THE PYTHON PROGRAMMING LANGUAGE

MSC INFORMATION SYSTEMS AND SERVICES

SPECIALIZATION: BIG DATA AND ANALYTICS

PROJECT TITLE: SURVIVAL HEART FAILURE

__LAST NAME:__ PANAGIOTAKOPOULOS

__FIRST NAME:__ GEORGIOS

__REGISTRATION No:__ ME2030

__SUPERVISOR__

SGOUROS NIKOLAOS

Deadline: 28/2/2021

Table of contents

Execution procedure steps 3

Python Code 7

Execution procedure steps

1. We observe the distributions of the variables for the two groups of the dependent variable with histograms and boxplots and we conclude that the data do not need any cleaning. The variables:
    - *Serum_creatinine*
    - *Creatine_phosphokinase*
    - *Platelets*

   are slanted to the right and the variable serum_sodium is slanted to the left.

   We apply the Random Forest method for feature selection and the results are presented in Figure 1. First we observe that the first 7 variables seem to contribute to the classification of the model. We note that initially the variables are normalized with an average value of 0 and a standard deviation of 1. Applying various methods with training and test compartments we finally conclude that the first 3 variables (time, ejection_fraction, serum_creatinine) really contribute to the predictability of the model. These results are in line with the results of the authors of the work we analyze.

2. We calculate the 12 basic Principal Components (PC) and find that the first 11 explain 95% of the variability of the 12 initial characteristics. Repeating the above procedure of Question 1 with the Random Forest method for feature selection, we arrive at Figure 2. The First 5 Basic Components (PC2, PC4, PC3, PC11, PC7) seem to contribute to the model classification. The results of the two methods are presented in Question 3.

3. First of all, we normalize the initial variables (time, ejection_fraction, serum_creatinine) and the Principal Components (PC2, PC4, PC3, PC11, PC7). Then we enter the above two sets of variables in two logistic regression models dividing the data set in half for training (50%) and test (50%) splits, and calculate the area under the curve (Area Under Curve - AUC) see . Figures 3 and 4. The areas are approximately equal AUC = 0.924, with the initial variables (time, ejection_fraction, serum_creatinine), and AUC = 0.927, the basic components (PC2, PC4, PC3, PC11, PC7), with the second area be slightly larger. Finally, we conclude that we can achieve the same and slightly better results with the principal components method (PCA) but in the end we use more variables with the method (PCA). The main goal of PCAs is to reduce the number of features and not to increase it as it happened in our work. We just performed the PCA method experimentally and as an academic exercise.

Figure 1. Classification of Normalized Characteristics of Initial Variables by the Random Forest Method

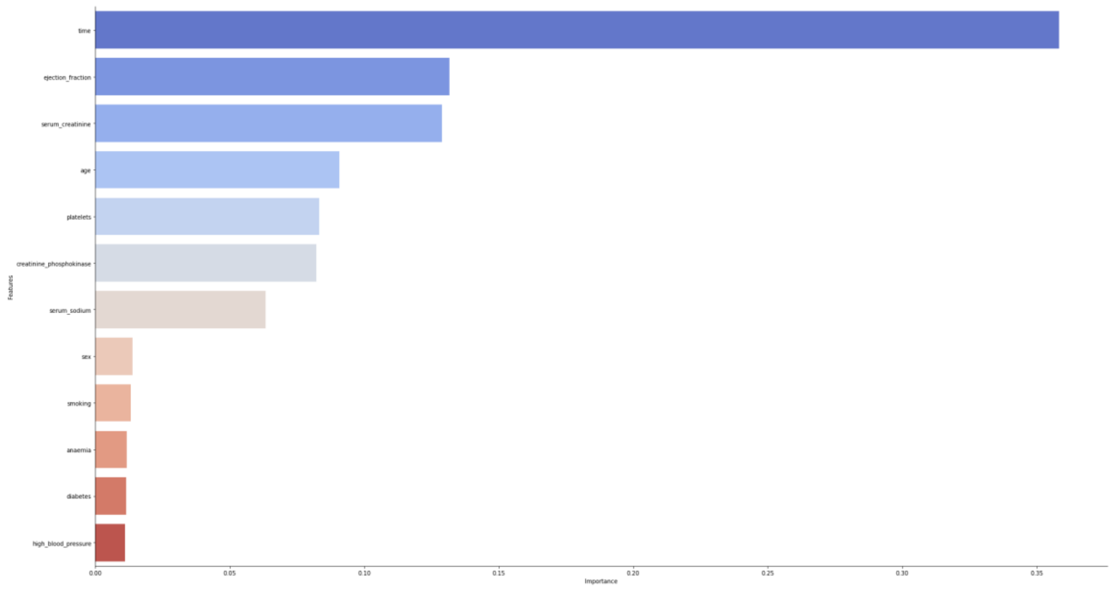


Figure 2. Classification of Normalized Characteristics of Key Components by the Random Forest Method

