Capstone Project - 3

Supervised – ML – Cardiovascular Risk Prediction

Sri Harish A

Contents

Addressing the problem

The purpose of this project is to try a machine learning approach for **Cardiovascular Risk Prediction** by given the id, sex, and information about the health condition of the person. This project contains: Exploratory data analysis, feature engineering, choosing appropriate features, cross algorithms, cross validation, tuning the algorithms, analysis of feature importance, analysis of model performance. The predictions of future could help for saving a person's life and could help them to get rid off unhealthy habits. Another point of view is to test the machine learning algorithms how good are at solving this problem.

Addressing the problem

How Cardiovascular Heart Disease Risk Prediction helps?

Cardiovascular disease is the leading cause of death worldwide and a major public health concern. Therefore, its risk assessment is crucial to many existing treatment guidelines.

Risk estimates are also being used to predict the magnitude of future cardiovascular disease mortality and morbidity at the population level and in specific subgroups to inform policymakers and health authorities about these risks. Additionally, risk prediction inspires individuals to change their lifestyle and behavior and to adhere to medications.

Why CHD Risk prediction is important?

The disease is caused when the heart's blood vessels, the coronary arteries, become narrowed or blocked and can't supply enough blood to the heart. This can cause a heart attack or angina. The good news is that if we tackle these risk factors, we can reduce our numbers of people developing heart disease.

By CHD risk prediction, we can analyze the risk factors which are contributing more to develop a heart disease.

The ability to predict risk for and from cardiovascular disease is increasingly important for several reasons. Perhaps foremost among these is the need to determine individual risk to better plan investigation and management with the greatest accuracy and safety for patients and lowest costs for the health care system. This will be even more important as personalized medicine becomes more widespread.

Features Summary

Independent variables:

- **Id** – Unique identification number for each patient.
- **Age** – Age of the patient(Continuous).
- **Education** – Educational qualification of the patients(ordinal).
- **Sex** – Male or Female('M' or 'F').
- **Is_smoking** – Whether or not the patient is a current smoker(Binary).
- **CigsPerDay** – The number of cigarettes that the patient smoked on average in one day(Continuous).
- **BPMeds** – Whether or not the patient was on blood pressure medication(Nominal).
- **prevalentStroke** – Whether or not the patient previously had a stroke(Nominal).

Features Summary (continued)

Independent variables:

- **prevalentHyp** – Whether or not the patient was hypertensive(Nominal).
- **Diabetes** – Whether or not the patient had diabetes(Nominal).
- **Tot_Chol** – Total Cholesterol level(Continuous).
- **Sys_BP** – Systolic blood pressure(Continuous).
- **Dia_BP** – Diastolic blood pressure(Continuous).
- **BMI** – Body Mass Index(Continuous).
- **Heart rate** – The speed at which the heart beats(Continuous).
- **Glucose** – Glucose level in blood(Continuous).

Dependent variable:

- **TenYearCHD** – Whether the patient will develop a risk of Cardiovascular Disease in a 10 years(Binary).

Outliers

- An outlier is an **extremely high or extremely low data point** relative to the nearest data point and the rest of the neighboring co-existing values in a data graph or dataset you're working with.
- **Ways to detect outliers:**
- Interquartile range
- Box plot
- Scatter plot
- Z – score
- In the given dataset we have used Box plot to detect outliers.

Outliers (continued)

Z–Score – To handle outliers.

```
# Z Score based technique to remove outliers
lst = ['cigsPerDay','totChol','sysBP','diaBP','BMI','heartRate','glucose']
for i in lst:
  lower_limit = df[i].mean() - 3*df[i].std()
  print(i+'Lower_limit:',round(lower_limit,2))
  upper_limit = df[i].mean() + 3*df[i].std()
  print(i+'Upper_limit:',round(upper_limit,2))
  df[i] = np.where(df[i]>upper_limit,upper_limit,np.where(df[i]<lower_limit,lower_limit,df[i]))
```

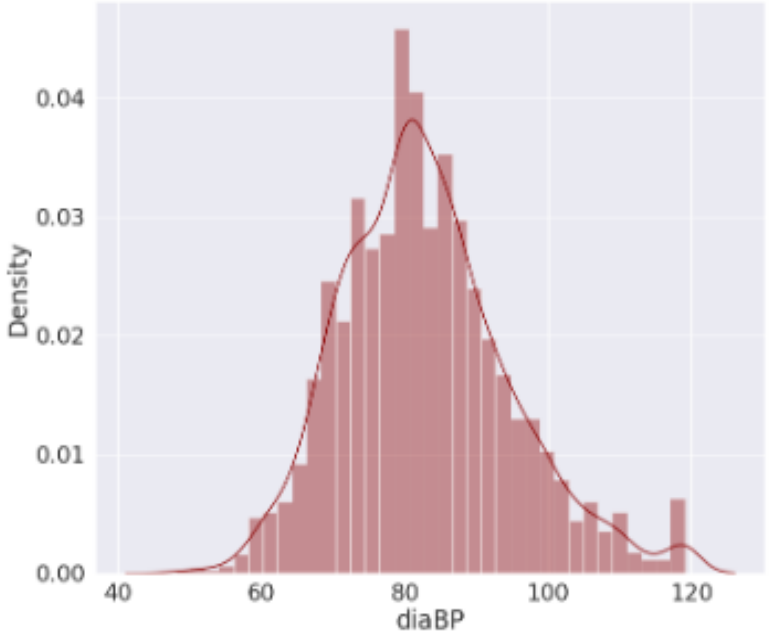
NULL value treatment

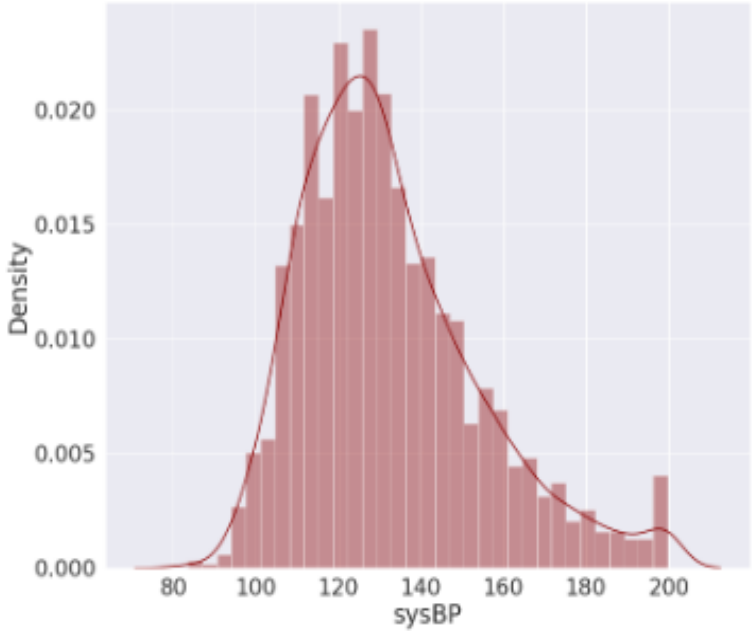
Null values are imputed with most closest value as possible by grouping features

```
# Treating NULL values
# 'education'
df['education'].fillna(0,inplace = True)
df.groupby('education').agg({'age':'mean'})
# print(df.groupby('education').agg({'age':'mean'}))
# From this we can replace education '0' with '1' because based on the average the education varies.
df['education'] = df['education'].apply(lambda x : 1 if x == 0 else x)

# 'cigsPerDay'
df[df['is_smoking'] == 1].groupby('sex').agg({'cigsPerDay':'mean'})
# print(df[df['is_smoking'] == 1].groupby('sex').agg({'cigsPerDay':'mean'}))
# As the average number of cigarettes consumed by Females is 14 and Males is 22
df.loc[ (df.sex == 0) & (df['cigsPerDay'].isna()), 'cigsPerDay'] =  14
df.loc[ (df.sex == 1) & (df['cigsPerDay'].isna()), 'cigsPerDay'] =  22

# 'glucose'
df.groupby('diabetes').agg({'glucose':'mean'})
# print(df.groupby('diabetes').agg({'glucose':'mean'}))
# Average Glucose level of Patients without Diabetes is 79 and with Diabetes is 172
df.loc[ (df.diabetes == 0) & (df['glucose'].isna()), 'glucose'] =  79
df.loc[ (df.diabetes == 1) & (df['glucose'].isna()), 'glucose'] =  172

# 'BPMeds'
df.loc[(df.prevalentHyp == 1) & (df['BPMeds'].isna()), 'BPMeds'] =  1
```

