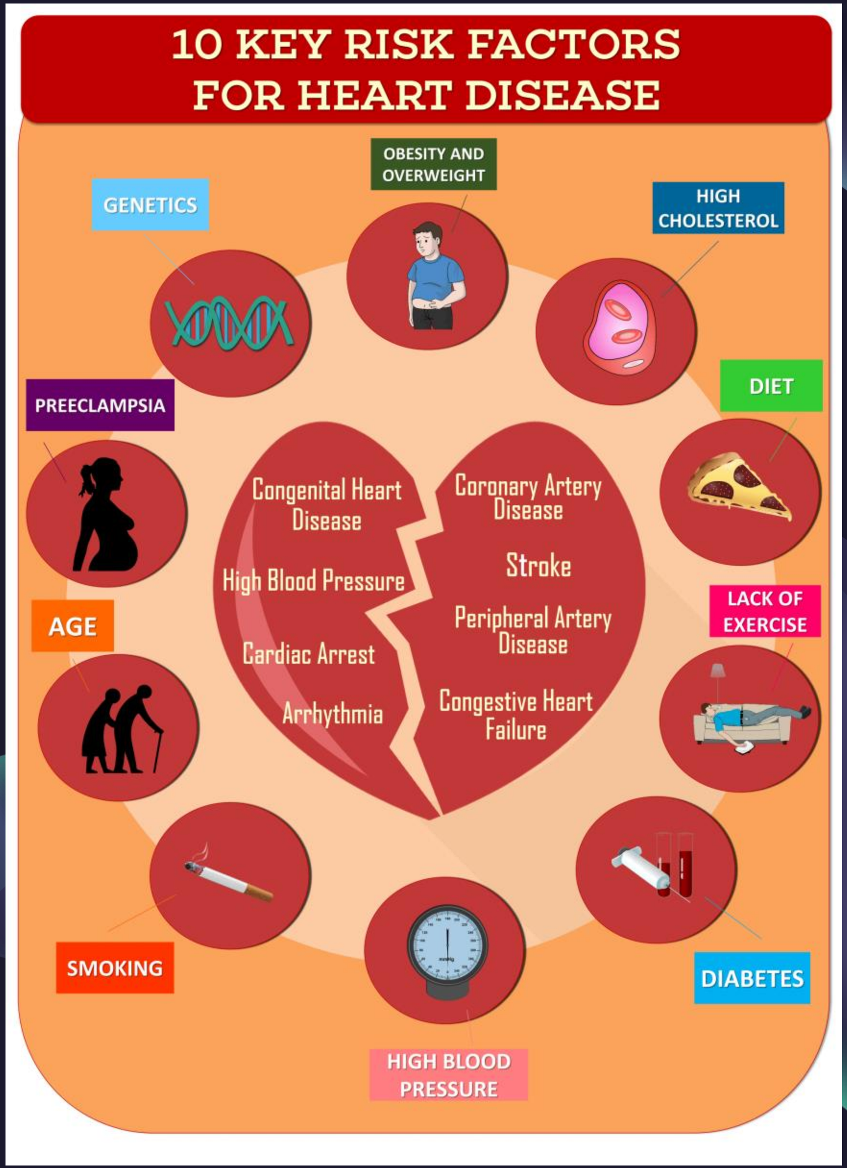
Presentation by: Aline Vo, Annie Joseph Rajan, Stuti Poudel, Brian Stumm, and Jules Gikundiro

Heart Disease

- The leading cause of death globally ~ 18Million/Year
- Cost the US ~ $240 Billion
- Goal – Risk Prediction

Risk Factors:

- Smoking
- Exercise
- Cancer
- General Health
- BMI
- Depression
- Diabetes



Source: https://www.who.int/news-room/fact-sheets/detail/cardiovascular-diseases-(cvds)

Data Cleaning and Preprocessing

Data Cleaning

- Created a unique ID for each row of information
- Created two DataFrames

  - Consumption DB

  - Non-Consumption DB
- Created CSV files
- Imported clean data into SQL

Data Set: https://www.kaggle.com/datasets/alphiree/cardiovascular-diseases-risk-prediction-dataset

