ΘΕΩΡΙΑ ΑΠΟΦΑΣΕΩΝ

ΕΡΓΑΣΤΗΡΙΑΚΗ ΑΣΚΗΣΗ

ΧΕΙΜΕΡΙΝΟ ΕΞΑΜΗΝΟ 2021-2022

Ονοματεπώνυμα Μελών Ομάδας:

ΤΣΙΓΚΑΣ ΓΕΩΡΓΙΟΣ ,Αριθμός Μητρώου:1054378

ΦΙΛΛΙΠΙΔΗΣ ΠΡΟΔΡΟΜΟΣ,Αριθμός Μητρώου: 1046160

Στην παρούσα αναφορά θα περιγράψουμε το θέμα που επιλέξαμε να ασχοληθούμε στα πλαίσια της Εργαστηριακής Άσκησης του μαθήματος επιλογής «Θεωρία Αποφάσεων» για το χειμερινό εξάμηνο 2021-2022.

Επιλέξαμε να ασχοληθούμε με το πρώτο προσφερόμενο θέμα άσκησης στην σχετική εκφώνηση της εργαστηριακής άσκησης το οποίο είχε να κάνει με το «Dataset 1:Heart Disease Prediction».Όπως μας πληροφορεί και η εκφώνηση, πρόκειται για ένα πρόβλημα δυαδικής ταξινόμησης(binary classification) καθώς δεδομένου ενός πλήθους εξαρτημένων μεταβλητών που θα αναφέρουμε στη συνέχεια πρέπει να προβλέψουμε την τιμή/ανεξάρτητη μεταβλητή «target» που αναπαριστά τον 10ετη κίνδυνο Στεφανιαίας Νόσου,η οποία μπορεί να είναι 1(εμφάνιση καρδιακής νόσου) ή 0 (μη εμφάνιση καρδιακής νόσου).

Αξίζει να δούμε τις εξαρτημένες μεταβλητές σε αυτό το σύνολο δεδομένων ώστε να εξοικειωθούμε με το πλαίσιο του προβλήματος:

1. ***age***
2. ***sex***
3. ***cp :*** chest pain type (4 values)
4. ***restbps :*** resting blood pressure
5. ***chol:*** serum cholestoral in mg/dl
6. ***fbs :*** fasting blood sugar > 120 mg/dl
7. ***restecg :*** resting electrocardiographic results (values 0,1,2)
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
12. ***ca :*** number of major vessels (0-3) colored by flourosopy
13. ***thal:*** 3 = normal; 6 = fixed defect; 7 = reversable defect

Αφού είδαμε τη δομή του συνόλου δεδομένων με το οποίο θα ασχοληθούμε επιχειρούμε να ανακαλύψουμε σχέσεις μεταξύ των εξαρτημένων μεταβλητών με την μεταβλητή-στόχος ‘target’ που αναπαριστά όπως είπαμε την ύπαρξη ή απουσία καρδιακής νόσου:

(Για το παρόν πρόβλημα θα χρησιμοποιήσαμε το περιβάλλον Anaconda και το εργαλείο Jupyter Notebook που προσφέρει,και την γλώσσα προγραμματισμού Python με έκδοση 3.8.5).

Φορτώνουμε τις σχετικές βιβλιοθήκες που θα χρησιμοποιήσουμε:

In [336]:
```
import pandas as pd
import re
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
import matplotlib.pyplot as plt
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LogisticRegression
from sklearn.linear_model import LinearRegression
from sklearn.neighbors import KNeighborsClassifier
from sklearn.ensemble import RandomForestClassifier
from sklearn.metrics import roc_curve, auc
from xgboost import XGBClassifier
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import accuracy_score
from sklearn.metrics import classification_report
from sklearn.svm import SVC
from sklearn.naive_bayes import GaussianNB
from sklearn.preprocessing import StandardScaler
from pandas import DataFrame
from sklearn.tree import DecisionTreeClassifier
from sklearn.metrics import accuracy_score,f1_score,recall_score,precision_score
from sklearn.preprocessing import StandardScaler
from imblearn.over_sampling import SMOTE
smote = SMOTE(sampling_strategy='not majority')
from imblearn.under_sampling import RandomUnderSampler
from keras import backend as K

#from sklearn.preprocessing import LabelEncoder
from keras.wrappers.scikit_learn import KerasClassifier
from keras.utils import np_utils

from keras.models import Sequential
from keras.layers import Dense, Dropout
from keras.callbacks import EarlyStopping

import warnings
warnings.filterwarnings('ignore')
```

Εικόνα 1:Importing necessary libraries

Έπειτα αφού φορτώσουμε το dataset που έχει δοθεί για το πρόβλημα με την εντολή read_csv

In [290]:
```
#heart_df = pd.read_csv(r"C:\Dataset 1_Heart.csv")

heart_df = pd.read_csv(r"C:\DataHeartTest.csv")
```

Εικόνα 2:Loading the dataset

,θα εκτελέσουμε την εντολή “heart_df.describe()” η οποία θα μας δώσει στατιστικές πληροφορίες για το σύνολο δεδομένων καθώς και μέγιστες ή ελάχιστες τιμές για κάθε μεταβλητή χαρακτηριστικό:

In [8]:
```
heart_df.describe()
```

Out[8]:

	age	sex	cp	trestbps	chol	fbs	restecg	thalach	exang	oldpeak	slope	ca	th
ount	303.000000	303.000000	303.000000	303.000000	303.000000	303.000000	303.000000	303.000000	303.000000	303.000000	303.000000	303.000000	303.0000
ean	54.366337	0.683168	0.966997	131.623762	246.264026	0.148515	0.528053	149.646865	0.326733	1.039604	1.399340	0.729373	2.3135
std	9.082101	0.466011	1.032052	17.538143	51.830751	0.356198	0.525860	22.905161	0.469794	1.161075	0.616226	1.022606	0.6122
min	29.000000	0.000000	0.000000	94.000000	126.000000	0.000000	0.000000	71.000000	0.000000	0.000000	0.000000	0.000000	0.0000
25%	47.500000	0.000000	0.000000	120.000000	211.000000	0.000000	0.000000	133.500000	0.000000	0.000000	1.000000	0.000000	2.0000
50%	55.000000	1.000000	1.000000	130.000000	240.000000	0.000000	1.000000	153.000000	0.000000	0.800000	1.000000	0.000000	2.0000
75%	61.000000	1.000000	2.000000	140.000000	274.500000	0.000000	1.000000	166.000000	1.000000	1.600000	2.000000	1.000000	3.0000
max	77.000000	1.000000	3.000000	200.000000	564.000000	1.000000	2.000000	202.000000	1.000000	6.200000	2.000000	4.000000	3.0000

Εικόνα 3:Statistical Info for our dataset

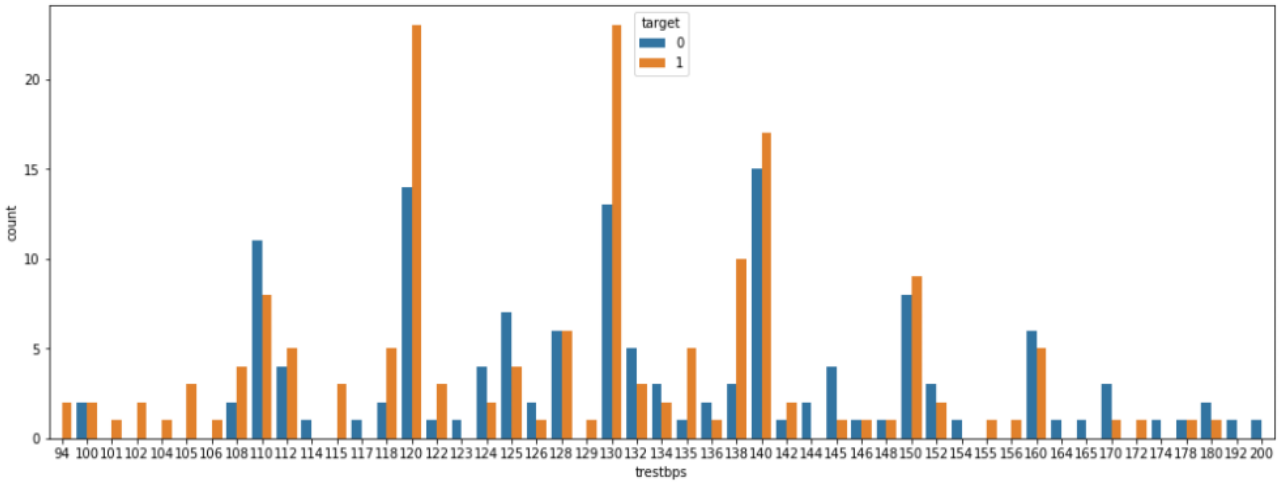
Παρατηρούμε πως για η μεταβλητή “ca” παίρνει μέγιστη τιμή το 4.0 το οποίο δεν πρέπει να συμβαίνει καθώς το εύρος τιμών της είναι το (0-3) όπως έχουμε αναφέρει προηγουμένως.Επίσης για την μεταβλητη thal παρατηρούμε ότι έχει ελάχιστη τιμή 0 η οποία αντιστοιχεί μόνο σε δύο γραμμές από τις αρχικές 303 του συνόλου και επίσης η υπαρξη αυτής της τιμής υπερβαίνει το πλήθος των τιμών που μπορεί να πάρει το thal το οποίο είναι 3 τιμές, συνεπώς θεωρούμε την τιμή αυτή ως outlier που θα πρέπει να αφαιρέσουμε.Συνεπώς πρέπει να αφαιρέσουμε από το σύνολο δεδομένων τις γραμμές εκείνες με τιμή ca > 3 και thal μεγαλύτερο του 0 (έχουμε βεβαιωθεί ότι δεν υπάρχει άλλη τιμή που βρίσκεται ενδιάμεσα σε αυτές που προαναφέρθηκαν για αυτές τις μεταβλητές μέσα από σχετική ανάλυση που είναι εμφανής στον κώδικα μας):

```
#thal  A blood disorder called thalassemia (1 = normal; 2 = fixed defect; 3 = reversable defect), we drop the 0 value since it ap
#thal_zero = heart_df[heart_df['thal'] == 0]
#print("Patients with 0 thal" ,thal_zero)

heart_df = heart_df[heart_df['thal'] > 0]

# we remove the rows with ca = 4 which is faulty since it goes from 0 to 3
heart_df[heart_df['ca'] == 4]
heart_df = heart_df[heart_df['ca'] < 4]
```

Στη συνέχεια προχωράμε σε γραφική αναπαράσταση της σχέσης των εξαρτημένων μεταβλητών του συνόλου δεδομένων με την ανεξαρτητη μεταβλητή-στόχο «target»:

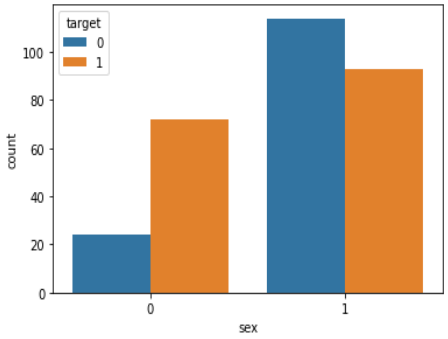
Παρουσιάζουμε ένα μέρος της ανάλυστηςτής εδώ με κάποια screenshots:

```
In [61]: ax = plt.subplots(figsize=(17, 6))
         ax = sns.countplot(x='trestbps', hue="target", data=heart_df)

         plt.show()
```



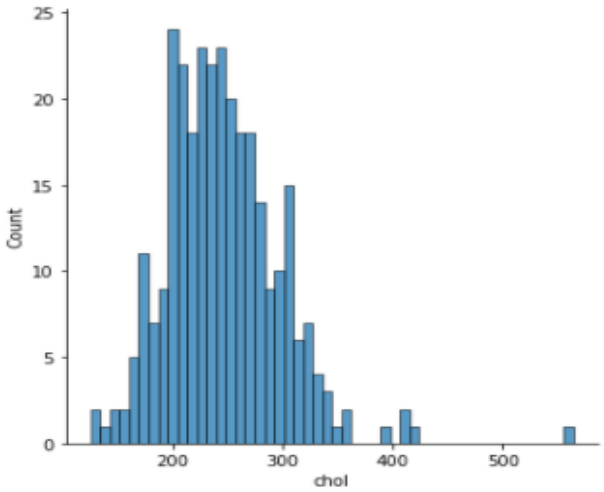
Εικόνα 4:Frequency of trestbps values

```
In [24]: ax = sns.countplot(x='sex', hue="target", data=heart_df)#1-male,0-female

         plt.show()
```



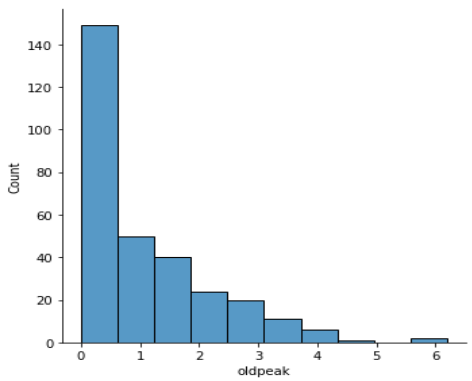
Εικόνα 5:Target value counts for the 2 sexes

```
In [68]: #chol

         ax = sns.displot(heart_df,x = "chol",bins=50)
         plt.show()
```



Εικόνα 6:Distribution of chol values count in bins

```
[81]: ax = sns.displot(heart_df,x = "oldpeak",bins=10)
      plt.show()
```

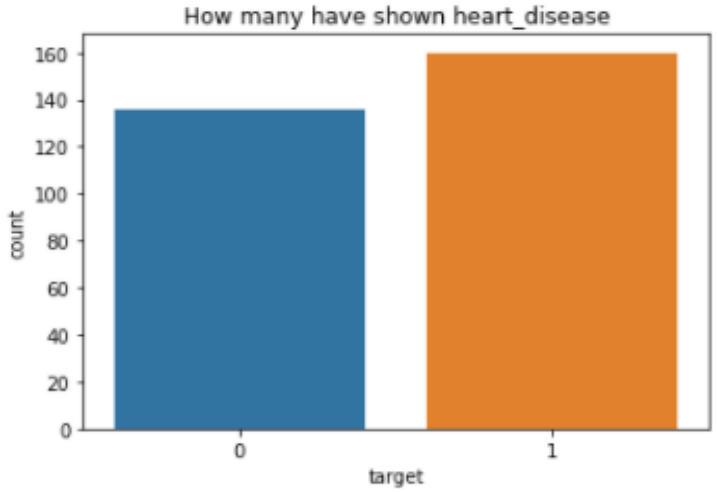


Εικόνα 7:oldpeak value counts distribution

Από την εικόνα 5 παρατηρούμε πως οι γυναίκες συγκριτης με τους άνδρες είχαν μεγαλύτερο ποσοστό εμφανίσεων καρδιακής νόσου (target = 1) ενώ στην εικόνα 7 παρατηρούμε πως ένα μεγάλο ποσοστό των τιμών της μεταβλητής “oldpeak" είναι 0 η οποία συνδέεται με την “ST depression” κατά την διάρκεια άσκησης σε σχέση με ξεκούραση.

