ΠΟΛΥΤΕΧΝΕΙΟ ΚΡΗΤΗΣ

ΣΧΟΛΗ ΗΛΕΚΤΡΟΛΟΓΩΝ ΜΗΧΑΝΙΚΩΝ & ΜΗΧΑΝΙΚΩΝ ΥΠΟΛΟΓΙΣΤΩΝ

Πρόβλεψη Μετοχών σε Πραγματικό Χρόνο

ΔΙΠΛΩΜΑΤΙΚΗ ΕΡΓΑΣΙΑ

Αργύρης Μούλιος

Επιτροπή διπλωματικής

Αναπληρωτής Καθηγητής Αντώνιος Δεληγιαννάκης (επιβλέπων)
Καθηγητής Μίνως Γαροφαλάκης
Αναπληρωτής Καθηγητής Γεώργιος Χαλκιαδάκης

ΧΑΝΙΑ, ΑΠΡΙΛΙΟΣ, 2019

Abstract

“Big Data analytics” is the method by which we look at big data to reveal hidden patterns, incomprehensible relationships, and other important data that can be used to solve decision-making problems. In recent years there has been an increasing interest in big data due to their rapid growth and since it covers different application domains. An important area of Big Data application is in the financial system and more specifically in the field of stock prediction. In this case, the data streams can be massive and continuous. Many times in stock forecasting, it is crucial the data to be processed and draw conclusions in real-time in order to end up with useful forecasts which can provide us with in time solutions in decision-making problems. The aim of this diploma thesis is to identify time-delayed correlated pairs among thousands of shares, in a real-time and distributed way, in order to extract information that will be used in their prognosis. More specifically, we are interested in extracting information about the course of shares that affects the course of others, the time lag and the correlation degree in this phenomenon. To achieve this goal, it is critical to ensure the scalability of the techniques we are about to use, in order to maintain a reasonable time for the results outputs, despite the increasing volume of incoming data. Therefore, we implemented the algorithm covering locality sensitive hashing (CLSH) without false negatives, on top of the distributed system Apache Storm, which we will analyze and present as well as the experimental results of this study.

Πρόλογος

Με τον όρο “Big Data analytics” αναφερόμαστε στη μέθοδο κατά την οποία μελετούμε μεγάλα δεδομένα (Big Data) με σκοπό την εύρεση και ανάλυση κρυφών μοτίβων, ακατανόητων σχέσεων και άλλων σημαντικών δεδομένων τα οποία χρησιμοποιούνται για την επίλυση προβλημάτων που απαιτούν λήψεις αποφάσεων. Τα τελευταία χρόνια έχει εκδηλωθεί ένα αυξανόμενο ενδιαφέρον για τα μεγάλα δεδομένα εξαιτίας της ραγδαίας ανάπτυξης τους και λόγω του ότι καλύπτουν διάφορους τομείς εφαρμογών. Ένας σημαντικός τομέας εφαρμογής των Big Data είναι στο χρηματοοικονομικό σύστημα και πιο συγκεκριμένα στην προσπάθεια της πρόγνωσης μετοχών. Στη συγκεκριμένη περίπτωση η ροή των δεδομένων μπορεί να είναι εκτεταμένη και διαρκής. Πολλές φορές κατά την πρόγνωση μετοχών είναι επιτακτική η ανάγκη για επεξεργασία των δεδομένων και εξαγωγή συμπερασμάτων σε πραγματικό χρόνο ώστε οι προβλέψεις να είναι χρήσιμες, παρέχοντας τη δυνατότητα έγκαιρης λήψης αποφάσεων. Στόχος αυτής της διπλωματικής εργασίας είναι ο εντοπισμός χρόνο-καθυστερημένων συσχετίσεων ζευγών μεταξύ χιλιάδων μετοχών, σε πραγματικό χρόνο και κατανεμημένα, ώστε να εξάγουμε πληροφορίες οι οποίες θα χρησιμοποιηθούν στην πρόγνωση αυτών. Πιο συγκεκριμένα ενδιαφερόμαστε για την εξαγωγή πληροφοριών σχετικά με την πορεία μετοχών η οποία επηρεάζει την πορεία άλλων, τη χρονική καθυστέρηση καθώς και το βαθμό συσχέτισης αυτού του φαινομένου. Για την επίτευξη του άνωθεν στόχου, κρίσιμης σημασίας είναι η κλιμακωσιμότητα των τεχνικών που θα χρησιμοποιήσουμε ώστε να είναι εφικτή η διατήρηση μιας λογικής χρονικής απόδοσης ως προς την εξαγωγή αποτελεσμάτων, παρά την αύξηση του όγκου των εισερχόμενων δεδομένων. Για το λόγο αυτό υιοθετήσαμε και εφαρμόσαμε τον αλγόριθμο covering locality sensitive hashing (CLSH) χωρίς false negatives, πάνω στο κατανεμημένο σύστημα Apache Storm, τον οποίο θα αναλύσουμε και θα παρουσιάσουμε καθώς και τα πειραματικά αποτελέσματα που τελικά προέκυψαν σε αυτή τη μελέτη.

Ευχαριστίες

Αρχικά θα ήθελα να ευχαριστήσω τον καθηγητή μου Αντώνιο Δεληγιαννάκη ο οποίος παρά τον εργασιακό του φόρτο αποδέχθηκε να με αναλάβει και με βοήθησε στη διεκπεραίωση της εργασίας μέχρι τέλους.

Έπειτα μείζονος σημασίας ήταν η υποστήριξη, ηθική και οικονομική, της οικογένειάς μου. Ευχαριστώ εκ βάθους καρδίας τους γονείς μου Γεώργιο και Χρυσάνθη καθώς και την αδελφή μου Μαρίνα η οποία στάθηκε πάντα δίπλα μου ως ηθικός αρωγός και μέντορας.

Τέλος σημαντικά πρόσωπα στη φοιτητική μου ζωή ήταν οι καλύτεροι μου φίλοι και συμφοιτητές Γεώργιος Γληγόρης και Αλέξανδρος Λουκόπουλος, οι οποίοι εκτός από τις αξέχαστες στιγμές, μου προσέφεραν ηθική και ψυχολογική υποστήριξη σε όλους τους τομείς της νέας μου τότε ζωής.

Πίνακας Περιεχομένων

Ευρετήριο Εικόνων

1 Εισαγωγή

1.1 Ο Χρηματιστηριακός Τομέας

Τα μεγάλα δεδομένα και η ανάλυσή τους είναι ένας ταχέως αναπτυσσόμενος και ζωτικός τομέας σε διαφόρους ερευνητικούς-επιστημονικούς κλάδους και μη. Συχνά είναι επιτακτική η ανάγκη της επεξεργασίας και ανάλυσης μεγάλου όγκου δεδομένων σε πραγματικό χρόνο (real time) καθώς τα χρονικά όρια για τη λήψη σημαντικών αποφάσεων είναι περιορισμένα (π.χ. ανίχνευση και μπλοκάρισμα κλεμμένης πιστωτικής κάρτας κατά τη χρήση της).

Ένα σημαντικό πεδίο είναι το χρηματοοικονομικό και πιο συγκεκριμένα οι συναλλαγές μετοχών στα χρηματιστήρια. Τα χρηματιστήρια τα δημιούργησε η επιδίωξη για την εξεύρεση βραχυπρόθεσμων αλλά κυρίως μακροπρόθεσμων κεφαλαίων και η ανάγκη για σύναψη αγοροπωλησιών μεγάλων ποσοτήτων εμπορευμάτων που βρίσκονται μακρυά από τον τόπο διαπραγμάτευσης τους, ενώ απαιτούνταν για αυτά σοβαρά κεφάλαια αλλά και η τάση για κερδοφορία. Η οργανωμένη μορφή τους οφείλεται στην ταχύτητα διενέργειας των συναλλαγών, στην αμεσότητά τους, στη δημοσιότητα αυτών όπου φαίνονται δημόσια όλα τα χαρακτηριστικά τους (προσφορά, ζήτηση, ποσότητα και αξία) καθώς και στην καθαρότητα τους. Τα χρηματιστήρια διευκολύνουν τις συναλλαγές, γιατί επιτρέπουν στους εκπροσώπους της προσφοράς και της ζήτησης να βρίσκονται ταυτόχρονα στον συγκεκριμένο τόπο διαπραγμάτευσης, επιτρέπουν την ελεύθερη διαμόρφωση τιμών των αγαθών με βάση τον θεμελιώδη νόμο της προσφοράς και της ζήτησης περιορίζοντας έτσι τον κίνδυνο της δημιουργίας τεχνητών τιμών και τέλος δίνουν την ευκαιρία στις επιχειρήσεις να εξεύρουν κεφάλαια αλλά και στους επενδυτές να μπορούν να διαθέσουν τα χρήματα που έχουν στην επένδυση τους σε τίτλους, με την προσδοκία του κέρδους, συμβάλλοντας έτσι στην τόνωση της παραγωγικότητας και γενικότερα στην ανάπτυξη της χώρας που λειτουργεί το χρηματιστήριο.

Με βάση τα άνωθεν είναι προφανές το φάσμα χρηματιστηριακού ενδιαφέροντος όπως και η ανάγκη της ανάλυσης του μεγάλου όγκου δεδομένων που παράγει ένα χρηματιστήριο , έγκυρα και έγκαιρα καθώς και σε ευρεία κλίμακα. Αυτή η ανάλυση

μπορεί να βοηθήσει οδηγώντας σε ορθές χρηματικές επενδύσεις επί των μετοχών προς αγοραπωλησία.

1.2 Κύρια Ιδέα

Σε αυτή τη μελέτη θα εστιάσουμε στην πρόβλεψη των ροπών (άνοδος-κάθοδος της τιμής) των μετοχών (streaming time series). Η ιδέα βασίζεται στο χρηματιστηριακό φαινόμενο της συσχέτισης (correlation) των μετοχών. Πιο συγκεκριμένα οι ροπές μιας μετοχής μπορεί να επηρεάζουν αντίστοιχα τις ροπές μιας ή και περισσότερων άλλων μετοχών. Για παράδειγμα η χρηματιστηριακή πορεία της Google μπορεί να επηρεάσει αναλόγως μια πληθώρα άλλων εταιρειών, αντιστοίχου τομέα ή μη. Οι χρηματοοικονομικές αυτές αλληλεπιδράσεις-συσχετίσεις δεν συμβαίνουν ταυτόχρονα, απαιτούν κάποιο χρόνο διάδοσης και λέμε σε αυτή την περίπτωση πως οι προαναφερθείσες μετοχές ακολουθούν αυτή της Google. Τέλος αυτές οι συσχετίσεις μπορεί να είναι ευθείς (μια μετοχή ακολουθεί τις ροπές μιας άλλης) ή ανάστροφες (μια μετοχή παρουσιάζει αντίστροφες ροπές από μια άλλη). Σε αυτή τη μελέτη θα εστιάσουμε στις ευθείες συσχετίσεις.

Εκμεταλλευόμενοι το παραπάνω φαινόμενο θα προσπαθήσουμε να εντοπίσουμε συσχετισμένα ζεύγη, μεταξύ χιλιάδων μετοχών, τα οποία παρουσιάζουν χρονική καθυστέρηση (lag) στη συσχέτιση τους, βάση της οποίας θα κάνουμε τις προγνώσεις. Η υλοποίηση του όλου εγχειρήματος βασίζεται σε δυο βασικούς πυλώνες. Ο πρώτος είναι το framework Apache Storm το οποίο είναι ένα εργαλείο που μας προσφέρει τη δυνατότητα επεξεργασίας μεγάλου όγκου δεδομένων, κατανεμημένα και σε πραγματικό χρόνο. Το δεύτερο βασικό στοιχείο της μελέτης μας είναι η προσέγγιση του προβλήματος για την εύρεση συσχετισμένων ζευγών μετοχών, με τη χρήση του covering locality sensitive hashing (CLSH) αλγορίθμου εφαρμόζοντάς τον στο εργαλείο Apache Storm. Τέλος, εξετάσαμε τη συμπεριφορά του αλγορίθμου, πάνω σε συνθετικό data set, μεταβάλλοντας διάφορες παραμέτρους του καθώς και τις δυνατότητες κλιμάκωσής του.

1.3 Δομή

Στο κεφάλαιο 2 θα αναλύσουμε τις δυσκολίες που κληθήκαμε να αντιμετωπίσουμε κατά την επίλυση του προβλήματος. Στο κεφάλαιο 3 αναφέρουμε το υπόβαθρο μελέτης και γνώσεων που απαιτήθηκαν. Στο κεφάλαιο 4 κάνουμε ανάλυση του εργαλείου Apache Storm. Στο κεφάλαιο 5 κάνουμε εκτενή ανάλυση της εφαρμογής του αλγορίθμου CLSH στην πρόγνωση μετοχών, όπως την προσεγγίσαμε σε αυτή τη διπλωματική εργασία. Στο κεφάλαιο 6 παρουσιάζουμε και αναλύουμε τα αποτελέσματα που εξήχθησαν. Στο κεφάλαιο 7 κάνουμε μια αναφορά σε σχετικές μελέτες που έχουν γίνει ενώ στο κεφάλαιο 8 παραθέτουμε τα συμπεράσματα μας επί της μελέτης μας.

2 Δυσκολίες Εγχειρήματος

2.1 Προβλήματα προς Επίλυση

Το μοντέλο που θα προταθεί θα πρέπει να φέρει συγκεκριμένα χαρακτηριστικά που θα ικανοποιούν τη φύση των δεδομένων και του προβλήματος. Πιο συγκεκριμένα:

- **Streaming data processing:** Τα δεδομένα ρέουν προς το μοντέλο μας, το οποίο θα πρέπει να τα επεξεργάζεται κατά τη λήψη τους και να εξάγει αποτελέσματα σε ένα πέρασμα των δεδομένων.
- **Real time or near-real time processing:** Η επεξεργασία των δεδομένων σε πραγματικό ή σε σχεδόν πραγματικό χρόνο είναι ήσσονος σημασίας καθώς η λήψη της απόφασης για την πώληση ή αγορά μιας μετοχής πρέπει να γίνει εντός πεπερασμένων χρονικών ορίων και συγκεκριμένα πριν την επόμενη ανανέωση τιμών των μετοχών, αλλιώς δεν θα είχε νόημα αξιοποίησης η πρόγνωση και το όλο εγχείρημα.
- **Distributed computation model:** Ο αποδοτικός διαμοιρασμός του υπολογιστικού φόρτου καθώς και όγκου των δεδομένων, μέσω της αλγοριθμικής παραλληλοποίησης, είναι επιτακτικός. Το μοντέλο μας έτσι θα μπορεί να επιλύει υποσύνολα του αρχικού προβλήματος σε διαφορετικά μηχανήματα και σε παράλληλο χρόνο, ο συνδυασμός των οποίων θα οδηγήσει στο τελικό αποτέλεσμα. Με αυτόν τον τρόπο το σύστημά μας θα είναι scalable, δηλαδή θα μπορεί να ανταπεξέλθει σε ενδεχόμενη αύξηση του όγκου των εισαχθέντων δεδομένων.
- **Quadratic algorithmic cost:** Στόχος είναι η εύρεση συσχετισμένων ζευγών. Κατά την αφελή (naive) αλγοριθμική προσέγγιση επίλυσης θα πρέπει να συγκριθούν όλα τα ζεύγη μετοχών (όλες με όλες). Για παράδειγμα αν έχουμε 5.000 μετοχές θα πρέπει να υλοποιηθούν 24.995.000 [1] συγκρίσεις. Αυτό προσδίδει τετραγωνικό αλγοριθμικό κόστος το οποίο κάνει το σύστημά μας να

1 Δεν μας ενδιαφέρουν οι αυτοσυσχετίσεις. Έτσι τα ζεύγη θα είναι 5.000*4.999 αφού για παράδειγμα πρέπει να ελεγχθεί αν η μετοχή 1 μπορεί να προβλέψει την μετοχή 2 σε κάποιο lag της και αντιστρόφως.

μη κλιμακώνεται, αφού για να ανταπεξέλθει σε μια σχετικά μικρή αύξηση του όγκου δεδομένων χρειάζεται πολύ μεγάλη αύξηση στους πόρους (υπολογιστικής ισχύος).

3 Υπόβαθρο

Σε αυτό το κεφάλαιο θα παραθέσουμε το γνωσιακό υπόβαθρο που χρειάστηκε για την διεκπεραίωση της επίλυσης του προβλήματος της πρόγνωσης μετοχών σε πραγματικό χρόνο.

3.1 Μετρικές Απόστασης

Έστω ότι έχουμε ένα σύνολο στοιχείων. Μια μετρική απόστασης είναι μια συνάρτηση d(x, y) η οποία δέχεται ως παραμέτρους δύο στοιχεία του συνόλου αυτού, εξάγει έναν πραγματικό αριθμό και ικανοποιεί τα παρακάτω κριτήρια:

1. $d(x, y) \geq 0$ (θετικές αποστάσεις)
2. $d(x, y) = 0$, αν και μόνο αν $x=y$ (τα στοιχεία συμπίπτουν)
3. $d(x, y) = d(y, x)$ (συμμετρία απόστασης)
4. $d(x, y) \leq d(x, z) + d(z, y)$ (τριγωνική ανισότητα)

Υπάρχουν πολλές μετρικές απόστασης. Μερικές από αυτές είναι η ευκλείδεια, η Jaccard, η Cosine similarity, η Hamming απόσταση κ.α.. Σε αυτή την ανάλυση θα χρησιμοποιήσουμε την απόσταση Hamming.

3.1.1 Απόσταση Hamming

Έστω ένα συνόλου διανυσμάτων. Ορίζουμε ως απόσταση Hamming μεταξύ δυο εξ αυτών των διανυσμάτων ως το πλήθος των στοιχείων στα οποία διαφέρουν. Όπως είναι λογικό η απόσταση Hamming δεν μπορεί να είναι αρνητική. Αν είναι μηδέν τότε τα δύο αυτά διανύσματα συμπίπτουν. Επίσης το αποτέλεσμα δεν επηρεάζεται

από το ποιο διάνυσμα εκ των δύο θεωρείται πρώτο κατά τη μέτρηση της απόστασης. Τέλος η τριγωνική ανισότητα είναι εμφανής. Πιο συγκεκριμένα, αν τα διανύσματα x και z διαφέρουν κατά m στοιχεία και τα διανύσματα z και y κατά n, τότε τα x και y δεν μπορούν να διαφέρουν σε περισσότερα από m+n στοιχεία. Όπως βλέπουμε τα αξιώματα της Ενότητας 3.1 ικανοποιούνται. Συνήθως η απόσταση Hamming εφαρμόζεται σε boolean διανύσματα, δηλαδή που αποτελούνται από 0 και 1 (bit vectors), ωστόσο μπορεί να εφαρμοστεί σε διανύσματα που διαθέτουν στοιχεία από οποιοδήποτε σύνολο.

Παράδειγμα: Η απόσταση Hamming μεταξύ των δυαδικών διανυσμάτων 11101 και 11110 είναι 2, καθώς διαφέρουν στο τέταρτο και πέμπτο στοιχείο ενώ συμφωνούν στο πρώτο, δεύτερο και τρίτο.

3.1.1.1 Η Λογική Πίσω Από την Επιλογή της Μετρικής Hamming

Σκοπός μας είναι η πρόβλεψη των ροπών των μετοχών με απώτερο στόχο τη λήψη σωστής απόφασης για την αγοραπωλησία αυτών. Τα πιθανά σενάρια είναι η πτώση της τιμής της μετοχής, η σταθερότητα και η άνοδος. Κατά την επικείμενη πτώση μας ενδιαφέρει η πώλησή της μετοχής, ενώ κατά την πιθανή άνοδο ή σταθερότητα ενδιαφερόμαστε για την παραμονή της στην κατοχή μας. Όσο αφορά την αγορά, είναι προφανές ότι στοχεύουμε σε μετοχές οι τιμές των οποίων πρόκειται να ανέβουν ή να διατηρηθούν σταθερές, με το τελευταίο σενάριο να μη μας επηρεάζει αρνητικά λόγω της ουδετερότητάς του. Ο παραπάνω συλλογισμός κάνει εξαίρετη την επιλογή της εν λόγω μετρικής καθώς ενδιαφερόμαστε για δυο διακριτά γεγονότα (πτώση – άνοδος/σταθερότητα), τα οποία θα αναπαραστήσουμε με δυαδικές τιμές.

