ΙΣΜΙΡΟΓΛΟΥ ΒΑΣΙΛΙΚΗ
GE15017
VISMIROGLOU@GMAIL.COM

ΧΡΙΣΤΟΦΙΛΟΠΟΥΛΟΣ ΑΝΔΡΕΑΣ
GE18044
ANDREASCHRISTOPHILOPOULOS@GMAIL.COM

Αναγνώριση Προτύπων και Νευρωνικά Δίκτυα

Τελική Εργασία

Σχολή Εφαρμοσμένων Μαθηματικών και Φυσικών Επιστημών 2021-22

Η συγκεκριμένη εργασία έχει ως στόχο την ανάλυση δεδομένων λευκού οίνου και τον προσδιορισμό των χαρακτηριστικών που τον καθορίζουν ποιοτικό. Στη διάθεσή μας έχουμε ένα σύνολο δεδομένων 4898 κρασιών, με 11 χαρακτηριστικά το κάθε ένα καθώς και μια τελική βαθμολογία «ποιότητας».

1] Επιλέξτε το κατώφλι βαθμολογίας που διαχωρίζει τις δύο κλάσεις ώστε να υπάρχει περίπου ίσος αριθμός γεγονότων (κρασιών) σε κάθε κλάση.

Αυτό επιτυγχάνεται βρίσκοντας αρχικά το median της ποιότητας, το οποίο προκύπτει πως είναι 6. Στη συνέχεια υπολογίζουμε το πλήθος των γεγονότων με ποιότητα χαμηλότερη ή υψηλότερη αυτής της τιμής και προσθέτουμε στο μικρότερο σύνολο τα γεγονότα με ποιότητα ακριβώς 6. Με τον τρόπο αυτό έχουμε την καλύτερη δυνατή κατανομή των γεγονότων σε δύο κλάσεις. Τελικά έχουμε δύο κλάσεις, ορισμένες ως «καλής ποιότητας» και «κακής ποιότητας» οίνος, με τα αντίστοιχα κατώφλια <6 και ≥6.

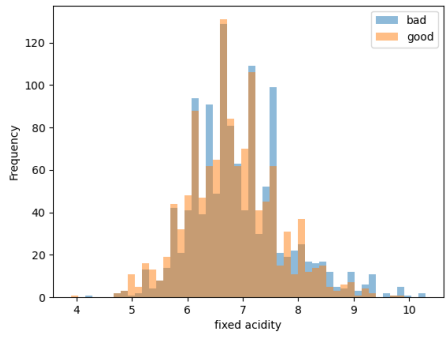
```
#Read the csv file and set the first row as a header
#Define our variables (all but quality) and the target (quality)
dataset = read_csv("winequalitywhite.csv", sep=';', header=0)
names = dataset.columns.tolist()
vars = names[:-1]
target = [names[-1]]

#Find quality median
#Check the number of instances with quality higher or lower than median
#Add the instances of quality = median to the shortest set defining the classification
threshold
median = dataset.median(axis = 0)['quality']
data_good = dataset[dataset['quality'] > median]
data_bad = dataset[dataset['quality'] < median]

if len(data_good.index) < len(data_bad.index):
    data_good = data_good.append(dataset[dataset['quality'] == median])
else:
    data_bad = data_bad.append(dataset[dataset['quality'] == median])
```

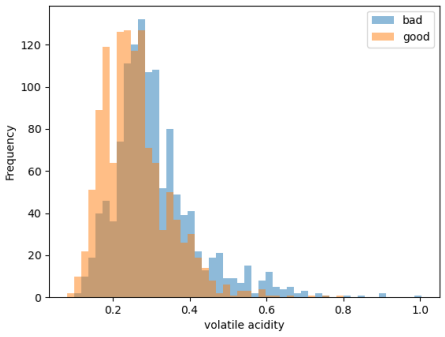
2] Να χωρίσετε με τυχαίο τρόπο τα δεδομένα σε σύνολο εκπαίδευσης (75%) και σύνολο αξιολόγησης (25%) με ίσο αριθμό γεγονότων σε κάθε κλάση.

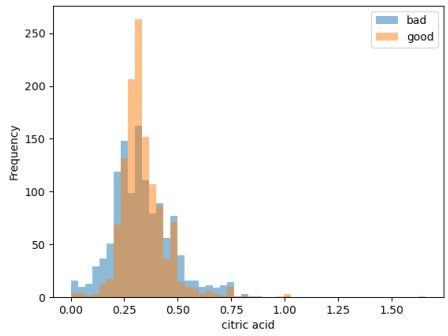
Μιας και οι κλάσεις δεν έχουν ίσο αριθμό γεγονότων, βρίσκουμε την μικρότερη των δύο και χωρίζουμε τα δεδομένα μας σε σύνολο εκπαίδευσης και σύνολο αξιολόγησης με βάση το μικρότερο πλήθος. Τα υπόλοιπα δεδομένα αγνοούνται. Έχουμε δηλαδή συνολικά 1640 γεγονότα σε κάθε κλάση που χωρίζονται σε σύνολο εκπαίδευσης και σύνολο αξιολόγησης.

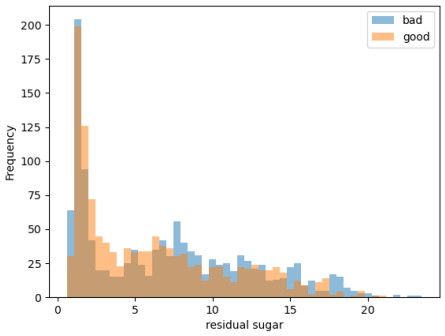
```
#Change the quality values to binary since binary classification is requested.
data_good.loc[:, ['quality']] = 1
data_bad.loc[:, ['quality']] = 0

# The good and bad sets are not equal. We only keep enough of the larger set to equalize
them
eventNo = min(len(data_good.index),len(data_bad.index))

# split into train (75%) and test (25%) datasets
train_good, test_good = train_test_split(data_good, test_size=int(0.25*eventNo),
train_size=int(0.75*eventNo))
train_bad, test_bad = train_test_split(data_bad, test_size=int(0.25*eventNo),
train_size=int(0.75*eventNo))
```

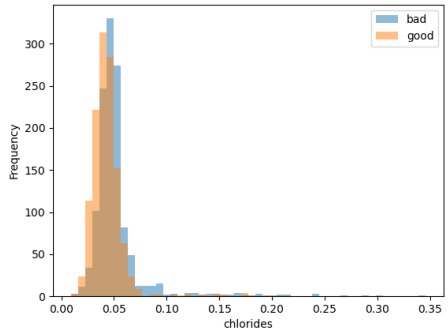
3] Να απεικονίσετε για το σύνολο εκπαίδευσης τις κατανομές των 11 μεταβλητών (ένα διάγραμμα για κάθε μεταβλητή στο οποίο να φαίνεται ξεχωριστά η κατανομή για τα γεγονότα κάθε κλάσης).