Exploratory Data Analysis

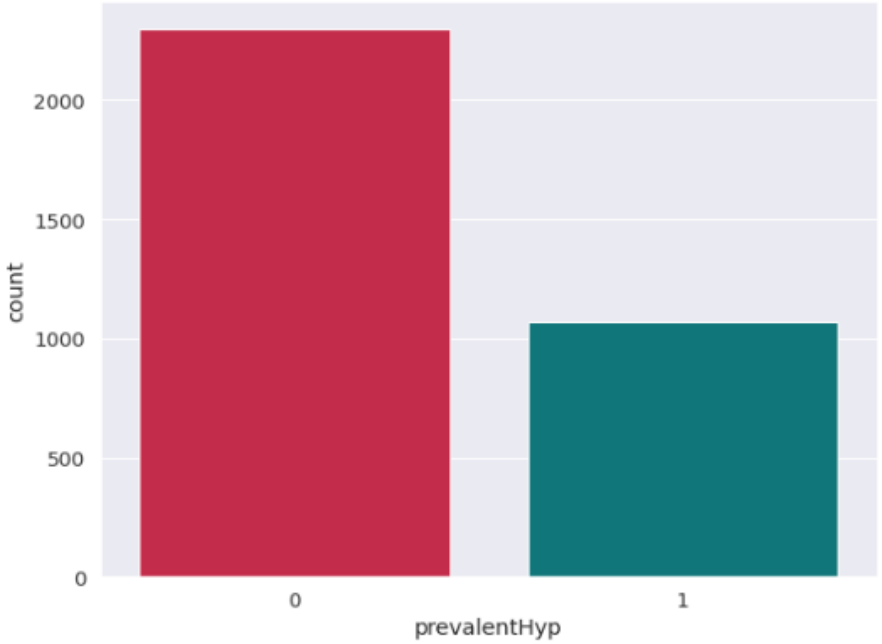
Distribution plot for Numeric independent variable.

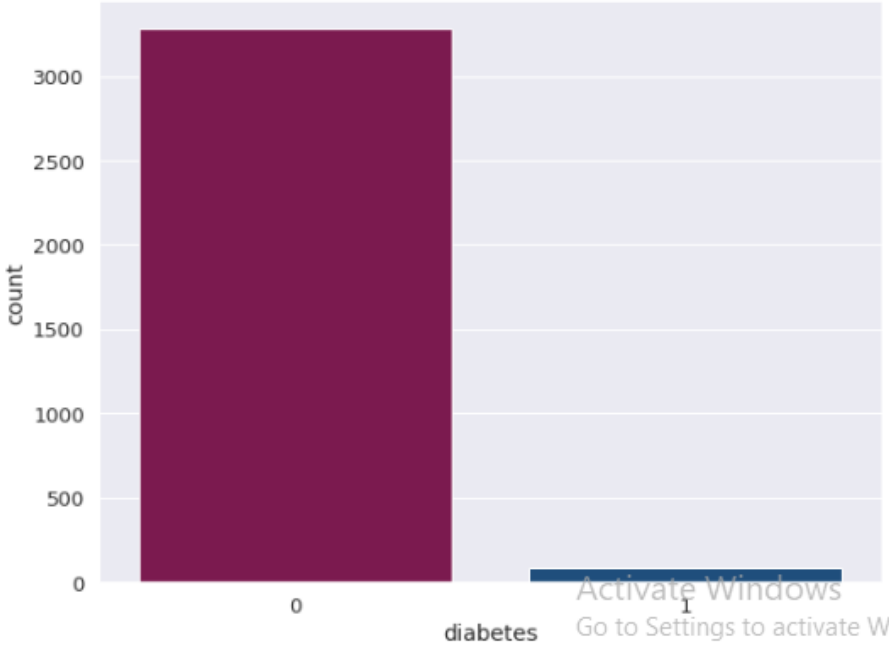




Exploratory Data Analysis (continued)

Count plot for Categorical independent variables.

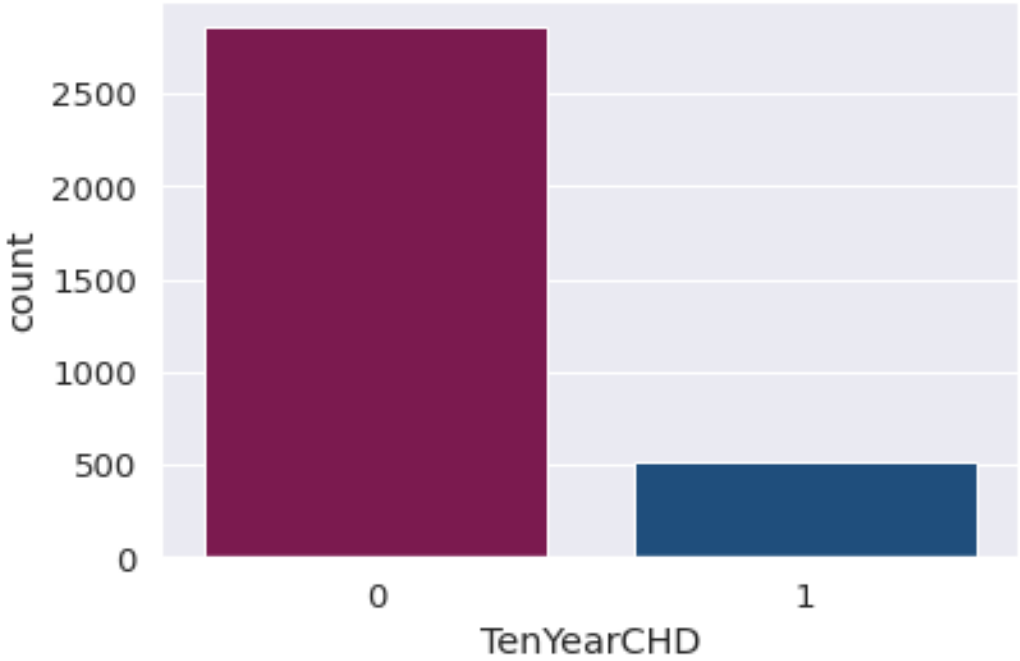




Exploratory Data Analysis (continued)

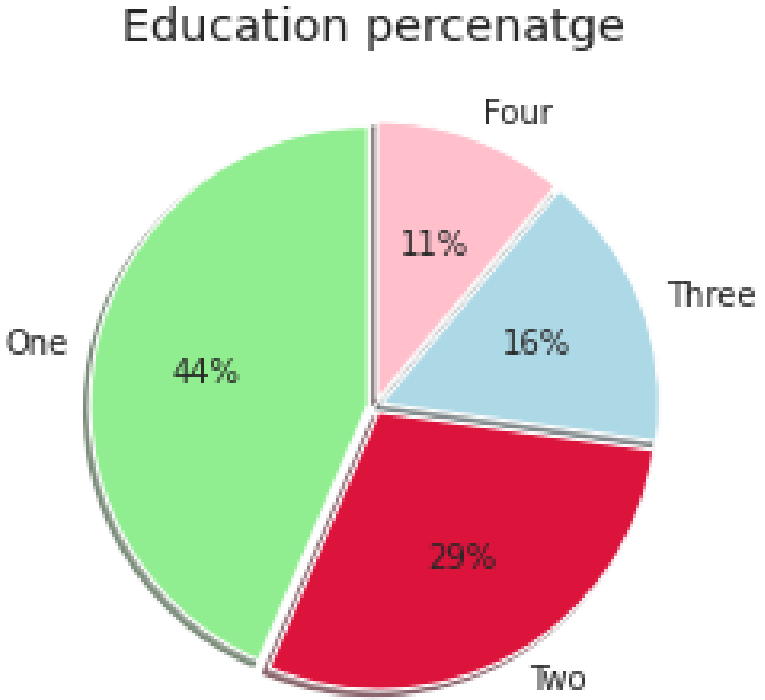
Count plot for dependent variable.