3.2 Locality Sensitive Hashing (LSH)

Πολλές φορές καλούμαστε να λύσουμε προβλήματα εύρεσης ζευγών σε ένα σύνολο τα οποία παρουσιάζουν ομοιότητα πάνω από ένα προκαθορισμένο κατώφλι. Σε αυτές τις περιπτώσεις ο στόχος μας είναι να εξετάσουμε μόνο εκείνα τα ζεύγη τα οποία

είναι πιο πιθανό να έχουν ομοιότητα. Η γενική θεωρία που μας εξοπλίζει με την παραπάνω δυνατότητα ονομάζεται locality sensitive hashing (LSH).

Η γενική ιδέα του LSH είναι η αντιστοίχηση (“hash”) αντικείμενων αρκετές φορές με τέτοιο τρόπο ώστε αυτά που παρουσιάζουν ομοιότητα να είναι πιο πιθανό να αντιστοιχιστούν στον ίδιο “κουβά” (bucket) από ότι τα ανόμοια. Τα ζεύγη που θα καταλήξουν στον ίδιο “κουβά”, από οποιαδήποτε αντιστοίχηση, είναι υποψήφια και ελέγχουμε μόνο αυτά. Κύριος στόχος μας είναι τα ανόμοια ζεύγη να μην καταλήξουν ως υποψήφια και ως εκ τούτου να μην ελεγχθούν. Κάποια απ αυτά ωστόσο, ονόματι false positives, θα πάνε ενάντια στον άνωθεν στόχο με αποτέλεσμα να σταλούν προς έλεγχο επικύρωσης. Παρόλα αυτά επιδιώκουμε να αποτελούν μικρό ποσοστό εκ του συνόλου των ανόμοιων αντικειμένων. Επιπροσθέτως για τα όμοια ζεύγη επιθυμούμε να αντιστοιχιστούν στον ίδιο “κουβά” τουλάχιστον μια φορά. Αυτά που θα παρεκκλίνουν ονομάζονται false negatives και κύρια επιδίωξή μας για αυτά είναι επίσης να αποτελούν μια μικρή μερίδα εκ του συνόλου των πράγματι ομοίων ζευγών.

3.2.1 Η Θεωρία των LSH Συναρτήσεων

Σε αυτή την ενότητα θα αναλύσουμε τη θεωρία των οικογενειών των συναρτήσεων οι οποίες μπορούν να μας προσφέρουν αποδοτικά τα υποψήφια προς ομοιότητα ζεύγη. Αυτές οι συναρτήσεις μπορούν να εφαρμοστούν σε διάφορα είδη συνόλων δεδομένων και μετρικών απόστασης. Υπάρχουν τρεις συνθήκες τις οποίες πρέπει να ικανοποιούν:

1. Οφείλουν να εξασφαλίζουν στα κοντινά-συσχετισμένα ζεύγη υψηλότερες πιθανότητες να είναι υποψήφια συγκριτικά με τα μακρινά. (Αναλύεται επακριβώς στο 3.3.2)
2. Πρέπει να είναι στατιστικά ανεξάρτητες. Δηλαδή να είναι εφικτή η εκτίμηση της πιθανότητας του γεγονότος, κατά το οποίο δύο ή περισσότερες συναρτήσεις μας δίνουν όλες σίγουρη απόκριση στην εύρεση όλων των όμοιων ζευγών, μέσω του κανόνα γινομένου ανεξάρτητων γεγονότων.
3. Να είναι αποδοτικές μέσω των παρακάτω δύο κριτηρίων:

A) Πρέπει να είναι ικανές για την εύρεση των υποψηφίων ζευγών σε πολύ λιγότερο χρόνο από όσο χρειάζεται ο έλεγχος όλων των ζευγών.

B) Πρέπει να μπορούν να συνδυαστούν με σκοπό τη δημιουργία νέων συναρτήσεων οι οποίες θα είναι πιο αποδοτικές στην αποφυγή των false positives & false negatives. Οι νέες αυτές συναρτήσεις επίσης πρέπει να ικανοποιούν την μόλις προαναφερθείσα συνθήκη (A).

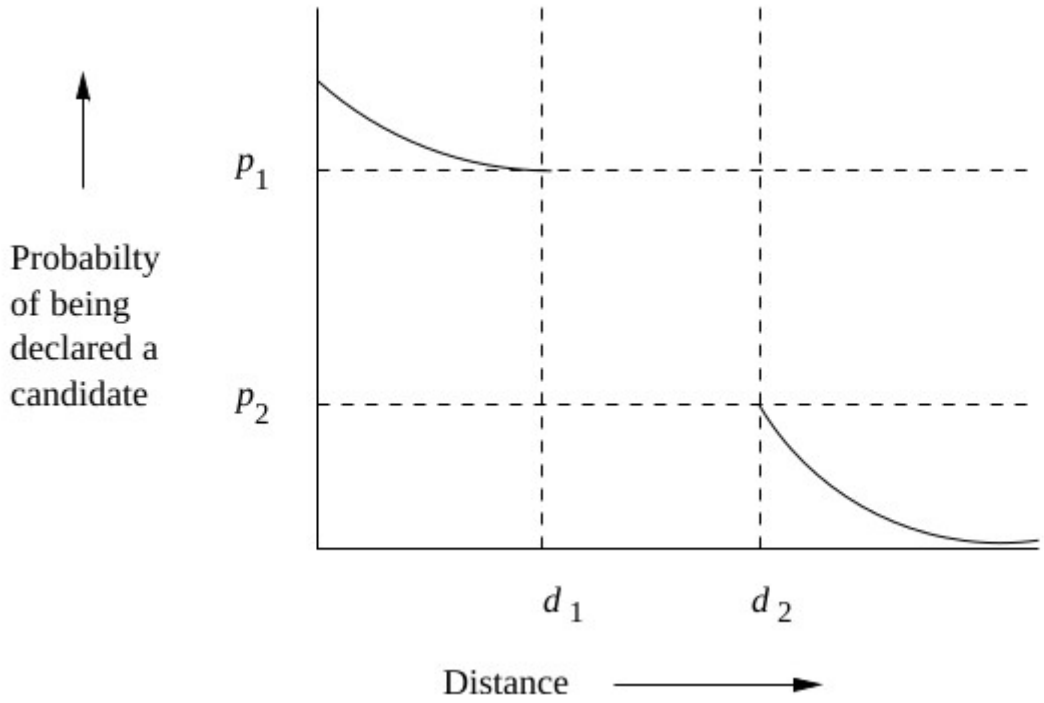
3.2.2 Οι LSH Συναρτήσεις

Έστω συναρτήσεις οι οποίες δέχονται ως είσοδο δύο στοιχεία και εξάγουν απόφαση σχετικά με το αν αυτά πρέπει να είναι υποψήφια ζεύγη. Συνήθως η συνάρτηση f αντιστοιχίζει αντικείμενα και το κριτήριο της υποψηφιότητας έγκειται στο αν το αποτέλεσμα είναι ισότητα. Χάριν απλότητας θα χρησιμοποιήσουμε το συμβολισμό f (x) = f (y) για να αποδώσουμε την δημιουργία υποψηφίου ζεύγους ενώ σε αντίθετη περίπτωση τα εν λόγω ζεύγη δεν θα είναι υποψήφια μέχρι κάποια άλλη συνάρτηση να μας το υποδείξει. Ένα σύνολο από συναρτήσεις αυτής της μορφής θα ονομάζεται οικογένεια συναρτήσεων.

Έστω d1 < d2, δυο αποστάσεις σύμφωνες με κάποια μετρική. Μια οικογένεια F συναρτήσεων ονομάζεται (d1, d2, p1, p2)-sensitive αν για κάθε f στην F ισχύουν:

1. Αν $d(x, y) \leq d1$, τότε η πιθανότητα f (x) = f (y) είναι τουλάχιστον p1.
2. Αν $d(x, y) \geq d2$, τότε η πιθανότητα f (x) = f (y) είναι το πολύ p2.

Η Εικόνα 3.1 παρουσιάζει τι περιμένουμε πιθανοτικά, σχετικά με την κατάταξη των ζευγών ως υποψήφια, από μια συνάρτηση της οικογένειας (d1, d2, p1, p2)-sensitive. Η περιοχή της γραφικής στην οποία η απόσταση των αντικειμένων κυμαίνεται μεταξύ d1 και d2, μπορεί να γίνει όσο μικρή επιθυμούμε. Εν γένει το αντίτιμο σε αυτή την περίπτωση είναι να πλησιάσουν και οι πιθανότητες p1-p2. Αποδεικνύεται [1] πως είναι εφικτό να απομακρύνουμε τις πιθανότητες p1-p2 ενώ κρατάμε σταθερές τις αποστάσεις d1-d2.



Εικόνα 3.1: *Συμπεριφορά μιας (d1, d2, p1, p2)-sensitive συνάρτησης*

(Εικόνα από [1])

Οικογένειες LSH συναρτήσεων μπορούν να δημιουργηθούν για διάφορες μετρικές, μεταξύ των οποίων είναι η Jaccard, η Hamming, η Cosine και η ευκλείδεια απόσταση. Εμείς θα εστιάσουμε στην απόσταση Hamming.

3.2.3 Οικογένειες Συναρτήσεων για την Απόσταση Hamming

Σε αυτή την ενότητα θα παραθέσουμε τη γενική ιδέα της δημιουργίας locality-sensitive οικογένειας συναρτήσεων για την απόσταση Hamming. Έστω ότι έχουμε ένα σύνολο διανυσμάτων d-διαστάσεων και με h(x, y) συμβολίζουμε την Hamming απόστασή τους. Αν πάρουμε οποιαδήποτε στοιχείο από αυτά τα διανύσματα (π.χ. το i-οστό) τότε μπορούμε να ορίσουμε τη συνάρτηση $f_i(x)$ να είναι το εν λόγω bit του διανύσματος. Έτσι $f_i(x) = f_i(y)$ αν και μόνον αν τα διανύσματα x και y συμφωνούν στο i-οστό στοιχείο. Η πιθανότητα να συμβεί το παραπάνω για ένα τυχαία επιλεγμένο στοιχείο στη θέση i, είναι $1 − h(x, y)/d$.

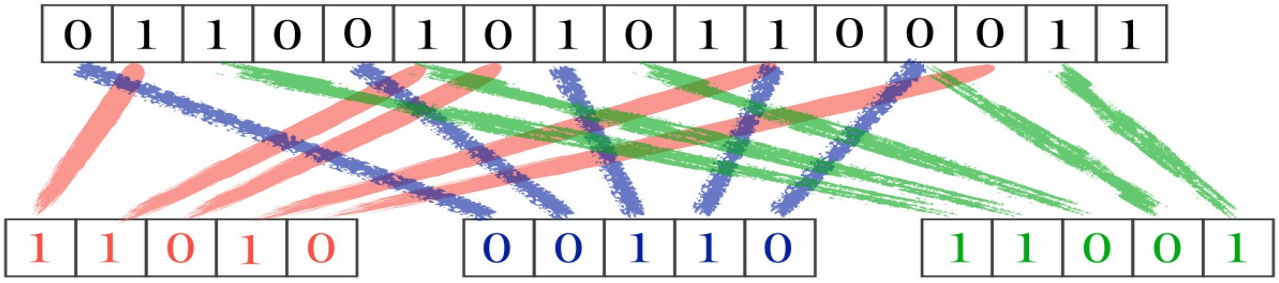
Η οικογένεια F αποτελείται από συναρτήσεις {f1, f2,..., fd} και είναι (d1, d2, 1 − d1 /d, 1 − d2 /d)-sensitive οικογένεια συναρτήσεων κατακερματισμού, οι οποίες ισχύουν για κάθε d1 < d2.

3.2.4 Μια Εκδοχή του LSH για τη Μετρική Hamming

Σε αυτή την ενότητα θα εξετάσουμε μια εφαρμογή του LSH για τη μετρική Hamming [2]. Πιο συγκεκριμένα, θα περιοριστούμε σε δυαδικά διανύσματα που ανήκουν στο σύνολο $\{0,1\}^d$. Κάθε συνάρτηση κατακερματισμού θα δειγματοληπτεί ένα τυχαίο bit από κάθε διάνυσμα. Το σύνολο των συναρτήσεων αυτών θα αποτελούν την οικογένεια συναρτήσεων κατακερματισμού. Για να αυξήσουμε το διάστημα (να απομακρύνουμε) τις πιθανότητες p1 και p2 (βλέπε Εικόνα 3.1) :

- δημιουργούμε πολλές τέτοιες συναρτήσεις το οποίο θα οδηγήσει στα εξής γεγονότα:
  - Η πιθανότητα σύγκρουσης (σύμπτωσης) ομοίων αντικείμενων μειώνεται.
  - Η πιθανότητα σύγκρουσης ανόμοιων αντικείμενων μειώνεται περισσότερο.
- Επαναλαμβάνουμε πολλές φορές το προηγούμενο βήμα
  - Η πιθανότητα σύγκρουσης ομοίων αντικειμένων αυξάνεται.

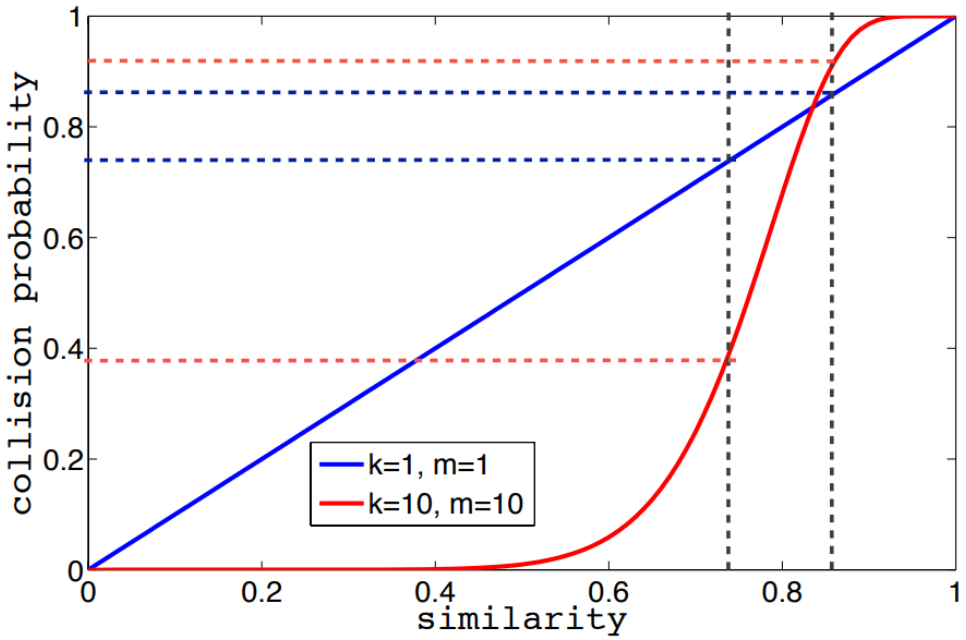
Στην Εικόνα 3.2 βλέπουμε το random bit sampling επί ενός τυχαίου δυαδικού διανύσματος. Συμβολίζουμε με k το πλήθος των τυχαίων bit που δειγματοληπτούμε και με m τον αριθμό των επαναλήψεων της δειγματοληψίας. Έτσι εδώ έχουμε k=5 και m=3.



Εικόνα 3.2: *bit sampling*

(Εικόνα από [2])

Στην Εικόνα 3.3 βλέπουμε τη γραφική παράσταση της πιθανότητας σύγκρουσης ζευγών συναρτήσει του βαθμού ομοιότητας τους για δύο περιπτώσεις. Σε κάθε μια από αυτές αντιστοιχούν διαφορετικές παράμετροι πλήθους τυχαίων bit προς δειγματοληψία και αριθμού επαναλήψεων αυτής.



Εικόνα 3.3: *probability of collision*

$$Pr[h(x) = h(y)] = 1 - (1 - s^k)^m$$

(Εικόνα από [2])

```
preprocessing
input: set of vectors X
     for i=1...m times
          for each x in X
               form xi by sampling k random bits of x
               store x in bucket given by f(xi)

query
input: query vector q
     Z = ∅
     for i=1...m times
          form qi by sampling k random bits of q
          Zi = { points found in the bucket f(qi) }
          Z = Z ∪ Zi
     output all z in Z such that sH(q,z) ≥ s
```

***Εικόνα 3.4**: Εύρεση ομοιότητας μεταξύ διανυσμάτων και κάποιου "query point" (ψευδοκώδικας)*

(Εικόνα από [2])

Στην Εικόνα 3.4 παρουσιάζεται ο ψευδοκώδικας για την εύρεση ομοιότητας μεταξύ διανυσμάτων και κάποιου "query point". Τα βασικά βήματα είναι:

- κάνουμε hash όλα τα διανύσματα.
- κάνουμε hash το query point.
- ελέγχουμε τα υποψήφια σημεία που επιστρέφονται.

```
input: set of vectors X
    P = ∅
    for i=1...m times
        for each x in X
            form x_i by sampling k random bits of x
            store x in bucket given by f(x_i)
        Pi = { pairs of points colliding in a bucket }
        P = P U P_i
    output all pairs p=(x,y) in P such that s_H(x,y) ≥ s
```

Εικόνα 3.5: *Εύρεση ομοιότητας μεταξύ όλων των διανυσμάτων (ψευδοκώδικας)*

(Εικόνα από [2])

Στην Εικόνα 3.5 βλέπουμε τον ψευδοκώδικα για την εύρεση ομοιότητας μεταξύ όλων των διανυσμάτων. Τα βασικά βήματα είναι:

- κάνουμε hash όλα τα διανύσματα.
- ελέγχουμε όλα τα διανύσματα που συγκρούονται σε κάθε κουβά.

3.3 Ο Αλγόριθμος Covering LSH Χωρίς False Negatives (CLSH)

Η βιβλιογραφία που αφορά μεθόδους φιλτραρίσματος με απουσία false negatives για τη μετρική Hamming, είναι σχετικά μικρή. Εδώ θα αναλύσουμε τη μελέτη του Rasmus Pagh και την προσέγγιση επίλυσης του άνωθεν προβλήματος [3].

3.3.1 Θεωρήματα, Αποδείξεις & Ορισμοί

Χρησιμοποιούμε Hamming projection family της μορφής:

$$\mathcal{H}_{\mathcal{A}} = \{x \mapsto x \wedge a \mid a \in \mathcal{A}\}$$

όπου $A \subseteq \{0, 1\}^d$. Τα διανύσματα στον A θα ονομάζονται bitMasks.

Θεωρούμε ότι η απόσταση r<=d/2 (όπου d διαστάσεις διανυσμάτων προς εύρεση συσχέτισης και r η απόσταση Hamming αυτών).

Θεώρημα 3.1 (από [3]):Για κάθε $A \subseteq \{0, 1\}^d$, για κάθε $h \in H_A$ και για κάθε $x,y \in \{0, 1\}^d$ έχουμε $h(x) = h(y)$ αν και μόνο αν $h(x \oplus y) = 0$.

Για να είμαστε σίγουροι ότι για όλα τα ζεύγη εντός απόστασης r θα έχουμε κάποια σύγκρουση (collision) από κάποια συνάρτηση κατακερματισμού (hash function) θέλουμε διανύσματα που ανήκουν στην hamming projection family τα οποία θα μηδενίζουν όλους τους δυνατούς r συνδυασμούς από άσσους στις d διαστάσεις (r-covering).

Ορισμός 3.2 (από [3]): Για $A \subseteq \{0, 1\}^d$, η Hamming projection family H_A είναι r-covering αν για κάθε $x \in \{0, 1\}^d$ με $||x|| \leq r$ [2], υπάρχει $h \in H_A$ ώστε $h(x) = 0$. Η οικογένεια αυτή λέγεται ότι έχει βάρος ω αν $||a|| \geq \omega d$ για κάθε $a \in A$.

Ενδιαφερόμαστε για r-covering οικογένειες που έχουν μη μηδενικές bitMasks ώστε να έχουμε σπάνια collisions μεταξύ των διανυσμάτων που δεν είναι κοντινά. Οι bitMasks στις εν λόγω οικογένειες θα έχουν βάρος κοντά στο 1/2 και θα αντιστοιχίζονται στο $\{0,1\}^{r+1}$. Η οικογένεια θα βασίζεται σε μια συνάρτηση m: {1,2....,d} -> $\{0,1\}^{r+1}$ η οποία θα αντιστοιχίζει θέσεις bit σε δυαδικά διανύσματα (bit vectors) διαστάσεων r+1. Ορίζουμε τέτοια οικογένεια από bitMasks a(v) με $a(v) \in \{0, 1\}^d$:

$$a(v)_i = \langle m(i), v \rangle \mod 2$$

Όπου <m(i),v> εσωτερικό γινόμενο του m(i) και του v (προγραμματιστικά bitcount(m(i) & v)). Θεωρούμε την άνωθεν οικογένεια με μη μηδενικό v ως:

2 ||x|| : το πλήθος από 1-bits στο διάνυσμα x.

$$A(m) = \{a(v) \mid v \in \{0, 1\}^{r+1} \setminus \{0\}\}$$

Θεώρημα 3.3 (από [3]): Για κάθε $m : \{1, \ldots, d\} \rightarrow \{0, 1\}^{r+1}$, η Hamming projection family $H_{A(m)}$ είναι r-covering.