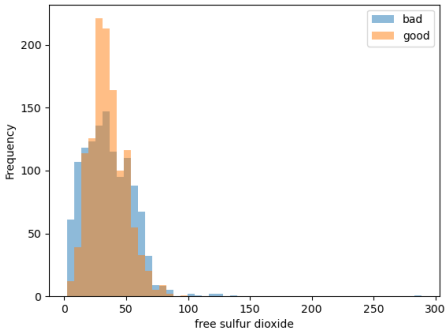


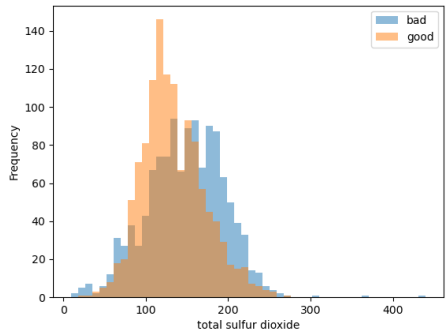


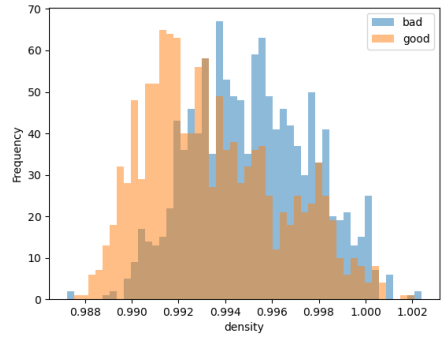


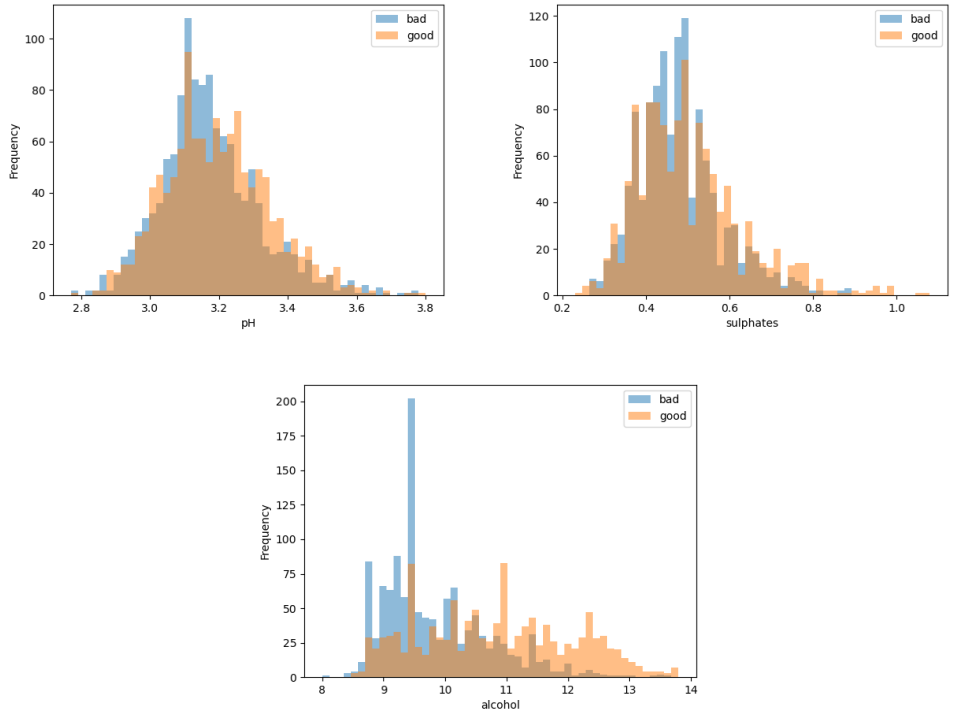










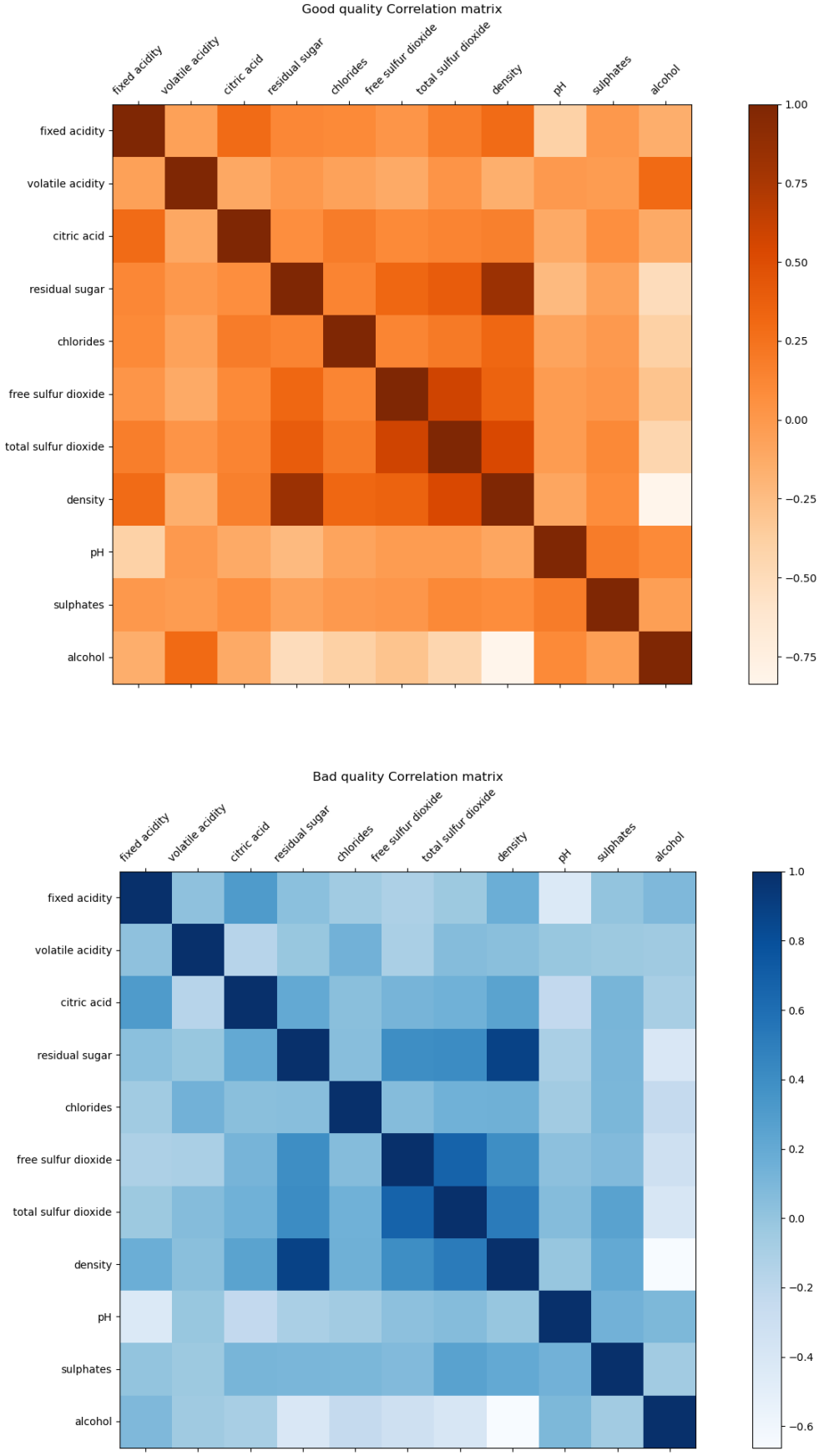


Σχήμα 3.1: Κατανομές του συνόλου εκπαίδευσης για τις μεταβλητές a) fixed acidity, b) volatile acidity, c) citric acid, d) residual sugar, e) chlorides, f) free sulfur dioxide, g) total sulfur dioxide, h) density, i) pH, j) sulphates, k) alcohol

«Με το μάτι» παρατηρούμε ότι μεγαλύτερη διαχωρισιμότητα έχουν οι μεταβλητές αλκοόλ και πυκνότητα. Βέβαια ακόμα και σε αυτές υπάρχει σοβαρή επικάλυψη, συνεπώς δεν μπορεί να προκύψει κάποιο άμεσο συμπέρασμα από τις γραφικές παραστάσεις.

4] Να απεικονίσετε για το σύνολο εκπαίδευσης τις συσχετίσεις όλων των μεταβλητών (ξεχωριστά για τις δύο κλάσεις).

```
# heatmap of variable correlations for the "good wines" class
pyplot.matshow(train_good_sourcevars.corr(), cmap='Oranges')
pyplot.xticks(range(train_good_sourcevars.select_dtypes(['number']).shape[1]),
train_good_sourcevars.select_dtypes(['number']).columns, fontsize=10, rotation = 45)
