ΜΈΘΟΔΟΙ ΣΤΑΤΙΣΤΙΚΉΣ ΚΑΙ ΜΗΧΑΝΙΚΉΣ ΜΆΘΗΣΗΣ

ΝΙΚΌΛΑΟΣ ΠΑΠΑΓΕΩΡΓΊΟΥ 3200131

ΕΡΓΑΣΙΑ 2:

Πριν ξεκινήσουμε να αναλύουμε τα δεδομένα μας και τα μοντέλα μας είναι πολύ σημαντικό να κάνουμε μια αναφορά στις μεταβλητές που μας δίνονται και θα μας απασχολήσουν.

ΜΕΤΑΒΛΗΤΕΣ:

ΣΤΟΙΧΕΊΑ ΠΟΥ ΑΦΟΡΟΎΝ ΤΟΝ ΠΕΛΆΤΗ:

Age: Αφορά την ηλικία του πελάτη ,numeric

Job: Αφορά την εργασία του πελάτη , categorical

Marital: Αφορά την οικογενειακή κατάσταση (παντρεμένος , χωρισμένος , ελεύθερος) , categorical

Education: Αφορά το επίπεδο μόρφωσης του πελάτη , categorical

Default: Αν ο πελάτης έχει πίστωση σε αθέτηση , categorical

Housing: Αν ο πελάτης έχει στεγαστικό δάνειο , categorical

Loan: Αν ο πελάτης έχει προσωπικό δάνειο , categorical

ΣΤΟΙΧΕΊΑ ΠΟΥ ΑΦΟΡΟΎΝ ΤΗΝ ΤΕΛΕΥΤΑΊΑ ΕΠΑΦΉ ΓΙΑ ΤΗΝ ΤΡΈΧΟΥΣΑ ΚΑΜΠΆΝΙΑ:

Contact: Αφορά τον τρόπο με τον οποίο έγινε η τελευταία επικοινωνία , categorical

Month: Αφορά τον τελευταίο μήνα που έγινε η τελευταία επαφή , categorical

Day-of-the-week: Αφορά τη τελευταία μέρα που πραγματοποιήθηκε επαφή , categorical

Duration: Αφορά τη διάρκεια της τελευταίας επαφής σε δευτερόλεπτα ,numeric

ΑΛΛΕΣ ΜΕΤΑΒΛΗΤΈΣ:

Campaign: Αφορά τον αριθμό επαφών που πραγματοποιήθηκαν για αυτήν την καμπάνια στον πελάτη αυτό , numeric

Pdays: Αφορά τις μέρες που έχουν περάσει από την τελευταία επαφή με το πελάτη για παλαιότερη καμπάνια (Όταν συναντάμε τιμή 999 σημαίνει ότι δεν έχει υπάρξει ποτέ επαφή) , numeric

Previous: Αφορά τον αριθμό των επαφών που έγιναν με αυτό τον πελάτη πριν από αυτή την καμπάνια , numeric

Poutcome: Αφορά το αποτέλεσμα που είχε γι αυτόν τον πελάτη η προηγούμενη καμπάνια (success,failure) , categorical

Emp.var.rate: Αφορά το ποσοστό διακύμανσης της απασχόλησης (τριμηνιαίος δείκτης) , numeric

Cons.price.idx: Αφορά τον δείκτη τιμών του καταναλωτή (μηνιαίος δείκτης) , numeric

Cons.conf.idx: Αφορά τον δείκτη καταναλωτικής εμπιστοσύνης (μηνιαίος δείκτης) ,numeric

Euribor3m: Αφορά το επιτόκιο (ημερήσιος δείκτης) , numeric

Nr.employed: Αφορά τον αριθμό των εργαζομένων (τριμηνιαίος δείκτης) , numeric

ΜΕΤΑΒΛΗΤΉ ΠΟΥ ΘΈΛΟΥΜΕ ΝΑ ΠΡΟΒΛΈΨΟΥΜΕ:

SUBSCRIBED: Αφορά το εάν ο πελάτης έκανε εγγραφή δηλαδή αγόρασε το προϊόν ή όχι , binary

ΔΕΔΟΜΈΝΑ:

Τα δεδομένα που μας δίνονται αφορούν τηλεφωνήματα τηλεμαρκετινγκ για την πώληση ενός καινούργιου προϊόντος μακροπρόθεσμων καταθέσεων. Οι παραπάνω μεταβλητές αφορούν τις κλήσεις που πραγματοποιήθηκαν στην διάρκεια αυτής της καμπάνιας, κάποια προσωπικά στοιχείων των πελατών/πολιτών που πήραν μέρος , στοιχεία από προηγούμενες καμπάνιες καθώς και στοιχεία για διάφορους οικονομικούς δείκτες. Αποτέλεσμα της καμπάνιας αποτελεί η μεταβλητή SUBSCRIBE η οποία δείχνει αν ο πελάτης εγγράφηκε/αγόρασε το προιόν ή όχι. Έχουμε ένα σύνολο 39.883 τηλεφωνικών κλήσεων για να μελετήσουμε οι οποίες έλαβαν μέρος ανάμεσα στον Μάιο του 2009 και στον Ιούνιο του 2010. **Τώρα καλούμαστε να μελετήσουμε τα δεδομένα και τις μεταβλητές που μας δίνονται να τα επεξεργαστούμε και να μπορέσουμε να προβλέψουμε αν η επαφή θα είναι επιτυχημένη ή όχι**.

ΣΚΟΠΟΣ: Να προβλέψουμε με όσο το δυνατόν μεγαλύτερη ακρίβεια το αν η επαφή με το πελάτη θα είναι επιτυχημένη ή όχι.

Παρατηρώντας τα δεδομένα και τις τιμές κάποιων μεταβλητών βλέπουμε πως κάποιες μεταβλητές εμφανίζουν ίδιες τιμές στο μεγαλύτερο αριθμό των παρατηρήσεων όπως είναι η τιμή 999 στη μεταβλητή pdays η οποία δείχνει ότι δεν έχει ξαναγίνει επαφή για προηγούμενη καμπάνια με το πελάτη αυτό.

```
> count_999 <- sum(data$pdays == 999)
> print(count_999)
[1] 38831
>
```

Βλέπουμε έτσι ότι ένα ποσοστό >95% των παρατηρήσεων έχει ίδια τιμή στη μεταβλητή αυτή επομένως θα μπορούσαμε να την αφαιρέσουμε καθώς δεν θα μας έδινε μεγάλη πληροφορία με σκοπό την πρόβλεψη της μεταβλητής που θέλαμε. Κάτι ανάλογο θα μπορούσαμε να κάνουμε και για άλλες μεταβλητές αλλά αυτή παρουσιάζει το μεγαλύτερο ποσοστό επανεμφάνισης.

Παράλληλα μπορούμε να δούμε και το correlation matrix των numeric μεταβλητών που έχουμε με σκοπό να δούμε το βαθμό συσχέτισης τόσο των μεταβλητών μεταξύ τους όσο και της μεταβλητής που θέλουμε να προβλέψουμε σε σχέση με τις υπόλοιπες

```
> dfcor<-data[,-c(2:10,15)]
> dfcor$SUBSCRIBED <-(ifelse(dfcor$SUBSCRIBED == "no", 1, 2))
> correlation_matrix<-cor(dfcor)
> print(correlation_matrix)
                      age     duration     campaign        pdays     previous emp.var.rate cons.price.idx cons.conf.idx    euribor3m  nr.employed  SUBSCRIBED
age            1.000000000 -0.004774988  0.004226729 -0.03100257  0.016182030  0.00433750   -0.005213373   0.139640509  0.01933262 -0.007095074  0.02114890
duration      -0.004774988  1.000000000 -0.072516990 -0.03485204  0.007959439 -0.02299321   -0.002325485  -0.006751207 -0.02476727 -0.033163801  0.41146883
campaign       0.004226729 -0.072516990  1.000000000  0.04702925 -0.078454268  0.14653872    0.142029694  -0.016811694  0.12887596  0.141080185 -0.06009786
pdays         -0.031002573 -0.034852035  0.047029253  1.00000000 -0.525682302  0.26527461    0.179545704  -0.131136181  0.26185957  0.308811518 -0.27066380
previous       0.016182030  0.007959439 -0.078454268 -0.52568230  1.000000000 -0.44447310   -0.347881571  -0.037384425 -0.44989776 -0.463145905  0.17861017
emp.var.rate   0.004337500 -0.022993209  0.146538724  0.26527461 -0.444473097  1.00000000    0.846902517   0.197729701  0.97811598  0.954185682 -0.28262780
cons.price.idx -0.005213373 -0.002325485  0.142029694  0.17954570 -0.347881571  0.84690252    1.000000000   0.086605926  0.80401771  0.732718079 -0.20255751
cons.conf.idx   0.139640509 -0.006751207 -0.016811694 -0.13113618 -0.037384425  0.19772970    0.086605926   1.000000000  0.28239050  0.087052501  0.06877015
euribor3m       0.019332623 -0.024767270  0.128875959  0.26185957 -0.449897765  0.97811598    0.804017713   0.282390498  1.00000000  0.960852737 -0.27343015
nr.employed    -0.007095074 -0.033163801  0.141080185  0.30881152 -0.463145905  0.95418568    0.732718079   0.087052501  0.96085274  1.000000000 -0.30444936
SUBSCRIBED      0.021148897  0.411468832 -0.060097858 -0.27066380  0.178610172 -0.28262780   -0.202557514   0.068770150 -0.27343015 -0.304449363  1.00000000
```

Από τα μεγάλα ποσοστά συσχέτισης κάποιων μεταβλητών μπορούμε να σκεφτούμε μεταβλητές τις οποίες θα μπορούσαμε να διώξουμε καθώς έχουν κοινή μεταβολή και πιθανόν να αντικατοπτρίζουν κοινά πράγματα. Τέτοιες μεταβλητές είναι η Nr.employed και η Emp.var.rate (με 0.95 συσχέτιση) όπου με μια γρήγορη ματιά αρχικά στη σημασία τους βλέπουμε ότι αναφέρονται σε ίδιο δείκτη ,ενώ και σαν τιμές έχουν παρόμοια διακύμανση . Έτσι μπορούμε να διώξουμε την μια από τις δύο και να μην έχουμε σημαντική απώλεια. Επιλέγουμε να διώξουμε την Emp.var.rate καθώς έχει μικρότερη συσχέτιση σε απόλυτη τιμή με την μεταβλητή που θέλουμε να προβλέψουμε και θέλουμε όσο γίνεται οι μεταβλητές μας να έχουν ισχυρή συσχέτιση με τη μεταβλητή SUBSCRIBED. Έχοντας αυτό σαν γνώμονα μπορούμε να δούμε και μεταβλητές όπως age,campaign, Cons.conf.idx οι οποίες σε απόλυτη τιμή έχουν πολύ μικρή συσχέτιση με την SUBSCRIBED και είναι καλό να το θυμόμαστε για παρακάτω. Για αρχή στη μεταβλητή df2 με την οποία θα δουλέψουμε όλες τι μεθόδους και θα φωρτόσουμε τα δεδομένα θα αφαιρέσουμε τις μεταβλητές pdays για τον λόγο που αναφέραμε και πάνω, την Emp.var.rate και την housing.

