MACHINE LEARNING - MILESTONE 1

DAVID GOMEZ CAMARGO

1. INTRODUCTION

Cardiovascular diseases (CVDs) are a group of disorders of the heart and blood vessels. Nearly half of all U.S. adults have some type of cardiovascular disease, a percentage that reflects recently updated guidelines for treating high blood pressure, also known as hypertension, that can lead to heart attack, heart failure and stroke.

This project can be used to explore the risk factors of CVDs in adults. The objective would be to understand how certain demographic factors, health behaviors and biological markers affect the development of heart disease. The dataset contains information on age, gender, height, weight, blood pressure values, cholesterol levels, glucose levels, smoking habits and alcohol consumption of over 70 thousand individuals. Additionally, it outlines if the person is active or not and if they have any cardiovascular diseases.

2. CORRELATION

User Amalia Nurkahasanah searched for correlated attributes whose information contents are partially contained in another attribute (Cardiovascular diseases in this case) since more correlated the attributes are, the more interdependent they are, and the more redundance we get [Figure 1]. Also presented a correlation heatmap making it easy to visualize all 14 attributes and understand its correlation at a glance [Figure 2].

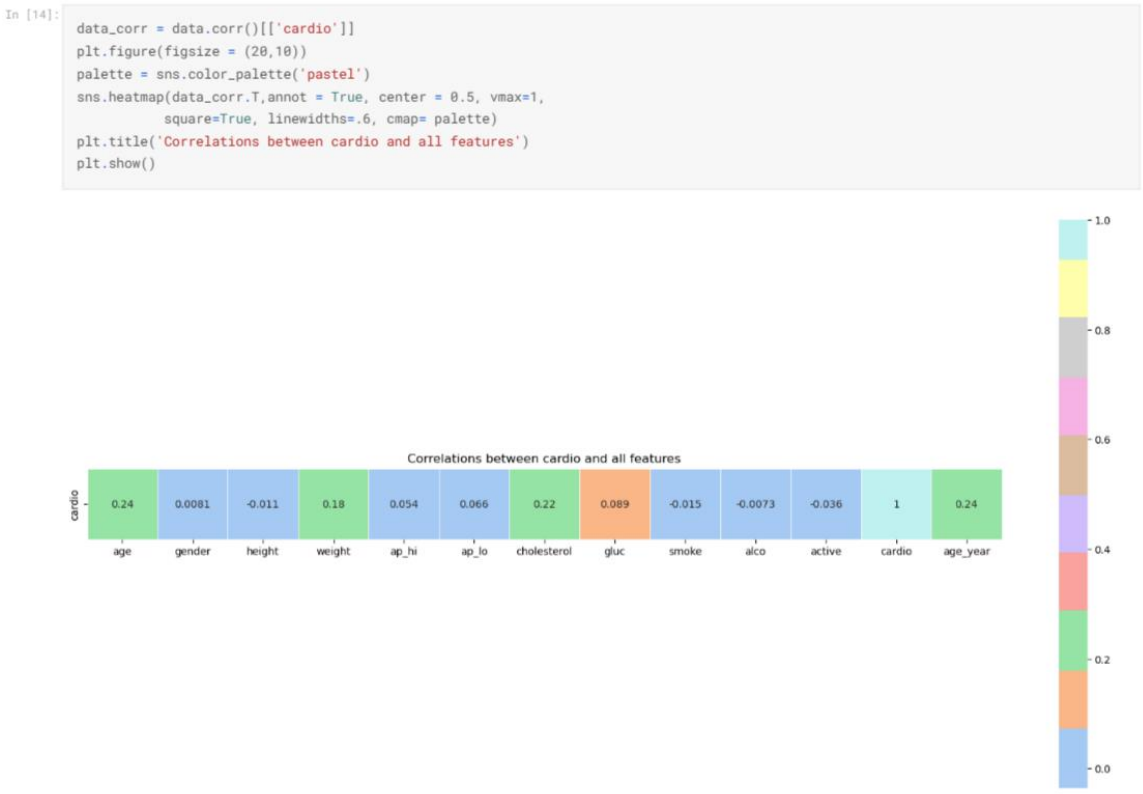

Figure 1. Correlation between CVDs and other features in dataset.

```
In [4]:
import seaborn as sns
import matplotlib.pyplot as plt
corr=data.corr()
fig = plt.figure(figsize=(15,12))
r = sns.heatmap(corr, cmap='Greens')
r.set_title('Correlation')
```

```
Out[4]:
Text(0.5, 1.0, 'Correlation')
```