pyplot.yticks(range(train_good_sourcevars.select_dtypes(['number']).shape[1]),
train_good_sourcevars.select_dtypes(['number']).columns, fontsize=10)
cb = pyplot.colorbar()
pyplot.title('Good quality Correlation matrix')

# heatmap of variable correlations for the "bad wines" class
pyplot.matshow(train_bad_sourcevars.corr(), cmap='Blues')
pyplot.xticks(range(train_bad_sourcevars.select_dtypes(['number']).shape[1]),
train_bad_sourcevars.select_dtypes(['number']).columns, fontsize=10, rotation=45)
pyplot.yticks(range(train_bad_sourcevars.select_dtypes(['number']).shape[1]),
train_bad_sourcevars.select_dtypes(['number']).columns, fontsize=10)
cb = pyplot.colorbar()
pyplot.title('Bad quality Correlation matrix')
pyplot.show()
```

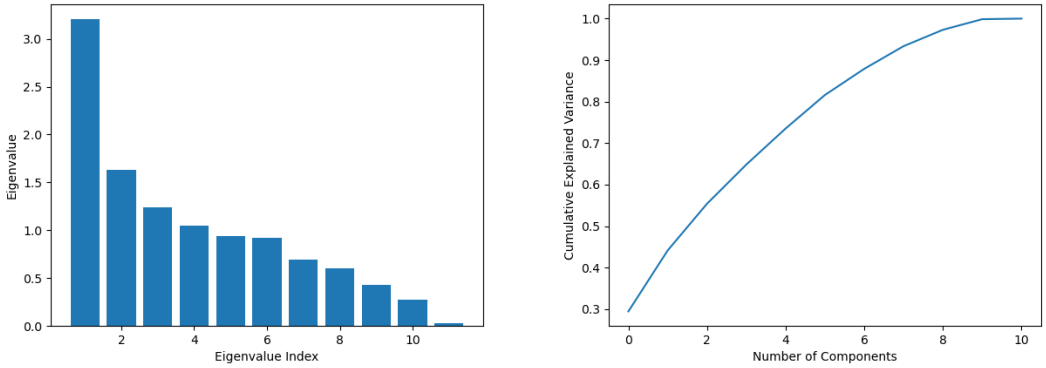


Σχήμα 4.1: Γραφική απεικόνιση των πινάκων συσχέτισης των μεταβλητών α) της «καλής» κλάσης, β) της «κακής» κλάσης

Παρατηρούμε ότι οι συσχετίσεις μεταβλητών μοιάζουν αρκετά και για τις δύο κλάσεις. Συνεπώς όποιες διαφορές μπορεί να υπάρχουν, δεν είναι εύκολο να εντοπιστούν από τις γραφικές παραστάσεις.

