Machine Learning I

Final Project Report

Institute of Business Administration - IBA

CardioVascular Risk Prediction

with Deployment on Streamlit

Submitted By:

Bilal Naseem - ERP ID: 13216

Kanza Nasim - ERP ID: 27259

May 29, 2023

Abstract

This report presents details the steps and findings for the Final Machine Learning I Project. The Project focused on predicting cardiovascular disease. The dataset consisted of 3,390 observations and 17 attributes, encompassing demographic, behavioral, and medical information. The dataset was chosen after extensive search and came with challenges such as imbalanced target variable, null values, and skewed distributions. Extensive exploratory data analysis was conducted to gain insights into the dataset's characteristics and uncover patterns. Extensive Feature engineering was applied, by dealing with nulls, outliers, feature extraction and reduction, resulting in the creation of over 25 derived datasets and 18 Python working notebooks. Multiple ensemble models, including neural networks, soft voting, and stacking, were employed. Hyperparameter optimization was performed using various techniques, with the objective of maximizing the **area under the precision-recall curve (AUC-PR)**. Different probability thresholds were explored to achieve the best recall while maintaining a good balance of precision. Additionally, Synthetic Minority Over-sampling Technique (SMOTE) was attempted to address class imbalance but did not yield a significant improvement in the model's performance.
To ensure explainability, Lime, Shap, and counterfactual explanations were applied to the top four models obtained from hyperparameter optimization. The interpretability analysis focused on understanding the models' results, identifying their strengths, weaknesses, and insights derived from the explanations. Finally, based on the evaluation metrics and interpretability analysis, logistic regression emerged as the best-performing model, demonstrating the highest AUC-PR. This model was selected for deployment on Streamlit, providing a user-friendly interface to predict cardiovascular disease risk based on the input features.

All python working notebooks, the dataset, streamlit code can be found in the GitHub repository. The streamlit app link can also be found below:

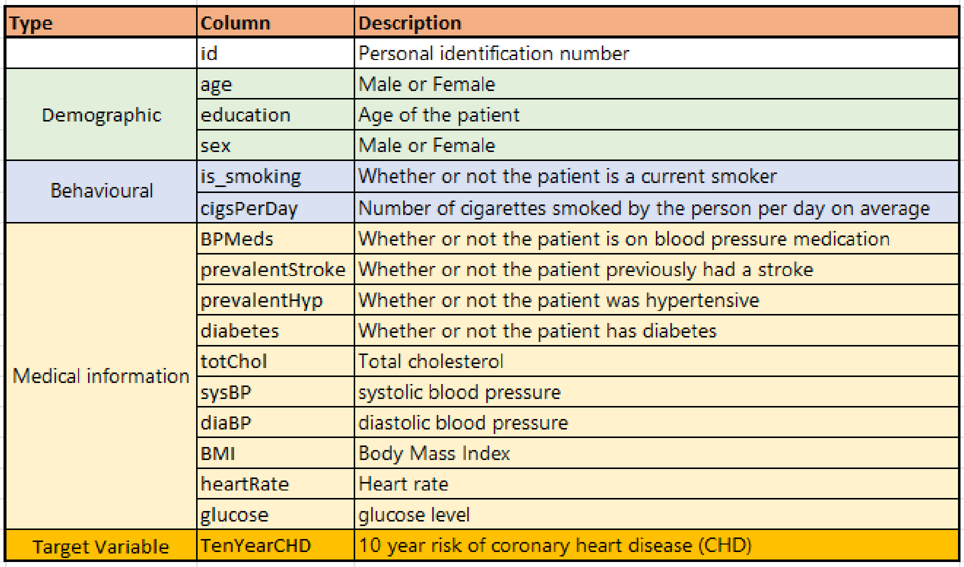
- GitHub Repository Link
- Streamlit app Link
- Dataset Link

Contents

1 Introduction

1.1 About the data

Extensive search was conducted to find a suitable dataset for this project. The Cardiovascular Risk Prediction dataset was chosen considering the requirements of Explainable AI in the project, and a medical dataset would be a good choice for it. It also meets other challenges we were looking for in our dataset such as imbalanced target variable, null values, skewed distributions etc. The dataset contains demographic, behavioral and medical information of individuals along with the target variable TenYearCHD as the target variable which represents the 10 year risk of coronary heart disease (CHD). The following is description of dataset attributes taken from the kaggle dataset page.

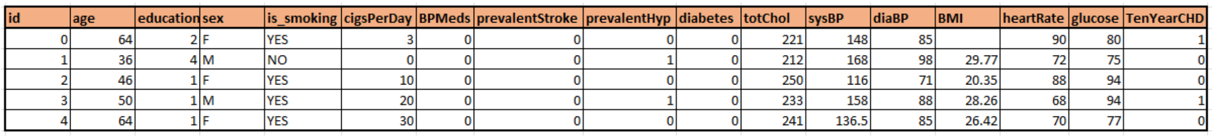


Type	Column	Description
	id	Personal identification number
Demographic	age	Male or Female
	education	Age of the patient
	sex	Male or Female
Behavioural	is_smoking	Whether or not the patient is a current smoker
	cigsPerDay	Number of cigarettes smoked by the person per day on average
Medical information	BPMeds	Whether or not the patient is on blood pressure medication
	prevalentStroke	Whether or not the patient previously had a stroke
	prevalentHyp	Whether or not the patient was hypertensive
	diabetes	Whether or not the patient has diabetes
	totChol	Total cholesterol
	sysBP	systolic blood pressure
	diaBP	diastolic blood pressure
	BMI	Body Mass Index
	heartRate	Heart rate
	glucose	glucose level
Target Variable	TenYearCHD	10 year risk of coronary heart disease (CHD)

Figure 1: Dataset attribute description

1.2 Exploratory Data Analysis

The dataset contains 3390 records and 17 attributes. The following are the first 5 rows of the dataset:



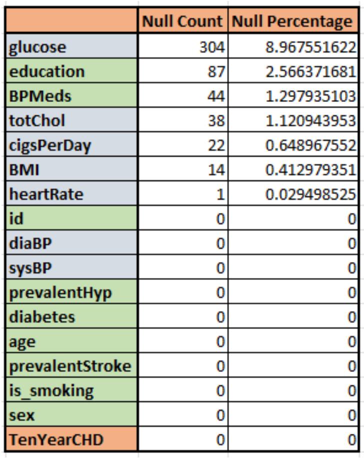
id	age	education	sex	is_smoking	cigsPerDay	BPMeds	prevalentStroke	prevalentHyp	diabetes	totChol	sysBP	diaBP	BMI	heartRate	glucose	TenYearCHD
0	64	2	F	YES	3	0	0	0	0	221	148	85		90	80	1
1	36	4	M	NO	0	0	0	1	0	212	168	98	29.77	72	75	0
2	46	1	F	YES	10	0	0	0	0	250	116	71	20.35	88	94	0
3	50	1	M	YES	20	0	0	1	0	233	158	88	28.26	68	94	1
4	64	1	F	YES	30	0	0	0	0	241	136.5	85	26.42	70	77	0

Figure 2: First 5 rows of the dataset

It can be observed that `sex, is_smoking, BPMeds, prevalentStroke, prevalentHyp, diabetes` are Nominal features, whereas `age, cigsPerDay, totChol, sysBP, diaBP, BMI, heartRate, glucose` are continuous. The target variable `TenYearCHD` (10 year risk of CVD) is Nominal. Education is be assumed to be ordinal.
`sex, education, is_smoking, BPMeds, prevalentStroke, prevalentHyp, diabetes` are taken as categorical columns and the rest as numerical.

1.2.1 Checking for Nulls

Multiple features of the dataset contained nulls. The following images show the numbers and distribution of them:



	Null Count	Null Percentage
glucose	304	8.967551622
education	87	2.566371681
BPMeds	44	1.297935103
totChol	38	1.120943953
cigsPerDay	22	0.648967552
BMI	14	0.412979351
heartRate	1	0.029498525
id	0	0
diaBP	0	0
sysBP	0	0
prevalentHyp	0	0
diabetes	0	0
age	0	0
prevalentStroke	0	0
is_smoking	0	0
sex	0	0
TenYearCHD	0	0

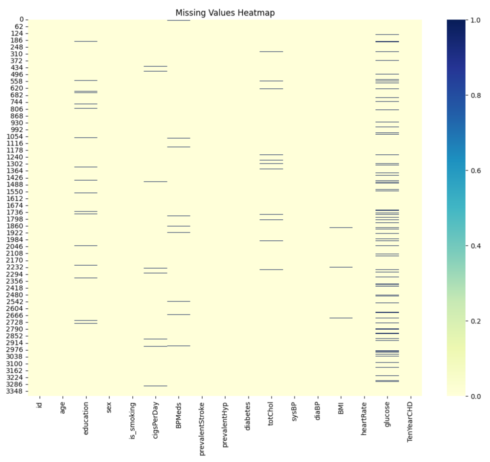


Figure 3: Number of Nulls and their Distribution

1.2.2 Distribution of Numerical Columns

Distribution of all Numerical columns were plotted using seaborn's histplot and are shown figure 6. The mean and median lines are also shown in it. Q-Q plots were also plotted to have a better understanding about the skewness of the data, results are in figure 6.