‘Επειτα παρατηρούμε την κατανομή των τιμών της ανεξάρτητης μεταβλητής target το οποίο θα μας πληροφορήσει για το πόσο ισορροπημένο είναι το σύνολο δεδομένων που έχουμε:

```
In [5]: sns.countplot(x='target', data=heart_df)
        plt.title("How many have shown heart_disease")
        plt.show()
```



Παρατηρούμε πως το πλήθος των περιπτώσεων που δεν εμφανίζεται καρδιακή νόσος(target = 0) και αυτών που εμφανίζεται (target = 1) είναι συγκριτικά κοντά μεταξύ τους συνεπώς το σύνολο δεδομένων μας είναι καλά ισορροπημένο σε σχέση με την ανεξάρτητη μεταβλητή που επιθυμούμε να προβλέψουμε.

Στη συνέχεια επιχειρούμε να παρατηρήσουμε τη συσχέτιση μεταξύ των εξαρτημένων μεταβλητών/χαρακτηριστικών με την μεταβλητή στόχο ‘target’ του συνόλου δεδομένων μας εκτελώντας τις εντολές:

```
In [26]: correlation = heart_df.corr()

In [27]: correlation['target'].sort_values(ascending=False)

         #we calculate the correlation of the dataset variables with the target variable
         #the correlation coefficient has a range from from -1 to +1.
         #when the coefficient is close to +1 this means a strong positive correlation and when the coefficient is close to -1 this means
         #when the coefficient is close to 0 then that means that there is no correlation

Out[27]: target      1.000000
         thalach     0.426655
         cp          0.423425
         slope       0.337825
         restecg     0.131716
         fbs        -0.004680
         chol       -0.076541
         trestbps   -0.148922
         age        -0.225453
         sex        -0.285322
         thal       -0.364399
         exang      -0.425085
         oldpeak    -0.428804
         ca         -0.467158
         Name: target, dtype: float64
```

Εικόνα 8:Correlation of dependent variables with target

Όπως γνωρίζουμε μπορούμε να έχουμε θετική ή αρνητική συσχέτιση μεταξύ δυο μεταβλητών και στην προκειμένη περίπτωση η τιμή +1 αποτελεί την μέγιστη τιμή της θετικής συσχέτισης και το -1 την μέγιστη τιμή της αρνητικής συσχέτισης. Η τιμή της συσχέτισης όσο πλησιάζει στο +1 μας πληροφορεί πως η συσχέτιση της μεταβλητής σε σχέση με κάποιο άλλο στόχο μεταβλητή(εδώ το 'target') είναι τέτοια έτσι ώστε καθώς αυξάνεται η μεταβλητή στόχος θα αυξάνεται και η μεταβλητή για την οποία βρίσκουμε την συσχέτιση αυτή,κατά ανάλογο ρυθμό. Παρόμοια και για την περίπτωση που η τιμή της συσχέτισης πλησιάζει στο -1 το οποίο μας πληροφορεί ότι καθώς η τιμή της μεταβλητής στόχου αυξάνεται ή μειώνεται κατά ανάλογο ρυθμό η μεταβλητή για την οποία βρίσκουμε την συσχέτιση με αυτήν κάνει αντίθετη αλλαγή από αυτήν(δηλαδή μείωση όταν η μεταβλητή στόχος αυξάνεται και αντίστοιχα για αύξηση).Όταν η τιμή της συσχέτισης είναι κοντά στο 0 αυτό μας πληροφορεί πως αλλαγές στην τιμή ενός εκ των δύο μεταβλητών για τις οποίες βρίσκουμε την συσχέτισή τους, δεν έχει ως αποτέλεσμα σημαντικές αλλαγές στην τιμή της άλλης μεταβλητής.

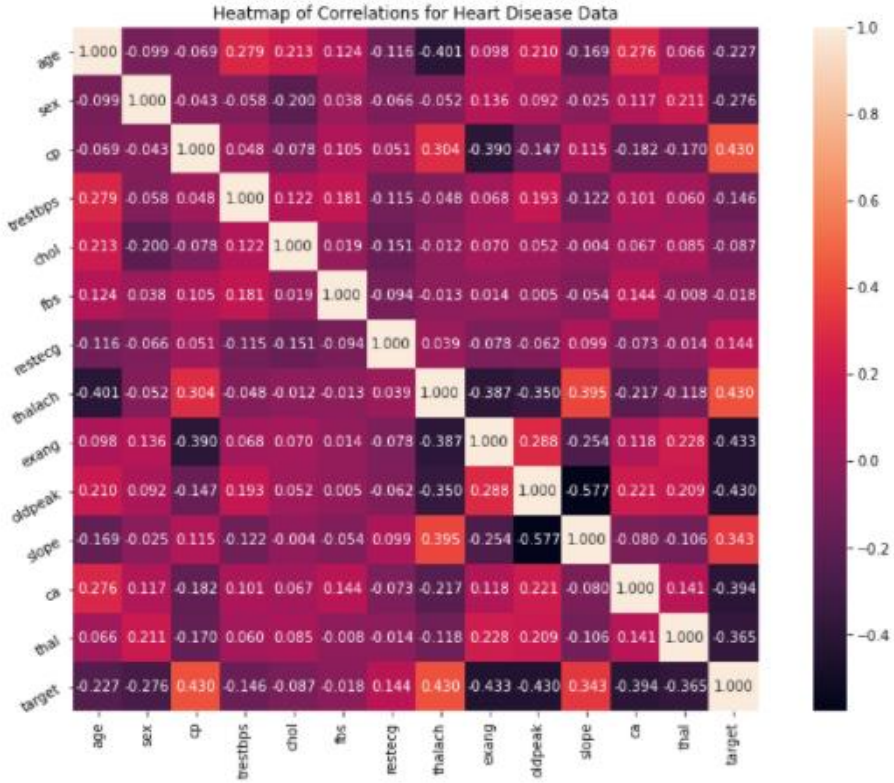
Από την εκτέλεση των εντολών που έχουμε στην εικόνα 8 παρατηρούμε πως οι μεταβλητές "fbs"(fasting blood sugar) και "chol"(χοληστερίνη) έχουν τιμές συσχέτισης με την μεταβλητή target *-0.004680* και *-0.076541* αντίστοιχα, οι οποίες είναι πολύ κοντά στο 0 συνεπώς αυτές οι δύο μεταβλητές επηρεάζουν ελάχιστα την τιμή μεταβλητή στόχος "target",άρα συνεπώς και την εμφάνιση σχετικής καρδιακής νόσου.

Τις μεγαλύτερες τιμές θετικής συσχέτισης με την μεταβλητή "target" παρατηρούμε ότι έχουν οι μεταβλητές "thalach"(maximum heart rate achieved),"cp"(chest pain) και "slope"(slope of the peak exercise ST segment) με τιμές *+0.426655,+ 0.423425, 0.337825* το οποίο μας πληροφορεί συγκεκριμένη αύξηση τιμών αυτών έχει ως αποτέλεσμα αυξημένη πιθανότητα για το 'target' να είναι 1 , δηλαδή να εμφανιστεί καρδιακή νόσος.

Από την άλλη οι μεταβλητές "thal","exang"(exercise induced angina),"oldpeak"(ST depression induced by exercise relative to rest),ca(number of major vessels) έχουν την μεγαλύτερη αρνητική συσχέτιση με τη μεταβλητή target με τιμές *-0.364399, -0.425085, -0.428804,-0.467158* αντίστοιχα το οποίο σημαίνει ότι πιθανή αύξηση της τιμής τους μπορεί να μειώσει την πιθανότητα να εμφανιστεί καρδιακή νόσος δηλαδή το target να γίνει ίσο με 1.

Στη συνέχεια κάνουμε μια πιο λεπτομερή μελέτη για τις συσχετίσεις των μεταβλητών στο σύνολο δεδομένων που έχουμε και παράγουμε ένα correlation heatmap όπως φαίνεται παρακάτω:

```
In [119]: plt.figure(figsize=(14,9))
          plt.title('Heatmap of Correlations for Heart Disease Data')
          heat = sns.heatmap(correlation, square=True, annot=True, fmt='.3f', linecolor='white')
          heat.set_xticklabels(heat.get_xticklabels(), rotation=90)
          heat.set_yticklabels(heat.get_yticklabels(), rotation=30)
          plt.show()
```



Εικόνα 9:Correlation Heatmap

Με τον ίδιο τρόπο όπως παρατηρήσαμε τις τιμές συσχέτισης για τις εξαρτημένες μεταβλητές σε σχέση με το target μπορούμε να εξάγουμε χρήσιμες πληροφορίες για τις σχέσεις μεταξύ των εξαρτημένων μεταβλητών του συνόλου δεδομένων μας.Ενδεικτικά παρατηρούμε πως:

- αξιοσημείωτη αρνητική συσχέτιση μεταξύ του “oldpeak” και “slope” (-0.577)
- θετική συσχέτιση μεταξύ της μεταβλητής “thalach” και “target”(+0.430)
- “thal” και “fbs” συσχέτιση είναι πολύ κοντά στο 0

Στη συνέχεια θα προσπαθήσουμε να εντοπίσουμε πιθανούς outliers στο σύνολο δεδομένων μας με μια σειρά μεθόδων:

Αρχικά ορίζουμε την συνάρτηση **spot_outlier** η οποία παίρνει ως είσοδο ένα dataframe(df) που εδώ είναι το σύνολο δεδομένων μας και δυο χαρακτηριστικά/μεταβλητές που θα χρησιμοποιηθούν για να δούμε πιθανούς outliers που εμφανίζονται :

Συγκεκριμένα ορίζουμε την παρακάτω συνάρτηση:

```
In [274]: def spot_outlier(df,x_axis,y_axis):#this function will take a dataframe as input and 2 features in the form of x_axis and y_axis

              xaxis_df = pd.DataFrame(data=x_axis)
              yaxis_df = pd.DataFrame(data=y_axis)
              #print(feat_df.columns)
              x_feat = str(xaxis_df.columns)
              y_feat = str(yaxis_df.columns)

              plt.scatter(x=x_axis[df.target==1], y=y_axis[(df.target==1)], c="red")
              plt.scatter(x=x_axis[df.target==0], y=y_axis[(df.target==0)])
              plt.legend(["Disease", "Not Disease"])
              plt.xlabel(x_feat)
              plt.ylabel(y_feat)
              plt.show()
```

Figure 10:Spot Outlier Function

Εκτελούμε την παραπάνω συνάρτηση με ορίσματα “age” και “thalach” και παρατηρούμε από το γράφημα που παράγεται πως υπάρχουν κάποιοι outliers που αποκλίνουν από το κύριο cluster που σχηματίζεται:

```
In [305]: spot_outlier(heart_df,heart_df['age'],heart_df['thalach'])
```

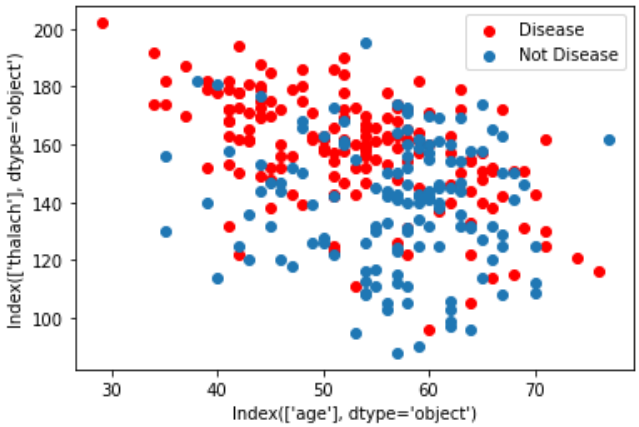


Figure 11:Outlier Detection for thalach and age

Θα εφαρμόσουμε μια μέθοδο εντοπισμού outliers που θα μας επιτρέψει να αφαιρέσουμε κάποιους από αυτούς με μεγαλύτερη ακρίβεια από το σύνολο δεδομένων μας,η μέθοδος αυτή που θα χρησιμοποιήσουμε ονομάζεται **Z-score** μέθοδος.Με αυτήν την μέθοδο θα εντοπίσουμε σημεία-δεδομένα στο σύνολο δεδομένων μας που διαφέρουν σημαντικά σε σχέση με άλλα σημεία και τα οποία θα αφαιρέσουμε καθώς θα αποτελούν outliers:

```
In [294]: #Z-score method

          #If the z score of a data point is more than 3 (because it cover 99.7% of area), it indicates that the data value is quite differ
          #A Z-score of zero represents a value that equals the mean. The further away an observation's Z-score is from zero, the more unus
          out=[]
          def Zscore_outlier(heart_df):
              m = np.mean(heart_df)
              sd = np.std(heart_df)
              for i in heart_df:
                  z = (i-m)/sd
                  if np.abs(z) > 3:
                      out.append(i)
              print("Outliers:",out)
          Zscore_outlier(heart_df['chol'])
          #we see that the value 564 is an outlier
```

Figure 12:Zscore_outlier function

Στη συνέχεια ορίζουμε την συνάρτηση **calc_zscore** η οποία καλεί την **Zscore_outlier** για να εντοπίσει outliers για κάθε χαρακτηριστικό/εξαρτημένη μεταβλητή του συνόλου δεδομένων μας:

```
In [318]: def calc_zscore():
              for feature in outl_list:
                  print("The possible outliers for "+feature+ " are ")
                  Zscore_outlier(heart_df[feature])

In [319]: calc_zscore()

The possible outliers for cp are
Outliers: [417, 564, 407, 409, 71]
The possible outliers for trestbps are
Outliers: [417, 564, 407, 409, 71, 200, 192]
The possible outliers for chol are
Outliers: [417, 564, 407, 409, 71, 200, 192, 417, 394, 407, 409]
The possible outliers for fbs are
Outliers: [417, 564, 407, 409, 71, 200, 192, 417, 394, 407, 409]
The possible outliers for restecg are
Outliers: [417, 564, 407, 409, 71, 200, 192, 417, 394, 407, 409]
The possible outliers for thalach are
Outliers: [417, 564, 407, 409, 71, 200, 192, 417, 394, 407, 409]
The possible outliers for exang are
Outliers: [417, 564, 407, 409, 71, 200, 192, 417, 394, 407, 409]
The possible outliers for oldpeak are
Outliers: [417, 564, 407, 409, 71, 200, 192, 417, 394, 407, 409, 6.2, 5.6]
The possible outliers for slope are
Outliers: [417, 564, 407, 409, 71, 200, 192, 417, 394, 407, 409, 6.2, 5.6]
The possible outliers for ca are
Outliers: [417, 564, 407, 409, 71, 200, 192, 417, 394, 407, 409, 6.2, 5.6]
The possible outliers for thal are
Outliers: [417, 564, 407, 409, 71, 200, 192, 417, 394, 407, 409, 6.2, 5.6]
```

Figure 13:Possible outliers using zscore

Από την παραπάνω έξοδο παρατηρούμε πως οι τιμές 5.6,6.2,221,204,71 μπορούν να αφαιρεθούν ως outliers καθώς στα σύνολα τιμών που βρίσκονται απέχουν σημαντικά από τις υπόλοιπες τιμές στο ίδιο σύνολο.