5] Να κάνετε ανάλυση PCA στο σύνολο εκπαίδευσης, αφού πρώτα κάνετε τυποποίηση των μεταβλητών. Δείξτε την κατανομή των ιδιοτιμών του πίνακα διασποράς. Σχολιάστε το αποτέλεσμα.

```
data_train_sourcevars_scaled =
preprocessing.StandardScaler().fit_transform(data_train_sourcevars)
pca = PCA()
data_train_sourcevars_scaled_transformed = pca.fit_transform(data_train_sourcevars_scaled)
eigenvalues = pca.explained_variance_
```

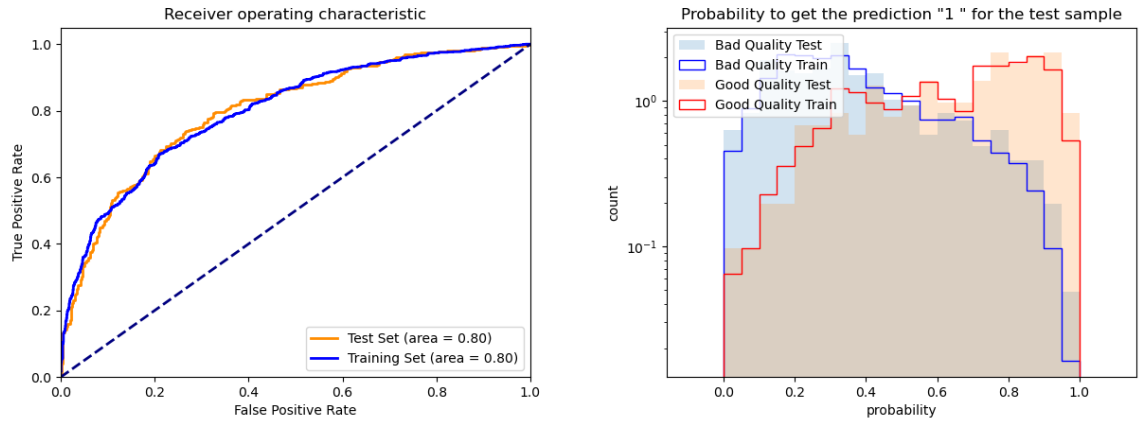


Σχήμα 5.1: α) Κατανομή των ιδιοτιμών του πίνακα διασποράς. β) Ποσοστό περιγραφής δείγματος σε σχέση με τον αριθμό των μεταβλητών

Χρησιμοποιούμε τη μέθοδο PCA για να ελαττώσουμε τις διαστάσεις του προβλήματος. Στην καινούρια βάση παρατηρούμε πως οι πρώτες 6 μεταβλητές θα μπορούσαν να περιγράψουν με αρκετά μεγάλη ακρίβεια το πρόβλημα (~85% της πληροφορίας). Επιπλέον βλέπουμε πως με τις 10 πρώτες έχουμε ουσιαστικά το 100% της πληροφορίας. Παρόλα αυτά, το πλήθος των χαρακτηριστικών είναι μικρό και εύκολα διαχειρίσιμο. Επιπλέον, οι καινούριες αυτές μεταβλητές δεν αντιστοιχούν στα αρχικά μας χαρακτηριστικά αλλά σε γραμμικούς συνδυασμούς αυτών, κάνοντάς τες λιγότερο ερμηνεύσιμες.

6] Να υλοποιήσετε έναν γραμμικό ταξινομητή ελαχίστων τετραγώνων με κατάλληλη βελτιστοποίηση των παραμέτρων του.

```
clf_LDA = LinearDiscriminantAnalysis(solver="lsqr", shrinkage = "auto")
clf_LDA.fit(data_train_sourcevars[vars], data_train_targetvar[target].values.ravel())

#probability to get the prediction "1" for the bad train sample (Ideally these would be
low)
P0 = clf_LDA.predict_proba(test_bad_sourcevars)[:,1]

#probability to get the prediction "1" for the good train sample (Ideally these would be
high)
P1 = clf_LDA.predict_proba(test_good_sourcevars)[:,1]
```



Σχήμα 6.1: α) Η καμπύλη ROC για τα δύο σύνολα του ταξινομητή ελαχίστων τετραγώνων. β) Η κατανομή πιθανοτήτων πρόβλεψης «1» στο σύνολο ελέγχου

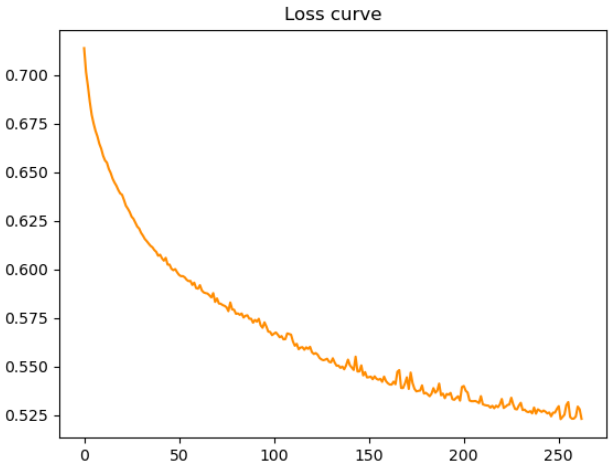
	ACCURACY	AUC	TIME(S)
TRAINING GROUP	0.72	0.80	0.01
TESTING GROUP	0.73	0.80	

Πίνακας 6.1: Χαρακτηριστικές ποσότητες του ταξινομητή ελαχίστων τετραγώνων

7] Να υλοποιήσετε ένα νευρωνικό δίκτυο (με ένα ή δύο κρυφά στρώματα και κατάλληλο αριθμό νευρώνων) που να διαχωρίζει τις δύο κλάσεις με κατάλληλη βελτιστοποίηση των παραμέτρων του. Δείξτε την καμπύλη εκπαίδευσης και σχολιάστε το φαινόμενο της υπερεκπαίδευσης.

```
clf_nn = MLPClassifier(
    hidden_layer_sizes = (32),
    activation = "logistic",
    verbose = False,
    max_iter = 800
)
clf_nn.fit(data_train_sourcevars[vars].values,
data_train_targetvar[target].values.ravel())
```