Θεώρημα 3.5 (από [3]): Για όλα τα $x, y \in \{0, 1\}^d$ και για τυχαία $m:\{1, \ldots, d\} \rightarrow \{0, 1\}^{r+1}$ ισχύουν:

(1) Αν $||x - y|| \leq r$ τότε $Pr[\exists h \in H_{A(m)} : h(x) = h(y)] = 1$.

(2) Η αναμενόμενη τιμή πλήθους collisions για κάποιο ζεύγος είναι :
$E[|\{h \in H_{A(m)} \mid h(x) = h(y)\}|] < 2^{r+1-||x-y||}$ (π.χ. αν η απόσταση του x και y είναι μηδέν τότε για κάθε hash function θα έχουμε collision).

Nearest neighbor search: Στο [3] αποδεικνύεται ότι ο εγγύτερος γείτονας έχει απόσταση τουλάχιστον $\lfloor \log v \rfloor$ (v αναπαριστάται ως ακέραιος). Αυτό σημαίνει πως όταν ένα σημείο x με την ελάχιστη απόσταση, μεταξύ των υπολοίπων υποψηφίων για αυτό το v, βρεθεί σε απόσταση το πολύ $\lfloor \log(v + 1) \rfloor$ το επιστρέφουμε ως εγγύτερο γείτονα και σταματάμε την αναζήτηση. Αυτό θα εφαρμοστεί στην περίπτωση που θέλουμε προγνώσεις προερχόμενες από μεγίστως συσχετισμένες μετοχές.

Στην Εικόνα 3.6 παρουσιάζεται ο ψευδοκώδικας για τη δημιουργία των bitMasks και αποθήκευσης αυτών σε μια δομή δεδομένων, το hashing των δυαδικών διανυσμάτων καθώς και η αναζήτηση του εγγύτερου γείτονα δοθέντος ενός query point.

```
procedure INITIALIZECOVERING(d, r)
  for v ∈ {0,1}^{r+1} do A[v] := 0^d
  for i := 1 to d do
    m := RANDOM({0,1}^{r+1}\{0})
    for v ∈ {0,1}^{r+1} do A[v]_i := ⟨m, v⟩ mod 2
  end for
end

function BUILDDATASTRUCTURE(S, r)
  D = ∅
  for x ∈ S, v ∈ {0,1}^{r+1}\{0} do
    D[x ∧ A[v]] := D[x ∧ A[v]] ∪ {x}
  return D
end
```

```
function NEARESTNEIGHBOR(D, r, y)
  best := ∞
  nn := null
  for v := 1 to 2^{r+1} − 1 do
    for x ∈ D[y ∧ A[BITVEC(v, r + 1)]] do
      if ||x − y|| < best then
        best = ||x − y||
        nn = x
      end if
    end for
    if best ≤ ⌊log(v + 1)⌋ then return nn
  end for
  return null
end
```

***Εικόνα 3.6**: CLSH (ψευδοκώδικας)*

(Εικόνα από [3])

3.4 Ροές Δεδομένων

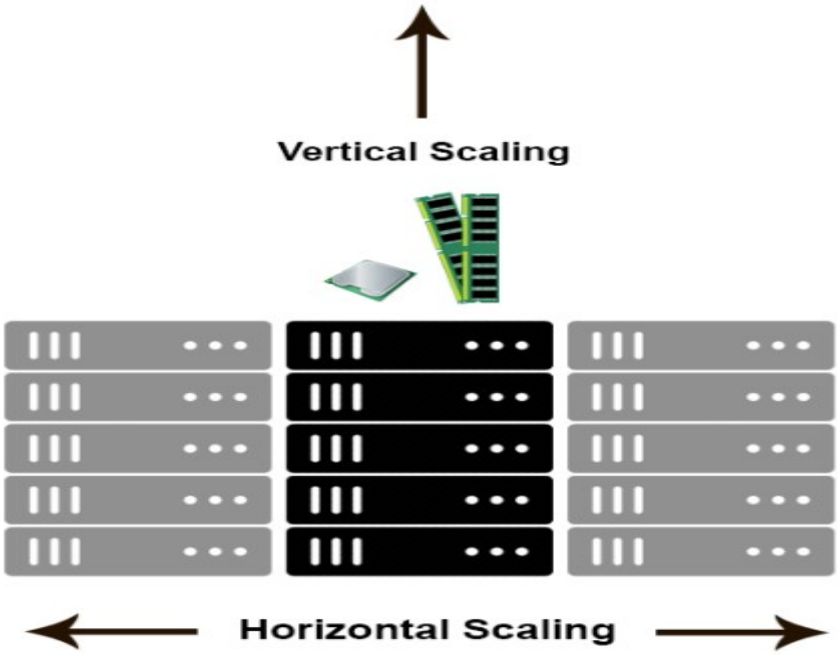
Ροές δεδομένων (Streaming data) ονομάζονται τα δεδομένα τα οποία δημιουργούνται με συνεχή ροή από χιλιάδες πηγές δεδομένων που στέλνουν αυτά τα data records ταυτόχρονα και σε μικρό μέγεθος (της τάξης των kilobytes). Αυτά τα δεδομένα πρέπει να επεξεργαστούν διαδοχικά και σταδιακά record-by-record ή με συρόμενα παράθυρα χρόνου (sliding time windows) καθώς οι προαναφερθείσες ροές είναι πιθανώς μη πεπερασμένες και δημιουργούν συνεχώς νέα σημεία, με αποτέλεσμα οι αλγόριθμοι επεξεργασίας τους (streaming algorithms) να μην είναι εφικτό να αποθηκεύσουν ολόκληρη τη ροή και ύστερα να εκτελέσουν υπολογισμούς πάνω σε αυτή. Έτσι οι streaming αλγόριθμοι πρέπει να επεξεργάζονται τα δεδομένα που λαμβάνουν σε ένα “πέρασμα” χωρίς να τα επανεπισκέπτονται ή αφού συγκεντρώσουν ένα υποσύνολο αυτών.

3.5 Κατανεμημένα Συστήματα

Ένα κατανεμημένο σύστημα [4] με απλά λόγια, είναι ένα γκρουπ από υπολογιστές οι οποίοι δουλεύουν μαζί με τέτοιο τρόπο ώστε να δίνουν την ψευδαίσθηση ενός ενιαίου υπολογιστικού συστήματος στο χρήστη. Αυτά τα μηχανήματα έχουνε κοινή κατάσταση, λειτουργούν ταυτόχρονα και μπορούν να αποτύχουν ανεξάρτητα χωρίς να επηρεάσουν το υπόλοιπο σύστημα. Η κατανεμημένη επεξεργασία αποτελεί ένα υποσύνολο της παράλληλης επεξεργασίας, στο οποίο όλες οι cpu έχουν ιδιωτικές τοπικές μνήμες με ξεχωριστούς χώρους διευθύνσεων. Είναι δηλαδή ανεξάρτητοι, δικτυωμένοι υπολογιστές που επικοινωνούν και συντονίζονται μέσω ανταλλαγής μηνυμάτων. Τα μηχανήματα ενός κατανεμημένου συστήματος αλληλεπιδρούν μεταξύ τους για την επίτευξη ενός κοινού στόχου.

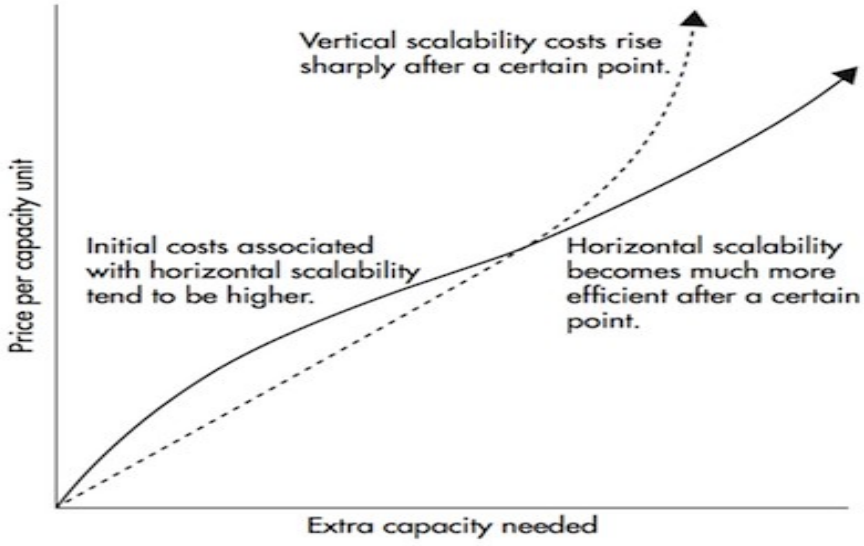
Για να επιτύχουμε την επίλυση ενός προβλήματος σε ένα κατανεμημένο σύστημα, πρέπει να διαμοιράσουμε το πρόβλημα σε διεργασίες (tasks), έτσι ώστε κάθε υπολογιστής τους συστήματος να τρέχει μία ή περισσότερες από αυτές. Τέλος θα πρέπει να συνδυάσουμε όλα τα επιμέρους αποτελέσματα που θα εξαχθούν από αυτά τα tasks, ώστε να λάβουμε την τελική λύση του προβλήματος. Αυτό σημαίνει πως μόνο παραλληλοποιήσιμοι αλγόριθμοι έχουν νόημα και μπορούν να είναι αποδοτικοί σε τέτοια συστήματα.

Τα κατανεμημένα συστήματα είναι σημαντικά γιατί μας επιτρέπουν το horizontal scaling. Σε ένα σενάριο επίλυσης ενός προβλήματος σε έναν υπολογιστή, για να διαχειριστούμε την αυξανόμενη ζήτηση πόρων (π.χ. λόγω αύξησης του πλήθους ροών δεδομένων προς επεξεργασία), θα χρειαζόταν να αναβαθμίσουμε το υλικό (hardware). Αυτό ονομάζεται vertical scaling. Το vertical scaling είναι χρήσιμο όσο μπορεί να υφίσταται, καθώς ύστερα από ένα σημείο θα διαπιστώσουμε πως ακόμα και το καλύτερο hardware ή και ο καλύτερος υπερυπολογιστής δεν είναι σε θέση να ανταπεξέλθει στην επίλυση σύγχρονων προβλημάτων μεγάλων δεδομένων. Το horizontal scaling ουσιαστικά είναι η προσθήκη περισσότερων μηχανημάτων αντί της αναβάθμισης του hardware σε έναν υπολογιστή. Στην Εικόνα 3.7 φαίνεται η οπτικοποίηση του διαχωρισμού μεταξύ horizontal και vertical scaling.



***Εικόνα 3.7**: horizontal & vertical scaling*

(Εικόνα από https://stackoverflow.com/questions/11707879/difference-between-scaling-horizontally-and-vertically-for-databases)



***Εικόνα 3.8:** horizontal vs vertical scaling*

(Εικόνα από [4])

Στην Εικόνα 3.8 παρατίθενται οι καμπύλες του κόστους για το horizontal και vertical scaling συναρτήσει της ανάγκης για αύξηση πόρων. Το horizontal scaling γίνεται σημαντικά φθηνότερο μετά από ένα σημείο αλλά δεν είναι αυτός ο κύριο λόγος της προτίμησης του. Η πραγματική υπεροχή του έγκειται στο γεγονός ότι δεν έχει κάποιο όριο, σε αντίθεση με το vertical scaling. Έτσι στο σενάριο πτώσης της απόδοσης του συστήματος μας, απλά προσθέτουμε σε αυτό ένα ακόμα μηχάνημα. Η δυνατότητα προσθηκών τείνει θεωρητικά στο άπειρο.

4 Apache Storm

4.1 Εισαγωγή

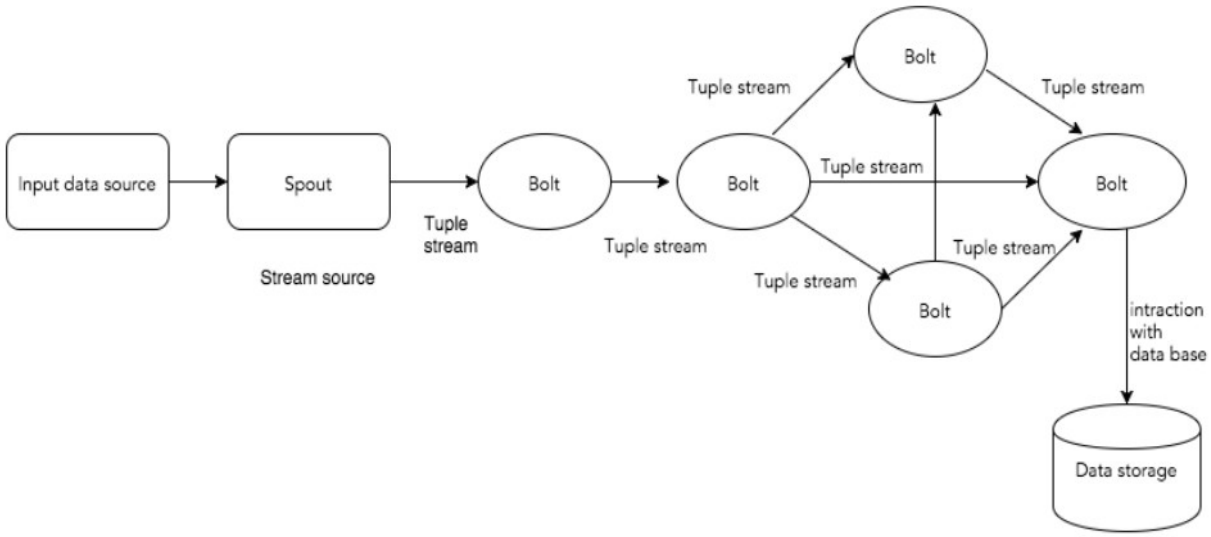
Το apache storm [5] [6] [7] [8] είναι ένα κατανεμημένο σύστημα πραγματικού χρόνου για μεγάλα δεδομένα. Δημιουργήθηκε για την ανεκτική σε σφάλματα (fault-tolerant) και horizontal scalable επεξεργασία τεράστιων όγκων δεδομένων. Έχει τη δυνατότητα υποδοχής ύψιστων ρυθμών ροών δεδομένων. Επιπροσθέτως είναι απλό και παρέχει τη δυνατότητα εκτέλεσης όλων των ειδών παράλληλης επεξεργασίας πάνω σε δεδομένα πραγματικού χρόνου. Τέλος είναι εύκολο στην εγκατάσταση και εγγυάται πως κάθε μήνυμα που εισέρχεται στην τοπολογία θα επεξεργαστεί τουλάχιστον μια φόρα.

Τα πλεονεκτήματα του storm:

- Είναι open source και εύχρηστο. Μπορεί να χρησιμοποιηθεί από μικρές και μεγάλου βεληνεκούς εταιρείες.
- Είναι ανεκτικό σε σφάλματα, ευπροσάρμοστο και αξιόπιστο. Επίσης υποστηρίζει τον προγραμματισμό σε οποιαδήποτε γλώσσα.
- Παρέχει τη δυνατότητα επεξεργασίας ροών δεδομένων σε πραγματικό χρόνο.
- Γρήγορο λόγω της μεγάλης ισχύος επεξεργασίας δεδομένων.
- Μπορεί να διατηρήσει την απόδοση του κατά την αύξηση του φόρτου επεξεργασίας. Αυτό το επιτυγχάνει με τη γραμμική προσθήκη πόρων.
- Εκτελεί ανανέωση αποτελεσμάτων σε δευτερόλεπτα ή λεπτά, ανάλογα με το πρόβλημα. Παρουσιάζει μικρή καθυστέρηση (low latency).
- Εγγυάται την επεξεργασία των δεδομένων ακόμα και αν κάποιος από τους κόμβους του cluster βγει εκτός λειτουργίας ή τα μηνύματα χαθούν.

Μετά την παραπάνω ανάλυση είναι προφανείς οι λόγοι της επιλογής του εν λόγω κατανεμημένου συστήματος για την προσέγγιση επίλυσης του προβλήματος που πραγματευόμαστε σε αυτή τη μελέτη.

4.2 Βασικές Έννοιες



Εικόνα 4.1: *Apache Storm core concept*

(Εικόνα από [5])

Στην Εικόνα 4.1 παρουσιάζεται η βασική ιδέα του Apache Storm. Το storm δέχεται στην είσοδό του ροές δεδομένων πραγματικού χρόνου και τις επεξεργάζεται με τη βοήθεια μιας ακολουθίας μικρών μονάδων επεξεργασίας, εξάγοντας χρήσιμες πληροφορίες. Οι συνιστώσες του πιο αναλυτικά είναι:

- **Tuples:** Είναι η κύρια δομή δεδομένων στο storm. Ουσιαστικά πρόκειται για μια λίστα διατεταγμένων στοιχείων οποιουδήποτε τύπου δεδομένων. Απαρτίζεται από μια αλληλουχία τιμών διαχωρισμένων μεταξύ τους με κόμμα, η οποία εισέρχεται στον cluster.
- **Stream:** Είναι μια ακολουθία από tuples.
- **Spouts:** Πρόκειται για την πηγή της ροής των δεδομένων. Σε ένα spout μπορεί να εισέρχονται δεδομένα από πηγές όπως το Twitter Streaming API, Apache Kafka queue κ.α. . Επίσης μπορούν να διαβάζουν δεδομένα από άλλου είδους πηγές όπως data sets.

- **Bolts:** Είναι οι μονάδες επεξεργασίας του storm. Τα spouts αποστέλλουν δεδομένα στα bolts. Τα bolts επεξεργάζονται τα ληφθέντα δεδομένα δημιουργώντας νέες ροές ως έξοδο, τις οποίες προωθούν σε ένα ή περισσότερα άλλα bolts. Τέλος μπορούν να εκτελούν μια πληθώρα διεργασιών όπως filtering, aggregation, joining και να αλληλεπιδρούν με πηγές και βάσεις δεδομένων.

4.2.1 Τοπολογία

Η ακολουθία της σύνδεσης των spouts και των bolts μεταξύ τους σχηματίζουν μια τοπολογία. Η λογική της real-time εφαρμογής που θέλουμε να υλοποιήσουμε, περιγράφεται στην εν λόγω τοπολογία. Στην ουσία πρόκειται για έναν κατευθυνόμενο γράφο, με τις κορυφές να υποδηλώνουν κάποιο υπολογισμό και τις ακμές κάποια ροή δεδομένων.

Μια τοπολογία ξεκινάει με τα spouts να αποστέλλουν δεδομένα σε ένα ή περισσότερα bolts. Ένα bolt διαθέτει κάποια λογική επεξεργασίας και η έξοδός του μπορεί να αποτελεί είσοδο σε ένα ή περισσότερα άλλα bolts.

Το storm έχει ως στόχο να διατηρεί την τοπολογία ενεργή μέχρι να την τερματίσει ο χρήστης. Τέλος έχει την ικανότητα να τρέξει οποιοδήποτε αριθμό από τοπολογίες του δοθεί.

4.2.2 Tasks

Τα spouts και τα bolts πρέπει να εκτελούνται καταλλήλως σε συγκεκριμένη σειρά ώστε να “αποτυπωθεί” σωστά η λογική του αλγορίθμου/εφαρμογής και να τρέξει η τοπολογία επιτυχώς. Η εκτέλεση κάθε μιας από τις προαναφερθείσες δομικές μονάδες της τοπολογίας, ονομάζεται task. Κάθε spout και bolt μπορεί να απαρτίζεται από πολλαπλά στιγμιότυπα που τρέχουν σε πολλαπλά και αυτοτελή νήματα (threads).

4.2.3 Workers

Μια τοπολογία διεκπεραιώνεται κατανεμημένα με τη βοήθεια των κόμβων εργατών (workers). Πιο συγκεκριμένα, τα tasks διαμοιράζονται ομοιόμορφα στους workers. Ένας worker έχει ως καθήκον να είναι έτοιμος για την ανάληψη εισερχομένων εργασιών και να σταματάει ή να αρχίζει μια διεργασία κατά την άφιξη μιας νέας τέτοιας εργασίας.