ΕΦΑΡΜΟΓΗ ΜΟΝΤΕΛΩΝ ΜΗΧΑΝΙΚΗΣ ΜΑΘΗΣΗΣ

Έπειτα από την διερεύνηση του συνόλου δεδομένων μας για σχέσεις μεταξύ των εξαρτημένων μεταβλητών και μεταξύ αυτών και της μεταβλητής στόχου “target” καθώς και την αφαίρεση outliers ή τιμών που αποτελούσαν θόρυβο στα δεδομένα μας θα προχωρήσουμε στη κατασκευή μοντέλων μηχανικής μάθησης για το σύνολο δεδομένων που έχουμε:

Ορίζουμε την συνάρτηση **doPredicts(X_train,Y_train,X_test,Y_test,X_np)** η οποία εφαρμόζει μια σειρά μοντέλων μηχανικής μάθησης στο σύνολο δεδομένων μας αφού έχει χωριστεί για εκπαίδευση ώστε να παραχθούν τα σύνολα παράμετροι που βλέπουμε στον ορισμο της.Τα μοντέλα αυτά περιλαμβάνουν **Logistic Regression,SVM classifier,Naïve Bayes Classifier,Decision Tree classifier,KNN classifier(k-nearest neighbors),Random Forest classifier,XGBoost classifier και ένα νευρωνικό δίκτυο** που κατασκευάσαμε.

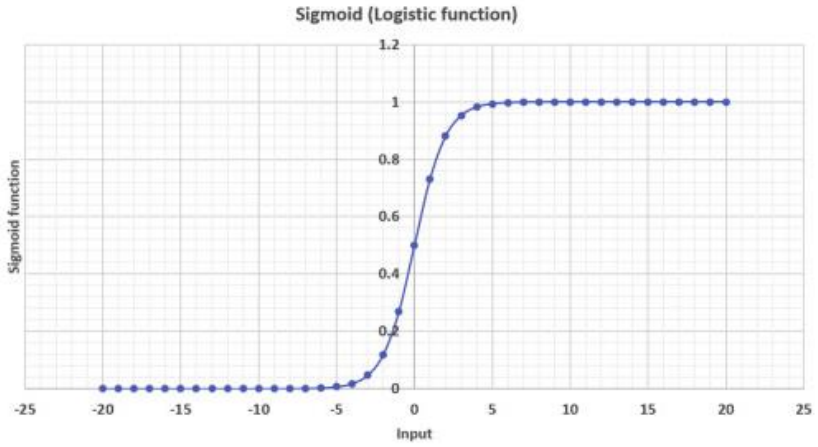
Κατ’ αρχήν ας δούμε συνοπτικά τι αποσκοπεί να πετύχει κάθε μοντέλο ταξινόμησης:

- **Logistic Regression**

Το μοντέλο του Logistic Regression είναι ένας αλγόριθμος επιβλεπόμενης μηχανικής μάθησης που βασίζεται στην λογιστική συνάρτηση :

$$F(x) = \frac{1}{1+e^{-x}}$$

Η μορφή της συνάρτησης είναι η παρακάτω:



Εικόνα 14:Logistic Regression Function Example

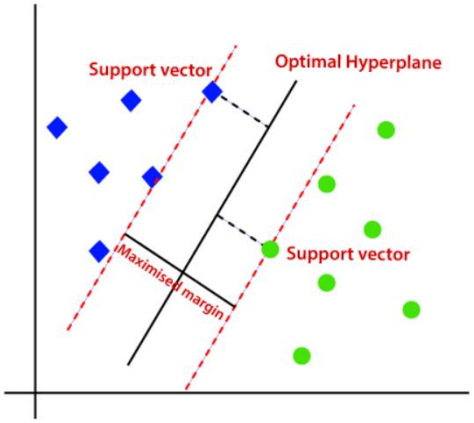
Το μοντέλο του Logistic Regression συνήθως προτιμάται για προβλήματα δυαδικής ταξινόμησης(όταν συνήθως η μεταβλητή στόχος είναι κατηγορική).

- **SVM Classifier (Support Vector Machine)**

Σκοπός του αλγορίθμου επιβλεπόμενης μάθησης Support Vector Machine (SVM) είναι να βρει ένα υπερεπίπεδο όπου οι 2 κλάσεις είναι διαχωρίσιμες. Υπερεπίπεδο ονομάζουμε τον Ευκλείδειό χώρο n-1 διαστάσεων τον οποίο χωρίζει τον χώρο n διαστάσεων σε 2 ξεχωριστά τμήματα.

Παράδειγμα SVM:

Έστω ότι έχουμε το παρακάτω γράφημα για την απεικόνιση των δεδομένων:



Εικόνα 15:SVM classifier example

Όπως βλέπουμε στο γράφημα στις 2 διαστάσεις τα δεδομένα είναι διαχωρίσιμα και το υπερεπίπεδο είναι η γραμμή που τα διαχωρίζει. Τα support vectors είναι τα ευθύγραμμα τμήματα που περνάνε από το κοντινότερο σημείο της κάθε κλάσης από το υπερεπίπεδο. Ο αλγόριθμος βρίσκει το βέλτιστο υπερεπίπεδο μεγιστοποιώντας την απόσταση (margin) μεταξύ των support vectors. Όταν οι κλάσεις δεν είναι γραμμικά διαχωρίσιμες ο αλγόριθμος προσπαθεί να βρει το υπερεπίπεδο σε μεγαλύτερη διάσταση έτσι ώστε να είναι.

- **Naïve Bayes Classifier(Gaussian Naïve Bayes)**

Ο Naive Bayes Classifier φτιάχνει ένα μοντέλο με βάση τις δεσμευμένες πιθανότητες κάθε χαρακτηριστικού δεδομένου ότι ανήκουν(target=1) ή όχι(target=0) στην κλάση 1.

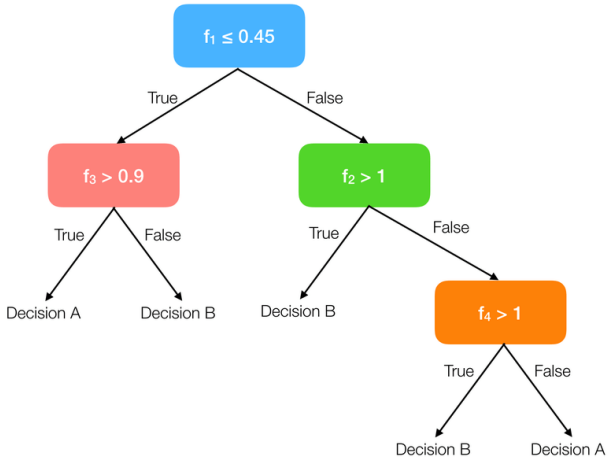
Ο Gaussian Naïve Bayes ταξινομητής που χρησιμοποιούμε στα πλαίσια της άσκησης θεωρεί πιθανότητες για τα χαρακτηριστικά που ακολουθούν την Gaussian κατανομή ως εξής:

$$P(x_i \mid y) = \frac{1}{\sqrt{2\pi\sigma_y^2}} \exp\left(-\frac{(x_i - \mu_y)^2}{2\sigma_y^2}\right)$$

Εικόνα 16:Gaussian Naive Bayes Formula

- **Decision Tree Classifier**

Τα δένδρα απόφασης είναι μια επιβλεπόμενη μέθοδος μηχανικής μάθησης που χρησιμοποιούνται για ταξινόμηση και παλινδρόμηση. Ο στόχος είναι να δημιουργηθεί ένα μοντέλο που θα προβλέπει την τιμή της μεταβλητή-στόχου μαθαίνοντας απλούς κανόνες απόφασης που παράγονται από χαρακτηριστικά/εξαρτημένες μεταβλητές του συνόλου δεδομένων μας.



Εικόνα 17:Sample Decision Tree

- **K-Nearest Neighbors Classifier**

Η ταξινόμηση που βασίζεται σε "γείτονες" δεδομένα αποτελεί ένα μοντέλο μηχανικής μάθησης που επιχειρεί να ταξινομήσει το σύνολο δεδομένων που δέχεται ως είσοδο βασιζόμενο σε ένα απλό ψήφο πλειοψηφίας των κοντινότερων γειτόνων κάθε σημείου-δεδομένου και τελικά κάθε σημείο-δεδομένο κατατάσσεται στην κλάση δεδομένων για την οποία έχει τους περισσότερους αντιπρόσωπους κοντινότερους γείτονες του.

Στην K-Nearest Neighbors ταξινόμηση επιλέγουμε ένα αριθμό k έστω γειτόνων γύρω από κάθε σημείο-δεδομένο για το οποίο θα εκτελέσουμε την παραπάνω διαδικασία που περιγράφηκε προκειμένου να το ταξινομήσουμε σε μια κλάση δεδομένων βασιζόμενοι στον αριθμό k γειτόνων που έχουμε επιλέξει.

- **Random-Forest Classifier**

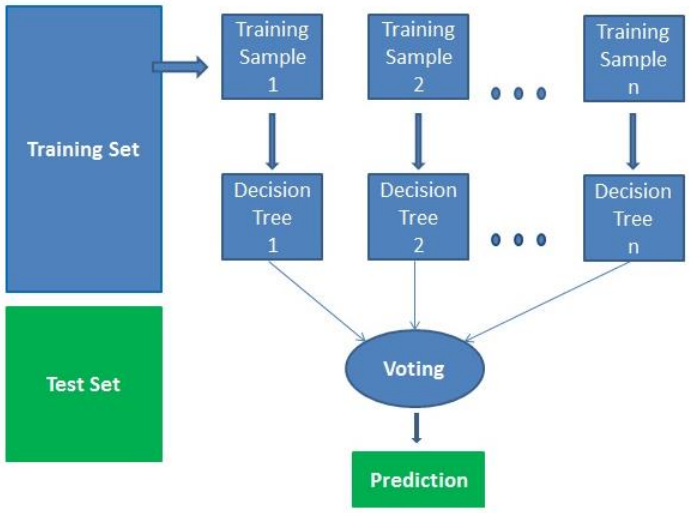
Ο αλγόριθμος Random-Forest αποτελεί μια επιβλεπόμενη μέθοδο μηχανικής μάθησης που μπορεί να χρησιμοποιηθεί για ταξινόμηση ή και για παλινδρόμηση.Ο αλγόριθμος ακολουθεί τα εξής βήματα:

1)Επιλέγει τυχαία δείγματα από ένα επιλεγμένο σύνολο δεδομένων.

2)Κατασκευάζει ένα δένδρο απόφασης για κάθε δείγμα και λαμβάνει ένα αποτέλεσμα πρόβλεψης από κάθε δένδρο απόφασης

3)Πραγματοποιεί ψηφοφορία για κάθε αποτέλεσμα πρόβλεψης

4)Επιλέγει το αποτέλεσμα πρόβλεψης με τις περισσότερες ψήφους σαν την τελική πρόβλεψη



Εικόνα 18:Random Forest algorithm process

- **XGBoost(Extreme Gradient Boosting)**

Ο αλγόριθμος XGBoost χρησιμοποιεί την τεχνική του Gradient Boosting προκειμένου να κάνει προβλέψεις για ένα σύνολο δεδομένων.Η τεχνική του Boosting συνδυάζει ένα σύνολο από "weak learners" και προσφέρει βελτιωμένη ακρίβεια πρόβλεψης. Σε οποιαδήποτε φάση/στιγμιότυπο (instant) έστω t του αλγορίθμου οι έξοδοι του μοντέλου αποκτούν κατάλληλο βάρος(weight) ανάλογα με τις εξόδους της προηγούμενης φάσης t-1.Τα αποτελέσματα που αποτέλεσαν σωστές προβλέψεις θα αποκτούν μικρότερο βάρος από αυτά που ταξινομήθηκαν λάθος(misclassified) τα οποία αποκτούν μεγαλύτερο βάρος.Επίσης ως "weak learners" θεωρούμε αυτούς που είναι λίγο καλύτεροι στις προβλέψεις του από προβλέπτες που κάνουν τυχαίο μάντεμα(λίγο καλύτεροι από ποσοστό σωστής πρόβλεψης 50%).