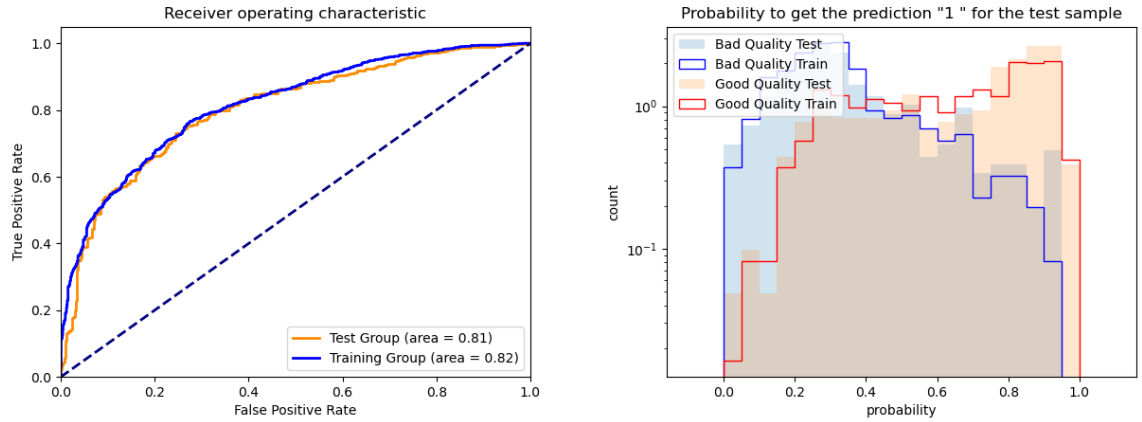
Λόγω χαμηλού πλήθους δεδομένων, δεν ήταν απαραίτητη η χρήση δεύτερου κρυφού στρώματος. Η καμπύλη ελαχιστοποίησης σφάλματος ακολουθεί. Παρατηρούμε ότι το σφάλμα συνεχώς μειώνεται που σημαίνει ότι το δίκτυό μας «μαθαίνει». Η μέθοδος που χρησιμοποιήσαμε ελέγχει τη μεταβολή του σφάλματος και σταματάει τη διαδικασία όταν αυτό δεν μειώνεται δραστικά για συγκεκριμένο πλήθος εποχών.



Σχήμα 7.1: Καμπύλη εκμάθησης του νευρωνικού δικτύου

```
#NN Loss curve
pyplot.plot(clf_nn.loss_curve_,color="darkorange",label="MLP1")
pyplot.title("Loss curve")
pyplot.show()

#probability to get the prediction "1" for the bad train sample (Ideally these would be
low)
P0 = clf_nn.predict_proba(test_bad_sourcevars)[:,1]
#probability to get the prediction "1" for the good train sample (Ideally these would be
high)
P1 = clf_nn.predict_proba(test_good_sourcevars)[:,1]
```

Στην προσπάθεια ελέγχου υπερεκπαίδευσης χρησιμοποιήσαμε τη μέθοδο KS. Όπως βλέπουμε και στο παρακάτω γράφημα, οι κατανομές από τα σύνολα ελέγχου και εκπαίδευσης μοιάζουν αρκετά. Η επικάλυψη των κλάσεων καθώς και το μεγάλο πλήθος «counts» στις μεσαίες τιμές μας δείχνουν πως το μοντέλο μας δεν παρουσιάζει υψηλή βεβαιότητα κατά την ταξινόμηση, γεγονός που μπορεί από μόνο του να οδηγήσει στις μικρές διαφορές μεταξύ των κατανομών. Συνεπώς αυτές δεν μας οδηγούν στο συμπέρασμα ότι το σύστημά μας δυσκολεύεται να γενικεύσει.



Σχήμα 7.2: α) Η καμπύλη ROC για τα δύο σύνολα του Νευρωνικού δικτύου. β) Η κατανομή πιθανοτήτων πρόβλεψης «1» στο σύνολο ελέγχου

Αντίστοιχα αποτελέσματα παίρνουμε και από την καμπύλη ROC καθώς η απόδοση στα δύο σύνολα είναι παρόμοια.

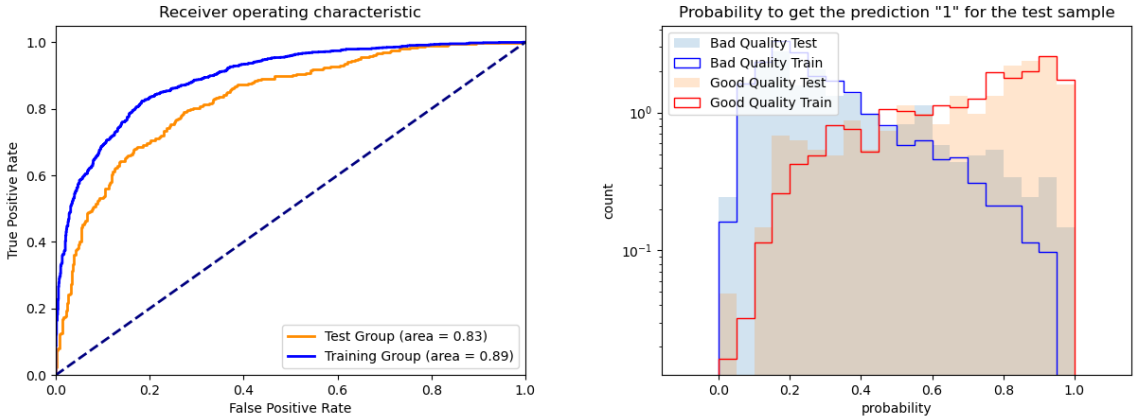
	ACCURACY	AUC	TIME(S)
TRAINING GROUP	0.73	0.82	1.94
TESTING GROUP	0.74	0.81	

Πίνακας 7.1: Χαρακτηριστικές ποσότητες του νευρωνικού δικτύου

8] Να υλοποιήσετε ένα ενδυναμωμένο δέντρο απόφασης που να διαχωρίζει τις δύο κλάσεις. Επιλέξτε κατάλληλο αριθμό δέντρων .