<u>Τρόπος σκέψης για σύγκριση μεθόδων:</u>

Ξεκινώντας είναι σημαντικό να αναφέρουμε ότι θα χωρίσουμε τις παρατηρήσεις των δεδομένων μας σε δύο μέρη. Αυτές στις οποίες θα εφαρμόσουμε αρχικά τις διάφορες μεθόδους classification οι οποίες θα είναι σε αριθμό 31.000(περίπου το 80% των παρατηρήσεων) και τις υπόλοιπες 8883 τις οποίες θα χρησιμοποιήσουμε μετά για να ξανά εφαρμόσουμε τις μεθόδους του classification και θα είναι αυτές οι οποίες θα μας δώσουν πιο σαφή αποτελέσματα σχετικά με την καλύτερη απόδοση μεθόδου, δηλαδή ποια λειτούργησε καλύτερα στα δεδομένα μας.

<u>LDA:</u>

Η μέθοδος LDA(Linear Discriminant Analysis) αποτελεί μια τεχνική κατηγοριοποίησης και διάκρισης η οποία χρησιμοποιείται κυρίως για την ανάλυση πολλαπλών μεταβλητών. Στο πλαίσιο της ταξινόμησης και του classification, η LDA αναζητά τον γραμμικό συνδυασμό των χαρακτηριστικών που μεγιστοποιεί τον διαχωρισμό μεταξύ διαφορετικών κατηγοριών. Συγκεκριμένα η LDA επιχειρεί να βρει τη γραμμική συνάρτηση των χαρακτηριστικών που διαχωρίζει καλύτερα τις διάφορες κατηγορίες στα δεδομένα μας.

Στα πλαίσια της εφαρμογής του LDA classification πραγματοποιήσαμε δοκιμές τρέχοντας την μέθοδο στα δεδομένα χωρίς περαιτέρω επεξεργασία στο πρώτο στάδιο, μετά πραγματοποιήσαμε την ίδια διαδικασία αλλά πρώτα είχαμε χωρίσει τα δεδομένα μας σε train και test και τέλος εφαρμόσαμε και LDA cross validation(10 folds) ώστε να έχουμε μια πιο καλή εικόνα των αποτελεσμάτων και των μετρικών που μας ενδιαφέρουν. **Η μέθοδος cross validation χωρίζει τα δεδομένα σε n folds(ανάλογα με το πόσα επιθυμεί ο χρήστης) και σε κάθε επανάληψη κάνει train με n-1 folds και test με το ένα fold που δε χρησιμοποιήθηκε το train, κάθε φορά αλλάζει το fold αυτό.**

LDA:

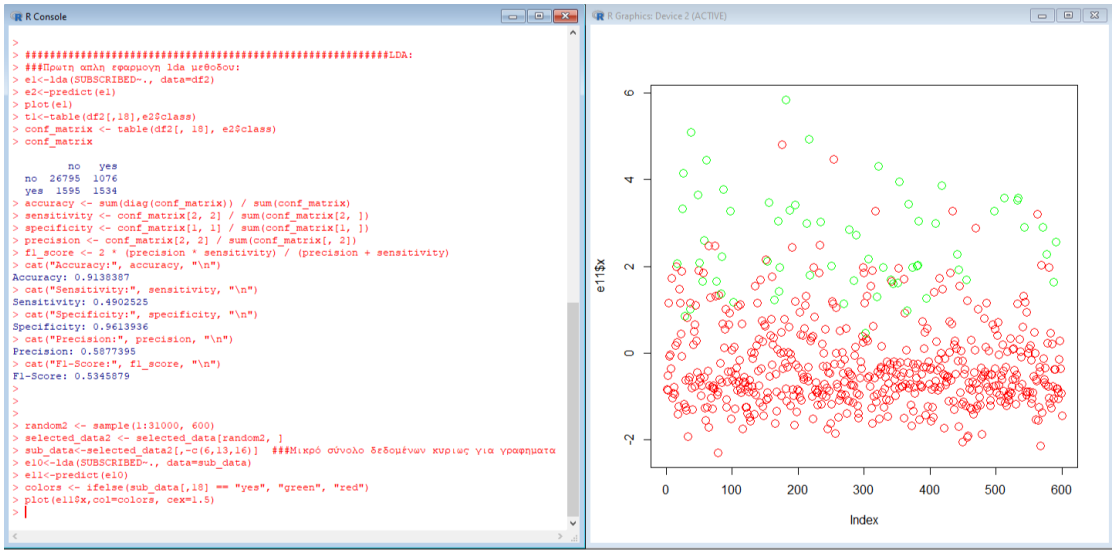
Για αρχή κατεβάσαμε την βιβλιοθήκη “MASS” η οποία χρειάζεται για να εφαρμόσουμε την μέθοδο lda στα δεδομένα μας και ξεκινήσαμε να την υλοποιούμε όπως είπαμε χωρίς περεταίρω επεξεργασία για αρχή. Φορτώσαμε έτσι σε μια μεταβλητή df2 τις 31.000 που αναφέραμε και πάνω με σκοπό σε αυτές να εφαρμόσουμε την μέθοδο μας και αφαιρέσαμε τις μεταβλητές που όπως αναφέραμε πάνω δε θα μας βοηθούσαν ιδιαίτερα στην πρόβλεψη των αποτελεσμάτων μας.(df2<-selected_data[,-c(6,13,16)])

Έτσι ξεκινάμε με την εφαρμογή:
Οι μετρικές που θα μας απασχολήσουν είναι:
Accuracy: Υπολογίζει το συνολικό ποσοστό σωστών προβλέψεων σε σχέση με τον συνολικό αριθμό παραδειγμάτων.

Sensitivity: Υπολογίζει το ποσοστό των πραγματικά θετικών παραδειγμάτων που προβλέφθηκαν σωστά από το μοντέλο.

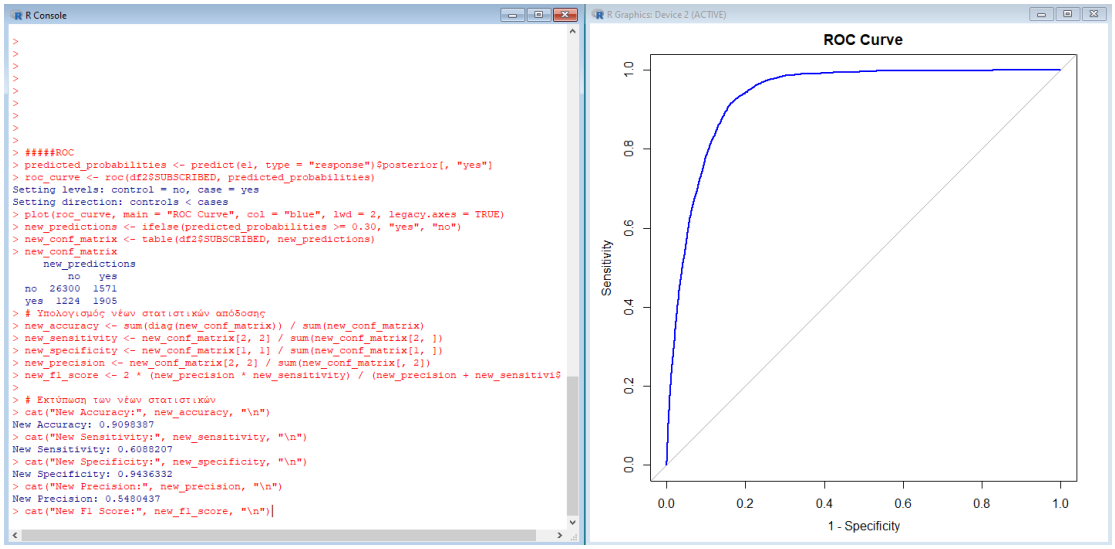
Specificity: Υπολογίζει το ποσοστό των πραγματικά αρνητικών παραδειγμάτων που προβλέφθηκαν σωστά από το μοντέλο.

F1 score: Υπολογίζει τον γεωμετρικό μέσο όρο της precision και sensitivity, παρέχοντας μια μετρική που συνδυάζει τις δύο.



Παρατηρούμε ένα πολύ καλό ποσοστό ακρίβειας/accuracy:0.913 και ένα πολύ ικανοποιητικό ποσοστό specificity=0.96. Από την άλλη έχουμε ένα μέτριο ποσοστό στο sensitivity=0.49 και ομοίως στο precision και στο f1 score, ένας πιθανώς λόγος μπορεί να είναι και η διαφορά σε αριθμό των no με των yes μέσα στα δεδομένα. Αυτά μπορούσαν να γίνουν αντιληπτά και από το conf_matrix το οποίο μας δίνει τα στοιχεία που αναλύουν οι παραπάνω μετρικές και πράγματι παρατηρούμε κακό True Positive το οποίο εκφράζει και η μετρική της ευαισθησίας και αναφέρεται σε αυτές τις παρατηρήσεις όπου το μοντέλο προέβλεψε σωστά ότι ανήκουν στην κλάση “yes” και υψηλό false Negative (παρατηρήσεις που λανθασμένα κατατάχθηκαν στην κλάση no ενώ ανήκουν στην yes)

Για να το διορθώσουμε αυτό θα πρέπει να ασχοληθούμε με το threshold το οποίο είναι μια πιθανότητα βάση της οποίας γίνεται η κατάταξη των παρατηρήσεων και σε default κατάσταση είναι 0.5. Εμείς μέσω της ROC Curve θα βρούμε το βέλτιστο threshold και έπειτα θα το προσαρμόσουμε στην lda.



Επιλέγουμε για threshold το σημείο από το οποίο η καμπύλη απέχει λιγότερο από την άνω αριστερά γωνία (0.0 , 1.0) . Όπως γίνεται αντιληπτό από το σχήμα αυτό το σημείο είναι

κοντά στο χ=0.3 επομένως θα πορευτούμε με την πιθανότητα threshold=0.3 για την συνέχεια και βλέπουμε τα νέα αποτελέσματα.

Πλέον παρατηρούμε μια μικρή μείωση στο accuracy, αλλά μια σημαντική αύξηση στο TPrate ή αλλιώς Sensitivity και στο f1 score ενώ ταυτόχρονα και το conf_matrix φαίνεται βελτιωμένο.

Επομένως με αυτό το threshold θα πορευτούμε και στη συνέχεια.

Τέλος θα δούμε και την αλλαγή των μετρικών αυτών αν αφαιρέσουμε κάποιες μεταβλητές με χαμηλή συσχέτιση με την SUBSCRIBED όπως η age και η Cons.conf.idx και κάποιες όπως την marital η οποία δε σχετίζεται άμεσα με τις υπόλοιπες μεταβλητές και έχει επαναληψιμότητα. (age, Cons.conf.idx,marital)

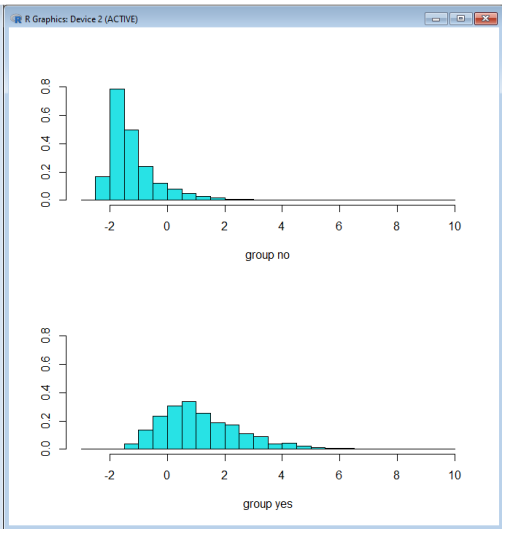
```
> df_without<-df2[,-c(1,3,15)]   ###Αφαιρουμε μεταβλητες με χαμηλη συσχετιση με την SUBSCRIBED
> el<-lda(SUBSCRIBED~., data=df_without)
> predicted_probabilities <- predict(el, type = "response")$posterior[, "yes"]
> new_predictions <- ifelse(predicted_probabilities >= 0.30, "yes", "no")
> conf_matrix <- table(df_without[, 15], new_predictions)
> conf_matrix
     new_predictions
         no   yes
  no  26301  1570
  yes  1226  1903
> accuracy <- sum(diag(conf_matrix)) / sum(conf_matrix)
> sensitivity <- conf_matrix[2, 2] / sum(conf_matrix[2, ])
> specificity <- conf_matrix[1, 1] / sum(conf_matrix[1, ])
> precision <- conf_matrix[2, 2] / sum(conf_matrix[, 2])
> f1_score <- 2 * (precision * sensitivity) / (precision + sensitivity)
> cat("Accuracy:", accuracy, "\n")
Accuracy: 0.9098065
> cat("Sensitivity:", sensitivity, "\n")
Sensitivity: 0.6081815
> cat("Specificity:", specificity, "\n")
Specificity: 0.943669
> cat("Precision:", precision, "\n")
Precision: 0.5479413
> cat("F1-Score:", f1_score, "\n")
F1-Score: 0.576492
>
```

Όπως βλέπουμε τα αποτελέσματα μένουν σχεδόν ίδια κάτι που μας ενθαρρύνει στο να αφαιρέσουμε αυτές τις μεταβλητές στη συνέχεια.

