**Diabetes Test Prediction**

**Table of contents**

**Step 1: Answering the question:**

**1.1.** Introduction.

**1.2.** Required Libraries

**1.3.** The Problem Domain

**1.4.** Data Analysis Checklist

**1.5.** Loading the Datest.

**Step 2: EDA Exploratory Data Analysis**

**2.1.** Using pie plot.

**2.2.** Using box plot.

**2.3.** Using Andrews curves.

**2.4.** Using Parallel coordinates plot.

**2.5.** Using Radviz Plot.

**2.6.** Using hist plot.

**2.7.** Using Violin plot.

**2.8.** Using Pair and KDE Kernel Density Estimate Plot.

# Step 3: Tidying the data.

**3.1.** Fill NaN Methods Comparison

**3.2.** Add features

**3.3.** Automatic Outlier Detection Algorithms

**3.4.** Feature Selection Methods

**3.5.** Imbalanced Correction Methods

**Step 5: Building the classifier**

**Step 1: Answering the question:**

**1.1. Introduction.**

The Pima Indians Diabetes Dataset involves predicting the onset of diabetes within five years in Pima Indians given medical details. It is a binary (2-class) classification problem.

Several familiar types of classification models algorithms utilized:

1. To choose the best classification algorithms and efficiently perform another appropriate comparison between the same algorithms.
2. To compare the utilized feature engineering and pre-processing methods.
3. To get a broad range of choices.

Utilized classification models are respectively:

1. Logistic Regression
2. Linear Discriminant Analysis
3. K Neighbors Classifier
4. Decision Tree Classifier
5. Gaussian NB
6. Support Vector Classifier
7. XGBoost Classifier

**1.2. Required libraries**

This notebook uses several Python packages that come standard with the Google Colaboratory. The primary libraries that we'll be operating are respectively:

* **NumPy**: Provides a fast numerical array structure and helper functions.
* **Pandas**: Provides a DataFrame structure to store data in memory and work with it easily and efficiently.
* **scikit-learn**: The essential Machine Learning package in Python.
* **XGBoost:** Optimized distributed gradient boosting library designed to be highly efficient, **flexible and portable matplotlib**: Basic plotting library in Python; most other Python plotting libraries are built on top of it.
* **Seaborn**: Advanced statistical plotting library.
* **watermark**: A Jupyter Notebook extension for printing timestamps, version numbers, and hardware information.

**1.3. The problem domain**

Our company just got funded to create a smartphone app that automatically female diabetes detection to use in remote villages in India, from simple devices for each test attribute and fill it in the smartphone, we will be building part of the data analysis pipeline for this app.

We tasked by the Head of Data Science to create a machine learning model, the model takes eight attributes from the user and detects diabetes based on those attributes alone.

We got a dataset from the field researchers to develop the model, which includes predicting the onset of diabetes within five years in Pima-Indians given medical details. With the following attributes:

* Number of times pregnant.
* Plasma glucose concentration a 2 hours in an oral glucose tolerance test.
* Diastolic blood pressure (mm Hg).
* Triceps skinfold thickness (mm).
* 2-Hour serum insulin (mu U/ml).
* Body mass index (weight kg / height m2).
* Diabetes pedigree function.
* Age (years).
* Class variable (0 or 1).

**1.4. Data analysis checklist:**

The data analysis checklist:

1. **Specify** **the type of data analytic question (e.g. exploration, association causality) before touching the data:** We are trying to detect female diabetes tests (Positive test or Negative test) based on eight continuous attributes.
2. **Define the metric for success before beginning:** We will use the accuracy to quantify how well our model is performing. they told us that we should achieve at least 77% accuracy.

**1.5. Loading the Dataset:**

The *!wget*, *!mkdir*, *!head -n* and *!cat | wc -l* commands was used to download the dataset from the website on the Google Colab, make new directory folder and name it data, show rows from the dataset(Figure 1.1) , and know the number of instances, respectively.

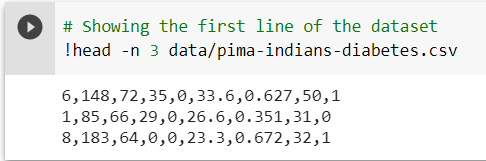


Figure 1.1: Showing first three rows of the dataset

From the previous figure we can notes the following:

1. There is no header row so (*header= None*) must be used while we read the csv.
2. There is no need to use *Sep* *parameter* because the separation between the values is (,) as the default separation of *Panda csv Sep*.