4.2.4 Stream Grouping

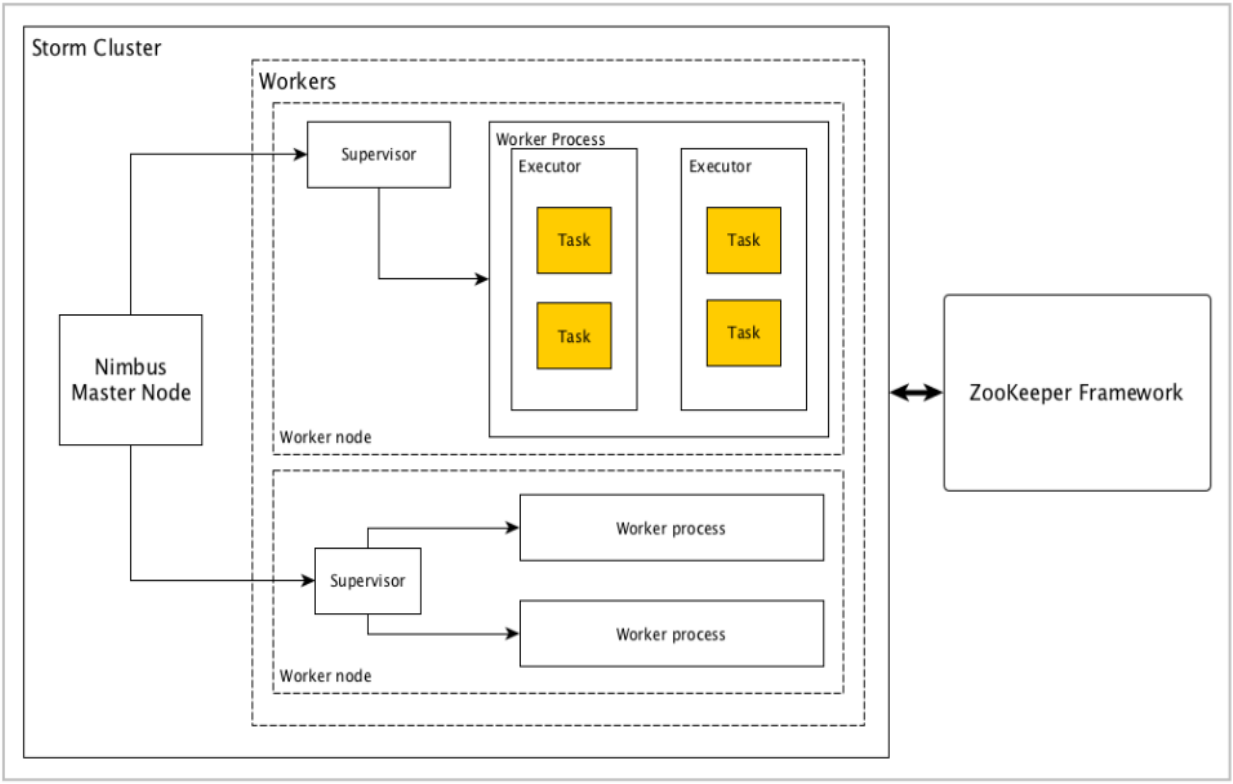
Όπως είπαμε, τα δεδομένα κυκλοφορούν από τα spouts στα bolts ή και μεταξύ των bolts. Το stream grouping υποβάλλει τον τρόπο κίνησης των δεδομένων αυτών στην τοπολογία. Πιο αναλυτικά υπάρχουν οι εξής τρόποι διαμέρισης των tuples:

- **Shuffle grouping:** Τα tuples κατανέμονται ομοιόμορφα και τυχαία στα bolt-tasks.
- **Fields grouping:** Τα tuples διαμοιράζονται με βάση τα πεδία που θα δηλώσουμε. Για παράδειγμα, αν μία ροή θέλουμε να γίνει grouping κατά το πεδίο “stock-id”, τότε τα tuples με το ίδιο “stock-id” θα πηγαίνουν πάντα στο ίδιο task.
- **All grouping:** Τα tuples αντιγράφονται σε όλα τα bolt-tasks.
- **Global grouping:** Όλα τα tuples κατευθύνονται σε ένα και μοναδικό bolt-task. Για την ακρίβεια πηγαίνει στο task με το μικρότερο id.
- **None grouping:** Σε αυτή την κατηγορία δε μας ενδιαφέρει ο τρόπος με τον οποίο θα κατανεμηθούν τα tuples. Ουσιαστικά είναι το ίδιο με το shuffle grouping. Παρόλα αυτά όμως το storm θα αναγκάσει, όπου είναι δυνατό, τα bolts και spouts που στέλνουν tuples με αυτή την κατηγορία grouping και τα bolts-παραλήπτες αυτών, να εκτελεστούν από το ίδιο νήμα.
- **Direct grouping:** Σε αυτή την κατηγορία grouping, ο αποστολέας των tuples αποφασίζει ποιο task του παραλήπτη θα τα λάβει. Τα direct groupings

μπορούν να εφαρμοστούν μόνο σε ροές που έχουν οριστεί ως “direct streams”.

4.3 Αρχιτεκτονική Cluster

Στην Εικόνα 4.2 παρουσιάζεται η εσωτερική αρχιτεκτονική του cluster του storm.



Εικόνα 4.2: *Apache storm cluster design*

(Εικόνα από [5])

Παρατηρώντας την Εικόνα 4.2 διακρίνουμε τις βασικές συνιστώσες που απαρτίζουν έναν storm cluster. Πιο αναλυτικά:

- **Nimbus:** Είναι ο κύριος κόμβος του cluster ενώ όλοι οι υπόλοιποι αποτελούν τους κόμβους εργάτες (worker nodes). Ο κύριος κόμβος κατανέμει τα

δεδομένα στους worker nodes και αναθέτει tasks σε αυτούς. Τέλος αναλαμβάνει την παρακολούθηση σφαλμάτων.

- **Supervisor:** Είναι οι κόμβοι που διεκπεραιώνουν τις οδηγίες του Nimbus. Ένας supervisor μπορεί να διαθέτει και να ελέγχει πολλαπλά worker processes για να ολοκληρώσει τα tasks που του έχει υποβάλλει ο Nimbus.
- **Worker process:** Εκτελεί tasks μιας συγκεκριμένης τοπολογίας. Ουσιαστικά δεν εκτελεί αμέσως τα tasks αλλά εμμέσως. Πιο συγκεκριμένα δημιουργεί executors στους οποίους αναθέτει την αποπεράτωση αυτών. Ένα worker process μπορεί να έχει πολλούς executors.
- **Executor:** Είναι ένα νήμα το οποίο δημιουργείται από ένα worker process. Ένας executor μπορεί να τρέχει ένα ή περισσότερα tasks που αφορούν μόνο ένα συγκεκριμένο spout ή bolt.
- **Task:** Έχει ως ρόλο την επεξεργασία των δεδομένων. Ουσιαστικά πρόκειται για ένα spout ή για ένα bolt.
- **Apache ZooKeeper framework:** Είναι μια υπηρεσία η οποία έχει ως στόχο το συγχρονισμό μίας ομάδα κόμβων (cluster) καθώς και τη διατήρηση των κοινών δεδομένων αυτής. Αυτό πραγματοποιείται με τη βοήθεια ισχυρών τεχνικών συγχρονισμού. Μιας και ο Nimbus είναι “stateless”, η ευθύνη για την παρακολούθηση της κατάστασης των κόμβων του cluster ανατίθεται στον ZooKeeper. Εκτός αυτού ο ZooKeeper βοηθάει στην αλληλεπίδραση των Supervisors με τον Nimbus και είναι υπεύθυνος να διατηρεί τις καταστάσεις τους. Τέλος αποθηκεύει την κατάσταση του συνολικού συστήματος και σε περίπτωση που ο Nimbus αποτύχει, τον επανεκκινεί και τον αναγκάζει να συνεχίσει από το σημείο που είχε μείνει.

4.4 Workflow

Για τη λειτουργία ενός storm cluster είναι αναγκαία η ύπαρξη ενός Nimbus και ενός ή περισσοτέρων Supervisors. Εξίσου σημαντικός κόμβος, καθώς θα αναλάβει το συντονισμό μεταξύ του Nimbus και των Supervisors, είναι και ο Apache ZooKeeper.

Η ροή εργασιών (workflow) του Apache Storm είναι η εξής:

- Ο Nimbus αναμένει για την υποβολή της τοπολογίας σε αυτόν.
- Μετά την υποβολή της τοπολογίας, ο Nimbus θα την επεξεργαστεί και θα συγκεντρώσει όλα τα tasks προς εκτέλεση καθώς και τη σειρά με την οποία πρέπει να διεκπεραιωθούν.
- Έπειτα τα tasks διαμοιράζονται σε όλους τους Supervisors από το Nimbus.
- Οι Supervisors στέλνουν παλμούς (“heartbeats”) στον Nimbus για να τον ενημερώσουν ότι είναι ζωντανοί, σε τακτά χρονικά διαστήματα.
- Αν κάποιος Supervisor πεθάνει, δηλαδή έχει πάψει την αποστολή heartbeats, ο Nimbus αναθέτει τα tasks του σε άλλον supervisor.
- Στην περίπτωση που πεθάνει ο Nimbus, οι Supervisors θα συνεχίσουν να δουλεύουν στα tasks που τους έχουν ανατεθεί.
- Μόλις ολοκληρωθούν όλα τα tasks κάποιου Supervisor, αυτός θα περιμένει να του ανατεθούν καινούρια.
- Όσο η ροή εργασιών κυλούσε ομαλά, ο νεκρός Nimbus έχει ήδη επανεκκινηθεί αυτόματα και συνεχίζει από κει που είχε σταματήσει.
- Πέραν του Nimbus μπορεί να επανεκκινηθεί αυτομάτως και ένας νεκρός Supervisor. Έτσι αφού και ο Nimbus και οι Supervisors μπορούν να επανέλθουν και να συνεχίσουν από κει που είχαν μείνει, μας δίνεται η εγγύηση για την επεξεργασία όλων των μηνυμάτων τουλάχιστον μια φορά.
- Αφού διεκπαιρεωθούν όλες οι τοπολογίες, ο Nimbus περιμένει την υποβολή νέας τοπολογίας και οι Supervisors να τους ανατεθούν νέα tasks.

5 Εφαρμογή του CLSH Αλγορίθμου στην Πρόγνωση Μετοχών

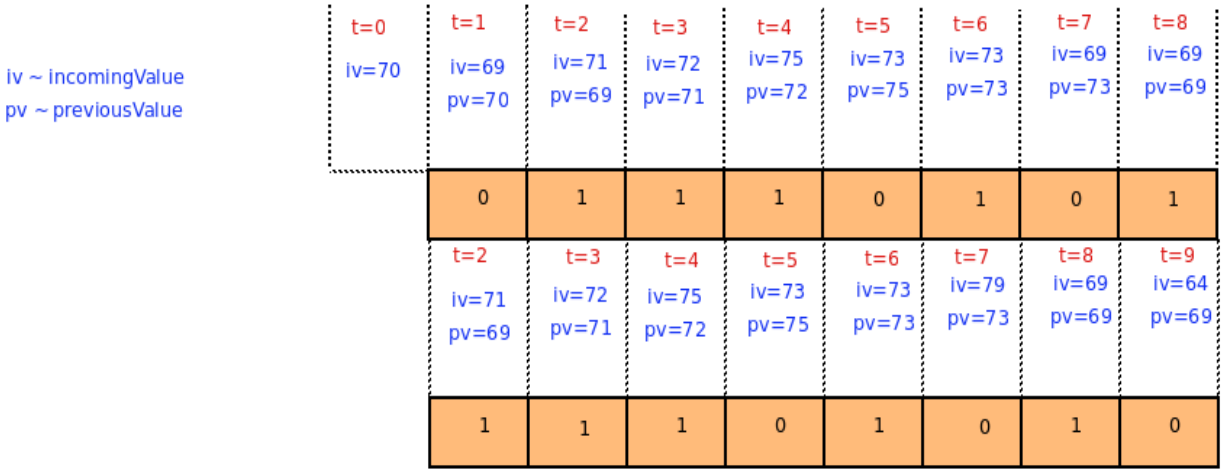
Όπως προαναφέραμε ο αλγόριθμος που θα εφαρμοστεί για την πρόγνωση μετοχών θα πρέπει να είναι streaming και παραλληλοποιήσιμος ώστε να εφαρμοστεί αποδοτικά στο framework Apache Storm. Ο αλγόριθμος που θα επιλέξουμε για να συνεχίσουμε τη μελέτη μας θα είναι ο covering LSH, καθώς υπερτερεί έναντι του κλασικού λόγω έλλειψης false negatives, με μικρό ή και καθόλου τίμημα κατά της απόδοσης.

5.1 Το Μοντέλο του Sliding Window που θα Χρησιμοποιήσουμε

Όπως αναφέραμε στο κεφάλαιο 3.6 ένας αλγόριθμος που επεξεργάζεται ροές δεδομένων, δεν είναι δυνατόν να τις αποθηκεύσει ολόκληρες. Τα δεδομένα αυτών θα πρέπει να επεξεργάζονται record-by-record ή να κρατάει ένα υποσύνολο αυτών σε κάποιο sliding window και να εκτελεί υπολογισμούς πάνω σε αυτό. Το παράθυρο αυτό θα πρέπει να ανανεώνεται, απελευθερώνοντας μνήμη που είχαν καταλάβει παλιά δεδομένα και εισάγοντας τα νεοεισερχόμενα.

Εν προκειμένη περίπτωση επιστρατεύουμε ένα binary sliding window στο οποίο αναθέτονται οι binary τιμές 0 (κάθοδος), 1(άνοδος, σταθερότητα) και ανανεώνεται “in a streaming way” σε κάθε νεοεισερχόμενη τιμή (ανά second). Το εν λόγω παράθυρο θα χρησιμοποιηθεί και για την επικύρωση των υποψήφιων συσχετισμένων ζευγών μετοχών, με βάση την απόσταση Hamming (candidates validation).

Στην Εικόνα 5.1 βλέπουμε ένα στιγμιότυπο του μοντέλου sliding window που χρησιμοποιούμε.

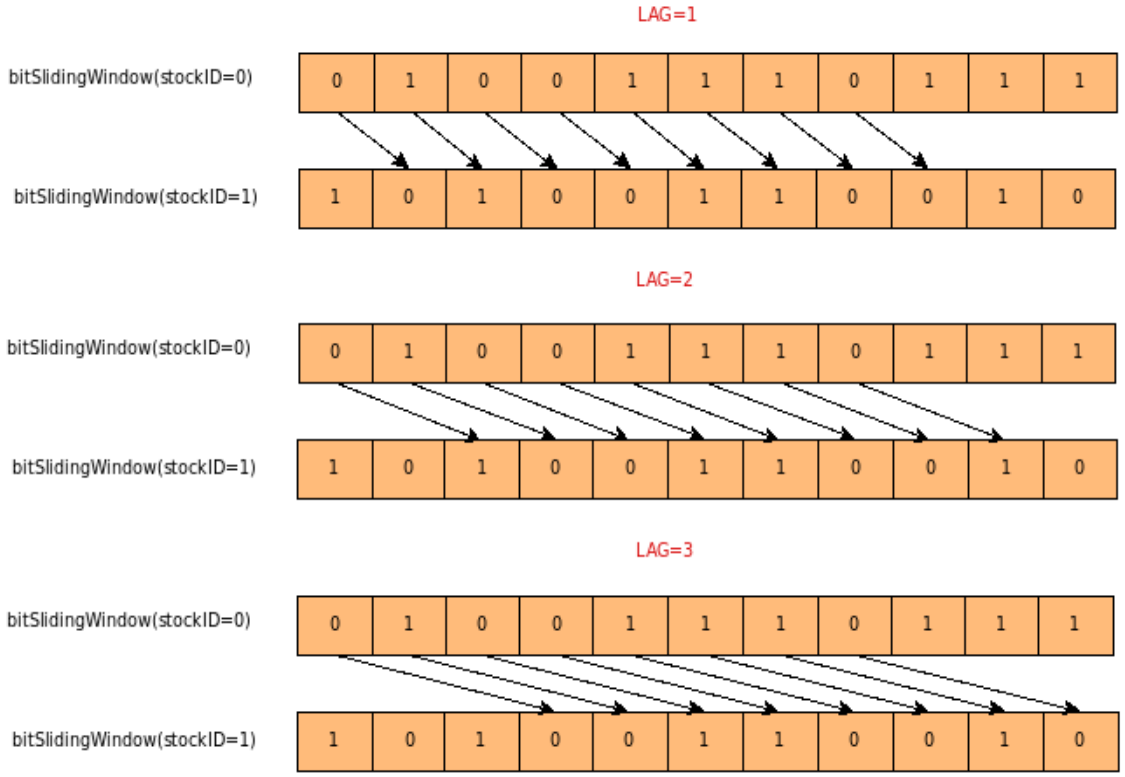


Εικόνα 5.1: *στιγμιότυπο sliding window*

5.2 Σύγκριση Μετοχών

Η συσχέτιση – βαθμός ομοιότητας των ροπών των μετοχών προκύπτει όπως εξηγήσαμε από τη μετρική της απόστασης Hamming. Για να προσδιορίσουμε αυτή την απόσταση πρέπει να προβούμε στη σύγκρισή μεταξύ των δυαδικών ακολουθιών των ζευγών. Επειδή στόχος μας είναι η πρόγνωση των ροπών των μετοχών, μας ενδιαφέρει η χρονοκαθυστερημένη (lagged) συσχέτιση αυτών. Πιο συγκεκριμένα αναζητούμε ομοιότητες μεταξύ υποσυνόλων των δυαδικών ακολουθιών των sliding windows, οι οποίες ανήκουν σε διαφορετικά χρονικά στιγμιότυπα. Έτσι αν εντοπιστεί συσχέτιση μεταξύ των άνωθεν ακολουθιών, μια εκ των οποίων είναι καθυστερημένη χρονικά σε σχέση με την άλλη, τότε αυτό σημαίνει πως πιθανώς η μία να ακολουθεί την πορεία της άλλης με τη συγκεκριμένη χρονοκαθυστέρηση (lag), δίνοντάς μας τη δυνατότητα πρόβλεψης της μέσω της προπορευόμενης.

Η σύγκριση θα γίνεται σε προκαθορισμένα lag από το χρήστη με σκοπό την εύρεση των ζευγών που έχουν συσχέτιση πάνω από ένα κατώφλι. Για παράδειγμα, στην Εικόνα 5.2, έστω ότι θέλουμε να κάνουμε σύγκριση σε el2compare=8 στοιχεία επιτρέποντας max_lag=3, τότε θα πρέπει να διατηρούμε ένα sliding window μεγέθους 11 στοιχείων.



Εικόνα 5.2: *Σύγκριση των δυαδικών υποσυνόλων των μετοχών*

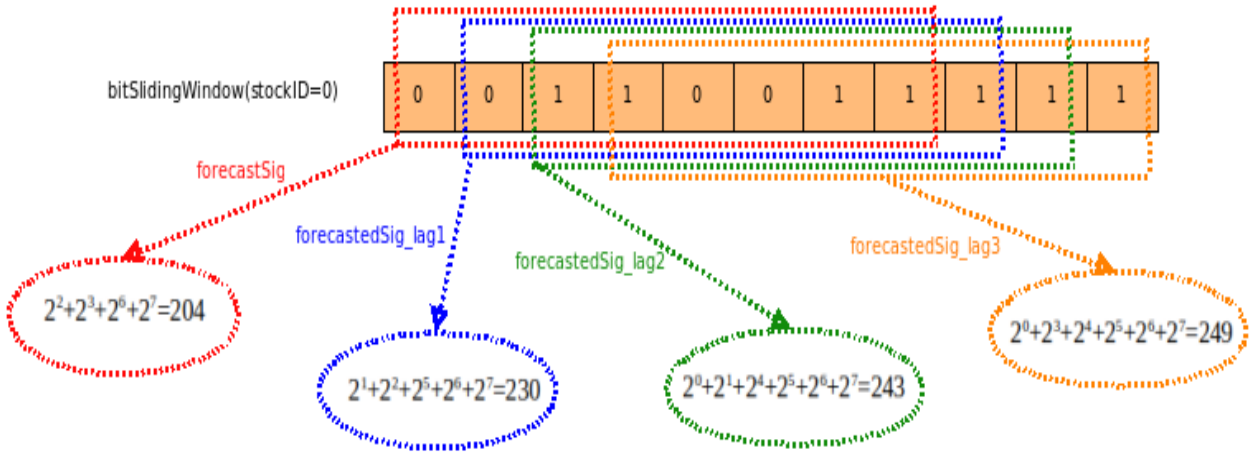
5.3 Υπολογισμός των Signatures

Ουσιαστικά πρόκειται για την μετατροπή των δυαδικών ακολουθιών στους ακεραίους στους οποίους αυτές αντιστοιχούν. Τα signatures θα λάβουν μέρος στο σχηματισμό των τιμών κατακερματισμού (hash values) από τις συναρτήσεις κατακερματισμού (hash functions), όπως θα δούμε στη συνέχεια. Έτσι μετατρέποντάς τα σε ακεραίους μειώνουμε το μέγεθος της μνήμης που χρειάζεται για την ταυτοποίηση ενός bucket καθώς και το χρόνο εύρεσής του [9].