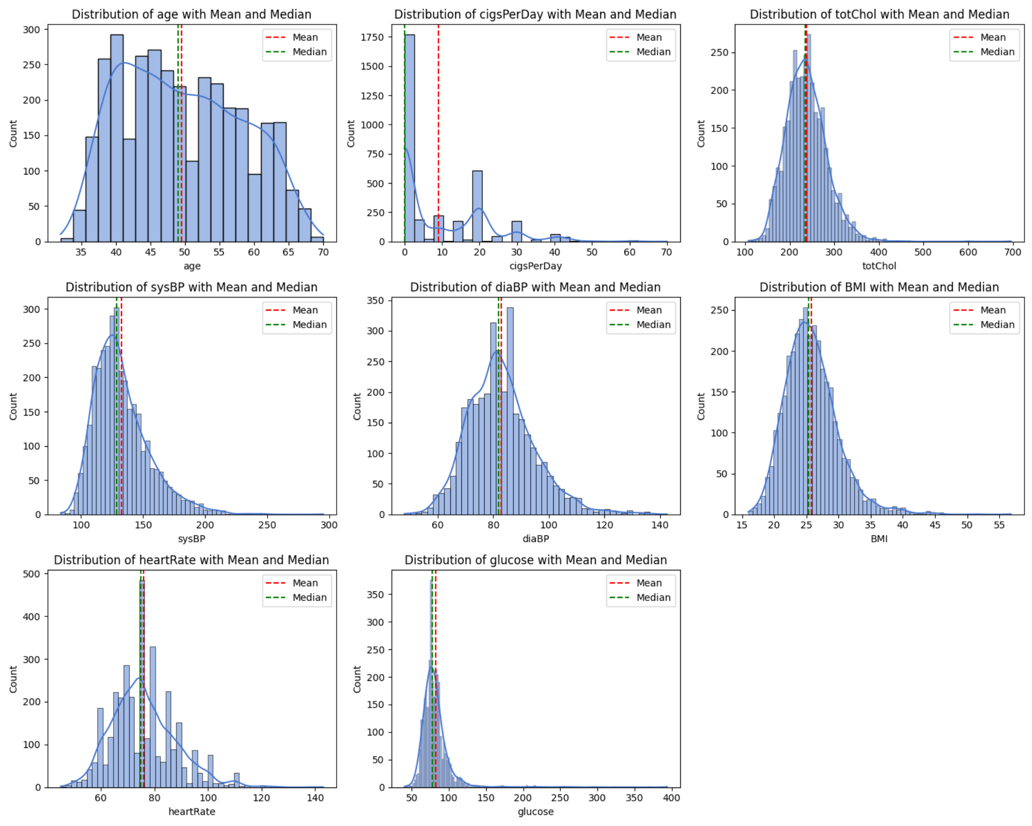


Figure 4: Distribution of numerical columns

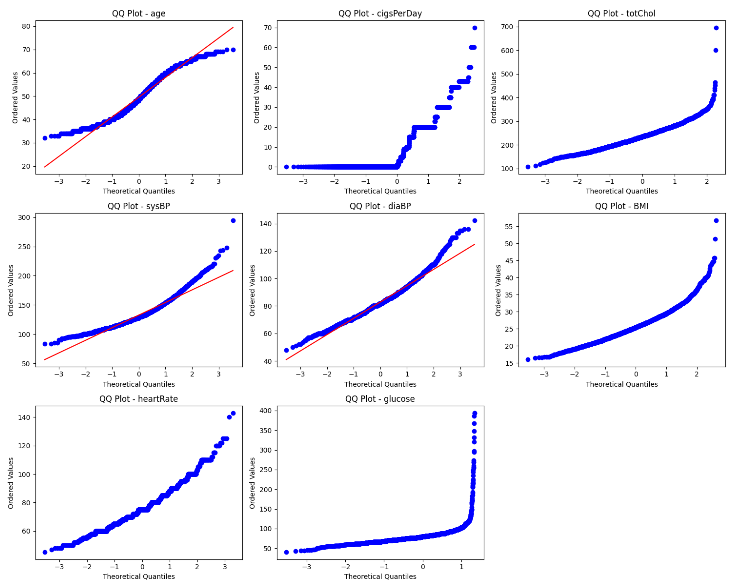


Figure 5: Q-Q plots of numerical columns

It can be observed that most variables are skewed right and some have outliers also. The average age of people in the dataset is just below 50 years.

1.2.3 Distribution of Categorical Features

Distribution of all categorical columns was plotted using seaborn's countplot. Results are in figure 7.

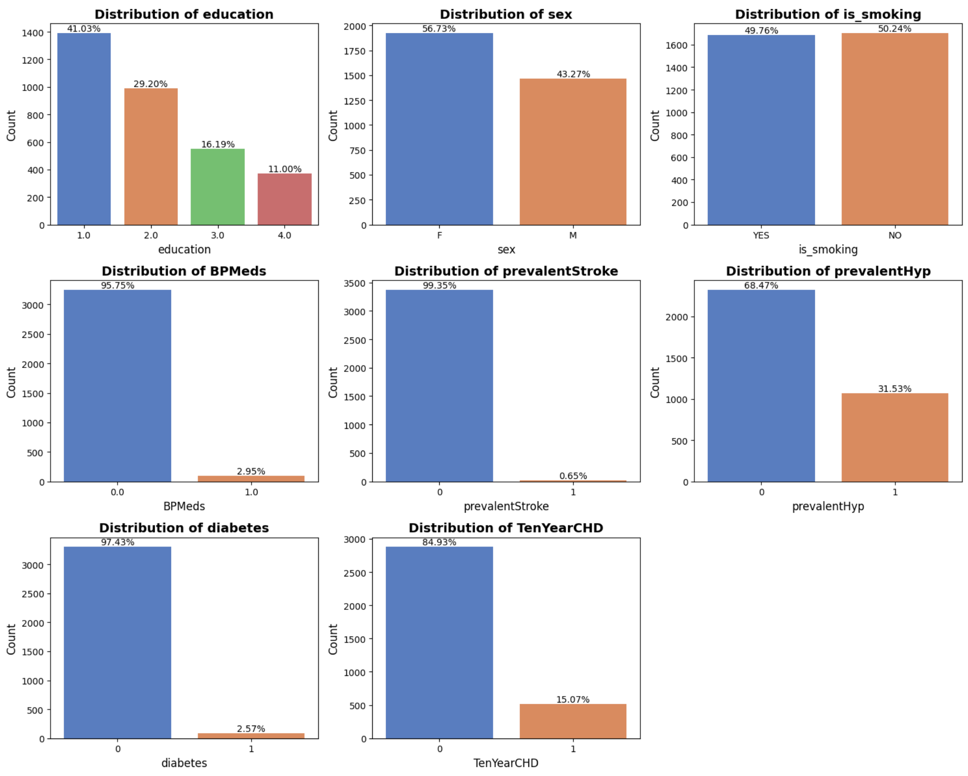


Figure 6: Distribution of Categorical columns

From figure 7 the following observations can be made:

- There are more number of females present in the dataset than males.
- Assuming that the values in Education feature are hierarchical in ascending order, more number of people are less educated in the dataset
- Only 87 people have diabetes (2% of the dataset)
- Only 22 people have had a recorded history of stroke (0.6% of dataset)
- Only 3% (100) people are taking medications (BPMeds) eventhough there are 31.5% (1069) people with a history of being hypertensive.
- It can also be seen that the target variable TenYearCHD is imbalanced, and 85% of people belong to class 0 (having no risk of developing CHD in the next 10 years). Before moving on to applying models, this would be needed to treated first.

1.2.4 Relationship between Continuous Variables and Dependant Variable

Violin along with Strip plot were plotted to visualize the relationship between the Continuous variables and the Dependent variable. The violin plot was chosen as it helps to effectively visualise the distribution of a continuous variable across the different categories of a categorical variable. The results are in the figure below:

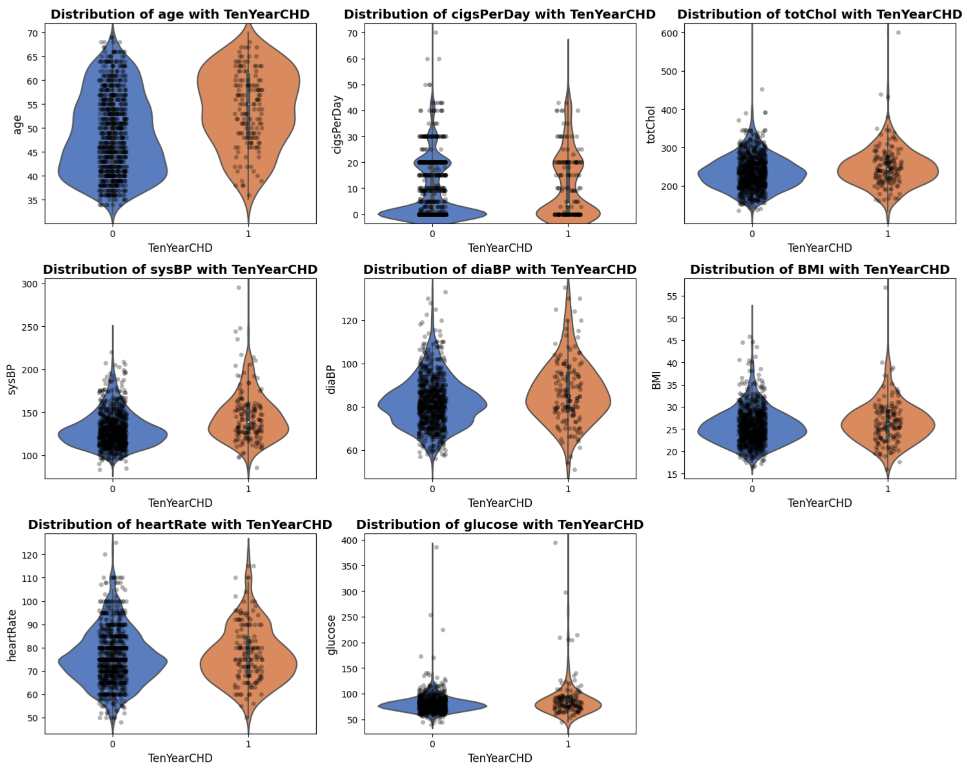


Figure 7: Relationship between Continuous Variables and Dependant Variable

From figure 7 it can be observed that:

- Age is a major factor contributing to risk of CVD. More number of older patients have a risk of CVD than younger patients

- More number of non-smokers are present among those who have no risk of CVD in next ten years.

1.2.5 Relationship between Categorical Variables and Target Variable

Countplots were plotted categorical features and the target variable side by side of each other. Also the percentage of distribution of each category was also mentioned in the figure.