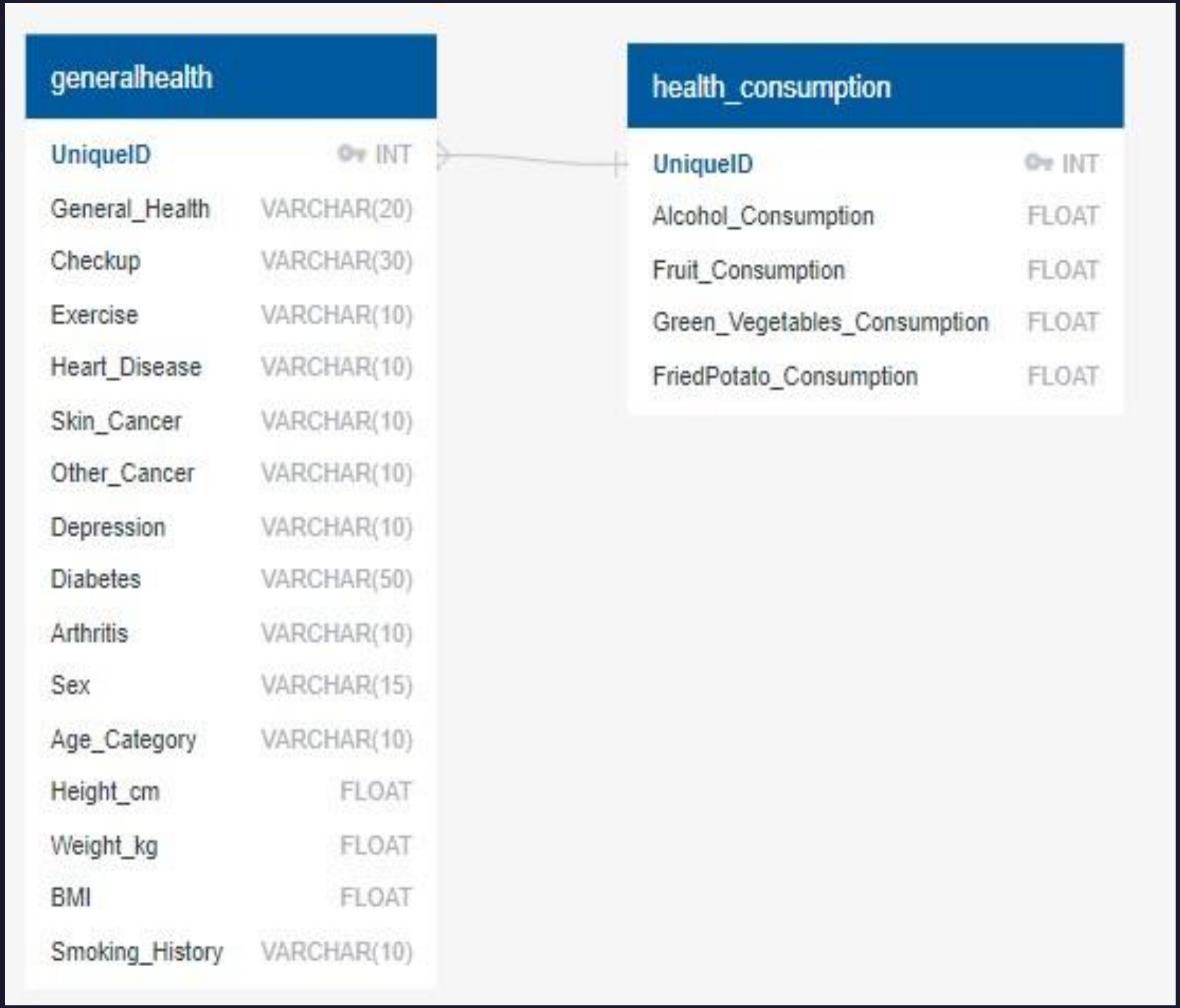
SQL

- Created two tables

  - General Health

  - Health consumption

- ERD



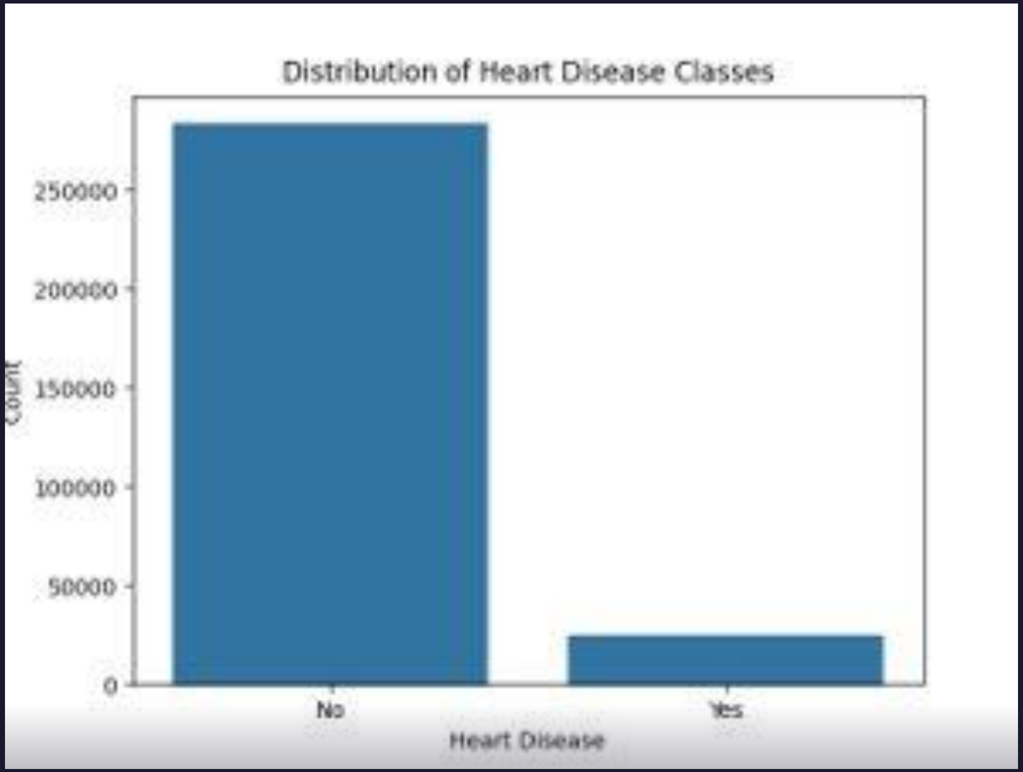