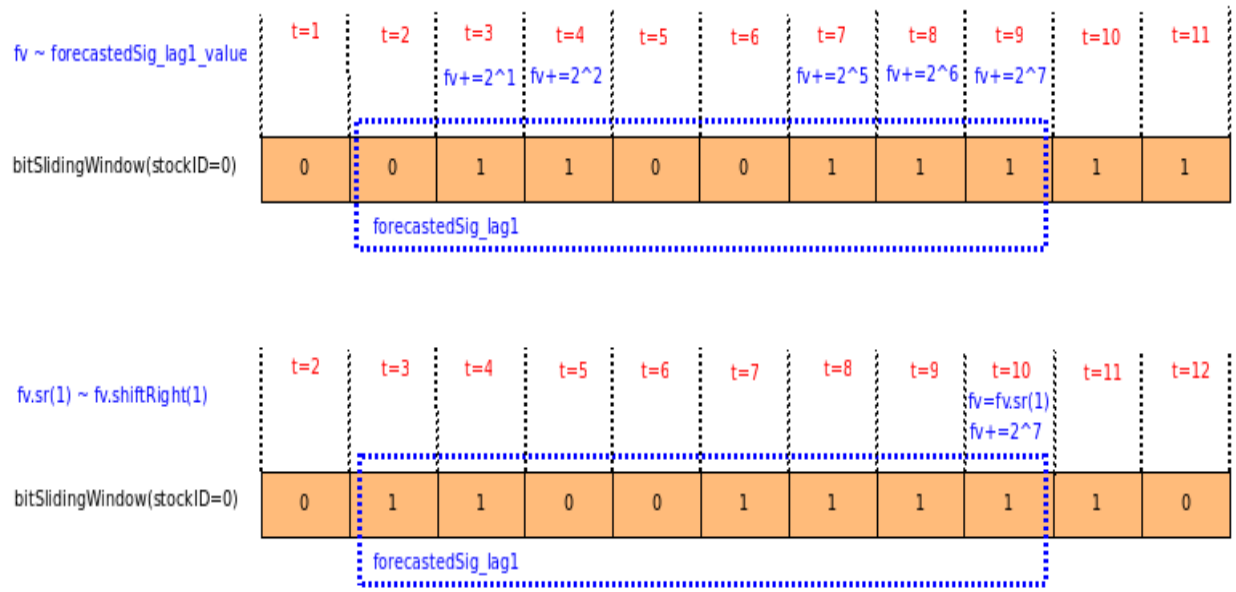
Θα υπάρχουν δύο είδη signatures. Ένα signature για κάθε μετοχή από την οποία θα προβλέψουμε (forecastSig) και ένα για κάθε lag κάθε μετοχής η οποία τίθεται προς πρόβλεψη (forecastedSig) .

Για παράδειγμα έστω ότι θέλουμε να κάνουμε σύγκριση σε el2compare=8 στοιχεία επιτρέποντας max_lag=3, τότε sliding_window_size=11. Στην Εικόνα 5.3 φαίνεται η

λογική υπολογισμού των signatures μιας τυχαίας μετοχής μέσω off-line αλγορίθμου. Στην Εικόνα 5.4 παρουσιάζεται η λογική υπολογισμού ενός εκ των singatures (για το lag=1) της ίδιας μετοχής μέσω on-line αλγορίθμου.



Εικόνα 5.3: *Signature update (off-line algorithm)*

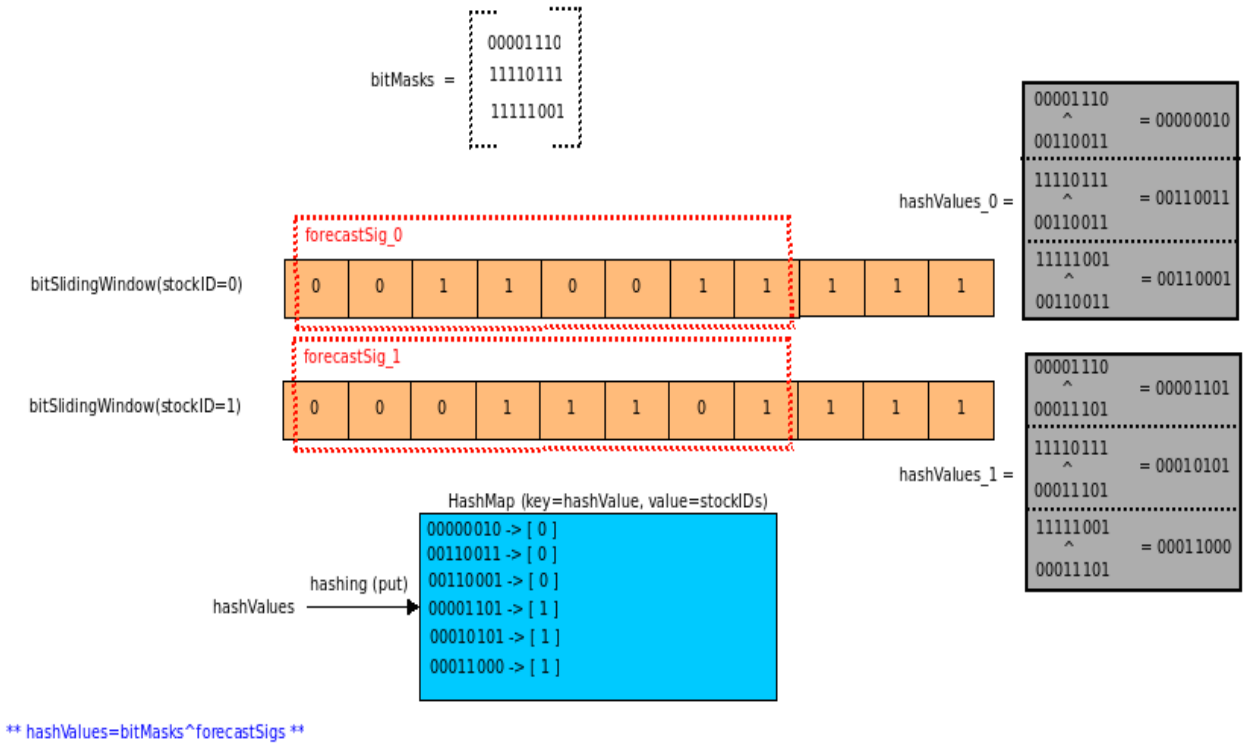


Εικόνα 5.4: *Signature update για lag=1 (on-line algorithm)*

5.4 Hashing Μετοχών

Σε αυτό το στάδιο για κάθε μετοχή από την οποία θα προβλέψουμε πραγματοποιούμε “bit-wise and” (^) μεταξύ του signature της (forecastSig) και κάθε bitMask, έπειτα την εισάγουμε σύμφωνα με τα hash values που προκύπτουν σε έναν hash map.

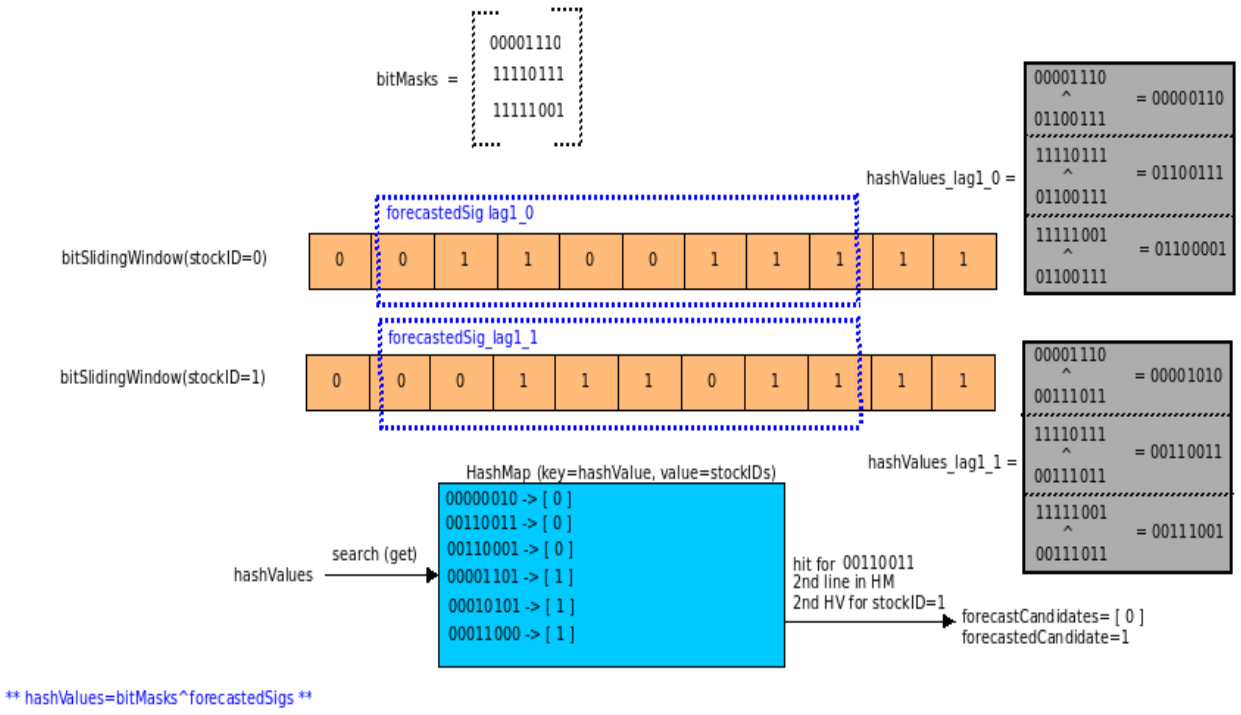
Έστω ένα στιγμιότυπο κατά το οποίο έχουμε 2 μετοχές και έχουμε θέσει r=1, δηλαδή θέλουμε τα ζεύγη που έχουν απόσταση Hamming το πολύ 1. Επίσης sliding_window_size=11 και max_lag=3 οπότε και el2compare=8. Στην Εικόνα 5.5 παρουσιάζεται το hashing για το προαναφερθέν στιγμιότυπο.



Εικόνα 5.5: *Αποθήκευση μετοχών από τις οποίες θα προβλέψουμε*

5.5 Εύρεση Υποψήφιων Συσχετισμένων Ζευγών

Για κάθε μετοχή προς πρόβλεψη, κάθε lag αυτής και κάθε bitMask υπολογίζω τα hash values, με βάση τα οποία κάνω την αναζήτηση στον hash map και λαμβάνω, αν υπάρχουν, τις υποψήφιες μετοχές με τις οποίες είναι συσχετισμένη. Υπάρχει η δυνατότητα να επιλέξουμε αν θα σταματήσει αυτή η αναζήτηση για κάθε μετοχή αφού έχουμε την πρώτη εύρεση συσχέτισής της ή αν θα συνεχίσουμε την αναζήτηση για την εύρεση της μετοχής με την οποία έχει τη μικρότερη απόσταση (βλέπε Ενότητα 3.4.1). Συνεχίζοντας το παράδειγμα που αναφέρθηκε στην Ενότητα 5.4, στην Εικόνα 5.6 παρουσιάζεται η λογική αναζήτησης υποψηφίων ζευγών.

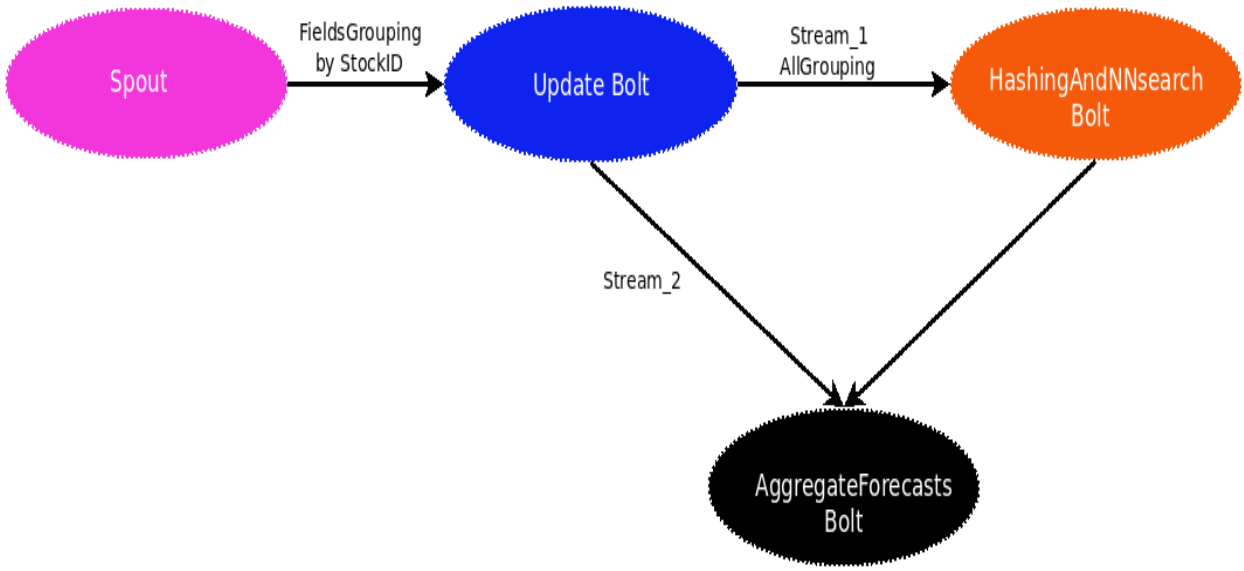


Εικόνα 5.6: *Στιγμιότυπο αναζήτησης υποψήφιων συσχετισμένων ζευγών*

Στην Εικόνα 5.6 η διαδικασία θα συνεχιστεί και για τα εναπομείναντα lags. Στην πραγματικότητα τα bitMasks όπως και τα signatures είναι ακέραιοι αλλά για λόγους κατανόησης τα παραδείγματα στην Εικόνα 5.5 και 5.6 έγιναν στο δυαδικό.

5.6 Κατανεμημένη Υλοποίηση

Σε αυτή την ενότητα θα αναλύσουμε την τοπολογία στην οποία εφαρμόσαμε τον αλγόριθμο, στο Apache Storm framework (Εικόνα 5.7).



Εικόνα 5.7: *Τοπολογία Storm*

Spout

Είναι η πηγή δεδομένων του κατανεμημένου συστήματός μας. Δέχεται δεδομένα από μια εξωτερική πηγή (π.χ. από μια βάση δεδομένων, ένα αρχείο κ.α.). Κάνει emit tuples της μορφής ("newValue","stockID","time"**)**. Ουσιαστικά στέλνει για κάθε δευτερόλεπτο τη νεοεισερχόμενη τιμή κάθε μετοχής.

Update Bolt

Κάνει update τα bitSlidingWindows και τα signatures των μετοχών για κάθε νεοεισερχόμενη τιμή (on-line algorithms).

Stream1: Μόλις τα sliding windows γεμίσουν (time>slidingWindowSize), κάνει emit tuples της μορφής ("forecastSig","forecastedSig","slidingWindow","stockID","time"**)** προς το HashingAndNNsearch bolt.

Stream2: Μόλις time>slidingWindowSize+1 κάνει emit tuples της μορφής ("stockID","incomingTrend","time","startTime"**)**, προς το AggregateForecasts bolt. Έτσι στέλνει τις νεοεισερχόμενες ροπές των μετοχών για να γίνει το validation των προγνώσεων.

HashingAndNNsearch Bolt:

Αρχικά (στο prepare method), το κάθε task αναγνωρίζει το index του μέσα σε μια λίστα από τα taskIDs αυτού του bolt. Έπειτα αναλαμβάνει τα bitMasks που του αναλογούν ως εξής:

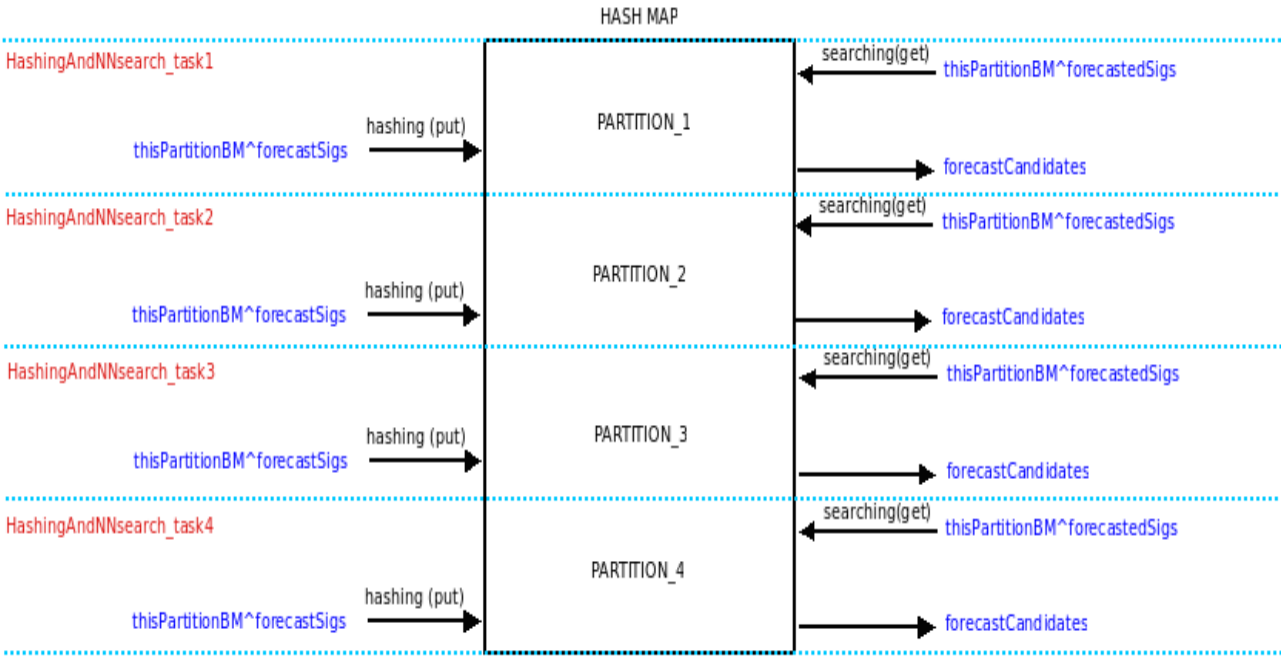
```
for (v=0+thisTaskIndex; v<numOfBitMasks; v+=numOfTasks)
        thisPartitionBM.add(bitMasks[v])
```

Όπου thisTaskIndex είναι το index του τρέχοντος task, numOfTasks είναι το σύνολο των tasks, bitMasks[] είναι ένας πίνακας με το σύνολο των bitMasks και thisPartitionBM είναι μια λίστα με τα bitMasks που θα αντιστοιχιστούν στο τρέχον task.

Για παράδειγμα έστω ότι έχουμε r=2 <=> numOfBitMasks=7 και 4 HashingAndNNsearch tasks. Τότε τα bitMask indexes θα αντιστοιχιστούν στα tasks ως εξής:

```
TASK_1 <= thisPartitionBM[0,4]
TASK_2 <= thisPartitionBM[1,5]
TASK_3 <= thisPartitionBM[2,6]
TASK_4 <= thisPartitionBM[3]
```

Σκοπός είναι κάθε task να αναλάβει ένα partition του ολικού hash map με όσο το δυνατόν ομοιόμορφη κατανομή του work load (τα bitMasks είναι τυχαία και ακολουθούν ομοιόμορφη κατανομή, δηλαδή οι θέσεις από 1-bits που καλύπτουν είναι τυχαίες, όπως και το πλήθος τους σε κάθε bitMask). Έτσι επιτυγχάνουμε πέραν της παραλληλοποίησης του υπολογισμού των hash values, της εισαγωγής-αναζήτησης στο hash map και της επικύρωσης ζευγών, puts & gets σε μικρότερους hash maps καθώς και κατανέμουμε τη χωρική πολυπλοκότητα μεταξύ των κόμβων. Στην Εικόνα 5.8 παρουσιάζεται η οπτικοποίηση της παραπάνω ανάλυσης.