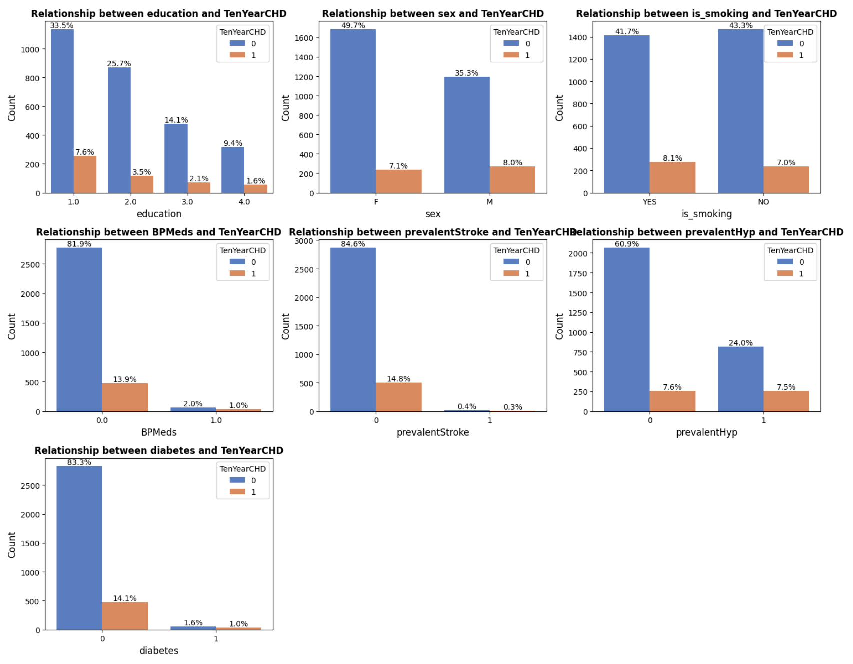


Figure 8: Relationship between Categorical Variables and Target Variable

From figure 8 it can be observed that:

- Males have a slightly higher percentage of risk of CVD compared to females
- Those with lower level of education have a higher percentage of CVD risk, followed by those with highest level of education
- Very little effect of smoking on CVD risk can be observed.
- Patients who take BP Medications have a significantly higher risk of getting CVD (33%) than those who do not take the same (14.5%)
- Those who have a history of Stroke have a higher risk of getting CVD (45.5%) than those who don't have a history of stroke (15%)
- Those who have a history of Hypertension have a higher risk of getting CVD (24%) than those who don't have a history of Hypertension (11%)
- 38% of those who have diabetes have a risk of getting CVD while only 15% of those with no diabetes have a risk of CVD

1.2.6 Correlation amongst Variables

High correlation between a few of the independent variables can be observed in the correlation Heatmap, like between sysBP and diaBP, prevalentHyp and sysBP, and diabetes and glucose.

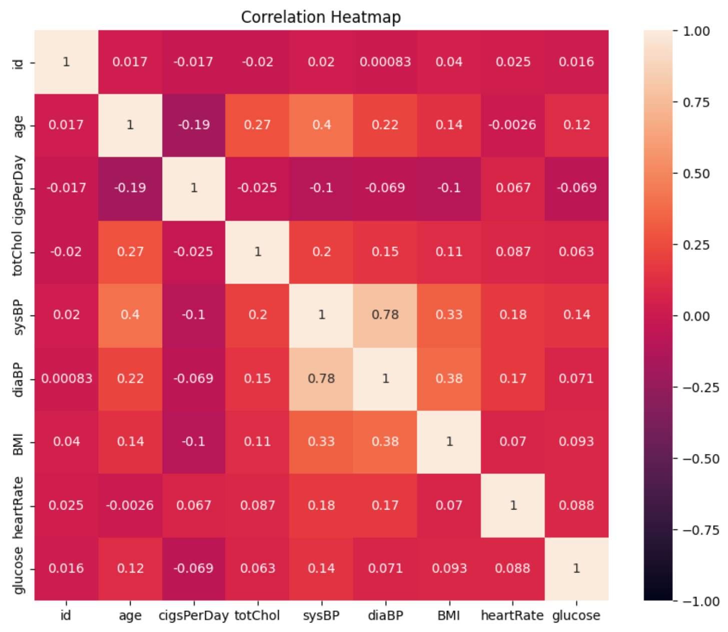


Figure 9: Correlation Heatmap

Conclusions:

- `education` has almost no correlation with the target variable TenYearCHD and so can be dropped.
- `is_smoking`is removed because of the redundancy of information with respect to cigsPerDay. This is further discussed in section 2.1
- `prevalentStroke` is removed because of the high class imbalance - only 0.5% of the dataset having one class

2 datasets were created one with these columns and the other without and performance of both datastes were compared. This is discussed in section 2.2.1. Also insights gathered from this section are used in section 2 for feature engineering which explains handling of nulls, outliers, and feature engineering done based on the insights gathered from this section of EDA.

2 Feature Engineering

2.1 Dealing with Nulls

As it can be seen from figure 3 of section 1.2.1, 7 columns of the dataframe contain nulls. Insights gathered from the EDA section were used to impute nulls. Also the education column was dropped entirely, so imputation wasn't needed for it.

BPMeds

Since nulls in BPMeds account to less than 2% of the data, the rows containing null rows were removed entirely.

cigsPerDay

There are 2 columns in the dataset related to smoking: 1. `cigsPerDay` and `is_smoking`. A pairplot was made to analyze the relationship between the two.

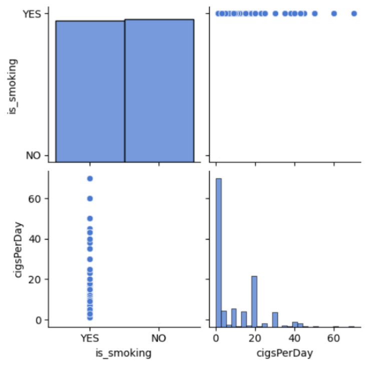


Figure 10: cigsPerDay and is_smoking Pairplot

It can be observed that whenever `is_smoking = "NO"` the value of cigsPerDday is always 0. This explains the right skewness of the cigsperday column in figure 4.Also from figure 4 it can be seen that there are many outliers in this column, so median would be a more appropriate method for imputation. However most of the values in this column are 0 (where is_smoking = 0) and the mean and median both are 0. And so it would be much more appropriate to impute the median from subset of the data where `is_smoking = "YES"`. This was done using the code: `df[df['is_smoking'] == 'YES']['cigsPerDay'].median()` which comes out to be 20.

totChol, BMI, heartRate

Since nulls in these columns were less than 1.1% of the data and columns were slightly skewed, they were imputed using the median of the entire data.

Glucose

The glucose column contained ≈ 9% nulls and so it was imputed using 3 different techniques, and in each of the 2 datasets at the end of EDA section imputation was done using the below methods.

1. Median
2. Iterative Imputer
3. KNN Imputer (k=5)

After this stage 6 datasets were created. Iterative imputer and KNN gave better results than median imputation.The results of performance of each dataset are in figure 13. I was observed that Iterative imputer gave the best results whereas imputing with median gave the worst results.

2.2 Dealing with Outliers

It was observed in section 1.2.2 that all numerical columns are either slightly or heavily right skewed. There are 2 standard methods of dealing with outliers: 1. Trimming, and 2. Winsorizing. Since the many columns contain outliers **Winsorizing** was opted and gave excellent results. Winsorizing preserves the information contained in extreme values by replacing outliers with less extreme but still representative values and reduces the impact of outliers. Different thresholds were applied on different columns and the before and after results of winsorizing are in figures 11 and 12. Datasets with and without winsorizing were compared (12 datasets at this stage), and datasets with winsorizing gave significantly better results.

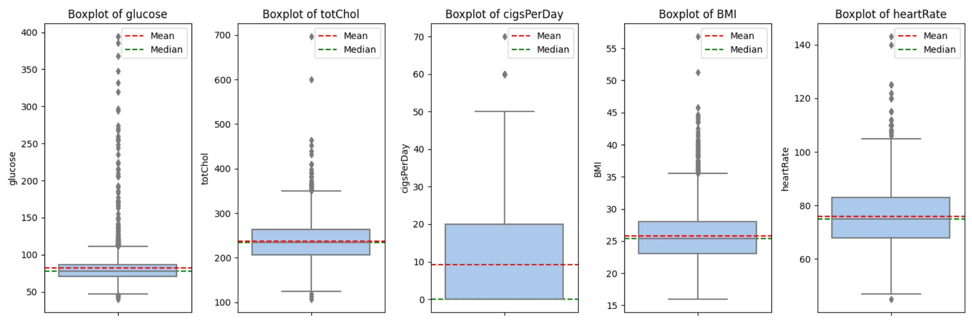


Figure 11: Numerical columns before Winsorizing

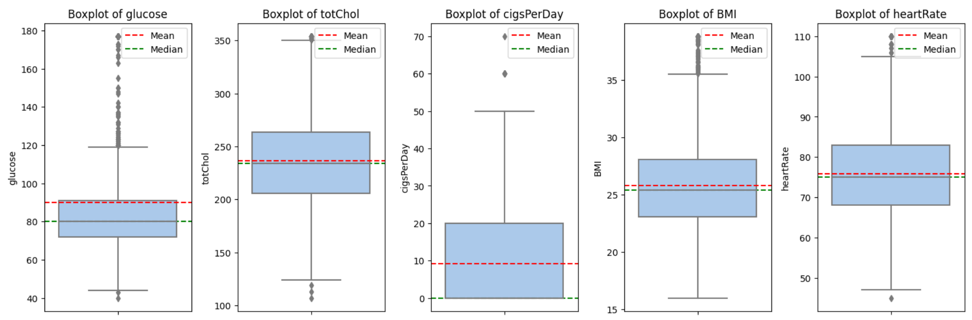
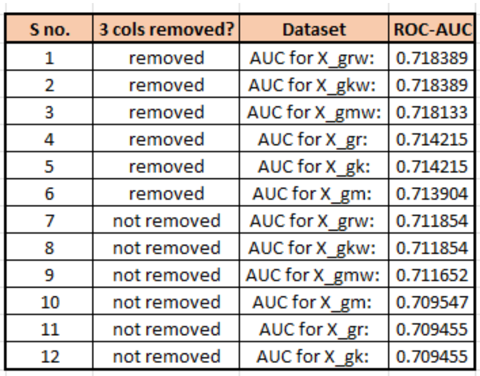


Figure 12: Numerical columns after Winsorizing

2.2.1 Dataset comparision

12 Datasets were created upto this stage and their performance was checked with the help of Naive Bayes. Roc was chosen as the metric to tell which dataset is better in general. However later on in the project more emphasis was given to **AUC-PR** becuase of the imbalanced nature of our dataset and the objective of the project. One hot encoding was done before applying the models. Results are in figure below: Datasets ending with subscript w are the ones on which winsorizing has been applied, and they show give better results than on which it wasn't applied.



