In [1]:

```
# This Python 3 environment comes with many helpful analytics libraries installed
# It is defined by the kaggle/python Docker image: https://github.com/kaggle/docker-python
# For example, here's several helpful packages to load

import numpy as np # linear algebra
import pandas as pd # data processing, CSV file I/O (e.g. pd.read_csv)

# Input data files are available in the read-only "../input/" directory
# For example, running this (by clicking run or pressing Shift+Enter) will list all files under the input directory

import os
for dirname, _, filenames in os.walk('/kaggle/input'):
    for filename in filenames:
        print(os.path.join(dirname, filename))

# You can write up to 5GB to the current directory (/kaggle/working/) that gets preserved as output when you create a version using "Save & Run All"
# You can also write temporary files to /kaggle/temp/, but they won't be saved outside of the current session
%pip install autoviz # installing and importing autoviz, another library for automatic data visualization
```

```
The following command must be run outside of the IPython shell:

    $ pip install autoviz # installing and importing autoviz, another library for automatic data visualization

The Python package manager (pip) can only be used from outside of IPython.
Please reissue the `pip` command in a separate terminal or command prompt.

See the Python documentation for more information on how to install packages:

    https://docs.python.org/3/installing/
```

In [ ]:

```
import pandas as pd
import pandas_profiling
from autoviz.AutoViz_Class import AutoViz_Class
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import accuracy_score, precision_score,recall_score, f1_score
data = pd.read_csv('../input/logistic-regression-heart-disease-prediction/framingham_heart_disease.csv')
from sklearn.metrics import roc_curve, roc_auc_score
```

In [ ]:

```
print(data.columns)
data.head(10)
```

In [ ]:

```
report = pandas_profiling.ProfileReport(data)
```

In [ ]:

```
#Displays all details of descriptive analysis on data

display(report)
```

In [ ]:

```
AV = AutoViz_Class()
```

In [ ]:

```
# Let's now visualize the plots generated by AutoViz.
report_2 = AV.AutoViz('../input/logistic-regression-heart-disease-prediction/framingham_heart_disease.csv')
```

In [ ]:

```
# showing how many coumns have null values
data.isnull().sum()
```

In [ ]:

```
# Assigning mean value of the column data to all null values
mean_cigs_per_day = round(data['cigsPerDay'].mean())
mean_BPMeds= round(data['BPMeds']).mean()
mean_totChol = round(data['totChol']).mean()
mean_BMI = round(data['BMI']).mean()
mean_glucose =round(data['glucose']).mean()
data1 =data
data1['cigsPerDay'].fillna(mean_cigs_per_day,inplace= True)
data1['BPMeds'].fillna(mean_BPMeds,inplace = True)
data1['totChol'].fillna(mean_totChol,inplace =True)
data1['BMI'].fillna(mean_BMI, inplace = True)
data1['glucose'].fillna(mean_glucose, inplace =True)
data1 =data1.fillna(0)
print(data1.isnull().sum())
```

In [ ]:

```
##Defining training and test data and predicting with logistic regression function
X = data1[['male','age','cigsPerDay','BPMeds','prevalentStroke','prevalentHyp','diabetes','BMI','totChol','sysBP','glucose']]
y = data1['TenYearCHD']
X_train, X_test,y_train, y_test = train_test_split(X,y, test_size=0.3,random_state =1)
model = LogisticRegression()
model.fit(X_train,y_train)
y_pred =model.predict(X_test)
#printing logistic regression equation
print('Logistic Regression equation, {} + {}'.format(model.coef_ , model.intercept_))
```

In [ ]:

```
print('accuracy score is {:4f}'.format(accuracy_score(y_test,y_pred)))
print('Precision score: ', precision_score(y_test, y_pred,average='micro'))
print('Recall score: ', recall_score(y_test, y_pred,average='micro'))
#checking AUC under ROC curve
y_score = model.predict_proba(X_test)[:,1]
false_positive_rate, true_positive_rate, threshold = roc_curve(y_test, y_score)
print('roc_auc_score for Logistic Regression: ', roc_auc_score(y_test, y_score))
```

In [ ]:

```
# Plotting ROC Curve
import matplotlib.pyplot as plt
plt.subplots(1, figsize=(10,10))
plt.title('Receiver Operating Characteristic - Logistic regression')
plt.plot(false_positive_rate, true_positive_rate)
plt.plot([0, 1], ls="--")
plt.plot([0, 0], [1, 0] , c=".7"), plt.plot([1, 1] , c=".7")
plt.ylabel('True Positive Rate')
plt.xlabel('False Positive Rate')
plt.show()
```