CAPSTONE PROJECT

BUSINESS OBJECTIVE/ UNDERSTANDING:

Predicting and diagnosing heart attack is the biggest challenge in the medical industry and it is based on factors like physical examination, symptoms, and signs of the patient. Factors which influence heart attack are cholesterol levels of the body, smoking habits, obesity, family history of diseases, blood pressure and working environment. Machine learning algorithms play a vital and accurate role in predicting heart attack. With the use of this project, we can use classification like ML algorithms to predict the risk of a heart attack.

Heart attack is perceived as the deadliest disease in the human life across the world. This type of disease, the heart is not capable of pushing the required quantity of blood to the remaining organs of the human body to accomplish the regular functionalities. Heart diseases are concertedly contributed by hypertension, diabetes, overweight and unhealthy lifestyle. The client wants us to predict the probability of heart attack happening to their patients. This will help them to take proactive health measures such as promoting new health related schemes.

DATA DESCRIPTION AND PREPROCESSING:

Data Dictionary:

- **Patient_ID:** Unique ID of different patients.
- **Gender:** Gender of the patient.
- **Age:** Age of the patient.
- **HyperTension:** A person has history of Hypertension or not
- **Heart_Disease:** A person has history of heart disease or not.
- **Is_Married:** Whether the person is married or not.
- **Employment_Type:** Determines whether the patient is a working professional in a Private/Govt sectors, never worked or children.
- **Residential_type:** Specifies whether the patient is from Urban/Rural areas.
- **Glucose_Levels:** Average glucose levels of a patient.
- **BMI_Values:** Considering height and weight of a patient.
- **Smoking_Habits:** Classifies whether the patient is a regular smoker, past smoker or never smoked.
- **Heart_Attack:** Chances of getting heart attack (Dependent Variable)

DATA PREPARATION:

- Null values in the column are replaced by median value and stored.
- Null values in 'Smoking_Habits' are filled with 'never smoked'.
  Null value imputation done for both the independent variables.

  Outlier treatment:
- We plot boxplot on log transformed values of BMI_Values. Outlier values are also displayed.
- Eliminating outliers based on IQR technique. When we plot boxplot, the outliers are neglected.
- We plot distplot on outlier treated BMI_Values and we get a near normal distribution.
- Skewness of Glucose_Levels is very high. We plot boxplot on the column. We infer that this column has many outliers too.
- Glucose_Levels is highly positively skewed. By using skew function.

EXPLORATORY DATA ANALYSIS & BUSINESS INSIGHTS:

Multi-collinearity:

As VIF values are less than 5, we can conclude that there is no multi collinearity amongst independent variables.

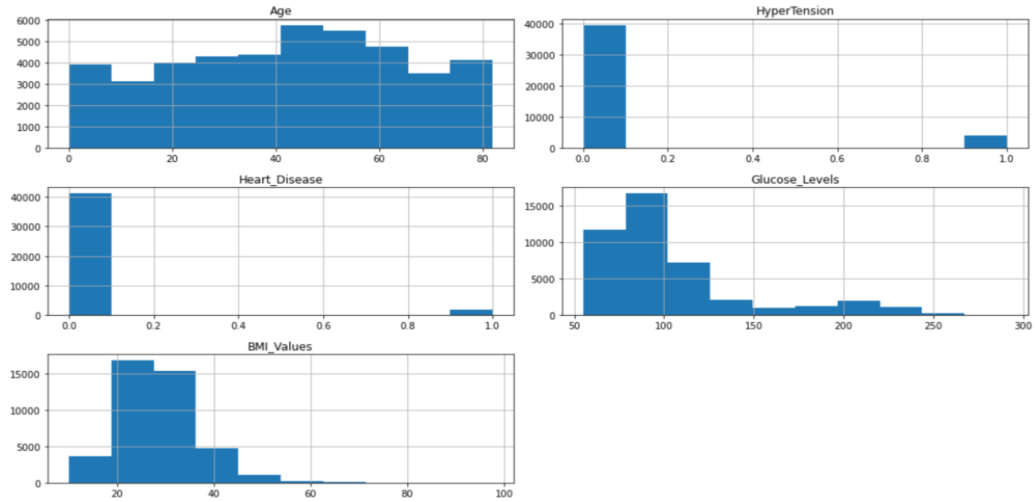
```
vif=pd.DataFrame()
vif['VIF']=[variance_inflation_factor(df_pre.values,i) for i in range(df_pre.shape[1])]
vif['Features']=df_pre.columns
vif.sort_values('VIF',ascending=False)
```