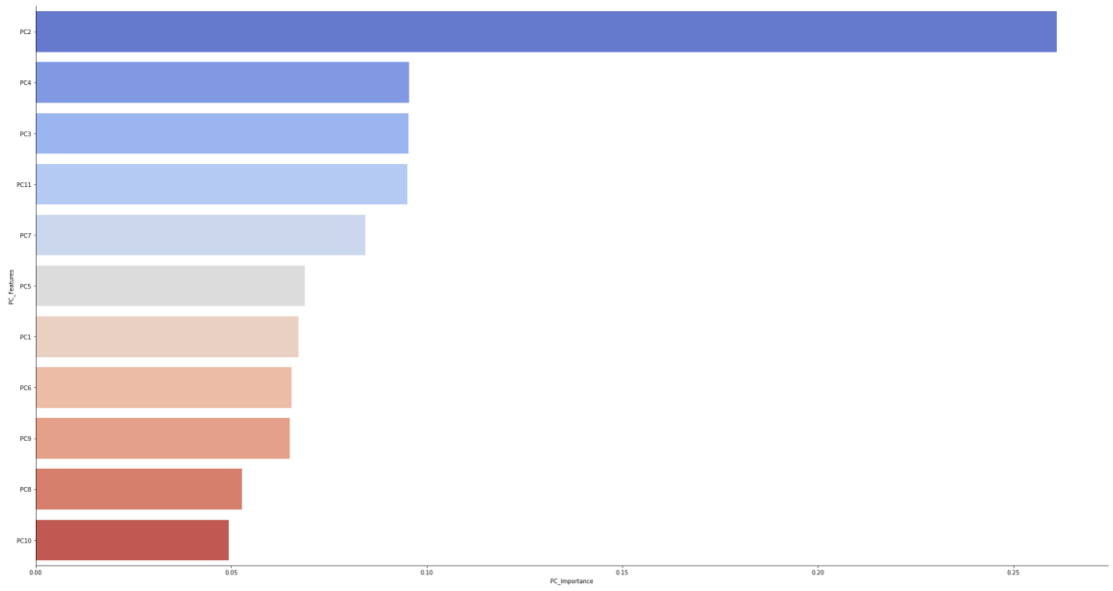


Figure 3. ROC Curve with the method of Logistic Regression with the First Three Basic Characteristics (time, ejection_fraction, serum_creatinine) of Figure 1.

AUC=0.924.

