Ατομική Διπλωματική Εργασία

# ΜΕΛΕΤΗ ΤΗΣ ΔΙΑΔΟΣΗΣ ΤΩΝ ΒΙΝΤΕΟ ΤΟΥ YOUTUBE ΣΤΟ ΜΕΣΟ ΚΟΙΝΩΝΙΚΗΣ ΔΙΚΤΥΩΣΗΣ TWITTER

**Πουής Λούκας**

**ΠΑΝΕΠΙΣΤΗΜΙΟ ΚΥΠΡΟΥ**

**ΤΜΗΜΑ ΠΛΗΡΟΦΟΡΙΚΗΣ**

**Μάιος 2016**

# ΠΑΝΕΠΙΣΤΗΜΙΟ ΚΥΠΡΟΥ

## ΤΜΗΜΑ ΠΛΗΡΟΦΟΡΙΚΗΣ


Μελέτη της διάδοσης των βίντεο του YouTube στο Μέσο Κοινωνικής Δικτύωσης Twitter


Λούκας Πουής

Επιβλέπων Καθηγητής
Πάλλης Γιώργος


Η Ατομική Διπλωματική Εργασία υποβλήθηκε προς μερική εκπλήρωση των απαιτήσεων απόκτησης του πτυχίου Πληροφορικής του Τμήματος Πληροφορικής του Πανεπιστημίου Κύπρου

Μάιος 2016

## Ευχαριστίες

Θα ήθελα να ευχαριστήσω τον επιβλέποντα καθηγητή μου Δρ. Γιώργο Πάλλη, αλλά και τον Δρ. Χρύση Γεωργίου, για την συνεχή καθοδήγηση, κατανόηση και βοήθεια που μου προσέφεραν κατά την διάρκεια αυτής της εργασίας. Επίσης θα ήθελα να τους ευχαριστήσω για την εμπιστοσύνη τους προς το πρόσωπό μου και για την ευκαιρία που μου έδωσαν να συνεργαστώ μαζί τους.

Θα ήθελα ακόμα να ευχαριστήσω το Στέφανο Αντάρη και το Νικόλα Λουλλούδη για την βοήθεια και την καθοδήγηση που μου έδωσαν για την χρήση των τεχνολογιών που χρησιμοποίησα.

Ένα ιδιαίτερο ευχαριστώ θα ήθελα να πω στο Χάρη Ευσταθιάδη, ο οποίος μου παρείχε όλα τα δεδομένα από το Twitter που συνέλεξε, για να τα αναλύσω στην διπλωματική μου.

Τέλος, θα ήθελα να πω ένα μεγάλο ευχαριστώ στους φίλους και στην οικογένεια μου για την συμπαράσταση και στήριξη που μου έδειξαν κατά τη διάρκεια αυτής της εργασίας.

# Περίληψη

Σε αυτή τη εποχή που ζούμε, τα μέσα κοινωνικής δικτύωσης έχουν γίνει μέρος της ζωής του κάθε ανθρώπου. Ο καθένας από εμάς μοιράζεται σε αυτά τις σκέψεις και τις δραστηριότητες του μέσα από κείμενο ή μέσα από φωτογραφίες και βίντεο που ανεβάζει, με σκοπό να μεταδώσει αυτές τις πληροφορίες στους φίλους του ή στους γνωστούς του.

Λογικό επακόλουθο αυτής της δραστηριότητας είναι η συσσώρευση τεράστιου όγκου δεδομένων καθημερινά στα κοινωνικά δίκτυα. Και όσον αφορά περιεχόμενο κειμένου ή φωτογραφίες, τα μεγέθη είναι αμελητέα. Όταν όμως έχει να κάνει με βίντεο (ειδικά υψηλής ευκρίνειας), τότε έχουμε να αντιμετωπίσουμε μεγάλα μεγέθη σε σχέση με τα άλλα περιεχόμενα.

Στόχος αυτής της διπλωματικής είναι η μελέτη των πληροφοριών που παρέχονται από τα κοινωνικά δίκτυα, έτσι ώστε να είμαστε σε θέση να κατανοήσουμε ποια είναι τα χαρακτηριστικά αυτά που επηρεάζουν την διάδοση των βίντεο σε αυτά τα δίκτυα. Πιο συγκεκριμένα, θα μελετήσουμε το κοινωνικό δίκτυο Twitter (ένα από τα πιο δημοφιλή κοινωνικά δίκτυα στο πλανήτη) και τα βίντεο από το YouTube (την δημοφιλέστερη πλατφόρμα διανομής βίντεο) που οι χρήστες ανεβάζουν σε αυτό.

# Περιεχόμενα

## Λίστα Σχημάτων

## Λίστα Πινάκων

# Κεφάλαιο 1

## Εισαγωγή



### 1.1 Κίνητρα

Στη σημερινή εποχή τα Μέσα Κοινωνικής Δικτύωσης έχουν επικρατήσει στις προτιμήσεις των διαδικτυακών χρηστών όσο αφορά την ψυχαγωγία, αλλά και την ενημέρωση τους. Λογική συνέπεια αυτού του γεγονότος είναι η διανομή τεράστιου όγκου δεδομένων καθημερινά μέσα από τα κοινωνικά δίκτυα.

Το Facebook είναι ένα κοινωνικό δίκτυο που δημιουργήθηκε το 2004. Με βάση στατιστικά του 2015 [1] έχει παραπάνω από 1.59 δισεκατομμύρια μηνιαίους ενεργούς χρήστες. Κατά μέσο όρο, ένας χρήστης περνά 20 λεπτά τη μέρα στο Facebook. Κάθε μέρα ανεβαίνουν 300 εκατομμύρια νέες φωτογραφίες. Κάθε λεπτό στο Facebook ανεβαίνουν κατά μέσο όρο 510 νέα σχόλια, 293 χιλιάδες νέα status και 136 χιλιάδες νέες φωτογραφίες.

Το Twitter, ένα άλλο κοινωνικό δίκτυο, δημιουργήθηκε το 2006 και έγινε παγκόσμια δημοφιλές με περισσότερους από 100 εκατομμύρια χρήστες και 340 εκατομμύρια tweets την ημέρα (βάση στατιστικών του 2012). Η υπηρεσία αυτή δεχόταν 1.6 δισεκατομμύρια search queries τη μέρα. Το 2013, το Twitter ήταν μια από τις πιο δημοφιλής σελίδες. Το Μάη του 2015, το Twitter είχε περισσότερους από 500 εκατομμύρια χρήστες, από τους οποίους περισσότεροι από 332 εκατομμύρια ήταν ενεργοί χρήστες [2].

Παρατηρώντας λοιπόν αυτά τα στατιστικά στοιχεία, μπορούμε να αντιληφθούμε την σημασία που έχει η αποδοτική διαχείριση όλων αυτών των δεδομένων. Η ύπαρξη αποδοτικών μηχανισμών για διαχείριση του περιεχομένου στα κοινωνικά δίκτυα έχει γίνει αδήριτη ανάγκη. Και αν εξαιρέσουμε τα περιεχόμενα κειμένου ή φωτογραφιών που διακινούνται λόγω του σχετικά μικρού μεγέθους που έχουν, αλλά και του υψηλού bandwidth που έχουμε σήμερα, δεν μπορούμε να κάνουμε το ίδιο και με τα βίντεο, τα οποία έχουν σαφώς μεγαλύτερο μέγεθος.

Σύμφωνα με μια μελέτη [3], οι χρήστες στα κοινωνικά δίκτυα είναι αυτοί που επιλέγουν το τι θα δουν και επηρεάζονται σε μεγάλο βαθμό από την αλληλεπίδραση με το κοινωνικό δίκτυο. Ο παραλήπτης ενός περιεχομένου μπορεί να το προωθήσει ή και όχι, ανάλογα με το κοινωνικό δίκτυο στο οποίο δημιουργήθηκε και μεταδόθηκε το συγκεκριμένο περιεχόμενο, και όχι μόνο ανάλογα με το πόσο ελκυστικό είναι. Για παράδειγμα, μια μελέτη [4] στο Twitter’s Vine (μια πολύ δημοφιλής μικρή κινητή υπηρεσία ανταλλαγής βίντεο) έδειξε ότι το πόσο δημοφιλές θα γίνει ένα βίντεο εξαρτάται λιγότερο από το περιεχόμενο του βίντεο και περισσότερο από το ποιος το μετέδωσε.

Επομένως, τα κοινωνικά δίκτυα μπορούν να αποτελέσουν μια πολύ σημαντική πηγή πληροφοριών, όπου αναλύοντας την συμπεριφορά των χρηστών, μπορούμε να εξάγουμε χρήσιμα συμπεράσματα για το πότε κάποιος θα ενδιαφερθεί για ένα περιεχόμενο.

Η βασική υπόθεση στην οποία βασίζεται αυτή η μελέτη, είναι ότι ο συνδυασμός δύο κύριων ειδών καναλιών διάχυσης περιεχομένου (diffusion channels) επηρεάζει το πόσο δημοφιλές θα γίνει το περιεχόμενο. Αυτά είναι τα συμβατικά κανάλια διάχυσης περιεχομένου (conventional diffusion channels) και τα κανάλια κοινωνικών δικτύων (online social networking channels). Τα συμβατικά κανάλια διάχυσης περιεχομένου είναι δίκτυα μαζικής ενημέρωσης, όπως για παράδειγμα νέες σελίδες, blogs, κριτικές για προϊόντα, τα οποία μπορούν να θεωρηθούν και ως άμεση προσπάθεια μάρκετινγκ, σχεδιασμένα για να κατευθύνουν την προσοχή στα βίντεο. Αντιθέτως, τα κανάλια κοινωνικών δικτύων χρησιμοποιούν τις ομάδες χρηστών για να παράξουν τη διάδοση των περιεχομένων, αφού οι χρήστες αξιοποιούν τους κοινωνικούς τους κύκλους για τη μετάδοση του περιεχομένου με τους διαδικτυακούς τους φίλους ή ακολούθους. Αυτό που οδηγεί τους χρήστες να μεταδώσουν ένα περιεχόμενο παραμένει υπό μελέτη [5], αλλά

εμπειρικά στοιχεία έχουν δείξει ότι κάποια περιεχόμενα (τα οποία αναφέρονται ως viral) δημιουργούν αρκετό ενδιαφέρον σε ένα χρήστη για να αναμεταδώσει το περιεχόμενο που δημοσίευσαν οι φίλοι του στο κοινωνικό δίκτυο.

Αξιοποιώντας τις πληροφορίες που μας παρέχουν τα κοινωνικά δίκτυα, όπως τη σχέση μεταξύ των χρηστών, την κοινωνική τους δραστηριότητα, την τοποθεσία και την ζώνη ώρας (time zone) που έχουν, μπορεί να γίνει πρόβλεψη για τα περιεχόμενα που οι χρήστες θα ζητήσουν. Έχοντας ως γνώση αυτή την πρόβλεψη, μπορούμε να τη χρησιμοποιήσουμε για να προ-ανακαλέσουμε το περιεχόμενο αυτό κοντά στους χρήστες που θα το ζητήσουν, αυξάνοντας την αποδοτικότητα του δικτύου και μειώνοντας το χρόνο απόκρισης και την συμφόρηση στους εξυπηρετητές που έχουν αποθηκευμένο το περιεχόμενο. Αυτό θα είναι ακόμα πιο σημαντικό για περιεχόμενα που παρουσιάζουν το φαινόμενο του “long tail” [6] (δηλαδή περιεχόμενα τα οποία δεν βρίσκονται σε μεγάλη ζήτηση ή δεν ενδιαφέρουν το κοινό τη συγκεκριμένη χρονική περίοδο, αλλά επειδή συνολικά είναι πάρα πολλά δημιουργούν μια μεγάλη ουρά από δεδομένα στο παγκόσμιο ιστό) και ξαφνικά μπορούν να γίνουν πολύ δημοφιλή ανάμεσα στους χρήστες του κοινωνικού δικτύου.

### 1.2 Στόχοι

Αυτή η διπλωματική εργασία έχει ως στόχο να μελετήσει τη δημοτικότητας ενός περιεχομένου σε μια υπηρεσία διάδοσης βίντεο και ταυτόχρονα τη διάδοση του περιεχομένου μέσα σε ένα κοινωνικό δίκτυο. Πιο συγκεκριμένα, ο στόχος είναι να παρατηρήσουμε τις σχέσεις μεταξύ της διάδοσης ενός περιεχομένου βίντεο στο κοινωνικό δίκτυο Twitter και της δημοσιότητας του βίντεο στο YouTube, της δημοφιλέστερης πλατφόρμας διανομής βίντεο. Ο λόγος που επιλέξαμε τη μελέτη των περιεχομένων που αφορούν βίντεο είναι γιατί, σε αντίθεση με τα άλλα περιεχόμενα κειμένου και φωτογραφιών, τα βίντεο έχουν πολύ μεγαλύτερο μέγεθος και κατ' επέκταση απαιτούν πολύ υψηλότερο bandwidth.

Αξιολογώντας την αλληλεπίδραση μεταξύ της δημοσιότητας (popularity) ενός βίντεο στο YouTube (δηλαδή της τάσης να ελκύει τους χρήστες να δουν το συγκεκριμένο περιεχόμενο) και της διάδοσης (virality) του βίντεο στο Twitter (δηλαδή της ενδεχόμενης αναμετάδοσης του συγκεκριμένου βίντεο από πολλούς χρήστες) και χρησιμοποιώντας τα

δεδομένα που παρέχονται από το Twitter και το YouTube που αφορούν το συγκεκριμένο βίντεο και τη διάδοσή του, στοχεύουμε να είμαστε σε θέση να εξάγουμε χρήσιμα συμπεράσματα για τα χαρακτηριστικά τα οποία επηρεάζουν σημαντικά το πόσο δημοφιλές θα γίνει το περιεχόμενο αυτό μέσα στο κοινωνικό δίκτυο.

Αυτή η πληροφορία θα είναι πολύ χρήσιμη στους Παροχείς Υπηρεσιών Διαδικτύου (Internet Service Providers), έτσι ώστε να την χρησιμοποιήσουν για να κάνουν πιο αποδοτική την διαχείριση των περιεχομένων, αφού θα μπορούν να αξιοποιήσουν αυτά τα συμπεράσματα για να προβλέπουν από πριν ποιες ομάδες χρηστών θα ενδιαφερθούν για ένα συγκεκριμένο περιεχόμενο, αλλά και να διαχειριστούν αποτελεσματικότερα τους πόρους τους.

### 1.3 Μεθοδολογία

Αρχικά έπρεπε να γίνει μελέτη της σχετικής βιβλιογραφίας για να συλλέξουμε γνώσεις για τα κοινωνικά δίκτυα και τη διάδοση των περιεχομένων σε αυτά. Μετά που κατανοήσαμε κάποιες βασικές γνώσεις για αυτό τον τομέα, έπρεπε να μελετήσουμε τις μεθοδολογίες και τα υπολογιστικά μοντέλα που χρησιμοποιούνται για τέτοιου είδους προβλήματα. Αφού αποκτήσαμε μια σφαιρική γνώση για τα μοντέλα αυτά, έπρεπε να γίνει εξοικείωση με τα υπολογιστικά μοντέλα που θα χρησιμοποιούσαμε στη δική μας μελέτη.

Ακολούθως έγινε η συλλογή των δεδομένων από το Twitter, στα οποία έπρεπε να γίνει κάποια προ-επεξεργασία. Πιο συγκεκριμένα έπρεπε να βρούμε ποια tweets περιείχαν σύνδεσμο σε μορφή σμίκρυνσης, ενώ στη συνέχεια έπρεπε να γίνει ανάκτηση των πραγματικών συνδέσμων για να κρατήσουμε μόνο τους συνδέσμους προς το YouTube. Χρησιμοποιώντας τους συνδέσμους προς το YouTube για τη εξαγωγή των χαρακτηριστικών των βίντεο, ολοκληρώσαμε το σύνολο δεδομένων μας με τα χαρακτηριστικά από το Twitter και το YouTube.

Έχοντας το ολοκληρωμένο σύνολο δεδομένων, προχωρήσαμε με την μεθοδολογία μας και εκπαιδεύσαμε ένα Gradient Boosting Classifier για να μπορεί να προβλέπει την διάδοση του βίντεο. Χρησιμοποιώντας τις προβλέψεις του classifier, μπορέσαμε να

βρούμε τις μοναδικές ιδιότητες του περιεχομένου, αλλά και το πώς τα χαρακτηριστικά επηρεάζουν τη διάδοση του, τόσο στο Twitter όσο και στο YouTube.

Τέλος, παρουσιάζουμε τα γενικά συμπεράσματα αλλά και την μελλοντική εργασία που μπορεί να γίνει σε αυτό το θέμα.

### 1.4 Οργάνωση Διπλωματικής Εργασίας

Στο Κεφάλαιο 2 δίνεται το υπόβαθρο, δηλαδή περιγράφονται τα διάφορα μοντέλα και τεχνολογίες που χρησιμοποιήθηκαν για το κομμάτι της υλοποίησης της διπλωματικής εργασίας. Ακόμα περιγράφονται βασικές έννοιες που αφορούν τη μελέτη αυτή, καθώς και προηγούμενη μελέτη και σχετική βιβλιογραφία με το θέμα μας.

Στο Κεφάλαιο 3 γίνεται λεπτομερής επεξήγηση της μεθοδολογίας συλλογής και επεξεργασίας των δεδομένων που χρησιμοποιήθηκαν.

Στο Κεφάλαιο 4 αναγνωρίζονται τα βασικά χαρακτηριστικά της διάδοσης του περιεχομένου, καθώς και οι παρατηρήσεις και τα αποτελέσματα που προέκυψαν από την ανάλυση.

Τέλος, στο Κεφάλαιο 5 παρουσιάζονται τα γενικά συμπεράσματα και προτείνονται ορισμένα θέματα μελλοντικής εργασίας που μπορούν να γίνουν σχετικά με την συγκεκριμένη μελέτη.

# Κεφάλαιο 2

## Υπόβαθρο και Συναφής Βιβλιογραφία



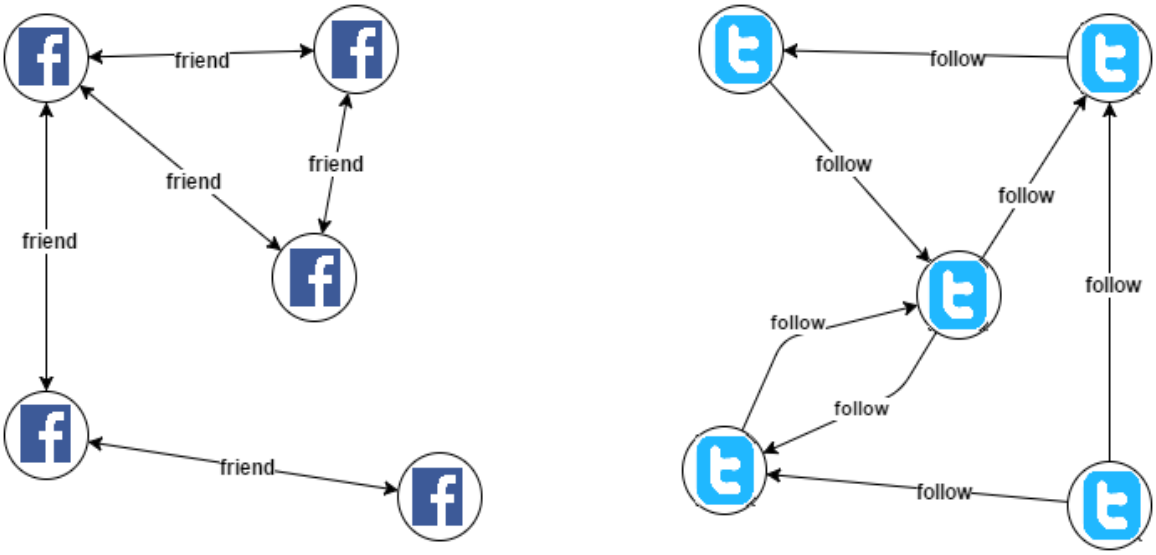
### 2.1 Κοινωνικά Δίκτυα

Τα κοινωνικά δίκτυα είναι ιστοσελίδες οι οποίες επιτρέπουν την αλληλεπίδραση μεταξύ των χρηστών. Οι χρήστες επικοινωνούν μεταξύ τους και μπορούν να ανταλλάξουν πληροφορίες και περιεχόμενα όπως σχόλια, φωτογραφίες και βίντεο. Οι σελίδες αυτές μπορούν να χαρακτηριστούν και σαν παγκόσμιες εικονικές κοινότητες. Τα μεγαλύτερα και πιο διαδεδομένα κοινωνικά δίκτυα στις μέρες μας (Απρίλης 2016) είναι το Facebook και το Twitter [7].

Στο Facebook υπάρχει η έννοια της κοινωνικής ροής (social feed), όπου ο κάθε χρήστης μπορεί να δημοσιεύσει οποιοδήποτε περιεχόμενο επιθυμεί, π.χ. σχόλια, φωτογραφίες, βίντεο κτλ. Ακόμη, μπορεί να δει όλα τα περιεχόμενα που έχουν δημοσιεύσει οι φίλοι του ή και το τι τους αρέσει. Στο Twitter οι χρήστες είναι πιο περιορισμένοι, αφού μπορούν να δημοσιεύσουν μόνο τα tweets, τα οποία έχουν μέγιστο μέγεθος 140 χαρακτήρες και με αυτά μπορούν να ενημερώσουν όσους τους ακολουθούν.