Data Imbalance:
From this we could clearly see the data imbalance.



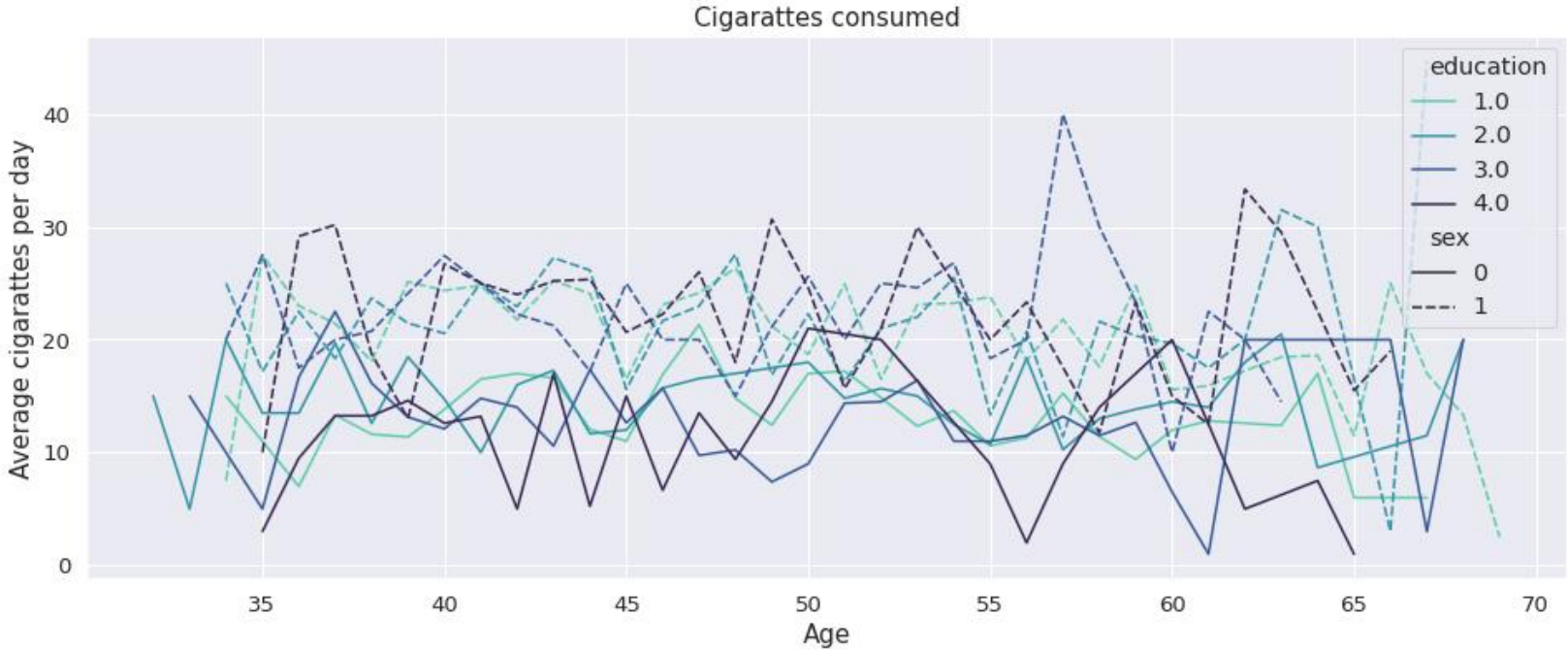
Exploratory Data Analysis (continued)

Pie chart for Categorical independent variable



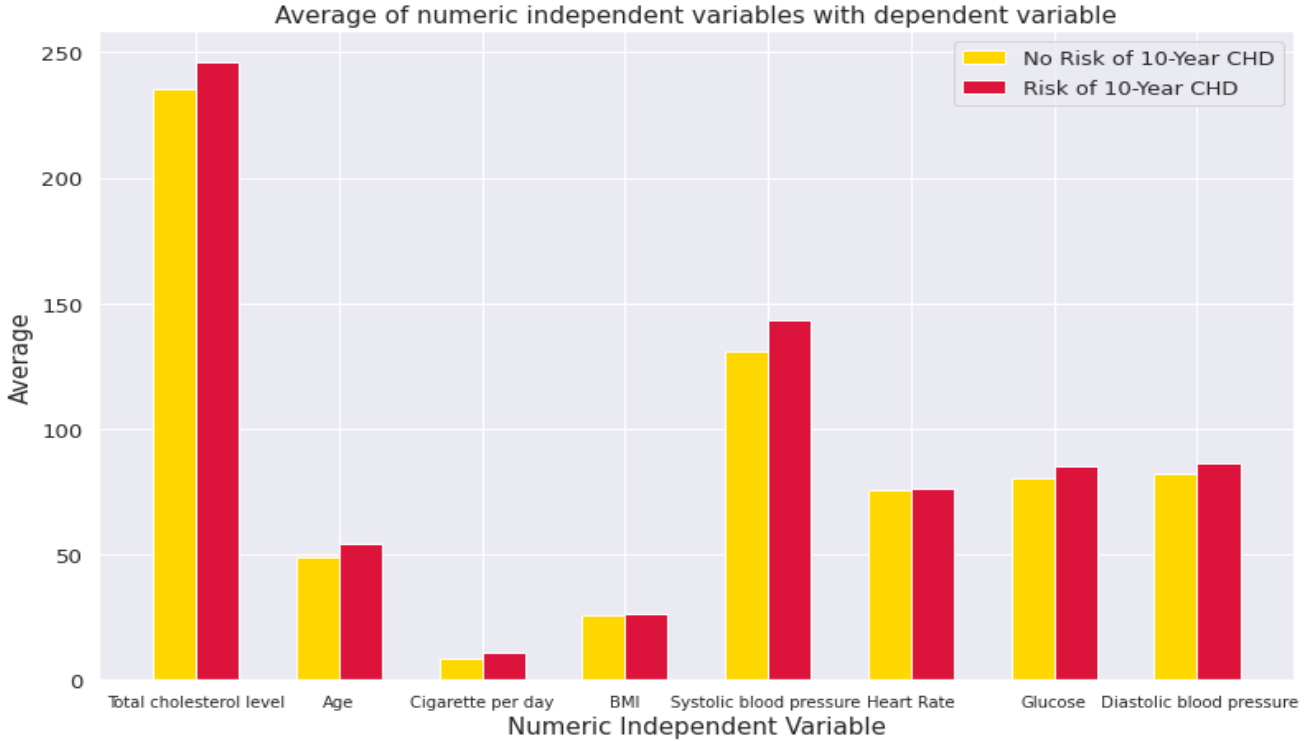
Exploratory Data Analysis (continued)