- **Νευρωνικό Δίκτυο(Neural Network)**

Κατασκεύασαμε ένα απλό νευρώνικό δίκτυο 3 επιπέδων(1 επίπεδο εισόδου, 1 εσωτερικό επίπεδο και 1 επίπεδο εξόδου) χρησιμοποιώντας την βιβλιοθήκη keras.Συγκεκριμένα ορίζουμε ότι το μοντέλο μας θα αποτελεί μια απλή τοποθέτηση νευρωνικών επιπέδων καλώντας την κλάση Sequential().Ορίζουμε στο πρώτο επίπεδο το σχήμα της εισόδου που θα είναι η πρώτη διάσταση του συνόλου X αφού έχει μετατραπεί σε numpy array(το σύνολο X αποτελεί το σύνολο δεδομένων μας μετα από προεπεξεργασία που έχει αναφερθεί παραπάνω,αφαίρεση της στήλης target και χρήση dummy columns προκειμένου να κωδικοποιηθούν κατάλληλα τα κατηγορικά χαρακτηριστικά).

Στη συνάρτηση **doPredicts()** ορίζουμε μοντέλα που αντιστοιχούν σε κάθε ένα από τα παραπάνω που περιγράψαμε. Συγκεκριμένα αξίζει να σημειωθούν κάποια πράγματα για τις ρυθμίσεις που χρησιμοποιήσαμε για κάποια από αυτά καθώς και κατά περίπτωση η λειτουργία τους:

- Στο μοντέλο Logistic Regression που χρησιμοποιούμε επιλέγουμε solver liblinear διότι λειτουργεί καλύτερα για μικρότερα σύνολα δεδομένων σαν το δικό μας.
- Στα μοντέλα SVM,GaussianNB,Decision Tree Classifier χρησιμοποιήσαμε τις προκαθορισμένες παραμέτρου κάθε μοντέλου.
- Στην μέθοδο KNN(K-nearest neighbors) χρησιμοποιήσαμε της προκαθορισμένη παράμετρο των k=5 γειτόνων.
- Στην μέθοδο Random Forest χρησιμοποιήσαμε την επιλογή n=1000 εκτιμητές(estimators).
- Στο νευρωνικό δίκτυο που κατασκεύασαμε χρησιμοποιήσαμε την τεχνική του **Early Stopping**(Πρόωρο σταμάτημα) προκειμένου να αποφύγουμε το **overfitting** που θα μπορούσε να προκληθεί από την εκπαίδευση, δηλαδή την περίπτωση που το μοντέλο προσαρμόζεται σε τέτοιο βαθμό στο σύνολο δεδομένων που του δίνουμε ως είσοδο με αποτέλεσμα να έχει αρκετά χειρότερη απόδοση και ακρίβεια πρόβλεψης για δεδομένα που δεν έχουν δοθεί ακόμα σε αυτό.Για το early stopping θέτουμε ως τη μεταβλητή με βάση την οποία θα κρίνουμε για σταμάτημα της διαδικασίας μάθησης την μετρική "accuracy"(ακρίβεια πρόβλεψης) και θέτουμε την patience(υπομονή) ως 7 καθώς παρατηρήσαμε ότι μας δίνει τα καλύτερα αποτελέσματα σε σχέση με άλλες τιμές.Επίσης αξίζει να

σημειώσουμε πως η εκπαίδευση θα γίνεται για 80 εποχές με batch_size =8 και validation split = 0.3 τα οποία μας δίνουν τα καλύτερα αποτελέσματα πρόβλεψης για το νευρωνικό δίκτυο που κατασκευάσαμε σε σχέσει με άλλες δοκιμασμένες ρυθμίσεις όπως διαφορετικό batch size,patience,monitor variable.

Επίσης στην συνάρτηση doPredicts() ορίζουμε μια δομή λεξικού με όνομα PredDict όπου αποθηκεύουμε προβλέψεις για τα μοντέλα που χρησιμοποιήσαμε καθώς και τα ονόματα τους.

Στη συνέχεια για κάθε τέτοιο μοντέλο στο λεξικό αυτό θα τυπώσουμε ένα classification report το οποίο μας πληροφορεί σχετικά με τις μετρικές **precision(ακρίβεια),recall(ανάκληση),f1_score και του support** που πέτυχε το μοντέλο για την εκπαίδευση που εκτέλεσε. Επίσης τυπώνουμε και το AUC score που προκύπτει από την κατασκευή της καμπύλης ROC.Όσο πιο κοντά στο 1 είναι το f1_score και το AUC score τόσο καλύτερη θα είναι η πρόβλεψη του μοντέλου,επίσης όσο η ακρίβεια πλησιάζει το 100% τόσο πιο ακριβής θα θεωρείται η πρόβλεψη ωστόσο θα πρέπει να είμαστε προσεκτικοί να μην έχουμε overfitting του μοντέλου μας.

Θεωρούμε σημαντικό πριν προχωρήσουμε να δούμε κάποια πράγματα για τις καμπύλες **ROC, το AUC** score και τις έννοιες **TP(True Positive),TN(True Negative),FP(False Positive),FN(False Negative)**:

- **True Positive (TP) :** Ως "true positive" θεωρούμε την περίπτωση που το μοντέλο μηχανική μάθησης προβλέπει σωστά την "θετική" κλάση(δηλαδή ένας ασθενής που θα πρέπει να ταξινομηθεί ως καρδιοπαθής ταξινομείται σε αυτήν την κατηγορία).

- **True Negative (TN)** : Ως "true negative" θεωρούμε την περίπτωση που το μοντέλο μηχανικής μάθησης προβλέπει σωστά την "αρνητική" κλάση(δηλαδή ένας ασθενής που θα πρέπει να ταξινομηθεί ως μη καρδιοπαθής θα ταξινομηθεί σε αυτή την κατηγορία).

- **False Positive (FP) :** Ως "false positive" θεωρούμε την περίπτωση που το μοντέλο μηχανικής μάθησης προβλέπει λάθος την "θετική" κλάση (δηλαδή ένας ασθενής που δεν πρέπει να ταξινομηθεί ως καρδιοπαθής ταξινομείται σε αυτή την κατηγορία).

- **False Negative (FN) :** Ως "false negative" θεωρούμε την περίπτωση που το μοντέλο μηχανικής μάθησης προβλέπει λάθος την "αρνητική" κλάση(δηλαδή ένας ασθενής που θα πρέπει να ταξινομηθεί ως καρδιοπαθής ταξινομείται στην κατηγορία μη-καρδιοπαθής).

Η καμπύλη **ROC** (**Receiver Operator Characteristic**) είναι μια μετρική αξιολόγησης για δυαδικά προβλήματα ταξινόμησης. Συγκεκριμένα αποτελεί μια καμπύλη πιθανότητας αναπαριστά γραφικά **το TPR** (**True Positive Rate**,το οποίο είναι ο λόγος TP/(TP+FN) και εκφράζει την πιθανότητα ότι η πρόβλεψη της "θετικής" κλάσης είναι σωστή όπως αναφέραμε παραπάνω) σε σχέση με το **FPR**

(**False Positive Rate**, το οποίο είναι ο λόγος FP/(FP+TN) και εκφράζει την πιθανότητα να υπάρξει FP αποτέλεσμα από το μοντέλο μας, δηλαδή την πιθανότητα η πρόβλεψη της "θετικής" κλάσης να είναι λάθος) .Η περιοχή κάτω από την καμπύλη αυτή (**Area Under Curve - AUC**) αποτελεί μετρική της ικανότητας του ταξινομητή να "ξεχωρίζει" τις κλάσεις μεταξύ τους άρα και κατά πόσο μπορεί να πραγματοποιήσει σωστή πρόβλεψη κλάσης για αντίστοιχα στιγμιότυπα εισόδου. Όσο μεγαλύτερο είναι το **AUC** τόσο καλύτερη θεωρείται η ικανότητα του μοντέλου να προβλέπει σωστά την κλάση που αντιστοιχεί στο στιγμιότυπο εισόδου. Οι τιμές που παίρνει το AUC είναι μεταξύ 0 και 1,συνεπώς όσο πιο κοντά είναι στο 1 τόσο καλύτερη και αξιόπιστη η προβλεπτική ικανότητα του μοντέλου μας.

Επιπρόσθετα κάποιοι επιπρόσθετοι χρήσιμοι ορισμοί μετρικών που θα χρησιμοποιήσουμε:

- **Precision**: Ο λόγος των σωστών εκτιμήσεων για την κλάση προς το άθροισμα των σωστών και των λάθος εκτιμήσεων της κλάσης .
- **Recall**: Ο λόγος των σωστών εκτιμήσεων για την κλάση προς το άθροισμα των σωστών εκτιμήσεων της κλάσης και των λάθος εκτιμήσεων της άλλης κλάσης.
- **F1 Score**: Ο λόγος επί 2 του γινομένου recall και precision προς το άθροισμα του recall και του precision.
- **Support**: Πλήθος δειγμάτων της κάθε κλάσης στο διάνυσμα που χρησιμοποιούμε για την αξιολόγηση του μοντέλου.

Στη συνέχεια ορίζουμε την συνάρτηση ***model_sel()*** με ορίσματα το σύνολο X που περιέχει το σύνολο δεδομένων εκτός από τη στήλη της μεταβλητής στόχου(target) και το Y που είναι η στήλη της μεταβλητής στόχου(target).Επιπλέον ορίσματα είναι οι επιλογές smote και undersamp για τις οποίες αν τις θέσουμε κάποια από αυτές σε 1 θα γίνει η αντίστοιχη τεχνική sampling,δηλαδή για τιμή 1 στο smote θα κάνει υπερδειγματοληψία στο σύνολο δεδομένων προκειμένου να εξισορροπήσει πιθανή ανισορροπία κλάσεων και 1 στο undersamp θα εφαρμόσει τυχαία υποδειγματοληψία (RandomUnderSampler) στο σύνολο δεδομένων μας.Αν τεθούν οι παράμετροι αυτές στο 0 δεν εκτελείται κάποια μέθοδος υπερδειγματοληψίας ή υποδειγματοληψίας και επίσης δεν μπορούμε να επιτελέσουμε και τις δύο μεθόδους ταυτόχρονα στο σύνολο δεδομένων μας.Επειδή ωστόσο το σύνολο δεδομένων μας δεν παρουσιάζει κάποια σοβαρή ανισορροπία ως προς τις δύο κλάσεις που αποτελούν την μεταβλητή στόχο target,δηλαδή 1 για εμφάνιση καρδιακής νόσου και 0 για την μη εμφάνιση της,δεν επιλέγουμε να χρησιμοποιήσουμε κάποια από αυτές στον πειραματισμό μας αλλα θα τις αφήσουμε ως υλοποιημένη λειτουργικότητα στον κώδικα μας σε περίπτωση μελλοντικής χρήσης για ανισσοροπημένα σύνολα δεδομένων.

Ιδιαίτερο ενδιαφέρον έχουν τα 3 τελευταία ορίσματα της συνάρτησης ***model_sel(),***συγκεκριμένα το όρισμα **standard,normalz** και **preproc** όπου αν θέσουμε 1 στην standard επιτελείται standardization στο σύνολο δεδομένων μας, αν θέσουμε 1 στο normalz πραγματοποιείται normalization στο σύνολο δεδομένων μας και το 1 στο preproc επιτρέπει

σε τυχόν ενεργοποιημένες μεθόδους προεπεξεργασίας που περιλαμβάνουν τις προηγούμενες παραμέτρους που αναλύσαμε να επιδράσουν στο σύνολο δεδομένων μας αλλιώς δεν επιτελείται κάποια από αυτές ακόμα και εάν έχουν ενεργοποιηθεί(δηλαδή preproc=0).

Αξίζει να επισημάνουμε κάποιες έννοιες:

Standardization: Επιτελούμε standardization στα χαρακτηριστικά του συνόλου ως εξής:

Η standardized τιμή του δείγματος σημείου x που ανήκει στο σύνολο δεδομένων μας υπολογίζεται από : z = (x- u) / s ,όπου u είναι η μέση τιμή των δειγμάτων εκπαίδευσης και s είναι η κανονική απόκλιση (standard deviation) των δειγμάτων εκπαίδευσης.

Normalization: Είναι μια διαδικασία μετασχηματισμού δεδομένων κατά την οποία αριθμητικές τιμές χαρακτηριστικών αντικαθίστανται από άλλες αντίστοιχες τιμές που βρίσκονται σε ένα συγκεκριμένο εύρος τιμών.