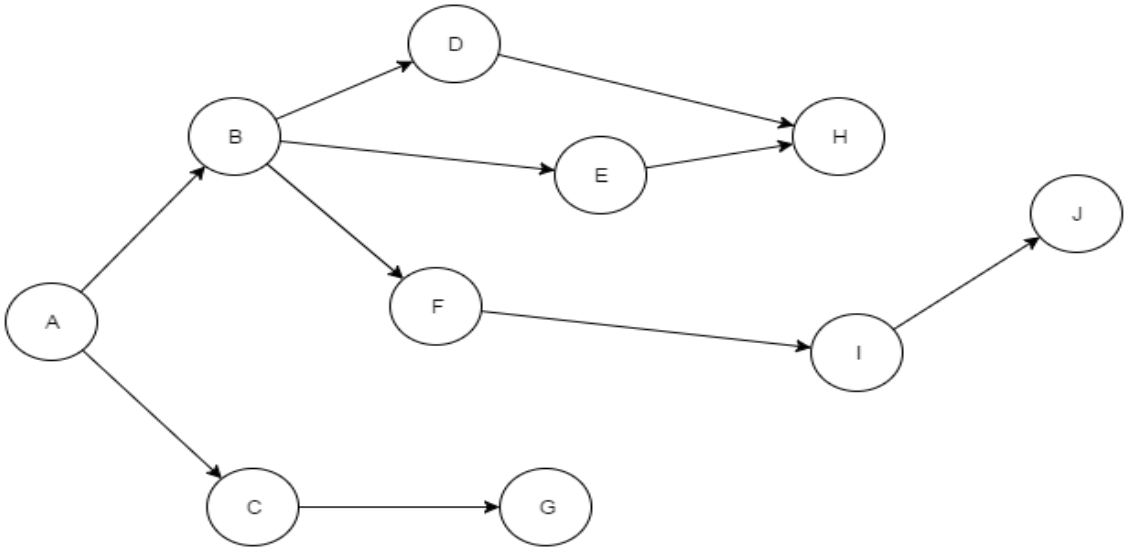
Τα δύο αυτά κοινωνικά δίκτυα έχουν ως κύριο μέλημα την επικοινωνία μεταξύ των χρηστών. Η μεγάλη τους διαφορά είναι ότι στο Facebook υπάρχει η έννοια του φίλου (friend), ενώ στο Twitter η έννοια του ακόλουθου (follower). Στο Facebook η σχέση

μεταξύ δύο χρηστών είναι αμφίδρομη, αφού για να είναι κάποιος φίλος με κάποιον άλλο, πρέπει να ισχύει και το αντίθετο. Αντιθέτως, στο Twitter δεν ισχύει αυτό, αφού κάποιος μπορεί να ακολουθεί κάποιον άλλο, χωρίς απαραίτητα να ισχύει και το ανάποδο. Όπως φαίνεται και στο Σχήμα 2.1.1, μπορούμε να αναπαραστήσουμε τις σχέσεις μεταξύ των χρηστών στο Facebook σαν ένα μη κατευθυνόμενο γράφο, ενώ στο Twitter σαν ένα κατευθυνόμενο γράφο.




**Σχήμα 2.1.1:** Σχέσεις μεταξύ των χρηστών στο Facebook και στο Twitter

Η σημαντικότερη έννοια των κοινωνικών δικτύων, την οποία και θα μελετήσουμε, είναι η έννοια της κοινωνικής διάδοσης (social cascading). Με το όρο κοινωνική διάδοση εννοούμε την αναμετάδοση ενός περιεχομένου που αρχικά δημοσίευσε κάποιος χρήστης στο κοινωνικό δίκτυο. Για να το δούμε πιο παραστατικά, έστω ένας χρήστης A δημοσιεύει ένα περιεχόμενο το οποίο του αρέσει, όπως φαίνεται και στο Σχήμα 2.1.2. Αυτό το περιεχόμενο θα φτάσει σε πολλούς άλλους χρήστες, μεταξύ αυτών και οι χρήστες B και C. Κάποιοι από αυτούς τους χρήστες θα το αναμεταδώσουν και το περιεχόμενο αυτό θα φτάσει σε περισσότερους χρήστες (η αναμετάδοση του περιεχομένου από τον B φτάνει στους D,E και F, ενώ από τον C στο G), οι οποίοι με τη σειρά τους μπορεί να το αναμεταδώσουν (χρήστες D, E και F), ενώ κάποιοι άλλοι μπορεί να μην το κάνουν (χρήστης G). Αυτό μπορεί να συνεχίζεται μέχρι να μην υπάρξουν άλλες αναμεταδώσεις. Αυτό έχει ως αποτέλεσμα το περιεχόμενο αυτό να φτάσει σε πολλούς χρήστες. Αυτή η αναμετάδοση του περιεχομένου ονομάζεται κοινωνική διάδοση.




**Σχήμα 2.1.2:** Παράδειγμα κοινωνικής διάδοσης περιεχομένου που ξεκινά από τον χρήστη A

### 2.2 Μοντέλα και Τεχνολογίες

Σε αυτή τη μελέτη αναλύουμε σύνολα δεδομένων μεγάλου μεγέθους. Για να μπορέσουμε να το κάνουμε αυτό θα βασιστούμε σε μοντέλα και τεχνολογίες που μπορούν να υποστηρίξουν αποδοτικά την επεξεργασία ενός τόσο μεγάλου όγκου δεδομένων.

#### 2.2.1 Μοντέλο Map-Reduce

Είναι ένα προγραμματιστικό μοντέλο [8], το οποίο χρησιμοποιείται για επεξεργασία μεγάλου όγκου δεδομένων. Η επεξεργασία μπορεί να γίνει και με παράλληλη λειτουργία.

Η κύρια ιδέα είναι ότι έχουμε ζευγάρια «κλειδί-τιμή» από δεδομένα (<key, value> pairs) και υπάρχουν δύο βασικές λειτουργίες:

1) η λειτουργία **map**, η οποία δέχεται σαν είσοδο ένα ζεύγος κλειδί-τιμή και κάνοντας κάποια επεξεργασία σε αυτό, παράγει σαν έξοδο ένα νέο ζεύγος κλειδί-τιμή
2) η λειτουργία **reduce**, δέχεται σαν είσοδο ένα σύνολο από τιμές που έχουν το ίδιο κλειδί (ζεύγη <κλειδί, λίστα με τιμές> δηλαδή, που παράχθηκαν από τις λειτουργίες map) και μειώνει το σύνολο των τιμών, παράγοντας ένα νέο ζεύγος κλειδί-τιμή.

### 2.2.2 Google Map-Reduce και Hadoop

Το Google File System [9] είναι μία υλοποίηση του μοντέλου Map-Reduce. Χρησιμοποιείται αποκλειστικά από την Google, όμως δεν υπάρχουν πολλές λεπτομέρειες που αφορούν την υλοποίηση αυτή (μάλλον είναι σε γλώσσα C++).

Το Hadoop [10] είναι η υλοποίηση ανοιχτού κώδικα (open-source) του Map-Reduce και είναι υλοποιημένο σε Java. Η υλοποίηση του βασίστηκε κυρίως σε δύο άρθρα [11] [12] που δημοσίευσαν οι εμπνευστές του Map-Reduce και περιέγραφαν το μοντέλο. Είναι δημιουργία της Apache Foundation, με οικονομική βοήθεια κυρίως από την Yahoo.

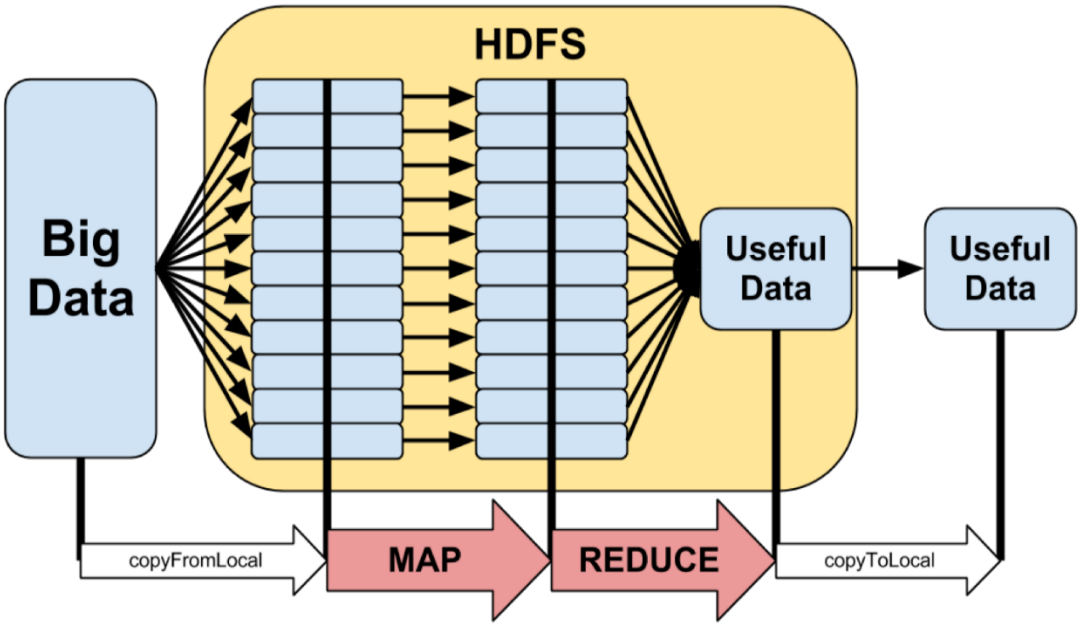
Το Hadoop επιτρέπει την επεξεργασία και ανάλυση πολύ μεγάλου όγκου δεδομένων σε κλίμακα Peta Byte (1 Peta Byte = 1024 Tera Bytes). Ο σκοπός της ανάλυσης είναι η εξόρυξη χρήσιμων πληροφοριών και τάσεων, όπως φαίνεται και στο Σχήμα 2.2.1.

Εκτός από την Yahoo, χρησιμοποιείται ευρέως και από άλλους οργανισμούς, όπως Facebook, AOL, Netflix, Amazon. Apple, eBay κ.α. Το Facebook δήλωσε τον Ιούνιο του 2012 ότι κατέχει το πιο μεγάλο Hadoop Cluster παγκοσμίως, με συνολικό όγκο δεδομένων περίπου 100 PB και αυξάνεται κάθε μέρα κατά 0.5 PB περίπου.

Ο πυρήνας του Hadoop αποτελείται από δύο μέρη:

1) το κατανεμημένο σύστημα αρχείων Hadoop Distributed File System (HDFS), που είναι υπεύθυνο για την αποθήκευση των δεδομένων
2) το προγραμματιστικό μοντέλο του Map-Reduce, που είναι υπεύθυνο για την επεξεργασία των δεδομένων

Ο μηχανισμός του Map-Reduce στο Hadoop ακολουθεί το μοντέλο του αφέντη - υπηρέτη (master - slave). Ο κύριος κόμβος (αφέντης) είναι υπεύθυνος για το καταμερισμό των εργασιών. Οι κόμβοι υπηρέτες εκτελούν τις εργασίες που τους αναθέτει ο κόμβος - αφέντης και επιστρέφουν τα αποτελέσματα που παράγουν σε αυτόν.



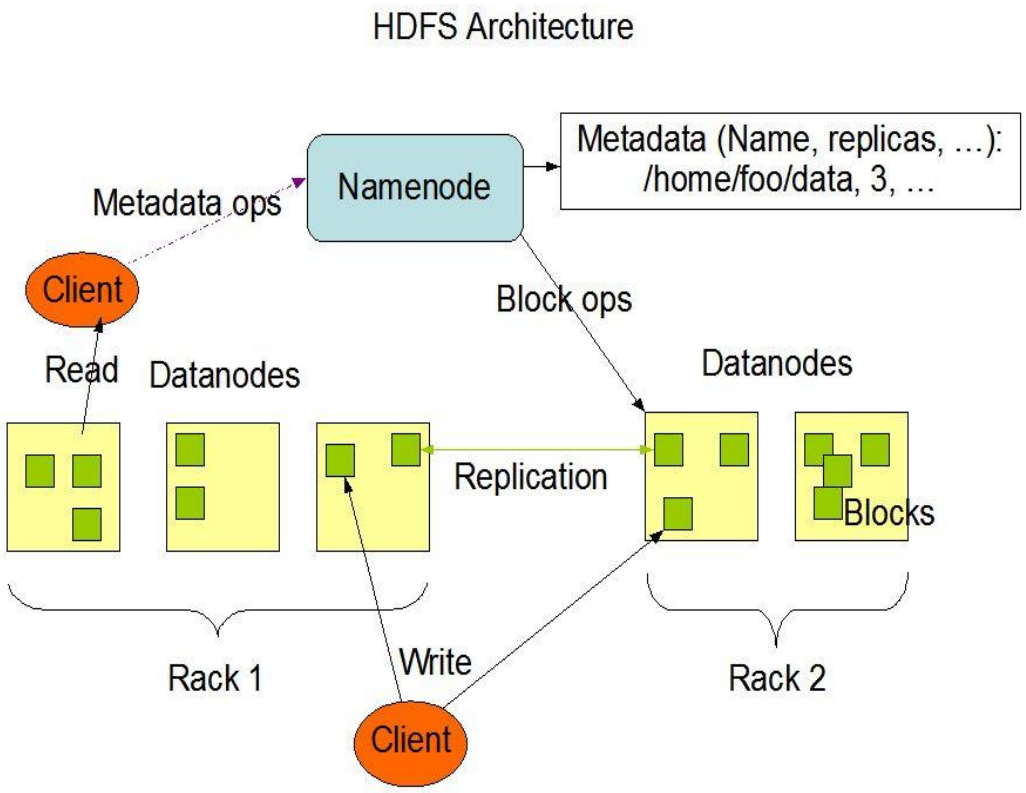

**Σχήμα 2.2.1:** Οι λειτουργίες Map και Reduce στο Hadoop [13]

### 2.2.3 Hadoop Distributed File System (HDFS)

Το Hadoop βασίζεται στο κατανεμημένο σύστημα αρχείων HDFS [14]. Είναι εμπνευσμένο από το Google File System και είναι σχεδιασμένο για την αποθήκευση μεγάλου όγκου δεδομένων. Το HDFS έχει data block size συνήθως 64 MB. Επομένως δεν είναι αποδοτικό για να διαχειρίζεται μικρά αρχεία (με μέγεθος πολύ πιο μικρό από 64 MB), αλλά είναι σχεδιασμένο για αποδοτική διαχείριση αρχείων με τεράστιο μέγεθος.

Το HDFS παρέχει αξιοπιστία μέσω αντιγραφής (replication) των δεδομένων σε περισσότερους από ένα κόμβους / υπολογιστές (nodes). Με αυτό τον τρόπο, όταν ένας κόμβος δεν είναι διαθέσιμος, τα δεδομένα μπορούν να ανακτηθούν από άλλους κόμβους.

Όπως φαίνεται στο Σχήμα 2.2.2, ο κεντρικός node (master node) στο HDFS έχει το ρόλο του Name Node. Ο Name Node διατηρεί διάφορες μετα-πληροφορίες (metadata) για το σύστημα αρχείων, όπως τον Πίνακα (index) που περιγράφει πού βρίσκεται το κάθε αρχείο ή κομμάτι αρχείου (chunk). Οι υπόλοιποι κόμβοι έχουν το ρόλο του Data Node (κόμβοι εργάτες).



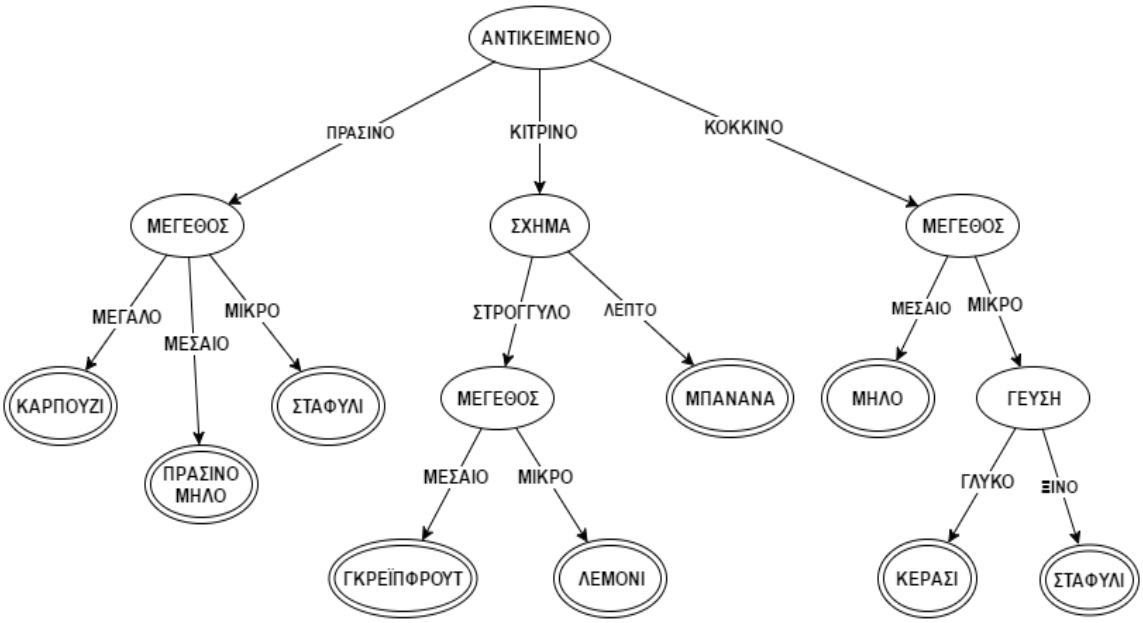

**Σχήμα 2.2.2:** Αναπαράσταση της αρχιτεκτονικής του HDFS [14]

### 2.2.4 Gradient Boosting Classifier

Γενικά, τα gradient boosted δέντρα απόφασης (gradient boosted decision trees) [15] χρησιμοποιούνται ευρέως στη μηχανική μάθηση σε προβλήματα κατηγοριοποίησης και παλινδρόμησης. Έχει την βάση του στην επιβλεπόμενη μάθηση (supervised learning), όπου δοθέντος ενός συνόλου δεδομένων εκπαίδευσης και των επιθυμητών αποτελεσμάτων του κάθε αντικείμενου σε αυτό το σύνολο, το μοντέλο μπορεί να εκπαιδευτεί έτσι ώστε να μπορεί να αποφασίζει το επιθυμητό αποτέλεσμα οποιουδήποτε αντικειμένου που βρίσκεται εκτός του συνόλου εκπαίδευσης.

Η μάθηση μέσα από δέντρο απόφασης είναι η μέθοδος που χρησιμοποιεί δέντρο απόφασης ως μοντέλο πρόβλεψης, όπου παρατηρώντας τα χαρακτηριστικά ενός αντικειμένου μπορεί να αποφασίσει το επιθυμητό αποτέλεσμα (κατηγορία) του αντικειμένου. Τα δεντρικά μοντέλα στα οποία οι τιμές που μπορεί να πάρει το επιθυμητό αποτέλεσμα (κατηγορίες) ανήκουν σε ένα πεπερασμένο σύνολο τιμών, ονομάζονται δέντρα κατηγοριοποίησης (classification trees). Σε αυτές τις δεντρικές δομές, όπως δείχνει και το παράδειγμα στο Σχήμα 2.2.3, τα φύλλα αναπαριστούν τις κατηγορίες, ενώ

οι διακλαδώσεις στους εσωτερικούς κόμβους αναπαριστούν τους συνδέσμους μεταξύ των χαρακτηριστικών που οδηγούν σε αυτές τις κατηγορίες.




**Σχήμα 2.2.3:** Παράδειγμα δέντρου απόφασης

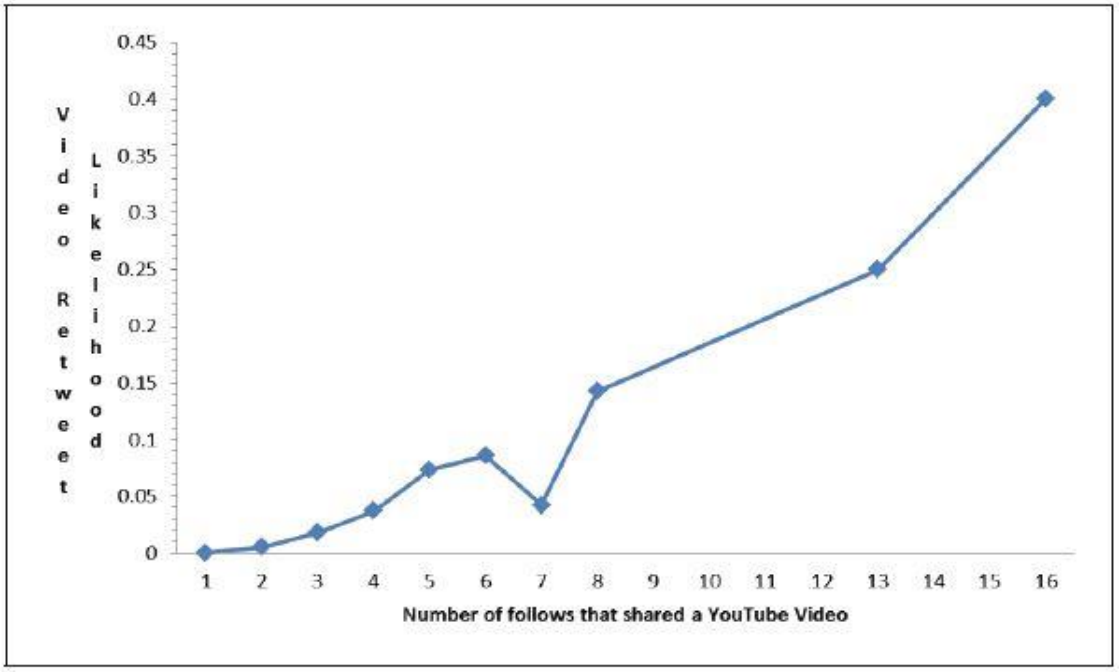
Το Gradient Boosting Classifier [16] είναι μία υλοποίηση των gradient boosting δέντρων απόφασης από την βιβλιοθήκη scikit-learn, το οποίο είναι υλοποιημένο σε γλώσσα Python.

### 2.3 Σχετική Βιβλιογραφία

Σε προηγούμενη μελέτη που έγινε σε διπλωματική εργασία του Πανεπιστημίου Κύπρου [17], μελετήθηκε η επιρροή του κοινωνικού δικτύου Twitter στο YouTube. Συγκεκριμένα, τα δεδομένα συλλέχθηκαν μεταξύ της περιόδου από το Δεκέμβρη του 2011 μέχρι τον Απρίλη του 2012 και αφορούσαν χαρακτηριστικά 37 εκατομμυρίων χρηστών του Twitter, οι οποίοι είχαν 7 δισεκατομμύρια απευθείας συνδέσμους μεταξύ τους. Από αυτά όμως, για πρακτικούς λόγους, επιλέχθηκαν 1.4 εκατομμύρια χρήστες με τυχαίο τρόπο, οι οποίοι δημοσίευσαν περίπου 300 εκατομμύρια tweets.