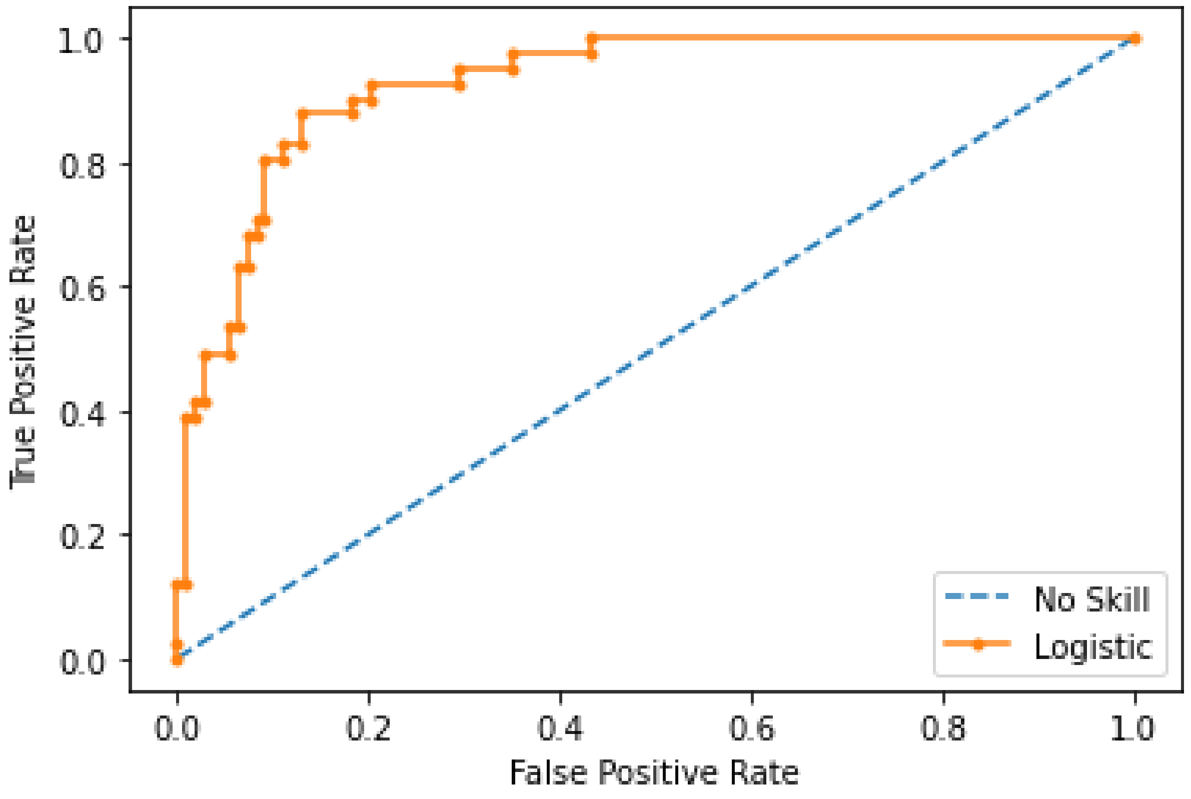
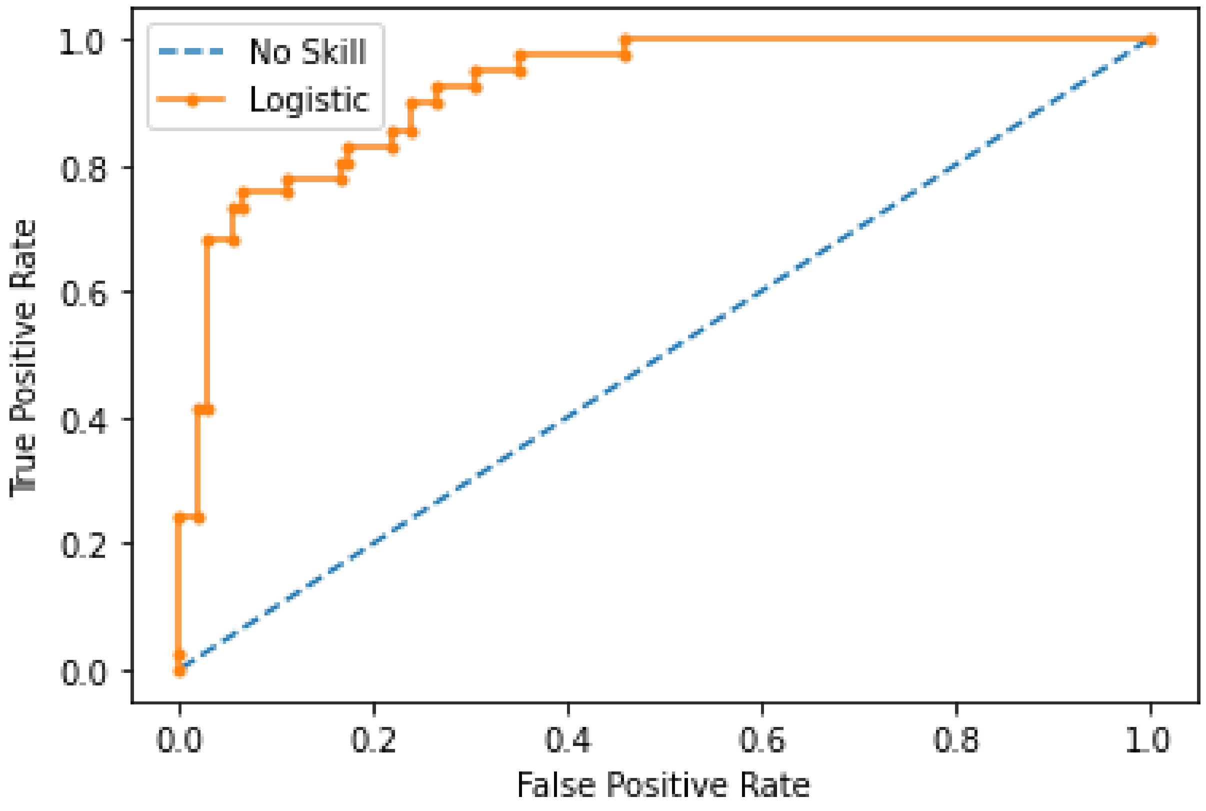


Figure 4. ROC Curve with the method of Logistic Regression with the 5 First Basic Components (PC2, PC4, PC3, PC11, PC7) of Figure 2 (AUC = 0.927).