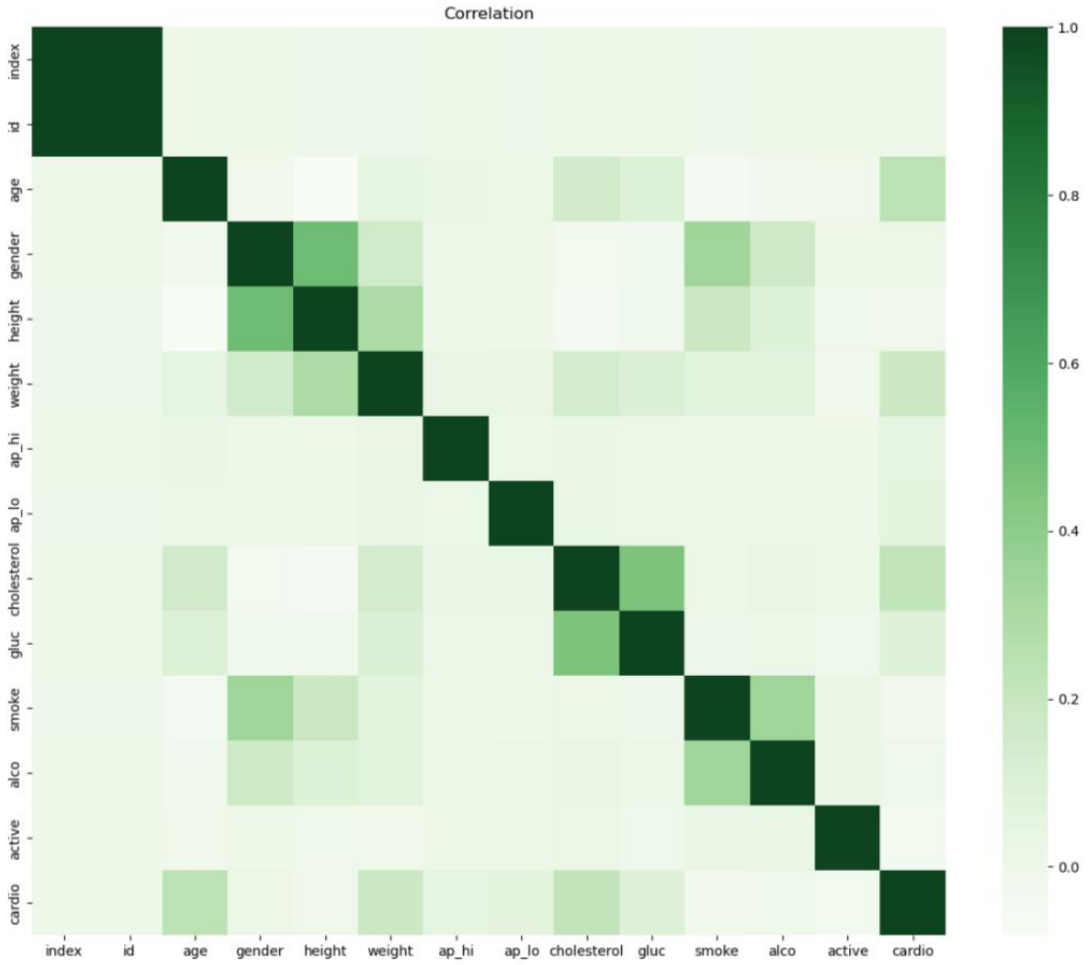

Figure 2. Heatmap between features available in dataset.

3. ROC CURVE, VIF AND GRID-SEARCH

On the other hand, user Bubpha Woratham generated a Receiver Operating Characteristic (ROC) curve showing the performance of various classification models at all classification thresholds (Curve plots two parameters: True Positive Rate and False Positive Rate).

Next, the Variance Inflation Factor (VIF) was estimated, which is used to detect the presence of multicollinearity. VIF measured how much the variance of the estimated regression coefficients are inflated as compared to when the predictor variables are not linearly related [Figure 3].