Μετά τη συλλογή και ανάλυση των δεδομένων, κατέληξαν στο συμπέρασμα ότι όσο περισσότεροι χρήστες από αυτούς που ακολουθά κάποιος έχουν διαδώσει ή αναμεταδώσει το ίδιο περιεχόμενο, τόσο περισσότερη είναι η πιθανότητα αυτός ο

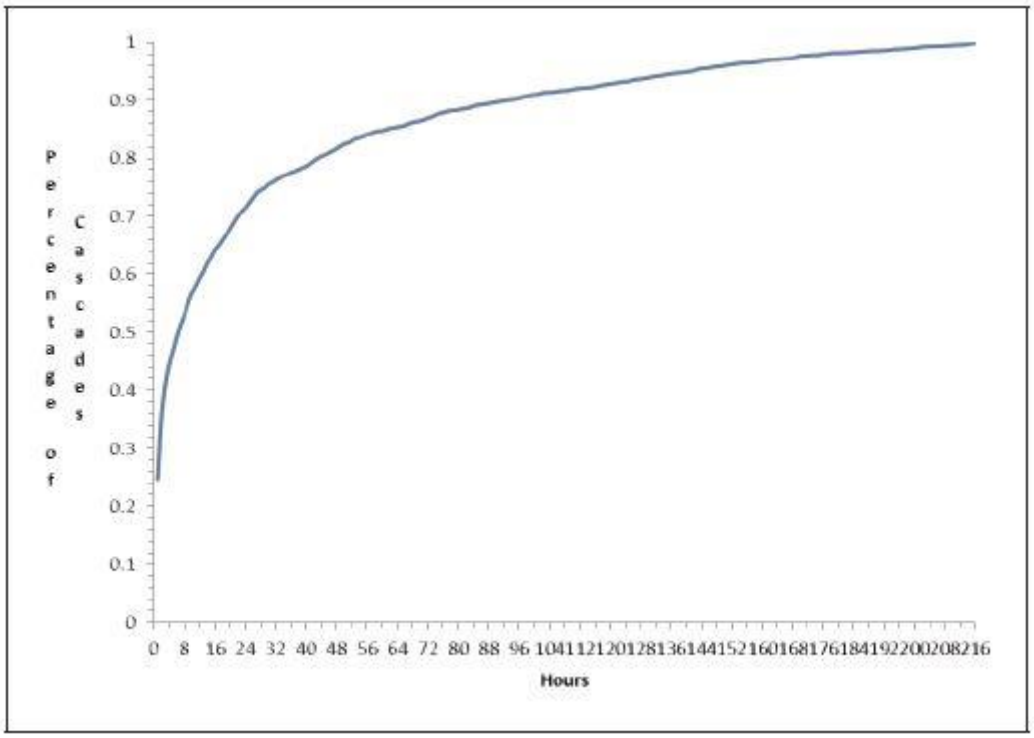
χρήστης να αναμεταδώσει το περιεχόμενο αυτό όπως φαίνεται και στο Σχήμα 2.3.1, με την πιθανότητα να αυξάνεται ακόμα περισσότερο όταν οι χρήστες ακολουθούν ο ένας τον άλλο (όπως ο κεντρικός και ο κάτω αριστερά κόμβος του Twitter στο Σχήμα 2.1.1).




**Σχήμα 2.3.1:** Η πιθανότητα αναμετάδοσης ενός περιεχομένου ανά τον αριθμό αυτών που ακολουθούν και μετέδωσαν ένα βίντεο του YouTube [17]

Επιπλέον, παρατηρήθηκε ότι οι χρήστες επηρεάζονται περισσότερο από χρήστες που βρίσκονται γεωγραφικά πιο κοντά. Η γεωγραφική τοποθεσία του κάθε χρήστη προέκυψε με την υπόθεση ότι χρήστες που βρίσκονται στο ίδιο time zone [18] έχουν και την ίδια γεωγραφική τοποθεσία.

Ακόμη μία σημαντική παρατήρηση φαίνεται στο Σχήμα 2.3.2. Αυτή είναι ότι η διάδοση των βίντεο γίνεται σε μεγάλη κλίμακα σε πολύ σύντομο χρονικό διάστημα από τη μέρα που το βίντεο αναμεταδόθηκε για πρώτη φορά. Πιο συγκεκριμένα το 70% της συνολικής διάδοσης του περιεχομένου γίνεται τις πρώτες 24 ώρες και στη συνέχεια ο ρυθμός αναμετάδοσης του εξασθενεί.



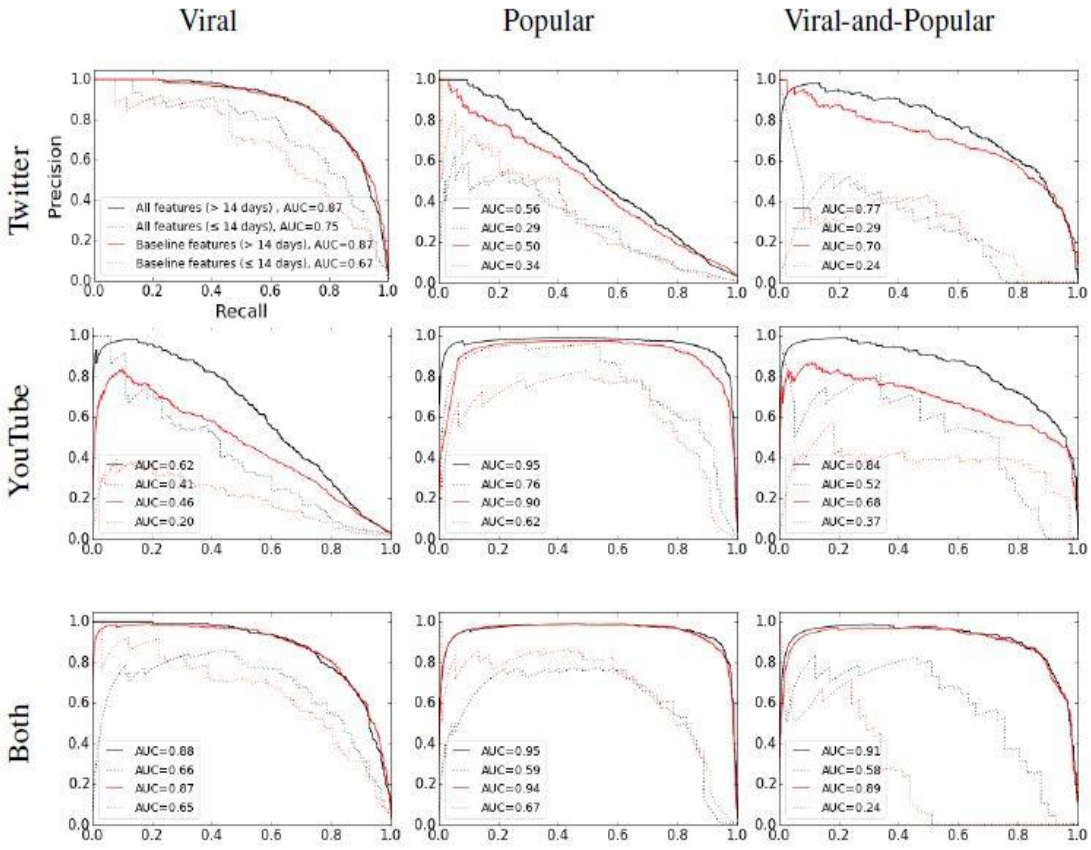

**Σχήμα 2.3.2:** Ποσοστό μετάδοσης ανά τις ώρες από την στιγμή που αναμεταδόθηκε για πρώτη φορά [17]

Στο [19] έγινε μια πολύ ενδιαφέρον μελέτη στην οποία βασίστηκε η διπλωματική αυτή. Πιο συγκεκριμένα, μελετήθηκε η δημοτικότητα (popularity) των βίντεο στο YouTube και παράλληλα η μετάδοση τους (virality) στο Twitter. Τα αποτελέσματα τους έδειξαν ότι χρησιμοποιώντας τα χαρακτηριστικά του βίντεο από το YouTube, αλλά και από το Twitter, μπορούν να κάνουν μια πάρα πολύ καλή πρόβλεψη της δημοτικότητας και της διάδοσης του βίντεο.

Η συλλογή των δεδομένων γινόταν για δύο εβδομάδες, όπου καθημερινά μάζευαν δεδομένα για 200 χιλιάδες βίντεο του YouTube που διαδόθηκαν στο Twitter, από τα οποία κράτησαν τα 20 χιλιάδες για να σχηματίσουν το σύνολο δεδομένων τους.

Όπως φαίνεται στο Σχήμα 2.3.3, χρησιμοποιώντας ως μετρική αξιολόγησης της πρόβλεψης τους το Precision-Recall Score [20], φαίνεται ξεκάθαρα ότι χρησιμοποιώντας τα χαρακτηριστικά του Twitter δίνουν μια πολύ καλή πρόβλεψη για τη διάδοση του βίντεο στο Twitter, ενώ χρησιμοποιώντας τα χαρακτηριστικά του YouTube δίνουν πολύ καλή πρόβλεψη για τη δημοτικότητα του βίντεο στο YouTube. Παρ' όλα αυτά,

χρησιμοποιώντας τα χαρακτηριστικά του Twitter για την πρόβλεψη της δημοτικότητας του βίντεο στο YouTube και αντίστροφα, δίνουν μια αρκετά ικανοποιητική πρόβλεψη, αλλά με μικρότερη ακρίβεια. Η πρόβλεψη όμως με τη ψηλότερη ακρίβεια μπορεί να επιτευχθεί όταν χρησιμοποιούνται τα χαρακτηριστικά και από τα δύο κοινωνικά δίκτυα για να προβλεφθεί η δημοτικότητα και η μετάδοση σε αυτά.




**Σχήμα 2.3.3:** Τα αποτελέσματα της πρόβλεψης της δημοτικότητας των βίντεο στο [19]

Το πιο σημαντικό επίτευγμα αυτής της μελέτης όμως, είναι το γεγονός ότι μπόρεσαν να προβλέψουν τη δημοτικότητα και διάδοση του περιεχομένου σε μικρή χρονική περίοδο εκπαίδευσης και με μικρό σύνολο δεδομένων. Πιο αναλυτικά, έχοντας ένα σχετικά μικρό σύνολο από δεδομένα εκπαίδευσης μίας μόνο μέρας, μπόρεσαν να κάνουν πρόβλεψη αρκετά μεγάλης ακρίβειας μέχρι και 7 μέρες πιο μετά. Τα καλύτερα αποτελέσματα όμως, όπου υπάρχει ισορροπία μεταξύ ακρίβειας και μικρού χρονικού διαστήματος, επιτεύχθηκαν με δεδομένα εκπαίδευσης 3 ημερών και δεδομένα επαλήθευσης επίσης 3 ημερών, όπως φαίνεται και στο Σχήμα 2.3.4.

| Training window size | Label window size (offset = 1) | | | Offset (label window size= 3) | | |
|---|---|---|---|---|---|---|
| | 1 day | 3 days | 7 days | 1 day | 3 days | 7 days |
| 1 day | 0.769 | 0.809 | 0.825 | 0.807 | 0.798 | 0.788 |
| 3 days | 0.770 | 0.823 | 0.830 | 0.824 | 0.817 | 0.790 |
| 7 days | 0.778 | 0.815 | 0.824 | 0.816 | 0.800 | 0.771 |

**Σχήμα 2.3.4:** Επίδραση της μεταβολής του χρονικού διαστήματος εκπαίδευσης και επαλήθευσης στη πρόβλεψη της δημοτικότητας και της διάδοσης του βίντεο στο Twitter

# Κεφάλαιο 3

# Συλλογή και Επεξεργασία Δεδομένων



Σε αυτό το κεφάλαιο, περιγράφουμε τη μεθοδολογία συλλογής των δεδομένων μας καθώς και τα χαρακτηριστικά του Twitter και του YouTube που περιέχονται στο σύνολο δεδομένων. Πιο συγκεκριμένα περιγράφεται το σύνολο δεδομένων από το Twitter, η διαδικασία εύρεσης των tweets που περιέχουν σύνδεσμο προς το YouTube, η διαδικασία εξαγωγής των χαρακτηριστικών από το API του YouTube, και τέλος, η προεπεξεργασία που γίνεται σε αυτό το σύνολο δεδομένων.

### 3.1 Συλλογή Δεδομένων από Twitter

Για να μπορέσουμε να εκπαιδεύσουμε ένα σύστημα που θα προβλέπει την διάδοση των βίντεο, υπάρχει η αναγκαία προϋπόθεση της συλλογής ενός μεγάλου όγκου δεδομένων. Σε αυτή την μελέτη έχει δοθεί ένα σύνολο δεδομένων που αποτελείται από 50 εκατομμύρια tweets μαζί με τα χαρακτηριστικά τους, καθώς και ένα σύνολο δεδομένων με τους χρήστες που δημοσίευσαν αυτά τα tweets και τα χαρακτηριστικά του κάθε χρήστη. Τα σύνολα αυτά έχουν συλλεχθεί από το Χάρη Ευσταθιάδη μέσα από το API του Twitter, στη μελέτη που έχει κάνει στο [21].

Ο λόγος που επιλέχθηκε το κοινωνικό δίκτυο Twitter για την συλλογή των δεδομένων είναι λόγω της προσβασιμότητας των δεδομένων. Το Twitter έχει καθιερωθεί ως ένα από τα δημοφιλέστερα κοινωνικά δίκτυα για σκοπούς πληροφόρησης και ενημέρωσης. Τα δεδομένα σε αυτό είναι ελεύθερα, εκτός και αν ο χρήστης επιλέξει ρητά να τα αποκρύψει,

σε αντίθεση με άλλα κοινωνικά δίκτυα όπου για να εξάγεις δεδομένα θα πρέπει να λάβεις την έγκριση των χρηστών ή του κοινωνικού δικτύου.

Η συλλογή όμως ενός τόσο μεγάλου όγκου δεδομένων είναι πολύ χρονοβόρα, λόγω των περιορισμών που επιβάλλει το Twitter στις αιτήσεις εξαγωγής δεδομένων που μπορεί να κάνει κάποιος. Για την εξαγωγή των συνόλων αυτών στο [21], μάζεψαν δημόσιες πληροφορίες για χρήστες που δημοσίευσαν tweets, οι οποίες περιλαμβάνουν τη δραστηριότητα του χρήστη, τον κοινωνικό του κύκλο στο δίκτυο με συνδέσμους από και προς αυτόν, αλλά και προσωπικές του πληροφορίες. Για να επεκτείνουν το σύνολο δεδομένων, χρησιμοποίησαν αυτούς τους χρήστες ως τροφοδότες (seeders). Για το κάθε ένα χρήστη από αυτούς, επέλεξαν με τυχαίο τρόπο χρήστες από το κοινωνικό τους κύκλο και μάζεψαν τα ίδια δεδομένα και για αυτούς.

| **Όνομα χαρακτηριστικού** | **Περιγραφή** |
|---|---|
| User ID | Ταυτότητα του χρήστη |
| Followers count | Αριθμός των χρηστών που τον ακολουθούν |
| Followings count | Αριθμός των χρηστών που ακολουθεί ο χρήστης |
| Is Verified Celebrity | Αν ο χρήστης είναι κάποιος διάσημος |

**Πίνακας 3.1.1:** Χαρακτηριστικά για κάθε χρήστη του Twitter

Δοθέντος αυτού του συνόλου δεδομένων λοιπόν, έπρεπε να βρούμε ποια tweets περιέχουν διαδικτυακούς συνδέσμους (URLs). Εδώ πρέπει να αναφέρουμε ότι λόγω του περιορισμού στο μέγεθος ενός tweet (140 χαρακτήρες μέγιστο), όλοι οι σύνδεσμοι βρίσκονται σε μορφή σμίκρυνσης με την υπηρεσία σμίκρυνσης « t.co » του Twitter [22].

Επομένως, χρησιμοποιώντας το Hadoop κρατήσαμε μόνο τα tweets που περιείχαν συνδέσμους σε μορφή σμίκρυνσης και τα πιο κύρια χαρακτηριστικά τους, όπως φαίνονται στους Πίνακες 3.1.1 και 3.1.2.

| **Όνομα χαρακτηριστικού** | **Περιγραφή** |
|---|---|
| Tweet ID | Η ταυτότητα του tweet |
| User ID | Η ταυτότητα του χρήστη που το δημοσίευσε |
| Is retweet | Αν το tweet είναι αναμετάδοση κάποιου άλλου tweet |
| Is retweeted | Αν το tweet έχει αναμεταδοθεί |
| Retweet count | Ο συνολικός αριθμός των αναμεταδόσεων του tweet (ο αριθμός αυτός περιλαμβάνει και τις αναμεταδόσεις των retweets) |
| Favorite count | Αριθμός των χρηστών που το επέλεξαν ως αγαπημένο τους |
| Day | Η μέρα δημοσίευσης του tweet |
| Month | Ο μήνας δημοσίευσης του tweet |
| Year | Το έτος δημοσίευσης του tweet |
| Minutes | Το λεπτό δημοσίευσης του tweet |
| Hours | Η ώρα δημοσίευσης του tweet |
| Tiny URL | Σύνδεσμος σε μορφή σμίκρυνσης |

**Πίνακας 3.1.2:** Χαρακτηριστικά για κάθε tweet στο Twitter

### 3.2 Εξαγωγή Πραγματικών Συνδέσμων

Από το αρχικό μας σύνολο δεδομένων, βρήκαμε 12.831.717 tweets που περιείχαν σύνδεσμο σε μορφή σμίκρυνσης. Έχοντας το νέο σύνολο δεδομένων, το επόμενο βήμα ήταν να κρατήσουμε μόνο αυτά που περιέχουν συνδέσμους στο YouTube. Για να μπορέσουμε να το κάνουμε αυτό θα έπρεπε πρώτα να εξάγουμε τους πραγματικούς συνδέσμους από αυτούς που είναι σε μορφή σμίκρυνσης.

Για να γίνει αυτό χρησιμοποιήσαμε το λογισμικό ανοιχτού κώδικα στο [23], το οποίο μέσω της βιβλιοθήκης Java.net στέλνει αιτήσεις στο παγκόσμιο ιστό έτσι ώστε να πάρει πίσω τον πραγματικό σύνδεσμο. Εκτελώντας τον συγκεκριμένο κώδικα μέσα από το Hadoop, αποκτήσαμε τους πραγματικούς συνδέσμους και κρατήσαμε μόνο αυτούς που περιείχαν σύνδεσμο στο YouTube.

Σε αυτό το σημείο πρέπει να σημειωθεί ότι το συγκεκριμένο λογισμικό του Hadoop έτρεχε σε ένα Cluster επεξεργαστών. Λόγω τεχνικών προβλημάτων δεν κατέστη δυνατό

να εξάγουμε όλους τους συνδέσμους, αλλά καταφέραμε να πάρουμε μόνο το 35%, το οποίο μας έδωσε περίπου 200000 συνδέσμους προς το YouTube. Σε μελλοντική φάση θα χρησιμοποιηθεί ολόκληρο το σύνολο αποτελεσμάτων.

### 3.3 Τροποποίηση Συνδέσμων YouTube στην Απλή τους Μορφή

Οι πραγματικοί σύνδεσμοι που προέκυψαν από τους συνδέσμους σε μορφή σμίκρυνσης δεν ήταν όμως στην απλή μορφή του YouTube, δηλαδή στη μορφή "www.youtube.com/watch?v=", ακολουθούμενο από το κλειδί (ταυτότητα βίντεο) έντεκα ψηφίων. Η απλή μορφή είναι απαραίτητη για την εξαγωγή των δεδομένων από το API του YouTube [24]. Επομένως έπρεπε να τροποποιηθούν.

Από τις διάφορες μορφές που υπήρχαν, μερικές ήταν ελλιπείς, ενώ άλλες δεν αφορούσαν σύνδεσμο σε βίντεο, αλλά αφορούσαν προφίλ χρηστών του YouTube. Επίσης υπήρχαν κάποιες που αφορούσαν λίστες αναπαραγωγής, αλλά ήταν αδύνατο να ανακτηθεί το κλειδί με τα έντεκα ψηφία. Αυτές οι μορφές όμως ήταν ελάχιστες (λιγότερο από 1% του συνόλου δεδομένων), για αυτό και αφαιρέθηκαν από το σύνολο δεδομένων.

Οι περισσότερες μορφές συνδέσμων από το σύνολο δεδομένων, οι οποίες και τροποποιήθηκαν μέσω του Hadoop, αφορούσαν τις εξής μορφές:

- Κάποιοι σύνδεσμοι αφορούσαν πρωτόκολλα HTTP, ενώ άλλα HTTPS [25]. Στην τροποποίηση έγιναν όλοι οι σύνδεσμοι στην πιο ελεύθερη HTTP μορφή
- Σύνδεσμοι της μορφής "www.youtube.com/watch?v=", ακολουθούμενοι από το κλειδί έντεκα ψηφίων (μετά το κλειδί υπήρχαν σύνδεσμοι που ακολουθούνταν και από άλλες πληροφορίες, οι οποίες δεν ανήκουν στην απλή μορφή του συνδέσμου). Αυτές οι επιπλέον πληροφορίες αφαιρέθηκαν
- Σύνδεσμοι της μορφής "www.youtube.com/watch?", ακολουθούμενοι από άλλες πληροφορίες (οι οποίες δεν ανήκουν στην απλή μορφή του συνδέσμου) και στη συνέχεια από τους χαρακτήρες "v=" και το κλειδί έντεκα ψηφίων (μετά από το κλειδί υπήρχαν σύνδεσμοι που ακολουθούνταν και από άλλες πληροφορίες, αχρείαστες). Από αυτές αφαιρέθηκαν όλες οι επιπλέον πληροφορίες
- Σύνδεσμοι της μορφής "youtu.be/", η οποία είναι μια άλλη μορφή σμίκρυνσης, αυτή του YouTube [26]. Η μορφή αυτή ακολουθείται από το κλειδί έντεκα ψηφίων (μετά το κλειδί υπήρχαν σύνδεσμοι που ακολουθούνταν και από άλλες

πληροφορίες, οι οποίες δεν ανήκουν στην απλή μορφή του συνδέσμου). Η τροποποίηση που έγινε εδώ ήταν η αλλαγή του "youtu.be/" σε "www.youtube.com/watch?v=", και η πρόσθεση του κλειδιού των έντεκα ψηφίων στη συνέχεια

- Σύνδεσμοι της μορφής "www.youtube.com/embed/", ακολουθούμενοι από το κλειδί έντεκα ψηφίων, όπου αφαιρέθηκε το "embed/" και αντικαταστάθηκε με το "watch?v= ", ακολουθούμενο από το κλειδί

### 3.4 Εξαγωγή Μετα-Δεδομένων από το YouTube API

Χρησιμοποιώντας κώδικα JavaScript και PHP, στείλαμε αιτήματα (post requests) στο YouTube API για να εξάγουμε τα μετα-δεδομένα των συνδέσμων του συνόλου δεδομένων, σύμφωνα με τις λειτουργίες που μας παρέχει το συγκεκριμένο API [24].