	VIF	Features
7	4.348724	Is_Married_Yes
13	3.947810	Smoking_Habits_never smoked
9	3.567592	Employment_Type_Private
0	2.878119	Age
11	2.621537	Employment_Type_children
12	1.908225	Residential_type_Urban
10	1.786402	Employment_Type_Self-employed
3	1.664401	Gender_Male
14	1.644969	Smoking_Habits_smokes
5	1.212316	HyperTension_1
6	1.152496	Heart_Disease_1
2	1.134972	BMI_Values
1	1.109704	Glucose_Levels
8	1.030362	Employment_Type_Never_worked
4	1.000482	Gender_Other

Distribution of Variables:

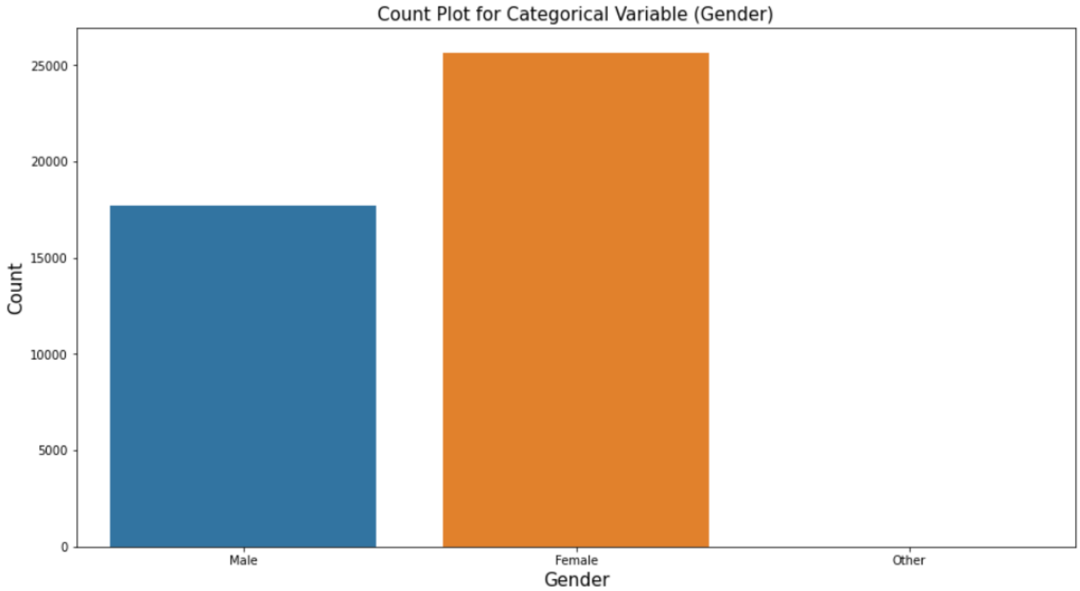
Distribution of numeric independent variables.

```
df.drop('Heart_Attack', axis = 1).hist()
plt.tight_layout()
plt.show()
print('Skewness:')
df.drop('Heart_Attack', axis = 1).skew()
```



Distribution of categoric independent variable.

```
sns.countplot(df.Gender)

plt.title('Count Plot for Categorical Variable (Gender)', fontsize = 15)
plt.xlabel('Gender', fontsize = 15)
plt.ylabel('Count', fontsize = 15)

plt.show()
```

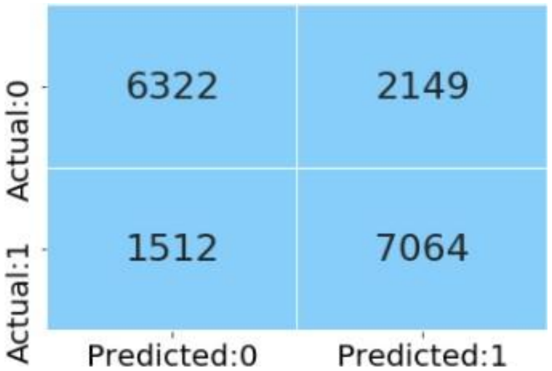


BASIC MODEL:

3. Logistic Regression ¶

```
logreg = sm.Logit(y_train, X_train).fit()

print(logreg.summary())
```

```
Warning: Maximum number of iterations has been exceeded.
         Current function value: 0.454994
         Iterations: 35
                           Logit Regression Results
==============================================================================
Dep. Variable:                      y   No. Observations:                68187
Model:                          Logit   Df Residuals:                    68171
Method:                           MLE   Df Model:                           15
Date:                Sat, 16 Oct 2021   Pseudo R-squ.:                  0.3436
Time:                        21:22:06   Log-Likelihood:                -31025.
converged:                      False   LL-Null:                       -47264.
Covariance Type:            nonrobust   LLR p-value:                     0.000
================================================================================================
                                  coef    std err          z      P>|z|      [0.025      0.975]
------------------------------------------------------------------------------------------------
const                          -2.0961      0.052    -40.555      0.000      -2.197      -1.995
Age                             1.8365      0.018    100.875      0.000       1.801       1.872
Glucose_Levels                  0.1781      0.008     21.130      0.000       0.162       0.195
BMI_Values                     -0.0857      0.013     -6.642      0.000      -0.111      -0.060
Gender_Male                     0.1283      0.021      6.160      0.000       0.087       0.169
Gender_Other                  -15.1774   1492.670     -0.010      0.992   -2940.756    2910.401
HyperTension_1                  0.2211      0.027      8.040      0.000       0.167       0.275
Heart_Disease_1                 0.5147      0.035     14.851      0.000       0.447       0.583
Is_Married_Yes                  0.2676      0.036      7.367      0.000       0.196       0.339
Employment_Type_Never_worked  -16.2871   1812.461     -0.009      0.993   -3568.646    3536.071
Employment_Type_Private         0.4022      0.032     12.644      0.000       0.340       0.465
Employment_Type_Self-employed   0.3148      0.036      8.826      0.000       0.245       0.385
Employment_Type_children        0.1533      0.160      0.959      0.338      -0.160       0.467
Residential_type_Urban          0.1244      0.020      6.136      0.000       0.085       0.164
Smoking_Habits_never smoked    -0.0802      0.025     -3.233      0.001      -0.129      -0.032
Smoking_Habits_smokes           0.2080      0.033      6.377      0.000       0.144       0.272
================================================================================================
```



Accuracy score for Logistic Regression method is 0.8578.

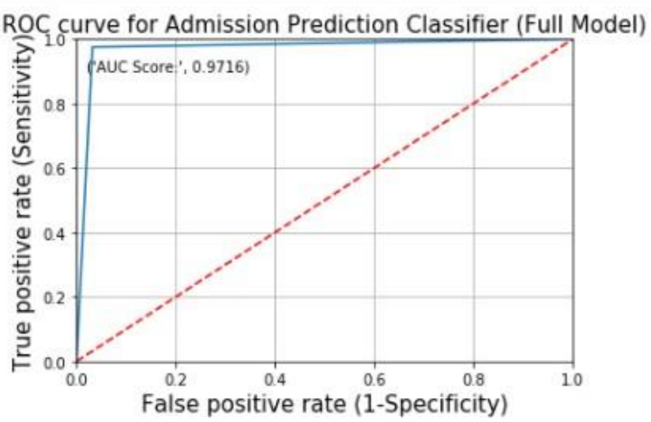
Decision Tree

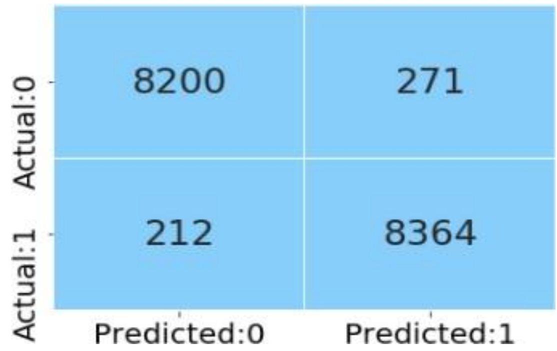
```
from sklearn.tree import DecisionTreeClassifier
classifier= DecisionTreeClassifier(criterion='entropy', random_state=0)
classifier.fit(X_train, y_train)
```

```
DecisionTreeClassifier(class_weight=None, criterion='entropy', max_depth=None,
            max_features=None, max_leaf_nodes=None,
            min_impurity_decrease=0.0, min_impurity_split=None,
            min_samples_leaf=1, min_samples_split=2,
            min_weight_fraction_leaf=0.0, presort=False, random_state=0,
            splitter='best')
```

```
y_pred= classifier.predict(X_test)
```

```
fpr, tpr, thresholds = roc_curve(y_test, y_pred)

plt.plot(fpr, tpr)

plt.xlim([0.0, 1.0])
plt.ylim([0.0, 1.0])

plt.plot([0, 1], [0, 1],'r--')

plt.title('ROC curve for Admission Prediction Classifier (Full Model)', fontsize = 15)
plt.xlabel('False positive rate (1-Specificity)', fontsize = 15)
plt.ylabel('True positive rate (Sensitivity)', fontsize = 15)

plt.text(x = 0.02, y = 0.9, s = ('AUC Score:', round(metrics.roc_auc_score(y_test, y_pred),4)))

plt.grid(True)
```