Then, a GridSearchCV hyperparameter tunning was performed to determine the optimal values for the different models. Lastly, the VIF value [Figure 4] and ROC graph [Figure 5] was presented for models after hyperparameter tunning.

In [27]:

```python
# VIF
from statsmodels.stats.outliers_influence import variance_inflation_factor
predictions = [classifier.predict(X_test) for classifier in classifiers]

# Stack the predictions into a single array
predictions = np.column_stack(predictions)

# Convert the predictions array to a dataframe
predictions = pd.DataFrame(predictions, columns=[classifier.__class__.__name__ for classifier in classifiers])

# Compute the VIF for each predictor
vif = pd.Series([variance_inflation_factor(predictions.values, i) for i in range(predictions.shape[1])],
               index=predictions.columns)
print(vif)

threshold = 5
# Initialize a list to keep track of removed predictors
removed_predictors = []

# Loop until all predictors have a VIF value less than the threshold
while True:
    # Compute the VIF for each predictor
    vif = pd.Series([variance_inflation_factor(predictions.values, i) for i in range(predictions.shape[1])],
                    index=predictions.columns)

    # Check if all predictors have a VIF value less than the threshold
    if all(vif < threshold):
        break

    # Find the predictor with the highest VIF value
    max_vif = vif.idxmax()

    # Remove the predictor with the highest VIF value
    removed_predictors.append(max_vif)
    predictions = predictions.drop(max_vif, axis=1)

print("Removed predictors:", removed_predictors)
```

```
LogisticRegression            5.611441
DecisionTreeClassifier        2.579891
RandomForestClassifier        4.854194
SVC                           7.070792
GaussianNB                    5.165676
GradientBoostingClassifier    8.437446
dtype: float64
Removed predictors: ['GradientBoostingClassifier', 'SVC']
```

Figure 3. Variance Inflation Factor (VIF) comparison.

```
LogisticRegression         5.924779
DecisionTreeClassifier     9.201907
RandomForestClassifier    12.350995
GaussianNB                 6.086877
dtype: float64
Removed predictors: ['RandomForestClassifier', 'GaussianNB']
```

Figure 4. Variance Inflation Factor (VIF) comparison after tunning.

