Heart Data Analysis(Classification)

- **Objective of the analysis**

  The objective of this analysis is to be able to define a model to predict if a patient can have an heart attack based on some heart parameters

- **Description of the data set**

  For this project I'm going to use a Data Set representing Heart Attack classification. Each case of heart attack has some parameters:
  1. age - age in years
  2. sex - sex (1 = male; 0 = female)
  3. cp - chest pain type (1 = typical angina; 2 = atypical angina; 3 = non-anginal pain; 0 = asymptomatic)
  4. trestbps - resting blood pressure (in mm Hg on admission to the hospital)
  5. chol - serum cholesterol in mg/dl
  6. fbs - fasting blood sugar > 120 mg/dl (1 = true; 0 = false)
  7. restecg - resting electrocardiographic results (1 = normal; 2 = having ST-T wave abnormality; 0 = hypertrophy)
  8. thalach - maximum heart rate achieved
  9. exang - exercise induced angina (1 = yes; 0 = no)
  10. oldpeak - ST depression induced by exercise relative to rest
  11. slope - the slope of the peak exercise ST segment (2 = upsloping; 1 = flat; 0 = downsloping)
  12. ca - number of major vessels (0-3) colored by fluoroscopy
  13. thal - 2 = normal; 1 = fixed defect; 3 = reversible defect
  14. num - the predicted attribute - diagnosis of heart disease (angiographic disease status)
  (Value 0 = < diameter narrowing; Value 1 = > 50% diameter narrowing)

- **Data Describe**

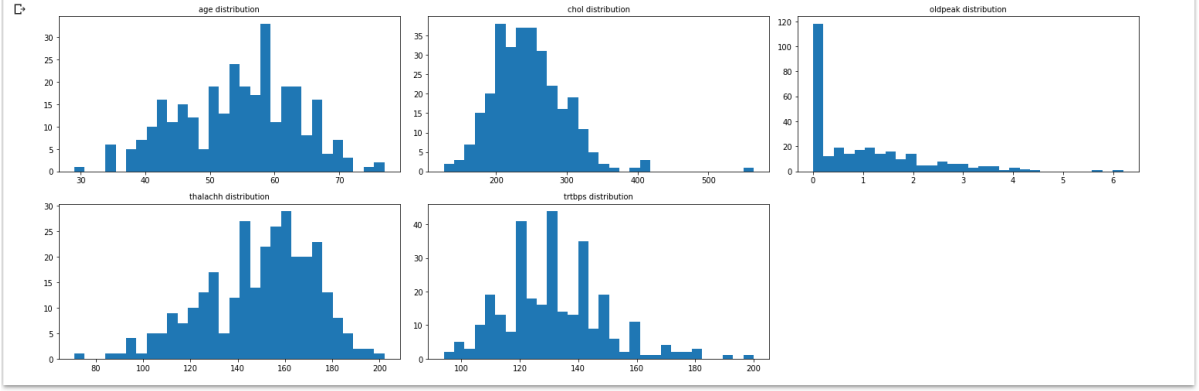
```
#Numerical Data
Heart_Data.describe()
```