Το API όμως, επιβάλλει περιορισμούς στο τι μετα-δεδομένα μπορεί να εξαχθούν για τον κάθε σύνδεσμο. Τα δεδομένα που συλλέξαμε φαίνονται στο Πίνακα 3.4.1.

| **Όνομα** | **Περιγραφή** |
|---|---|
| Video ID | Η ταυτότητα του βίντεο |
| Published At | Ο χρόνος δημοσίευσης του βίντεο (ημερομηνία και ώρα) |
| Channel Id | Η ταυτότητα του καναλιού, από την οποία δημοσιεύτηκε το βίντεο |
| View Count | Αριθμός των συνολικών θεάσεων του βίντεο |
| Like Count | Αριθμός των χρηστών που δήλωσαν ότι τους άρεσε το βίντεο |
| Dislike Count | Αριθμός των χρηστών που δήλωσαν ότι δεν τους άρεσε το βίντεο |
| Favorite Count | Αριθμός των χρηστών που δήλωσαν το βίντεο ως ένα από τα αγαπημένα τους |
| Category Id | Η κατηγορία στην οποία ανήκει το βίντεο |
| Comment Count | Αριθμός των σχολίων που έκαναν οι χρήστες για το βίντεο |

**Πίνακας 3.4.1:** Χαρακτηριστικά για κάθε βίντεο του YouTube

Ένα χαρακτηριστικό που αξίζει να αναφερθούμε είναι η κατηγορία του βίντεο. Υπάρχουν 32 διαφορετικές κατηγορίες βίντεο, οι οποίες φαίνονται στον Πίνακα 3.4.2.

Αξίζει να σημειωθεί ότι πολλά από τα βίντεο είχαν αφαιρεθεί από το YouTube και επομένως δεν μπορέσαμε να εξάγουμε τα χαρακτηριστικά τους. Ακόμη, κάποιοι χρήστες δεν επέτρεπαν να εξάγουμε τις πιο πάνω πληροφορίες. Αποτέλεσμα των δύο πιο πάνω θεμάτων ήταν να καταφέρουμε να συλλέξουμε μετα-δεδομένα μόνο για 75000 βίντεο από τα 200000 που είχαμε αρχικά.

Τα μετα-δεδομένα που πήραμε ήταν σε μορφή JSON [27] και έπρεπε να τα μετατρέψουμε σε μορφή απλού κειμένου (πιο συγκεκριμένα σε μορφή csv). Επίσης έπρεπε να αφαιρέσουμε από το σύνολο δεδομένων τους συνδέσμους στο YouTube, των οποίων τα βίντεο είχαν αφαιρεθεί, με αποτέλεσμα να μην μας επιστρέψουν τα αναμενόμενα δεδομένα. Αυτό έγινε με την χρήση κώδικα σε Python.

| **Ταυτότητα** | **Όνομα** | | **Ταυτότητα** | **Όνομα** |
|---|---|---|---|---|
| 1 | Film &Animation | | 29 | Nonprofits & Activism |
| 2 | Autos & Vehicles | | 30 | Movies |
| 10 | Music | | 31 | Anime / Animation |
| 15 | Pets & Animals | | 32 | Action / Adventure |
| 17 | Sports | | 33 | Classics |
| 18 | Short Movies | | 34 | Comedy |
| 19 | Travel & Events | | 35 | Documentary |
| 20 | Gaming | | 36 | Drama |
| 21 | Videoblogging | | 37 | Family |
| 22 | People & Blogs | | 38 | Foreign |
| 23 | Comedy | | 39 | Horror |
| 24 | Entertainment | | 40 | Sci-Fi / Fantasy |
| 25 | News & Politics | | 41 | Thriller |
| 26 | How to & Style | | 42 | Shorts |
| 27 | Education | | 43 | Shows |
| 28 | Science & Technology | | 44 | Trailers |

**Πίνακας 3.4.2:** Κατηγορίες βίντεο του YouTube

### 3.5 Χρονολογική Ταξινόμηση των Tweets

Το επόμενο βήμα ήταν η ταξινόμηση των δεδομένων σε αύξουσα σειρά, με βάση το χρόνο δημοσίευσης τους στο Twitter. Για να γίνει αυτό, χρησιμοποιήθηκε κώδικας σε Java. Ο αλγόριθμος ταξινόμησης που χρησιμοποιήθηκε είναι ο αλγόριθμος Merge Sort [28].

Ο συγκεκριμένος αλγόριθμος είναι τύπου διαίρει και βασίλευε και είναι ένας πολύ αποτελεσματικός αλγόριθμος ταξινόμησης για μεγάλο όγκο δεδομένων, αφού έχει πολυπλοκότητα λογαριθμική ως προς το μέγεθος του προβλήματος (για μέγεθος προβλήματος n, έχει πολυπλοκότητα O(n * lg(n)) ). Ο αλγόριθμος αυτός μοιράζει το πρόβλημα σε μικρότερα υποπροβλήματα και επιλύει αναδρομικά το πρόβλημα ταξινομώντας πρώτα τα μικρότερα υποπροβλήματα.

### 3.6 Αντιστοίχιση Χαρακτηριστικών Tweets - Χρηστών

Το αρχικό σύνολο δεδομένων που μας δόθηκε περιείχε δύο ξεχωριστά αρχεία: ένα με τα χαρακτηριστικά των Tweets και ένα με τα χαρακτηριστικά των Users (βλ. Πίνακες 3.1.1 και 3.1.2). Σε αυτή τη φάση, όπου ο όγκος δεδομένων σε θέμα χώρου έχει μειωθεί σημαντικά (αφαιρέθηκαν όλα τα αχρείαστα δεδομένα ή δεδομένα των οποίων τα περιεχόμενα έχουν αφαιρεθεί), πραγματοποιήθηκε η συνένωση των δύο συνόλων δεδομένων χρησιμοποιώντας ως κλειδί στην ένωση αυτή, την ταυτότητα του χρήστη του Twitter.

Για να γίνει αυτό χρησιμοποιήθηκε κώδικας σε Java και η δομή Hash Map [29], για πιο αποδοτική εύρεση της αντιστοίχισης με την ταυτότητα των χρηστών.

# Κεφάλαιο 4

# Αναγνώριση Βασικών Χαρακτηριστικών και Αποτελέσματα



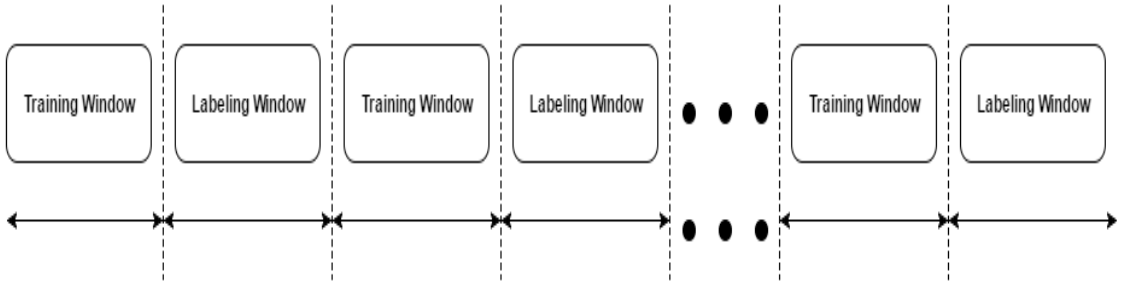
Σε αυτό το κεφάλαιο, παρουσιάζεται η αξιολόγηση της απόδοσης του classifier, ο οποίος έχει εκπαιδευτεί για να προβλέπει τη διάδοση (virality) του βίντεο στο Twitter και τη δημοτικότητα (popularity) του στο YouTube, καθώς γίνεται και ανάλυση της αλληλεπίδρασης των χαρακτηριστικών μεταξύ των δύο συστημάτων (cross-system interactions). Επίσης, γίνεται επεξήγηση των αποτελεσμάτων για την ανάλυση της σημαντικότητας των χαρακτηριστικών και εξάγονται σημαντικές πληροφορίες για την απόδοση του classifier και της αλληλεπίδρασης των χαρακτηριστικών μεταξύ συστημάτων. Τέλος, παρουσιάζεται μελέτη για το πώς το μέγεθος των παραθύρων εκπαίδευσης και επαλήθευσης επηρεάζουν την πρόβλεψη του classifier.

## 4.1 Μεθοδολογία

Η ανάλυση μας αφορά προβλήματα κατηγοριοποίησης, όπου στοχεύουμε να προβλέπουμε την κατηγορία των βίντεο ως viral (ευρέως διαδεδομένα στο Twitter) ή / και popular (ευρέως διαδεδομένα στο YouTube). Πιο συγκεκριμένα θα εκπαιδεύσουμε τρεις gradient boosting classifiers, όπου ο πρώτος θα προβλέπει αν ένα βίντεο είναι viral ή όχι, ο δεύτερος αν ένα βίντεο είναι popular ή όχι και ο τρίτος αν ένα βίντεο είναι και viral και popular ή όχι.

Σε ένα πρόβλημα κατηγοριοποίησης (το οποίο είναι πρόβλημα επιβλεπόμενης μάθησης [30]), απαραίτητη προϋπόθεση είναι η ύπαρξη ενός συνόλου χαρακτηριστικών και το επιθυμητό τους αποτέλεσμα (κατηγορία που ανήκει το στοιχείο με τα συγκεκριμένα χαρακτηριστικά). Το σύνολο των χαρακτηριστικών αποτελείται από το σύνολο δεδομένων με τα χαρακτηριστικά που πήραμε από το Twitter και το YouTube. Η κατηγορία όμως, στην οποία ανήκει το κάθε βίντεο δεν είναι γνωστή. Για να μπορέσουμε να χρησιμοποιήσουμε τον classifier, έπρεπε πρώτα να βρούμε σε ποια κατηγορία ανήκει το κάθε βίντεο.

Για να μπορέσουμε να δούμε αν ένα βίντεο είναι viral ή / και popular ορίσαμε την έννοια του παραθύρου εκπαίδευσης (training window) και του παραθύρου επαλήθευσης (labeling window). Το παράθυρο εκπαίδευσης αποτελείται από τα βίντεο και τα χαρακτηριστικά τους, τα οποία θα χρησιμοποιηθούν για να εκπαιδεύσουμε τον classifier. Το παράθυρο επαλήθευσης είναι η χρονική περίοδος που έρχεται αμέσως μετά το παράθυρο εκπαίδευσης, όπου σε αυτό θα δούμε τη διάδοση του βίντεο έτσι ώστε να αποφασίσουμε σε ποια κατηγορία ανήκει. Τα παράθυρα εκπαίδευσης και επαλήθευσης φαίνονται πιο αναπαραστατικά στο Σχήμα 4.1.1




**Σχήμα 4.1.1:** Παράθυρα Εκπαίδευσης και Επαλήθευσης

Για να αποφασίσουμε σε ποια κατηγορία ανήκει, θα χρησιμοποιήσουμε το retweet count ως μέτρο για τη κατηγορία viral, ενώ θα χρησιμοποιήσουμε το YouTube views ως μέτρο για το popularity. Έχοντας το σύνολο δεδομένων για τη κατηγορία viral και το σύνολο δεδομένων για τη κατηγορία popular, θέσαμε ως viral-popular κατηγορία τα βίντεο που ανήκουν στη τομή των δύο αυτών συνόλων.

Ο λόγος επιλογής του retweet count ως το χαρακτηριστικό που δείχνει τη διάδοση του βίντεο στο Twitter είναι γιατί το χαρακτηριστικό αυτό είναι συνδεδεμένο με το αρχικό

tweet και έτσι κάθε φορά που το αρχικό tweet ή ένα από τα retweet του γίνεται αναμετάδοση, τότε αυτόματα αλλάζει και η τιμή του retweet count. Για παράδειγμα έστω ότι έχουμε το tweet A, το οποίο έχει A.retweet_count = 20. Έστω τώρα ότι κάποιος κάνει retweet το A, ας το ονομάσουμε tweet B. Αν τώρα το tweet B γίνει 2 φορές retweet, τότε η τιμή που θα έχουν τα A.retweet_count και B.retweet_count είναι:
A.retweet_count = 22, B.retweet_count = 22.

Επομένως μέσω προγράμματος σε Python χωρίσαμε τα δεδομένα στα σύνολα των δύο παραθύρων εκπαίδευσης και επαλήθευσης, κρατώντας μόνο τα δεδομένα που υπήρχαν και στα δύο σύνολα. Ο λόγος που έγινε αυτό, είναι γιατί αν ένα βίντεο υπήρχε στα δεδομένα εκπαίδευσης αλλά όχι στα δεδομένα επαλήθευσης, τότε θα ήταν αδύνατο να βρούμε σε ποια κατηγορία ανήκει. Έτσι αν ένα βίντεο υπήρχε περισσότερες από μια φορές στο παράθυρο επαλήθευσης, αλλά είχε εξαχθεί από το Twitter σε διαφορετικές χρονικές στιγμές, τότε για να αποφασίσουμε τη κατηγορία παρατηρούσαμε αυτό με το μεγαλύτερο retweet count. Επιλέξαμε ως viral περίπου το 10% των δεδομένων επαλήθευσης με το ψηλότερο retweet count, ενώ ως popular επιλέξαμε περίπου το 40% των βίντεο με τα ψηλότερα views.

Σε αυτό το σημείο, είναι σημαντικό να αναφέρουμε ότι το μέγεθος των παραθύρων παίζει σημαντικό ρόλο στη πρόβλεψη και θα αναλυθεί η επίδραση του σε κατοπινό στάδιο της μελέτης αυτής. Για αυτό το λόγο, η ανακατανομή των βίντεο στα δύο σύνολα θα γίνεται κάθε φορά που αλλάζει το μέγεθος των παραθύρων.

Όσο αφορά τη μέθοδο αξιολόγησης της απόδοσης του classifier, θα χρησιμοποιήσουμε τις εξής μετρικές:

1) Mean Accuracy: Είναι η μέση τιμή επιτυχίας του classifier, δηλαδή είναι ο μέσος όρος των φορών που ο classifier προβλέπει σωστά την κατηγορία
2) AUC score [31] (Area Under the Curve): Είναι μια μέθοδος αξιολόγησης για classifiers η οποία χρησιμοποιείται ευρέως και υπολογίζεται ως εξής:
   precision(i) = {positive correct predictions} / {positive prediction}
   recall(i) = {positive correct predictions} / {positive targets}
   AUC score(i) = sum(0.5 * (precision(i) + precision(i+1)) * (recall(i+1) - recall(i)))
   όπου i είναι η μια δοκιμή, η οποία αυξάνεται ανάλογα με το άξονα του recall,

positive correct predictions είναι ο αριθμός των σωστών προβλέψεων της κατηγορίας 1 (ο classifier επιστρέφει 1 όταν το βίντεο είναι ευρέως διαδεδομένο, διαφορετικά 0), positive prediction είναι ο αριθμός όλων των θετικών προβλέψεων που επέστρεψε ο classifier, positive targets είναι ο αριθμός των σωστών προβλέψεων που θα έπρεπε να επέστρεφε ο classifier

3) F1 score: Και αυτή η μέθοδος χρησιμοποιεί τα precision και recall (Σχήμα 4.1.2) και υπολογίζεται ως 2 * (precision * recall) / (precision + recall)

Για την αξιολόγηση του classifier ακολουθήσαμε τη μεθοδολογία 10-fold validation, δηλαδή χρησιμοποιήσαμε ως δεδομένα εκπαίδευσης του classifier το 90% των βίντεο που ανήκουν στο παράθυρο εκπαίδευσης και στο παράθυρο επαλήθευσης, έτσι ώστε να γνωρίζουμε ποια βίντεο ανήκουν σε κάθε κατηγορία. Το υπόλοιπο 10% των βίντεο θα χρησιμοποιηθούν ως δεδομένα ελέγχου (test set), έτσι ώστε έχοντας τα χαρακτηριστικά τους να προσπαθήσουμε να προβλέψουμε την κατηγορία τους. Οι προβλέψεις του classifier για τα δεδομένα ελέγχου θα χρησιμοποιηθούν για τον υπολογισμό του AUC score και του Mean Accuracy, έτσι ώστε να αξιολογήσουμε την απόδοση του classifier. Ακόμη, χρησιμοποιήθηκε η μεθοδολογία cross-validation, δηλαδή έγινε επανάληψη του πειράματος 10 φορές, με διαφορετικά δεδομένα ελέγχου κάθε φορά, με σκοπό να έχουμε πιο αξιόπιστη μέτρηση της απόδοσης του classifier.

### 4.2 Πρόβλεψη Διάδοσης και Αξιολόγηση

Αρχικά, γίνεται μελέτη της πρόβλεψης της διάδοσης στο Twitter αλλά και πρόβλεψη της κατηγορίας viral-popular, χρησιμοποιώντας χαρακτηριστικά μόνο του Twitter. Στη συνέχεια γίνεται μελέτη για τις αλληλεπιδράσεις μεταξύ των συστημάτων, όπου χρησιμοποιούνται χαρακτηριστικά του Twitter για την πρόβλεψη της δημοτικότητας του βίντεο στο YouTube, καθώς και χαρακτηριστικά του YouTube για την πρόβλεψη της διάδοσης του βίντεο στο Twitter. Τέλος, γίνεται χρήση όλων των χαρακτηριστικών και από τα δύο συστήματα, για την πρόβλεψη της κλάσης viral-popular, δηλαδή τα βίντεο που έχουν ψηλή δημοτικότητα στο YouTube, αλλά και μεγάλη διάδοση στο Twitter.

Με τη χρήση όλων αυτών των χαρακτηριστικών γίνεται η εκπαίδευση του classifier. Επίσης, χρησιμοποιήθηκε ως μέτρο σύγκρισης ένας δεύτερος classifier, baseline

classifier, ο οποίος χρησιμοποιεί μόνο το retweet count ως χαρακτηριστικό του Twitter και μόνο τα views ως χαρακτηριστικό του YouTube.

Για τα συγκεκριμένα πειράματα, θέσαμε και το παράθυρο εκπαίδευσης να είναι 3 μέρες και το παράθυρο επαλήθευσης να είναι οι αμέσως επόμενες 3 μέρες. Οι γραφικές παραστάσεις παρουσιάζουν τις καμπύλες AUC (AUC Curves), όπου ο άξονας Y περιέχει τις τιμές του Precision, ενώ ο άξονας X περιέχει τις τιμές του Recall.

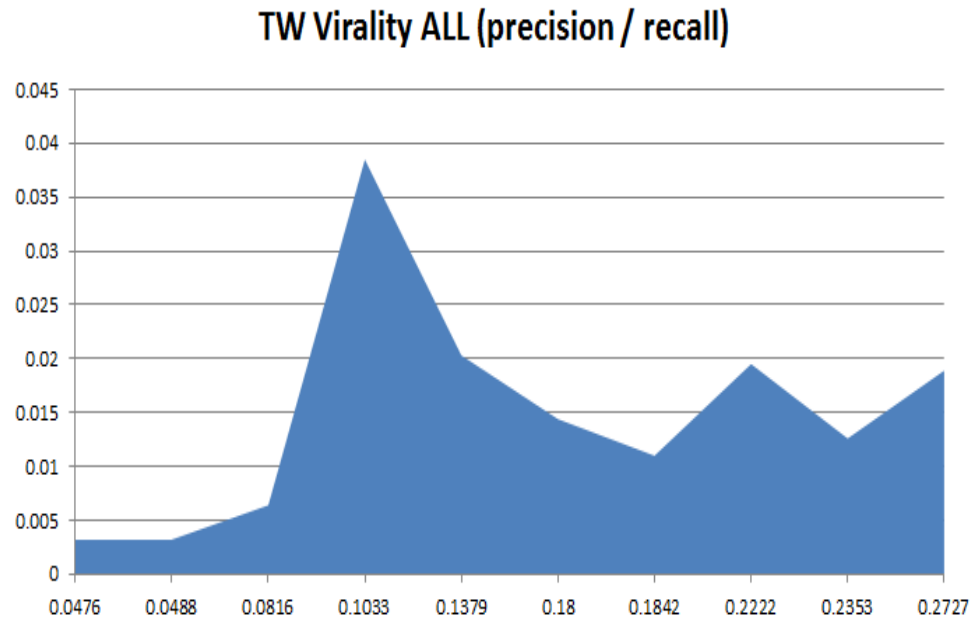
Αξίζει να σημειωθεί ότι δεν γίνεται μελέτη για την πρόβλεψη της δημοτικότητας στο YouTube χρησιμοποιώντας χαρακτηριστικά του YouTube. Ο λόγος είναι το ότι τα δεδομένα για τα χαρακτηριστικά του YouTube που έχουμε για τα βίντεο τα έχουμε εξάγει όλα σε μεταγενέστερη ημερομηνία και όχι στην ημερομηνία που δημοσιεύτηκαν στο Twitter (όπως δηλαδή έγινε με τα tweets). Έτσι δεν μπορεί να γίνει πρόβλεψη, αφού ο classifier αναγνωρίζει ότι τα δεδομένα στα παράθυρα εκπαίδευσης και επαλήθευσης είναι τα ίδια, με αποτέλεσμα να αποφασίζει πάντοτε σωστά την κατηγορία. Επίσης τα αποτελέσματα του AUC score είναι χαμηλά, λόγω του μικρού συνόλου δεδομένων που έχουμε και της έλλειψης πολλών βίντεο που έγιναν ευρέως διαδεδομένα.