Υλοποιήσαμε τη μέθοδο gradient boosting classifier με μέγιστο βάθος 3. Από τις γραφικές παραστάσεις μπορούμε να διακρίνουμε μια μεγαλύτερη διαχωρισιμότητα σε σχέση με τις παραπάνω μεθόδους. Συμπεριλαμβάνεται επιπλέον πίνακας με χαρακτηριστικά

```
clf_gbm = GradientBoostingClassifier(
  n_estimators=200,
  learning_rate=0.05,
  max_depth=3,
  subsample=0.5,
  validation_fraction=0.1,
  n_iter_no_change=20,
  max_features='log2',
  verbose=1)
clf_gbm.fit(data_train_sourcevars, data_train_targetvar[target].values.ravel())
```

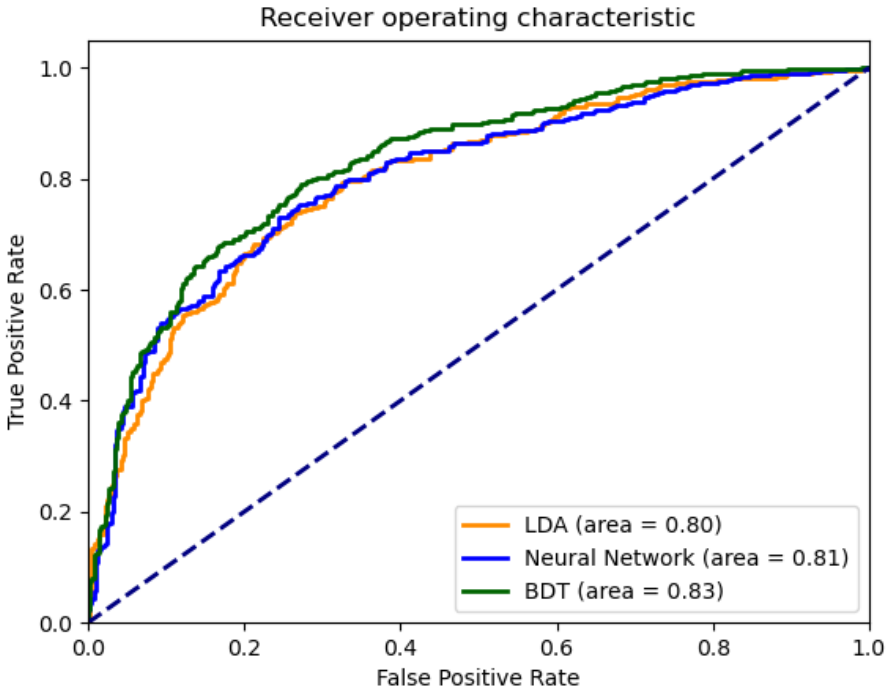


Σχήμα 8.1: α) Η καμπύλη ROC για τα δύο σύνολα της μεθόδου ενδυναμωμένου δέντρου απόφασης. β) Η κατανομή πιθανοτήτων πρόβλεψης «1» στο σύνολο ελέγχου

	ACCURACY	AUC	TIME(S)
TRAINING GROUP	0.81	0.89	0.29
TESTING GROUP	0.76	0.83	

Πίνακας 8.1: Χαρακτηριστικές ποσότητες του ενδυναμωμένου δέντρου απόφασης

9] Να συγκρίνετε την απόδοση των ταξινομητών με κατάλληλη επιλογή μετρικής και να σχολιάσετε τα αποτελέσματα.



Σχήμα 9.1: Καμπύλη ROC για το σύνολο ελέγχου και από τις τρεις μεθόδους

CLASSIFIER	TRAINING TIME (S)
LEAST SQUARES	0.01
BOOSTED DECISION TREE	1.94
NEURAL NETWORK	0.29

Πίνακας 9.1: Χρόνοι εκπαίδευσης

	ACCURACY	PRECISION	RECALL	F1-SCORE	SUPPORT
LEAST SQUARES	0.73	0.67	0.73	0.73	2460
NEURAL NETWORK	0.74	0.68	0.71	0.73	2460
BOOSTED DECISION TREE	0.76	0.69	0.76	0.76	2460

Πίνακα 9.2: Βασικά χαρακτηριστικά όλων των μεθόδων

Η σύγκριση των ταξινομητών γίνεται με δύο διαφορετικούς τρόπους. Γραφικά μπορούμε να δούμε τις καμπύλες ROC, στις οποίες φαίνεται πως τα δέντρα απόφασης έχουν καλύτερη απόδοση. Το νευρωνικό δίκτυο και η μέθοδος ελαχίστων τετραγώνων βέβαια είναι συγκρίσιμες με ελαφρώς μικρότερο εμβαδόν. Η δεύτερη μέθοδος σύγκρισης είναι με χρήση βασικών χαρακτηριστικών που φαίνονται στον πίνακα [9.2]. Και εδώ παρατηρούμε αντίστοιχα αποτελέσματα με το ενδυναμωμένο δέντρο απόφασης να έχει τα καλύτερα, ενώ ακολουθεί το νευρωνικό δίκτυο και η γραμμική μέθοδος.