S no.	3 cols removed?	Dataset	ROC-AUC
1	removed	AUC for X_grw:	0.718389
2	removed	AUC for X_gkw:	0.718389
3	removed	AUC for X_gmw:	0.718133
4	removed	AUC for X_gr:	0.714215
5	removed	AUC for X_gk:	0.714215
6	removed	AUC for X_gm:	0.713904
7	not removed	AUC for X_grw:	0.711854
8	not removed	AUC for X_gkw:	0.711854
9	not removed	AUC for X_gmw:	0.711652
10	not removed	AUC for X_gm:	0.709547
11	not removed	AUC for X_gr:	0.709455
12	not removed	AUC for X_gk:	0.709455

Figure 13: Derived Datasets performance comparision

2.2.2 Finding Optimal Winsorizing Thresholds

Since winsorizing gave a boost in ROC-AUC, we tried finding the optimal threshold for each column which would give the highest score. A nested for loop was created iterating over multiple values of both lower and upper threshold on all columns returning the thresholds which give the highest ROC-AUC through Naive Bayes. The code is present in `optimal_winsor.ipynb` file. The ROC jumped to **0.7454 from 0.71838** after optimizing the thresholds. Also Applying MinMaxScaler before Naive Bayes improved the results slightly. The working of this file is present in `optimal_winsor.ipynb` file.

2.3 Mathematical Transformations

Multiple Mathematical Transformations were applied on numerical columns such as Log and Yeo-Johnson, but didn't give an improvement in score. This could be due to removal of outliers before this stage, which made the distributions more normal. Working is in `optimal_winsor.ipynb` file.

2.4 Mean_BP Feature Extraction

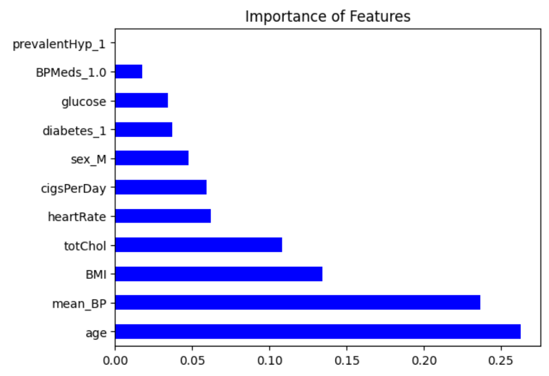
In section 1.22.6, 2 features of the dataset which showed to be very highly correlated were sysBP (Maximum BP) and diaBP (Minimum BP). So a new feature `mean_BP` was created by taking the average of `sysBP` and `diaBP` and old were removed. Dataset with and without this feature were compared, and this feature gave a next big jump in the ROC score to **0.74758**.

2.5 Feature Reduction

Experimentation was done with feature reduction using Gradient Boosting and PCA. Notebooks containing working of this section are `feature_red.ipynb` and `PCA.ipynb`.

2.5.1 Feature Reduction using Gradient Boosting

In section 3.5, while comparing different models it was observed that Gradient Boosting gave the highest AUC-PR. The following images show the feature importance and AUC-PR score vs number of features using Gradient Boosting. The AUC-PR was calculated by fitting Naive Bayes on the training dataframe and calculating the score on the test set.



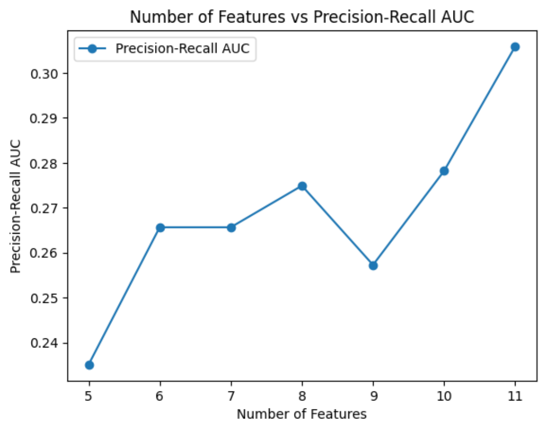


Figure 14: Feature Importance and AUC-PR score after feature reduction

The results show that the `prevelantHyp` feature has no importance for the Gradient Boosting Algorithm and was rightly dropped in the EDA section. Also the maximum AUC-PR score was gained using all features (excluding the ones dropped in section 1), and reducing the features causes a drop in score.

2.5.2 Feature Extraction using PCA

PCA was applied on both the original dataset and the improved datasets by varying the number of components (14, 11, 9, 8) but both gave a drop in score. The working is present in `PCA.ipynb`

2.6 Best Dataset

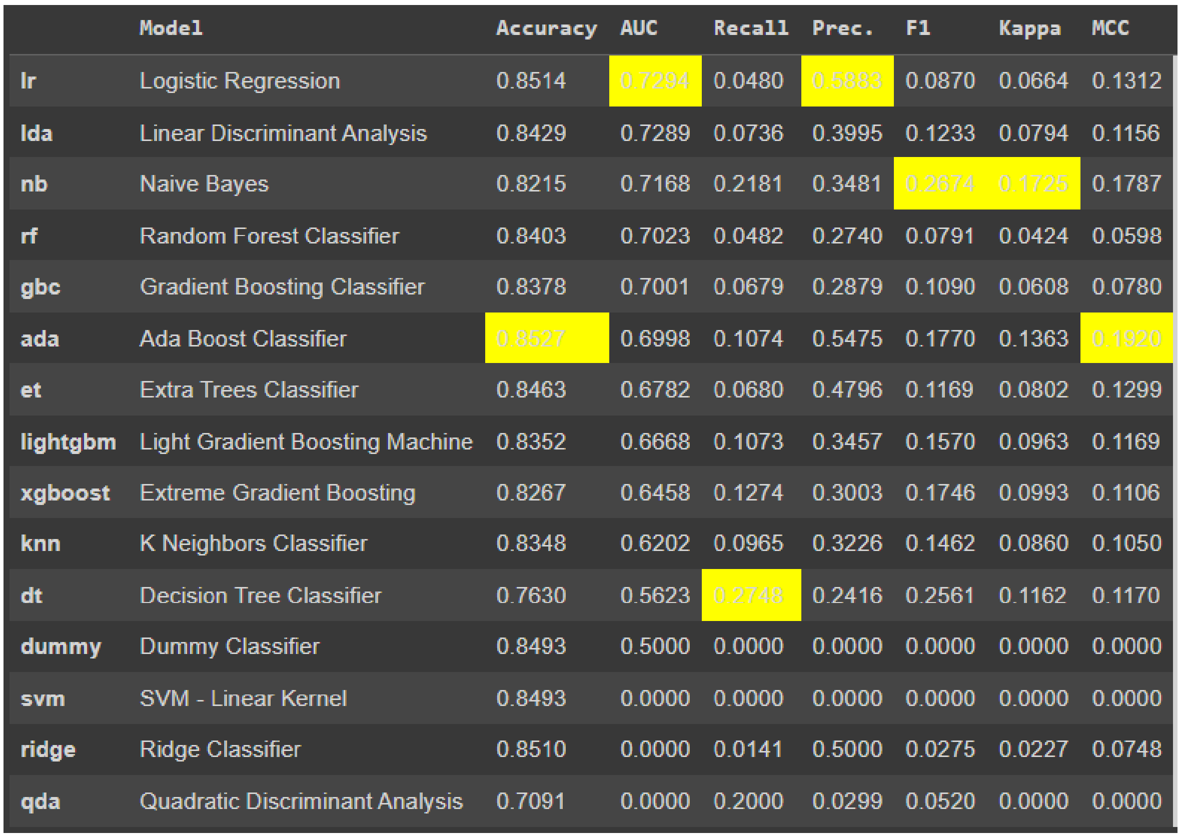
After numerous experiments back and forth, the best data cleaning and feature engineering approach which gave the highest AUC-ROC and AUC-PRC consisted of the following steps:

1. Dropping rows with which have nulls in the `'BPMeds'` column
2. Imputing the `'cigsPerDay'` column with the median of the subset of the column (where `'is_smoking' == 'YES'`
3. Dropping `'id'`, `'education'`, `'is_smoking'`, `'prevalentStroke'` columns
4. Imputing `'totChol'`, `'BMI'`, `'heartRate'`, `'glucose'` column using median
5. Extracting the `'mean_BP'` column by taking the average of `sys_BP` and `dia_BP` and dropping them
6. Winsorizing all numerical columns using optimized thresholds
7. Applying MinMaxScaler()

3 Applying Machine Learning Models

3.1 AutoML using PyCaret

AutoML was run on the best dataset after feature engineering using PyCaret, and the following model comparison was obtained.