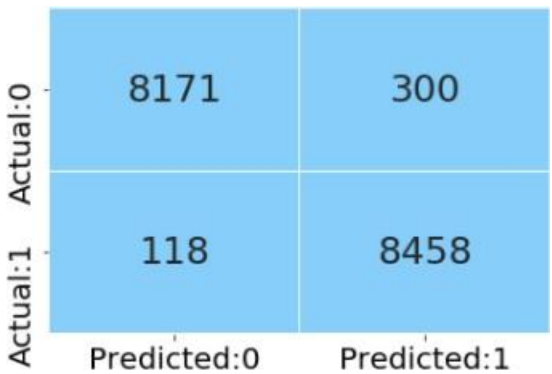
Accuracy score for Decision Tree method is 0.9716.

Random Forest

```
from sklearn.ensemble import RandomForestClassifier

model=RandomForestClassifier(n_estimators=50,random_state=1)
```

```
model.fit(X_train,y_train)
```

```
RandomForestClassifier(bootstrap=True, class_weight=None, criterion='gini',
            max_depth=None, max_features='auto', max_leaf_nodes=None,
            min_impurity_decrease=0.0, min_impurity_split=None,
            min_samples_leaf=1, min_samples_split=2,
            min_weight_fraction_leaf=0.0, n_estimators=50, n_jobs=None,
            oob_score=False, random_state=1, verbose=0, warm_start=False)
```



Accuracy score for Random Forest method is 0.9754.

FEATURE ENGINEERING & FEATURE EXTRACTION:

Transformation:

- All the categorical variables are stored in a new dataframe named df1_cat.
- Dummy variables are created for all categorical variables using get_dummies function from pandas and all encoded columns are stored in df1_cat. .(one hot encoding).

Scaling the data:

- Performing standardization on numerical dataframe num_df using standard scaler function from sklearn.preprocessing module.
- We are concatenating the scaled numerical variables and encoded categorical variables and storing it in df_pre.df_pre is the preprocessed dataframe ready for modelling.
- Concatenating preprocessed df_new with target variable to form a dataframe df_corr.We are plotting heatmap on df_corr to find correlation amongst two independent variables.

Recursive Feature Elimination (RFE):

- In the linear Classification module, we learn about various techniques for selecting the significant features in the dataset. In this example, let us consider the RFE method for feature selection.

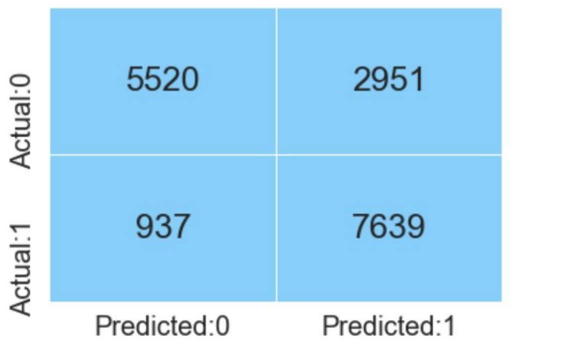
```
X_train_rfe = X_train.iloc[:,1:]
X_test_rfe = X_test.iloc[:,1:]

logreg = LogisticRegression()

rfe_model = RFE(estimator = logreg, n_features_to_select = 5)

rfe_model = rfe_model.fit(X_train_rfe, y_train)

feat_index = pd.Series(data = rfe_model.ranking_, index = X_train_rfe.columns)

signi_feat_rfe = feat_index[feat_index==1].index

print(signi_feat_rfe)
```

```
Index(['Age', 'Gender_Other', 'Heart_Disease_1',
       'Employment_Type_Never_worked', 'Smoking_Habits_smokes'],
      dtype='object')
```