LDA-TRAIN/TEST:

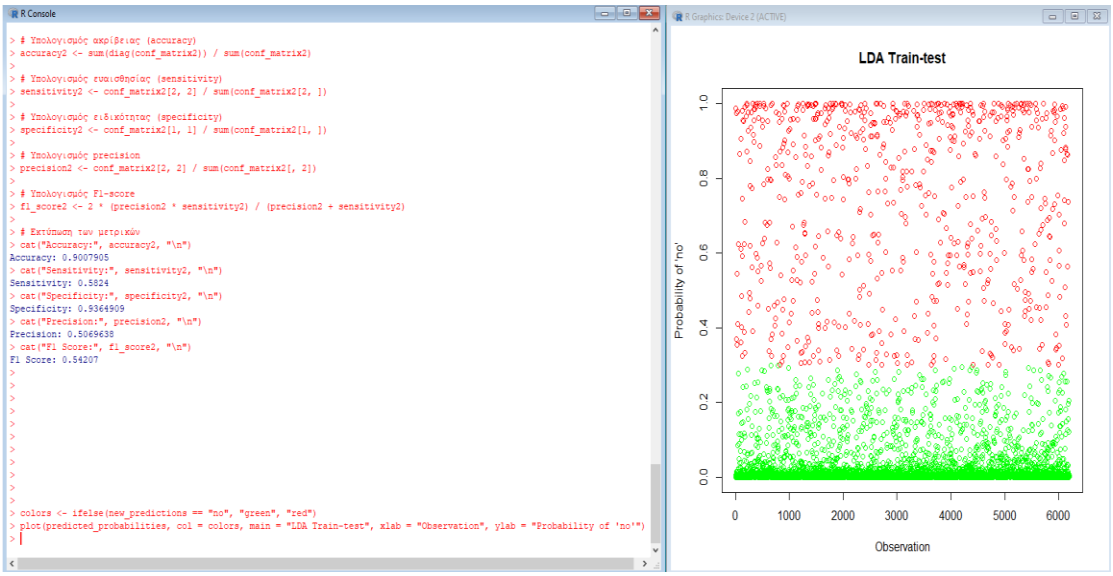
Ομοίως θα εφαρμόσουμε την μέθοδο lda απλά τώρα χωρίζοντας τα δεδομένα μας σε train και test έτσι ώστε για αρχή να εκπαιδεύσουμε το μοντέλο μας στα train δεδομένα και μετά να κάνουμε predict τα test παρακάτω ο κώδικας και τα αποτελέσματα:

```
> ############################################################LDA TRAIN-TEST:
> set.seed(121)
> ind<-createDataPartition(df2$SUBSCRIBED,p=0.8,list=FALSE)
> train<- df2[ind,]
> test<-df2[-ind,]
>
> e4<-lda(SUBSCRIBED~.,data=train)
> plot(e4)
> predicted_probabilities <- predict(e4, newdata = test, type = "response")$posterior[, "yes"]
> new_predictions <- ifelse(predicted_probabilities >= 0.30, "yes", "no")
> # Υπολογισμός confusion matrix
> conf_matrix2 <- table(test$SUBSCRIBED, new_predictions)
>
> # Υπολογισμός ακρίβειας (accuracy)
> accuracy2 <- sum(diag(conf_matrix2)) / sum(conf_matrix2)
>
> # Υπολογισμός ευαισθησίας (sensitivity)
> sensitivity2 <- conf_matrix2[2, 2] / sum(conf_matrix2[2, ])
>
> # Υπολογισμός ειδικότητας (specificity)
> specificity2 <- conf_matrix2[1, 1] / sum(conf_matrix2[1, ])
>
> # Υπολογισμός precision
> precision2 <- conf_matrix2[2, 2] / sum(conf_matrix2[, 2])
>
> # Υπολογισμός F1-score
> f1_score2 <- 2 * (precision2 * sensitivity2) / (precision2 + sensitivity2)
>
> # Εκτύπωση των μετρικών
> cat("Accuracy:", accuracy2, "\n")
Accuracy: 0.9007905
> cat("Sensitivity:", sensitivity2, "\n")
Sensitivity: 0.5824
> cat("Specificity:", specificity2, "\n")
Specificity: 0.9364909
> cat("Precision:", precision2, "\n")
Precision: 0.5069638
> cat("F1 Score:", f1_score2, "\n")
F1 Score: 0.54207
>
```

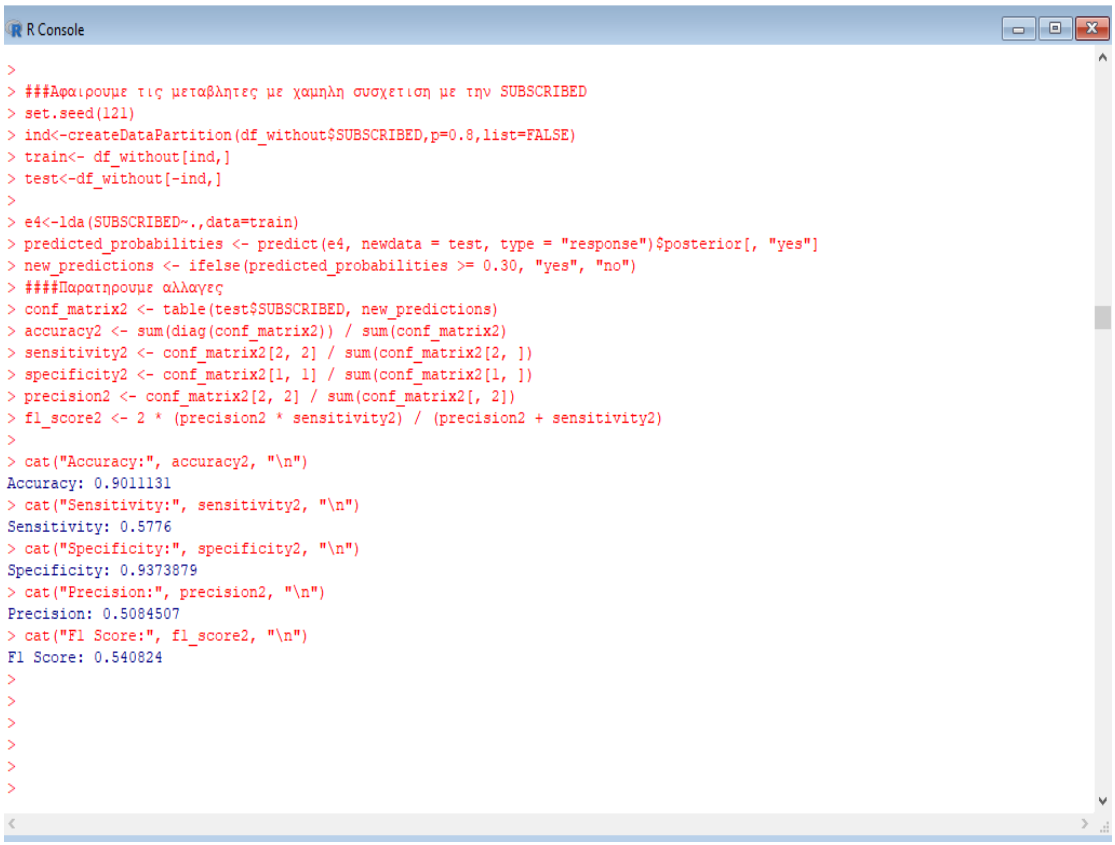


Όπως μπορούμε να παρατηρήσουμε τα αποτελέσματα των στατιστικών μας στοιχείων είναι παρόμοια με αυτά της από πάνω lda μεθόδου ενώ και εδώ χρησιμοποιήσαμε για threshold το 0.3 για τον ίδιο λόγο με το από πάνω παράδειγμα.

Επιπλέον μπορούμε να δούμε και γραφικά τη κατανομή :



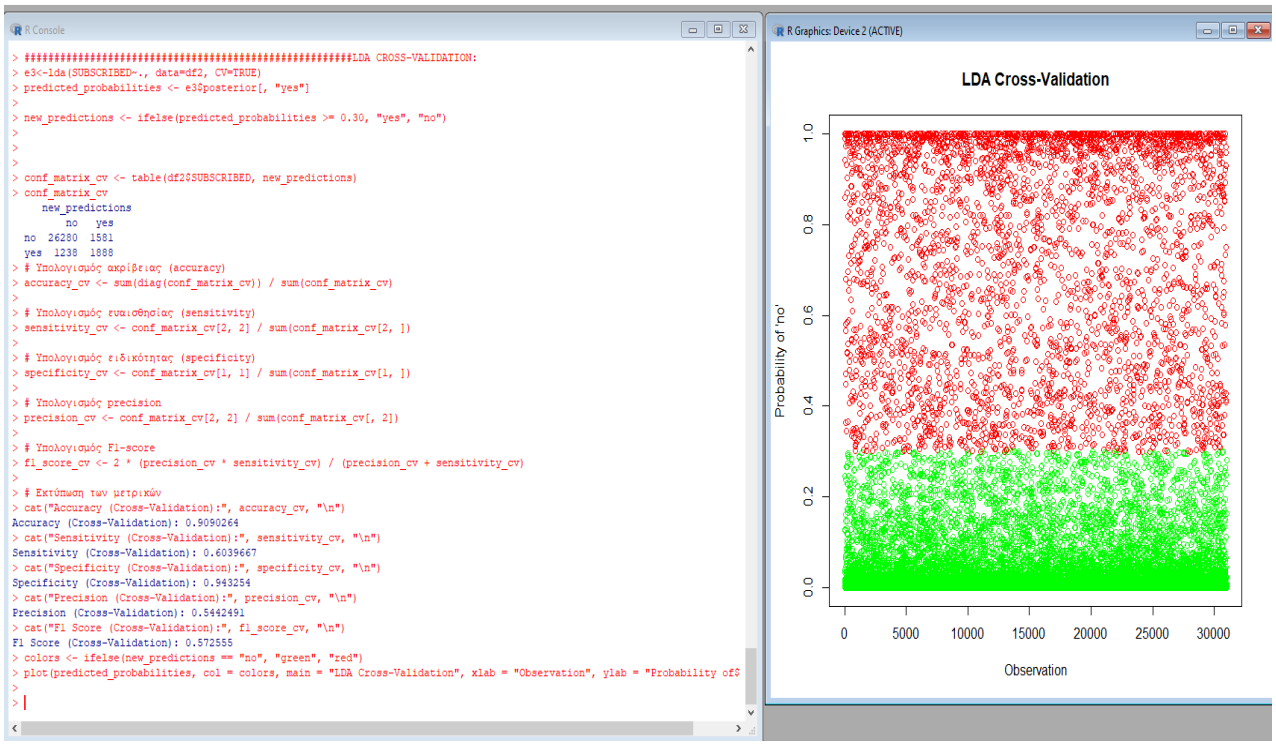
Αποκτούμε έτσι μια ιδέα για το πώς κατατάσσονται οι παρατηρήσεις μας στο μοντέλο αυτό. Τέλος εφαρμόσαμε και εδώ την ίδια διαδικασία αφαιρώντας τις μεταβλητές με ασθενή συσχέτιση και βρήκαμε πάλι παρόμοια στατιστικά ,οπότε η ιδέα το να απαλλαχθούμε από αυτές τις μεταβλητές ενισχύετε.