Εικόνα 5.8: *Παράδειγμα διαμέρισης των hash values*

Έτσι κάθε task δέχεται ένα-ένα τα signatures όλων των μετοχών που ενδεχομένως θα προβλέψουν, όλων των μετοχών προς πρόβλεψη και τα sliding windows επίσης όλων των μετοχών. Έπειτα εισάγει στο hash map κάθε μετοχή, από την οποία θα κάνουμε ενδεχομένως πρόβλεψη, για κάθε hash value της (forecastSig^thisPartitionBitMasks). Μόλις για κάποιο time stamp ο αριθμός των προαναφερθεισών μετοχών που έκανε hash γίνει ίσος με τον αριθμό των μετοχών που επεξεργαζόμαστε, τότε ξεκινάει την εύρεση των υποψηφίων ζευγών καθώς και την επικύρωση αυτών (αν υπάρχουν).

Έτσι για κάθε μετοχή προς πρόβλεψη, για κάθε lag της και κάθε hash value της (forceastedSig^ thisPartitionBitMasks) αναζητά στο hash map το κάθε ένα από αυτά. Υπάρχει η δυνατότητα μόλις βρεθεί κάποια μετοχή που μπορεί να την προβλέψει (μετά το validation φυσικά), να σταματήσει η αναζήτηση και να προχωρήσουμε στην επόμενη μετοχή προς πρόβλεψη ή να συνεχίσουμε μέχρι να βρούμε τον εγγύτερο γείτονα, με τη συνθήκη που υπάρχει στον ψευδοκώδικα της Εικόνας 3.6 στην Ενότητα 3.4.1 (αφού εξετάσουμε όλα τα lags και πριν προχωρήσουμε στο επόμενο bitMask). Τέλος για κάθε συσχετισμένο ζεύγος που βρίσκει κάνει emit, προς το **AggregateForecasts Bolt,** tuples της μορφής ("forecastedKey","forecastedValue", "distance", "time"). Μόλις τελειώσει για το εν λόγω time stamp κάνει emit, και πάλι προς το **AggregateForecasts Bolt**, tuples της μορφής ("pairs_send_for_validation", "time") και έπειτα κάνει clear τα δεδομένα που είχε αποθηκεύσει και το αφορούν. Το forecastedKey είναι η μετοχή την οποία θα προβλέψουμε και το forecastedValue η τιμή της πρόβλεψης αυτής, το distance είναι η απόστασή της από την μετοχή μέσω της οποίας θα την προβλέψουμε, το pairs_send_for_validation είναι το πλήθος των ζευγών που τελικά στάλθηκαν προς επικύρωση για αυτό το time stamp και τέλος το πεδίο time είναι το time stamp των δεδομένων που επεξεργαζόμαστε.

AggregateForecasts Bolt

Συναθροίζει τις προβλέψεις και τις νεοεισερχόμενες ροπές των μετοχών, αθροίζει το πλήθος των ζευγών που υπολογίστηκαν. Τέλος πραγματοποιεί επικύρωση των προβλέψεων και κρατάει το συνολικό χρόνο του αλγορίθμου. Όσον αφορά την εύρεση του εγγύτερου γείτονα, τα **HashingAndNNsearch** tasks βρίσκουν τοπικά ελάχιστα. Εδώ αν έχει ενεργοποιηθεί η επιλογή πρόβλεψης από εγγύτερο γείτονα, αποθηκεύονται ως προγνώσεις τα ολικά ελάχιστα με τη βοήθεια του πεδίου "distance" που αναφέραμε στην προηγούμενη παράγραφο.

5.6.1 Παρατηρήσεις

Η τοπολογία έχει άνω όριο στην παραλληλία, για το **HashingAndNNsearch Bolt**, το πλήθος των bitMasks. Θα μπορούσαμε να εξωτερικεύσουμε των υπολογισμό των hash values στο **Update Bolt** αλλά θα είχαμε πολλά περισσότερα tuples προς

αποστολή από αυτό προς το **HashingAndNNsearch Bolt** και θα αυξάναμε το serialization cost κατά την αποστολή σε ξένα nodes. Επίσης θα μπορούσαμε να δημιουργήσουμε ένα επιπλέον-νέο bolt στο οποίο θα εξωτερικεύαμε το candidates validation από το **HashingAndNNsearch Bolt** σε αυτό. Σε αυτή την περίπτωση θα είχαμε λιγότερη ανάγκη για παραλληλία στο **HashingAndNNsearch Bolt** και κατ' επέκταση λιγότερα streams προς replication (λόγω του allGrouping) κατά την αποστολή τους από το **Update Bolt** προς τα tasks του **HashingAndNNsearch Bolt.** Έτσι όμως οι συνθήκες τερματισμού αναζήτησης, και κατά την επιλογή να σταματήσουμε στην εύρεση του πρώτου συσχετισμένου ζεύγους αλλά και κατά την εύρεση του εγγύτερου γείτονα, θα μεταφέρονταν στο νέο bolt. Αυτό θα είχε ως αποτέλεσμα το **HashingAndNNsearch Bolt** να κάνει parse όλο το hash map κάθε φορά. Επίσης θα είχαμε και έξτρα serializations και καθυστερήσεις δικτύου λόγω του νέου bolt που θα εκτελούσε το validation. Τέλος για ένα μέσο ρεαλιστικό σενάριο, επειδή το πλήθος των bitMasks ανεβαίνει εκθετικά με βάση το 2, έχουμε αρκετά περιθώρια παραλληλίας να διαθέσουμε αλλά και όταν φτάσουμε αυτά τα περιθώρια (είτε λόγω μικρής ακτίνας, είτε λόγω παραλληλίας), θα μειωθούν και τα ζεύγη προς validation ανά **HashingAndNNsearch** task αντιστοίχως.

6 Πειράματα

Σε αυτό το κεφάλαιο θα αναλύσουμε τα δεδομένα που χρησιμοποιήσαμε για να τρέξουμε τα πειράματα και τα αποτελέσματα που εξήχθησαν. Η τοπολογία εφαρμόστηκε στον cluster του softnet, ο οποίος αποτελείται από 11 όμοια μηχανήματα Dell PowerEdge R300 με τα εξής χαρακτηριστικά:

- Quad Core Xeon X3323 2.5GHz
- 8GB RAM
- 500GB HDD

6.1 Συνθετικά Δεδομένα (synthetic data set)

Δημιουργήσαμε synthetic data set (sd) 100.000 μετοχών και 360 δευτερολέπτων. Αρχικά δίνουμε μια τυχαία τιμή (100*random[0,1)) για κάθε μετοχή. Έπειτα χωρίζουμε τις μετοχές σε 100 γκρουπ. Ως εκ τούτου κάθε γκρουπ θα έχει 1000 μετοχές. Σε κάθε γκρουπ η πρώτη μετοχή (ταξινομημένες σύμφωνα με τα ακέραια stockIDs) θα είναι ο leader και οι υπόλοιπες οι followers. Έτσι για κάθε second>0 κάνουμε update αρχικά τον leader κάθε γκρουπ, αναθέτοντας μια τιμή (random [0,1)) η οποία με πιθανότητα 0.5 θα προστεθεί στην προηγούμενη και με ίση πιθανότητα θα αφαιρεθεί (άνοδος-κάθοδος αντίστοιχα). Στη συνέχεια αναθέτω μια τυχαία τιμή για κάθε follower όπως περιέγραψα στον leader. Αν ο leader παρουσίασε άνοδο για το εν λόγω δευτερόλεπτο τότε η προαναφερθείσα τιμή κάθε follower θα προστεθεί στην προηγούμενη τιμή του, αλλιώς θα αφαιρεθεί. Αυτός ο συντονισμός (following) θα πραγματοποιείται με πιθανότητα 0.9, ενώ με πιθανότητα 0.1 οι followers θα εκτελούν αντίστροφη πορεία από τον leader. Έπειτα αφού εκτελεστούν τα παραπάνω για όλα τα δευτερόλεπτα κάνουμε τους followers randomShift[1,10] για να προσομοιώσουμε τα lags. Οι τιμές που θα γίνουν padding στην αρχή της λίστας των τιμών κάθε follower, θα είναι η τιμή του για time=0, η οποία θα γίνει αντιγραφή ίσες φορές με το αντίστοιχο randomShift. Τέλος αποθηκεύουμε σε αρχείο

τα πρώτα 360 δευτερόλεπτα κάθε μετοχής. Οι παραπάνω τυχαίες μεταβλητές ακολουθούν ομοιόμορφη κατανομή.

6.2 Σημειογραφία και Πληροφορίες

Συμβολισμοί:

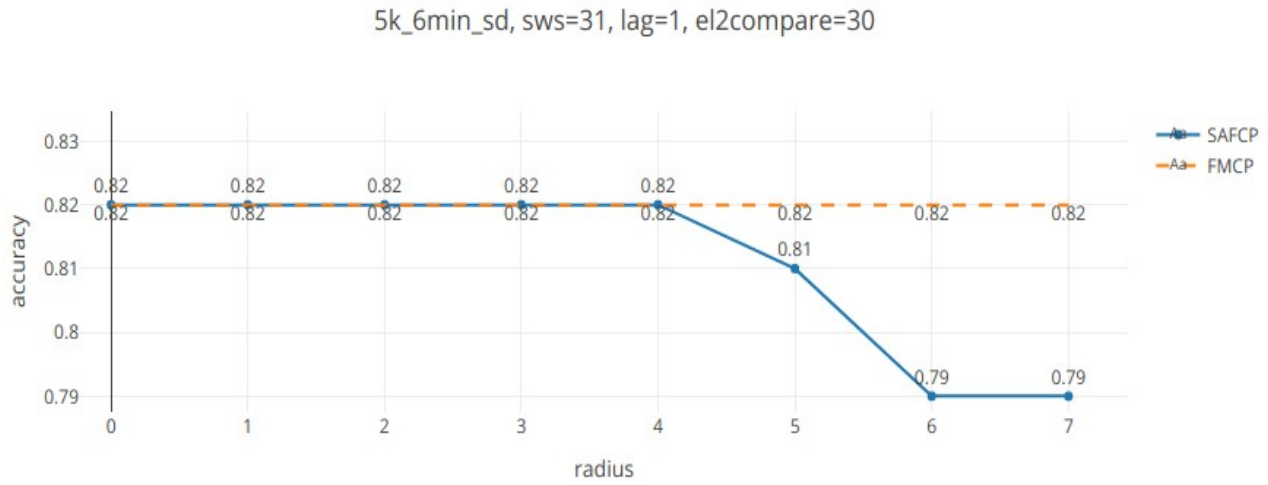
- **SAFCP:** stop at first correlated pair ~ σταματάμε στο πρώτο συσχετισμένο ζεύγος
- **FMCP:** find max correlated pair ~ συνεχίζουμε μέχρι την εύρεση ζεύγους με τη μέγιστη συσχέτιση
- **Xk_6min_sd:** X*1000 stocks, 6 minutes, synthetic data set ~ X*1000 μετοχές διάρκειας 6 λεπτών συνθετικού data set
- **sws:** sliding window size ~ μέγεθος sliding window
- **el2compare:** elements to compare in each lag ~ στοιχεία προς σύγκριση σε κάθε lag
- **r:** radius – the desirable ceiling for Hamming distance ~ το επιθυμητό άνω όριο για την απόσταση Hamming

Το mean validated pairs (θα το συναντήσουμε σε γραφικές παρακάτω) είναι το μέσο πλήθος των ζευγών ανά δευτερόλεπτο εισόδου που στάλθηκαν προς επικύρωση συσχέτισης και τελικά υπολογίστηκαν (για παράδειγμα, ένα ζεύγος μπορεί να στάλθηκε προς έλεγχο ενώ ο συγκεκριμένος συνδυασμούς kye1_key2_lag είχε υπολογιστεί ξανά στο παρελθόν, με αποτέλεσμα να ακυρωθεί η τρέχουσα διαδικασία ελέγχου). Όταν λέμε μέσο πλήθος εννοούμε το συνολικό πλήθος διά το πλήθος των διακριτών δευτερόλεπτων του data set που πέρασαν από το **HashingAndNNsearch Bolt.** Αυτό ισχύει και για το mean forecasts (επίσης θα το δούμε σε γραφικές παρακάτω).

Τέλος όσον αφορά τα bitMasks, είναι στατικά για κάθε συνδυασμό el2compare-r, δηλαδή υπολογίστηκαν μια φορά και τα φορτώνουμε από αρχείο. Έτσι θα έχουμε ξεκάθαρη εικόνα στις συγκρίσεις.

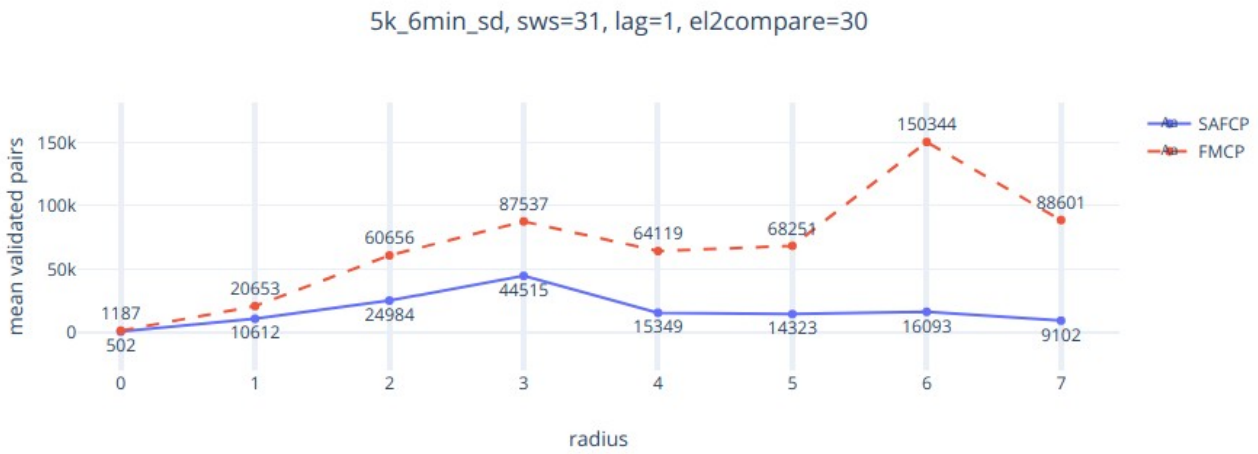
6.3 Εξερευνώντας τη Φύση του Data set και του Αλγορίθμου

Τα πειράματα πραγματοποιήθηκαν σε έναν κόμβο και με παραλληλία 1 σε κάθε bolt, μιας και σε αυτή την ενότητα δεν επικεντρωνόμαστε στο scalability.



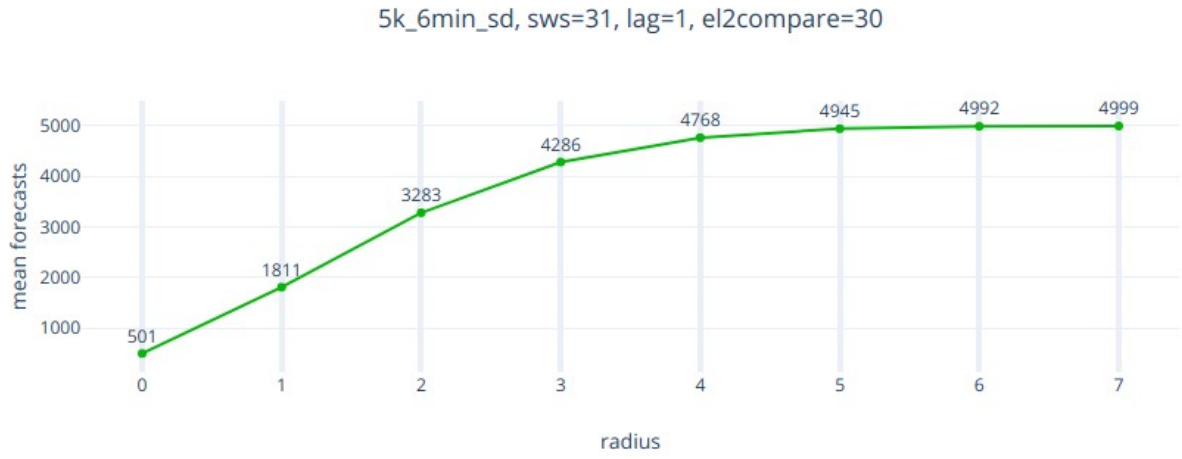
Εικόνα 6.1: *accuracy-radius for el2compare=30*

Στην Εικόνα 6.1 παρατίθεται η γραφική παράσταση, για el2compare=30, στην οποία φαίνεται το accuracy των προβλέψεων καθώς αυξάνεται η ακτίνα της απόστασης Hamming. Βλέπουμε ότι αρχικά οι δυο καμπύλες συμπίπτουν ενώ για ακτίνα μεγαλύτερη του 4 αρχίζουν να αποκλίνουν. Αυτό συμβαίνει καθώς όσο αυξάνεται η ακτίνα τόσο μειώνεται η ελάχιστη επιτρεπτή συσχέτιση μεταξύ ζευγών που ζητάμε, με αποτέλεσμα την αύξηση του πλήθους των χαμηλότερα συσχετισμένων ζευγών. Έτσι κατά την επιλογή SAFCP, σε αντίθεση με την FMCP η οποία θα ψάξει για πρόγνωση από εγγύτερους γείτονες, αυξάνονται οι πιθανότητες για πρόβλεψη από χαμηλά συσχετισμένες μετοχές.



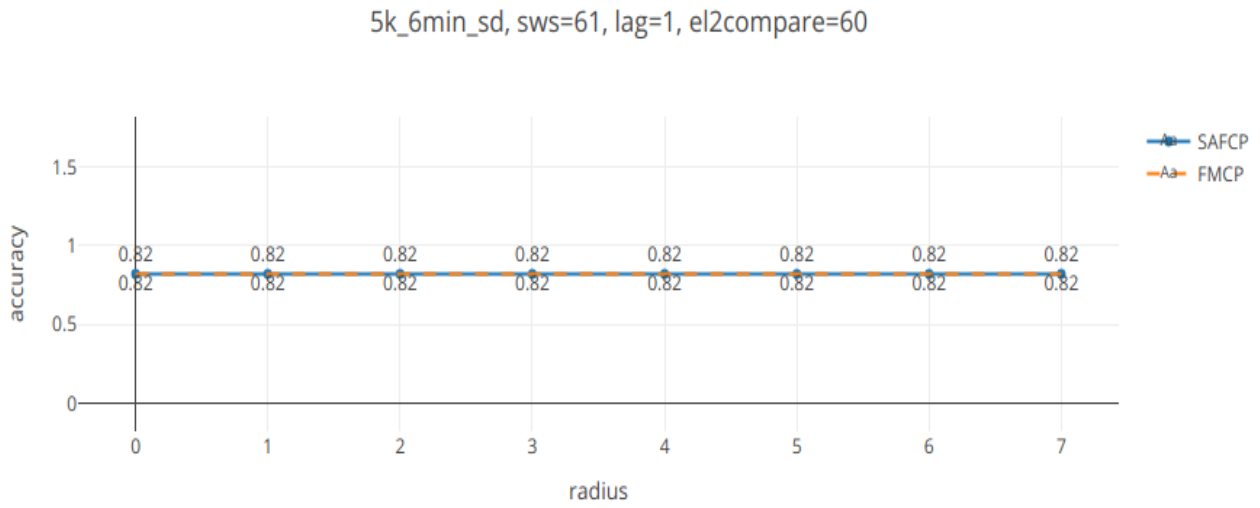
Εικόνα 6.2: *mean validated pairs-radius for el2compare=30*

Στην Εικόνα 6.2, παρατηρούμε αυξομειώσεις των mean validated pairs καθώς αυξάνεται η ακτίνα. Κατά την αύξηση της ακτίνας, όπως προαναφέραμε, μειώνεται και η ελάχιστη επιτρεπτή συσχέτιση που απαιτούμε (όσο μικρότερο το el2compare τόσο πιο ραγδαία). Αυτό αφενός σημαίνει πιθανώς πιο εύκολη εύρεση συσχετισμένου ζεύγους άρα και παύση αναζήτησης, αφετέρου όμως συνεπάγεται και αύξηση των false positives.