Line plot for Cigarettes consumption



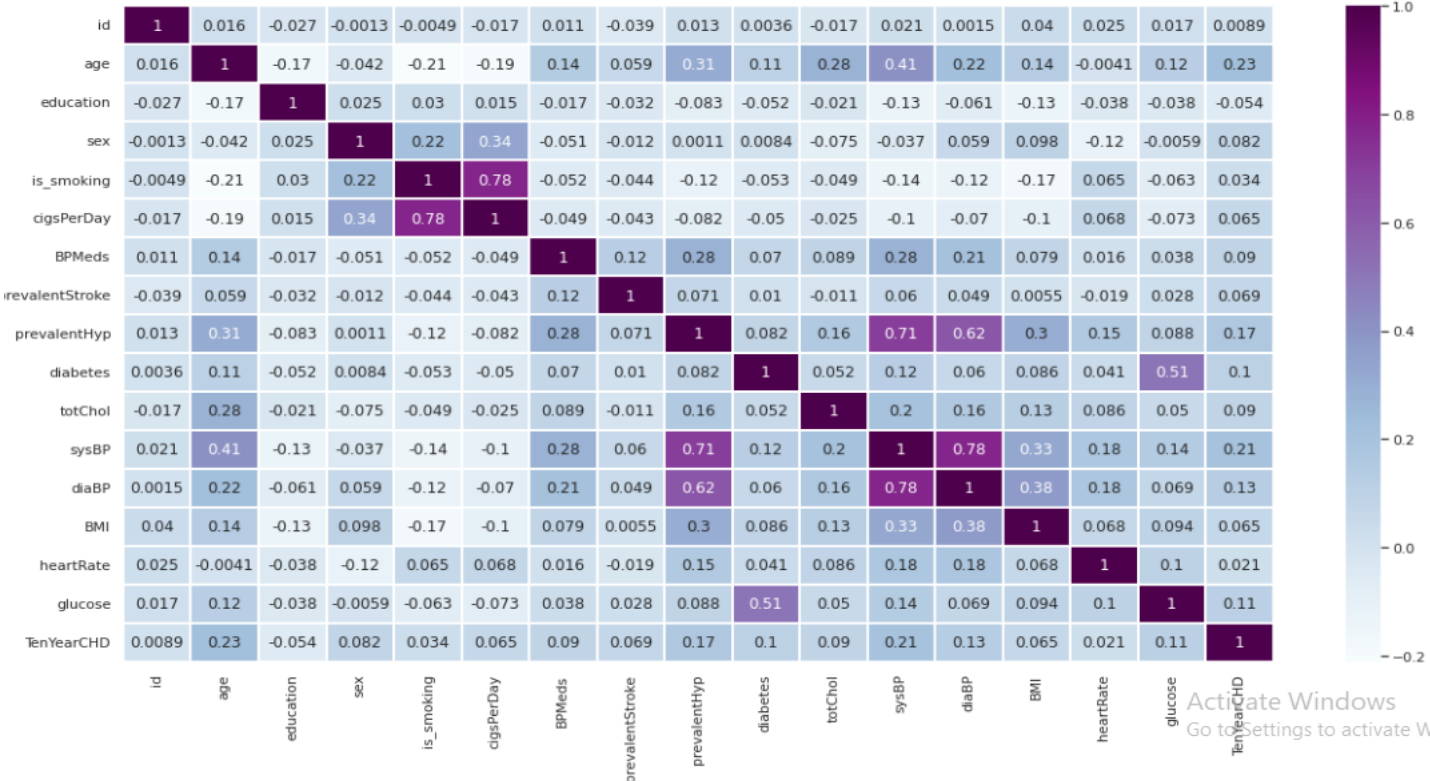
Exploratory Data Analysis (continued)

Bar plot for comparison between patients with and without risk of CHD.



Exploratory Data Analysis (continued)

Multicollinearity



Feature Engineering

- **The Metric Trap**
- One of the major issues when dealing with unbalanced datasets relates to the metrics used to evaluate their model. Using simpler metrics like accuracy score can be misleading. In a dataset with highly unbalanced classes, the classifier will always “predict” the most common class without performing any analysis of the features and it will have a high accuracy rate, obviously not the correct one.
- **Imbalance in our dataset**

```
No Risk of 10-Year CHD:  84.87 %
Risk of 10-Year CHD:  15.13 %
```

Feature Engineering (continued)

To encounter imbalance dataset following sampling techniques are used:

Random Over Sampling

- Randomly duplicate examples in the minority class.

Random Under Sampling

- Randomly delete examples in the majority class.

Synthetic Minority Oversampling Technique (SMOTE)

New examples are synthesized from the existing examples. This is a type of data augmentation for the minority class.

SMOTETomek

A hybrid method which is a mixture of under-sampling method (Tomek) with an oversampling method (SMOTE).

Feature Engineering (continued)

For our dataset Random Over Sampling is found to be effective.

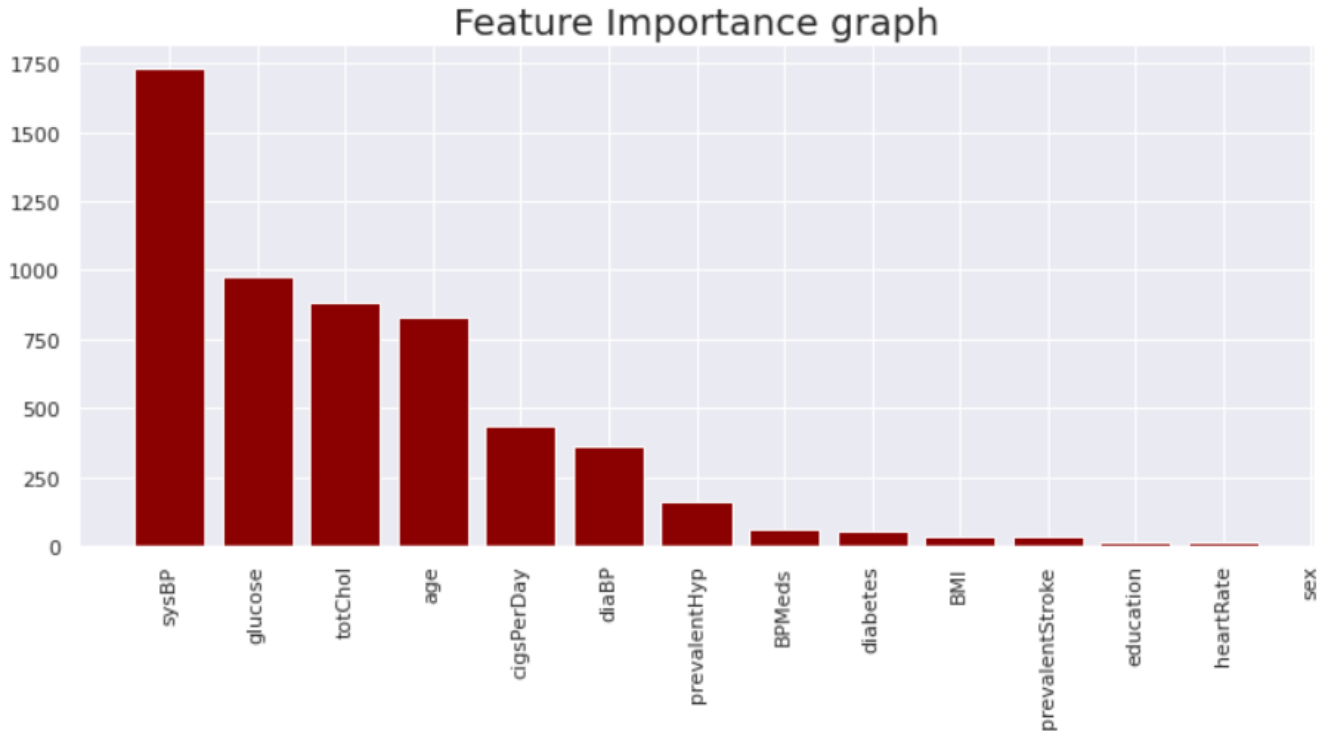
```
# Random Over Sampling
from imblearn.over_sampling import RandomOverSampler

ros = RandomOverSampler(sampling_strategy= 1.0,random_state = 0)
X_ros,y_ros = ros.fit_resample(data.iloc[:,0:-1], data['TenYearCHD'])
print('X ros :',X_ros.shape)
print('y ros :',y_ros.shape)
```

```
# Classes after sampling
ros_xy['TenYearCHD'].value_counts()
```

```
1    2856
0    2856
Name: TenYearCHD, dtype: int64
```

Feature Engineering (continued)

Feature Selection



Preparing dataset for modelling

Train Test Split

Train test split is a model validation procedure that allows you to simulate how a model would perform on new/unseen data.

```
X = data[independent_variables].values
y = data[dependent_variable].values
```

```
# Splitting the dataset into the Training set and Test set
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size = 0.25, random_state = 0)
```

- X_train: (2523, 15)
- X_test : (842, 15)
- y_train : (2523, 1)
- y_test : (842, 1)

Preparing dataset for modelling

Standardization

Standardizing a dataset involves rescaling the distribution of values so that the mean of observed values is 0 and the standard deviation is 1.

```
# Transforming data

# Standardization
scaler = MinMaxScaler()
X_train = scaler.fit_transform(X_train)
X_test= scaler.transform(X_test)
```