Visualization Dashboard

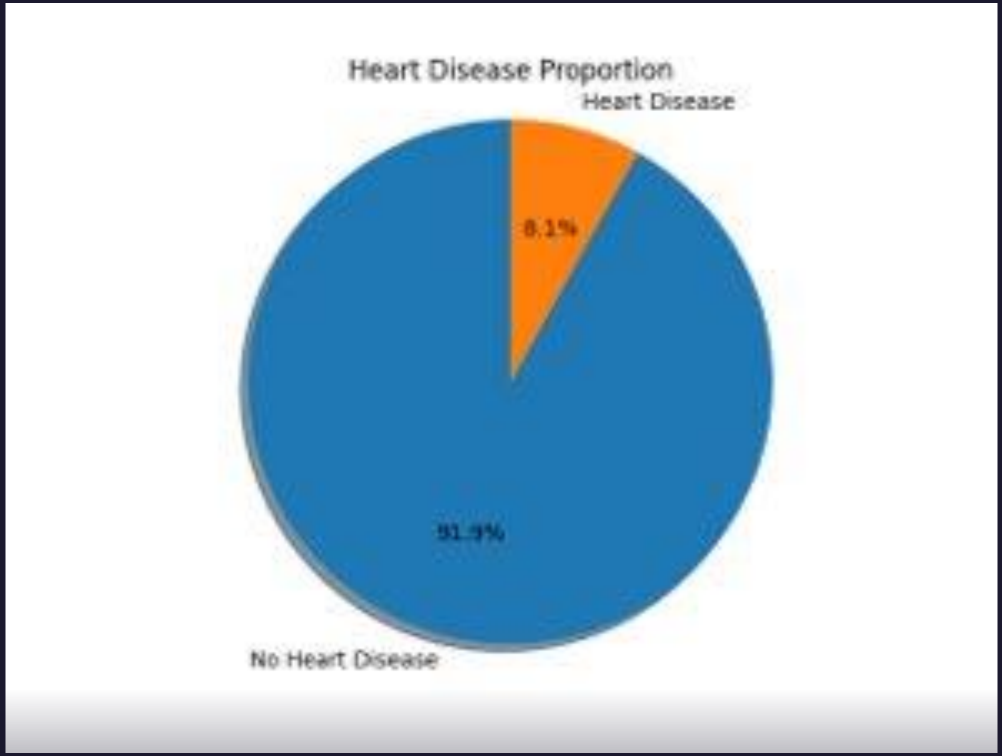
Machine Learning Models

Neural Network Model, Support Vector Machines Model, and Random Forest Classifier Model