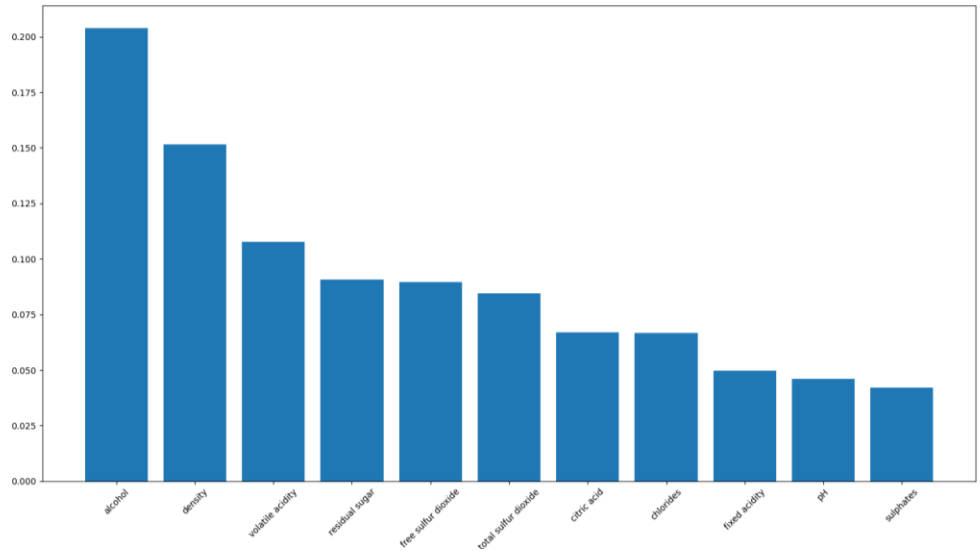
Για το πλήθος των δεδομένων που είχαμε η διαφορά στους χρόνους εκπαίδευσης δεν αποτελεί άμεσο ζήτημα. Παρόλα αυτά αξίζει να σημειωθεί πως το νευρωνικό δίκτυο χρειάζεται αρκετά περισσότερο χρόνο από τις άλλες διαδικασίες ενώ παράλληλα δεν δίνει τα καλύτερα αποτελέσματα.

Σημειώνεται πως για τον ταξινομητή ελαχίστων τετραγώνων έγινε επιλογή του attribute “shrinkage = auto” για τον υπολογισμό του πίνακα συνδιασποράς [1]. Χωρίς αυτή τη ρύθμιση, η εμπειρική φόρμουλα υπολογισμού κατέληγε σε αισθητά χειρότερα αποτελέσματα.

10] Για κάθε ταξινομητή, να κατατάξετε τις μεταβλητές σύμφωνα με την επίδρασή τους σε αυτόν. Σχολιάστε τις ομοιότητες και τις διαφορές μεταξύ των διαφορετικών ταξινομητών.

Για το boosted decision tree, χρησιμοποιήθηκε ή διαθέσιμη μέθοδος «feature_importances_». Τα αποτελέσματα φαίνονται παρακάτω

```
feature_importances = DataFrame({"Variable_Name":vars,
"Importance":clf_gbm.feature_importances_}).sort_values('Importance', ascending=False)
pyplot.bar(feature_importances["Variable_Name"], feature_importances["Importance"])
pyplot.xticks(rotation=45)
pyplot.show()
```

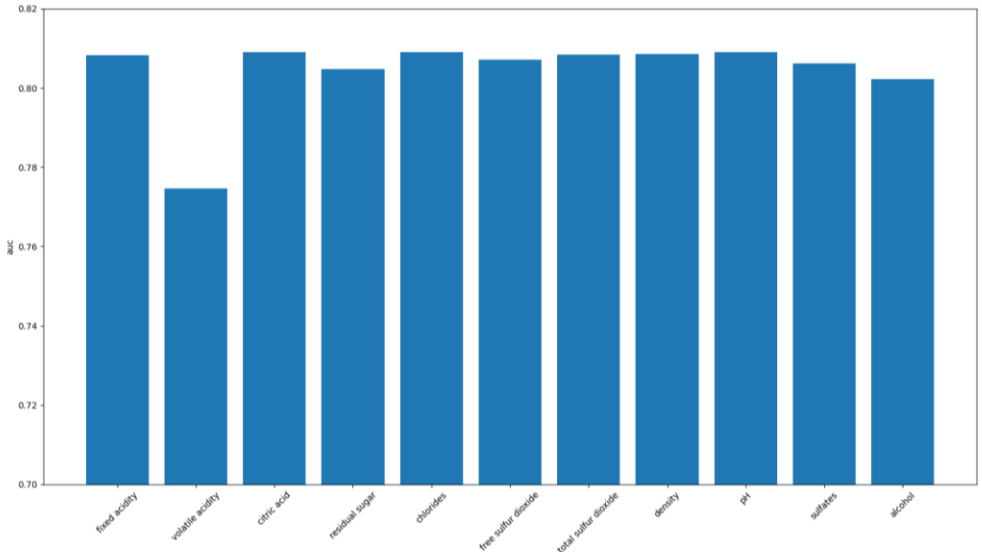


Σχήμα 10.1: Σημαντικότητα των μεταβλητών για το ενδυναμωμένο δέντρο απόφασης

Επειδή αντίστοιχη μέθοδος δεν είναι διαθέσιμη για το νευρωνικό δίκτυο και τον ταξινομητή ελαχίστων τετραγώνων, ακολουθήσαμε μια επαναληπτική διαδικασία, κατά την οποία ελέγξαμε το εμβαδόν της καμπύλης roc αφαιρώντας κάθε φορά μια από τις μεταβλητές. Τονίζεται σε αυτό το σημείο, ότι στις γραφικές παραστάσεις που ακολουθούν, οι πιο σημαντικές μεταβλητές έχουν και τη χαμηλότερη μπάρα, αφού ελλείψει αυτών μειώνεται περισσότερο η ακρίβεια του ταξινομητή.

```
clf = LinearDiscriminantAnalysis(solver="lsqr", shrinkage = "auto")

x = []
y = []

pyplot.figure()
for k in vars:
    vars1 = vars.copy()
    vars1.remove(k)
    print(vars1)
    clf.fit(data_train_sourcevars[vars1].values,
data_train_targetvar[target].values.ravel())
    fpr, tpr, threshold = roc_curve(data_train_targetvar,
clf.predict_proba(data_train_sourcevars[vars1])[:,1])
    roc_auc = auc(fpr, tpr)
    x.append(k)
    y.append(roc_auc)
```