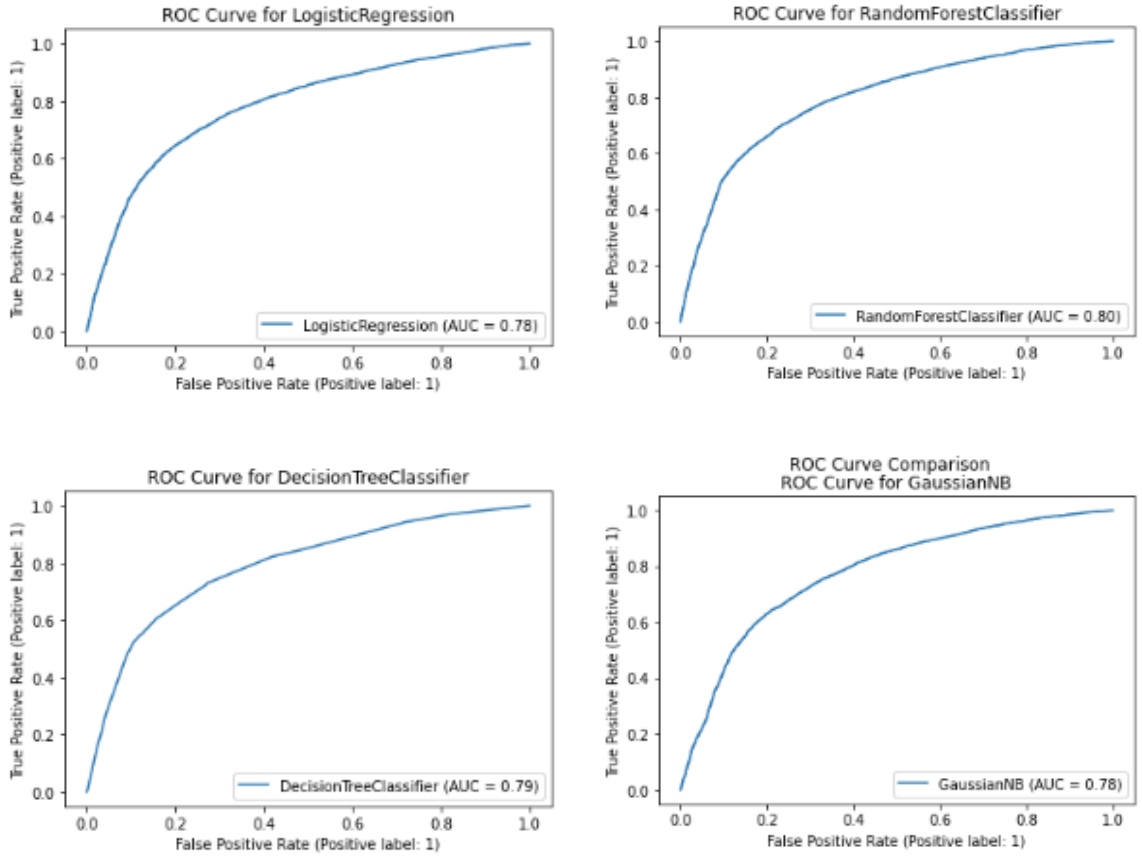

Figure 5. Receiver Operating Characteristic (ROC) curve after tunning.

4. XGB CLASSIFIER

Last of all, user [Muhammad Tayyabb] applied eXtreme Gradient Boosting, which is a boosting algorithm based on gradient boosted decision trees algorithm. XGBoost applies a better regularization technique to reduce overfitting, and it is one of the differences from the gradient boosting.

Using XGBClassifier

In [26]:
```
from xgboost import XGBClassifier
```

In [27]:
```
model = XGBClassifier(objective="binary:logistic")
model.fit(x_train_scale,y_train)
```

Out[27]:
```
XGBClassifier(base_score=0.5, booster='gbtree', callbacks=None,
              colsample_bylevel=1, colsample_bynode=1, colsample_bytree=1,
              early_stopping_rounds=None, enable_categorical=False,
              eval_metric=None, gamma=0, gpu_id=-1, grow_policy='depthwise',
              importance_type=None, interaction_constraints='',
              learning_rate=0.300000012, max_bin=256, max_cat_to_onehot=4,
              max_delta_step=0, max_depth=6, max_leaves=0, min_child_weight=1,
              missing=nan, monotone_constraints='()', n_estimators=100,
              n_jobs=0, num_parallel_tree=1, predictor='auto', random_state=0,
              reg_alpha=0, reg_lambda=1, ...)
```

```
In [28]:
model.score(x_test_scale,y_test)

Out[28]:
0.7262857142857143

In [29]:
y_pred = model.predict(x_test_scale)

In [30]:
cm = confusion_matrix(y_test,y_pred)
cm

Out[30]:
array([[6764, 2043],
       [2747, 5946]])

In [31]:
plt.figure(figsize=(12,6))
sns.heatmap(cm,annot=True,cmap='Blues',fmt='g')
plt.xlabel('Predicted Value')
plt.ylabel('Actual Value')
plt.show()
```

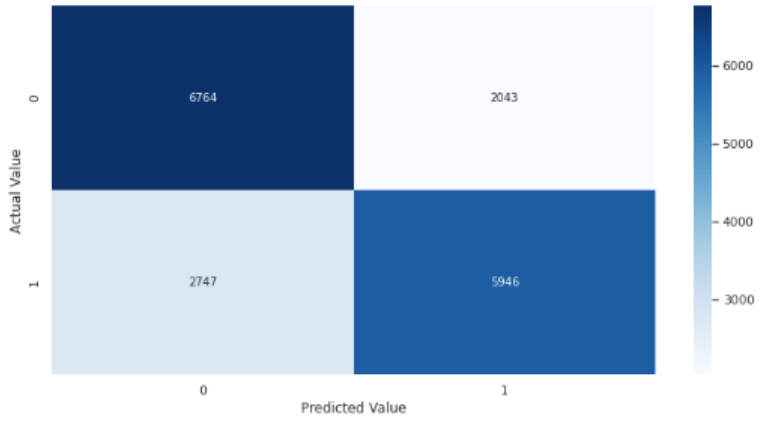

Figure 6. eXtreme Gradient Boosting.

After reviewing all these approaches to this dataset, I think this project can result in a great opportunity to apply multiple classification machine learning algorithms to explore the potential relations between risk factors and cardiovascular disease that can ultimately lead to improved understanding of this serious health issue and design better preventive measures.