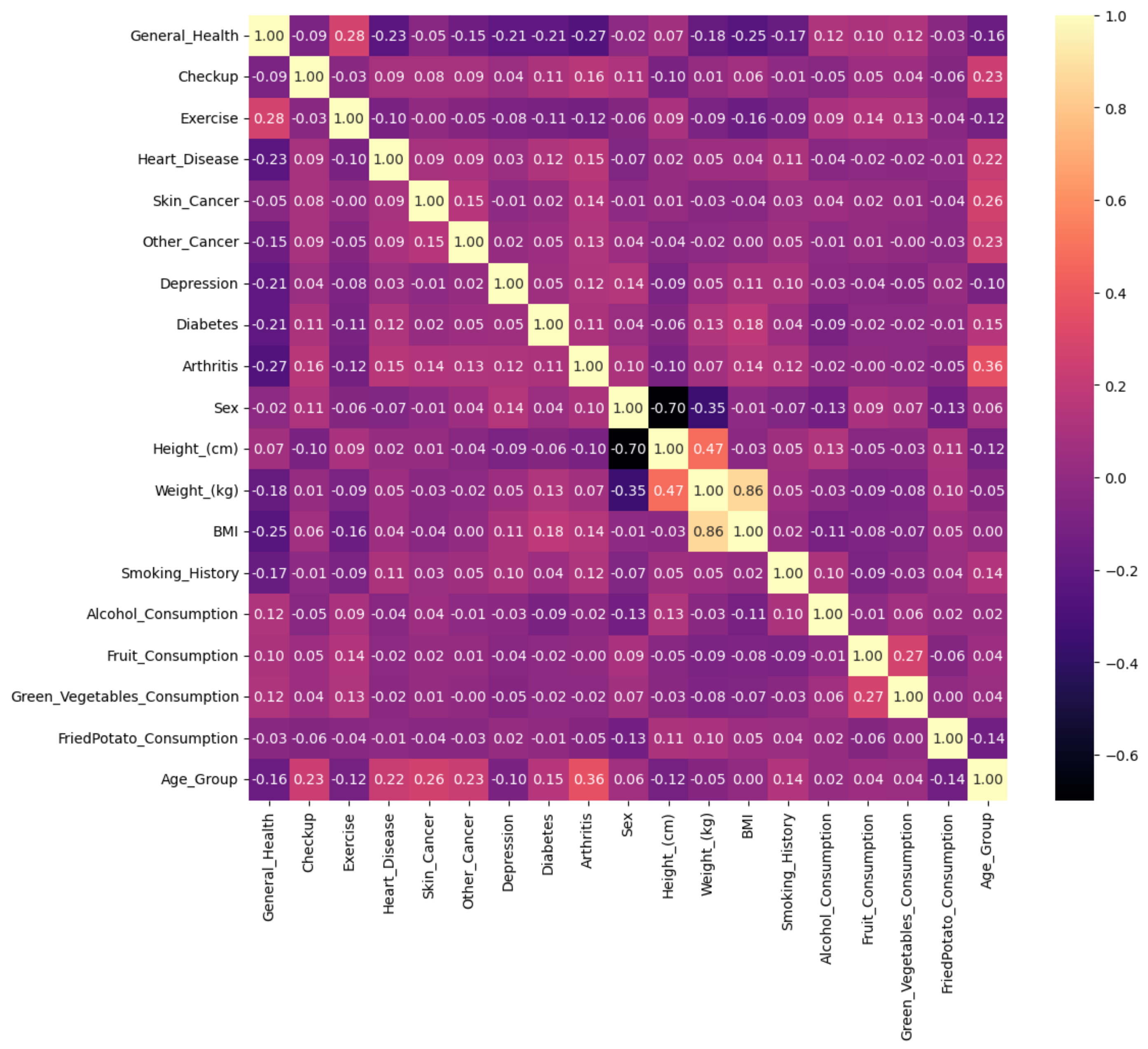
Preprocessing

- Examined distribution – imbalanced data
- Oversampling methods





Correlation Analysis



Neural Network Model

Input Layer :

- The number of neurons equal to the number of features in the input data.

First Hidden Layer:

- Number of Neurons: 80
- Activation Function: ReLU (Rectified Linear Unit)

Second Hidden Layer:

- Number of Neurons: 30
- Activation Function: ReLU

Output Layer:

- Number of Neurons: 1 (since it's a binary classification task)
- Activation Function: Sigmoid (to output probabilities for binary classification)

Neural Network Model Final Results

Before Optimization

```
2413/2413 - 2s - 994us/step - accuracy: 0.9180 - loss: 0.2314
Loss: 0.2313869297504425, Accuracy: 0.9180330038070679
```

After First Optimization

```
2413/2413 - 2s - 971us/step - accuracy: 0.9183 - loss: 0.2329
Loss: 0.23291058838367462, Accuracy: 0.9182920455932617
```

After Second Optimization

```
2413/2413 - 2s - 848us/step - accuracy: 0.9203 - loss: 0.2224
Loss: 0.2224111258983612, Accuracy: 0.9202605485916138
```

Support Vector Machines Model (SVM)

SVM Model

```
from sklearn.svm import SVC

# Create the SVM model with a rdf kernel
model = SVC(kernel='rbf')

# Fit the model to your training data
model.fit(X_train, y_train)
```

```
SVC
SVC()
```

```
# Calculate the classification report
from sklearn.metrics import classification_report
predictions = model.predict(X_test)
print(classification_report(y_test, predictions))
```

```
              precision    recall  f1-score   support

           0       0.92      1.00      0.96     70955
           1       0.00      0.00      0.00      6259