**Step 2: Exploratory Data Analysis**

Several types of plots were used to exploring the data as the following:

1. By using a pie plot to visualize and compute the difference between the categories (figure 2.1), we can perceive the notable difference between the numbers of categories. A balance issue must be considered.

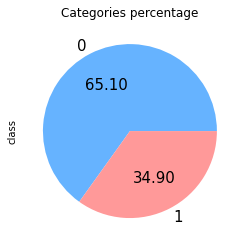


Figure 2.1: Categories percentage

1. By using **box plot** to have an indication of how the values in the data are spread out, and to visualize the distribution of values within each attribute (Figure 2.2), we can notes the following:
   1. All attributes values spread between 0 and 200 except insulin values,
   2. The variance of values is extremely, so we must use preprocessing methods before training.

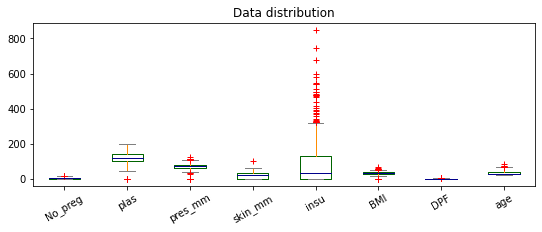


Figure 2.2: Data values distribution

1. By exploiting **Andrew’s curves** plot to visualize data clustering for each class, we can notes:
   * 1. Curves belonging to samples of a similar class aren't closer together.
     2. The curves of the two classes mix together and don’t define structures.
     3. It is problematic to target those classes, add features must be considered.

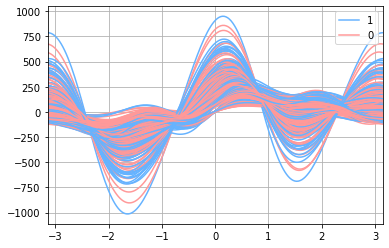


Figure 2.3: Andrew’s curves

1. By using Parallel coordinates plot to comparing variables together and observing the relationships between them (figure 2.4), we can notes that there are no significant phenomena for each class, between the attributes.

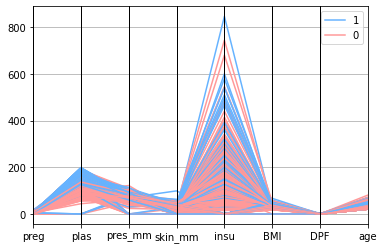


Figure 2.4: Parallel coordinates plot

1. By using **Radviz plotting** to recognize clustering attribute for each class (figure 2.5), we can notes the following:
2. Classes are clustering to the same attributes.
3. There are outlier instances, so outlier detection methods must be used.
4. Some attributes are does not affect the categories, so feature selection methods must be used.

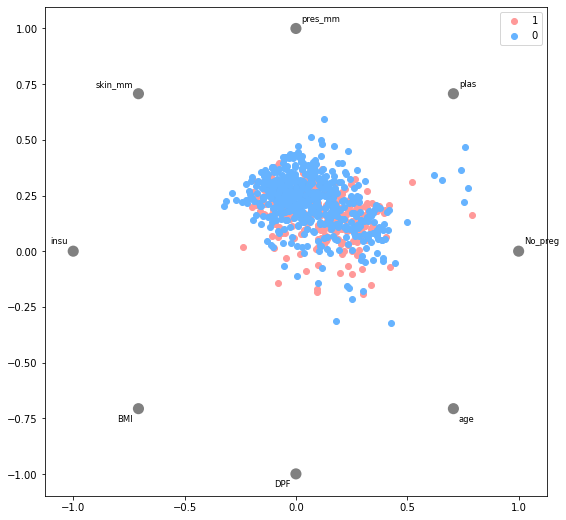


Figure 2.5: Radviz plotting

1. By using **histogram plots** we can visualize mean, median, standard deviation, and mode for the attributes values. We can note that we have wide range of data distribution so the preprocessing methods must be used to have good results.