	Model	Accuracy	AUC	Recall	Prec.	F1	Kappa	MCC
lr	Logistic Regression	0.8514	0.7294	0.0480	0.5883	0.0870	0.0664	0.1312
lda	Linear Discriminant Analysis	0.8429	0.7289	0.0736	0.3995	0.1233	0.0794	0.1156
nb	Naive Bayes	0.8215	0.7168	0.2181	0.3481	0.2674	0.1725	0.1787
rf	Random Forest Classifier	0.8403	0.7023	0.0482	0.2740	0.0791	0.0424	0.0598
gbc	Gradient Boosting Classifier	0.8378	0.7001	0.0679	0.2879	0.1090	0.0608	0.0780
ada	Ada Boost Classifier	0.8527	0.6998	0.1074	0.5475	0.1770	0.1363	0.1920
et	Extra Trees Classifier	0.8463	0.6782	0.0680	0.4796	0.1169	0.0802	0.1299
lightgbm	Light Gradient Boosting Machine	0.8352	0.6668	0.1073	0.3457	0.1570	0.0963	0.1169
xgboost	Extreme Gradient Boosting	0.8267	0.6458	0.1274	0.3003	0.1746	0.0993	0.1106
knn	K Neighbors Classifier	0.8348	0.6202	0.0965	0.3226	0.1462	0.0860	0.1050
dt	Decision Tree Classifier	0.7630	0.5623	0.2748	0.2416	0.2561	0.1162	0.1170
dummy	Dummy Classifier	0.8493	0.5000	0.0000	0.0000	0.0000	0.0000	0.0000
svm	SVM - Linear Kernel	0.8493	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000
ridge	Ridge Classifier	0.8510	0.0000	0.0141	0.5000	0.0275	0.0227	0.0748
qda	Quadratic Discriminant Analysis	0.7091	0.0000	0.2000	0.0299	0.0520	0.0000	0.0000

Figure 15: Model Comparision form AutoML using PyCaret

The results show that linear models such as Logistic Regression and Naive Bayes give far better results than Ensemble Models such as Random Forest and XGBoost. However these may not be the best models for our objective, as Recall was very low for all models. Recall is important for our model as we want to minimize False Negatives. Given that we need a high recall along with a good balance with precision (low False Positives), along with fact that our dataset is highly imbalanced, We opted to compare AUC-precision recall for our models from here onwards. In order to obtain the best model, 2 approaches were taken: 1. Smote was applied on the training dataset to improve the Target variable's distribution and calculate AUC-PRC, and 2. FInding the right balance between Precision and Recall using different Thresholds. This is further discussed in detail in section 4 along with a detailed description of which model was chosen as the best model and why. However before that Hyperparameter Optimization was conducted of the above models including Neural Networks after which the best ones were ensembled using Soft Voting and Stacking.

3.2 Hyperparameter Optimization

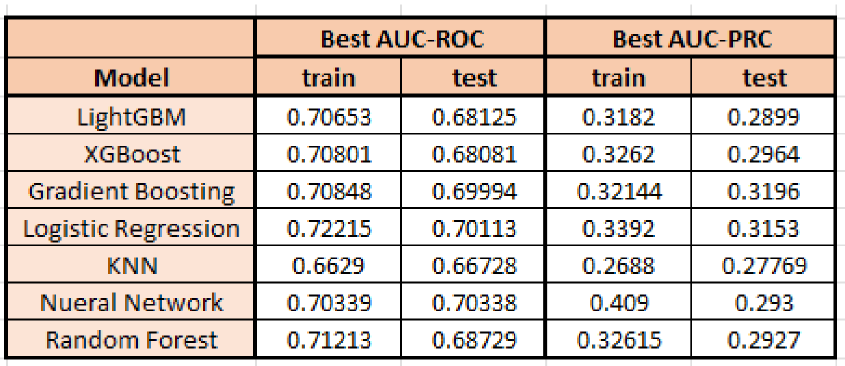
Hyperparamater Optimization was performed of multiple models using `Optuna` and `SkOpt` libraries. Bayesian Otpimization was chosen as it's known to give good results quickly and is a balance between GridSearch and Randomized Search. Scoring metric was set as both ROC-AUC and AUC-PR to see how each model performs for each metric. The dataset was split into Training and Test datasets with 80:20 ratio before optimization, to ensure no data leakage is present. Then each model with its optimized parameters was tested on the test datasets. **RepeatedStratifiedKFold** was done during Optimization as it is a classification dataset and is also imbalanced. Working of this section is in `Optimization_original_roc.ipynb` and `Optimization_original_AUCPRC.ipynb` files. Results are in figure 15.

3.2.1 Optuna TPE and CMA-ES

Hyperparameter optimization was using Optuna TPE (Tree-structured Parzen Estimator) which is a specific form of Bayesian Optimization. CMA-ES (Covariance Matrix Adaptation Evolution Strategy) is an evolutionary algorithm that efficiently searches for optimal solutions in continuous search spaces. Both these methods were applied on LightGBM, XGBoost, and Logistic Regression. However they did not give a better score than optimization through SkOpt. Working is in `Optimization_original_roc.ipynb` file.

3.2.2 SKOpt Bayesian Optimization

Bayesian Optimization was done using SkOpt on multiple models including Neural Network (Keras). For Boosting models Early stopping rounds were set to 10. RepeatedStratifiedKFold was setup to ensure equal distribution of the target variable in every fold. Number of iterations and random starting points were set as 100 for all. For the Neural Netowk RELu was set as the activation function for hidden layers, Sigmoid as the activation function for the output layer, and the loss function was set as Binary Cross Entropy. The Hyper parameters optimized in the Nueral Network were the number of Hidden Layers and the Learning Rate. Results are in the figure below:

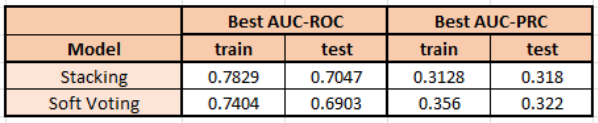


	Best AUC-ROC		Best AUC-PRC	
Model	**train**	**test**	**train**	**test**
LightGBM	0.70653	0.68125	0.3182	0.2899
XGBoost	0.70801	0.68081	0.3262	0.2964
Gradient Boosting	0.70848	0.69994	0.32144	0.3196
Logistic Regression	0.72215	0.70113	0.3392	0.3153
KNN	0.6629	0.66728	0.2688	0.27769
Nueral Network	0.70339	0.70338	0.409	0.293
Random Forest	0.71213	0.68729	0.32615	0.2927

Figure 16: Best Results from Individual Models after Hyperparameter Optimization

3.3 Stacking and Soft Voting

Stackign and Soft Voting were carried out of 5 best models (NN, XGB, GB, RF, LR) of figure 15 as base modes. Adding more models in the base led to more overfitting and a low score on the test set. For soft Voting more weight was given to neural network and logistic regression. The weights that gave the best results were `1.5, 1, 1, 1, 1.5`. For Stacking multiple meta models were tried including XGBoost, Logistic Regression, and Gradient Boost. The meta model which gave the best result was Logistic Regression with default parameters. Working of this section can be found the `Optimization_original_AUPRC.ipynb`. Results are in figure 16.



	Best AUC-ROC		Best AUC-PRC	
Model	train	test	train	test
Stacking	0.7829	0.7047	0.3128	0.318
Soft Voting	0.7404	0.6903	0.356	0.322

Figure 17: Best Results from Soft Voting and Stacking

3.4 Performance Comparison of Models

Each model was run on the training data with repeated StartifiedKFold with 10 splits and 3 iterations, the results of AUC-PR were plotted on a box plot with stripplots. The Red diamond shows the score on the test dataset.

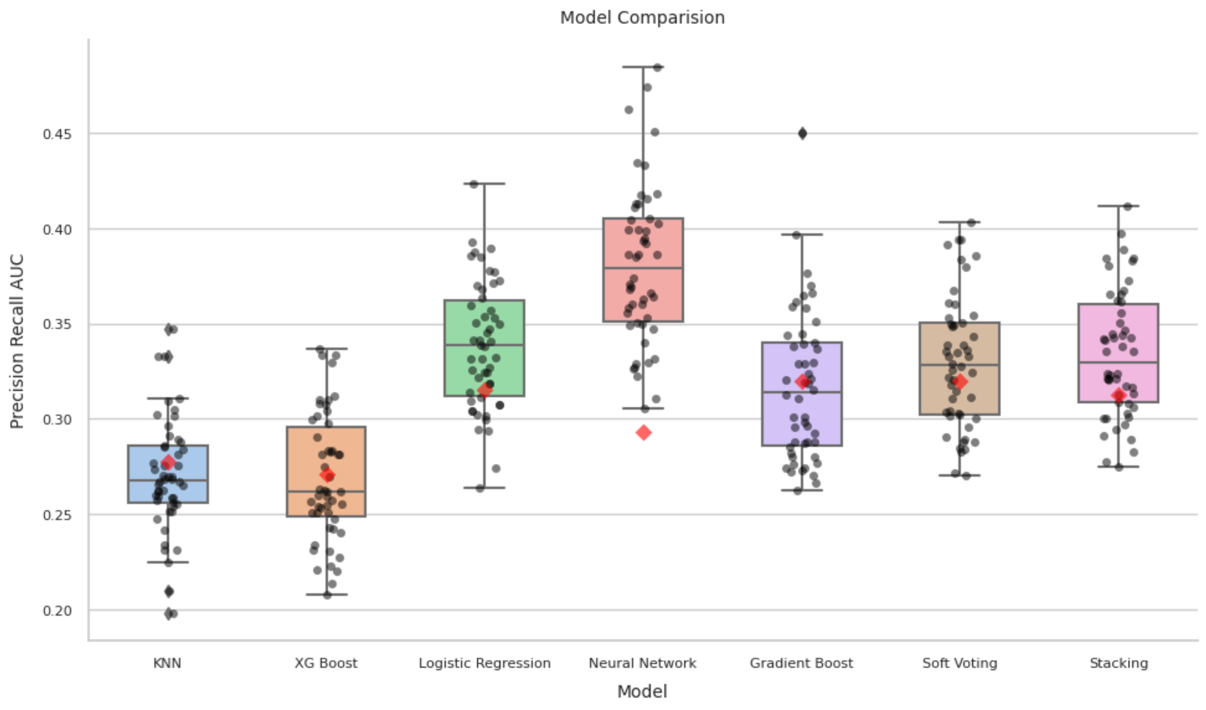


Figure 18: Model Comparision using Boxplot

The results show that Soft Voting had the highest score on the test dataset followed closely by Logisitc Regression and Gradient Boosting. Nueral Network had a good score but overfitted on the training data and had a very high Standard deviation. Stacking and Soft Voting had low standard deviation and comparable results but The following section 4 dives deeper into model comparision and looks at Recall and Precision vs Threshold for each model as well as AUC-PR and ROC-AUC for each of these models.