    accuracy                           0.92     77214
   macro avg       0.46      0.50      0.48     77214
weighted avg       0.84      0.92      0.88     77214
```

SVM Model with SMOTE

```
# Applying SMOTE
sm = SMOTE(random_state=42)
X_train_res, y_train_res = sm.fit_resample(X_train, y_train)

# Now, we can use the resampled data to train your model
model = SVC(kernel='rbf')
model.fit(X_train_res, y_train_res)
```

```
Accuracy: 0.7757142487113736

Confusion Matrix:
 [[56205 14750]
 [ 2568  3691]]

Classification Report:
              precision    recall  f1-score   support

           0       0.96      0.79      0.87     70955
           1       0.20      0.59      0.30      6259

    accuracy                           0.78     77214
   macro avg       0.58      0.69      0.58     77214
weighted avg       0.90      0.78      0.82     77214
```

SVM with Balanced class weight

```
from sklearn.svm import SVC

# Create an SVC model with balanced class weights
model = SVC(kernel='rbf', class_weight='balanced')

# Train the model with your data
model.fit(X_train, y_train)
```

```
SVC
SVC(class_weight='balanced')
```

```
Test Acc: 0.712

# Calculate the classification report
from sklearn.metrics import classification_report
predictions = model.predict(X_test)
print(classification_report(y_test, predictions))
```

```
              precision    recall  f1-score   support

           0       0.98      0.70      0.82     70915
           1       0.20      0.81      0.32      6299

    accuracy                           0.71     77214
   macro avg       0.59      0.76      0.57     77214
weighted avg       0.91      0.71      0.78     77214
```