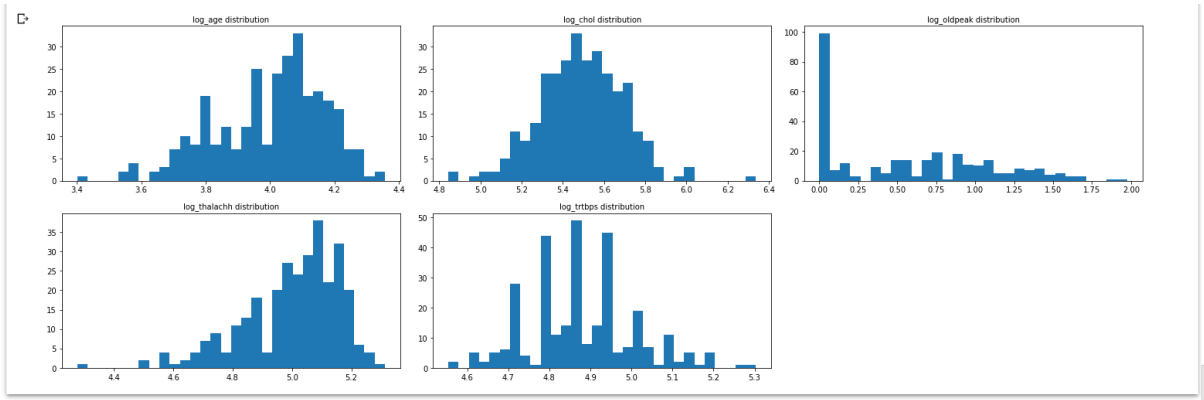
	age	sex	trtbps	chol	fbs	thalachh	exng	oldpeak	caa	output	restecg_hypetrophic	restecg_normal	restecg_abnormal	cp_asymptomatic
count	303.000000	303.000000	303.000000	303.000000	303.000000	303.000000	303.000000	303.000000	303.000000	303.000000	303.000000	303.000000	303.000000	303.000000
mean	54.366337	0.683168	131.623762	246.264026	0.148515	149.646865	0.326733	1.039604	0.729373	0.544554	0.485149	0.501650	0.013201	0.471947
std	9.082101	0.466011	17.538143	51.830751	0.356198	22.905161	0.469794	1.161075	1.022606	0.498835	0.500606	0.500824	0.114325	0.500038
min	29.000000	0.000000	94.000000	126.000000	0.000000	71.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000
25%	47.500000	0.000000	120.000000	211.000000	0.000000	133.500000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000
50%	55.000000	1.000000	130.000000	240.000000	0.000000	153.000000	0.000000	0.800000	0.000000	1.000000	0.000000	1.000000	0.000000	0.000000
75%	61.000000	1.000000	140.000000	274.500000	0.000000	166.000000	1.000000	1.600000	1.000000	1.000000	1.000000	1.000000	0.000000	1.000000
max	77.000000	1.000000	200.000000	564.000000	1.000000	202.000000	1.000000	6.200000	4.000000	1.000000	1.000000	1.000000	1.000000	1.000000

- **Numerical Data**

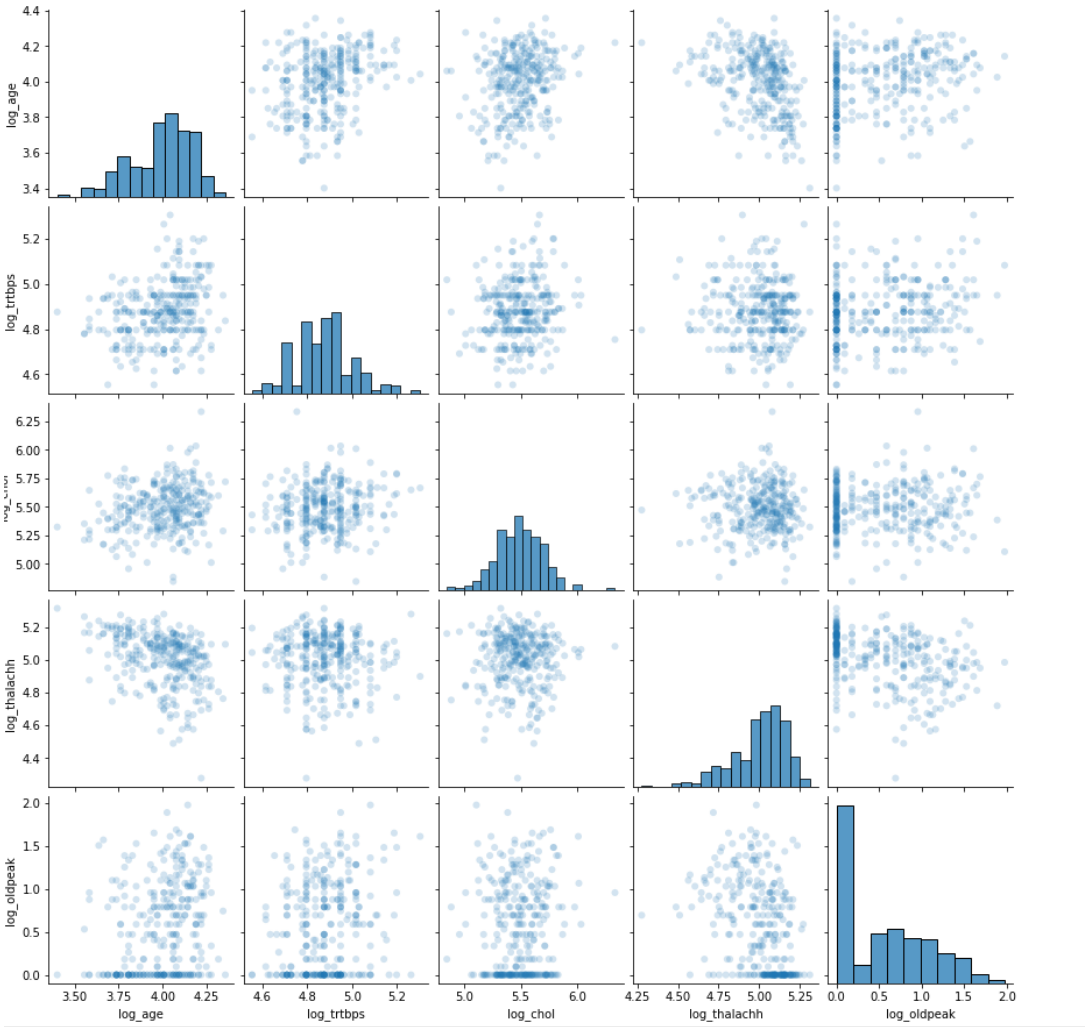
Numerical Data Distribution.

```
Heart_Data_num=Heart_Data.drop(columns=['slp','thall','restecg','cp','sex','output','caa','exng','fbs'])
hist_loop(data=Heart_Data_num,
          rows=3,
          cols=3,
          figsize=(20,10))
```