Applying Model

Logistic Regression

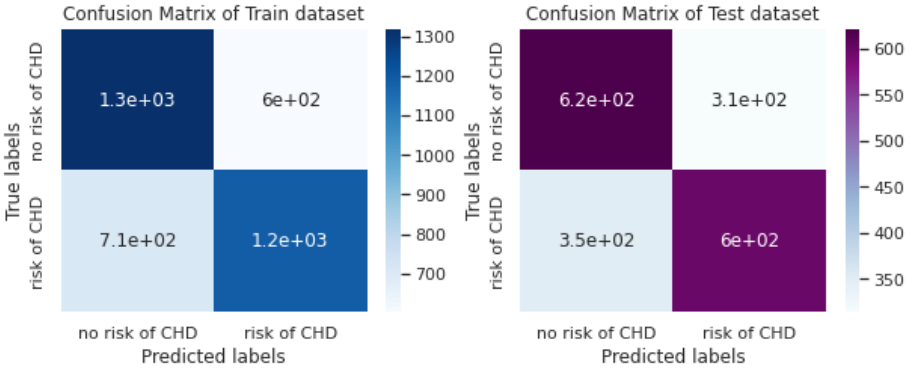
```
# Logistic Regression
# Creating an instance of Logistic Regression
logistic = LogisticRegression(fit_intercept = True)

# Hyperparameters
grid = {'C':10.0**np.arange(-2,3),'penalty':['l1','l2']}

# KFold
cv = KFold(n_splits = 5, random_state = None, shuffle = False)

# Grid Search
clf = GridSearchCV(logistic,grid,cv = cv,n_jobs = -1,scoring = 'f1_macro')

# Model fit
clf.fit(X_train, y_train)
```

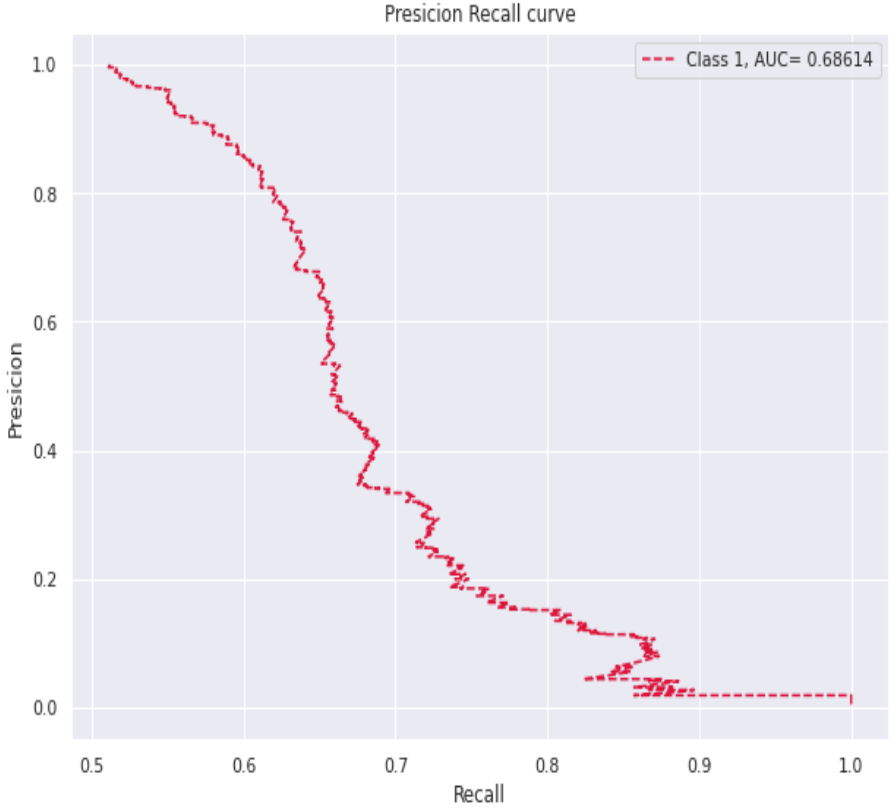
```
GridSearchCV(cv=KFold(n_splits=5, random_state=None, shuffle=False),
             estimator=LogisticRegression(), n_jobs=-1,
             param_grid={'C': array([1.e-02, 1.e-01, 1.e+00, 1.e+01, 1.e+02]),
                         'penalty': ['l1', 'l2']},
             scoring='f1_macro')
```

Applying Model

Model Validation(Logistic regression)

```
The ROC AUC score on train data is           : 0.71434
The ROC AUC score on test data is            : 0.69948
The Precision Recall score on train data is : 0.70048
The Precision Recall score on test data is   : 0.68614
              precision    recall  f1-score   support

           0       0.64      0.66      0.65       935
           1       0.66      0.63      0.64       950

    accuracy                           0.65      1885
   macro avg       0.65      0.65      0.65      1885
weighted avg       0.65      0.65      0.65      1885
```





Applying Model (continued)

Random Forest Classifier

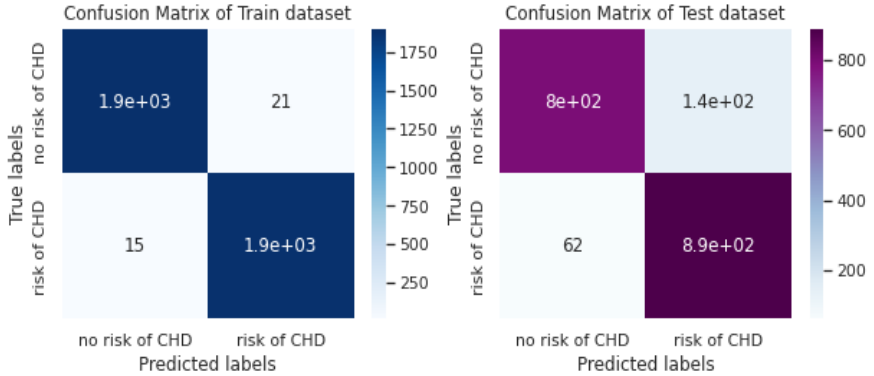
```
# Create an instance of the RandomForestClassifier
rfc_model = RandomForestClassifier(criterion = 'gini',class_weight = {0:1,1:1.5},random_state = 0)

# By adding weights to Class 1, False Negative can be reduced

# Grid search
rfc_grid = GridSearchCV(estimator=rfc_model,param_grid = param_dict,cv = 5, verbose=2, scoring='f1_macro')

# Model Fit
rfc_grid.fit(X_train,y_train)
```

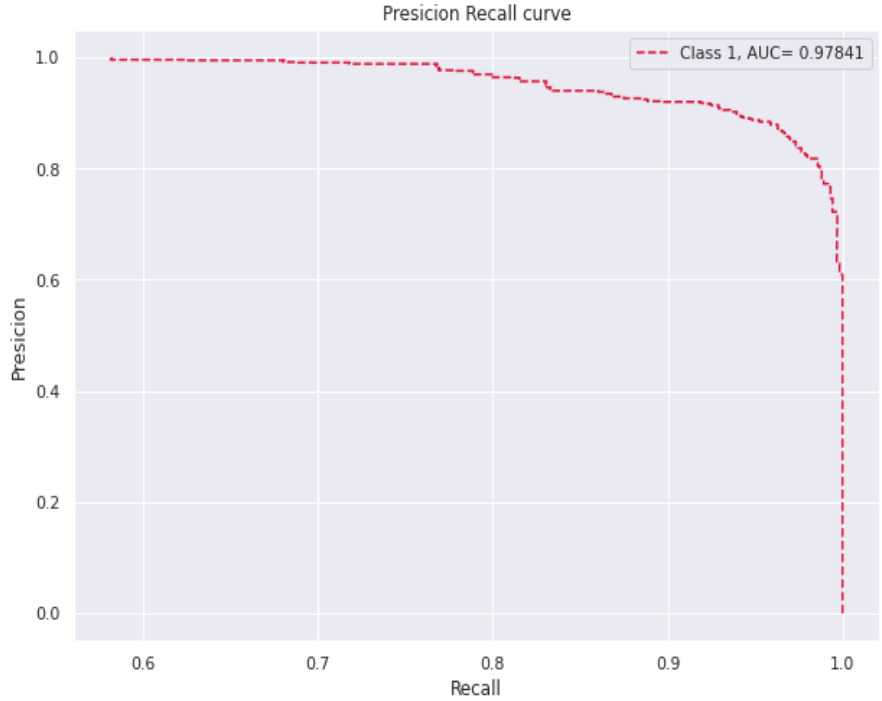
```
Fitting 5 folds for each of 1 candidates, totalling 5 fits
[CV] END max_depth=22, min_samples_leaf=3, min_samples_split=4, n_estimators=100; total time=   0.5s
[CV] END max_depth=22, min_samples_leaf=3, min_samples_split=4, n_estimators=100; total time=   0.5s
[CV] END max_depth=22, min_samples_leaf=3, min_samples_split=4, n_estimators=100; total time=   0.5s
[CV] END max_depth=22, min_samples_leaf=3, min_samples_split=4, n_estimators=100; total time=   0.5s
[CV] END max_depth=22, min_samples_leaf=3, min_samples_split=4, n_estimators=100; total time=   0.5s
GridSearchCV(cv=5,
             estimator=RandomForestClassifier(class_weight={0: 1, 1: 1.5},
                                              random_state=0),
             param_grid={'max_depth': [22], 'min_samples_leaf': [3],
                         'min_samples_split': [4], 'n_estimators': [100]},
             scoring='f1_macro', verbose=2)
```