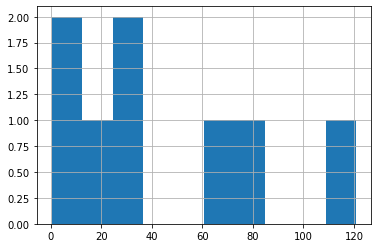
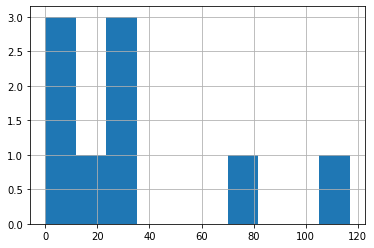
 

Figure 2.6.a: Mean plot Figure 2.6.b: Median plot

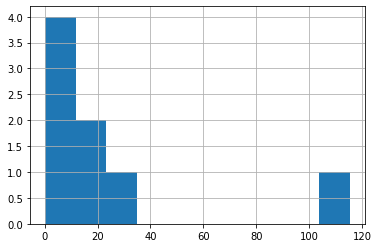


Figure 2.6.c: Standard deviation plot

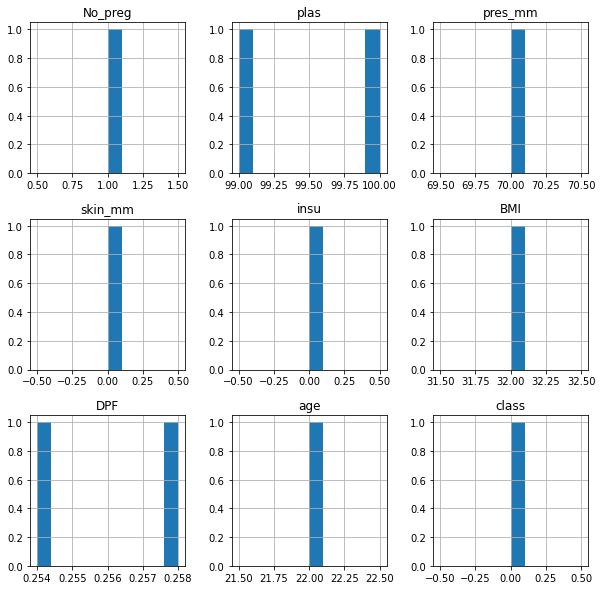
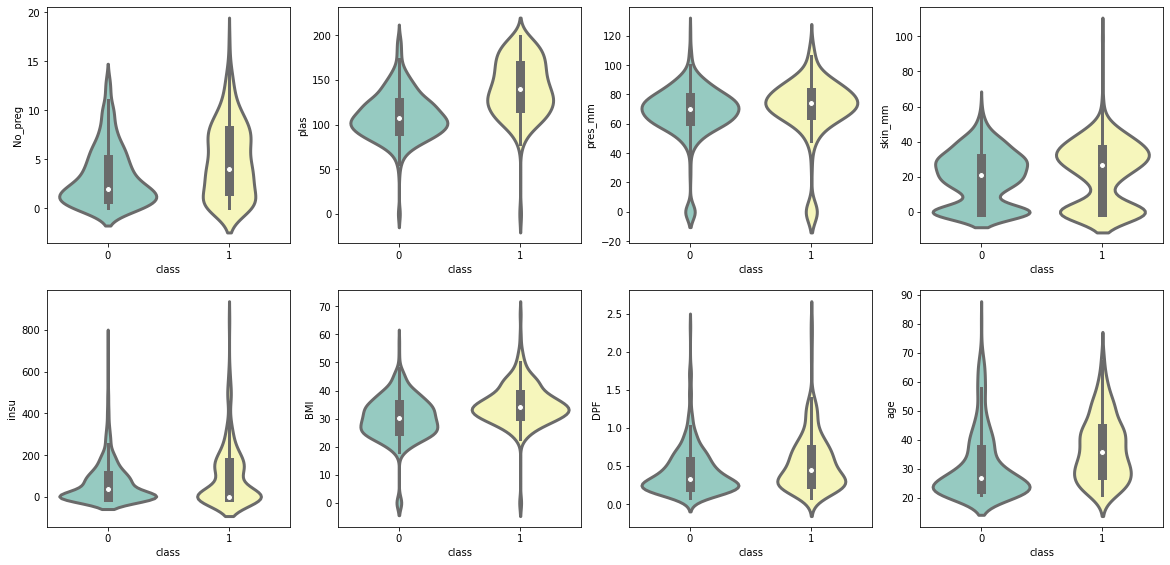


Figure 2.6.d: Mode plot

1. By using **violin plot** to shows the distribution of quantitative data across several levels of categorical variables such that those distributions can be compared and features a kernel density estimation of the underlying distribution we can notes
2. A lot of zeros values.
3. The mean values for each class is different so we must consider this when imputing *NaN* values.
4. There is an outlier values.



After talking with the field researchers, they fill the null values with zero so we must replace all not logical zeros values with NaN values.

1. By using **Pair and KDE plot** (figure 2.8) to visualize distribution of single variables and relationships between variables we can notes
2. Relationships between some attribute,
3. A lot of zeros also,
4. Probability distributions are close and same.