The pair plot doesn’t show any strong correlation between all numerical data



- **Finding the best model**

  For this analysis we need to compare scores between some classfication models, those models are : Logistic Regression , KNN , Decision Tree and Random Forest. To compare this model we are going to plot their confusion matrix and compare their scores.

Logistic Regression

```
# Standard logistic regression
lr = LogisticRegression(solver='liblinear').fit(X_train, y_train)
y_pred_lr = lr.predict(X_test)

precision_lr, recall_lr = (round(float(x),2) for x in list(score(y_test,
                                                                 y_pred_lr,
                                                                 average='weighted'))[:-2])
# writing all lr stats to metrics DataFrame
lr_stats = pd.Series({'precision':precision_lr,
                      'recall':recall_lr,
                      'accuracy':round(accuracy_score(y_test, y_pred_lr), 2),
                      'f1score':round(f1_score(y_test, y_pred_lr), 2),
                      'auc': round(roc_auc_score(y_test, y_pred_lr),2)},
                     name='Logistic Regression')
```

K Nearest Neighbours

```
#K Nearest Neighbors model

knn=KNeighborsClassifier(n_neighbors=3 , weights='distance')
knn = knn.fit(X_train,y_train)
y_pred_knn=knn.predict(X_test)
precision_knn, recall_knn = (round(float(x),2) for x in list(score(y_test,y_pred_knn,average='weighted'))[:-2])
# adding KNN stats to metrics DataFrame
knn_stats = pd.Series({'precision':precision_knn,
                       'recall':recall_knn,
                       'accuracy':round(accuracy_score(y_test, y_pred_knn), 2),
                       'f1score':round(f1_score(y_test, y_pred_knn), 2),
                       'auc': round(roc_auc_score(y_test, y_pred_knn),2)}, name='KNN')
```

Decision Tree Classifier

```
#Decision Tree
dt = DecisionTreeClassifier(random_state=42)
dt = dt.fit(X_train, y_train)
dt.tree_.node_count, dt.tree_.max_depth
```