### 4.2.1 Πρόβλεψη με βάση Χαρακτηριστικών του Twitter

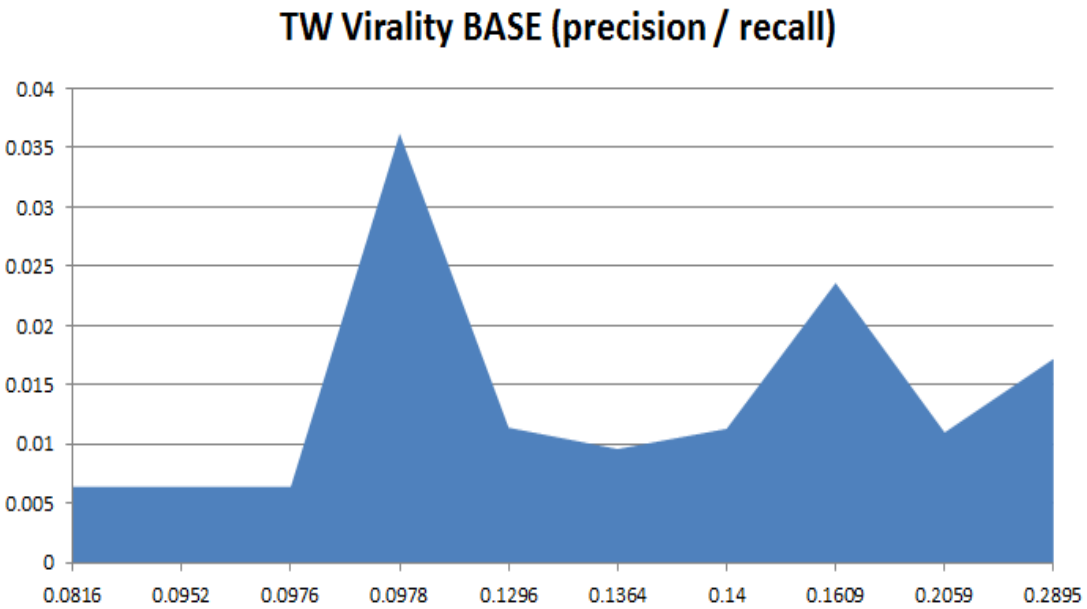
Αρχικά μελετήθηκε η πρόβλεψη της διάδοσης στο Twitter χρησιμοποιώντας χαρακτηριστικά μόνο του Twitter. Τα αποτελέσματα του classifier που χρησιμοποιεί όλα τα χαρακτηριστικά φαίνονται στο Σχήμα 4.2.1.1, ενώ τα χαρακτηριστικά του baseline classifier που χρησιμοποιεί μόνο το retweet count φαίνονται στο Σχήμα 4.2.1.2.

| | Όλα τα χαρακτηριστικά | Μόνο retweet count |
|---|---|---|
| AUC score | 0.0039 | 0.0033 |
| Mean Accuracy | 0.9111 | 0.9109 |

**Πίνακας 4.2.1.1:** Αξιολόγηση της πρόβλεψης της διάδοσης στο Twitter με χρήση μόνο των χαρακτηριστικών του Twitter




**Σχήμα 4.2.1.1:** Πρόβλεψη της διάδοσης στο Twitter με χρήση όλων των χαρακτηριστικών του Twitter



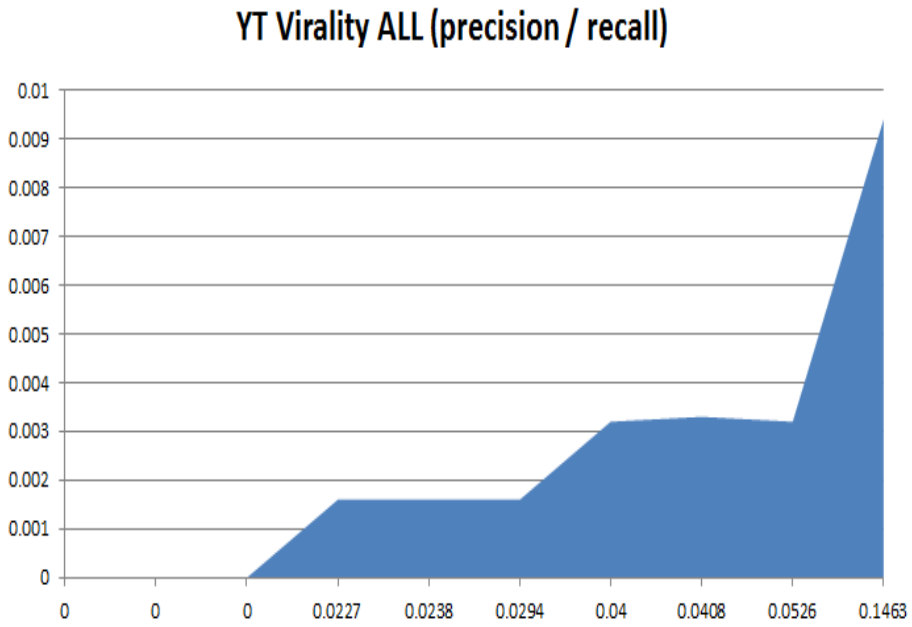

**Σχήμα 4.2.1.2:** Πρόβλεψη της διάδοσης στο Twitter με χρήση μόνο του χαρακτηριστικού retweet count του Twitter

Τα αποτελέσματα των δύο classifiers φαίνονται στο Πίνακα 4.2.1.1 και δείχνουν ότι και οι δύο κατάφεραν να πετύχουν μία πάρα πολύ καλή πρόβλεψη. Ο classifier με όλα τα χαρακτηριστικά είναι ελαφρώς καλύτερος στη πρόβλεψη, χωρίς όμως να υπάρχει σημαντική διαφορά μεταξύ τους.

### 4.2.2 Πρόβλεψη με Βάση την Αλληλεπίδραση των Συστημάτων

Σε αυτό το στάδιο μελετήθηκε πώς μπορεί να γίνει πρόβλεψη της δημοσιότητας του βίντεο στο YouTube με χρήση χαρακτηριστικών του Twitter, καθώς και το πώς μπορεί να γίνει πρόβλεψη της διάδοσης του βίντεο στο Twitter με χρήση χαρακτηριστικών του YouTube.

Πρώτα μελετήθηκε η πρόβλεψη της διάδοσης στο Twitter χρησιμοποιώντας χαρακτηριστικά μόνο του YouTube. Τα αποτελέσματα του classifier που χρησιμοποιεί όλα τα χαρακτηριστικά φαίνονται στο Σχήμα 4.2.2.1, ενώ τα χαρακτηριστικά του baseline classifier που χρησιμοποιεί μόνο το views φαίνονται στο Σχήμα 4.2.2.2.

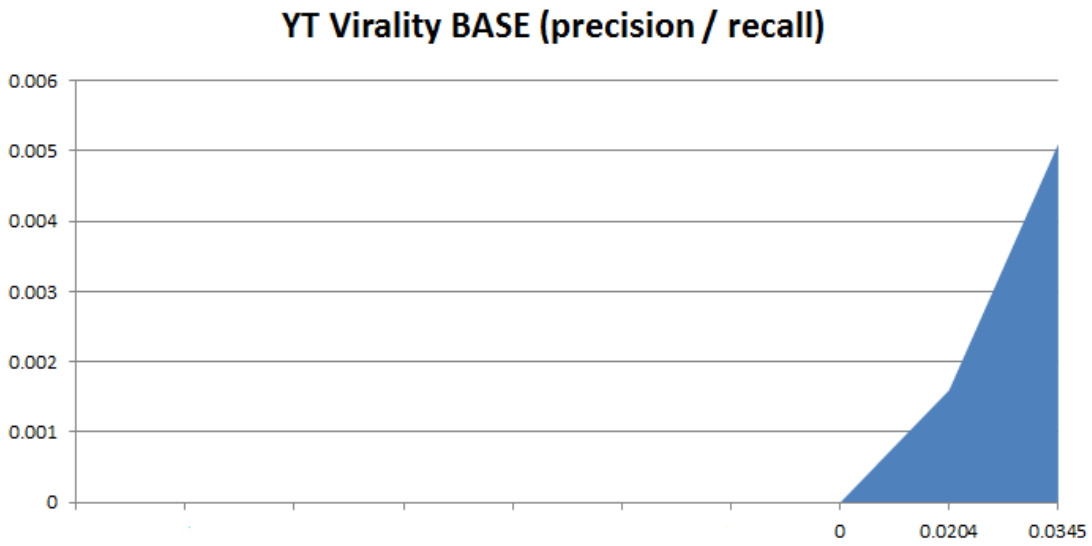



**Σχήμα 4.2.2.1:** Πρόβλεψη της διάδοσης στο Twitter με χρήση όλων των χαρακτηριστικών του YouTube

Τα αποτελέσματα των δύο classifiers φαίνονται στο Πίνακα 4.2.2.1 και δείχνουν ότι χρησιμοποιώντας τα χαρακτηριστικά του YouTube μπορεί να επιτευχθεί μία πάρα πολύ καλή πρόβλεψη της διάδοσης στο Twitter. Ο classifier με όλα τα χαρακτηριστικά είναι ελαφρώς καλύτερος στη πρόβλεψη, χωρίς όμως να υπάρχει σημαντική διαφορά μεταξύ τους.

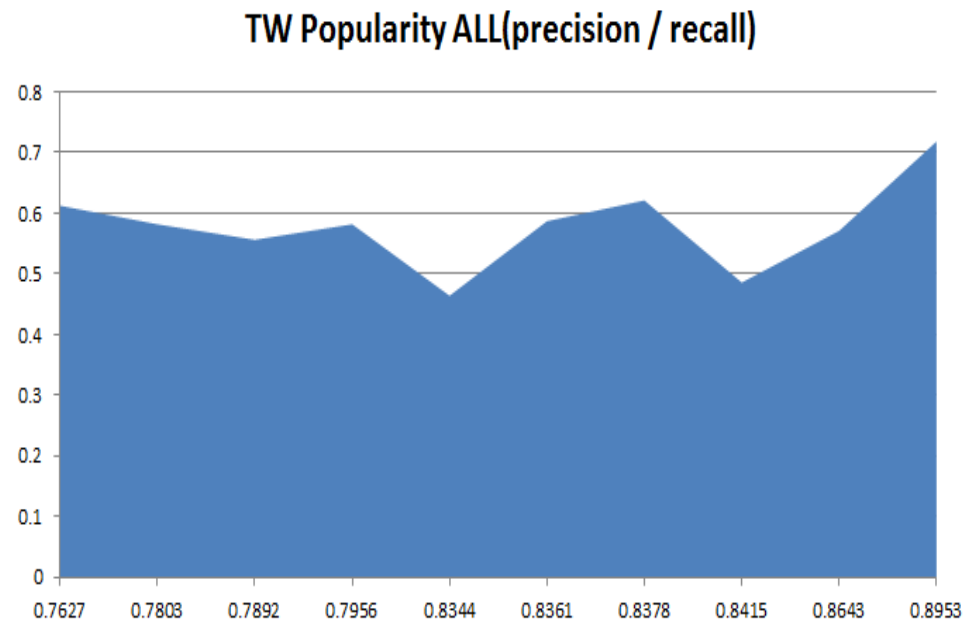
| | Όλα τα χαρακτηριστικά | Μόνο views |
|---|---|---|
| AUC score | 0.0007 | 0.0001 |
| Mean Accuracy | 0.9033 | 0.9009 |

**Πίνακας 4.2.2.1:** Αξιολόγηση της πρόβλεψης της διάδοσης στο Twitter με χρήση μόνο των χαρακτηριστικών του YouTube

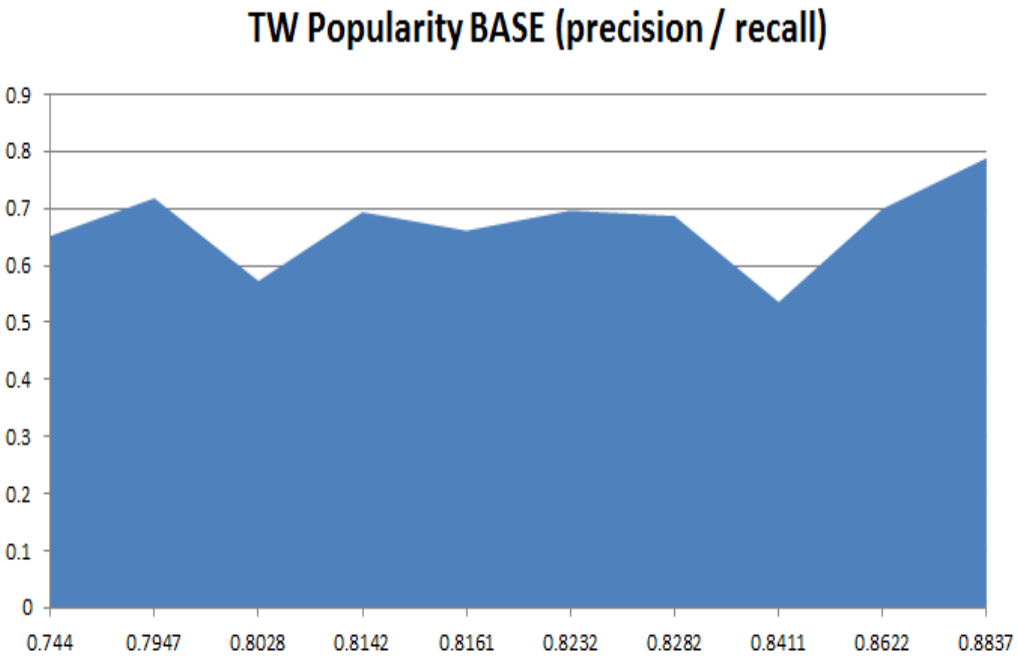



**Σχήμα 4.2.2.2:** Πρόβλεψη της διάδοσης στο Twitter με χρήση μόνο του χαρακτηριστικού views του YouTube

Στη συνέχεια, μελετήθηκε η πρόβλεψη της δημοτικότητας του βίντεο στο YouTube χρησιμοποιώντας χαρακτηριστικά μόνο του Twitter. Τα αποτελέσματα του classifier που χρησιμοποιεί όλα τα χαρακτηριστικά φαίνονται στο Σχήμα 4.2.2.3, ενώ τα χαρακτηριστικά του baseline classifier που χρησιμοποιεί μόνο το retweet count φαίνονται στο Σχήμα 4.2.2.4.




**Σχήμα 4.2.2.3:** Πρόβλεψη της δημοτικότητας στο YouTube με χρήση όλων των χαρακτηριστικών του Twitter




**Σχήμα 4.2.2.4:** Πρόβλεψη της δημοτικότητας στο YouTubeμε χρήση μόνο του χαρακτηριστικού retweet count του Twitter

Τα αποτελέσματα των δύο classifiers φαίνονται στο Πίνακα 4.2.2.2 και δείχνουν ότι χρησιμοποιώντας τα χαρακτηριστικά του Twitter μπορεί να επιτευχθεί μία αρκετά καλή πρόβλεψη της δημοτικότητας στο YouTube, όχι όμως όσο καλή ήταν η πρόβλεψη της διάδοσης του βίντεο στο Twitter με τη χρήση των χαρακτηριστικών του YouTube. Ο classifier με όλα τα χαρακτηριστικά σε αυτή την περίπτωση είναι σημαντικά καλύτερος στη πρόβλεψη από τον baseline classifier.

| | Όλα τα χαρακτηριστικά | Μόνο retweet count |
|---|---|---|
| AUC score | 0.0856 | 0.0837 |
| Mean Accuracy | 0.7341 | 0.6309 |

**Πίνακας 4.2.2.2:** Αξιολόγηση της πρόβλεψης της δημοτικότητας στο YouTube με χρήση μόνο των χαρακτηριστικών του Twitter

### 4.2.3 Πρόβλεψη Κατηγορίας Viral-Popular

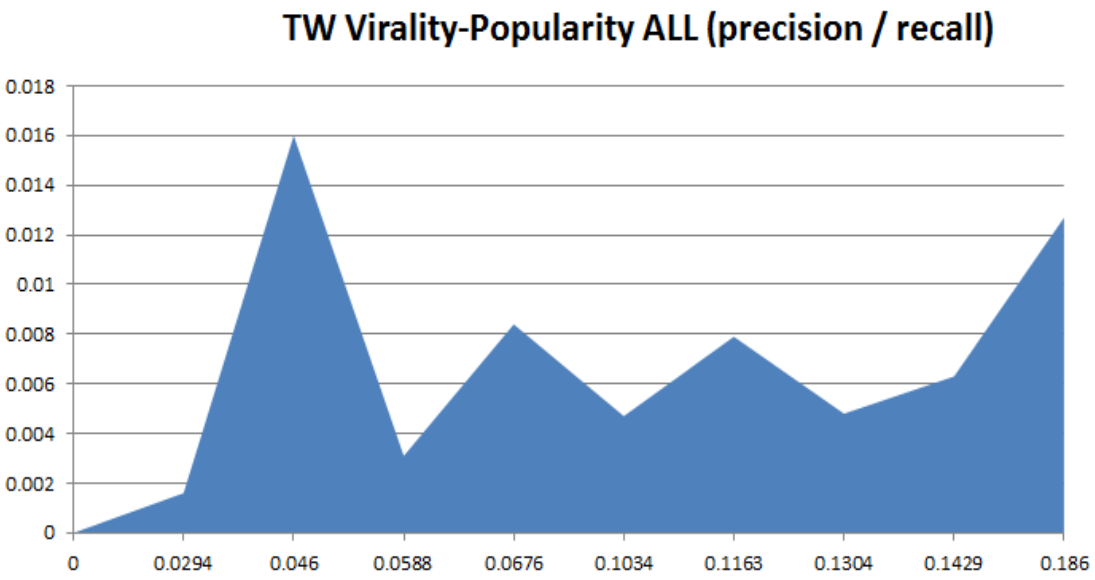
Σε αυτό το στάδιο μελετήθηκε πώς μπορεί να γίνει πρόβλεψη της κλάσης viral-popular, δηλαδή τα βίντεο που έχουν ψηλή δημοτικότητα στο YouTube, αλλά και μεγάλη διάδοση στο Twitter.

Αρχικά, μελετήθηκε η πρόβλεψη της κατηγορίας viral-popular, χρησιμοποιώντας χαρακτηριστικά μόνο του Twitter. Τα αποτελέσματα του classifier που χρησιμοποιεί όλα τα χαρακτηριστικά φαίνονται στο Σχήμα 4.2.3.1, ενώ τα χαρακτηριστικά του baseline classifier που χρησιμοποιεί μόνο το retweet count φαίνονται στο Σχήμα 4.2.3.2.

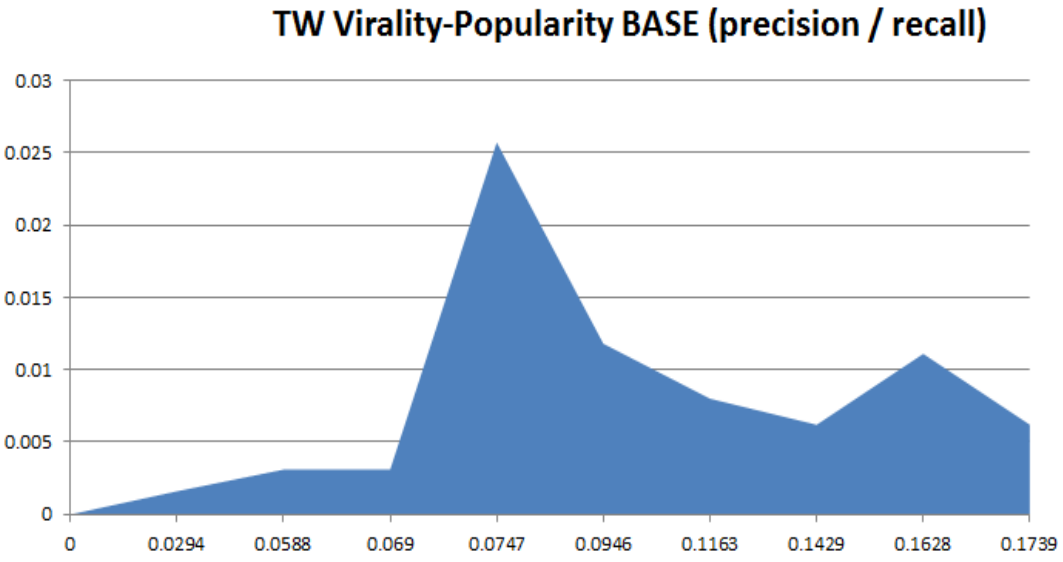
| | Όλα τα χαρακτηριστικά | Μόνο retweet count |
|---|---|---|
| AUC score | 0.0013 | 0.0012 |
| Mean Accuracy | 0.9188 | 0.9137 |

**Πίνακας 4.2.3.1:** Αξιολόγηση της πρόβλεψης της κατηγορίας viral-popular με χρήση μόνο των χαρακτηριστικών του Twitter

Τα αποτελέσματα των δύο classifiers φαίνονται στο Πίνακα 4.2.3.1 και δείχνουν ότι και σε αυτή την περίπτωση οι δύο classifiers κατάφεραν να πετύχουν μία πάρα πολύ καλή πρόβλεψη. Ο classifier με όλα τα χαρακτηριστικά είναι ελαφρώς καλύτερος στη πρόβλεψη, χωρίς όμως να υπάρχει σημαντική διαφορά μεταξύ τους.