Applying Model (continued)

Model Validation(Random forest classifier)

```
The ROC AUC score on train data is           : 0.99948
The ROC AUC score on test data is            : 0.97413
The Precision Recall score on train data is : 0.99952
The Precision Recall score on test data is   : 0.97841
              precision    recall  f1-score   support

           0       0.93      0.85      0.89       935
           1       0.86      0.93      0.90       950

    accuracy                           0.89      1885
   macro avg       0.90      0.89      0.89      1885
weighted avg       0.90      0.89      0.89      1885
```





Applying Model (continued)

Gradient Boost Classifier

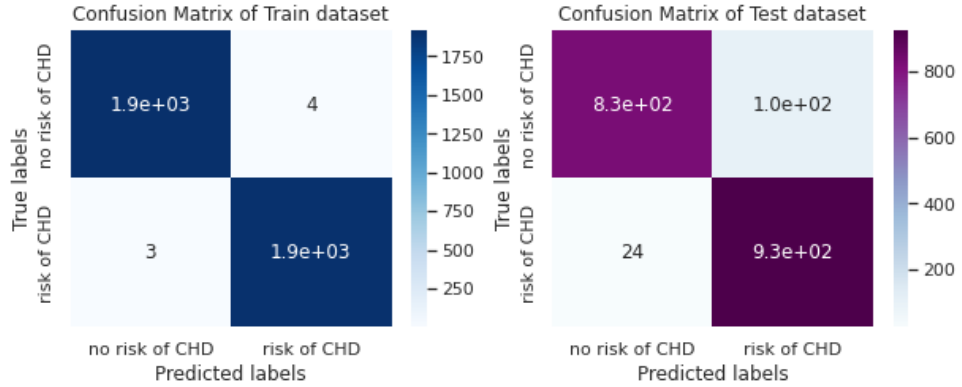
```
# Create an instance of the RandomForestClassifier
gbc_model = GradientBoostingClassifier(random_state = 0)

# Grid search
gbc_grid = GridSearchCV(estimator=gbc_model,param_grid = param_dict,cv = 5, verbose=2, scoring='roc_auc')

# Model Fit
gbc_grid.fit(X_train,y_train)
```

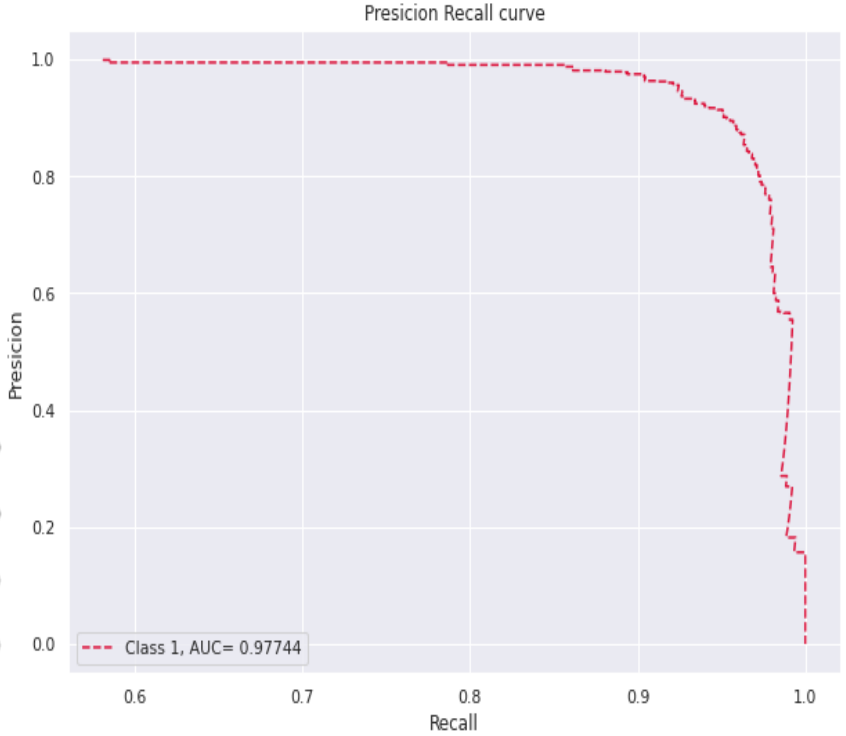
```
Fitting 5 folds for each of 1 candidates, totalling 5 fits
[CV] END max_depth=50, min_samples_leaf=70, min_samples_split=60, n_estimators=200; total time=   3.1s
[CV] END max_depth=50, min_samples_leaf=70, min_samples_split=60, n_estimators=200; total time=   3.0s
[CV] END max_depth=50, min_samples_leaf=70, min_samples_split=60, n_estimators=200; total time=   3.0s
[CV] END max_depth=50, min_samples_leaf=70, min_samples_split=60, n_estimators=200; total time=   3.0s
[CV] END max_depth=50, min_samples_leaf=70, min_samples_split=60, n_estimators=200; total time=   3.1s
GridSearchCV(cv=5, estimator=GradientBoostingClassifier(random_state=0),
             param_grid={'max_depth': [50], 'min_samples_leaf': [70],
                         'min_samples_split': [60], 'n_estimators': [200]},
             scoring='roc_auc', verbose=2)
```

Applying Model (continued)

Model Validation(Gradient Boost classifier)

```
The ROC AUC score on train data is           : 0.99998
The ROC AUC score on test data is            : 0.97932
The Precision Recall score on train data is : 0.99998
The Precision Recall score on test data is   : 0.97744
              precision    recall  f1-score   support

           0       0.97      0.89      0.93       935
           1       0.90      0.97      0.93       950

    accuracy                           0.93      1885
   macro avg       0.94      0.93      0.93      1885
weighted avg       0.93      0.93      0.93      1885
```





Applying Model (continued)

XGB Classifier

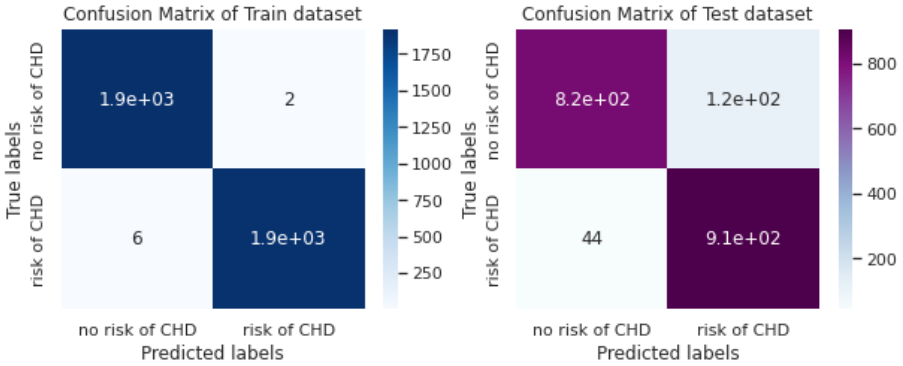
```
# Create an instance of the RandomForestClassifier
xgb_model = XGBClassifier(random_state = 0,seed = 1)

# Grid search
xgb_grid = GridSearchCV(estimator=xgb_model,param_grid = param_dict,cv = 5, verbose=2, scoring='roc_auc')

# Model Fit
xgb_grid.fit(X_train,y_train)
```