4 Selecting the Best Model

4.1 Using Probability Threshold

To get a deeper dive into each of figure 15 model's performance, we plotted AUC-PR, ROC-AUC and Precision Recall vs Probability Threshold. Results are in the figures below:

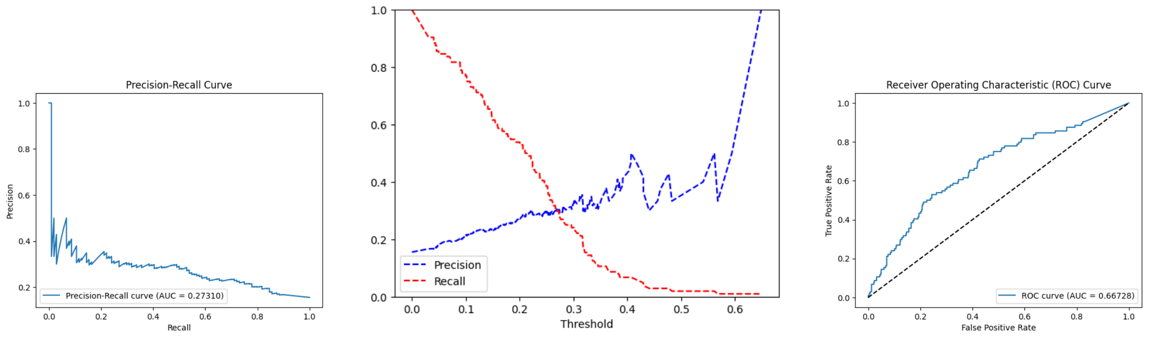


Figure 19: AUC-PR, ROC-AUC and PR vs Probability Threshold for KNN

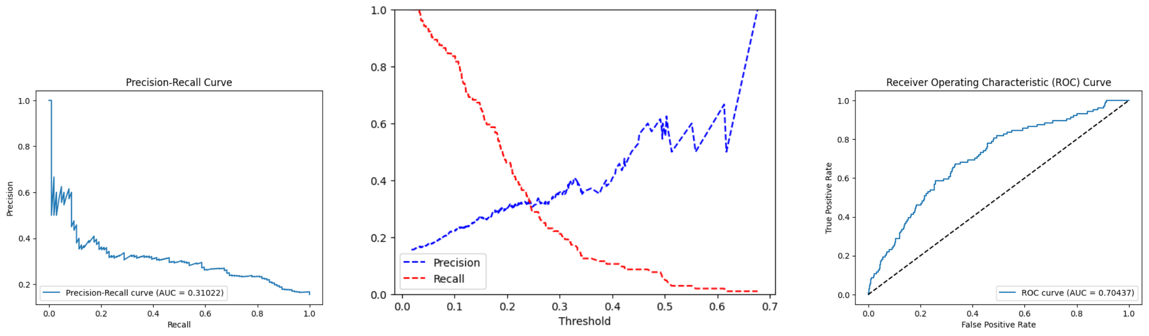


Figure 20: AUC-PR, ROC-AUC and PR vs Probability Threshold for Logistic Regression

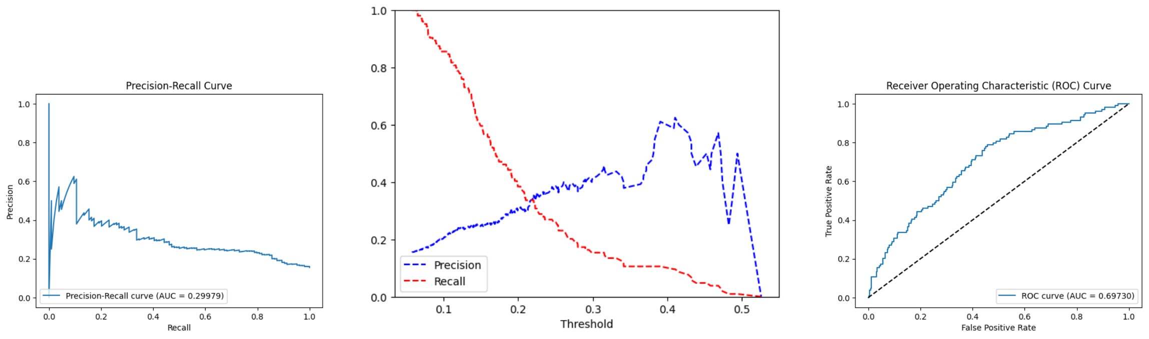


Figure 21: AUC-PR, ROC-AUC and PR vs Probability Threshold for Gradient Boost

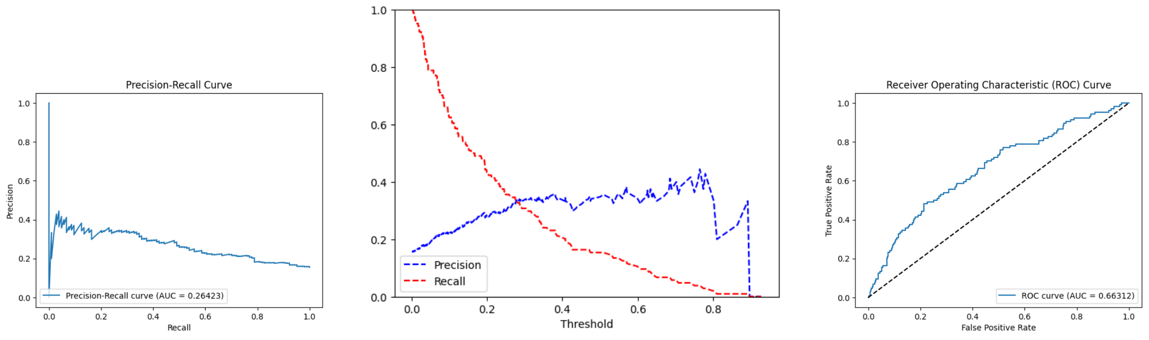


Figure 22: AUC-PR, ROC-AUC and PR vs Probability Threshold for XG Boost

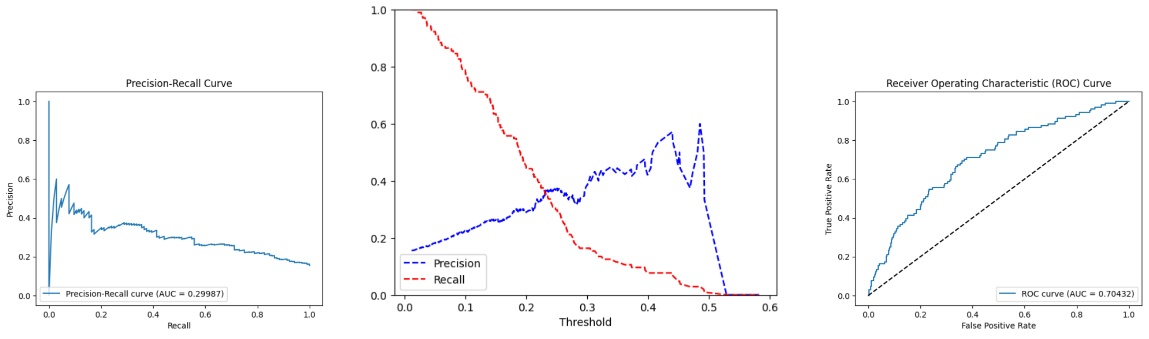


Figure 23: AUC-PR, ROC-AUC and PR vs Probability Threshold for Neural Netowrk

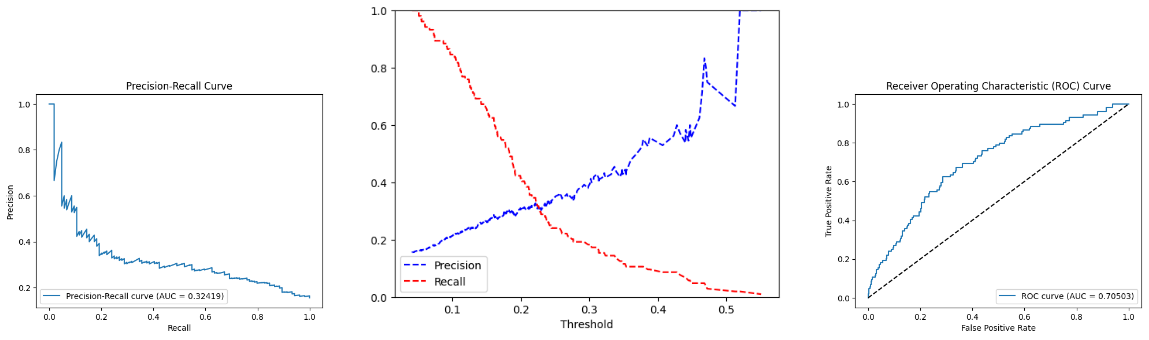


Figure 24: AUC-PR, ROC-AUC and PR vs Probability Threshold for Soft Voting

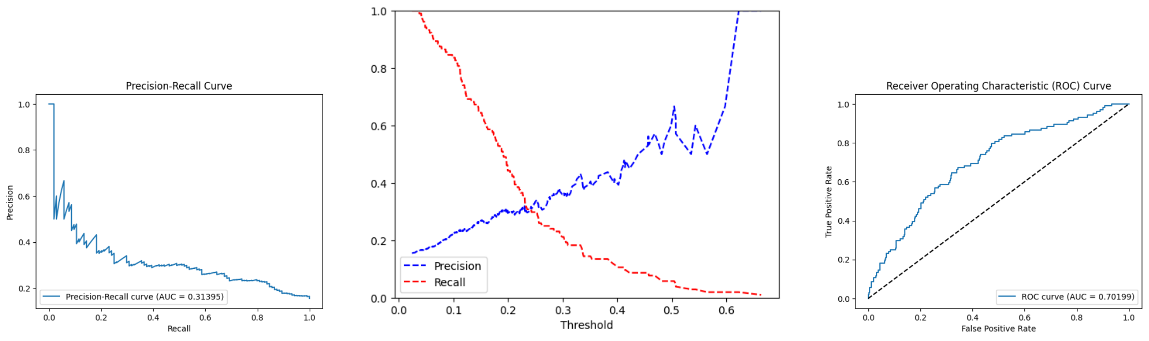


Figure 25: AUC-PR, ROC-AUC and PR vs Probability Threshold for Stacking

4.2 Using SMOTE

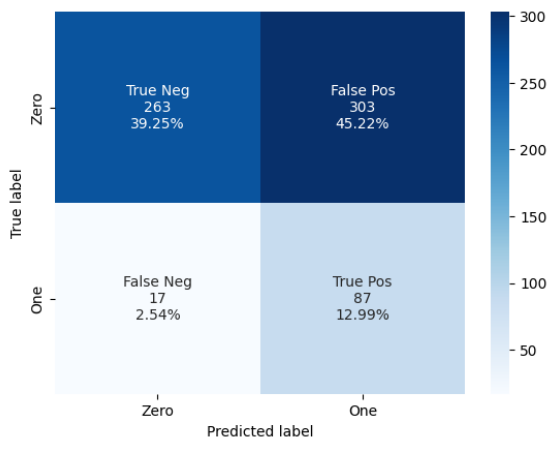
Since the dataset had was highly imbalanced (85% class 0 and 15% class 1), SMOTE was done on the training set (working notebooks are `Thresholds.ipynb` and `SMOTE_and_PyCaret.ipynb` by experimenting with various hyperparameters:

- Default parameters
- Sampling strategy of 0.5
- Oversampling 0.3
- Undersampling 0.3
- 0.2 oversample and 0.5 undersample

After applying SMOTE on the training dataset, AUC-PR was found on the test dataset, however no improvement in score was observed. Score through Logistic Regression after SMOTE was 0.2864.