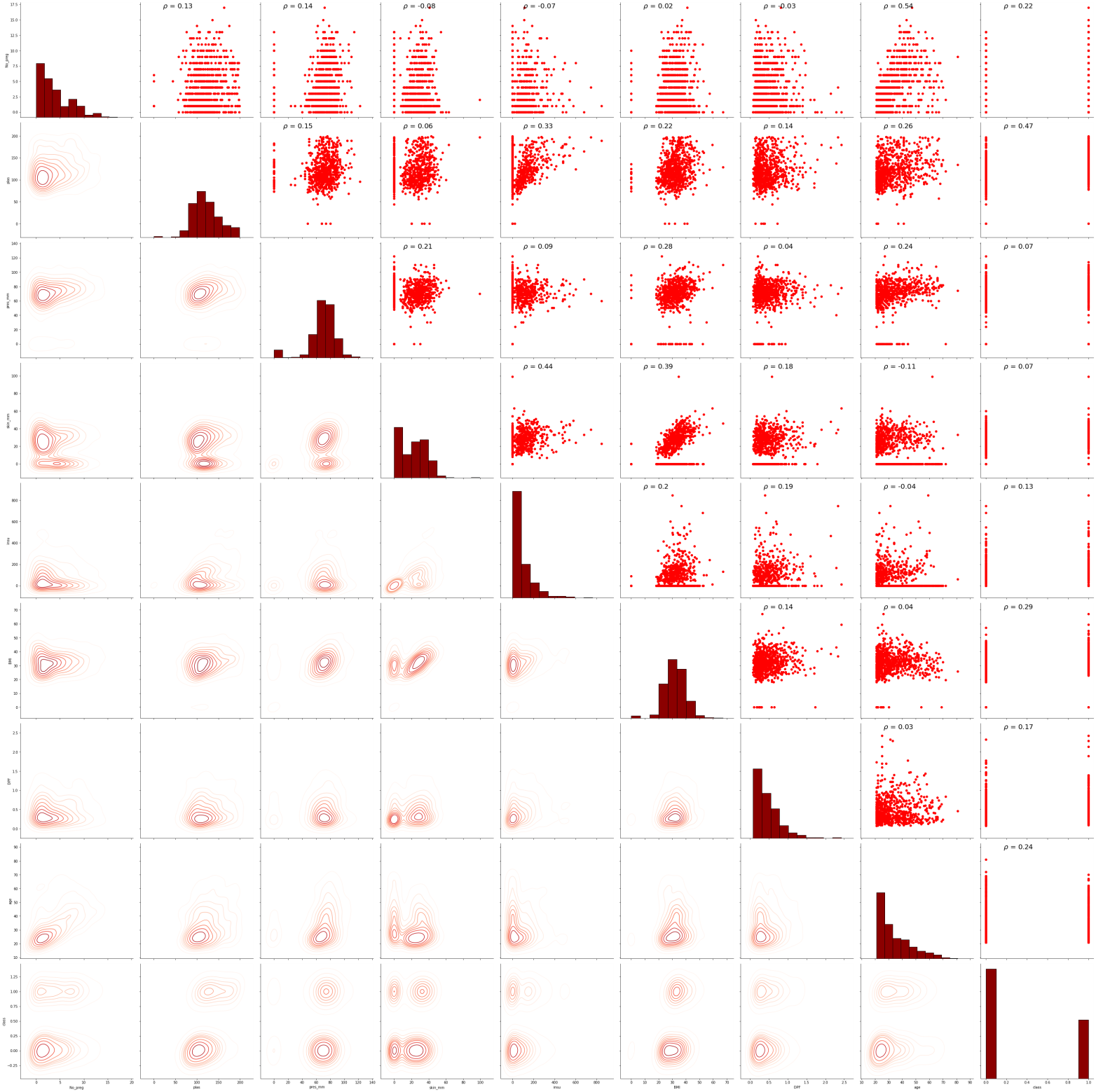


Figure 2.8: Pair and KDE plot

1. By using **Correlation Matrix Heat map** (figure 2.9) to illustrate the relationship between variables we can notes no significant case of multicollinearity is observed because all of correlation coefficients are less than 0.7.