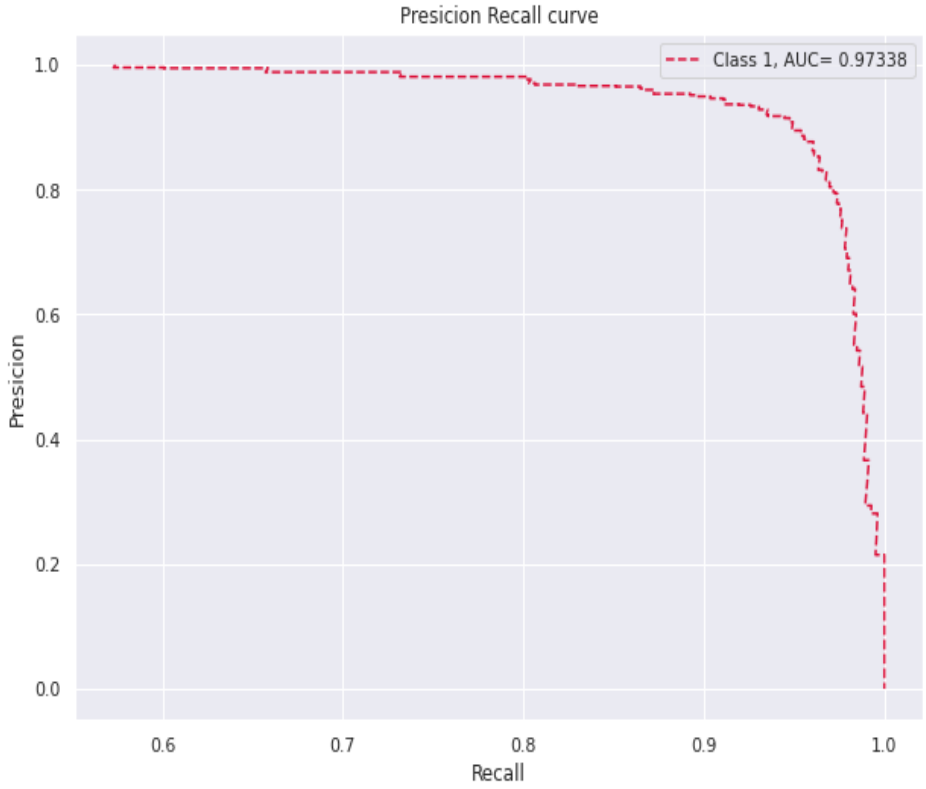
```
Fitting 5 folds for each of 1 candidates, totalling 5 fits
[CV] END gamma=1, learning_rate=0.2, max_depth=10, min_child_weight=1, n_estimators=100, reg_lambda=1, subsample =0.5; total time=   0.6s
[CV] END gamma=1, learning_rate=0.2, max_depth=10, min_child_weight=1, n_estimators=100, reg_lambda=1, subsample =0.5; total time=   0.6s
[CV] END gamma=1, learning_rate=0.2, max_depth=10, min_child_weight=1, n_estimators=100, reg_lambda=1, subsample =0.5; total time=   0.6s
[CV] END gamma=1, learning_rate=0.2, max_depth=10, min_child_weight=1, n_estimators=100, reg_lambda=1, subsample =0.5; total time=   0.6s
[CV] END gamma=1, learning_rate=0.2, max_depth=10, min_child_weight=1, n_estimators=100, reg_lambda=1, subsample =0.5; total time=   0.6s
GridSearchCV(cv=5, estimator=XGBClassifier(seed=1),
             param_grid={'gamma': [1], 'learning_rate': [0.2],
                         'max_depth': [10], 'min_child_weight': [1],
                         'n_estimators': [100], 'reg_lambda': [1],
                         'subsample ': [0.5]},
             scoring='roc_auc', verbose=2)
```

Applying Model (continued)

Model Validation(XGB Classifier)

```
The ROC AUC score on train data is           : 0.99995
The ROC AUC score on test data is            : 0.97255
The Precision Recall score on train data is  : 0.99995
The Precision Recall score on test data is   : 0.97338
              precision    recall  f1-score   support

           0       0.95      0.88      0.91       935
           1       0.89      0.95      0.92       950

    accuracy                           0.92      1885
   macro avg       0.92      0.92      0.92      1885
weighted avg       0.92      0.92      0.92      1885
```

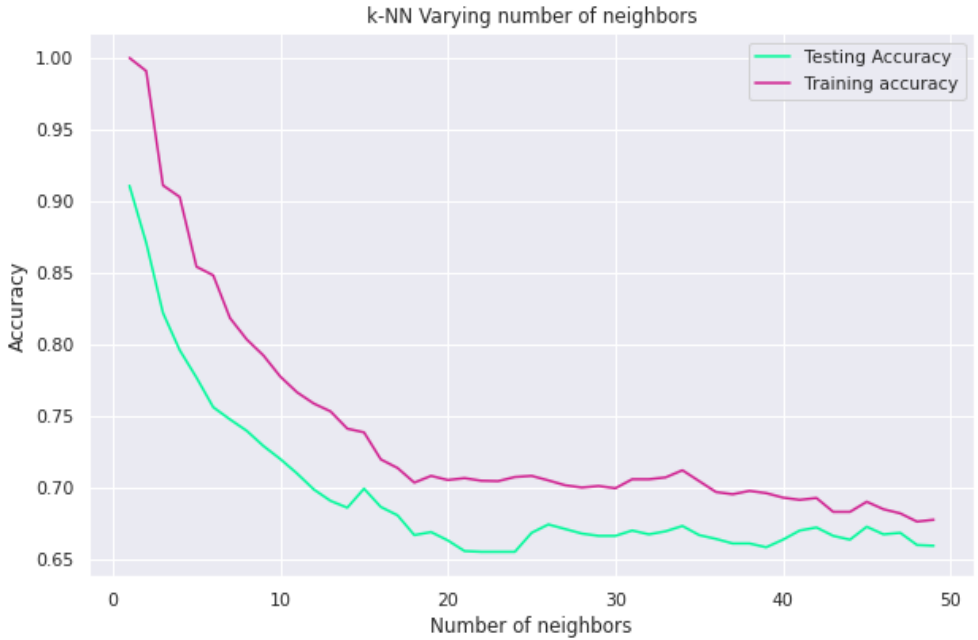




Applying Model (continued)

KNNeighbors Classifier

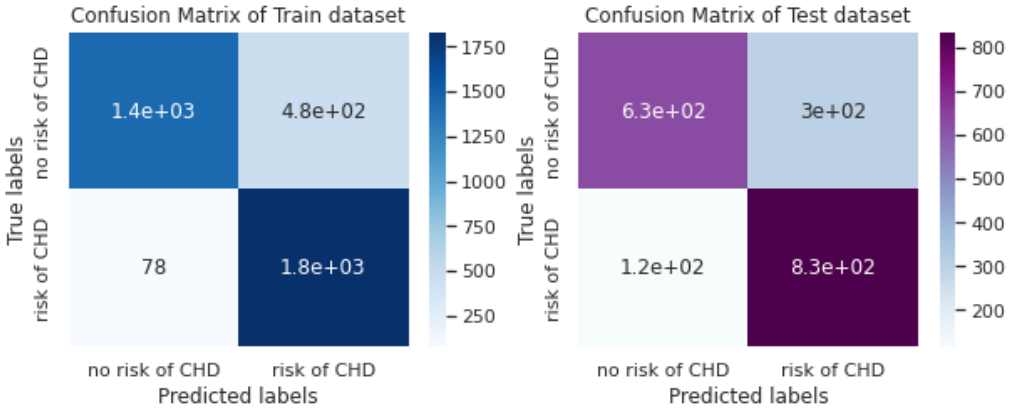
```
#Setup arrays to store training and test accuracies
neighbors = np.arange(1,50)
train_accuracy =np.empty(len(neighbors))
test_accuracy = np.empty(len(neighbors))

for i,k in enumerate(neighbors):
    # Setup a knn classifier with k neighbors
    knn = KNeighborsClassifier(n_neighbors=k)

    # Fit the model
    knn.fit(X_train, y_train)

    # Compute accuracy on the training set
    train_accuracy[i] = knn.score(X_train, y_train)

    # Compute accuracy on the test set
    test_accuracy[i] = knn.score(X_test, y_test)
```