**Σχήμα 4.2.3.1:** Πρόβλεψη της κατηγορίας viral-popular με χρήση όλων των χαρακτηριστικών του Twitter



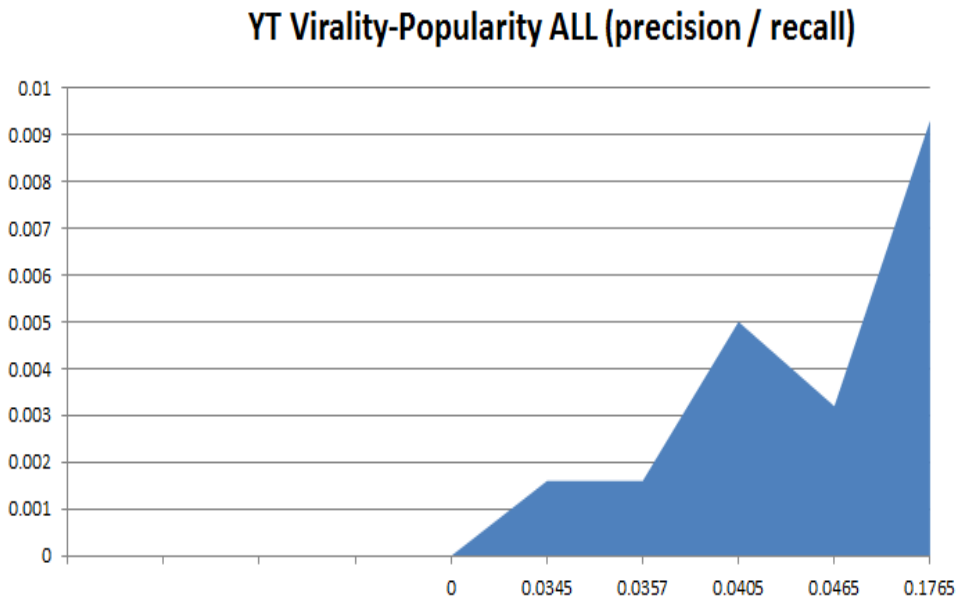

**Σχήμα 4.2.3.2:** Πρόβλεψη της κατηγορίας viral-popular με χρήση μόνο του χαρακτηριστικού retweet count του Twitter

Στη συνέχεια, μελετήθηκε η πρόβλεψη της κατηγορίας viral-popular, χρησιμοποιώντας χαρακτηριστικά μόνο τουYouTube. Τα αποτελέσματα του classifier που χρησιμοποιεί

όλα τα χαρακτηριστικά φαίνονται στο Σχήμα 4.2.3.3, ενώ τα χαρακτηριστικά του baseline classifier που χρησιμοποιεί μόνο το views φαίνονται στο Σχήμα 4.2.3.4.

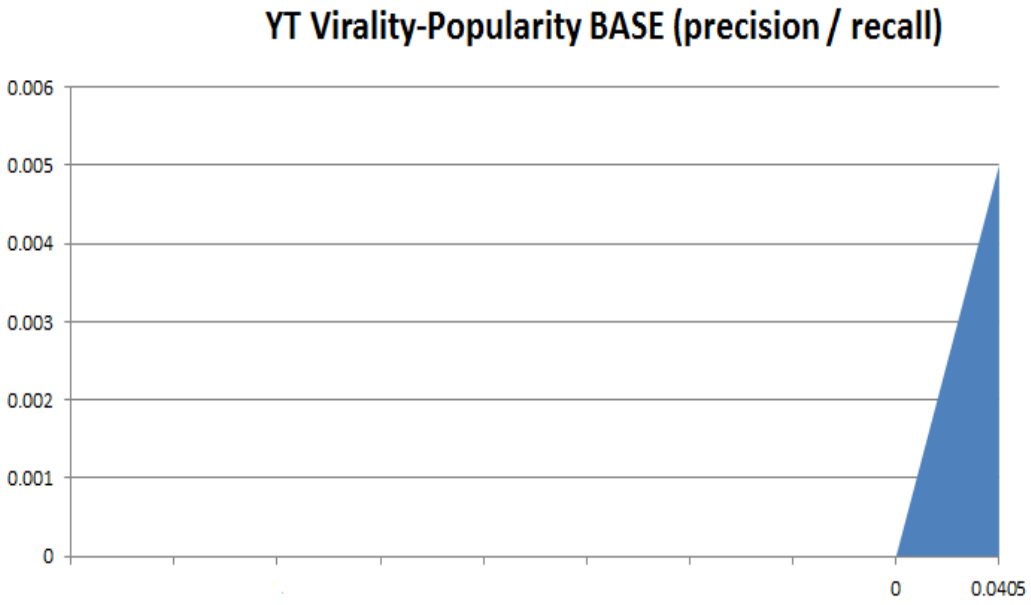
| | Όλα τα χαρακτηριστικά | Μόνο views |
|---|---|---|
| AUC score | 0.0009 | 0.0001 |
| Mean Accuracy | 0.9196 | 0.9158 |

**Πίνακας 4.2.3.2:** Αξιολόγηση της πρόβλεψης της κατηγορίας viral-popular με χρήση μόνο των χαρακτηριστικών του YouTube




**Σχήμα 4.2.3.3:** Πρόβλεψη της κατηγορίας viral-popular με χρήση όλων των χαρακτηριστικών του YouTube

Τα αποτελέσματα των δύο classifiers φαίνονται στο Πίνακα 4.2.3.2 και δείχνουν ότι οι δύο classifiers κατάφεραν να πετύχουν μία πάρα πολύ καλή πρόβλεψη. Ο classifier με όλα τα χαρακτηριστικά είναι ελαφρώς καλύτερος στη πρόβλεψη, όμως, όπως και στις προηγούμενες περιπτώσεις, δεν υπάρχει σημαντική διαφορά μεταξύ τους.



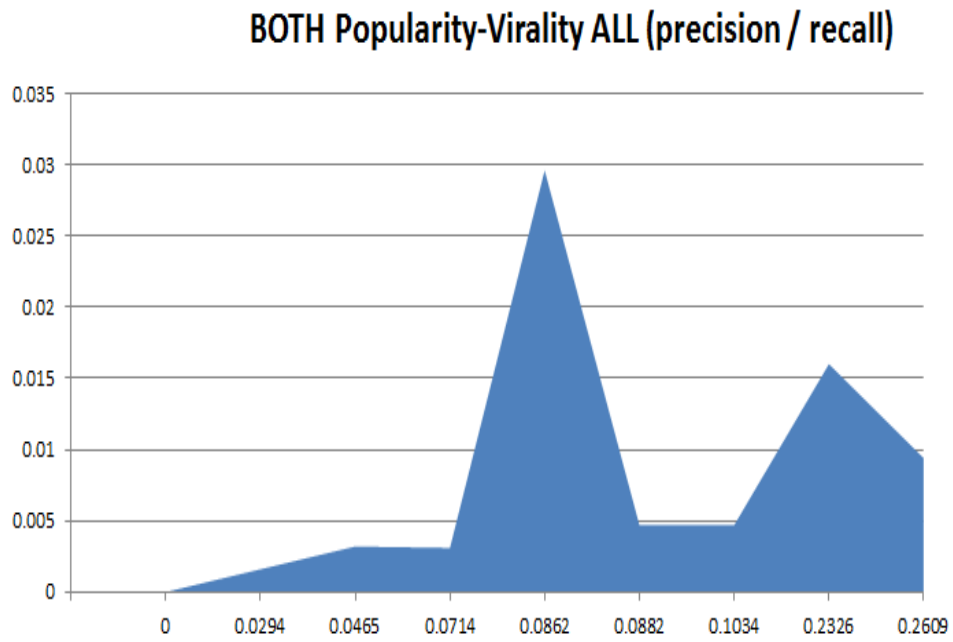

**Σχήμα 4.2.3.4:** Πρόβλεψη της κατηγορίας viral-popular με χρήση μόνο του χαρακτηριστικού views του YouTube

Τέλος, μελετήθηκε η πρόβλεψη της κατηγορίας viral-popular, χρησιμοποιώντας όλα τα χαρακτηριστικά και του Twitter και του YouTube. Τα αποτελέσματα του classifier που χρησιμοποιεί όλα τα χαρακτηριστικά φαίνονται στο Σχήμα 4.2.3.5, ενώ τα χαρακτηριστικά του baseline classifier που χρησιμοποιεί μόνο το retweet count και το views φαίνονται στο Σχήμα 4.2.3.6.

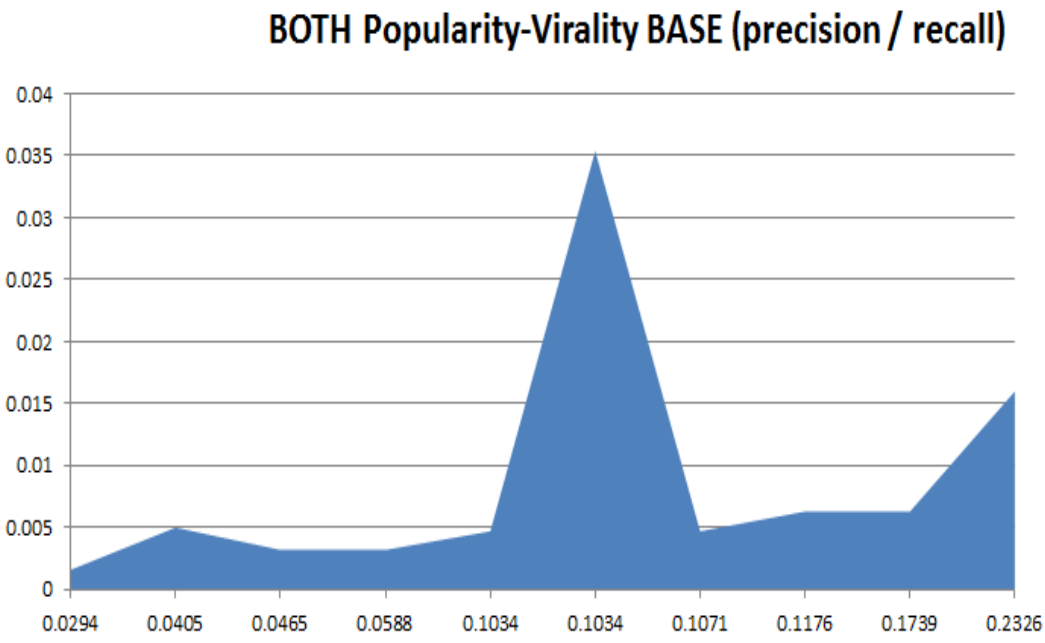
| | Όλα τα χαρακτηριστικά | Μόνο retweet count και views |
|---|---|---|
| AUC score | 0.0022 | 0.0014 |
| Mean Accuracy | 0.9211 | 0.9210 |

**Πίνακας 4.2.3.3:** Αξιολόγηση της πρόβλεψης της κατηγορίας viral-popular με χρήση όλων των χαρακτηριστικών

Τα αποτελέσματα των δύο classifiers φαίνονται στο Πίνακα 4.2.3.3 και δείχνουν ότι οι δύο classifiers κατάφεραν να πετύχουν μία πάρα πολύ καλή πρόβλεψη. Ο classifier με όλα τα χαρακτηριστικά είναι ελαφρώς καλύτερος στη πρόβλεψη. Όπως και στις προηγούμενες περιπτώσεις όμως, δεν υπάρχει σημαντική διαφορά μεταξύ τους.




**Σχήμα 4.2.3.5:** Πρόβλεψη της κατηγορίας viral-popular με χρήση όλων των χαρακτηριστικών




**Σχήμα 4.2.3.6:** Πρόβλεψη της κατηγορίας viral-popular με χρήση μόνο των χαρακτηριστικών retweet count και views

Τα συνοπτικά αποτελέσματα όλων των προβλέψεων φαίνονται στο Σχήμα 4.2.3.7. Όπως είχαμε αναφέρει και προηγουμένως, η πρόβλεψη της δημοτικότητας του βίντεο στο YouTube χρησιμοποιώντας χαρακτηριστικά του YouTube δεν είναι έγκυρη. Με βάση τα αποτελέσματα μπορούμε να διακρίνουμε ότι:

1) Η χρήση των χαρακτηριστικών του YouTube οδηγεί σε πολύ καλή πρόβλεψη της διάδοσης στο Twitter (Virality)
2) Η χρήση των χαρακτηριστικών του Twitter οδηγεί σε μια σχετικά καλή πρόβλεψη της δημοτικότητας του βίντεο στο YouTube (Popularity)
3) Η χρήση όλων των χαρακτηριστικών οδηγεί στη καλύτερη πρόβλεψη

| Χαρακτηριστικά | Virality | Popularity | Virality-Popularity |
|---|---|---|---|
| Twitter | AUC score = 0.0039<br>mean ACC = 0.9111 | AUC score = 0.0756<br>mean ACC = 0.7341 | AUC score = 0.0013<br>mean ACC = 0.9189 |
| YouTube | AUC score = 0.0007<br>mean ACC = 0.9033 | AUC score = 0<br>mean ACC = 1 | AUC score = 0.0009<br>mean ACC = 0.9196 |
| Twitter & YouTube | AUC score = 0.0052<br>mean ACC = 0.9079 | AUC score = 0<br>mean ACC = 1 | AUC score = 0.0022<br>mean ACC = 0.9211 |

**Σχήμα 4.2.3.7:** Τα αποτελέσματα όλων των προβλέψεων του classifier

### 4.3 Σημαντικότητα Χαρακτηριστικών

Η μελέτη της σημαντικότητας των χαρακτηριστικών βασίστηκε στη μελέτη της αλληλεπίδρασης μεταξύ των συστημάτων, αλλά και της πρόβλεψης της κατηγορίας viral-popular. Επίσης έγινε μελέτη της σημαντικότητας των χαρακτηριστικών στην κατηγορία μουσικής, όπου ανήκαν τα περισσότερα βίντεο. Θέσαμε όπως και προηγουμένως το παράθυρο εκπαίδευσης να είναι 3 μέρες και το παράθυρο επαλήθευσης να είναι οι αμέσως επόμενες 3 μέρες.

Με βάση τον τρόπο που ο classifier αποφασίζει την κατηγορία του κάθε βίντεο, είμαστε σε θέση να γνωρίζουμε ποια χαρακτηριστικά είναι αυτά που παίζουν το σημαντικότερο ρόλο. Με βάση τα αποτελέσματα της σημαντικότητας, γίνεται κανονικοποίηση των τιμών και θέτουμε το σημαντικότερο χαρακτηριστικό να έχει σημασία στη απόφαση 100%, με τα υπόλοιπα χαρακτηριστικά να προσαρμόζονται ανάλογα.

Στο Πίνακα 4.3.1 φαίνεται η σημαντικότητα των χαρακτηριστικών του Twitter στη πρόβλεψη της δημοτικότητας στο YouTube. Φαίνεται ότι τα δύο σημαντικότερα χαρακτηριστικά είναι το followers count και followings count, τα οποία έχουν πολύ μεγάλη διαφορά από το τρίτο σημαντικότερο χαρακτηριστικό που είναι το retweet count.

| **Χαρακτηριστικό** | **Σημαντικότητα (%)** |
|---|---|
| Followers count | 100 |
| Followings count | 98.2 |
| Retweet count | 32.6 |
| Favorite count | 3.3 |
| Is verified celebrity | 2.4 |
| Is retweet | 0.6 |
| Is retweeted | 0 |
| Is favorited | 0 |

**Πίνακας 4.3.1:** Η σημαντικότητα των χαρακτηριστικών του Twitter στη πρόβλεψη της δημοτικότητας στο YouTube

| **Χαρακτηριστικό** | **Σημαντικότητα (%)** |
|---|---|
| view count | 100 |
| Channel ID | 83.8 |
| like count | 81.7 |
| comment count | 69.9 |
| Dislike count | 57.4 |
| Category ID | 25.2 |
| Favorite count | 0 |

**Πίνακας 4.3.2:** Η σημαντικότητα των χαρακτηριστικών του YouTube στη πρόβλεψη της διάδοσης στο Twitter

Στο Πίνακα 4.3.2 φαίνεται η σημαντικότητα των χαρακτηριστικών του YouTube στη πρόβλεψη της διάδοσης του βίντεο στο Twitter. Εδώ, σε αντίθεση με τα χαρακτηριστικά του Twitter που υπήρχε μεγάλη διαφορά σημαντικότητας, φαίνεται ότι όλα τα χαρακτηριστικά είναι αρκετά σημαντικά, με κυριότερο το view count,ακολουθούμενο από το channel ID και το like count .

Στον Πίνακα 4.3.3 φαίνεται η σημαντικότητα όλων των χαρακτηριστικών στην πρόβλεψη της κατηγορίας viral-popular. Αυτό που ξεχωρίζει είναι το γεγονός ότι τα χαρακτηριστικά του YouTube επηρεάζουν σε πολύ μεγαλύτερο βαθμό τον classifier από τα χαρακτηριστικά του Twitter, αφού τα πέντε σημαντικότερα χαρακτηριστικά είναι του YouTube.

| **Χαρακτηριστικό** | **Σημαντικότητα (%)** | | **Χαρακτηριστικό** | **Σημαντικότητα (%)** |
|---|---|---|---|---|
| Comment count | 100 | | Followers count | 21.9 |
| View count | 98 | | Category ID | 19.4 |
| Dislike count | 82.3 | | Favorite count | 10.3 |
| Like count | 67.1 | | Is verified celebrity | 2.6 |
| Channel ID | 56.5 | | Is retweeted | 0.4 |
| Followings count | 32.1 | | Is retweet | 0 |
| Retweet count | 30.6 | | Is favorited | 0 |

**Πίνακας 4.3.3:** Η σημαντικότητα όλων των χαρακτηριστικών στη πρόβλεψη της κατηγορίας viral-popular

Στο πίνακα 4.3.4 φαίνεται η σημαντικότητα των χαρακτηριστικών του Twitter στη πρόβλεψη της δημοτικότητας στο YouTube όσο αφορά την κατηγορία μουσικής. Φαίνεται ότι η σειρά της σημαντικότητας των χαρακτηριστικών δεν αλλάζει. Αυτό που αλλάζει όμως είναι το ποσοστό σημαντικότητας των χαρακτηριστικών, με σημαντικότερη την πτώση του followings count στο 68.3%. Το followers count παραμένει το σημαντικότερο.

Στο Πίνακα 4.3.5 φαίνεται η σημαντικότητα των χαρακτηριστικών του YouTube στη πρόβλεψη της διάδοσης των βίντεο κατηγορίας μουσικής στο Twitter. Εδώ, σε αντίθεση με τα χαρακτηριστικά του Twitter όπου δεν υπήρχε μεταβολή στη σειρά σημαντικότητας των χαρακτηριστικών, φαίνεται ότι η κατηγορία επηρεάζει τη σημαντικότητα των χαρακτηριστικών. Πιο συγκεκριμένα, μπορεί το view count να παραμένει το σημαντικότερο χαρακτηριστικό, όμως μια σημαντική αλλαγή είναι ότι το comment count

ανεβαίνει στη δεύτερη θέση και το channel ID, που ήταν δεύτερο, πέφτει στη τέταρτη θέση.

| **Χαρακτηριστικό** | **Σημαντικότητα (%)** |
|---|---|
| Followers count | 100 |
| Followings count | 68.3 |
| Retweet count | 32.4 |
| Favorite count | 8.6 |
| Is verified celebrity | 5.1 |
| Is retweet | 0.2 |
| Is retweeted | 0 |
| Is favorited | 0 |

**Πίνακας 4.3.4:** Η σημαντικότητα των χαρακτηριστικών του Twitter στη πρόβλεψη της δημοτικότητας των βίντεο μουσικής στο YouTube

| **Χαρακτηριστικό** | **Σημαντικότητα (%)** |
|---|---|
| view count | 100 |
| comment count | 84.6 |
| like count | 84 |
| Channel ID | 74.5 |
| Dislike count | 54.7 |
| Favorite count | 0 |

**Πίνακας 4.3.5:** Η σημαντικότητα των χαρακτηριστικών του YouTube στη πρόβλεψη της διάδοσης των βίντεο μουσικής στο Twitter

Τέλος, στον Πίνακα 4.3.6 φαίνεται η σημαντικότητα όλων των χαρακτηριστικών στην πρόβλεψη της κατηγορίας viral-popular. Παρά το γεγονός ότι υπάρχουν μεταβολές στη σημαντικότητα των χαρακτηριστικών, τα χαρακτηριστικά του YouTube συνεχίζουν να είναι αυτά που επηρεάζουν σε μεγαλύτερο βαθμό τη πρόβλεψη.

| **Χαρακτηριστικό** | **Σημαντικότητα (%)** | | **Χαρακτηριστικό** | **Σημαντικότητα (%)** |
|---|---|---|---|---|
| Comment count | 100 | | Followers count | 26.2 |
| Dislike count | 92.8 | | Is verified celebrity | 3.9 |
| View count | 76.7 | | Favorite count | 2.9 |
| Like count | 64.9 | | Is retweeted | 1.4 |
| Channel ID | 53.6 | | Is retweet | 0 |
| Retweet count | 29.7 | | Is favorited | 0 |
| Followings count | 29.2 | | | |

**Πίνακας 4.3.6:** Η σημαντικότητα όλων των χαρακτηριστικών στη πρόβλεψη της κατηγορίας viral-popular των βίντεο μουσικής

### 4.4 Σημαντικότητα Παραθύρων Εκπαίδευσης και Επαλήθευσης

Σε αυτό το στάδιο έγινε μελέτη της επιρροής του μεγέθους των παραθύρων εκπαίδευσης και επαλήθευσης στην πρόβλεψη της κατηγορίας του βίντεο. Η μελέτη αυτή βασίστηκε και πάλι στη μελέτη της αλληλεπίδρασης μεταξύ των συστημάτων, αλλά και της πρόβλεψης της κατηγορίας viral-popular.

| | Παράθυρο Επαλήθευσης (μέρες) | 1 | 3 | 7 |
|---|---|---|---|---|
| Παράθυρο Εκπαίδευσης (μέρες) | | | | |
| 1 | | F1 = 0. 6494<br>ACC = 0.744 | F1 = 0. 6792<br>ACC = 0.7183 | F1 = 0. 6984<br>ACC = 0.7335 |
| 3 | | F1 = 0. 6542<br>ACC = 0.7557 | F1 = 0. 6778<br>ACC = 0.734 | F1 = 0.6602<br>ACC = 0.7444 |
| 7 | | F1 = 0.644<br>ACC = 0.7788 | F1 = 0.6652<br>ACC = 0.7663 | F1 = 0.7056<br>ACC = 0.7471 |

**Πίνακας 4.4.1:** Μέγεθος παραθύρων εκπαίδευσης και επαλήθευσης για πρόβλεψη της δημοτικότητας στο YouTube χρησιμοποιώντας χαρακτηριστικά του Twitter

Αρχικά μελετήθηκε η επιρροή του μεγέθους των παραθύρων στη πρόβλεψη της δημοτικότητας του βίντεο στο YouTube με τη χρήση χαρακτηριστικών του Twitter. Τα αποτελέσματα στον Πίνακα 4.4.1 δείχνουν ότι μπορεί να επιτευχθεί μια αξιόλογη πρόβλεψη, η οποία γίνεται καλύτερη όταν τα παράθυρα είναι μεγαλύτερα σε μέγεθος. Όμως, με δεδομένα εκπαίδευσης μόνο μιας μέρας μπορεί να πραγματοποιηθεί μια εξίσου καλή πρόβλεψη, όπως με μεγαλύτερα παράθυρα.