✓ 0.5s

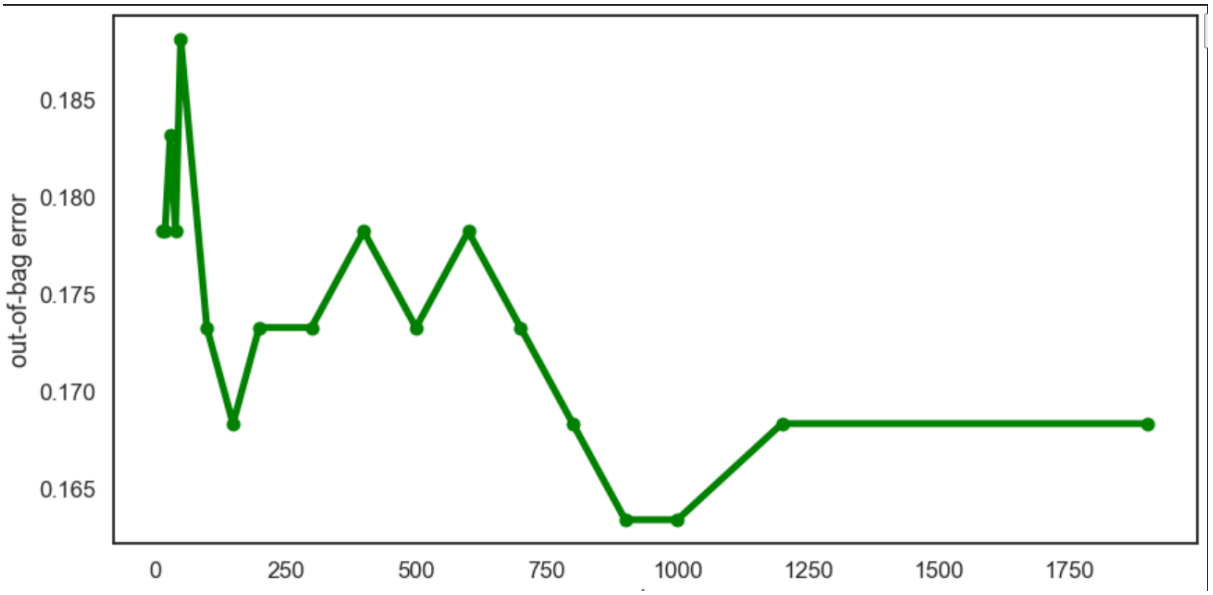
```
(71, 10)
```

```
y_train_pred = dt.predict(X_train)
y_pred_dt = dt.predict(X_test)

precision_dt, recall_dt = (round(float(x),2) for x in list(score(y_test,
                                                                 y_pred_dt,
                                                                 average='weighted'))[:-2])
# adding dt stats to metrics DataFrame
dt_stats = pd.Series({'precision':precision_dt,
                      'recall':recall_dt,
                      'accuracy':round(accuracy_score(y_test, y_pred_dt), 2),
                      'f1score':round(f1_score(y_test, y_pred_dt), 2),
                      'auc': round(roc_auc_score(y_test, y_pred_dt),2)}, name='Decision Tree')
```

Random Forest

For this model , first we have to find the best number of trees that minimize the OOB (out of the bag error) in this model is around 800 trees , but i choose to use 200 trees because of the computational work .

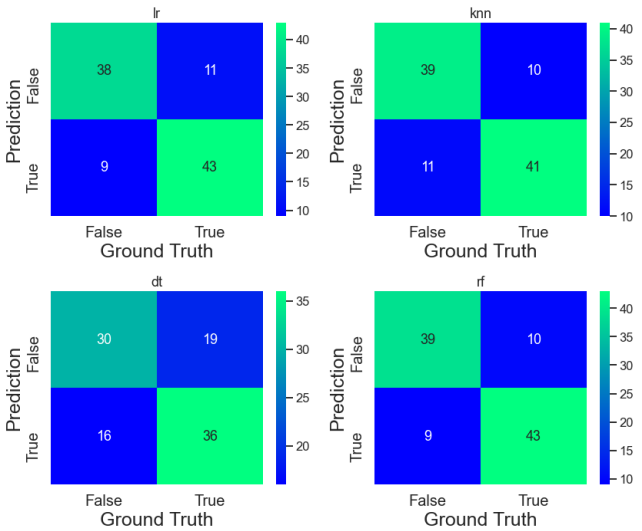


Then we use the chosen number of trees in the model

```
rf = RF.set_params(n_estimators=200)

y_pred_rf = rf.predict(X_test)
precision_rf, recall_rf = (round(float(x),2) for x in list(score(y_test,
                                                                 y_pred_rf,
                                                                 average='weighted'))[:-2])
rf_stats = pd.Series({'precision':precision_rf,
                      'recall':recall_rf,
                      'accuracy':round(accuracy_score(y_test, y_pred_rf), 2),
                      'f1score':round(f1_score(y_test, y_pred_rf), 2),
                      'auc': round(roc_auc_score(y_test, y_pred_rf),2)}, name='Random Forest')
```

Confusion Matrix



- ## Results and Findings

I have decided to put all scores from each model in a dataset to better understand the differences among all models.

	precision	recall	accuracy	f1score	auc
Logistic Regression	0.80	0.80	0.80	0.81	0.80
KNN	0.79	0.79	0.79	0.80	0.79
Decision Tree	0.65	0.65	0.65	0.67	0.65
Random Forest	0.81	0.81	0.81	0.82	0.81

The best models seems to be the random forest and the logistic regression and KNN the only one with significantly low scores is the decsion tree classifier caused by a large numver of features..Maybe with some boosting algorithms we could have even more accurate models , thanks to the low number of data.