Θα δούμε στην συνέχεια την απόδοση των μοντέλων μας:

RUN 01:

Καλούμε την συνάρτηση model_sel() ως εξής:

```
In [52]: model_sel(X,Y,0,0,1,1,1)
```

Figure 19:Calling model_sel-preprocessing mode

Όπως παρατηρούμε από τις τιμές των παραμέτρων επιτρέπουμε preprocessing των δεδομένων μας πριν την εκπαίδευση από κάποιο μοντέλο και συγκεκριμένα εφαρμόζουμε και τις δύο μεθόδους προεπεξεργασίας που αναλύσαμε προηγουμένως δηλαδή **standardization** και **normalization** και έχουμε τα εξής αποτελέσματα:

```
Model: "sequential_14"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
dense_42 (Dense)             (None, 32)                928
_________________________________________________________________
dense_43 (Dense)             (None, 28)                924
_________________________________________________________________
dense_44 (Dense)             (None, 1)                 29
=================================================================
Total params: 1,881
Trainable params: 1,881
Non-trainable params: 0
```

Figure 20:Neural Network architecture, Run 01

```
Epoch 23/80
26/26 [==============================] - 0s 7ms/step - loss: 0.3031 - accuracy: 0.8732 - f1_metric: 0.9121 - precision_metri
c: 0.8786 - recall_metric: 0.9628 - val_loss: 2.4081 - val_accuracy: 0.1011 - val_f1_metric: 0.0000e+00 - val_precision_metri
c: 0.0000e+00 - val_recall_metric: 0.0000e+00
```

Figure 21:Neural Network Perfomance-final epoch-Run 01

```
Logistic RegressionAccuracy: 87.6404
 Full report for Logistic Regression
              precision    recall  f1-score   support

           0       0.83      0.86      0.85        35
           1       0.91      0.89      0.90        54

    accuracy                           0.88        89
   macro avg       0.87      0.87      0.87        89
weighted avg       0.88      0.88      0.88        89
```

Figure 22:Logistic Regression Report-Run 01

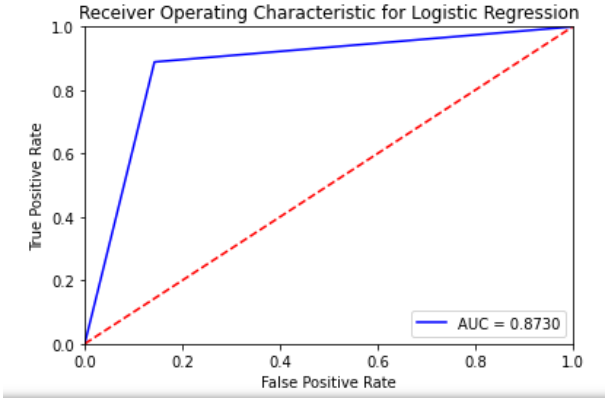


Figure 23:Logistic Regression ROC - Run 01

```
SVMAccuracy: 87.6404
 Full report for SVM
              precision    recall  f1-score   support

           0       0.83      0.86      0.85        35
           1       0.91      0.89      0.90        54

    accuracy                           0.88        89
   macro avg       0.87      0.87      0.87        89
weighted avg       0.88      0.88      0.88        89
```

Figure 24:SVM report - Run 01

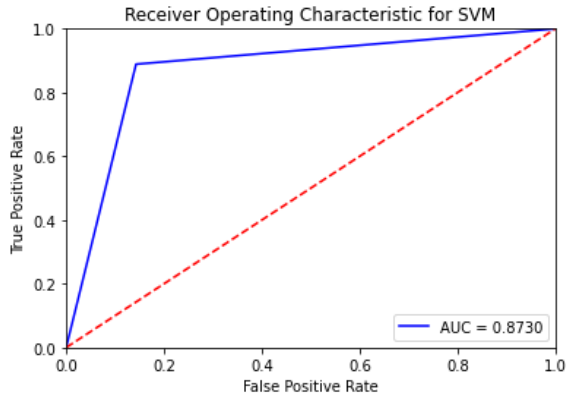


Figure 25:SVM ROC - Run 01

```
Naive BayesAccuracy: 86.5169
 Full report for Naive Bayes
              precision    recall  f1-score   support

           0       0.79      0.89      0.84        35
           1       0.92      0.85      0.88        54

    accuracy                           0.87        89
   macro avg       0.86      0.87      0.86        89
weighted avg       0.87      0.87      0.87        89
```

Figure 26:Naive Bayes Report - Run 01

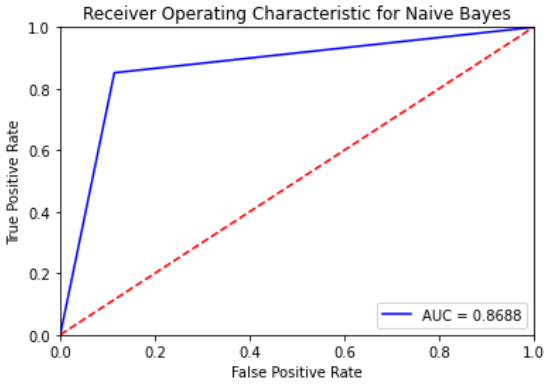


Figure 27:Naive Bayes ROC - Run 01

```
Decision TreeAccuracy: 82.0225
 Full report for Decision Tree
              precision    recall  f1-score   support

           0       0.77      0.77      0.77        35
           1       0.85      0.85      0.85        54

    accuracy                           0.82        89
   macro avg       0.81      0.81      0.81        89
weighted avg       0.82      0.82      0.82        89
```

Figure 28:Decision Tree - Run 01

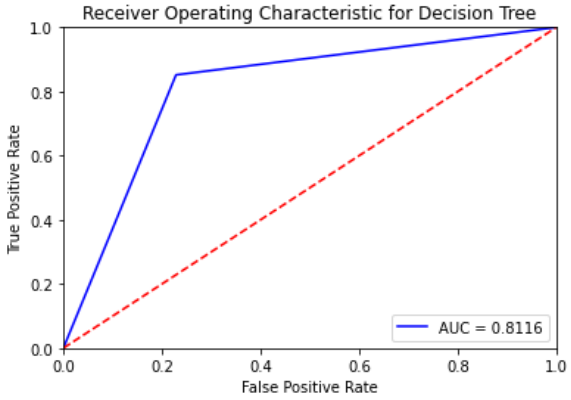


Figure 29:Decision Tree ROC - Run 01

K-nearest neighboorsAccuracy: 82.0225
Full report for K-nearest neighboors

	precision	recall	f1-score	support
0	0.74	0.83	0.78	35
1	0.88	0.81	0.85	54
accuracy			0.82	89
macro avg	0.81	0.82	0.81	89
weighted avg	0.83	0.82	0.82	89

Figure 30:KNN report - Run 01

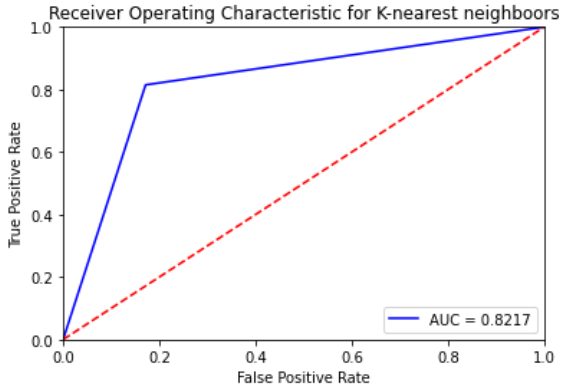


Figure 31: KNN ROC - Run 01

Random ForestAccuracy: 85.3933
Full report for Random Forest

	precision	recall	f1-score	support
0	0.81	0.83	0.82	35
1	0.89	0.87	0.88	54
accuracy			0.85	89
macro avg	0.85	0.85	0.85	89
weighted avg	0.85	0.85	0.85	89

Figure 32:Random Forest report - Run 01

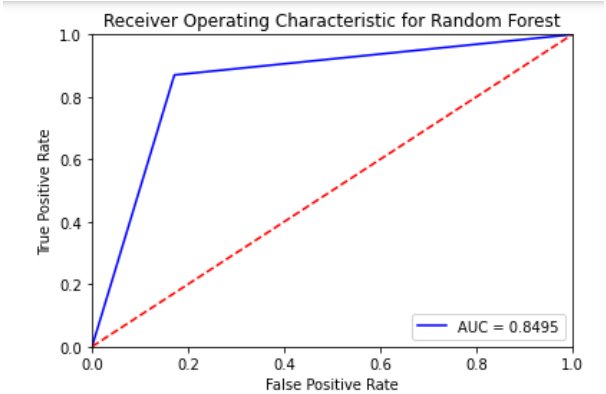


Figure 33: Random Forest ROC - Run 01

```
XgboostAccuracy: 82.0225
 Full report for Xgboost
              precision    recall  f1-score   support

           0       0.76      0.80      0.78        35
           1       0.87      0.83      0.85        54

    accuracy                           0.82        89
   macro avg       0.81      0.82      0.81        89
weighted avg       0.82      0.82      0.82        89
```

Figure 34:XGBoost report - Run 01

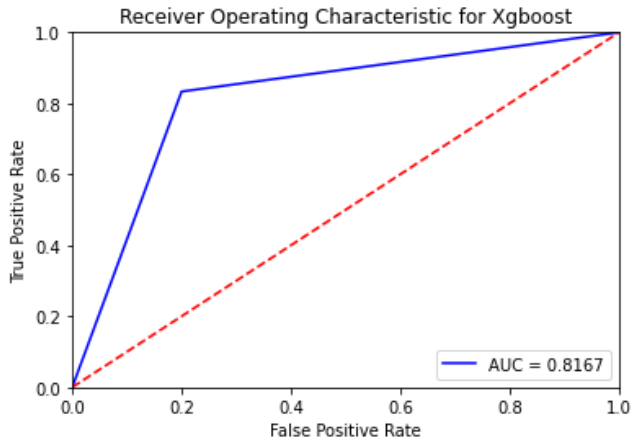


Figure 35:XGBoost ROC - Run 01

RUN 02 :

Καλούμε την συνάρτηση model_sel() ως εξής:

```
In [31]: model_sel(X,Y,0,0,1,0,1)
```

Figure 36:Model Selection - Run 02

Στην συγκεκριμένη προσομοίωση δηλαδή δεν κάνουμε normalization στα δεδομένα μας αλλά εφαρμόζουμε μόνο standardization και αποκτούμε τα παρακάτω αποτελέσματα:

```
Epoch 45/80
26/26 [==============================] - 0s 6ms/step - loss: 0.2445 - accuracy: 0.8878 - f1_metric: 0.9138 - precision_metri
c: 0.9135 - recall_metric: 0.9430 - val_loss: 0.9521 - val_accuracy: 0.5506 - val_f1_metric: 0.0000e+00 - val_precision_metri
c: 0.0000e+00 - val_recall_metric: 0.0000e+00
```

Figure 37:Neural Network-Last Epoch- Run 02

```
Logistic RegressionAccuracy: 86.5169
 Full report for Logistic Regression
              precision    recall  f1-score   support

           0       0.81      0.86      0.83        35
           1       0.90      0.87      0.89        54

    accuracy                           0.87        89
   macro avg       0.86      0.86      0.86        89
weighted avg       0.87      0.87      0.87        89
```

Figure 38:Logistic Regression-Report-Run 02

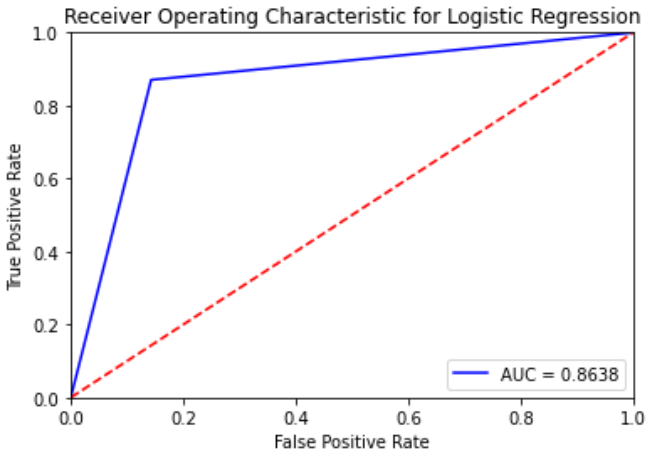


Figure 39:Logistic Regression - ROC - Run 02

SVMAccuracy: 83.1461
Full report for SVM

	precision	recall	f1-score	support
0	0.78	0.80	0.79	35
1	0.87	0.85	0.86	54
accuracy			0.83	89
macro avg	0.82	0.83	0.82	89
weighted avg	0.83	0.83	0.83	89

Figure 40:SVM report - Run 02

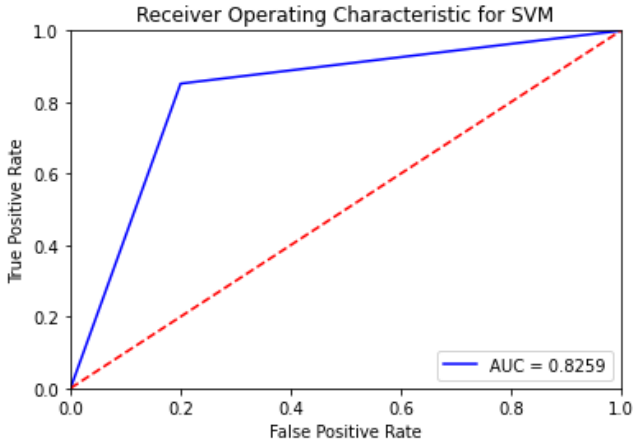


Figure 41:SVM ROC - Run 02

```
Naive BayesAccuracy: 87.6404
 Full report for Naive Bayes
              precision    recall  f1-score   support

           0       0.82      0.89      0.85        35
           1       0.92      0.87      0.90        54

    accuracy                           0.88        89
   macro avg       0.87      0.88      0.87        89
weighted avg       0.88      0.88      0.88        89
```

Figure 42:Naive Bayes report - Run 02

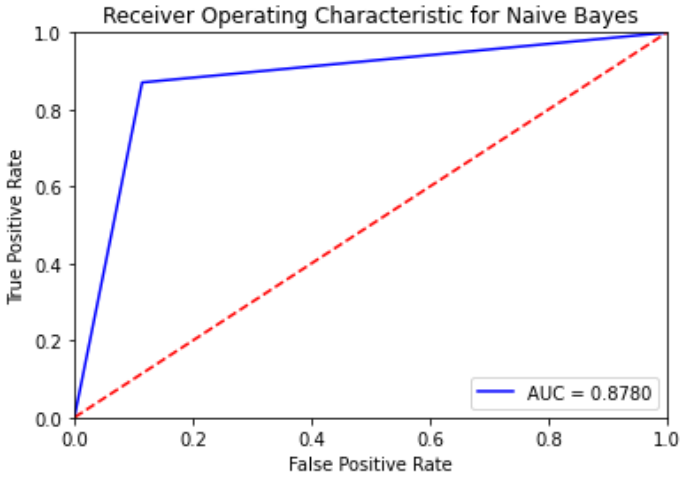


Figure 43:Naive Bayes ROC - Run 02

```
Decision TreeAccuracy: 77.5281
 Full report for Decision Tree
              precision    recall  f1-score   support

           0       0.70      0.74      0.72        35
           1       0.83      0.80      0.81        54

    accuracy                           0.78        89
   macro avg       0.76      0.77      0.77        89
weighted avg       0.78      0.78      0.78        89
```