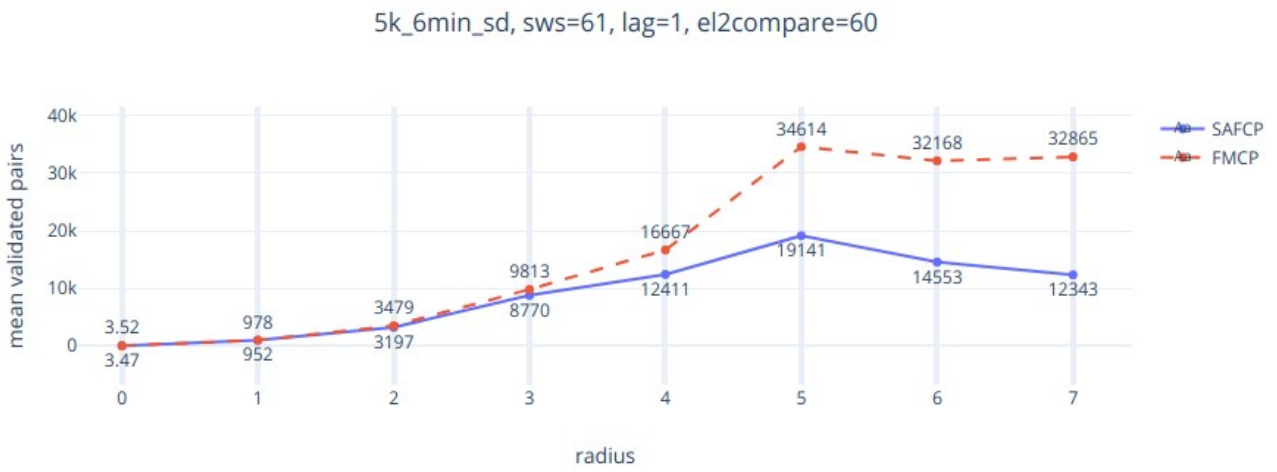
Εικόνα 6.3: *mean forecasts-radius for el2compare=30*

Στην Εικόνα 6.3 παρατίθεται το μέσο πλήθος προβλέψεων συναρτήσει της ακτίνας για el2compare=30. Βλέπουμε πως όσο αυξάνεται η ακτίνα αυξάνονται και οι προβλέψεις.



Εικόνα 6.4: *accuracy-radius for el2compare=60*

Στην Εικόνα 6.4 έχουμε και πάλι τη γραφική παράσταση “accuracy-radius” αλλά αυτή τη φορά για el2compare=60. Σε αντίθεση με την Εικόνα 6.1, εδώ έχουμε πλήρη σύμπτωση των καμπυλών. Αυτό οφείλεται στο γεγονός της αύξησης των στοιχείων προς σύγκριση (el2compare από 30 σε 60) το οποίο οδηγεί σε πιο μικρή πτώση της ελάχιστης επιτρεπτής συσχέτισης που απαιτούμε από τον αλγόριθμο, κατά την αντίστοιχη αύξηση της ακτίνας.

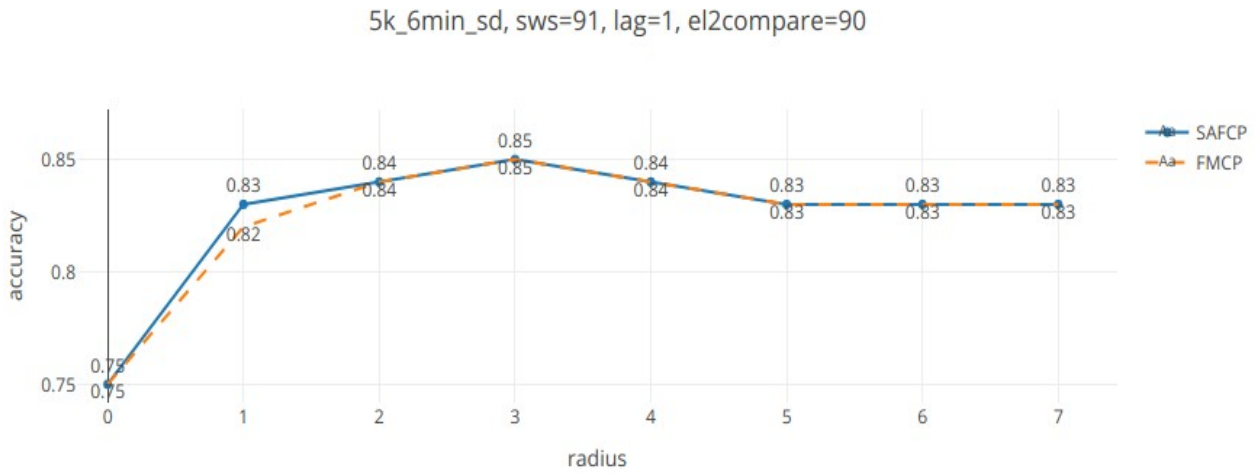


Εικόνα 6.5: *mean validated pairs-radius for el2compare=60*

Στην Εικόνα 6.5 και πάλι παρατηρούμε αυξομειώσεις στα mean validated pairs για τους λόγους που αναφέρθηκαν στην ανάλυση της Εικόνας 6.2, ωστόσο αυτή τη φορά είναι πιο ομαλές. Η εξομάλυνση αυτή οφείλεται στην αύξηση του el2compare, η οποία δεν καθιστά τόσο ραγδαία τη μείωση της ελάχιστη επιτρεπτής συσχέτισης που ζητάμε από τον αλγόριθμο, κατά την αντίστοιχη αύξηση της ακτίνας.

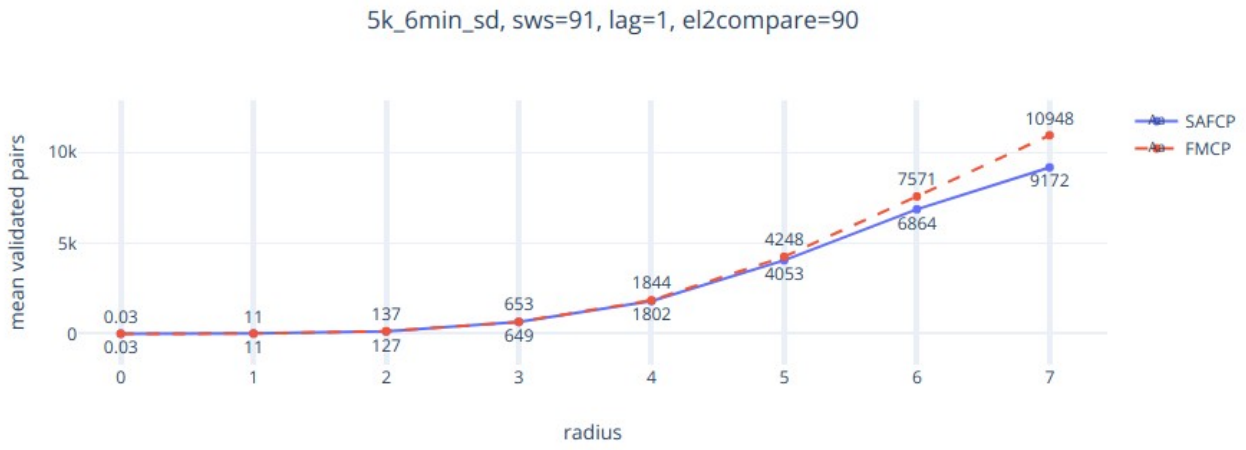
Εικόνα 6.6: *mean forecasts-radius for el2compare=60*

Στην Εικόνα 6.6 έχουμε το μέσο πλήθος προγνώσεων συναρτήσει της ακτίνας, αλλά αυτή τη φορά για el2compare=60. Είναι εμφανής και πάλι η αύξηση των προβλέψεων κατά την αύξηση της ακτίνας.



Εικόνα 6.7: *accuracy-radius for el2compare=90*

Στην Εικόνα 6.7 έχουμε τη γραφική “accuracy-radius” για el2compare=90. Παρατηρούμε πως για r=1 το SAFCP έχει υψηλότερο accuracy απ το FMCP. Αυτό συμβαίνει επειδή ο εγγύτερος γείτονας δεν εγγυάται καλύτερη πρόβλεψη απαραίτητα, απλά καλύτερη πιθανότητα για καλύτερη πρόβλεψη. Επειδή όμως, όπως βλέπουμε στην Εικόνα 6.9, το αντίστοιχο μέσο πλήθος προβλέψεων είναι πολύ μικρό (0.34) δεν έχουμε αντιπροσωπευτικό δείγμα. Τέλος παρατηρούμε πως και οι δυο καμπύλες τείνουν στη σταθερότητα όσο αυξάνεται η ακτίνα. Αυτό συμβαίνει καθώς τα mean forecasts αυξάνονται (βλέπε Εικόνα 6.9) με αποτέλεσμα το δείγμα να γίνεται πιο αντιπροσωπευτικό.



Εικόνα 6.8: *mean validated pairs-radius for el2compare=90*

Στην Εικόνα 6.8 παρουσιάζεται η γραφική για “mean validated pairs-radius” για el2compare=90. Βλέπουμε πως όσο αυξάνεται η ακτίνα το μέσο πλήθος ζευγών που επικυρώθηκαν αυξάνεται επίσης. Η αύξηση πλέον είναι μόνιμη και οι καμπύλες δεν παρουσιάζουν αυξομειώσεις.

Εικόνα 6.9: *mean forecasts-radius for el2compare=90*

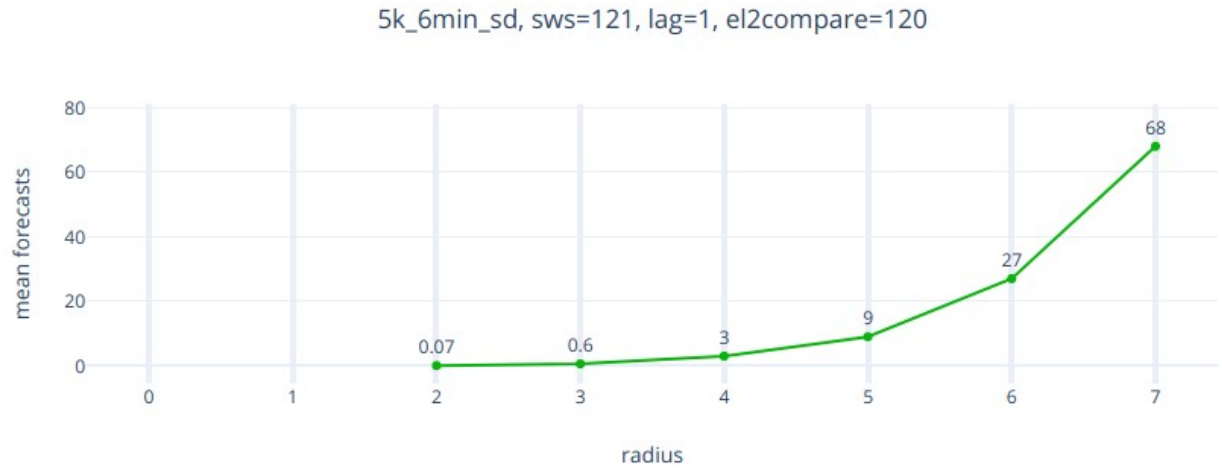
Στην Εικόνα 6.9 φαίνεται η γραφική “mean forecasts-radius” για el2compare=90. Βλέπουμε πως όσο αυξάνεται η ακτίνα παρατηρείται και αύξηση του μέσου πλήθους προβλέψεων.

Εικόνα 6.10: *accuracy-radius for el2compare=120*

Στην Εικόνα 6.10 παρατηρούμε πως μέχρι και για την τιμή 4 της ακτίνας, υφίσταται μια αύξηση στο accuracy το οποίο έπειτα φθίνει. Η αύξηση, του accuracy παρά την αύξηση της ακτίνας οφείλεται στο γεγονός του μη αντιπροσωπευτικού δείγματος του πλήθους προβλέψεων (βλέπε Εικόνα 6.12). Τέλος, η μείωση συμβαίνει καθώς με την αύξηση της ακτίνας επιτρέπουμε πρόγνωση από ζεύγη με όλο και χαμηλότερη συσχέτιση.

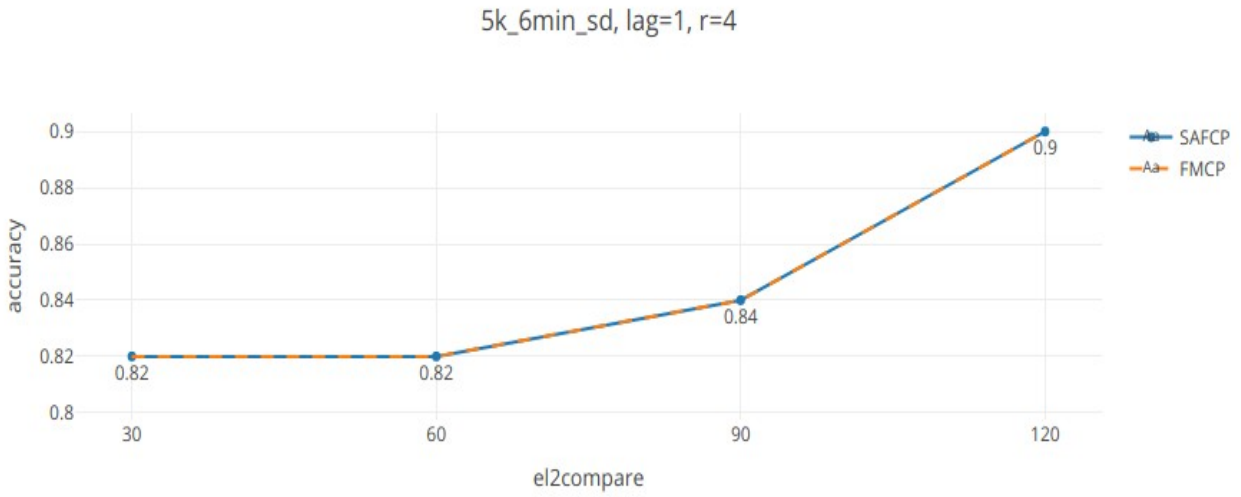
Εικόνα 6.11: *mean validated pairs-radius for el2compare=120*

Στην Εικόνα 6.11 παρατηρούμε ότι οι καμπύλες τείνουν να συμπέσουν. Αυτό συμβαίνει καθώς παρά τις υποφαινόμενες αυξήσεις στην ακτίνα ζητάμε από τον αλγόριθμο υψηλό βαθμό συσχέτισης (λόγω του μεγάλου el2compare), το οποίο οδηγεί σε μικρό πλήθος συσχετισμένων ζευγών καθώς και false positives.



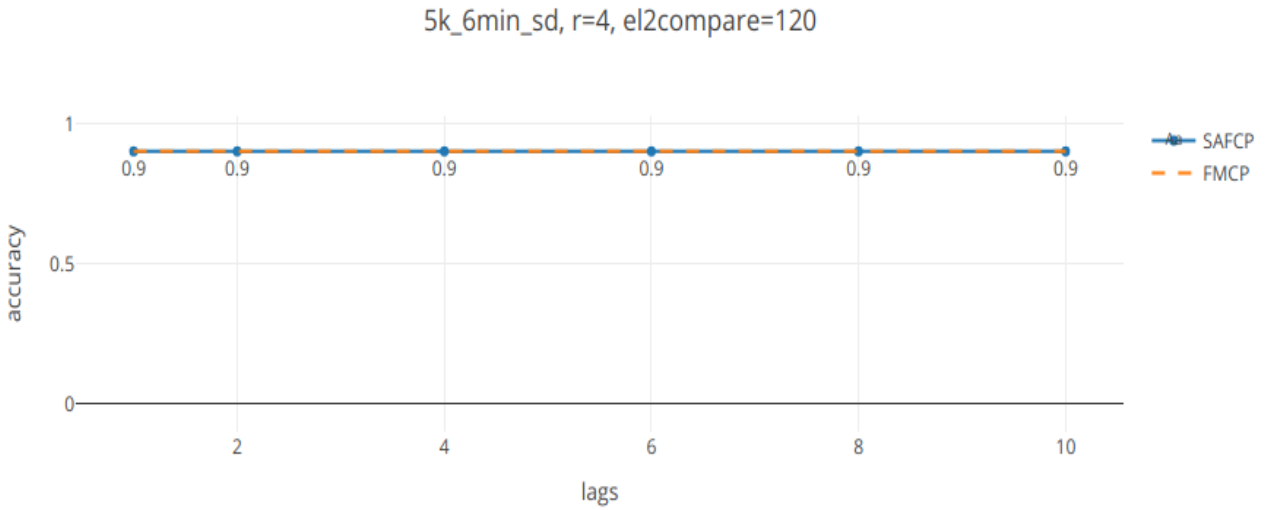
Εικόνα 6.12: *mean forecasts-radius for el2compare=120*

Στην Εικόνα 6.12 βλέπουμε τον τρόπο αύξησης της καμπύλης των μέσων προγνώσεων συναρτήσει της ακτίνας για el2compare=120.



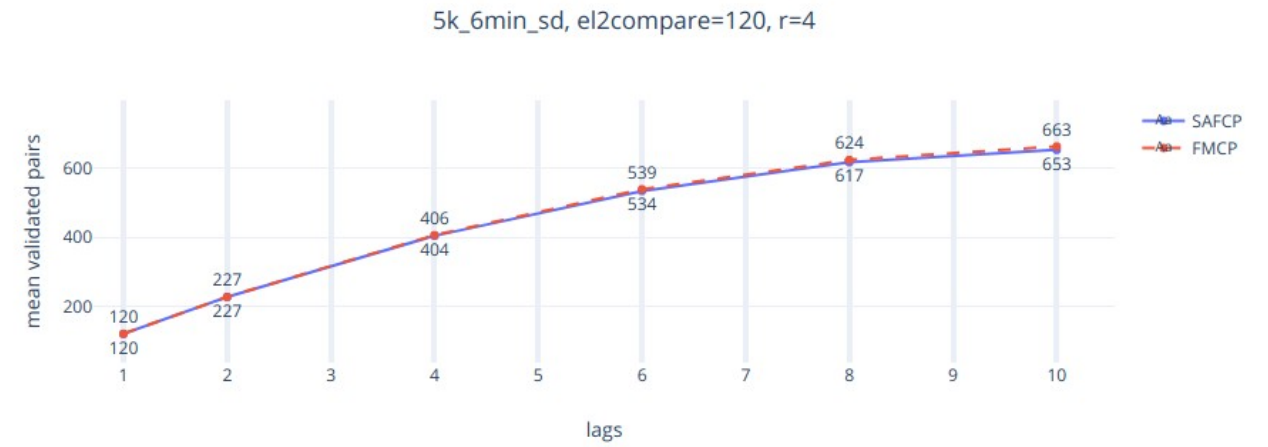
Εικόνα 6.13: *accuracy-el2compare*

Στην Εικόνα 6.13 έχουμε τη γραφική “accuracy-el2compare” για r=4, δηλαδή την τιμή της ακτίνας που παρατηρήθηκε το μέγιστο accuracy (βλέπε Εικόνα 6.10) στα παραπάνω πειράματα. Πράγματι το μεγαλύτερο accuracy το έχουμε για el2compare=120 και r=4, αλλά έχουμε τις λιγότερες προγνώσεις ανά δευτερόλεπτο καθώς και το πλήθος ζευγών προς επικύρωση. Θα συνεχίσουμε τα πειράματα με τις ρυθμίσεις για το μέγιστο accuracy.



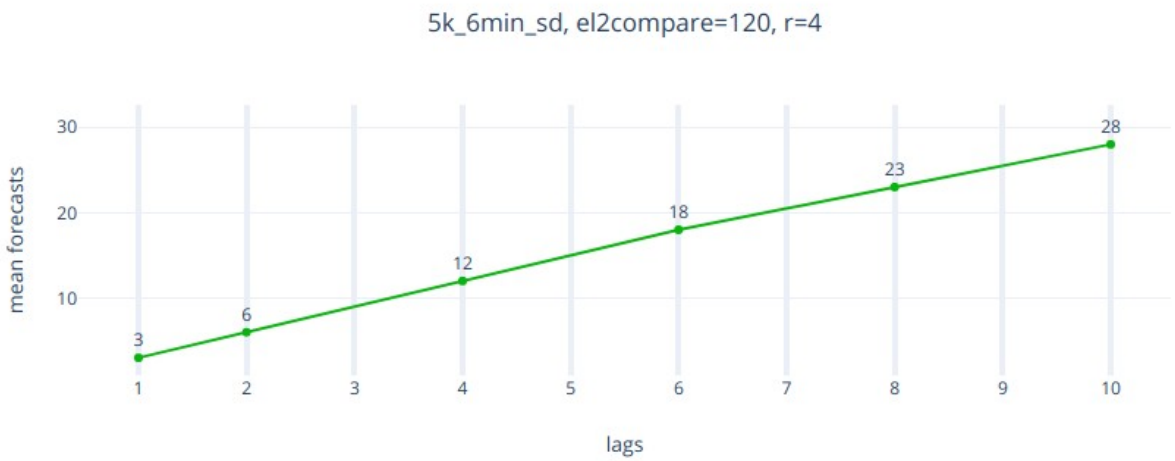
Εικόνα 6.14: *accuracy-lags*

Στην Εικόνα 6.14 βλέπουμε πως όσο αυξάνουμε τα lags η απόδοση μένει σταθερή ως ήταν αναμενόμενο από τη φύση του data set μιας και τα updates όλων των μετοχών ακολουθούν την ίδια διαδικασία με ομοιόμορφα κατανεμημένες πιθανότητες και ύστερα γίνονται shift επίσης με ομοιόμορφη κατανομή.