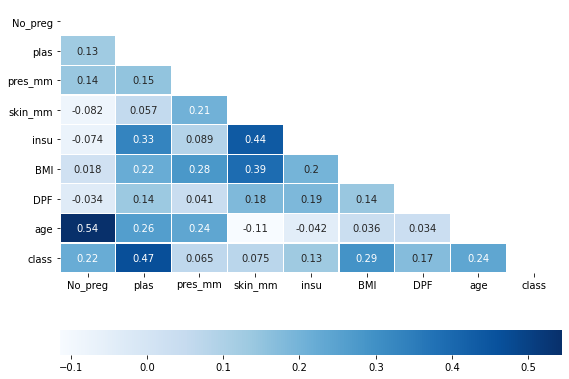


Figure 2.9: Correlation Matrix Heat map

# Step 3: Tidying the data

# Fill NaN Methods.

# Considering the results of the exploratory analysis, the two classifications were separated into two data frames. The missing values were filled in for each of them separately, after merging them, the comparison algorithms were trained.

# The comparison was performed by splitting the values into 75 percent for training and 25 percent for testing. 20 splitting performed and compute average accuracy and standard deviation.

# Table 3.1: Fill NaN Methods Comparison

# 

# 

# Figure 3.1: Fill NaN Methods Comparison

# The best fillNa method is Median\_fill, so it performed on the data.

# Add features.

# The process of adding new features was based on the following:

# BMI classification table

* 1. **>** 30 : Obese
  2. 25-30 : Overweight
  3. 20-25 : Healthy weight range
  4. 20-18 : Underweight
  5. <18 : Very Underweight

# But there is no one very underweight

# 2-Hour serum insulin classification

* 1. **>**140 : Normal
  2. 140-199 : pre-diabetic
  3. < 199 : diabetic

# 2-Hour in an oral glucose classification table

# >100: Normal

# 100-125: pre-diabetic

# < 125: Diabetic

# Nine features have been added with this step.

# Automatic Outlier Detection Algorithms

# Five algorithms were used and they are as follows

# DBSCAN

# Isolation Forest

# Minimum Covariance Determinant

# Local Outlier Factor

# One-Class SVM

# Table 3.3 Automatic Outlier Detection Algorithms

# 

# 

# The best Outlier Algorithm is Local Outlier Factor, so it performed on the data.

# Feature Selection Methods

# Six methods were used to select the best features, and they are as follows

# Removing features with low variance

# UFS Select K Best

# UFS Select F pr` False Positive Rate test

# Select From Model

# Sequential Feature Selection

# Principal Component Analysis

# 

# 

# The best Feature Selection Methods is Select From Model, so it performed on the data.

# Imbalanced Correction Methods

# Seven methods were used to Imbalanced Correction, and they are as follows:

# SMOTE

# Border line SMOTE

# SVM SMOTE

# Adaptive Synthetic Sampling (ADASYN)

# Random Over Sampler

# Random Under Sampler

# Combining Random Over and Under sampling

# 

# 

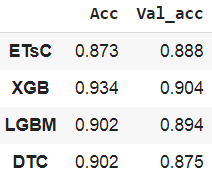
# The best Imbalanced Correction Methods is Random Over Sampler, but Random Over Sampling method is more efficient with massive data only, so the Borderline SMOTE performed on the data.

**Step 5: Building the classifier**

Tree classifiers are the best so it used to build the final model, five tree classifiers were used which it in the following:

1. XGBoost Classifier
2. Extra Trees Classifier
3. LGBM Classifier
4. Decision Tree Classifier

Table 5.1 model Comparison



The selected model is **XGBoost classifier**

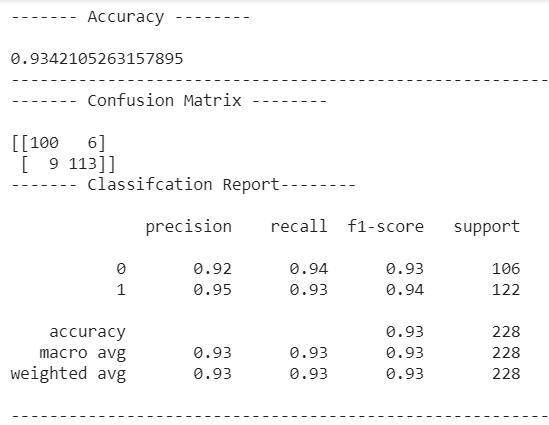


Figure 5.1: XGBoost classifier Confusion Matrix

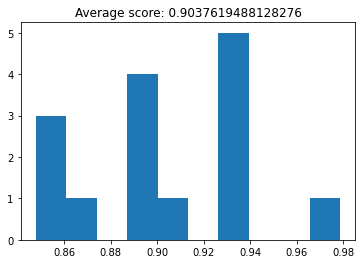


Figure 5.2: XGBoost classifier Cross-validation