SVM Model Final Report

```
from sklearn.svm import SVC

# Manually specifying the class weights
# Giving class 1 a higher weight
class_weights = {0: 1, 1: 10}

# Create an SVC model with custom class weights
model = SVC(kernel='rbf', class_weight=class_weights)
```

```
Test Acc: 0.739
```

```
# Calculate the classification report
from sklearn.metrics import classification_report
predictions = model.predict(X_test)
print(classification_report(y_test, predictions))
```

	precision	recall	f1-score	support
0	0.97	0.74	0.84	70915
1	0.21	0.78	0.33	6299
accuracy			0.74	77214
macro avg	0.59	0.76	0.58	77214
weighted avg	0.91	0.74	0.80	77214

Random Forest Classifier

```python
# Dependencies
from sklearn.ensemble import RandomForestClassifier
from sklearn.metrics import accuracy_score, confusion_matrix, classification_report, r2_score
```

```python
# Instantiate a Random Forest Classifier model
rf_model = RandomForestClassifier(n_estimators=100, class_weight='balanced', random_state=1)
```

```python
# Fit the model with training data
rf_model.fit(X_train, y_train)
```

```
RandomForestClassifier
RandomForestClassifier(class_weight='balanced', random_state=1)
```

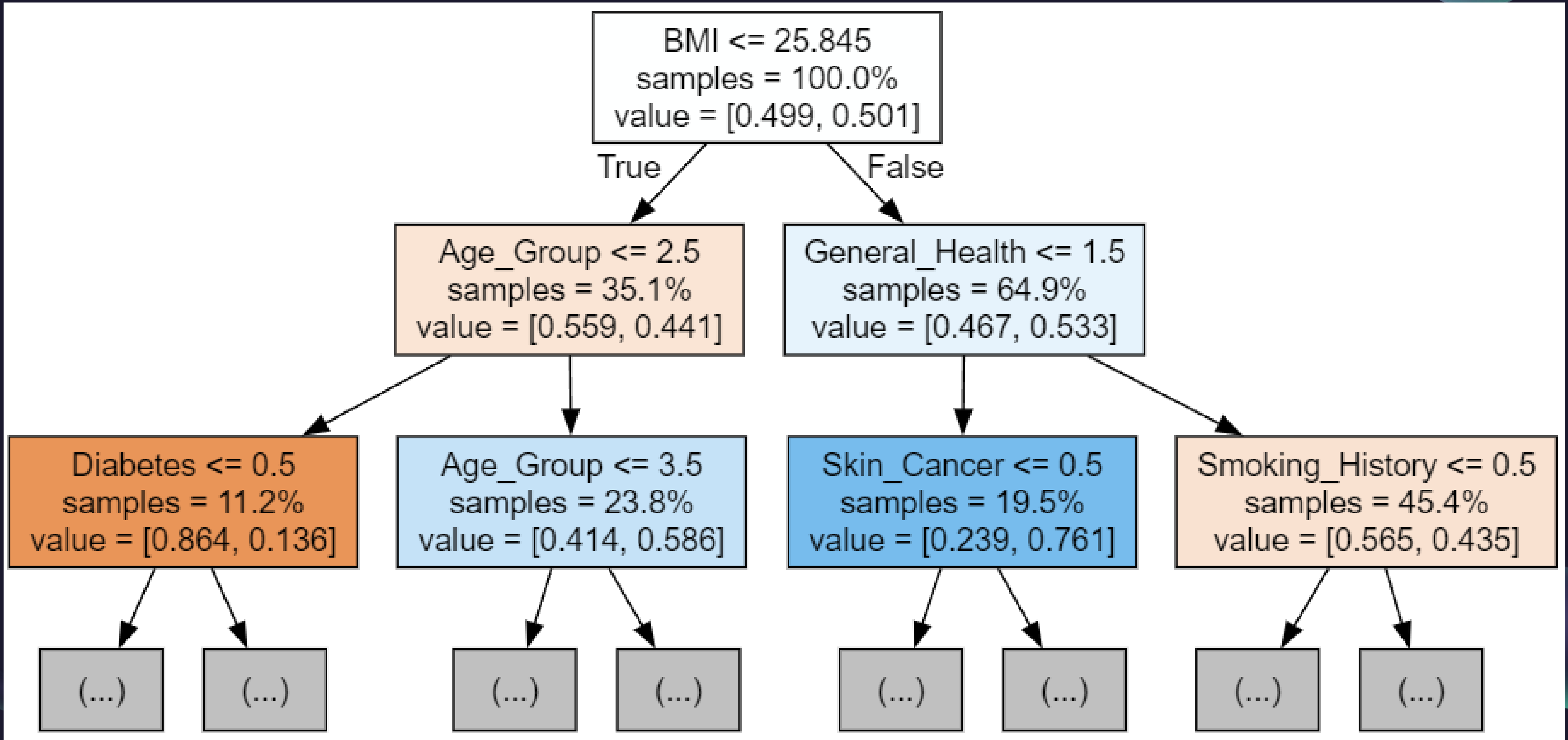
```python
# Make predictions on the test set
predictions = rf_model.predict(X_test)
```