<u>Results</u>

```
ROC_logistic(X_PC_scores_selected,y)
No Skill: ROC AUC=0.500
Logistic: ROC AUC=0.927
```

```
ROC_logistic(X_feat3,y)
No Skill: ROC AUC=0.500
Logistic: ROC AUC=0.924
```

Python Code

```
import pandas as pd
import numpy as np
import seaborn as sns
import os
from sklearn.preprocessing import StandardScaler
from sklearn.ensemble import RandomForestClassifier
from matplotlib import pyplot
os.chdir("C:\\Users\\Giorgos\\Desktop\\ergasies_metaptxiakwn\\ergasies sgourou")
df = pd.read_csv("heart_failure_clinical_records_dataset.csv")

# data information
df.head()
df.info()

################################################################################
###################### Random Forest Feature Selection #######################
################################################################################
X = df.iloc[:,:-1]
y = df.iloc[:,-1]

coln = X.columns

# Standardizing the features
X = StandardScaler().fit_transform(X)

rf = RandomForestClassifier(n_jobs=-1, n_estimators=50, verbose=3)
rf.fit(X,y)

fi = pd.DataFrame(rf.feature_importances_, coln)
fi.columns = ['Importance']
# Sort the dataframe
fi = fi.sort_values('Importance', ascending=False)
fi['Features'] = fi.index

sns.factorplot(x='Importance', y='Features', data = fi, kind="bar",
        size=14, aspect=1.9, palette='coolwarm')
pyplot.show()
################################################################################

################################################################################
###################### PC Feature Selection #######################
################################################################################
from sklearn.decomposition import PCA
```

```
pca = PCA(.95)

pca.fit(X)

X_PC = pca.transform(X)

X_PC = StandardScaler().fit_transform(X_PC)

rf_PC = RandomForestClassifier(n_jobs=-1, n_estimators=50, verbose=3)
rf_PC.fit(X_PC,y)

fi_PC = pd.DataFrame(rf_PC.feature_importances_)
fi_PC.columns = ['PC_Importance']

# Sort the dataframe
fi_PC['PC_Features'] = ['PC1','PC2','PC3','PC4','PC5','PC6','PC7','PC8','PC9','PC10','PC11']
fi_PC = fi_PC.sort_values('PC_Importance', ascending=False)

sns.factorplot(x='PC_Importance', y='PC_Features', data = fi_PC, kind="bar",
            size=14, aspect=1.9, palette='coolwarm')
pyplot.show()
###############################################################################
from sklearn.linear_model import LogisticRegression
from sklearn.model_selection import train_test_split
from sklearn.metrics import roc_curve
from sklearn.metrics import roc_auc_score

def ROC_logistic(Xmat,ymat):
   trainX, testX, trainy, testy = train_test_split(Xmat, ymat, test_size=0.5, random_state=2)
   # generate a no skill prediction (majority class)
   ns_probs = [0 for _ in range(len(testy))]
   # fit a model
   model = LogisticRegression(solver='lbfgs')
   model.fit(trainX, trainy)
   # predict probabilities
   lr_probs = model.predict_proba(testX)
   # keep probabilities for the positive outcome only
   lr_probs = lr_probs[:, 1]
   # calculate scores
   ns_auc = roc_auc_score(testy, ns_probs)
   lr_auc = roc_auc_score(testy, lr_probs)
   # summarize scores
   print('No Skill: ROC AUC=%.3f' % (ns_auc))
   print('Logistic: ROC AUC=%.3f' % (lr_auc))
```

```
    # calculate roc curves
    ns_fpr, ns_tpr, _ = roc_curve(testy, ns_probs)
    lr_fpr, lr_tpr, _ = roc_curve(testy, lr_probs)
    # plot the roc curve for the model
    pyplot.plot(ns_fpr, ns_tpr, linestyle='--', label='No Skill')
    pyplot.plot(lr_fpr, lr_tpr, marker='.', label='Logistic')
    # axis labels
    pyplot.xlabel('False Positive Rate')
    pyplot.ylabel('True Positive Rate')
    # show the legend
    pyplot.legend()
    # show the plot
    pyplot.show()


X_feat3 = df[['time', 'ejection_fraction', 'serum_creatinine']]

ROC_logistic(X_feat3,y)

pca = PCA(n_components=11)
X_PC_scores = pca.fit_transform(X)

X_PC_scores_selected = X_PC_scores[:,[1,3,2,10,6]]

ROC_logistic(X_PC_scores_selected,y)
```