Ακολούθως, μελετήθηκε η επιρροή του μεγέθους των παραθύρων στη πρόβλεψη της διάδοσης του βίντεο στο Twitter με τη χρήση χαρακτηριστικών του YouTube. Τα αποτελέσματα στον Πίνακα 4.4.2 δείχνουν ότι μπορεί να επιτευχθεί μια πάρα πολύ καλή πρόβλεψη, η οποία γίνεται καλύτερη όταν τα παράθυρα είναι μεγαλύτερα σε μέγεθος. Όμως, με δεδομένα εκπαίδευσης μόνο μιας μέρας μπορεί να πραγματοποιηθεί μια εξίσου καλή πρόβλεψη, όπως με μεγαλύτερα παράθυρα.

| | Παράθυρο Επαλήθευσης (μέρες) | 1 | 3 | 7 |
|---|---|---|---|---|
| Παράθυρο Εκπαίδευσης (μέρες) | | | | |
| 1 | | F1 = 0.0182<br>ACC = 0.907 | F1 = 0.0068<br>ACC = 0.8932 | F1 = 0.0096<br>ACC = 0.8464 |
| 3 | | F1 = 0.0078<br>ACC = 0.9233 | F1 = 0.0044<br>ACC = 0.9033 | F1 = 0.0076<br>ACC = 0.93 |
| 7 | | F1 = 0.0096<br>ACC = 0.9374 | F1 = 0.0022<br>ACC = 0.9401 | F1 = 0.0282<br>ACC = 0.9436 |

**Πίνακας 4.4.2:** Μέγεθος παραθύρων εκπαίδευσης και επαλήθευσης για πρόβλεψη της διάδοσης στο Twitter χρησιμοποιώντας χαρακτηριστικά του YouTube

Τέλος, μελετήθηκε η επιρροή του μεγέθους των παραθύρων στη πρόβλεψη της κατηγορίας viral-popular με τη χρήση όλων των χαρακτηριστικών. Τα αποτελέσματα στον Πίνακα 4.4.3 δείχνουν ότι με τη χρήση όλων των χαρακτηριστικών γίνεται η καλύτερη πρόβλεψη, ειδικότερα αν τα παράθυρα είναι μεγαλύτερα σε μέγεθος. Όμως, με

δεδομένα εκπαίδευσης μόνο μιας μέρας μπορεί να πραγματοποιηθεί μια εξίσου καλή πρόβλεψη, όπως με μεγαλύτερα παράθυρα.

| | Παράθυρο Επαλήθευσης (μέρες) | 1 | 3 | 7 |
|---|---|---|---|---|
| Παράθυρο Εκπαίδευσης (μέρες) | | | | |
| 1 | | F1 = 0.0356<br>ACC = 0.9323 | F1 = 0.0204<br>ACC = 0.9198 | F1 = 0.0262<br>ACC = 0.881 |
| 3 | | F1 = 0.0132<br>ACC = 0.945 | F1 = 0.0126<br>ACC = 0.921 | F1 = 0.0076<br>ACC = 0.93 |
| 7 | | F1 = 0.0136<br>ACC = 0.9517 | F1 = 0.0136<br>ACC = 0.9544 | F1 = 0.0206<br>ACC = 0.9531 |

**Πίνακας 4.4.3:** Μέγεθος παραθύρων εκπαίδευσης και επαλήθευσης για πρόβλεψη της κατηγορίας viral-popular χρησιμοποιώντας όλα τα χαρακτηριστικά

# Κεφάλαιο 5

# Συμπεράσματα



## 5.1 Γενικά Συμπεράσματα

Η μελέτη των χαρακτηριστικών που επηρεάζουν τη διάδοση του περιεχομένου στα κοινωνικά δίκτυα είναι ένας τομέας στον οποίο γίνεται μεγάλη έρευνα. Σε αυτή τη διπλωματική εργασία έγινε μελέτη των χαρακτηριστικών που επηρεάζουν τη διάδοση του περιεχομένου. Ιδιαίτερη έμφαση δόθηκε στη μελέτη της αλληλεπίδρασης μεταξύ των συστημάτων, αλλά και στη μελέτη των βίντεο που είναι ευρέως διαδεδομένα στο Twitter, αλλά ταυτόχρονα έχουν και μεγάλη δημοτικότητα στο YouTube.

Μέσα από τη μελέτη αυτή, παρατηρήσαμε ότι χρησιμοποιώντας τα χαρακτηριστικά του YouTube μπορεί να γίνει μια πολύ ακριβής πρόβλεψη της διάδοσης του βίντεο στο Twitter, ενώ χρησιμοποιώντας τα χαρακτηριστικά του Twitter μπορεί να γίνει μια σχετικά καλή πρόβλεψη της δημοτικότητας του βίντεο στο YouTube. Παρατηρήσαμε όμως, ότι ο συνδυασμός όλων των χαρακτηριστικών πετυχαίνει την καλύτερη πρόβλεψη.

Επίσης δόθηκε σημασία στη σημαντικότητα των χαρακτηριστικών και στην επιρροή που έχουν στη διάδοση του περιεχομένου στα δύο κοινωνικά δίκτυα, όπου παρατηρήσαμε ότι τα χαρακτηριστικά του YouTube επηρεάζουν σε πολύ μεγαλύτερο βαθμό τη διάδοση του βίντεο από τα χαρακτηριστικά του Twitter.

Μια σημαντική συνεισφορά της μελέτης αυτής είναι και η παρατήρηση ότι τα χαρακτηριστικά ενός βίντεο επηρεάζουν σε διαφορετικό βαθμό τη διάδοση του ανάλογα της κατηγορίας που ανήκει ένα βίντεο του YouTube. Εξ' όσο γνωρίζουμε, δεν υπάρχει κάποια παρόμοια μελέτη που ασχολήθηκε με τη επιρροή των χαρακτηριστικών στις διαφορετικές κατηγορίες βίντεο του YouTube.

Μια άλλη σημαντική παρατήρηση που έγινε αφορά το μέγεθος του παραθύρου εκπαίδευσης και επαλήθευσης, όπου φάνηκε το ότι όσο μεγαλύτερα τα παράθυρα, τόσο καλύτερη η πρόβλεψη. Παρ' όλα αυτά, χρησιμοποιώντας δεδομένα εκπαίδευσης μόλις μιας μέρας μπορεί να γίνει μία αρκετά καλή πρόβλεψη η οποία είναι πολύ κοντά στις προβλέψεις των μεγαλύτερων παραθύρων.

Τέλος, αξίζει να αναφερθεί ότι λόγω τεχνικών προβλημάτων δεν κατέστη δυνατό να έχουμε ολόκληρο το σύνολο δεδομένων στη διάθεση μας και επομένως έπρεπε να δουλέψουμε με ένα πολύ μικρότερο σύνολο δεδομένων, το μέγεθος του οποίου ίσως επηρέασε μερικώς τα αποτελέσματα μας. Ακόμα ένα πρόβλημα ήταν το γεγονός ότι τα δεδομένα που είχαμε για το YouTube τα είχαμε συλλέξει σε μεταγενέστερο στάδιο από τις ημερομηνίες συλλογής των δεδομένων από το Twitter. Αυτό μας εμπόδισε από το να χρησιμοποιήσουμε τα δεδομένα του YouTube για να μελετήσουμε τη δημοτικότητα του βίντεο στο YouTube.

### 5.2 Μελλοντική Εργασία

Σε μεταγενέστερο στάδιο θα πρέπει να χρησιμοποιηθεί ένα μεγαλύτερο σύνολο δεδομένων έτσι ώστε να είμαστε πιο σίγουροι για τα αποτελέσματα. Έχοντας το μεγαλύτερο σύνολο δεδομένων θα μπορεί να γίνει μελέτη για περισσότερες κατηγορίες βίντεο του YouTube και να εξαχθούν χρήσιμα συμπεράσματα για την κάθε κατηγορία.

Εκτός από τη μελέτη της διάδοσης των βίντεο, μια πολύ ενδιαφέρον μελέτη που μπορεί να γίνει είναι για τη διάδοση περιεχομένου που αφορούν ειδήσεις. Πιο συγκεκριμένα, είναι ενδιαφέρον να μελετηθεί η διάδοση ειδήσεων που αφορούν μεγάλα και αναπάντεχα γεγονότα, όπως αθλητικών ειδήσεων (π.χ. η κατάκτηση του πρωταθλήματος από μια μικρή ομάδα όπως η Λέστερ) ή τρομοκρατικών επιθέσεων (π.χ. τρομοκρατικές επιθέσεις στο Παρίσι), οι οποίες έγιναν viral ανά το παγκόσμιο.

# Βιβλιογραφία


[1] Zephoria Digital Marketing, "The Top 20 Valuable Facebook Statistics," [Online]. Available: https://zephoria.com/top-15-valuable-facebook-statistics/. [Accessed 27 April 2016].

[2] Wikipedia, "Twitter," [Online]. Available: https://en.wikipedia.org/wiki/Twitter. [Accessed 27 April 2016].

[3] Li, Haitao, Xiaoqiang Ma, Feng Wang, Jiangchuan Liu, and Ke Xu, "On popularity prediction of videos shared in online social networks," in *Proceedings of the 22nd ACM international conference on Conference on information & knowledge management*, 2013.

[4] Zhang, Lei, Feng Wang, and Jiangchuan Liu, "Understand Instant Video Clip Sharing on Mobile Platforms: Twitter's Vine as a Case Study," in *Proceedings of Network and Operating System Support on Digital Audio and Video Workshop*, 2014.

[5] Ma, Xiaoqiang, Haiyang Wang, Haitao Li, Jiangchuan Liu, and Hongbo Jiang, "Exploring sharing patterns for video recommendation on YouTube-like social media," *Multimedia Systems*, vol. 20, no. 6, pp. 675-691, 2014.

[6] Wikipedia, "Long Tail," [Online]. Available: https://en.wikipedia.org/wiki/Long_tail. [Accessed 19 May 2016].

[7] eBiz MBA, "Top 15 Most Popular Social Networking Sites," [Online]. Available: http://www.ebizmba.com/articles/social-networking-websites. [Accessed 27 April 2016].

[8] Wikipedia, "Map-Reduce," [Online]. Available: https://en.wikipedia.org/wiki/MapReduce. [Accessed 27 April 2016].

[9] Wikipedia, "Google File System," [Online]. Available: https://en.wikipedia.org/wiki/Google_File_System. [Accessed 27 April 2016].

[10] Apache Software Foundation, [Online]. Available: http://hadoop.apache.org/docs/current/. [Accessed 27 April 2016].

[11] Ghemawat, Sanjay, Howard Gobioff, and Shun-Tak Leung, "The Google file system," *ACM SIGOPS operating systems review*, vol. 37, no. 5, pp. 29-43, 2003.

[12] Dean, Jeffrey, and Sanjay Ghemawat, "MapReduce: simplified data processing on large clusters," *Communications of the ACM,* vol. 51, no. 1, pp. 107-113, 2008.

[13] Glenn K. Lockwood, "Conceptual Overview of Map-Reduce and Hadoop," [Online]. Available: http://www.glennklockwood.com/data-intensive/hadoop/overview.html. [Accessed 26 May 2016].

[14] Apache Software Foundation, "HDFS Architecture," [Online]. Available: http://hadoop.apache.org/docs/current/hadoop-project-dist/hadoop-hdfs/HdfsDesign.html. [Accessed 27 April 2016].

[15] Jerome H. Friedman, "Greedy Function Approximation: A Gradient Boosting Machine," *The Annals of Statistics,* vol. 29, no. 5, pp. 1189-1232, 2001.

[16] scikit-learn, "Gradient Boosting Classifier," [Online]. Available: http://scikit-learn.org/stable/modules/generated/sklearn.ensemble.GradientBoostingClassifier.html. [Accessed 28 April 2016].

[17] Christodoulou, George, Chryssis Georgiou, and George Pallis, "The role of twitter in youtube videos diffusion," in *Web Information Systems Engineering-WISE 2012*, 2012.

[18] Twitter, "Timezones," [Online]. Available: https://dev.twitter.com/ads/basics/timezones. [Accessed 27 April 2016].

[19] Vallet, David, Shlomo Berkovsky, Sebastien Ardon, Anirban Mahanti, and Mohamed Ali Kafaar, "Characterizing and Predicting Viral-and-Popular Video Content," in *Proceedings of the 24th ACM International on Conference on Information and Knowledge Management*, 2015.

[20] Wikipedia, "Precision and recall," [Online]. Available: https://en.wikipedia.org/wiki/Precision_and_recall. [Accessed 27 April 2016].

[21] Efstathiades, Hariton, Demetris Antoniades, George Pallis, and Marios D. Dikaiakos, "Identification of Key Locations based on Online Social Network Activity," in *Proceedings of the 2015 IEEE/ACM International Conference on Advances in Social Networks Analysis and Mining 2015*, 2015.

[22] Twitter, "About Twitter's link service," [Online]. Available: https://support.twitter.com/articles/109623. [Accessed 27 April 2016].

[23] A. Ghosh, "Tool URL expander," [Online]. Available: https://github.com/srccodes/tool-url-expander. [Accessed 27 April 2016].

[24] Google Developers, "YouTube API," [Online]. Available: https://developers.google.com/youtube/. [Accessed 6 May 2016].

[25] Instant SSL, "HTTP VS. HTTPS," [Online]. Available: https://www.instantssl.com/ssl-certificate-products/https.html. [Accessed 6 May 2016].

[26] readwrite, "YouTube short link sevice," [Online]. Available: http://readwrite.com/2009/12/21/youtube_gets_shorter_links_too/. [Accessed 27 April 2016].

[27] w3schools, "JSON," [Online]. Available: http://www.w3schools.com/json/. [Accessed 7 May 2016].

[28] Khana Academy, "Merge Sort," [Online]. Available: https://www.khanacademy.org/computing/computer-science/algorithms/merge-sort/a/overview-of-merge-sort. [Accessed 7 May 2016].

[29] Oracle, "Hash Map," [Online]. Available: https://docs.oracle.com/javase/7/docs/api/java/util/HashMap.html. [Accessed 7 May 2016].

[30] Wikipedia, "Supervised Learning," [Online]. Available: https://en.wikipedia.org/wiki/Supervised_learning. [Accessed 27 May 2016].

[31] ML Wiki, "ROC Analysis," [Online]. Available: http://mlwiki.org/index.php/ROC_Analysis. [Accessed 27 May 2016].

# Παράρτημα Α

Σε αυτό το παράρτημα παρουσιάζεται ο κώδικας όπου γίνεται η εξαγωγή των πραγματικών συνδέσμων από τους συνδέσμους σε μορφή σμίκρυνσης έτσι ώστε να κρατήσουμε μόνο τους συνδέσμους του YouTube. Έχει υλοποιηθεί για κατανεμημένο περιβάλλον Hadoop, έκδοσης 2.6.3.

```
import org.apache.hadoop.conf.Configuration;
import org.apache.hadoop.fs.Path;
import org.apache.hadoop.io.LongWritable;
import org.apache.hadoop.io.NullWritable;
import org.apache.hadoop.io.Text;
import org.apache.hadoop.mapreduce.Job;
import org.apache.hadoop.mapreduce.Mapper;
import org.apache.hadoop.mapreduce.Reducer;
import org.apache.hadoop.mapreduce.lib.input.FileInputFormat;
import org.apache.hadoop.mapreduce.lib.input.TextInputFormat;
import org.apache.hadoop.mapreduce.lib.output.FileOutputFormat;
import org.apache.hadoop.mapreduce.lib.output.TextOutputFormat;

import java.io.IOException;
import java.net.HttpURLConnection;
import java.net.Proxy;
import java.net.URL;

public class ExpandTinyURLs {

	public static class UrlExpander {

		/**
		 * @author Abhijit Ghosh
		 * @version 1.0
		 * @github https://github.com/srccodes/tool-url-expander
		 */
		public static String expandUrl(String shortenedUrl)
				throws IOException {
			URL url = new URL(shortenedUrl);
			// open connection
			HttpURLConnection httpURLConnection =
					(HttpURLConnection) url
					.openConnection(Proxy.NO_PROXY);

			// stop following browser redirect
			httpURLConnection.setInstanceFollowRedirects(false);

			// extract location header containing the actual
			// destination URL
			String expandedURL =
			httpURLConnection.getHeaderField("Location");
			httpURLConnection.disconnect();

			return expandedURL;
		}
```

```
	public static String keepYoutubeUrl(String url) {
		String short_yt = "http://youtu.be/";
		String full_yt = "http://www.youtube.com/watch?v=";
		if (url.contains(full_yt))
			return url;
		else if (url.contains(short_yt))
			return url;
		else if (url.contains("www.youtube.com"))
			return url;
		return null;
	}
}

public static class Map extends
		Mapper<LongWritable, Text, Text, NullWritable> {

	private Text tweet = new Text();

	public void map(LongWritable key, Text value, Context context)
			throws IOException, InterruptedException {
		String tw, tinyURL, expandedURL, answer, youtube_url;
		String[] tokens;
		tw = value.toString();
		if (tw == null)
			return;
		answer = "";
		// user_id,tweet_id,isRetweet,isRetweeted,isFavorited,
		// retweet_count,favorite_count,day,month,year,minutes,
		// hours,tinyURL
		tokens = tw.split(",");
		if (tokens.length != 13)
			return;
		tinyURL = tokens[tokens.length - 1];
		expandedURL = UrlExpander.expandUrl(tinyURL);
		if (expandedURL == null)
			return;
		youtube_url = UrlExpander.keepYoutubeUrl(expandedURL);
		if (youtube_url == null)
			return;
		for (int i = 0; i < tokens.length - 1; i++)
			answer += tokens[i] + ",";
		answer += youtube_url;
		tweet.set(answer);
		context.write(tweet, NullWritable.get());
	}
}

public static class Reduce extends
		Reducer<Text, NullWritable, Text, NullWritable> {

	public void reduce(Text key, Iterable<NullWritable> values,
			Context context) throws IOException,
					InterruptedException {
		context.write(key, NullWritable.get());
	}
}
```

```
	public static void main(String[] args) throws Exception {
		Configuration conf = new Configuration();

		Job job = Job.getInstance(conf, "expandtinyURLs");
		job.setJarByClass(ExpandTinyURLs.class);

		job.setOutputKeyClass(Text.class);
		job.setOutputValueClass(NullWritable.class);

		job.setMapperClass(Map.class);
		job.setCombinerClass(Reduce.class);
		job.setReducerClass(Reduce.class);

		job.setNumReduceTasks(0);

		job.setInputFormatClass(TextInputFormat.class);
		job.setOutputFormatClass(TextOutputFormat.class);

		FileInputFormat.addInputPath(job, new Path(args[0]));
		FileOutputFormat.setOutputPath(job, new Path(args[1]));

		System.exit(job.waitForCompletion(true) ? 0 : 1);
	}

}
```

# Παράρτημα Β

Σε αυτό το παράρτημα παρουσιάζονται οι δύο κώδικες σε PHP όπου γίνεται η εξαγωγή των χαρακτηριστικών των βίντεο από το API του YouTube. Απαιτείται όμως η ύπαρξη ενός API key.