Confusion Matrix

```python
# Generate a confusion matrix for the model
display(cm_df)
```

	Predicted 0	Predicted 1
Actual 0	54823	1999
Actual 1	7	56725

Decision Tree from the Forest



Random Forest Classifier Final Report

```python
# Look at the accuracy score
print(f"Accuracy Score : {acc_score}")
```

```
Accuracy Score : 0.9825369427761241
```

```python
# Look at the classification report
print("Classification Report")
print(classification_report(y_test, predictions))
```

Classification Report

	precision	recall	f1-score	support
0	1.00	0.97	0.98	56822
1	0.97	1.00	0.98	56732
accuracy			0.98	113554
macro avg	0.98	0.98	0.98	113554
weighted avg	0.98	0.98	0.98	113554

Model: RandomOverSampler

...

Classification Report

	precision	recall	f1-score	support
0	0.93	0.91	0.92	56822
1	0.91	0.93	0.92	56732
accuracy			0.92	113554
macro avg	0.92	0.92	0.92	113554
weighted avg	0.92	0.92	0.92	113554

Opt 2: SMOTE

Classification Report

	precision	recall	f1-score	support
0	0.94	0.92	0.93	56822
1	0.92	0.94	0.93	56732
accuracy			0.93	113554
macro avg	0.93	0.93	0.93	113554
weighted avg	0.93	0.93	0.93	113554

Opt 3: BorderlineSMOTE

Challenges and Limitations

- Feature Correlation Analysis shows weak to no correlation to target variable
- Missing Key Features such as, High Blood Pressure, High Cholesterol, Stress, and Family history
- Complexity of the Dataset biased toward minority classes
- Inadequate domain knowledge
- More binary variables than numerical variables

Conclusion

- Random Forest Classifier Model with RandomOverSampler had the best performance
- Accuracy score: 98.23%
- Confusion Matrix
  - True Positive (Actual 1) 56,725 times
  - True Negative (Actual 0) 54,823 times
  - False Negative (Actual 1) 7
  - False Positive (Actual 0) 1999

```
# Look at the accuracy score
print(f"Accuracy Score : {acc_score}")
```

```
Accuracy Score : 0.9823343959701992
```

```
# Look at the classification report
print("Classification Report")
print(classification_report(y_test, predictions))
```

```
Classification Report
              precision    recall  f1-score   support

           0       1.00      0.96      0.98     56822
           1       0.97      1.00      0.98     56732

    accuracy                           0.98    113554
   macro avg       0.98      0.98      0.98    113554
weighted avg       0.98      0.98      0.98    113554
```

```
# Generate a confusion matrix for the model
display(cm_df)
```

	Predicted 0	**Predicted 1**
Actual 0	54823	1999
Actual 1	7	56725

Thank you

Q & A