Συνεχίζουμε με το cross validation

LDA CROSS-VALIDATION:

Σε αυτό το σημείο θα κληθούμε να τρέξουμε LDA Cross Validation με threshold 0.3 και να μελετήσουμε τα αποτελέσματα που θα πάρουμε. Πάνω στη μέθοδο αυτή θα σταθούμε για να κατανοήσουμε καλύτερα την αποδοτικότητα της μεθόδου LDA καθώς με την επαναληπτική μέθοδο που πραγματοποιεί βγάζει πιο έγκυρα αποτελέσματα.



Τα αποτελέσματα και σε αυτήν την περίπτωση είναι αρκετά κοντινά με αυτά που βρήκαμε και στα προηγούμενα παραδείγματα και παραμένουν υψηλά οι δείκτες του accuracy και specificity και σε θετικό βαθμό οι υπόλοιπες. Την κατάταξη των παρατηρήσεων την βλέπουμε και στο plot που δημιουργήσαμε.

Τέλος κάναμε τη γνωστή σύγκριση αφαιρώντας τις μεταβλητές με ασθενή συσχέτιση και τα αποτελέσματα ήταν παρόμοια:

Penalized LDA: Αποτελεί παρακλάδι της LDA που ενσωματώνει κάποιο πέναλντι στην εκτίμηση των παραμέτρων με κύριο στόχο την αντιμετώπιση των προβλημάτων με την πολύκατηγορική κατανομή των μεταβλητών ανάκλησης και με απώτερο σκοπό την καταπολέμηση του overfitting.

Πρέπει να αναφέρουμε ότι την εφαρμογή του penalized lda την πραγματοποιήσαμε στις numeric μεταβλητές και το SUBSCRIBED

Επιπλέον βασική προϋπόθεση είναι το library(penalizedLDA)

Για αρχή θα τρέξουμε ένα ενδεικτικό μοντέλο με lambda=0.1 και θα έχουμε τη δυνατότητα με την εντολή pen$discrim να δούμε για το συγκεκριμένο lambda την συνεισφορά κάθε μεταβλητής στην διαδικασία. Αν κάποια μεταβλητή έχει μηδενική τιμή δείχνει ότι η συνεισφορά της είναι αμελητέα η μηδενική.

Ακόμα θα τρέξουμε και μια σειρά εντολών με σκοπό να βρούμε το βέλτιστο lambda που θα έπρεπε να χρησιμοποιήσουμε στη μέθοδο του penalizedLDA συγκρίνοντας τα διαφορετικά score που λαμβάνουμε για κάθε τιμή του lambda.

Παρακάτω τα screenshots:

```
R Console
>
> library(penalizedLDA)
> df1$SUBSCRIBED <- ifelse(df1$SUBSCRIBED == "yes", 1, 2)
> pen<-PenalizedLDA(df1[,-9],df1[,9],lambda=0.1,xte=df1[,-9],K=1)
> pen$discrim ### Δειχνει πως και ποσο τα χαρακτηριστικα επηρεαζουν την διαδικασια και τον τροπο με τον οποιο γινεται η δ$
            [,1]
[1,]  0.00000000
[2,] -0.68493328
[3,]  0.04184145
[4,] -0.23323218
[5,]  0.28560604
[6,] -0.05579069
[7,]  0.41186941
[8,]  0.46946631
> set.seed(121)
> index<-createDataPartition(df1$SUBSCRIBED,p=0.8,list=FALSE)
> train<- df1[index,]
> test<- df1[-index,]
> score<-NULL
> possiblelambda<- seq(0.05,0.25,by=0.01)
> for (lam in possiblelambda) {
+ pen<-PenalizedLDA(train[,-9],train[,9],lambda=lam,
+                   xte=test[,-9],K=1)
+ pen$discrim
+ pen$ypred
+ t<-table(test[,9],pen$ypred[,1])
+ score<-c(score,sum(diag(t))/sum(t))
+ }
> cbind(possiblelambda,score)    ###Βλεπουμε το βελτιστο και τιμες για ολα τα lambda και μετα κανουμε δοκιμες
      possiblelambda     score
 [1,]           0.05 0.8833871
 [2,]           0.06 0.8832258
 [3,]           0.07 0.8833871
 [4,]           0.08 0.8835484
 [5,]           0.09 0.8837097
 [6,]           0.10 0.8827419
 [7,]           0.11 0.8833871
 [8,]           0.12 0.8837097
 [9,]           0.13 0.8840323
[10,]           0.14 0.8850000
[11,]           0.15 0.8851613
[12,]           0.16 0.8859677
[13,]           0.17 0.8866129
[14,]           0.18 0.8877419
[15,]           0.19 0.8887097
[16,]           0.20 0.8890323
[17,]           0.21 0.8898387
[18,]           0.22 0.8906452
[19,]           0.23 0.8916129
[20,]           0.24 0.8927419
[21,]           0.25 0.8945161
```

Όπως μπορούμε να δούμε η βέλτιστη τιμή για το lambda είναι για lambda=0.25 και παρακάτω θα χρησιμοποιήσουμε το lambda αυτό:

```
> pen<-PenalizedLDA(df1[,-9],df1[,9],lambda=0.25,xte=df1[,-9],K=1)
> pen$discrim
            [,1]
[1,]  0.0000000
[2,] -0.7302709
[3,]  0.0000000
[4,] -0.1787995
[5,]  0.2427415
[6,]  0.0000000
[7,]  0.3968935
[8,]  0.4672122
> pen<-PenalizedLDA.cv(df1[,-9],df1[,9],lambdas=c(0.16,0.18,0.25),K=1,nfold = 10)  ###Cross validation με 10 folds για τρεις διαφορετικες τιμες lambda
Fold  1
123Fold  2
123Fold  3
123Fold  4
123Fold  5
123Fold  6
123Fold  7
123Fold  8
123Fold  9
123Fold  10
123> print(pen)
Cross-validation Results:
Values of Lambda considered:  0.16 0.18 0.25
Used only 1 value of K:  1
Mean CV Errors:  362.4 359.3 340.8
Mean Number of Nonzero Features:  7 6 5
Value of Lambda with lowest CV error:  0.25
>
```

Μπορούμε να καταλάβουμε και ποιες μεταβλητές είναι σημαντικές και δε πρέπει σε καμία περίπτωση να αφαιρεθούν όπως η duration η euribor3m και nr.employed οι οποίες βρίσκονται στις θέσεις 2,7,8 αντίστοιχα.

Σύμφωνα με το πάνω μισό επιβεβαιωνόμαστε για την υπόθεση μας για τις μεταβλητές age και campaign καθώς και εδώ παρατηρούμε ότι έχουν μηδενιστεί

(μη ξεχνάμε ότι έχουμε κρατήσει μόνο τις numeric μεταβλητές οπότε οι θέσεις των μεταβλητών άλλαξαν.) . Συνεχίζοντας πραγματοποιούμε cross-validation 10-folds βάζοντας 3 διαφορετικά lambda και παίρνουμε κοινό αποτέλεσμα όσον αφορά το βέλτιστο lambda. Αυτό φαίνεται καθώς και το error μειώθηκε και τις δύο φορές και έφτασε στο μικρότερο για lambda=0.25

RANDOM FOREST:

Το Random Forest είναι ένα σύνολο από δέντρα αποφάσεων που εκπαιδεύονται και συγκεκριμένα παίρνουμε μια πρόβλεψη από κάθε δέντρο και τις συνδυάζουμε για να δημιουργήσουμε μια γενική πρόβλεψη. Έτσι αντί να χρησιμοποιούμε ένα δέντρο υπολογίζουμε τον μέσο όρο σε μια συλλογή δένδρων.

Αρχικά χρησιμοποιούμε library(randomForest) η οποία μας επιτρέπει να χρησιμοποιήσουμε την μέθοδο random forest

Βασική προϋπόθεση για να πετύχουμε ένα καλό μοντέλο random forest αποτελεί το να βρούμε τα βέλτιστα ntree και mtry να οποία υποδηλώνουν πόσα δένδρα θα δημιουργηθούν και πόσα χαρακτηριστικά θα εξετασθούν σε κάθε διαίρεση κατά τη δημιουργία του δέντρου. Αφού βρούμε τα βέλτιστα(χαμηλότερο error) θα εφαρμόσουμε μετά random forest ξανά με αυτά σαν εισόδους και θα δούμε τα αποτελέσματα.

```
> library(randomForest)
randomForest 4.7-1.1
Type rfNews() to see new features/changes/bug fixes.

Attaching package: 'randomForest'

The following object is masked from 'package:ggplot2':

    margin

> data_rf<-df2
> oob<-NULL
> for (ntree in c(50, 100, 200, 300, 400, 500)) {
+   for (mtry in c(3, 4, 5, 6)) {
+ myRF <- randomForest(as.factor(SUBSCRIBED) ~ ., data = data_rf, ntree = ntree, mtry = mtry)
+ oob_error <- myRF$err.rate[ntree, 1]
+ cat("ntree:", ntree, ", mtry:", mtry, ", OOB Error:", oob_error, "\n")
+
+   }
+ }
ntree: 50 , mtry: 3 , OOB Error: 0.08264516
ntree: 50 , mtry: 4 , OOB Error: 0.08306452
ntree: 50 , mtry: 5 , OOB Error: 0.08329032
ntree: 50 , mtry: 6 , OOB Error: 0.08264516
ntree: 100 , mtry: 3 , OOB Error: 0.08209677
ntree: 100 , mtry: 4 , OOB Error: 0.0806129
ntree: 100 , mtry: 5 , OOB Error: 0.0813871
ntree: 100 , mtry: 6 , OOB Error: 0.08012903
ntree: 200 , mtry: 3 , OOB Error: 0.08183871
ntree: 200 , mtry: 4 , OOB Error: 0.08106452
ntree: 200 , mtry: 5 , OOB Error: 0.08129032
ntree: 200 , mtry: 6 , OOB Error: 0.08158065
ntree: 300 , mtry: 3 , OOB Error: 0.08093548
ntree: 300 , mtry: 4 , OOB Error: 0.08090323
ntree: 300 , mtry: 5 , OOB Error: 0.08070968
ntree: 300 , mtry: 6 , OOB Error: 0.08041935
ntree: 400 , mtry: 3 , OOB Error: 0.08180645
ntree: 400 , mtry: 4 , OOB Error: 0.08083871
ntree: 400 , mtry: 5 , OOB Error: 0.08064516
ntree: 400 , mtry: 6 , OOB Error: 0.08022581
ntree: 500 , mtry: 3 , OOB Error: 0.08132258
ntree: 500 , mtry: 4 , OOB Error: 0.08045161
ntree: 500 , mtry: 5 , OOB Error: 0.08041935
ntree: 500 , mtry: 6 , OOB Error: 0.081
> ###########To veltisto einai to ntree:400,mtry:6
>
```