Αρχικά ο κώδικας index.php:

```
<!DOCTYPE html>
<html lang="en">
   <head>
      <meta charset="utf-8">
      <meta http-equiv="X-UA-Compatible" content="IE=edge">
      <meta name="viewport" content="width=device-width, initial-scale=1">
      <title>Youtube link analysis</title>
   </head>
   <body>
      <div id="youtube-results"></div>
      <script
src="https://ajax.googleapis.com/ajax/libs/jquery/1.11.3/jquery.min.js"></script>
      <script>
         var file = "tw_yt_valid_urls_11.txt";
         var data = {data:[]};
         function getFile(){
            $.get(file,function(txt){
               var lines = txt.split("\n");
               for (var i = 0, len = lines.length; i < len; i++) {
                  var tokens = lines[i].split(",");
                  var link = tokens[tokens.length-1];
                  var youtubeid=link.substr(link.lastIndexOf("watch?v=")+8);
                  data['data'].push({'userid':tokens[0],'tweetid':tokens[1],
'isRetweet':tokens[2],'isRetweeted':tokens[3],'isFavorited':tokens[4],
```

```
'retweet_count':tokens[5],'favorite_count':tokens[6],'day':tokens[7],'month':tokens[8],'ye
ar':tokens[9],'minutes':tokens[10],'hours':tokens[11],'youtubeid':youtubeid});
                }
                 $.ajax({
                   url:"script.php",
                   method:"post",
                   data: data,
                   dataType: 'json',
                   success:function(response){
                   }
                });
            });
        }
        getFile();
    </script>
  </body>
</html>
```

Μετά ο κώδικας script.php:

```
<?php
        ini_set('max_execution_time', 500000);
  $API_KEY= // “insert an API key” //
  $links=$_POST['data'];
  $youtube =
"https://www.googleapis.com/youtube/v3/videos?fields=items(id,snippet(channelId,pub
lishedAt,categoryId,defaultAudioLanguage),statistics)&part=snippet,statistics&key={$
API_KEY}&id=";
  $response['data']=[];
  foreach ($links as $key => $link_status) {
    $results = json_decode(file_get_contents($youtube.$link_status['youtubeid']), true);
    $pair=[];
    $pair['youtube']=$results;
```

```
        $pair['userid']=$link_status['userid'];
        $pair['tweetid']=$link_status['tweetid'];
                $pair['isRetweet']=$link_status['isRetweet'];
                $pair['isRetweeted']=$link_status['isRetweeted'];
                $pair['isFavorited']=$link_status['isFavorited'];
                $pair['retweet_count']=$link_status['retweet_count'];
                $pair['favorite_count']=$link_status['favorite_count'];
                $pair['day']=$link_status['day'];
                $pair['month']=$link_status['month'];
                $pair['year']=$link_status['year'];
                $pair['minutes']=$link_status['minutes'];
                $pair['hours']=$link_status['hours'];
        array_push($response['data'], $pair);
    }
    $myFile = "tw_yt_metadata.json";
    $fh = fopen($myFile, 'w') or die("can't open file");
    $stringData = json_encode($response);
    fwrite($fh, $stringData);
    fclose($fh);
?>
```

# Παράρτημα Γ

Σε αυτό το παράρτημα παρουσιάζεται ο κώδικας Python όπου γίνεται η μετατροπή της JSON μορφής των δεδομένων σε μορφή CSV.

```
import json

input_file = open('tw_yt_metadata.json', 'r')
output_file = open('temp_metadata.csv', 'w')

data = json.load(input_file)
input_file.close()

output_file.write("%s,%s,%s,%s,%s,%s,%s,%s,%s,%s,%s,%s,%s,%s,%s,%s,%s,%s,%s,%s,%s,%s,%s,%s\n"%("userid","tweetid","isRetweet","isRetweeted","isFavorited","retweet_count","favorite_count","day","month","year","minutes","hours","id","publishedAt","channelId","categoryId","viewCount","likeCount","dislikeCount","favoriteCount","commentCount")) # names of fields

for i in data['data']:
    try:
        TypeError_check = i['youtube']['items'] is not None
    except TypeError:
        continue
    output_file.write("%s,%s,%s,%s,%s,%s,%s,%s,%s,%s,%s,%s,%s"% (i['userid'], i['tweetid'], i['isRetweet'], i['isRetweeted'], i['isFavorited'], i['retweet_count'], i['favorite_count'], i['day'], i['month'], i['year'], i['minutes'], i['hours']))
    if (i['youtube']['items'] != []) and (i['youtube']['items'] is not None):
        if 'id' in i['youtube']['items'][0]:
            output_file.write(",%s,"%(i['youtube']['items'][0]['id']))
        if 'snippet' in i['youtube']['items'][0]:
            if 'publishedAt' in i['youtube']['items'][0]['snippet']:
```

```
output_file.write("%s,"%(i['youtube']['items'][0]['snippet']['publishedAt']))
                if 'channelId' in i['youtube']['items'][0]['snippet']:

output_file.write("%s,"%(i['youtube']['items'][0]['snippet']['channelId']))
                if 'categoryId' in i['youtube']['items'][0]['snippet']:

output_file.write("%s,"%(i['youtube']['items'][0]['snippet']['categoryId']))

            if 'statistics' in i['youtube']['items'][0]:
                if 'viewCount' in i['youtube']['items'][0]['statistics']:

output_file.write("%s,"%(i['youtube']['items'][0]['statistics']['viewCount']))
                else:
                        output_file.write("0,")
                if 'likeCount' in i['youtube']['items'][0]['statistics']:

output_file.write("%s,"%(i['youtube']['items'][0]['statistics']['likeCount']))
                else:
                        output_file.write("0,")
                if 'dislikeCount' in i['youtube']['items'][0]['statistics']:

output_file.write("%s,"%(i['youtube']['items'][0]['statistics']['dislikeCount']))
                else:
                        output_file.write("0,")
                if 'favoriteCount' in i['youtube']['items'][0]['statistics']:

output_file.write("%s,"%(i['youtube']['items'][0]['statistics']['favoriteCount']))
                else:
                        output_file.write("0,")
                if 'commentCount' in i['youtube']['items'][0]['statistics']:

output_file.write("%s"%(i['youtube']['items'][0]['statistics']['commentCount']))
```

```
                    else:
                        output_file.write("0")
        output_file.write("\n")

output_file.close()

input_file_2 = open('temp_metadata.csv', 'r')
output_file_2 = open('tw_yt_features.csv', 'w')

for line in input_file_2:
        tokens = line.split(',')
        if len(tokens) > 12:
            output_file_2.write("%s"%(line))

input_file_2.close()
output_file_2.close()
```

# Παράρτημα Δ

Σε αυτό το παράρτημα παρουσιάζεται ο κώδικας Java όπου γίνεται η χρονολογική ταξινόμηση των δεδομένων χρησιμοποιώντας τον αλγόριθμο Merge Sort.

```
import java.io.BufferedReader;
import java.io.BufferedWriter;
import java.io.File;
import java.io.FileNotFoundException;
import java.io.FileReader;
import java.io.FileWriter;
import java.io.IOException;
import java.util.ArrayList;

public class SortTweets {

	final static String in_name = "tw_yt_features.csv";
	final static String out_name = "tw_yt_features_sorted.csv";
	private ArrayList<Tweet> tweets = new ArrayList<Tweet>();
	private String first_line;

	private Tweet[] array;
	private Tweet[] tempMergArr;
	private int length;

	private static class Tweet {
		long key; // year month day hours minutes
		String value; // the whole tweet

		private Tweet(String tw) {
			value = tw;
			key = createKey(tw);
		}

		private long createKey(String tw) {
			String[] tokens = tw.split(",");
			String temp;
			if (tokens.length < 12)
				return 0;
			else {
				try {
					int day = Integer.parseInt(tokens[7]);
					int month = Integer.parseInt(tokens[8]);
					int year = Integer.parseInt(tokens[9]);
					int minutes = Integer.parseInt(tokens[10]);
					int hours = Integer.parseInt(tokens[11]);
					temp = "" + year;
					if (month < 10)
						temp += "0" + month;
					else
						temp += month;
					if (day < 10)
						temp += "0" + day;
					else
```

```
                                        temp += day;
                                if (hours < 10)
                                        temp += "0" + hours;
                                else
                                        temp += hours;
                                if (minutes < 10)
                                        temp += "0" + minutes;
                                else
                                        temp += minutes;
                                return Long.parseLong(temp);
                        } catch (NumberFormatException nfe) {
                                return 0;
                        }
                }
        }
    }

    private void readData() {
        BufferedReader in_file = null;
        String line;
        try {
                in_file = new BufferedReader(new FileReader(new
File(in_name)));
                line = in_file.readLine();
                if (line != null) {
                        first_line = line;
                        line = in_file.readLine();
                }
                while (line != null) {
                        tweets.add(new Tweet(line));
                        line = in_file.readLine();
                }
        } catch (FileNotFoundException fnfe) {
                fnfe.printStackTrace();
                System.err.println("File Not Found!!!");
        } catch (IOException ioe) {
                ioe.printStackTrace();
                System.err.println("Error reading or writing of file.");
        } finally {
                try {
                        in_file.close();
                } catch (IOException e) {
                        e.printStackTrace();
                        System.err.println("Error at closing files.");
                        System.exit(-1);
                }
        }
    }

    public void writeData() {
        BufferedWriter out_file = null;
        try {
                out_file = new BufferedWriter(new FileWriter(new
File(out_name)));
                out_file.write(first_line + "\n");
                for (int i = 0; i < array.length; i++)
                        out_file.write(array[i].value + "\n");
        } catch (FileNotFoundException fnfe) {
```

```
                fnfe.printStackTrace();
                System.err.println("File Not Found!!!");
        } catch (IOException ioe) {
                ioe.printStackTrace();
                System.err.println("Error reading or writing of file.");
        } finally {
                try {
                        out_file.close();
                } catch (IOException e) {
                        e.printStackTrace();
                        System.err.println("Error at closing files.");
                        System.exit(-1);
                }
        }
}

private void execute() {
        readData();
        sort();
        writeData();
}

private void sort() {
        length = tweets.size();
        array = new Tweet[length];
        for (int i = 0; i < length; i++)
                array[i] = new Tweet(tweets.get(i).value);
        tempMergArr = new Tweet[length];
        for (int i = 0; i < length; i++)
                tempMergArr[i] = new Tweet(tweets.get(i).value);
        doMergeSort(0, length - 1);
}

private void doMergeSort(int lowerIndex, int higherIndex) {
        if (lowerIndex < higherIndex) {
                int middle = lowerIndex + (higherIndex - lowerIndex) / 2;
                // Below step sorts the left side of the array
                doMergeSort(lowerIndex, middle);
                // Below step sorts the right side of the array
                doMergeSort(middle + 1, higherIndex);
                // Now merge both sides
                mergeParts(lowerIndex, middle, higherIndex);
        }
}

private void mergeParts(int lowerIndex, int middle, int higherIndex) {
        for (int i = lowerIndex; i <= higherIndex; i++) {
                tempMergArr[i].key = array[i].key;
                tempMergArr[i].value = array[i].value;
        }
        int i = lowerIndex;
        int j = middle + 1;
        int k = lowerIndex;
        while (i <= middle && j <= higherIndex) {
                if (tempMergArr[i].key <= tempMergArr[j].key) {
                        array[k].key = tempMergArr[i].key;
                        array[k].value = tempMergArr[i].value;
                        i++;
```

```
                } else {
                        array[k].key = tempMergArr[j].key;
                        array[k].value = tempMergArr[j].value;
                        j++;
                }
                k++;
        }
        while (i <= middle) {
                array[k].key = tempMergArr[i].key;
                array[k].value = tempMergArr[i].value;
                k++;
                i++;
        }
    }

    public static void main(String[] args) {
        SortTweets sorter = new SortTweets();
        sorter.execute();
    }

}
```

# Παράρτημα Ε

Σε αυτό το παράρτημα παρουσιάζεται ο κώδικας Python όπου μέσω της βιβλιοθήκης scikit-learn γίνεται η εκπαίδευση του classifier.

```
from sklearn.ensemble import GradientBoostingClassifier
from sklearn.cross_validation import cross_val_score

################################################################
def classify (train_f, train_t, test_f, test_t):
    precisionAll = 0
    recallAll = 0
    positive = 0
    scores = []
    clf = GradientBoostingClassifier(n_estimators=100).fit(train_f, train_t)
    j = 0
    while j < len(test_t):
        score = clf.score([test_f[j]], [test_t[j]])
        if int(round(score)) == 1:
            precisionAll = precisionAll + 1
        if int(test_t[j]) == 1:
            recallAll = recallAll + 1
            if int(round(score)) == 1:
                positive = positive + 1
        scores.append(score)
        j = j + 1

    avg_score = 0
    for s in scores:
        avg_score = avg_score + s
    avg_score = avg_score / len(scores)
    precision = 0.0
    if precisionAll != 0:
```

```
            precision = float(positive) / float(precisionAll)
        recall = 0.0
        if recallAll != 0:
            recall = float(positive) / float(recallAll)
        f1score = 0.0
        if (precision + recall) != 0.0:
            f1score = (precision * recall) / (precision + recall)
        result = [avg_score, precision, recall, f1score]
        return result

#####################################################################
def featureImportance (training_set, targets_set, attributes):
        clf = GradientBoostingClassifier(n_estimators=100).fit(training_set, targets_set)
        importance = clf.feature_importances_

#       followers_count     0
#       followings_count    1
#       isVerifiedCelebrity 2
#       isRetweet           3
#       isRetweeted         4
#       isFavorited         5
#       retweet count       6
#       favorite_count      7
#       -------------------------
#       viewCount           8
#       likeCount           9
#       dislikeCount        10
#       favoriteCount       11
#       commentCount            12
#       channelId           13
#       categoryId          14

        list_importance = []
```

```
        i = 0
        while i < len(importance):
                list_importance.append([attributes[i],importance[i]])
                i = i + 1
        list_importance.sort(key=lambda x: x[1])
        return list_importance
#####################################################################

# read features and create tw / yt / both lists
features_file = open('features.txt','r')
training_set_tw = []
training_set_yt = []
training_set_both = []
for line in features_file:
        tw_features = []
        yt_features = []
        if "\n" in line:
                line = line[:-1]
        tokens = line.split(",")
        if len(tokens) < 2:
                continue
        training_set_both.append(tokens)
        k = 0
        while k <= 7:
                tw_features.append(tokens[k])
                k = k + 1
        training_set_tw.append(tw_features)
        k = 8
        while k <= 14:
                yt_features.append(tokens[k])
                k = k + 1
        training_set_yt.append(yt_features)
features_file.close()
```

```
# read targets and create tw / yt / both lists
targets_file = open('targets.txt', 'r')
targets_set_tw = []
targets_set_yt = []
targets_set_both = []
for line in targets_file:
        if "\n" in line:
                line = line[:-1]
        tokens = line.split(",")
        if len(tokens) != 3:
                continue
        targets_set_tw.append(tokens[0])
        targets_set_yt.append(tokens[1])
        targets_set_both.append(tokens[2])
targets_file.close()

#############################################################

test_limits = []
chunk_size = int(len(targets_set_both) / 10)
k=0
while k < 10:
        test_limits.append([int(k*chunk_size),int((k+1)*chunk_size)])
        k = k + 1

results_tw_tw = []
results_tw_yt = []
results_tw_both = []
results_yt_tw = []
results_yt_yt = []
results_yt_both = []
results_both_tw = []
```

```
results_both_yt = []
results_both_both = []

base_results_tw_tw = []
base_results_tw_yt = []
base_results_tw_both = []
base_results_yt_tw = []
base_results_yt_yt = []
base_results_yt_both = []
base_results_both_tw = []
base_results_both_yt = []
base_results_both_both = []

k=0
while k < 10:
        train_f_tw = []
        test_f_tw = []
        train_t_tw = []
        test_t_tw = []
        train_f_yt = []
        test_f_yt = []
        train_t_yt = []
        test_t_yt = []
        train_f_both = []
        test_f_both = []
        train_t_both = []
        test_t_both = []

        base_train_f_tw = []
        base_test_f_tw = []
        base_train_f_yt = []
        base_test_f_yt = []
        base_train_f_both = []
```

```
        base_test_f_both = []

        j = 0
        while j < len(targets_set_both):
                if (j>=test_limits[k][0]) and (j<test_limits[k][1]):
                        test_f_tw.append(training_set_tw[j])
                        test_t_tw.append(targets_set_tw[j])
                        test_f_yt.append(training_set_yt[j])
                        test_t_yt.append(targets_set_yt[j])
                        test_f_both.append(training_set_both[j])
                        test_t_both.append(targets_set_both[j])
                        base_test_f_tw.append([training_set_tw[j][6]])
                        base_test_f_yt.append([training_set_yt[j][0]])
                        base_test_f_both.append([training_set_both[j][6],
training_set_both[j][8]])
                else:
                        train_f_tw.append(training_set_tw[j])
                        train_t_tw.append(targets_set_tw[j])
                        train_f_yt.append(training_set_yt[j])
                        train_t_yt.append(targets_set_yt[j])
                        train_f_both.append(training_set_both[j])
                        train_t_both.append(targets_set_both[j])
                        base_train_f_tw.append([training_set_tw[j][6]])
                        base_train_f_yt.append([training_set_yt[j][0]])
                        base_train_f_both.append([training_set_both[j][6],
training_set_both[j][8]])
                j = j + 1

        # classifier
        results_tw_tw.append(classify(train_f_tw, train_t_tw, test_f_tw, test_t_tw))
        results_tw_yt.append(classify(train_f_tw, train_t_yt, test_f_tw, test_t_yt))
        results_tw_both.append(classify(train_f_tw,          train_t_both,          test_f_tw,
test_t_both))
```

```
        results_yt_tw.append(classify(train_f_yt, train_t_tw, test_f_yt, test_t_tw))
        results_yt_yt.append(classify(train_f_yt, train_t_yt, test_f_yt, test_t_yt))
        results_yt_both.append(classify(train_f_yt, train_t_both, test_f_yt, test_t_both))

        results_both_tw.append(classify(train_f_both, train_t_tw, test_f_both, test_t_tw))
        results_both_yt.append(classify(train_f_both, train_t_yt, test_f_both, test_t_yt))
        results_both_both.append(classify(train_f_both, train_t_both, test_f_both, test_t_both))

        # base classifier
        base_results_tw_tw.append(classify(base_train_f_tw, train_t_tw, base_test_f_tw, test_t_tw))
        base_results_tw_yt.append(classify(base_train_f_tw, train_t_yt, base_test_f_tw, test_t_yt))
        base_results_tw_both.append(classify(base_train_f_tw, train_t_both, base_test_f_tw, test_t_both))

        base_results_yt_tw.append(classify(base_train_f_yt, train_t_tw, base_test_f_yt, test_t_tw))
        base_results_yt_yt.append(classify(base_train_f_yt, train_t_yt, base_test_f_yt, test_t_yt))
        base_results_yt_both.append(classify(base_train_f_yt, train_t_both, base_test_f_yt, test_t_both))

        base_results_both_tw.append(classify(base_train_f_both, train_t_tw, base_test_f_both, test_t_tw))
        base_results_both_yt.append(classify(base_train_f_both, train_t_yt, base_test_f_both, test_t_yt))
        base_results_both_both.append(classify(base_train_f_both, train_t_both, base_test_f_both, test_t_both))
```

```
        k = k + 1

# create file for classifier results
results_file = open('resultsAUC.txt','w')

results_file.write("new\n")
for r in results_tw_tw:
        temp_str = str(r[0]) + "\t" + str(r[1]) + "\t" + str(r[2]) + "\t" + str(r[3]) + "\n"
        results_file.write(temp_str)
results_file.write("new\n")
for r in results_tw_yt:
        temp_str = str(r[0]) + "\t" + str(r[1]) + "\t" + str(r[2]) + "\t" + str(r[3]) + "\n"
        results_file.write(temp_str)
results_file.write("new\n")
for r in results_tw_both:
        temp_str = str(r[0]) + "\t" + str(r[1]) + "\t" + str(r[2]) + "\t" + str(r[3]) + "\n"
        results_file.write(temp_str)

results_file.write("new\n")
for r in results_yt_tw:
        temp_str = str(r[0]) + "\t" + str(r[1]) + "\t" + str(r[2]) + "\t" + str(r[3]) + "\n"
        results_file.write(temp_str)
results_file.write("new\n")
for r in results_yt_yt:
        temp_str = str(r[0]) + "\t" + str(r[1]) + "\t" + str(r[2]) + "\t" + str(r[3]) + "\n"
        results_file.write(temp_str)
results_file.write("new\n")
for r in results_yt_both:
        temp_str = str(r[0]) + "\t" + str(r[1]) + "\t" + str(r[2]) + "\t" + str(r[3]) + "\n"
        results_file.write(temp_str)

results_file.write("new\n")
for r in results_both_tw:
```

```
        temp_str = str(r[0]) + "\t" + str(r[1]) + "\t" + str(r[2]) + "\t" + str(r[3]) + "\n"
        results_file.write(temp_str)
results_file.write("new\n")
for r in results_both_yt:
        temp_str = str(r[0]) + "\t" + str(r[1]) + "\t" + str(r[2]) + "\t" + str(r[3]) + "\n"
        results_file.write(temp_str)
results_file.write("new\n")
for r in results_both_both:
        temp_str = str(r[0]) + "\t" + str(r[1]) + "\t" + str(r[2]) + "\t" + str(r[3]) + "\n"
        results_file.write(temp_str)

results_file.close()

# create file for base classifier results
base_results_file = open('baseresultsAUC.txt','w')

base_results_file.write("new\n")
for r in base_results_tw_tw:
        temp_str = str(r[0]) + "\t" + str(r[1]) + "\t" + str(r[2]) + "\t" + str(r[3]) + "\n"
        base_results_file.write(temp_str)
base_results_file.write("new\n")
for r in base_results_tw_yt:
        temp_str = str(r[0]) + "\t" + str(r[1]) + "\t" + str(r[2]) + "\t" + str(r[3]) + "\n"
        base_results_file.write(temp_str)
base_results_file.write("new\n")
for r in base_results_tw_both:
        temp_str = str(r[0]) + "\t" + str(r[1]) + "\t" + str(r[2]) + "\t" + str(r[3]) + "\n"
        base_results_file.write(temp_str)

base_results_file.write("new\n")
for r in base_results_yt_tw:
        temp_str = str(r[0]) + "\t" + str(r[1]) + "\t" + str(r[2]) + "\t" + str(r[3]) + "\n"
        base_results_file.write(temp_str)
```

```
base_results_file.write("new\n")
for r in base_results_yt_yt:
        temp_str = str(r[0]) + "\t" + str(r[1]) + "\t" + str(r[2]) + "\t" + str(r[3]) + "\n"
        base_results_file.write(temp_str)
base_results_file.write("new\n")
for r in base_results_yt_both:
        temp_str = str(r[0]) + "\t" + str(r[1]) + "\t" + str(r[2]) + "\t" + str(r[3]) + "\n"
        base_results_file.write(temp_str)

base_results_file.write("new\n")
for r in base_results_both_tw:
        temp_str = str(r[0]) + "\t" + str(r[1]) + "\t" + str(r[2]) + "\t" + str(r[3]) + "\n"
        base_results_file.write(temp_str)
base_results_file.write("new\n")
for r in base_results_both_yt:
        temp_str = str(r[0]) + "\t" + str(r[1]) + "\t" + str(r[2]) + "\t" + str(r[3]) + "\n"
        base_results_file.write(temp_str)
base_results_file.write("new\n")
for r in base_results_both_both:
        temp_str = str(r[0]) + "\t" + str(r[1]) + "\t" + str(r[2]) + "\t" + str(r[3]) + "\n"
        base_results_file.write(temp_str)

base_results_file.close()

# feature importance
attributes_both =
['followers_count','followings_count','isVerifiedCelebrity','isRetweet','isRetweeted','isFa
vorited','retweet_count','favorite_count','viewCount','likeCount','dislikeCount','favoriteC
ount','commentCount','channelId','categoryId']

attributes_tw =
['followers_count','followings_count','isVerifiedCelebrity','isRetweet','isRetweeted','isFa
vorited','retweet_count','favorite_count']
```

```
attributes_yt = ['viewCount','likeCount','dislikeCount','favoriteCount','commentCount','channelId','categoryId']

importance_tw_yt = featureImportance(training_set_tw, targets_set_yt, attributes_tw)
importance_yt_tw = featureImportance(training_set_yt, targets_set_tw, attributes_yt)
importance_both_both = featureImportance(training_set_both, targets_set_both, attributes_both)

feature_importance_file = open('feature_importance.txt','w')

feature_importance_file.write("new\n")
for r in importance_tw_yt:
        temp_str = r[0] + "\t" + str(r[1]) + "\n"
        feature_importance_file.write(temp_str)
feature_importance_file.write("new\n")
for r in importance_yt_tw:
        temp_str = r[0] + "\t" + str(r[1]) + "\n"
        feature_importance_file.write(temp_str)
feature_importance_file.write("new\n")
for r in importance_both_both:
        temp_str = r[0] + "\t" + str(r[1]) + "\n"
        feature_importance_file.write(temp_str)

feature_importance_file.close()
```