4.3 Selecting the Best Model

Since the objective of this project was of CardioVascular Risk Prediction, more emphasis was given on getting a high recall. AUC-PR was given improtance over ROC-AUC as correctly identifying individuals at risk of cardiovascular disease is a higher priority and the costs of false positives and false negatives is the objective of this project, and given the imbalanced nature of the dataset this would be more appropriate. Also, ROC-AUC is chosen when we want to evaluate the overall discrimination ability of the model and you want to balance performance across different thresholds.
Getting a high recall In context of CardioVascular Risk Prediction would mean the diagnostic test to correctly detect individuals with the disease, minimizing false negatives. Whereas precision would mean accuracy of a diagnostic test in correctly identifying individuals as positive, reducing false positive results. We want a high recall but not a very low precision as it would lead to unncessary test getting the diagnostic tests' cost high. From figure 15, and figures from section 4.1 it was observed that Logistic Regression gave the best results on both AUC-PR and ROC-AUC and gave the highest test recall. Given also the simplistic and interpretable nature of the model, it was takken as the best model and chosen for deployment. The balance between Recall and precision was chosen using probability threshold as 0.1. The Recall using this threshold was 0.837 and precision was 0.223. The confusion matrix of this model using this threshold is below:



Accuracy=0.522
Precision=0.223
Recall/Sensitivity/TPR=0.837
F1 Score=0.352
False Positive Rate (FPR)=0.061
False Negative Rate (FNR)=0.163
True Negative Rate (TNR)=0.939
Specificity=0.939

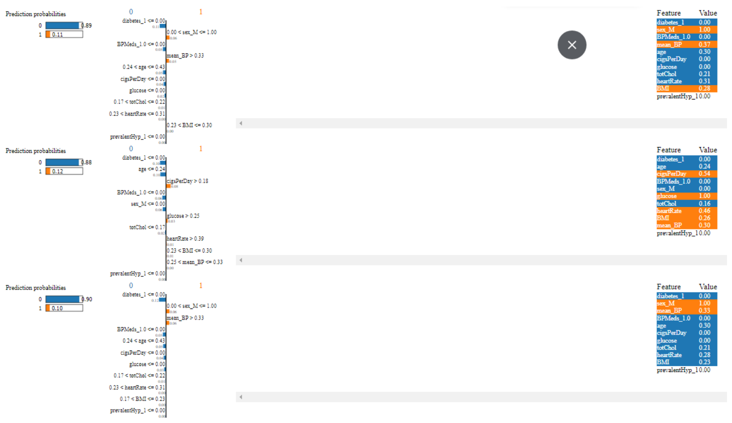
Figure 26: Confusion Matrix of the Best Model (Logistic Regression)

5 Explainable AI

Explainable AI aims to provide interpretable explanations for machine learning models' predictions. 3 methods which enable this are Lime, Shap and Counterfactual. In the following sections Lime and Shap methods are applied on our dataset using our best model (Logistic Regression) and explanations are generated. Working of this section can be found in the notebook `Lime_and_SHAP.ipynb`.

5.1 Lime

LIME, which stands for Local Interpretable Model-agnostic Explanations, is a method that helps explain how a machine learning model makes predictions for specific instances by creating simpler explanations based on locally interpretable models. It helps us understand why a model predicts a certain outcome for a particular input. Explaining individual predictions is important in assessing trusts and also compliments hold out set validations in model selection.
Local Explanations for 3 different instances (3, 8, and 15) were generated and the output is shown in the image below:



Feature	Value
diabetes_1	0.00
sex_M	1.00
BPMeds_1.0	0.00
mean_BP	0.37
age	0.30
cigsPerDay	0.00
glucose	0.00
totChol	0.21
heartRate	0.31
BMI	0.28
prevalentHyp_1	0.00

Feature	Value
diabetes_1	0.00
age	0.24
cigsPerDay	0.54
BPMeds_1.0	0.00
sex_M	0.00
glucose	1.00
totChol	0.16
heartRate	0.46
BMI	0.26
mean_BP	0.30
prevalentHyp_1	0.00

Feature	Value
diabetes_1	0.00
sex_M	1.00
mean_BP	0.33
BPMeds_1.0	0.00
age	0.30
cigsPerDay	0.00
glucose	0.00
totChol	0.21
heartRate	0.28
BMI	0.23
prevalentHyp_1	0.00

Figure 27: Lime explanations for 3 different instances

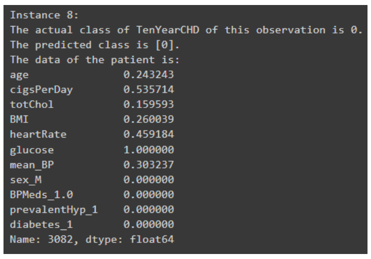
In the output, on the immediate left the bars are showing that all 3 instances are predicting instances 0 which a probability above 0.88. On the immediate right, there are the 5 most important features. The ones highlighted in blue are the ones which are supporitng class 0. On the far right Lime assigns weights to features based on how much each feature is contributing to the output by assigning a feature value with each feature.

5.2 SHAP

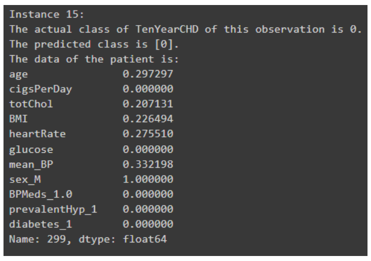
The explainability tool SHAP (SHapley Additive exPlanations) is an explainability tool which provides visualisation of model predictions and allows users to easily understand the contribution of each feature to the final prediction. SHAP provides both local and global interpretability. Local interpretability explains the prediction for a single instance, while global interpretability provides insights into the overall behavior of the model.

5.2.1 Shap Local Analysis

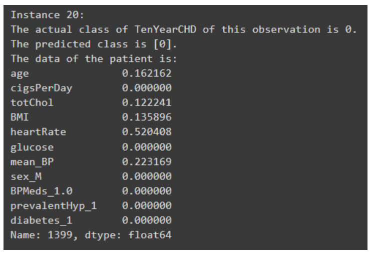
SHAP local analysis focuses on understanding the contributions of individual features to the prediction of a specific instance. Shap values were generated for 3 different instances 8, 15, 20, and a force plot was plotted for each instances. The output results are shown below. In the output each feature value represents the specific value of that feature for the selected instance. These values provide information about the characteristics or measurements of the instance related to each feature.



```
Instance 8:
The actual class of TenYearCHD of this observation is 0.
The predicted class is [0].
The data of the patient is:
age              0.243243
cigsPerDay       0.535714
totChol          0.159593
BMI              0.260039
heartRate        0.459184
glucose          1.000000
mean_BP          0.303237
sex_M            0.000000
BPMeds_1.0       0.000000
prevalentHyp_1   0.000000
diabetes_1       0.000000
Name: 3082, dtype: float64
```



```
Instance 15:
The actual class of TenYearCHD of this observation is 0.
The predicted class is [0].
The data of the patient is:
age              0.297297
cigsPerDay       0.000000
totChol          0.207131
BMI              0.226494
heartRate        0.275510
glucose          0.000000
mean_BP          0.332198
sex_M            1.000000
BPMeds_1.0       0.000000
prevalentHyp_1   0.000000
diabetes_1       0.000000
Name: 299, dtype: float64
```



```
Instance 20:
The actual class of TenYearCHD of this observation is 0.
The predicted class is [0].
The data of the patient is:
age              0.162162
cigsPerDay       0.000000
totChol          0.122241
BMI              0.135896
heartRate        0.520408
glucose          0.000000
mean_BP          0.223169
sex_M            0.000000
BPMeds_1.0       0.000000
prevalentHyp_1   0.000000
diabetes_1       0.000000
Name: 1399, dtype: float64
```

Figure 28: Shap Local Analysis for 3 different instances

Force Plots for individual Observations

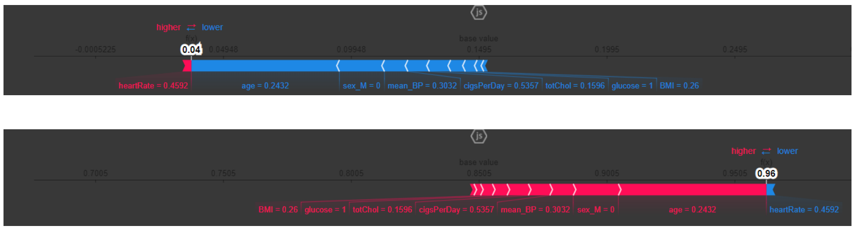


Figure 29: Shap Force Plots for Individual Observations

The above image shows Shap force plots and tell how features contributed to the model's prediction for a specific observation. The first part of the image is for class 1, and the score for this observation is 0.1495. For the second part of the image the score is 0.8505. Scores determine the model's prediction: higher scores predict class 1, lower scores predict class 0. Red and blue features are important for this prediction, with red pushing the score higher and blue pushing it lower.