Παρατηρούμε ‘ότι το χαμηλότερο oob error το βρίσκουμε για τον συνδυασμό ntree=400 και mtry=6 επομένως με αυτά σαν εισόδους θα λειτουργήσουμε

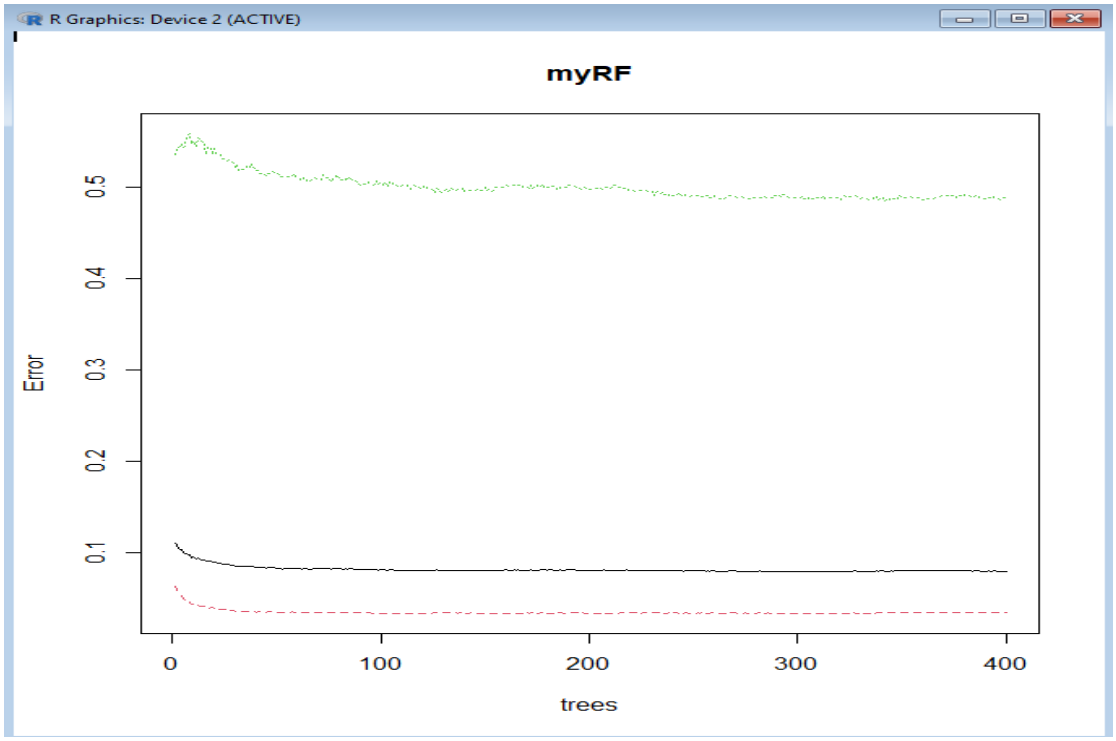
```
>
> myRF<- randomForest(as.factor(SUBSCRIBED) ~ ., data=data_rf, ntree=400,
+ mtry=6, importance=TRUE)
> myRF

Call:
 randomForest(formula = as.factor(SUBSCRIBED) ~ ., data = data_rf,      ntree = 400, mtry = 6, importance = TRUE)
               Type of random forest: classification
                     Number of trees: 400
No. of variables tried at each split: 6

        OOB estimate of  error rate: 8.1%
Confusion matrix:
       no  yes class.error
no  26892  979  0.03512612
yes  1531 1598  0.48929370
> plot(myRF)
> myRF$importance
                         no           yes MeanDecreaseAccuracy MeanDecreaseGini
age            1.888148e-03  0.0056581142         0.0022679860        568.09817
job            6.886292e-04 -0.0003476384         0.0005838695        286.41139
marital        2.547116e-04  0.0006223737         0.0002922508        134.29792
education      7.706869e-04  0.0025078774         0.0009457978        253.19903
default        1.756972e-04  0.0020859553         0.0003696173         50.90513
loan           1.023186e-05  0.0001942126         0.0000282609         88.14201
contact        2.322717e-03  0.0130684071         0.0034089066         54.31467
month          1.800888e-02  0.0090381194         0.0171072970        114.44453
day_of_week    5.436973e-03  0.0021156173         0.0051000482        234.26656
duration       2.169743e-02  0.2315851600         0.0429001265       1968.42469
campaign       4.215701e-04  0.0056700695         0.0009517911        243.85307
previous       9.206496e-04  0.0038541882         0.0012167552         66.99390
poutcome       6.363201e-04  0.0213493239         0.0027292289        154.47339
cons.price.idx 3.063624e-02  0.0059716324         0.0281347739        144.28032
cons.conf.idx  1.985214e-02  0.0135380636         0.0192138887        201.68317
euribor3m      4.172953e-02  0.0452180573         0.0420806922        662.25160
nr.employed    2.501909e-02  0.0487542750         0.0274096714        345.42294
> ################################ΣΥΤΚΡΙΝΟΥΜΕ:
> df_without2<-df2[,-c(5,6,7,12)]
> myRF<- randomForest(as.factor(SUBSCRIBED) ~ ., data=df_without2, ntree=400,mtry=6, importance=TRUE)
> myRF

Call:
 randomForest(formula = as.factor(SUBSCRIBED) ~ ., data = df_without2,      ntree = 400, mtry = 6, importance = TRUE)
               Type of random forest: classification
                     Number of trees: 400
No. of variables tried at each split: 6

        OOB estimate of  error rate: 8.2%
Confusion matrix:
       no  yes class.error
no  26852 1019   0.0365613
yes  1523 1606   0.4867370
>
```

Παραπάνω καταρχήν βλέπουμε το oob estimate of error rate= 8.1% ενώ στη συνέχεια ρίχνοντας μια ματιά στο MeanDecreaseGini μπορούμε να καταλάβουμε ποιες μεταβλητές είναι περισσότερο σημαντικές και το αντίθετο και κατά πόσο η κάθε μια συνέβαλε στη μείωση του Gini Index. Μεταβλητές όπως το default, loan , contact και previous φαίνεται να χουν μειωμένο αυτόν τον δείκτη ενώ άλλες όπως το duration εμφανίζονται αρκετά ισχυρές. Μπορούμε έτσι να αφαιρέσουμε τις λιγότερο επιδραστικές μεταβλητές πάνω στο Gini και να δουμε το αποτέλεσμα. Όπως θα δούμε και πάνω παρατηρούμε μια μικρή αύξηση στο oob error rate.

Πρέπει επίσης να ανφέρουμε ότι η μέθοδος έχει χαμηλό sensitivity κάτι που καταλαβαίνουμε από το matrix κάνοντας τις πράξεις του TP/(FN+TP)

Δε πρέπει να παραλείψουμε να αναφέρουμε ότι μεταβλητές όπως η duration η age και η euribor3m φαίνεται να επιφέρουν την μεγαλύτερη διακύμανση στις τιμές του Gini και επομένως να είναι ιδιαίτερα σημαντικές για την μέθοδο.

Error plot:



KNN:

Ο knn classifier αποτελεί έναν αλγόριθμο ο οποίος κατατάσσει κάθε νέο δείγμα σε μια κατηγορία βασιζόμενο στους k πλησιέστερους γείτονες(δηλαδή k πλησιέστερα δείγματα). Υπολογίζει την απόσταση του νέου δείγματος με τα υπόλοιπα, τις συγκρίνει και επιλέγει τα k δείγαματα με τις μικρότερες αποστάσεις και κατατάσσει την νεα εκεί.

Για αρχή θα κάνουμε χρήση της βιβλιοθήκης library(class) και πρέπει να αναφέρουμε ότι θα κάνουμε χρήση μόνο των numeric μεταβλητών και της SUBSCRIBED καθώς η knn αποτελεί μια μέθοδο στην οποία εργαζόμαστε με ευκλίδειες αποστάσεις και η χρησιμοποίηση μεταβλητών πέρα από numeric(κάνοντας κάθε μια factor) πιθανον να διαστρέβλωνε τα αποτελέσματα μας.

Ξεκινώντας πραγματοποιήσαμε κάποια απλά πειράματα με διαφορετικά k(και κάναμε knn σε scaled και μη scaled δεδομένα).

Στην συνέχεια μέσω της library(caret) και του 10fold cross validation (θα αναφέρουμε στο τέλος για την χρήση της βιβλιοθήκης) βρήκαμε το καλύτερο k για τα δεδομένα μας το οποίο μας έδινε το βέλτιστο Accuracy.

```
>
> ################################################################################KNN:
> set.seed(121)
> library(class)
> data_knn<-df2[,-c(2:9,13)]
> data_knn2<- cbind(data_knn[,9],scale(data_knn[,-9]))
> ### knn with k=3, initial data
> km3<-knn(data_knn[,-9],data_knn[,-9], cl=data_knn[,9],k=3)
> ### knn with k=3, standardize data
> km3scaled<-knn(data_knn2[,-1],data_knn2[,-1], cl=data_knn2[,1],k=3)
> table(data_knn2[,1],km3scaled)
     km3scaled
         no   yes
  no  27260   611
  yes  1089  2040
> ### knn with k=5
> km5<-knn(data_knn[,-9],data_knn[,-9], cl=data_knn[,9],k=5)
> km5scaled<-knn(data_knn2[,-1],data_knn2[,-1], cl=data_knn2[,1],k=5)
> table(data_knn2[,1],km5scaled)
     km5scaled
         no   yes
  no  27158   713
  yes  1301  1828
> ####Μπορουμε μεσω της caret να εντοπισουμε τη καλυτερη τιμη του k
> ctrl <- trainControl(method = "cv", number = 10)
> grid <- expand.grid(k = 1:10)
> knn_model <- train(data_knn2[, -1], data_knn2[, 1], method = "knn", trControl = ctrl, tuneGrid = grid)
> print(knn_model)
k-Nearest Neighbors

31000 samples
    8 predictor
    2 classes: 'no', 'yes'

No pre-processing
Resampling: Cross-Validated (10 fold)
Summary of sample sizes: 27901, 27900, 27899, 27900, 27900, 27900, ...
Resampling results across tuning parameters:

  k   Accuracy   Kappa
   1  0.8901290  0.3959572
   2  0.8915485  0.3994220
   3  0.9028712  0.4260619
   4  0.9028710  0.4235282
   5  0.9070326  0.4348223
   6  0.9057420  0.4289011
   7  0.9091616  0.4371581
   8  0.9095810  0.4398754
   9  0.9105808  0.4430327
  10  0.9101291  0.4372594

Accuracy was used to select the optimal model using the largest value.
The final value used for the model was k = 9.
```

Όπως μπορούμε να δούμε το βέλτιστο k είναι το k=9 και μας δίνει accuracy=0.9105 που είναι και το μεγαλύτερο.