Figure 44:Decision Tree report - Run 02

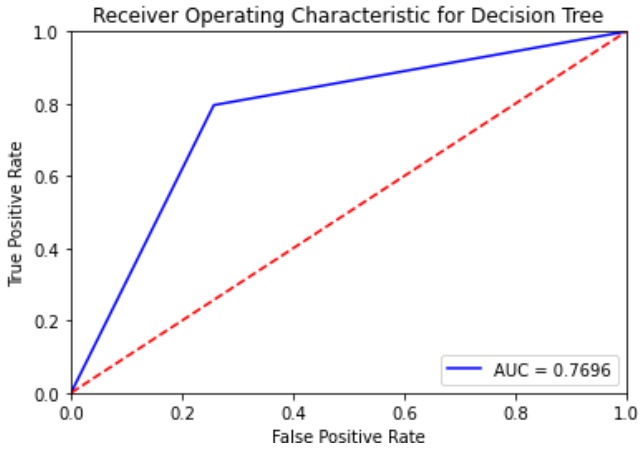


Figure 45:Decision Tree ROC - Run 02

K-nearest neighboorsAccuracy: 89.8876
Full report for K-nearest neighboors

	precision	recall	f1-score	support
0	0.82	0.94	0.88	35
1	0.96	0.87	0.91	54
accuracy			0.90	89
macro avg	0.89	0.91	0.90	89
weighted avg	0.91	0.90	0.90	89

Figure 46:KNN report - Run 02

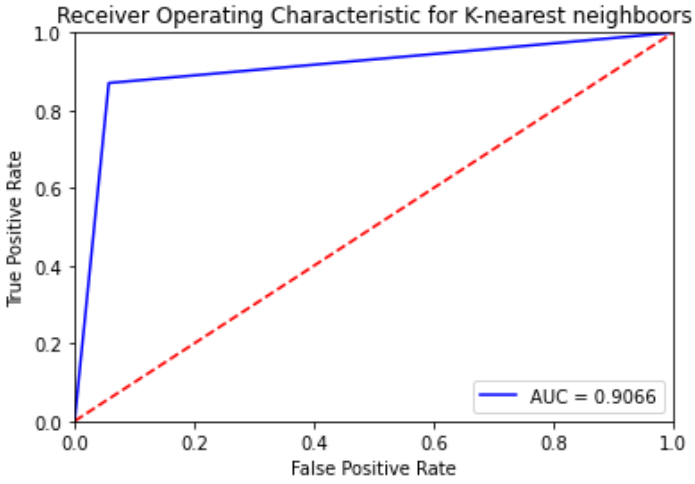


Figure 47:KNN ROC - Run 02

Random ForestAccuracy: 88.7640
Full report for Random Forest

	precision	recall	f1-score	support
0	0.82	0.91	0.86	35
1	0.94	0.87	0.90	54
accuracy			0.89	89
macro avg	0.88	0.89	0.88	89
weighted avg	0.89	0.89	0.89	89

Figure 48:Random Forest report - Run 02

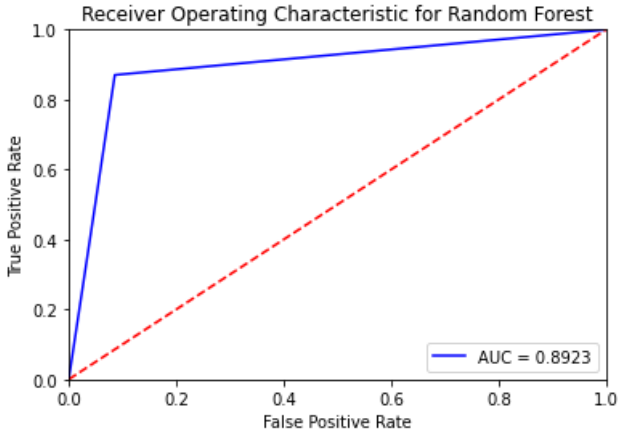


Figure 49:Random Forest ROC - Run 02

```
XgboostAccuracy: 80.8989
 Full report for Xgboost
              precision    recall  f1-score   support

           0       0.74      0.80      0.77        35
           1       0.86      0.81      0.84        54

    accuracy                           0.81        89
   macro avg       0.80      0.81      0.80        89
weighted avg       0.81      0.81      0.81        89
```

Figure 50:XGBoost report - Run 02

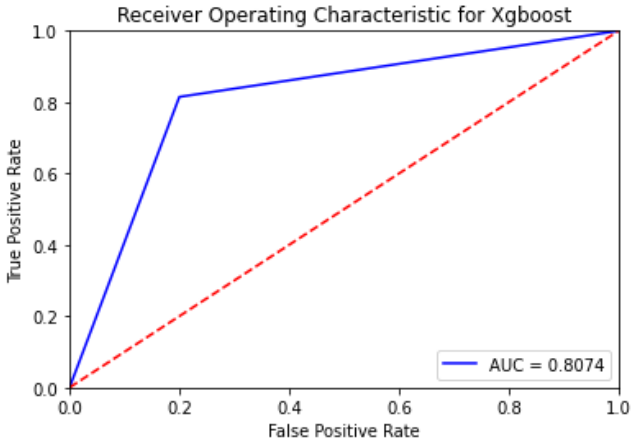


Figure 51:XGBoost ROC - Run 02

Run 03 :

Καλούμε την συνάρτηση model_sel() ως εξής:

```
In [32]: model_sel(X,Y,0,0,0,1,1)
```

Figure 52:Model_sel -Run 03

Δηλαδή σε αυτήν την προσομοίωση θα εφαρμόσουμε normalization στα δεδομένα μας αλλά όχι standardization και παίρνουμε τα παρακάτω αποτελέσματα:

```
Epoch 26/80
26/26 [==============================] - 0s 5ms/step - loss: 0.3345 - accuracy: 0.8683 - f1_metric: 0.9115 - precision_metri
c: 0.8947 - recall_metric: 0.9551 - val_loss: 0.8394 - val_accuracy: 0.5281 - val_f1_metric: 0.0000e+00 - val_precision_metri
c: 0.0000e+00 - val_recall_metric: 0.0000e+00
Logistic RegressionAccuracy: 87.6404
```

Figure 4:Neural Network -Last Epoch - Run 03

```
Logistic RegressionAccuracy: 87.6404
 Full report for Logistic Regression
              precision    recall  f1-score   support

           0       0.83      0.86      0.85        35
           1       0.91      0.89      0.90        54

    accuracy                           0.88        89
   macro avg       0.87      0.87      0.87        89
weighted avg       0.88      0.88      0.88        89
```

Figure 53:Logistic Regression report -Run 03

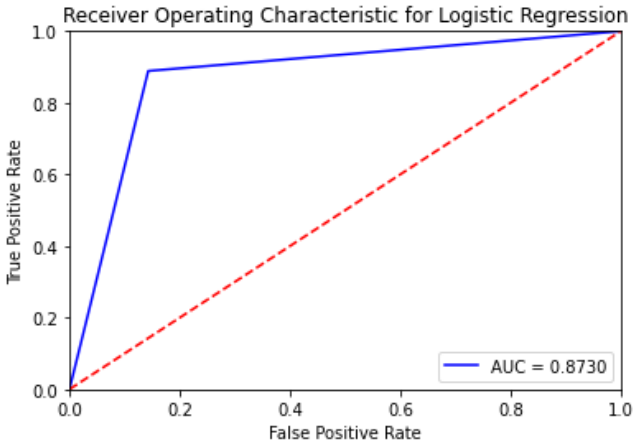


Figure 54:Logistic Regression ROC - Run 03

```
SVMAccuracy: 87.6404
 Full report for SVM
              precision    recall  f1-score   support

           0       0.83      0.86      0.85        35
           1       0.91      0.89      0.90        54

    accuracy                           0.88        89
   macro avg       0.87      0.87      0.87        89
weighted avg       0.88      0.88      0.88        89
```

Figure 55:SVM report - Run 03

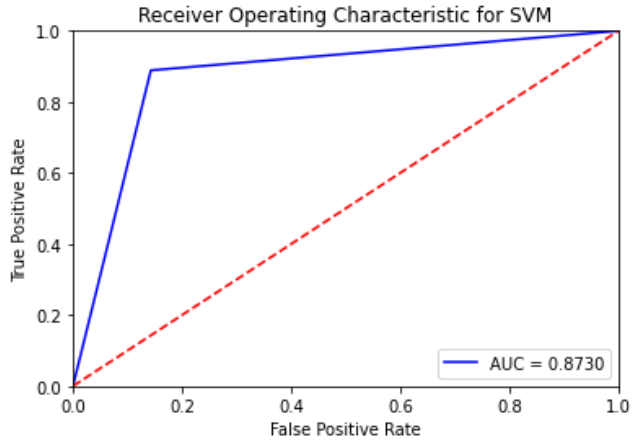


Figure 56:SVM ROC - Run 03

Naive BayesAccuracy: 86.5169
Full report for Naive Bayes

	precision	recall	f1-score	support
0	0.79	0.89	0.84	35
1	0.92	0.85	0.88	54
accuracy			0.87	89
macro avg	0.86	0.87	0.86	89
weighted avg	0.87	0.87	0.87	89

Figure 57:Naive Bayes report - Run 03

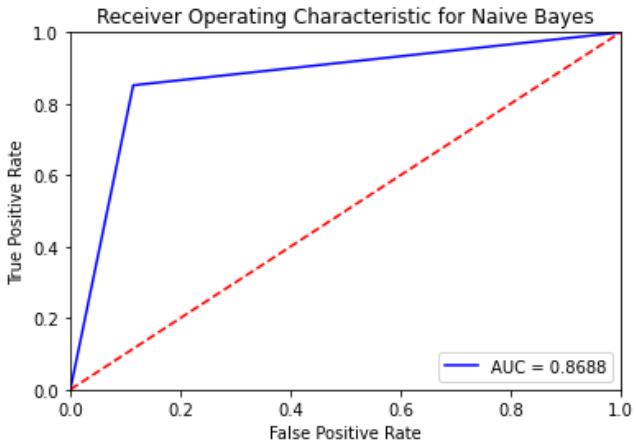


Figure 58:Naive Bayes ROC - Run 03

Decision TreeAccuracy: 82.0225
Full report for Decision Tree

	precision	recall	f1-score	support
0	0.77	0.77	0.77	35
1	0.85	0.85	0.85	54
accuracy			0.82	89
macro avg	0.81	0.81	0.81	89
weighted avg	0.82	0.82	0.82	89

Figure 59:Decision Tree report - Run 03

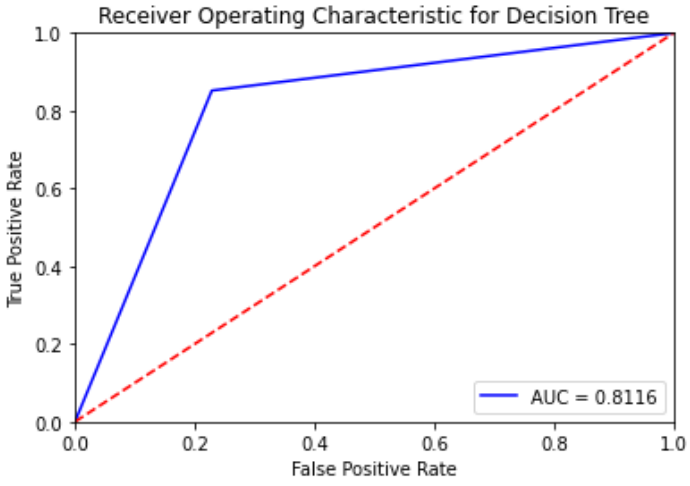


Figure 60:Decision Tree ROC - Run 03

K-nearest neighboorsAccuracy: 82.0225
Full report for K-nearest neighboors

	precision	recall	f1-score	support
0	0.74	0.83	0.78	35
1	0.88	0.81	0.85	54
accuracy			0.82	89
macro avg	0.81	0.82	0.81	89
weighted avg	0.83	0.82	0.82	89

Figure 61:KNN report - Run 03

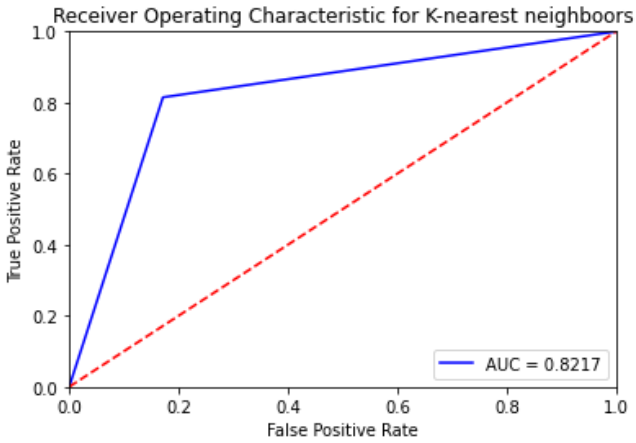


Figure 62:KNN ROC - Run 03

Random ForestAccuracy: 85.3933
Full report for Random Forest

	precision	recall	f1-score	support
0	0.81	0.83	0.82	35
1	0.89	0.87	0.88	54
accuracy			0.85	89
macro avg	0.85	0.85	0.85	89
weighted avg	0.85	0.85	0.85	89

Figure 63:Random Forest - Run 03

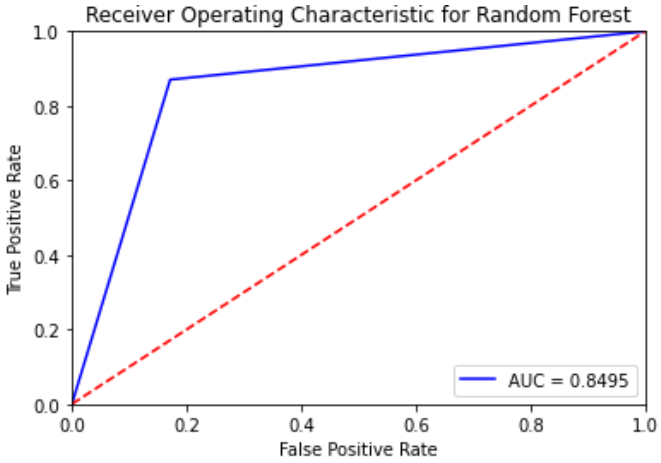


Figure 64:Random Forest ROC - Run 03

```
XgboostAccuracy: 82.0225
 Full report for Xgboost
              precision    recall  f1-score   support

           0       0.76      0.80      0.78        35
           1       0.87      0.83      0.85        54

    accuracy                           0.82        89
   macro avg       0.81      0.82      0.81        89
weighted avg       0.82      0.82      0.82        89
```