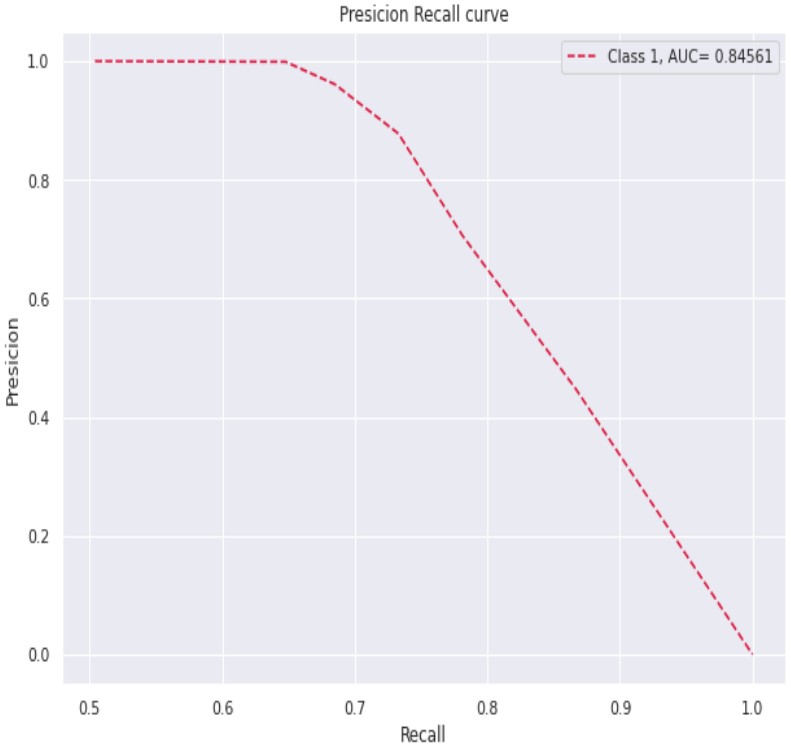


Applying Model (continued)

Model Validation(KNNeighbors Classifier)

```
The ROC AUC score on train data is           : 0.95462
The ROC AUC score on test data is            : 0.85219
The Precision Recall score on train data is  : 0.95932
The Precision Recall score on test data is   : 0.84561
              precision    recall  f1-score   support

           0       0.84      0.67      0.75       935
           1       0.73      0.88      0.80       950

    accuracy                           0.78      1885
   macro avg       0.79      0.78      0.77      1885
weighted avg       0.79      0.78      0.77      1885
```





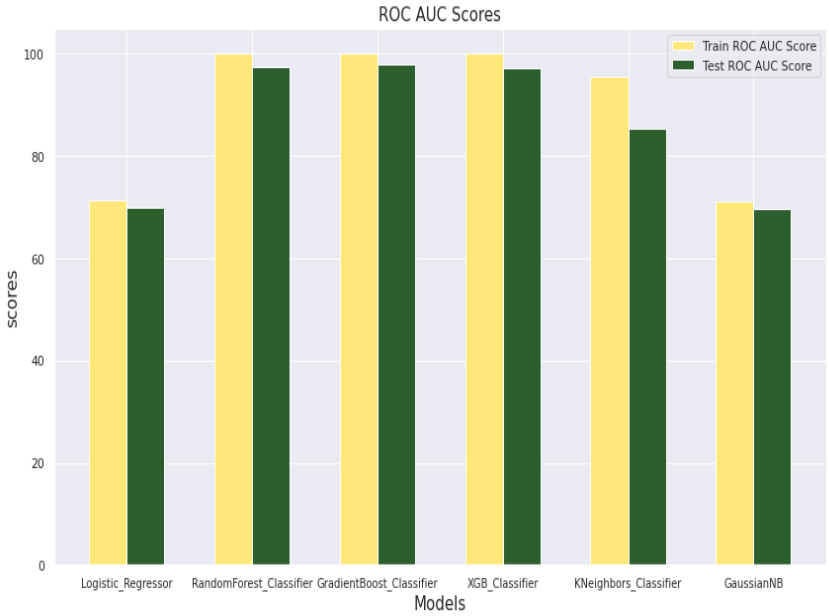
Model Performance

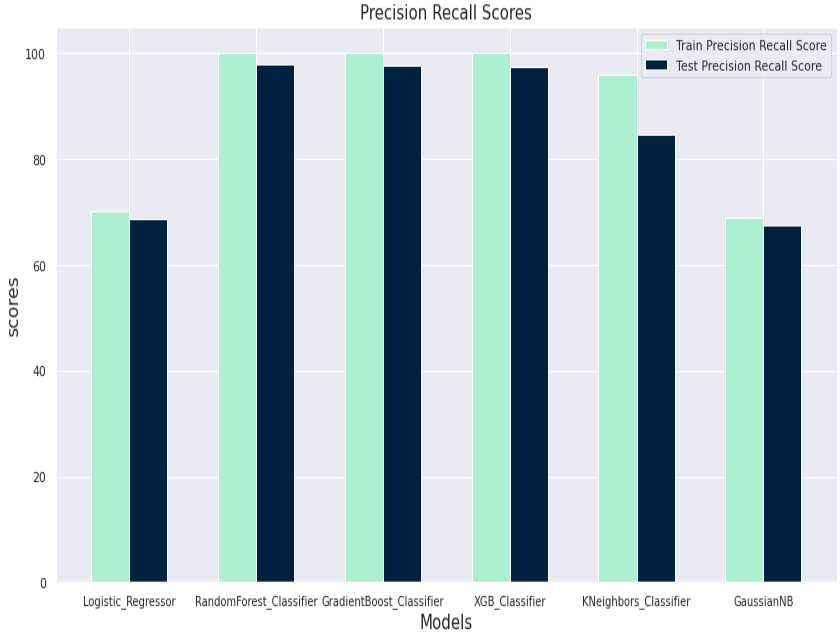
Used model's scores

	index	Models	Train ROC AUC score	Test ROC AUC score	Train Precision recall score	Test Precision recall score
0	0	Logistic_Regressor	71.4341	69.9479	70.0481	68.6141
1	1	RandomForest_Classifier	99.9481	97.4129	99.9523	97.8414
2	2	GradientBoost_Classifier	99.9982	97.9320	99.9982	97.7439
3	3	XGB_Classifier	99.9950	97.2553	99.9952	97.3383
4	4	KNeighbors_Classifier	95.4622	85.2187	95.9316	84.5614
5	5	GaussianNB	70.9767	69.6526	68.9231	67.4698

Model Performance (continued)

Visualization of model scores





Conclusion

- It’s quite obvious that smokers have a high risk of 10 year CHD.
- Patients with no history of hypertensive, stroke, diabetes have less risk of 10 year CHD.
- By using simple Logistic Regressor algorithm we were able to get the Precision Recall AUC score of 68 %
- Very little improvement in Precision Recall AUC score after using Random Forest classifier because of data imbalance.
- Data Imbalance is addressed by Random Over sampling technique..
- We got the best Precision Recall AUC score of **97.8 percent** using **Random Forest Classifier** with cross validation and hyper parameter tunning.
- Top five most important features are **SysBP,Glucose ,Totchol,age,Cigsperday**

Conclusion

- For the given dataset, **Random Forest Classifier** has proven to be the best fit model with,

Train Precision Recall AUC score	**: 99.9 %**
Test Precision Recall AUC score	**: 97.8%**
Train ROC AUC score	**: 99.9 %**
Test ROC AUC score	**: 97.4%**

Thank you