Έτσι θα πάρουμε αυτήν την τιμή και θα την χρησιμοποιήσουμε για να τρέξουμε ξανά το knn αλγόριθμο.

```
> result_cv <- knn.cv(data_knn2[, -1], data_knn2[, 1], k = 9)
> true_labels <- data_knn2[, 1]
> error_rate <- sum(result_cv != true_labels) / length(true_labels)
> cat("Error Rate:", error_rate, "\n")
Error Rate: 0.08977419
> accuracy <- sum(result_cv == true_labels) / length(true_labels)
> accuracy
[1] 0.9102258
>
```

Λαμβάνουμε έτσι μετά το cross validation accuracy=0.9102

SVM MODEL:

Η επόμενη και τελευταία μέθοδος που θα χρησιμοποιήσουμε είναι το SVM(Support Vector Machines). Το SVM αποτελεί μοντέλο μηχανικής μάθησης που χρησιμοποιείται για ταξινόμηση και παλινδρόμηση. Στον τομέα της ταξινόμησης, το SVM προσπαθεί να διαχωρίσει τις κλάσεις με ένα υπερεπίπεδο, επιλέγοντας αυτό που μεγιστοποιεί το περιθώριο μεταξύ των δύο κλάσεων. Το SVM δημιουργεί ένα υπερεπίπεδο στον χώρο των χαρακτηριστικών, προσπαθώντας να διαχωρίσει τα παραδείγματα των διάφορων κλάσεων.

Πριν δούμε τα αποτελέσματα του μοντέλου μας αρχικά θα χρησιμοποιήσουμε την βιβλιοθήκη library(e1071) και ύστερα πρέπει να εξηγήσουμε την χρήση δύο υπερπαραμέτρων που θα χρησιμοποιήσουμε στην υλοποίηση της μεθόδου.

ΥΠΕΡΠΑΡΆΜΕΤΡΟΙ:

C: Η παράμετρος C (cost parameter) προσθέτει ένα σφάλμα για κάθε παρατήρηση που ταξινομούμε λάθος , αν είναι μικρό το C , η ποινή για λανθασμένες κατατάξεις είναι μικρή, οπότε επιλέγεται όριο με μεγάλο περιθώριο Μ εις βάρος μεγαλύτερου αριθμού εσφαλμένων ταξινομήσεων. Αν το C είναι μεγάλο, το Μ προσπαθεί να ελαχιστοποιήσει τον αριθμό των λανθασμένων παρατηρήσεων λόγω της υψηλής ποινής που οδηγεί σε ένα όριο απόφασης με μικρότερο Μ. Αυτή η παράμετρος επιπλέον αφορά την πολυπλοκότητα του μοντέλου. Μικρές τιμές του C οδηγούν σε πιο απλά μοντέλα, ενώ μεγαλύτερες τιμές επιτρέπουν πιο περίπλοκα.

Gamma: Μία από τις κοινώς χρησιμοποιούμενες συναρτήσεις kernel είναι η radial basis function (RBF). Η παράμετρος γάμμα του RBF ελέγχει την επιρροή ενός μόνο σημείου training. Παράλληλα η παράμετρος επηρεάζει το πόσο "ευαίσθητο" είναι το μοντέλο στα δεδομένα εκπαίδευσης. Χαμηλές τιμές του gamma δείχνουν ότι το μοντέλο θα είναι λιγότερο ευαίσθητο στις μικρές αλλαγές των δεδομένων, ενώ υψηλές τιμές του γίνονται πιο "ευαίσθητες".

Στο μοντέλο που θα υλοποιήσουμε θα πραγματοποιήσουμε cross validation με 10folds και ταυτόχρονα θα ελέγξουμε ποια τιμή των υπερπαραμέτρων είναι η βέλτιστη για το μοντέλο μας.

```
> data.SVM<-df2
> data.SVM$SUBSCRIBED <- as.factor(ifelse(data.SVM$SUBSCRIBED == "no", 0, 1))
> set.seed(121)
> param_grid <- expand.grid(C = c(0.5, 1, 3), gamma = c(0.7, 1.5, 2))
> svm_tune <- tune(svm, SUBSCRIBED ~ ., data = data.SVM, kernel = "linear",
+                  ranges = list(C = c(0.5, 1, 3), gamma = c(0.7, 1.5, 2)),
+                  tunecontrol = tune.control(sampling = "cross", cross = 10))
> svm_tune

Parameter tuning of 'svm':

- sampling method: 10-fold cross validation

- best parameters:
   C gamma
 0.5   0.7

- best performance: 0.09064516
```

Εδώ παρατηρούμε ότι οι βέλτιστες τιμές είναι το C=0.5 και Gamma=0.7 . Η χαμηλή τιμή του C (C=0.5) υποδηλώνει προτίμηση για απλά μοντέλα. Αυτό μπορεί να οδηγήσει σε απλούστερα όρια αποφάσεων, που είναι λιγότερο περίπλοκα. Ταυτόχρονα χαμηλές τιμές του γ (γ=0.7) σημαίνουν ότι το μοντέλο είναι λιγότερο ευαίσθητο στις μικρές αλλαγές στα δεδομένα εκπαίδευσης. Αυτό μπορεί να συμβάλει στη γενίκευση του μοντέλου σε νέα, προς πρόβλεψη, δεδομένα.

Τώρα θα τρέξουμε το βέλτιστο αυτό μοντέλο με τις υπερπαραμέτρους που βρήκαμε και θα βρούμε το accuracy.

```
> best_svm <- svm(SUBSCRIBED ~ ., data = data.SVM, kernel = "linear", C = 0.5, gamma = 0.7)
> predictions <- predict(best_svm, newdata = data.SVM)
>
> accuracy <- mean(predictions == data.SVM$SUBSCRIBED)
> print(paste("Accuracy:", accuracy))
[1] "Accuracy: 0.909645161290323"
>
```

Βρίσκουμε Accuracy: 0.909 το οποίο είναι θετικό.

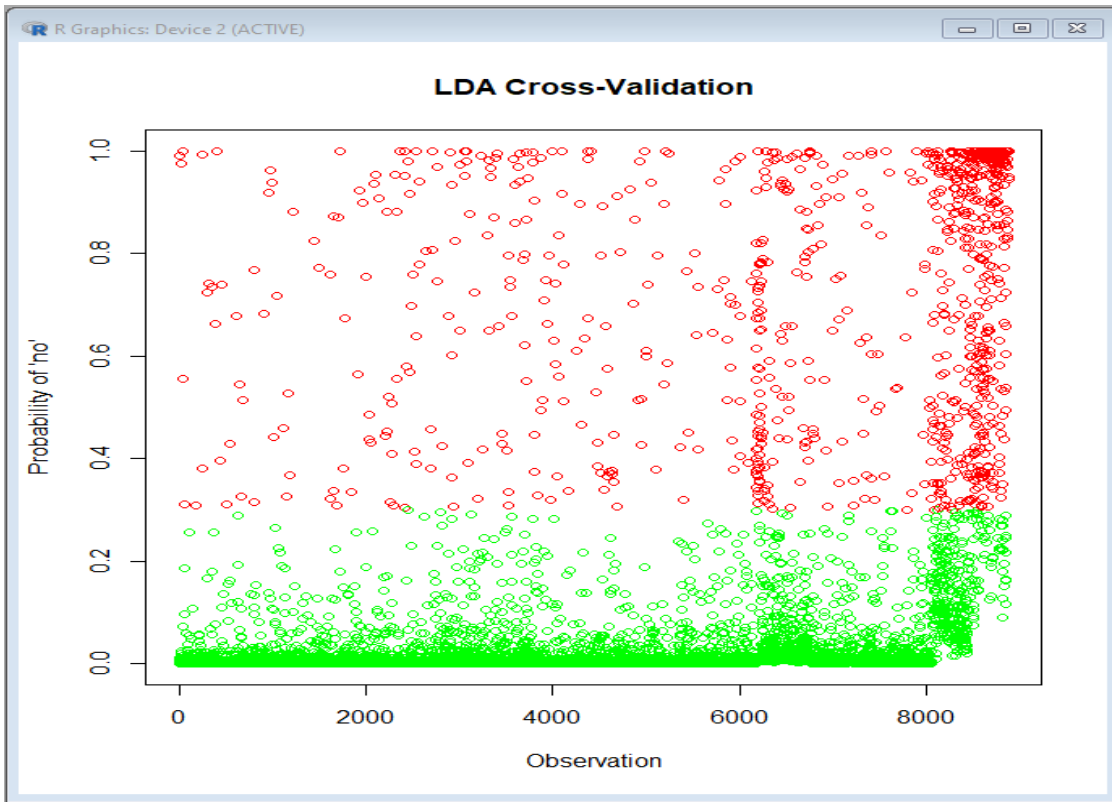
ΣΎΓΚΡΙΣΗ ΜΕΘΌΔΩΝ ΚΑΙ ΣΥΜΠΕΡΆΣΜΑΤΑ:

Τώρα ήρθε η ώρα να εφαρμόσουμε στα remaining_data τα οποία από την αρχή είχαμε διαχωρίσει και είναι σε αριθμό 8883 τα μοντέλα που δουλέψαμε παραπάνω και θα συνειδητοποιήσουμε ποιο είναι το ιδανικό μοντέλο για τα δεδομένα μας. Θα εφαρμόσουμε στα remaining_data το cross validation κάθε μεθόδου και εκεί θα λάβουμε τα πιο αντικειμενικά αποτελέσματα.

Ξεκινώντας με την LDA:

```
> dc<-remaining_data[,-c(6,13,16)]
> ##################################################################LDA C/V:
> ec<-lda(SUBSCRIBED~., data=dc, CV=TRUE)
> predicted_probabilities <- ec$posterior[, "yes"]
> new_predictions <- ifelse(predicted_probabilities >= 0.30, "yes", "no")
> conf_matrix_cv <- table(dc$SUBSCRIBED, new_predictions)
> conf_matrix_cv
     new_predictions
        no  yes
  no  7572  453
  yes  376  482
> # Υπολογισμός ακρίβειας (accuracy)
> accuracy_cv <- sum(diag(conf_matrix_cv)) / sum(conf_matrix_cv)
>
> # Υπολογισμός ευαισθησίας (sensitivity)
> sensitivity_cv <- conf_matrix_cv[2, 2] / sum(conf_matrix_cv[2, ])
>
> # Υπολογισμός ειδικότητας (specificity)
> specificity_cv <- conf_matrix_cv[1, 1] / sum(conf_matrix_cv[1, ])
>
> # Υπολογισμός precision
> precision_cv <- conf_matrix_cv[2, 2] / sum(conf_matrix_cv[, 2])
>
> # Υπολογισμός F1-score
> f1_score_cv <- 2 * (precision_cv * sensitivity_cv) / (precision_cv + sensitivity_cv)
>
> # Εκτύπωση των μετρικών
> cat("Accuracy (Cross-Validation):", accuracy_cv, "\n")
Accuracy (Cross-Validation): 0.9066757
> cat("Sensitivity (Cross-Validation):", sensitivity_cv, "\n")
Sensitivity (Cross-Validation): 0.5617716
> cat("Specificity (Cross-Validation):", specificity_cv, "\n")
Specificity (Cross-Validation): 0.9435514
> cat("Precision (Cross-Validation):", precision_cv, "\n")
Precision (Cross-Validation): 0.515508
> cat("F1 Score (Cross-Validation):", f1_score_cv, "\n")
F1 Score (Cross-Validation): 0.5376464
```

Παρατηρούμε θετικό ποσοστό accuracy παρόμοιο με αυτό που είχαμε όταν εφαρμόσαμε την lda στις 31.000 παρατηρήσεις ενώ και οι υπόλοιπες μετρικές κινούνται παρόμοια.