5.2.2 Shap Global Analysis

SHAP summary dot plot provides an overview of feature importance in a global analysis. It visualizes the SHAP values for each feature, sorted by importance, as dots. The x-axis represents the SHAP value for that feature at each instance, while the color of the dot represents the value of the feature. This plot helps identify important features for the model's predictions and shows how high or low feature values relate to the prediction. Features with a wide spread of SHAP values and a clear relationship between feature value and prediction tend to be more important.

The feature importances plot shows that age is the most important factor for a higher risk of CVD, while BMI and diabetes levels are the least important. This is a similar result as the Feature importances plot in the Gradient Boosting feature importance (section 2.5.1).
The second image, the dot summary plot for class 1 (predicted risk of CVD), there is a clear distribution of the feature values and the SHAP values. High value of age (the red dots) corresponds to a high (positive) SHAP value indicating a prediction in favour of class-1. Similarly, a low value of age corresponds to a lower (negative) SHAP values indicating a predicting in

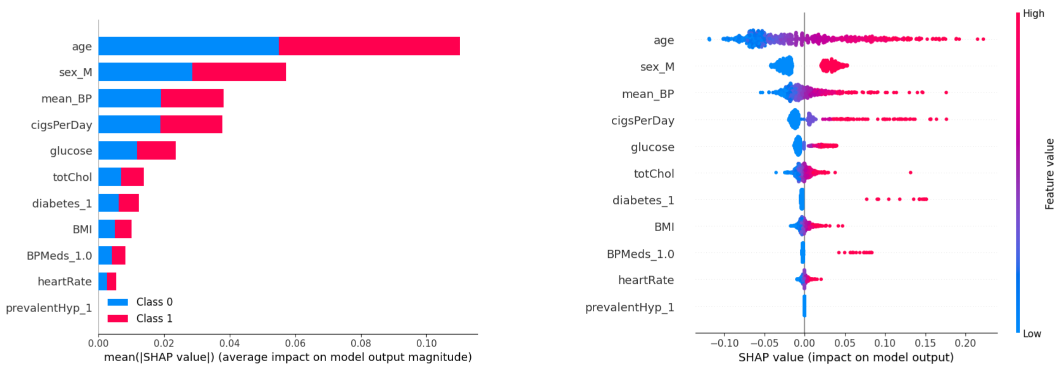


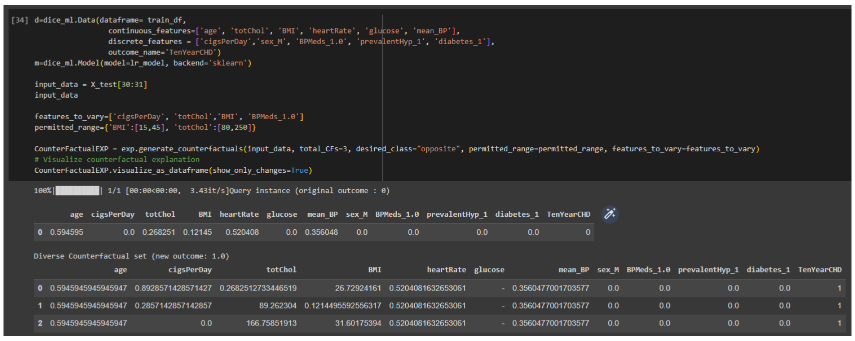
Figure 30: SHAP feature importance and dot plot (Global Analysis)

disfavour of class-1. In summary, the conclusions about the predictions of the model from this plot are:

- Older people have a higher risk of CVD
- Smoking more number of cigarettes increases risk of CVD
- Higher Blood Pressure and Cholestrol values increase risk of CVD
- Patients who take medications for BP have less risk of CVD

5.3 Counterfactual

Counterfactual reasoning helps us understand the smallest changes we can make to data so that it is classified differently by a machine learning model. It provides insights into how the model's predictions can be influenced by altering the input features.
On our dataset we applied Countrfactual using the logistic Regression model. Through manipulating the feature values in our specific situations, we observed how the model reacted to different inputs. We specifically chose certain features to change while keeping others, such as age and gender, constant as altering them wouldnt be logical. This enabled us to understand the decision-making of the model and gain insights into possible interventions or adjustments to achieve desired results. The code and ouput of the counterfactual is shown below:



```
[34] d=dice_ml.Data(dataframe= train_df,
                     continuous_features=['age', 'totChol', 'BMI', 'heartRate', 'glucose', 'mean_BP'],
                     discrete_features = ['cigsPerDay','sex_M', 'BPMeds_1.0', 'prevalentHyp_1', 'diabetes_1'],
                     outcome_name='TenYearCHD')
     m=dice_ml.Model(model=lr_model, backend='sklearn')

     input_data = X_test[30:31]
     input_data

     features_to_vary=['cigsPerDay', 'totChol','BMI', 'BPMeds_1.0']
     permitted_range={'BMI':[15,45], 'totChol':[80,250]}

     CounterFactualEXP = exp.generate_counterfactuals(input_data, total_CFs=3, desired_class="opposite", permitted_range=permitted_range, features_to_vary=features_to_vary)
     # Visualize counterfactual explanation
     CounterFactualEXP.visualize_as_dataframe(show_only_changes=True)

     100%|██████████| 1/1 [00:00<00:00,  3.43it/s]Query instance (original outcome : 0)
```

	age	cigsPerDay	totChol	BMI	heartRate	glucose	mean_BP	sex_M	BPMeds_1.0	prevalentHyp_1	diabetes_1	TenYearCHD
0	0.594595	0.0	0.268251	0.12145	0.520408	0.0	0.356048	0.0	0.0	0.0	0.0	0

```
Diverse Counterfactual set (new outcome: 1.0)
```

	age	cigsPerDay	totChol	BMI	heartRate	glucose	mean_BP	sex_M	BPMeds_1.0	prevalentHyp_1	diabetes_1	TenYearCHD
0	0.5945945945945947	0.8928571428571427	0.2682512733446519	26.72924161	0.5204081632653061	-	0.3560477001703577	0.0	0.0	0.0	0.0	1
1	0.5945945945945947	0.2857142857142857	89.262304	0.1214495592556317	0.5204081632653061	-	0.3560477001703577	0.0	0.0	0.0	0.0	1
2	0.5945945945945947	0.0	166.75851913	31.60175394	0.5204081632653061	-	0.3560477001703577	0.0	0.0	0.0	0.0	1

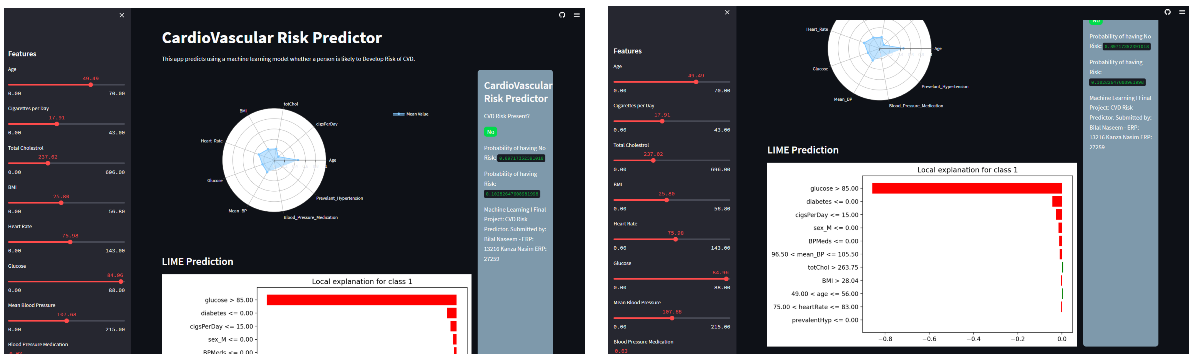
Figure 31: Counterfactual code and output

The code is used to generate a diverse set of counterfactual examples which aim to achieve the opposite outcome compared to the original instance. Using this we can explore the effects of modifying certain features on the predicted outcome. The Output table presents the modified feature values for each counterfactual example, highlighting the changes made. 1 represents the desired opposite outcome.By examining these counterfactual examples, we can gain insights into the relationship between specific feature values and the predicted outcome.

6 Model Deployment on Streamlit & Conclusion

The Best model of section 4.3 (Logistic Regression) was deployed on Streamlit. The best approach from jupyter notebooks was converted to production code in VS Code using a python virtual environment and all required packages and dependencies were installed in it. The app takes in inputs of continuous features using a slider and for discrete values using buttons and in the output displays a Spider Plot (Displaying the value of each feature wrt. to the mean of other features) along with the Prediction of having risk of CVD or No Risk. The App also shows a Lime Local Analysis Plot highlighting how much each feature is contributing to the predicted output.
The app can be accessed by cloning the GitHub Repository and running `streamlit app.py`. The App is also deployed on Streamlit cloud and can also be accessed from there. Links of both are mentioned below:

- GitHub Repository Link
- Streamlit app Link



CardioVascular Risk Predictor

This app predicts using a machine learning model whether a person is likely to Develop Risk of CVD.

CardioVascular Risk Predictor

CVD Risk Present? No

Probability of having No Risk: 0.89717352391018

Probability of having Risk: 0.10282647608981998

Machine Learning I Final Project: CVD Risk Predictor. Submitted by: Bilal Naseem - ERP: 13216 Kanza Nasim ERP: 27259

Figure 32: CardioVascular Risk Predictor Streamlit App

Conclusion

In our machine learning project, we successfully poerformed EDA, Feature Engineering, developed mueltiple ensemble models, optimized them, assessed and compared them using the most suitabe metric as per our objective, and deployed it using an interactive app on Streamlit to predict the likelihood of a person having CVD. To enhance interpretation, we used explanation methods such as LIME, Shapley, and Counterfactual Techniques.
Furthermore, leveraging additional features and consulting domain experts may provide further improvements to the model's predictive capabilities.