Build the logisitc regression model using the variables obtained from RFE.

```
logreg_rfe = sm.Logit(y_train, X_train[['Age', 'Gender_Other', 'Heart_Disease_1',
       'Employment_Type_Never_worked', 'Smoking_Habits_smokes']]).fit()

print(logreg_rfe.summary())
```

```
Warning: Maximum number of iterations has been exceeded.
         Current function value: 0.537523
         Iterations: 35
                           Logit Regression Results
==============================================================================
Dep. Variable:                      y   No. Observations:                68187
Model:                          Logit   Df Residuals:                    68182
Method:                           MLE   Df Model:                            4
Date:                Sat, 16 Oct 2021   Pseudo R-squ.:                  0.2245
Time:                        21:29:46   Log-Likelihood:                -36652.
converged:                      False   LL-Null:                       -47264.
Covariance Type:            nonrobust   LLR p-value:                     0.000
===============================================================================================
                                  coef    std err          z      P>|z|      [0.025      0.975]
-----------------------------------------------------------------------------------------------
Age                             1.1058      0.010    113.679      0.000       1.087       1.125
Gender_Other                  -16.4718   1412.571     -0.012      0.991   -2785.059    2752.116
Heart_Disease_1                 0.3811      0.033     11.513      0.000       0.316       0.446
Employment_Type_Never_worked  -19.9423   3307.995     -0.006      0.995   -6503.493    6463.609
Smoking_Habits_smokes          -0.6696      0.022    -29.909      0.000      -0.713      -0.626
===============================================================================================
```



Accuracy score for RFE method is 0.8578.

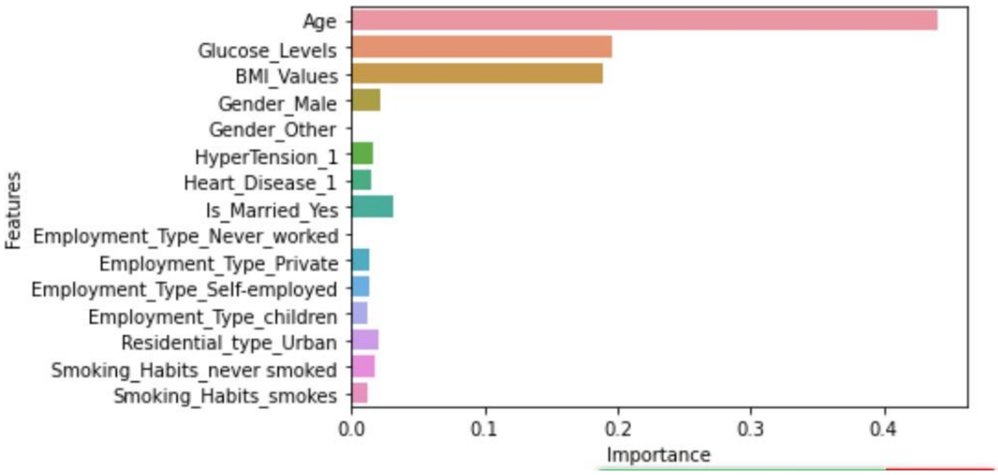
HYPER PARAMETERS TUNING:

We have found out the best parameters using Grid Search CV.

The parameters are as follows:

'criterion': 'gini', 'max_depth': 10, 'max_features': 'sqrt', 'max_leaf_nodes': 9, 'min_samples_leaf': 1,

'minsamples_split': 2, 'n_estimators': 30



Age is the most important feature in prediction of Heart attack of a patient.

COMPARISON AND SELECTION OF MODEL:

Based on our comparison done on all the Classification related algorithms, we found out that Random Forest model gives us the best accuracy score of 0.9754 and has the least False Negative values. Random Forest has the highest accuracy in predicting the correct classes.

RESULTS & DISCUSSION:

We proposed three methods in which comparative analysis was done and promising results were achieved. The conclusion which we found is that Random Forest machine learning algorithm performed better in this analysis.

DESCRIPTION OF CRITERION:

The methods which are used for comparison are Confusion Matrix, Precision, Specificity, Sensitivity, and F1 score. For some features which were in the dataset, Random Forest and decision tree classifier algorithms performed better in the ML approach when data preprocessing is applied.

The dataset size can be increased and then Machine learning with various other optimizations can be used and more promising results can be achieved.

Machine learning and various other optimization techniques can also be used so that the evaluation results can again be increased. More different ways of normalizing the data can be used and the results can be compared. And more ways could be found where we could integrate heart-disease-trained ML models with certain multimedia for the ease of patients.