Επίσης μπορούμε να αφαιρέσουμε και τις μεταβλητές age και campaign καθώς οι μετρικές μένουν παρόμοιες και το accuracy συγκεκριμένα αυξάνεται όπως βλέπουμε και παρακάτω.

```
> dc2<-dc[,-c(1,11)]
> ec<-lda(SUBSCRIBED~., data=dc2, CV=TRUE)
> predicted_probabilities <- ec$posterior[, "yes"]
> new_predictions <- ifelse(predicted_probabilities >= 0.30, "yes", "no")
> conf_matrix_cv <- table(dc$SUBSCRIBED, new_predictions)
> conf_matrix_cv
     new_predictions
        no  yes
  no  7572  453
  yes  373  485
> # Υπολογισμός ακρίβειας (accuracy)
> accuracy_cv <- sum(diag(conf_matrix_cv)) / sum(conf_matrix_cv)
>
> # Υπολογισμός ευαισθησίας (sensitivity)
> sensitivity_cv <- conf_matrix_cv[2, 2] / sum(conf_matrix_cv[2, ])
>
> # Υπολογισμός ειδικότητας (specificity)
> specificity_cv <- conf_matrix_cv[1, 1] / sum(conf_matrix_cv[1, ])
>
> # Υπολογισμός precision
> precision_cv <- conf_matrix_cv[2, 2] / sum(conf_matrix_cv[, 2])
>
> # Υπολογισμός F1-score
> f1_score_cv <- 2 * (precision_cv * sensitivity_cv) / (precision_cv + sensitivity_cv)
>
> # Εκτύπωση των μετρικών
> cat("Accuracy (Cross-Validation):", accuracy_cv, "\n")
Accuracy (Cross-Validation): 0.9070134
> cat("Sensitivity (Cross-Validation):", sensitivity_cv, "\n")
Sensitivity (Cross-Validation): 0.5652681
> cat("Specificity (Cross-Validation):", specificity_cv, "\n")
Specificity (Cross-Validation): 0.9435514
> cat("Precision (Cross-Validation):", precision_cv, "\n")
Precision (Cross-Validation): 0.5170576
> cat("F1 Score (Cross-Validation):", f1_score_cv, "\n")
F1 Score (Cross-Validation): 0.5400891
>
```

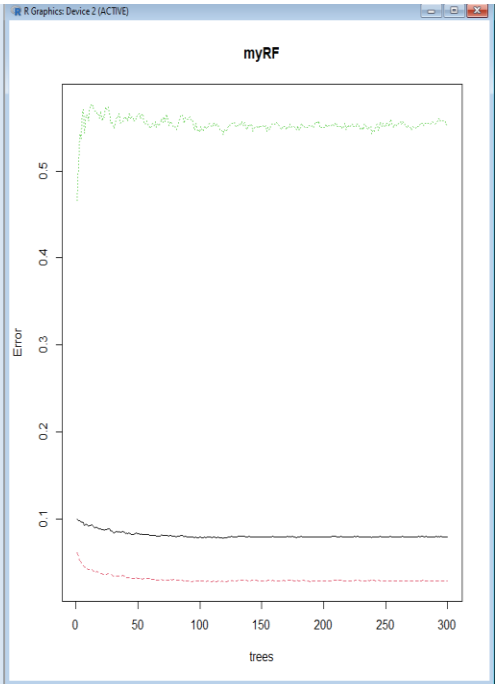
Penalized LDA: Ομοίως

```
> index<-createDataPartition(df1$SUBSCRIBED,p=0.8,list=FALSE)
> train<- df1[index,]
> test<- df1[-index,]
> score<-NULL
> possiblelambda<- seq(0.05,0.25,by=0.01)
> for (lam in possiblelambda) {
+ pen<-PenalizedLDA(train[,-9],train[,9],lambda=lam,
+                    xte=test[,-9],K=1)
+ pen$discrim
+ pen$ypred
+ t<-table(test[,9],pen$ypred[,1])
+ score<-c(score,sum(diag(t))/sum(t))
+ }
> cbind(possiblelambda,score)    ###Βλεπουμε το βελτιστο και τιμες για ολα τα lambda και μετα κανουμε δοκιμες
      possiblelambda     score
 [1,]           0.05 0.8817568
 [2,]           0.06 0.8823198
 [3,]           0.07 0.8834459
 [4,]           0.08 0.8834459
 [5,]           0.09 0.8834459
 [6,]           0.10 0.8834459
 [7,]           0.11 0.8834459
 [8,]           0.12 0.8845721
 [9,]           0.13 0.8862613
[10,]           0.14 0.8868243
[11,]           0.15 0.8879505
[12,]           0.16 0.8868243
[13,]           0.17 0.8868243
[14,]           0.18 0.8879505
[15,]           0.19 0.8896396
[16,]           0.20 0.8890766
[17,]           0.21 0.8913288
[18,]           0.22 0.8924550
[19,]           0.23 0.8935811
[20,]           0.24 0.8947072
[21,]           0.25 0.8947072
> pen<-PenalizedLDA.cv(df1[,-9],df1[,9],lambdas=c(0.16,0.18,0.25),K=1,nfold = 10)  ###Cross validation με 10 folds για τρεις$
Fold  1
123Fold  2
123Fold  3
123Fold  4
123Fold  5
123Fold  6
123Fold  7
123Fold  8
123Fold  9
123Fold  10
123> print(pen)
Cross-validation Results:
Values of Lambda considered:  0.16 0.18 0.25
Used only 1 value of K:  1
Mean CV Errors:  100 99.6 94.5
Mean Number of Nonzero Features:  7 6.4 5
Value of Lambda with lowest CV error:  0.25
>
```

Συνεχίζουμε με την Random forest:

Πραγματοποιούμε παρόμοια διαδικασία

```
> data_rf<-dc
> oob<-NULL
> for (ntree in c(50, 100, 200, 300, 400, 500)) {
+   for (mtry in c(3, 4, 5, 6)) {
+ myRF <- randomForest(as.factor(SUBSCRIBED) ~ ., data = data_rf, ntree = ntree, mtry = mtry)
+ oob_error <- myRF$err.rate[ntree, 1]
+ cat("ntree:", ntree, ", mtry:", mtry, ", OOB Error:", oob_error, "\n")
+
+   }
+ }
ntree: 50 , mtry: 3 , OOB Error: 0.08398064
ntree: 50 , mtry: 4 , OOB Error: 0.08656985
ntree: 50 , mtry: 5 , OOB Error: 0.081504
ntree: 50 , mtry: 6 , OOB Error: 0.08600698
ntree: 100 , mtry: 3 , OOB Error: 0.07835191
ntree: 100 , mtry: 4 , OOB Error: 0.0810537
ntree: 100 , mtry: 5 , OOB Error: 0.07891478
ntree: 100 , mtry: 6 , OOB Error: 0.08229202
ntree: 200 , mtry: 3 , OOB Error: 0.08139142
ntree: 200 , mtry: 4 , OOB Error: 0.08049083
ntree: 200 , mtry: 5 , OOB Error: 0.08094112
ntree: 200 , mtry: 6 , OOB Error: 0.07947765
ntree: 300 , mtry: 3 , OOB Error: 0.07992795
ntree: 300 , mtry: 4 , OOB Error: 0.08037825
ntree: 300 , mtry: 5 , OOB Error: 0.07778904
ntree: 300 , mtry: 6 , OOB Error: 0.0792525
ntree: 400 , mtry: 3 , OOB Error: 0.07868963
ntree: 400 , mtry: 4 , OOB Error: 0.07959023
ntree: 400 , mtry: 5 , OOB Error: 0.08004053
ntree: 400 , mtry: 6 , OOB Error: 0.08026568
ntree: 500 , mtry: 3 , OOB Error: 0.07902736
ntree: 500 , mtry: 4 , OOB Error: 0.07992795
ntree: 500 , mtry: 5 , OOB Error: 0.07913993
ntree: 500 , mtry: 6 , OOB Error: 0.07902736
> myRF<- randomForest(as.factor(SUBSCRIBED) ~ ., data=data_rf, ntree=300,
+ mtry=5, importance=TRUE)
> myRF

Call:
 randomForest(formula = as.factor(SUBSCRIBED) ~ ., data = data_rf,      ntree = 300, mtry = 5, importance = TRUE)
               Type of random forest: classification
                     Number of trees: 300
No. of variables tried at each split: 5

        OOB estimate of  error rate: 7.89%
Confusion matrix:
      no yes class.error
no  7798 227   0.0282866
yes  474 384   0.5524476
> plot(myRF)
```



Βρίσκουμε ως βέλτιστο ntree και mtry συνδιασμο το 300,5 και εφαρμόζουμε το RandomForest με αυτές τις τιμές βρίσκοντας ένα oob estimate error rate=7.89%

Accuracy = 1 - OOB error rate = 1 - 0.0789 = 0.9211

Αλλά και με χαμηλό όπως φαίνεται sensitivity καθώς το TP δεν φαίνεται τόσο μεγάλο σε σχέση με το FN. Sensitivity=384/(474+384)

```
> myRF$importance
                          no           yes MeanDecreaseAccuracy MeanDecreaseGini
age             0.0024760229 -0.0028881607         0.0019511968        148.09331
job             0.0011741266  0.0030239558         0.0013550645         80.85104
marital         0.0005279497  0.0018039712         0.0006551059         37.28899
education       0.0002922490  0.0040139974         0.0006488127         70.78700
default        -0.0002491968  0.0045170945         0.0002128977         16.33359
loan            0.0003026268  0.0001963189         0.0002914082         25.08648
contact         0.0025974757  0.0027853241         0.0026158323         14.12102
month           0.0181999551  0.0029035328         0.0167198887         35.94364
day_of_week     0.0037204478  0.0040524284         0.0037605975         71.00758
duration        0.0202394590  0.2086136510         0.0384387028        503.67819
campaign        0.0010607339 -0.0005539305         0.0009065039         65.40311
previous        0.0004676341  0.0033533862         0.0007527758         21.14665
poutcome        0.0005525384  0.0316139901         0.0035535387         54.79114
cons.price.idx  0.0364142750 -0.0050586013         0.0324068113         45.65245
cons.conf.idx   0.0176767969  0.0117922083         0.0171071657         65.97325
euribor3m       0.0471768512  0.0293826310         0.0454208792        183.04009
nr.employed     0.0249297125  0.0409542958         0.0264638328         93.56485
> data_rf2<-data_rf[,-c(5,6,7,12)]
> myRF<- randomForest(as.factor(SUBSCRIBED) ~ ., data=data_rf2, ntree=300,mtry=5, importance=TRUE)
> myRF

Call:
 randomForest(formula = as.factor(SUBSCRIBED) ~ ., data = data_rf2,      ntree = 300, mtry = 5, importance = TRUE)
               Type of random forest: classification
                     Number of trees: 300
No. of variables tried at each split: 5

        OOB estimate of  error rate: 8.17%
Confusion matrix:
      no yes class.error
no  7771 254  0.03165109
yes  472 386  0.55011655
> 
```

Βρίσκοντας και το importance των μεταβλητών αφαιρούμε αυτές(5,6,7,12) που είναι λιγότερο επιδραστικές πάνω στο Gini index και βρίσκουμε ένα αυξημένο oob error rate= 8.17. Σημαντικό είναι να αναφέρουμε ότι η μεταβλητή duration φαίνεται να είναι ιδιαίτερα σημαντική και κομβική και ακολουθούν άλλες όπως η euribor3m.