Figure 65:XGBoost report -Run 03

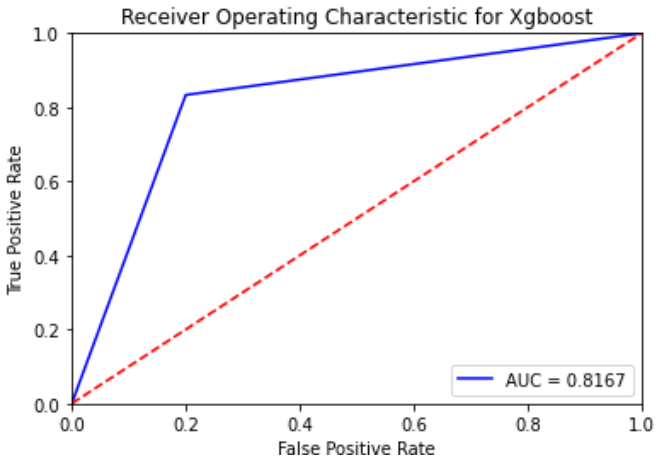


Figure 66:XGBoost ROC - Run 03

Run 04:

Καλούμε την συνάρτηση model_sel() ως εξής:

```
In [33]: model_sel(X,Y,0,0,0,0,0)
```

Figure 67:Model_sel-Run 04

Όπως παρατηρούμε και από τις τιμές των παραμέτρων σε αυτή την προσομοίωση δεν θα εφαρμόσουμε standardization ή normalization και έχουμε τα παρακάτω αποτελέσματα:

```
Epoch 36/80
26/26 [==============================] - 0s 6ms/step - loss: 0.2799 - accuracy: 0.8780 - f1_metric: 0.9186 - precision_metri
c: 0.9035 - recall_metric: 0.9521 - val_loss: 2.1098 - val_accuracy: 0.1573 - val_f1_metric: 0.0000e+00 - val_precision_metri
c: 0.0000e+00 - val_recall_metric: 0.0000e+00
```

Figure 68:Neural Network – Last Epoch – Run 04

Logistic RegressionAccuracy: 89.8876
Full report for Logistic Regression

	precision	recall	f1-score	support
0	0.86	0.89	0.87	35
1	0.92	0.91	0.92	54
accuracy			0.90	89
macro avg	0.89	0.90	0.89	89
weighted avg	0.90	0.90	0.90	89

Figure 69:Logistic Regression report - Run 04

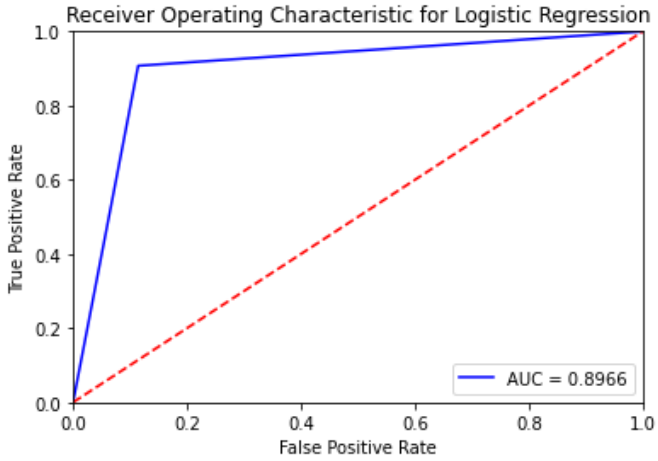


Figure 70:Logistic Regression ROC – Run 04

SVMAccuracy: 68.5393
Full report for SVM

	precision	recall	f1-score	support
0	0.65	0.43	0.52	35
1	0.70	0.85	0.77	54
accuracy			0.69	89
macro avg	0.67	0.64	0.64	89
weighted avg	0.68	0.69	0.67	89

Figure 71:SVM report – Run 04

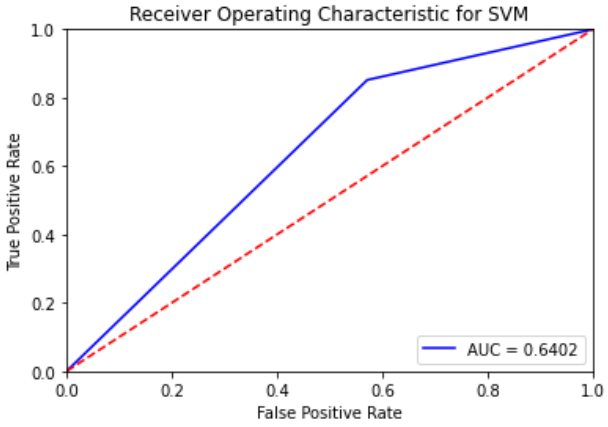


Figure 72:SVM ROC – Run 04

```
Naive BayesAccuracy: 87.6404
 Full report for Naive Bayes
              precision    recall  f1-score   support

           0       0.82      0.89      0.85        35
           1       0.92      0.87      0.90        54

    accuracy                           0.88        89
   macro avg       0.87      0.88      0.87        89
weighted avg       0.88      0.88      0.88        89
```

Figure 73:Naïve Bayes report – Run 04

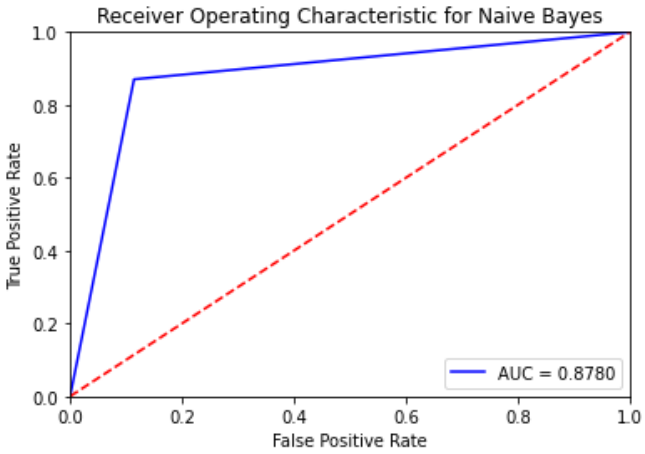


Figure 74:Naïve Bayes ROC – Run 04

```
Decision TreeAccuracy: 77.5281
 Full report for Decision Tree
              precision    recall  f1-score   support

           0       0.70      0.74      0.72        35
           1       0.83      0.80      0.81        54

    accuracy                           0.78        89
   macro avg       0.76      0.77      0.77        89
weighted avg       0.78      0.78      0.78        89
```

Figure 75:Decision Tree report – Run 04

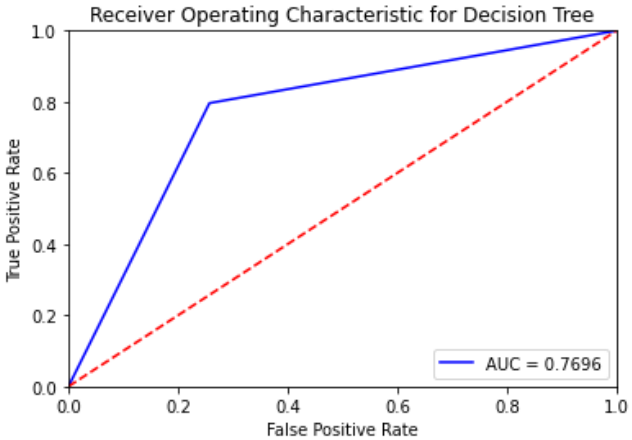


Figure 76:Decision Tree ROC – Run 04

K-nearest neighboorsAccuracy: 73.0337
Full report for K-nearest neighboors

	precision	recall	f1-score	support
0	0.70	0.54	0.61	35
1	0.74	0.85	0.79	54
accuracy			0.73	89
macro avg	0.72	0.70	0.70	89
weighted avg	0.73	0.73	0.72	89

Figure 77:KNN report – Run 04

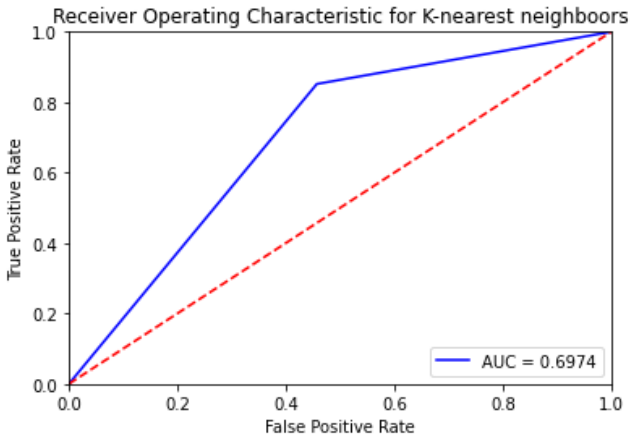


Figure 78:KNN ROC – Run 04

Random ForestAccuracy: 88.7640
Full report for Random Forest

	precision	recall	f1-score	support
0	0.82	0.91	0.86	35
1	0.94	0.87	0.90	54
accuracy			0.89	89
macro avg	0.88	0.89	0.88	89
weighted avg	0.89	0.89	0.89	89

Figure 79:Random Forest report – Run 04

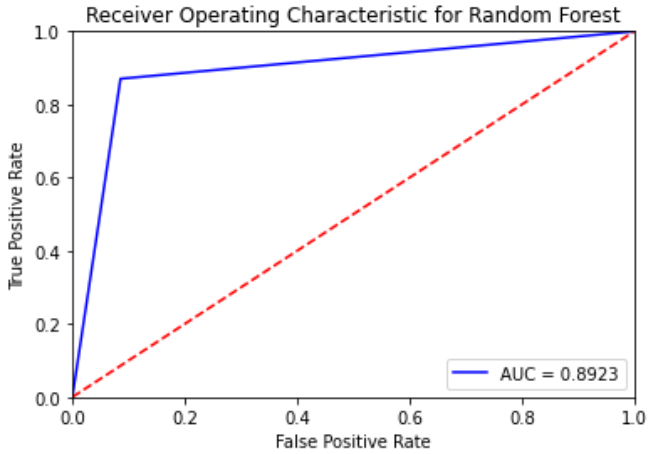


Figure 80: Random Forest ROC – Run 04

```
XgboostAccuracy: 80.8989
 Full report for Xgboost
              precision    recall  f1-score   support

           0       0.74      0.80      0.77        35
           1       0.86      0.81      0.84        54

    accuracy                           0.81        89
   macro avg       0.80      0.81      0.80        89
weighted avg       0.81      0.81      0.81        89
```

Figure 81:XGBoost report – Run 04

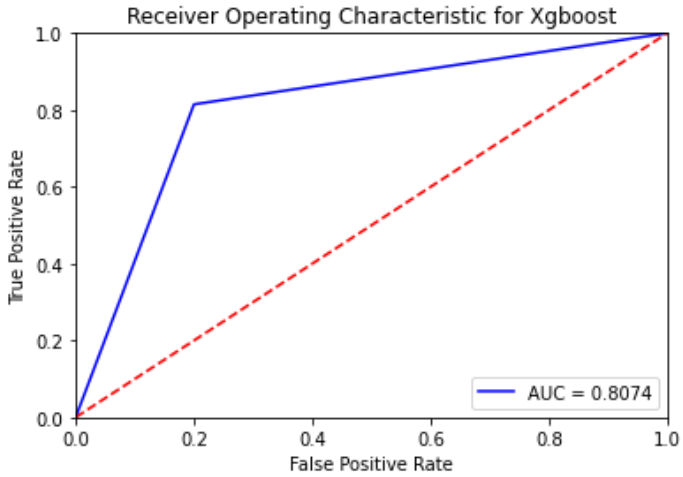


Figure 82:XGBoost ROC – Run 04

Σε αυτό το σημείο θα δούμε κάποια συμπεράσματα σχετικά με τις προσομοιώσεις που πραγματοποιήσαμε και την απόδοση των μοντέλων μηχανικής μάθησης που χρησιμοποιήσαμε:

(Εδώ σημειώνουμε πως πολλαπλασιάσαμε το AUC επι 100 ώστε η σύγκριση των μοντέλων να γίνει πιο εύκολη στη παρατήρηση, σε αυτήν την περίπτωση θεωρούμε πως το μέγιστο είναι το 100 το οποίο αναπαριστά το 1 που θα ήταν και το άνω όριο κλασσικά για το AUC).