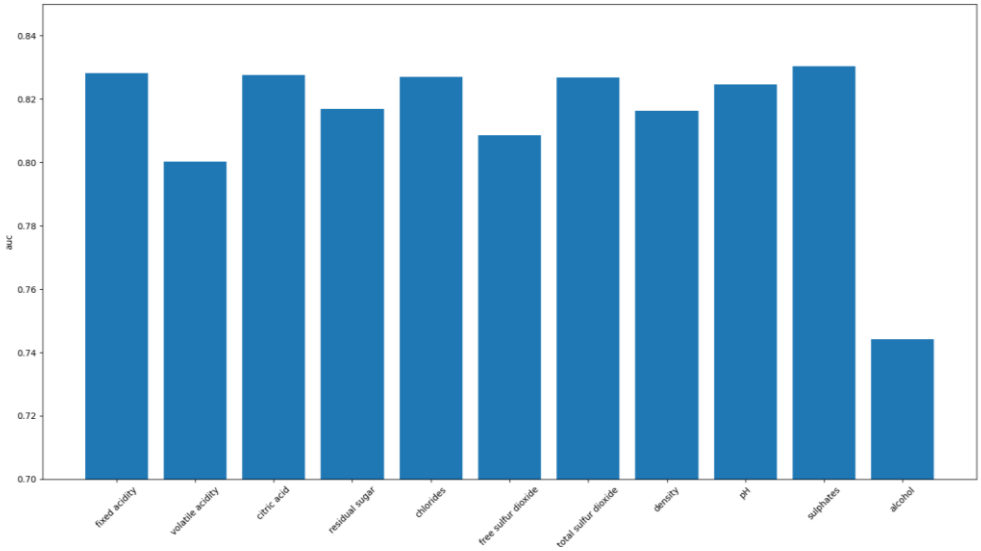
```
plot1 = pyplot.figure(1)
pyplot.ylabel('auc')
pyplot.xlabel('removed variable')
pyplot.bar(x, y)
ax = pyplot.gca()
ax.set_ylim([0.7, 0.82])
pyplot.xticks(rotation = 45)
pyplot.show()
```



Σχήμα 10.2: Εμβαδόν κάτω από την καμπύλη ROC όταν απουσιάζει η κάθε μεταβλητή για τον ταξινομητή ελαχίστων τετραγώνων

```
clf = MLPClassifier(
    hidden_layer_sizes=(32),
    activation="logistic",
    max_iter=800
)

x = []
y = []

pyplot.figure()
for k in vars:
    vars1 = vars.copy()
    vars1.remove(k)
    print(vars1)
    clf.fit(data_train_sourcevars[vars1].values,
data_train_targetvar[target].values.ravel())
    fpr, tpr, threshold = roc_curve(data_train_targetvar,
clf.predict_proba(data_train_sourcevars[vars1])[:,1])
    roc_auc = auc(fpr, tpr)
    x.append(k)
    y.append(roc_auc)

print(x,y)
```

```
plot1 = pyplot.figure(1)
pyplot.ylabel('auc')
pyplot.xlabel('removed variable')
pyplot.bar(x, y)
ax = pyplot.gca()
ax.set_ylim([0.70, 0.85])
pyplot.xticks(rotation = 45)
pyplot.show()
```



Σχήμα 10.2: Εμβαδόν κάτω από την καμπύλη ROC όταν απουσιάζει η κάθε μεταβλητή για το νευρωνικό δίκτυο

RANK	LEAST SQUARES	NEURAL NETWORK	BOOSTED DECISION TREE
1	Volatile Acidity	Alcohol	Alcohol
2	Alcohol	Volatile Acidity	Density
3	Residual Sugar	Free sulfur dioxide	Volatile Acidity
4	Sulfates	Density	Residual Sugar
5	Free Sulfur Dioxide	Residual Sugar	Free Sulfur Dioxide
6	Total Sulfur Dioxide	pH	Total Sulfur Dioxide
7	Density	Total sulfur Dioxide	Citric Acid
8	pH	Chlorides	Chlorides
9	Fixed Acidity	Citric Acid	Fixed Acidity
10	Citric Acid	Fixed Acidity	pH
11	Chlorides	Sulphates	Sulphates

Πίνακας 10.1: Σημαντικότητα μεταβλητών για τη κάθε μέθοδο

Παρατηρούμε ότι για τις δύο μη γραμμικές μεθόδους τα άκρα ταυτίζονται, με το αλκοόλ να είναι ύψιστης σημασίας με αρκετά μεγάλη διαφορά και τις θειικές ενώσεις να βρίσκονται στο τέλος της λίστας. Αντιθέτως η μέθοδος ελαχίστων τετραγώνων αποδίδει σχεδόν την ίδια σημασία στις περισσότερες μεταβλητές με μόνο την πτητική οξύτητα να ξεχωρίζει.

Παρά τις διαφορές, το νευρωνικό δίκτυο και τα ενδυναμωμένα δέντρα απόφασης έχουν παρόμοια συμπεριφορά αν σκεφτεί κανείς πως μέσα στις μεθόδους υπάρχουν στοιχεία τυχαιότητας και τα διαγράμματα μπορεί ελαφρώς να αλλάξουν σε κάθε επανάληψη.

11] Συμπεράσματα

Κοιτώντας τις τρεις μεθόδους ένα εύλογο συμπέρασμα είναι πως η ταξινόμηση των συγκεκριμένων δεδομένων σε δύο κλάσεις είναι λίγο «θολή». Αυτό μπορεί να φανεί και από τις αρχικές κατανομές των χαρακτηριστικών μας, η πλειοψηφία των οποίων δεν παρουσιάζουν ουσιαστική διαχωρισιμότητα. Παρόλα αυτά και οι τρεις κατάφεραν να ταξινομήσουν το σύνολο ελέγχου με κάποια επιτυχία και άρα αναγνώρισαν πρότυπα που δεν εντοπίζονται εύκολα χωρίς τέτοιου είδους ανάλυση.

Το νευρωνικό δίκτυο, αν και είναι η πιο απαιτητική μέθοδος από τις τρεις, δεν είχε ανάλογα αποτελέσματα. Τα ενδυναμωμένα δέντρα απόφασης ήταν τα πλέον αποδοτικά με αρκετά μικρό χρόνο εκπαίδευσης και τα καλύτερα αποτελέσματα ενώ ο γραμμικός ταξινομητής είχε επιτυχία παρά την έλλειψη κάποιας προφανής γραμμικότητας στο πρόβλημα.

12] Αναφορές

1. *Ledoit O, Wolf M. Honey, I Shrunk the Sample Covariance Matrix. The Journal of Portfolio Management 30(4), 110-119, 2004.*