Accuracy = 1 - OOB error rate = 1 - 0.0817 = 0.9183

KNN:

Όπως δουλέψαμε και πάνω για τα δεδομένα μας θα βρούμε πιο αποτελεί το βέλτιστο k και με βάση αυτό θα κάνουμε knn cross validation για να λάβουμε τα αποτελέσματα.

```
> dr<-dc[,-c(2:9,13)]
> dr2<- cbind(dr[,9],scale(dr[,-9]))
> ctrl <- trainControl(method = "cv", number = 10)
> grid <- expand.grid(k = 1:10)
> knn_model <- train(dr2[, -1], dr2[, 1], method = "knn", trControl = ctrl, tuneGrid = grid)
> print(knn_model)
k-Nearest Neighbors

8883 samples
   8 predictor
   2 classes: 'no', 'yes'

No pre-processing
Resampling: Cross-Validated (10 fold)
Summary of sample sizes: 7995, 7995, 7995, 7994, 7995, 7995, ...
Resampling results across tuning parameters:

  k   Accuracy   Kappa
   1  0.8869735  0.3433041
   2  0.8904652  0.3569839
   3  0.8978953  0.3504639
   4  0.9007109  0.3589208
   5  0.9064515  0.3727507
   6  0.9062256  0.3647803
   7  0.9074630  0.3611676
   8  0.9064506  0.3544009
   9  0.9081390  0.3556104
  10  0.9078003  0.3523285

Accuracy was used to select the optimal model using the largest value.
The final value used for the model was k = 9.
> result_cv <- knn.cv(dr2[, -1], dr2[, 1], k = 9)
> true_labels <- dr2[, 1]
> error_rate <- sum(result_cv != true_labels) / length(true_labels)
> cat("Error Rate:", error_rate, "\n")
Error Rate: 0.09152313
> accuracy <- sum(result_cv == true_labels) / length(true_labels)
> accuracy
[1] 0.9084769
>
```

Λαμβάνουμε βέλτιστο k=9 και το accuracy για αυτό το k αποτελεί το

Accuracy=0.908

SVM Model:

Εφαρμόζουμε τώρα στα δεδομένα μας ένα παρόμοιο SVM model με αυτό που χρησιμοποιήσαμε και πάνω και θα δούμε τα αποτελέσματα:

```
>
> data.SVM<-dc
> data.SVM$SUBSCRIBED <- as.factor(ifelse(data.SVM$SUBSCRIBED == "no", 0, 1))
> set.seed(121)
> param_grid <- expand.grid(C = c(0.5, 1, 3), gamma = c(0.7, 1.5, 2))
> svm_tune <- tune(svm, SUBSCRIBED~ ., data = data.SVM, kernel = "linear",
+                  ranges = list(C = c(0.5, 1, 3), gamma = c(0.7, 1.5, 2)),
+                  tunecontrol = tune.control(sampling = "cross", cross = 10))
> svm_tune

Parameter tuning of 'svm':

- sampling method: 10-fold cross validation

- best parameters:
   C gamma
 0.5   0.7

- best performance: 0.09006159

> best_svm <- svm(SUBSCRIBED ~ ., data = data.SVM, kernel = "linear", C = 0.5, gamma = 0.7)
> predictions <- predict(best_svm, newdata = data.SVM)
> accuracy <- mean(predictions == data.SVM$SUBSCRIBED)
> print(paste("Accuracy:", accuracy))
[1] "Accuracy: 0.91005291005291"
>
```

Βρίσκουμε ίδιες υπερπαραμέτρους όπως και όταν ασχοληθήκαμε με τις άλλες 31.000 παρατηρήσεις και εφαρμόζοντας 10fold cross validation βρίσκουμε γι αυτές τις υπερπαραμέτρους Accuracy= 0.910 και error = 0.0900 αντίστοιχα, κινείται δηλαδή σε κοινό άξονα με τα υπόλοιπα.

<u>Αξιολόγηση και επιλογή μοντέλων:</u>

Παρατηρώντας τα αποτελέσματα από όλα τα μοντέλα ήρθε η ώρα να κατασταλάξουμε για το πιο είναι το βέλτιστο μοντέλο και εξυπηρετεί καλύτερα τα δεδομένα μας και θα μας βοηθήσει περισσότερο στο να προβλέψουμε με μεγαλύτερη ακρίβεια το αν ένας πελάτης θα κάνει SUBSCRIBE ή όχι για το προιόν . Σαν μέτρο θα λάβουμε τις μετρικές τις οποίες βρήκαμε κάθε φορά σε κάθε μοντέλο με σκοπό να καταλήξουμε στο καλύτερο. Παρόλα αυτά μπορούμε να προσεγγίσουμε τα αποτελέσματα με διάφορους τρόπους. Αν επιθυμούσαμε να έχουμε την μεγαλύτερη δυνατή ακρίβεια-Accuracy τότε θα μπορούσαμε να επιλέξουμε την μέθοδο RandomForest η οποία μας προσφέρει accuracy=0.9211 και είναι μεγαλύτερο από τα υπόλοιπα accuracy των άλλων μεθόδων, παράλληλα παρατηρούμε ότι κάθε συνδυασμός ntree και mtry ακόμα και ας μην ήταν ο βέλτιστος μας έβγαζε accuracy μεγαλύτερο του 0.910 το οποίο μας δείχνει πόσο καλά προσαρμόζεται η μέθοδος του Random forest στα δεδομένα μας. Δε πρέπει όμως να παραλείψουμε ότι το sensitivity είναι σχετικά μικρό σε αυτήν την

περίπτωση και φαίνεται ότι επηρεάζεται από το μεγάλο πλήθος των τιμών «no» της μεταβλητής που επιδιώκουμε να προβλέψουμε. Αν επιλέγαμε την μέθοδο αυτή θα αποφασίζαμε να διώξουμε τις μεταβλητές default, loan , contact και previous καθώς από το importance που βρήκαμε φαίνεται ότι δεν είναι επιδραστικές στο Gini index και κατά συνέπεια στο ίδιο το μοντέλο. Μια άλλη προσέγγιση θα ήταν να εμπιστευτούμε την LDA μέθοδο η οποία μας παρέχει και αυτή αρκετά υψηλό Accuracy και ταυτόχρονα εκμεταλλευόμενοι το threshold θα μπορούσαμε να έχουμε καλύτερα αποτελέσματα για μετρικές όπως το sensitivity(μειώνοντας το κατώφλι). Ταυτόχρονα μέσω και της penalizedLDA έχουμε την ευκαιρία να είμαστε πιο βέβαιοι για τις μεταβλητές που πρέπει να αφαιρέσουμε και θα αποκτήσουμε καλύτερη άποψη πάνω στα δεδομένα. Ταυτόχρονα θα καταπολεμήσουμε και το overfitting οπότε θα είμαστε και πιο βέβαιοι για τις προβλέψεις και την εγκυρότητα των αποτελεσμάτων μας. Αν αποφασίζαμε να διαλέξουμε την LDA ως την ιδανική μέθοδο θα μπορούσαμε να διώξουμε τις μεταβλητές age και campaign οι οποίες αρχικά είχαν χαμηλή συσχέτιση με την μεταβλητή SUBSCRIBED και καθόλα την διάρκεια τόσο των LDA δοκιμών όσο και των penalizedLDA δοκιμών φάνηκαν να είναι αδύναμες και αμελητέες στην πρόβλεψη τον αποτελεσμάτων και όταν τις αφαιρούσαμε οι αλλαγές στις μετρικές και στα στατιστικά στοιχεία ήταν πολύ μικρές και σε μερικά παραδείγματα θετικές. Τις μεθόδους SVM και KNN δε θα επιλέγαμε να χρησιμοποιήσουμε σαν βέλτιστες την κάθε μια για διαφορετικούς λόγους. Από την μια η KNN μέθοδος θα μπορούσε να μας προβληματίσει εξαρχής καθώς για οποιαδήποτε τιμή του k μικρότερη του 9 παρουσίαζε το χαμηλότερο accuracy από τις υπόλοιπες μεθόδους που εφαρμόσαμε, ενώ παράλληλα ο βέλτιστος αριθμός κοντινότερων γειτόνων αναδείχθηκε ο k=9 κάτι που κάνει το μοντέλο μας αρκετά απλό και αφαιρεί την πολυπλοκότητα του, λόγο του σχετικά μεγάλου αριθμού του k. Από την άλλη στην υλοποίηση της SVM μεθόδου είδαμε ότι βέλτιστες τιμές για τις υπερπαραμέτρους C και gamma αποτέλεσαν οι μικρότερες τιμές που δώσαμε (0,5 και 0,7) κάτι που κάνει το μοντέλο μας πιο απλό και λιγότερο ευαίσθητο στις αλλαγές , συνεπώς υπάρχει ο φόβος το accuracy που βρίσκουμε να επηρεάζεται από τις υπερπαραμέτρους και η μέθοδος να μην είναι τόσο αντιπροσωπευτική. Συμπερασματικά, αν καλούμασταν να αποφασίσουμε για το ποία μέθοδο θα διαλέγαμε τότε ανάλογα με το τι θέλουμε να επιτύχουμε θα διαλέγαμε ανάμεσα στην RandomForest και στην LDA/PenalizedLDA. Αν γνώμονας μας για την επιλογή του βέλτιστου μοντέλου αποτελεί το υψηλότερο δυνατό accuracy θα προτιμούσαμε την Randomforest και θα αφαιρούσαμε τις μεταβλητές (default, loan, Contact,previous) για λόγους που έχουμε ήδη αναφέρει, ενώ αν θέλαμε ένα πιο ευέλικτο μοντέλο με περιθώρια αλλαγής, αύξησης και μείωσης των μετρικών στοιχείων ανάλογα με την επιθυμία του αντίστοιχου χρήστη και με βελτίωση του sensitivity ,θα τείναμε προς την LDA και θα προτιμούσαμε να απαλλαχθούμε από μεταβλητές όπως η age, campaign.Είναι αναγκαίο να αναφέρουμε ότι πάντα η επιλογή του κάθε μοντέλου διαφέρει με τους στόχους τις

φιλοδοξίες και τις διαφορετικές επιθυμίες του κάθε χρήστη και η αξιολόγηση μπορεί να διαφέρει.

ΣΗΜΑΝΤΙΚΕΣ ΜΕΤΑΒΛΗΤΕΣ:

Τέλος παρατηρώντας τις μεταβλητές καθόλα την διάρκεια των ελέγχων μας μπορούμε να πούμε με βεβαιότητα ότι η duration ήταν ιδιαίτερα σημαντική και κομβική σε όλες τις μεθόδους και κυρίως σε αυτές που ξεχωρίσαμε(RF ,LDA), ενώ ακολουθούν και οι μεταβλητές euribor3m και η nr.employed(Την δύναμη και την σημασία των μεταβλητών αυτών μπορούσαμε να την προβλέψουμε και από την μεγάλη συσχέτιση που εμφάνιζαν στο correlation matrix). Τόσο μέσω του importance από το random forest όσο και με την βοήθεια του penalizedLDA βγάλαμε τα συμπεράσματα μας για το πόσο καίριες ήταν οι μεταβλητές αυτές.

Επιπλέον tools που χρησιμοποιήσαμε:

Βιβλιοθήκη caret χρησιμοποιήθηκε στον διαχωρισμό των δεδομένων σε train και test στο createDataPartition καθώς και στο SVM για την υλοποίησή του.

Η βιβλιοθήκη "caret" (Classification And REgression Training) είναι μια R βιβλιοθήκη που παρέχει ένα ενιαίο πλαίσιο για την εκπαίδευση και αξιολόγηση μοντέλων μηχανικής μάθησης. Συγκεντρώνει πολλούς αλγορίθμους μάθησης, εργαλεία προεπεξεργασίας δεδομένων και διαδικασίες αξιολόγησης, προσφέροντας ένα ενιαίο περιβάλλον για την ανάπτυξη και τη βελτιστοποίηση μοντέλων μηχανικής μάθησης στη γλώσσα προγραμματισμού R.

Επιπλέον πληροφορίες :

https://cran.r-project.org/web/packages/caret/vignettes/caret.html