Στην προσομοίωση "**Run 01**" όπου εφαρμόσαμε στα δεδομένα εισόδου πριν ταξινομηθούν από κάποιο μοντέλο μηχανικής μάθησης,**προεπεξεργασία με τις τεχνικές standardization και normalization** παρατηρούμε τα εξής:

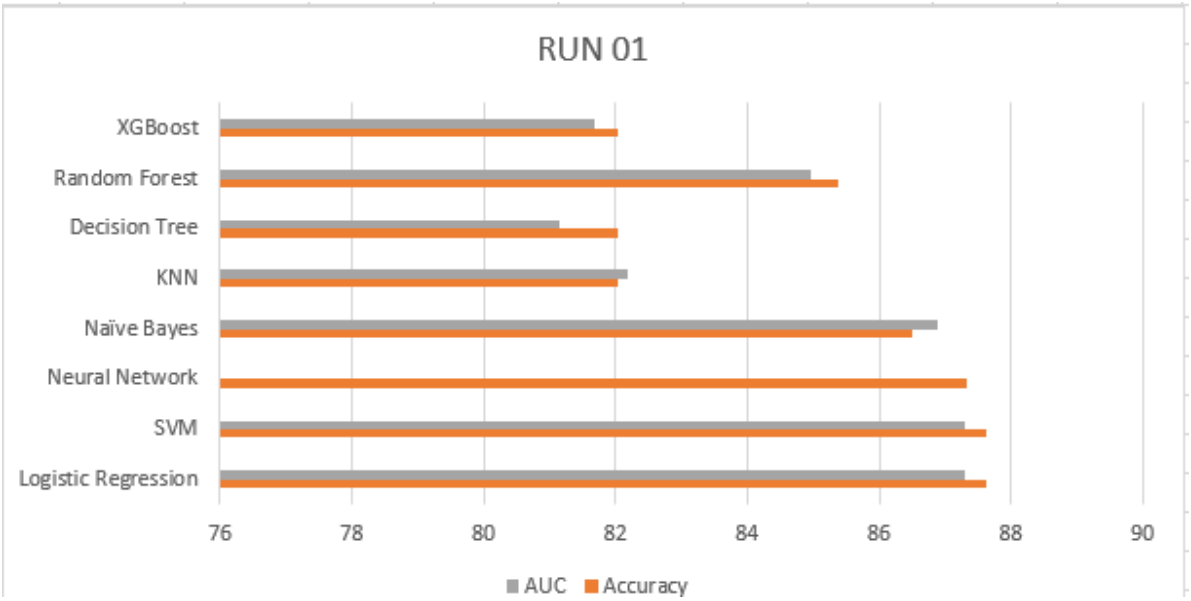


Figure 83:Run 01 - Models Perfomance Comparison

Παρατηρούμε πως τα μοντέλα μηχανικής μάθησης που επιτυγχάνουν την καλύτερη απόδοση πρόβλεψης για το σύνολο δεδομένων μας όταν έχουμε εφαρμόσει standardization και normalization στα δεδομένα μας πριν την εκπαίδευση είναι το "Logistic Regression","SVM",το νευρωνικό δίκτυο που σχεδιάσαμε καθώς και ο ταξινομητής Naïve Bayes τα οποία έχουν accuracy και AUC score πάνω από 86 στα 100.Τα υπόλοιπα μοντέλα παρατηρούμε πως δεν έχουν εξίσου καλή απόδοση για την συγκεκριμένη προεπεξεργασία και παρατηρούμε πως το Decision Tree έχει την χειρότερη απόδοση σε αυτή την περίπτωση από όλα τα υπόλοιπα μοντέλα.

Στην προσομοίωση "**Run 02**" κατά την οποία **εφαρμόσαμε standardization αλλά όχι normalization** παρατηρούμε τα εξής:

Τα μοντέλα μηχανικής μάθησης που έχουν AUC score και accuracy πάνω από 85 στα 100 είναι ο ταξινομητής K-nearest neighbors , Random Forest , το νευρωνικό δίκτυο που κατασκευάσαμε, Naïve Bayes και Logistic Regression.Ανάμεσα τους ο ταξινομητής K-nearest neighbors επιτυγχάνει το καλύτερο accuracy και AUC score με το accuracy να είναι πολύ κοντά στο 90% ενώ το AUC score να έχει ξεπεράσει το ίδιο κατώφλι.Παρατηρούμε πως πάλι την χειρότερη απόδοση ως προς το accuracy και AUC score έχει το Decision Tree στην συγκεκριμένη προσομοίωση όπου το AUC score του και accuracy είναι κάτω από 80%.

Στο παρακάτω γράφημα ράβδων των μετρικών AUC και accuracy γίνονται εμφανή όσα περιφράφηκαν για τα συγκεκριμένα μοντέλα:

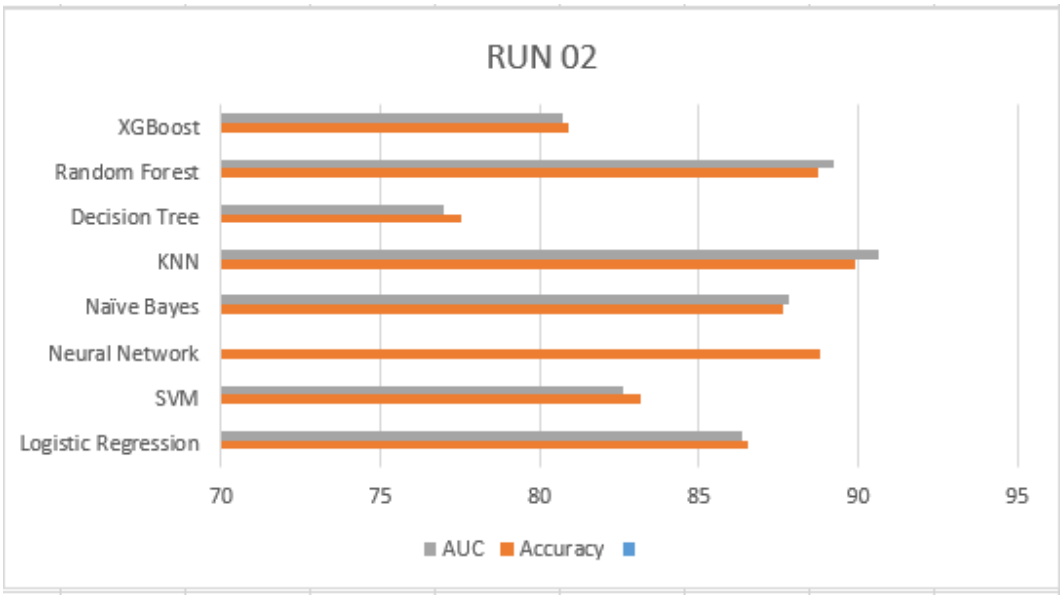


Figure 84:Run 02 - Metrics Perfomance Comparison

Στην προσομοίωση **"Run 03" εφαρμόσαμε normalization αλλά όχι standardization** στα δεδομένα εισόδου μας πριν την εκπαίδευση και έχουμε τα εξής αποτελέσματα:

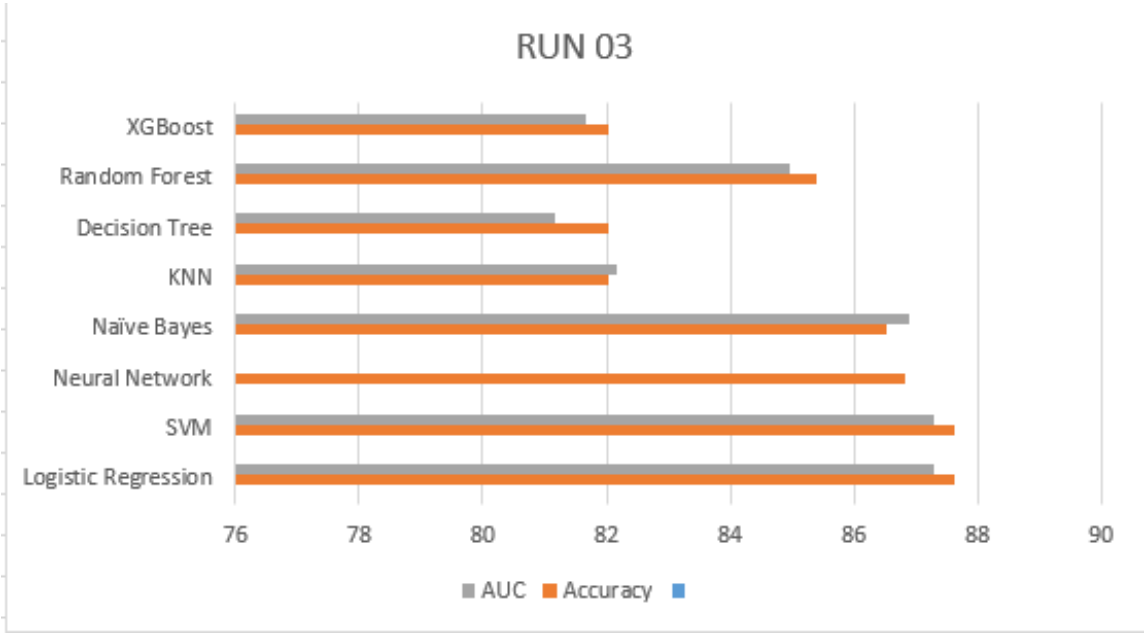


Figure 85:Run 03 - Metrics Perfomance Comparison

Παρατηρούμε ότι την καλύτερη απόδοση έχουν τα μοντέλα Logistic Regression,SVM,Neural Network και Naïve Bayes με AUC score και accuracy πάνω από 86% με τους Logistic Regression και SVM ταξινομητές να έχουν AUC και accuracy ιδιαίτερα κοντά στο 88% και οι οποίοι αποτελούν τους ταξινομητές με την καλύτερη απόδοση στην συγκεκριμένη προσομοίωση.Οι ταξινομητές K-nearest neighbors και Decision Tree παρατηρούμε πως έχουν

την χειρότερη απόδοση σε αυτή την προσομοίωση με το AUC score και accuracy τους να είναι κατω ή οριακά πάνω από το κατώφλι του 82%.

Στην τελευταία προσομοίωση που εκτελέσαμε, **"Run 04"** δεν εφαρμόσαμε ούτε standardization ούτε normalization στα δεδομένα εισόδου πριν διεξαχθεί η σχετική εκπαίδευση και καταλήγουμε στις εξής μετρικές:

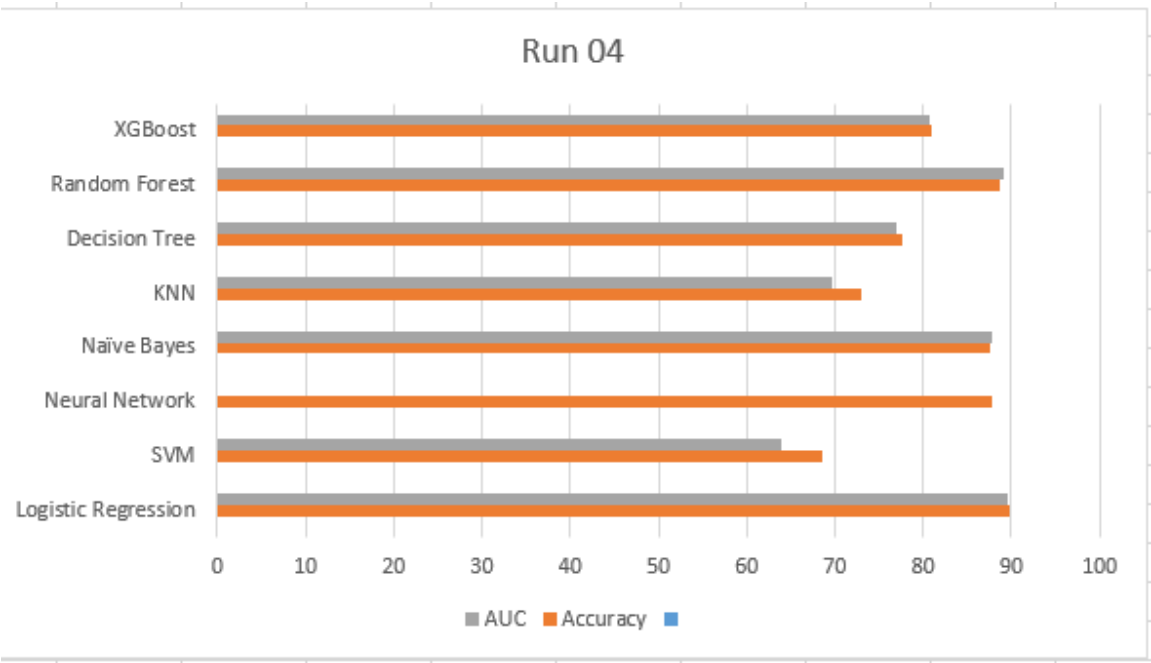


Figure 86:Run 04 - Metrics Perfomance Comparison

Εδώ παρατηρούμε πως την καλύτερη απόδοση από όλα τα υπόλοιπα μοντέλα ως προς το AUC score και accuracy έχει ο ταξινομητής Logistic Regression που σχεδόν αγγίζει το κατώφλι του 90% και για τις δύο μετρικές.Τα μοντέλα Random Forest,Naïve Bayes,Neural Network ακολουθούν το Logistic Regression σε απόδοση και έχουν τις σχετικές μετρικές πάνω από το κατώφλι του 80%.Για την συγκεκριμένη προσομοίωση παρατηρούμε πως την χειρότερη απόδοση δίνει ο SVM ταξινομητής με AUC score και accuracy κάτω από 70%.

Συμπερασματικά συμπεραίνουμε πως η έλλειψη εφαρμογής των τεχνικών normalization και standardization στα δεδομένα εισόδου πριν την εκπαίδευση έχει σημαντική επίπτωση στην απόδοση του SVM ταξινομητή ,ενώ αυτή η στρατηγική δίνει την καλύτερη απόδοση που παρατηρήσαμε κατά τον συνολικό πειραματισμό μας στο μοντέλο του Logistic Regression.Αξίζει ακόμα να επισημάνουμε ότι η εφαρμογή normalization αλλά όχι standardization τεχνικής μειώνει σημαντικά την απόδοση του KNN ταξινομητή.Απο την άλλη η εφαρμογή standardization χωρίς normalization μειώνει την απόδοση των ταξινομητών SVM,Logistic Regression και Decision Tree για ένα σημαντικό ποσοστό ενώ αυξάνει την απόδοση του KNN ταξινομητή που αγγίζει σχεδόν το 90% ως προς τις μετρικές accuracy και AUC score για τη συγκεκριμένη περίπτωση.Τέλος η εφαρμογή και των δύο τεχνικών προεπεξεργασίας στα δεδομένα μας,δηλαδή του standardization και normalization μας δίνει πολύ καλή απόδοση για τους ταξινομητές SVM και Logistic Regression.Η απόδοση του νευρωνικού δικτύου που κατασκεύσαμε παρατηρούμε πως δεν αλλάζει σημαντικά στις 4 διαφορετικές προσομοιώσεις που εκτελέσαμε αλλα καταγράψαμε ως την καλύτερη

απόδοση του, αυτή που είχε στην προσομοίωση **02** όπου εφαρμόστηκε standardization και όχι normalization.Τέλος αξίζει να σημειωθεί πως το νευρωνικό δίκτυο που κατασκεύσαμε κατάφερε να κρατήσει συγκρίσιμη καλή απόδοση(πάνω από 86% accuracy και f1_score πάνω από 91%) για όλες τις 4 προσομοιώσεις με τις μετρικές του να μην διαφέρουν σημαντικά μεταξύ τους από προσομοίωση σε προσομοίωση.