Εικόνα 6.15: *mean validated pairs-lags*

Στην Εικόνα 6.15 έχουμε τη γραφική “mean validated pairs-lags” για τις ρυθμίσεις που παρατηρήθηκε το μέγιστο accuracy. Η αύξηση των lags υποβάλει μια υπό-γραμμική (sublinear) αύξηση των mean validated pairs.



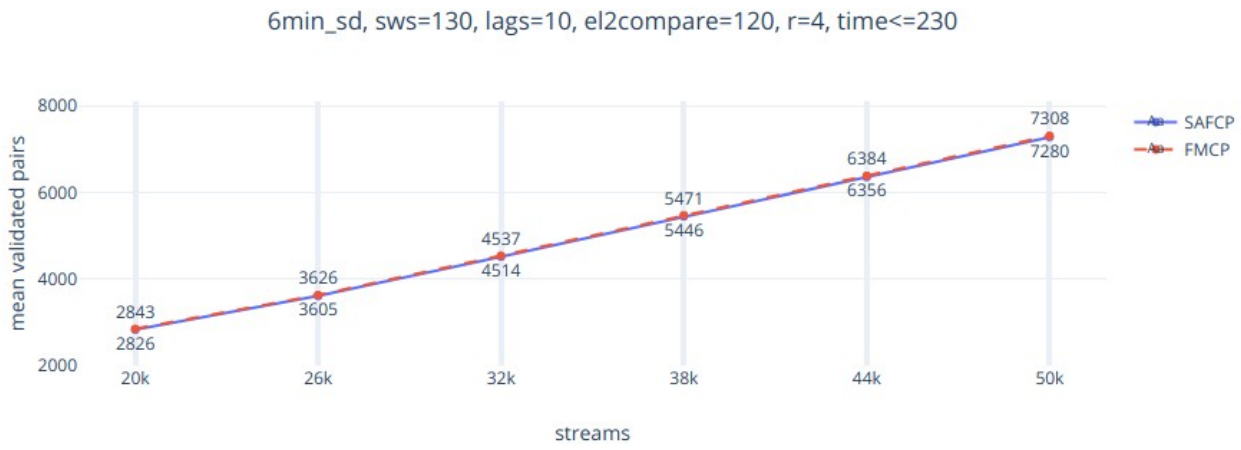
Εικόνα 6.16: *mean forecasts-lags*

Στην Εικόνα 6.16 έχουμε τη γραφική “mean forecasts-lags” για τις ρυθμίσεις που μας προσέφεραν το μέγιστο accuracy. Είναι εμφανές πως οι προγνώσεις αυξάνονται σχεδόν ανάλογα με τα lags. Αυτό εξηγείται επίσης από τη φύση του data set.

6.4 *Scalability ως προς την Αύξηση των Streams*

Εικόνα 6.17: *streams-workers*

Στην Εικόνα 6.17 αυξάνουμε το πλήθος των μετοχών και σταματάμε όταν ο αλγόριθμος δεν προλαβαίνει να προβλέψει εγκαίρως. Επειδή τα πειράματα έγιναν σε data sets 6 λεπτών, το μέγεθος του sliding window είναι μετρήσιμο, έτσι κάθε ένα από αυτά πραγματοποιήθηκαν εντός χρονικού ορίου (time<=360-130=230 sec), μιας και ο χρόνος αρχίζει να μετράει από τη στιγμή που το πρώτο tuple set σταλεί στο **HashingAndNNsearch Bolt** (δηλαδή όταν κάποιο sliding window γεμίσει). Παρατηρούμε πως η αύξηση των μηχανημάτων μας προσφέρει σταθερή αύξηση των streams αλλά σχετικά μικρή. Αυτό είναι λογικό λόγω του ότι σε ένα μηχάνημα δεν έχουμε serializations και καθυστερήσεις δικτύου (το allGrouping απ το **update** bolt στο **HashingAndNNsearch** είναι σχετικά ακριβό) . Η γραμμικότητα των καμπυλών οφείλεται στο γεγονός, πως για τις συγκεκριμένες παραμέτρους, η αύξηση των streams επιφέρει γραμμική αύξηση των υποψηφίων ζευγών προς σύγκριση (βλέπε Εικόνα 6.14).

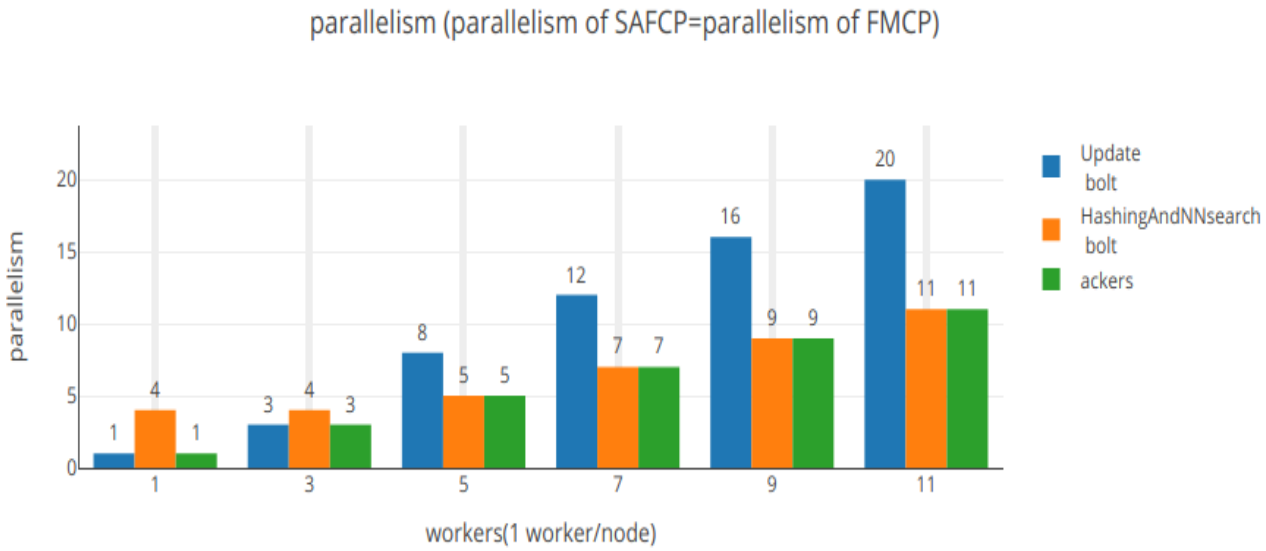


Εικόνα 6.18: *mean validated pairs-streams*

Στην Εικόνα 6.18 έχουμε τη γραφική “mean validated pairs-streams”. Η γραμμικότητα των καμπυλών οφείλεται αφενός στον υψηλό βαθμό συσχέτισης που ζητάμε από τον αλγόριθμο, ο οποίος οδηγεί σε μικρό πλήθος false positives, και αφετέρου στη φύση του data set. Αυτή η φύση, στη συγκεκριμένη περίπτωση, τροφοδοτεί τον αλγόριθμο με σταθερή αύξηση του αριθμού συσχετίσεων κατά την σταθερή αύξηση των streams (κάθε γκρουπ έχει 1000 μετοχές, έναν leader και 999 followers).

Εικόνα 6.19: *mean forecasts-streams*

Στην Εικόνα 6.19 παρατίθεται η γραφική “mean forecasts-streams”. Η αύξηση των μέσων προγνώσεων κατά την αύξηση των streams είναι επίσης γραμμική.



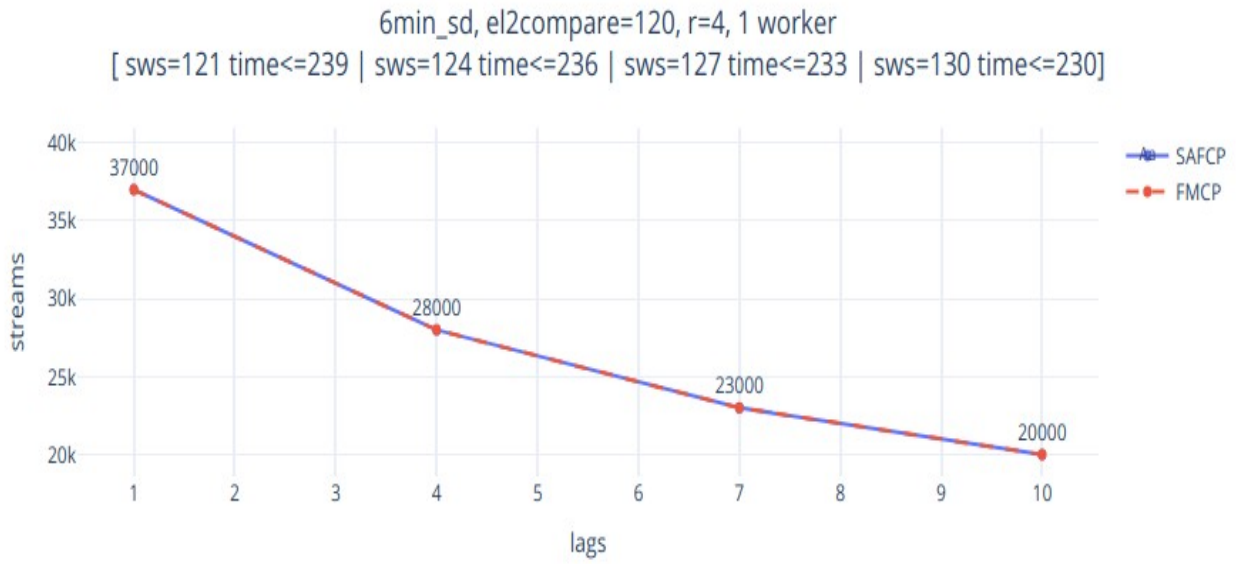
Εικόνα 6.20: *parallelism-workers*

Στην Εικόνα 6.20 φαίνεται η παραλληλία για κάθε μια από τις διακριτές περιπτώσεις της Εικόνας 6.17. Παρατηρούμε πως από τα 5 μηχανήματα και πάνω, το μπλε bolt

χρειάζεται περισσότερη παραλληλία από το πορτοκαλί, καθώς αυξάνεται το serialization cost λόγω του allGrouping κατά την αντίστοιχη αποστολή δεδομένων.

Το accuracy μένει σταθερό στο 0.9 ανεξαρτήτως του πλήθους των μετοχών, κάτι που εξηγείται απ τη φύση του data set αλλά και τον υψηλό βαθμό συσχέτισης που ζητάμε απ τον αλγόριθμο μέσω του συνδυασμού el2compare-r. Επίσης το μεγάλο πλήθος el2compare προσδίδει καλύτερη εκτίμηση ως προς την ύπαρξη πραγματικής συσχέτισης. Αυτές οι ρυθμίσεις δεν αφήνουν πολλά περιθώρια για πρόγνωση από μη πραγματικά συσχετισμένες μετοχές (μη leaders), δηλαδή ψευδό-συσχετίσεις, οι οποίες παρουσίασαν για κάποιο στιγμιότυπο το ζητούμενο ή μεγαλύτερο βαθμό συσχέτισης. Αυτό βέβαια αποκόπτει και πολλές πραγματικές συσχετίσεις.

Στα εναπομείναντα πειράματα, θα κρατήσουμε σταθερό τον αριθμό των workers (1) και θα ελέγξουμε το scalability του αλγορίθμου για διαφορετικές ρυθμίσεις του.



Εικόνα 6.21: *streams-lags*

Στην Εικόνα 6.21 φαίνεται το διάγραμμα “streams-lags”. Ουσιαστικά απεικονίζεται το πλήθος των streams που προλάβαμε να επεξεργαστούμε συναρτήσει της αύξησης των lags.

Εικόνα 6.22: *streams-radius*

Στην Εικόνα 6.22 παρατίθεται η γραφική παράσταση “streams-radius”. Πρόκειται για το πλήθος των streams που προλάβαμε να επεξεργαστούμε συναρτήσει της αύξησης της ακτίνας.

Εικόνα 6.23: *streams-el2compare*

Στην Εικόνα 6.23 έχουμε την γραφική παράσταση που μας γνωστοποιεί το πλήθος των streams που επεξεργαστήκαμε εγκαίρως συναρτήσει την αύξηση του el2compare. Βλέπουμε πως οι καμπύλες όσο μικραίνει το el2compare απομακρύνονται ενώ τείνουν να συμπέσουν όσο μεγαλώνει. Αυτό συμβαίνει καθώς κρατώντας την ακτίνα σταθερή και μειώνοντας το πλήθος των στοιχείων προς σύγκριση, μειώνεται και το κατώφλι συσχέτισης με αποτέλεσμα να έχουμε πολλά περισσότερα συσχετισμένα ζεύγη . Έτσι όσο περισσότερα συσχετισμένα υπάρχουν τόσο πιο εύκολη η εύρεση κάποιου κατά το SAFCP και επομένως η παύση της αναζήτησης.

7 Σχετικές Μελέτες

Η πρόβλεψη της χρηματιστηριακής αγοράς, ουσιαστικά αφορά την προσπάθεια καθορισμού της μελλοντικής αξίας ενός εταιρικού μετοχικού κεφαλαίου. Η επιτυχής πρόβλεψη της μελλοντικής τιμής ενός αποθέματος θα μεγιστοποιήσει τα κέρδη των επενδυτών. Όπως είναι λογικό, η πρόγνωση μετοχών είναι ένας τομέας εξαιρετικής σημαντικότητας και ζήτησης. Αυτό είχε ως αποτέλεσμα την εκτεταμένη μελέτη επί του θέματος, ώστε να προταθούν αλγοριθμικές λύσεις. Σε αυτό το κεφάλαιο θα παρουσιάσουμε περιληπτικά κάποιες από τις μελέτες που προσέγγισαν την επίλυση του προβλήματος αυτού καθώς και σε μια εργασία, η οποία εστιάζει στην εύρεση συσχετίσεων μεταξύ ζευγών μετοχών.

7.1 Πρόγνωση Μέσω Machine Learning Αλγορίθμων

Η Μηχανική εκμάθηση (machine learning) εφαρμόζεται σε μια σειρά από υπολογιστικές εργασίες, όπου τόσο ο σχεδιασμός όσο και ο ρητός προγραμματισμός των αλγορίθμων είναι ανέφικτος. Στο πεδίο της ανάλυσης δεδομένων, η μηχανική εκμάθηση είναι μια μέθοδος που χρησιμοποιείται για την επινόηση πολύπλοκων μοντέλων και αλγορίθμων που οδηγούν στην πρόβλεψη. Τα αναλυτικά μοντέλα επιτρέπουν στους ερευνητές, τους επιστήμονες δεδομένων, τους μηχανικούς και τους αναλυτές να παράγουν αξιόπιστες αποφάσεις και αποτελέσματα και να αναδείξουν αλληλοσυσχετίσεις μέσω της μάθησης από ιστορικές σχέσεις, κρυφά μοτίβα και τάσεις στα δεδομένα. Η εφαρμογή τέτοιων αλγορίθμων επομένως, δε θα μπορούσε να λείπει από το ερευνητικό πεδίο της πρόγνωσης μετοχών. Για παράδειγμα η δημοσίευση “A Hybrid Machine Learning System for Stock Market Forecasting” [10] πραγματεύεται την εφαρμογή ενός υβριδικού συστήματος μηχανικής εκμάθησης βασισμένο σε γενετικό αλγόριθμο και σε Support Vector Machines (SVM), με στόχο την πρόγνωση μετοχών. Επιπροσθέτως στη μελέτη "Stock Market Forecasting Using Machine Learning Algorithms" [11], εκμεταλλεύονται οι χρονικές συσχετίσεις μεταξύ των μετοχών ώστε να επιτευχθούν προβλέψεις επί των ροπών τους για την επόμενη μέρα. Αυτό το εγχείρημα υλοποιείται με Μηχανές Υποστήριξης Διανυσμάτων (SVM). Τέλος στη διπλωματική εργασία "Reinforcement Learning for

Financial Portfolio Management" [14] αναλύεται και εφαρμόζεται ένα σύστημα συναλλαγών, το οποίο βασίζεται σε reinforcement learning, που αποσκοπεί στο portfolio management.

Εν γένει έχουν αναπτυχθεί αρκετές τεχνικές βασισμένες σε deep learning οι οποίες παρέχουν αποδοτικά προβλέψεις μετοχών, αλλά εφαρμόζονται πάνω σε ιστορικά δεδομένα για να το επιτύχουν. Αυτές οι τεχνικές δεν μπορούν να εφαρμοστούν όμως πάνω σε ροές δεδομένων. Κλείνοντας, υπάρχουν πιο αποδοτικές τεχνικές βασιζόμενες σε reinforcement learning οι οποίες όμως αδυνατούν να διαχειριστούν την πρόβλεψη μετοχών σε πραγματικό χρόνο, το οποίο είναι το αντικείμενο που πραγματευόμαστε σε αυτή τη μελέτη με γνώμονα ότι οι συσχετίσεις, μέσω των οποίων θα κάνουμε τις προβλέψεις μας, μπορεί να μεταβάλλονται κατά την πάροδο του χρόνου.

7.2 Πρόγνωση Μέσω Ανάλυσης Συναισθημάτων στα Μέσα Κοινωνικής Δικτύωσης

Η τρέχουσα χρηματιστηριακή αγορά επηρεάζεται από τη συνολική διάθεση της κοινωνίας. Το κοινωνικό αίσθημα για μια εταιρεία μπορεί να είναι μία από τις σημαντικές μεταβλητές που επηρεάζουν την τιμή της μετοχής της. Σήμερα, η εμφάνιση των μέσων κοινωνικής δικτύωσης μας παρέχει μεγάλους όγκους δεδομένων εκ των οποίων μπορούμε να κάνουμε εκτίμηση της επικρατούσας κοινωνικής διάθεσης επί πληθώρας θεμάτων. Ως εκ τούτου, ενσωματώνοντας πληροφορίες από τα κοινωνικά μέσα, σε συνδυασμό με τις ιστορικές τιμές, μπορούμε να καταλήξουμε σε μοντέλα με βελτιωμένη προβλεπτική ικανότητα. Ο στόχος της έρευνας “Sentiment Analysis on Social Media for Stock Movement Prediction” [12], είναι να αναπτυχθεί ένα μοντέλο για την πρόβλεψη της κίνησης των τιμών των μετοχών χρησιμοποιώντας πληροφορίες από κοινωνικά μέσα ενημέρωσης.

7.3 Scaling out Streaming Time Series Analytics on Storm

Στην μελέτη αυτή [13] αναλύεται η προσπάθεια εύρεσης συσχετισμένων ζευγών μετοχών, σε πραγματικό χρόνο, στο framework storm.

Πιο συγκεκριμένα ένας εκ των αλγορίθμων επίλυσης του αρχικού προβλήματος, βασίζεται στον υπολογισμό των Pearson correlation coefficients. Συνεχίζοντας, δεδομένου των εξής δυο ιδιοτήτων του μετασχηματισμού Fourier:

- Διατηρεί την ευκλείδεια απόστασης
- Η πλειοψηφία της ενέργειας του σήματος είναι συγκεντρωμένη στους πρώτους συντελεστές

περιορίζει το υπολογιστικό κόστος εστιάζοντας στην εύρεση των πρώτων συντελεστών Fourier για να προσεγγίσει, χωρίς σημαντική απώλεια πληροφορίας, την αναπαράσταση των αρχικών τιμών των μετοχών, και μάλιστα με τη βοήθεια sliding windows εφαρμόζοντας on-line αλγόριθμο. Τέλος επιστρατεύει μια δομή πλέγματος στην οποία κατανέμει τα ζεύγη με βάση τους πρώτους συντελεστές Fourier που τα αναπαριστούν. Έτσι τα κοντινά ζεύγη, σύμφωνα με την ευκλείδεια απόσταση, θα βρεθούν στο ίδιο κελί. Μόνο τα εν λόγω ζεύγη θα σταλούν προς επικύρωση. Σημαντικό προς αναφορά είναι πως ο αλγόριθμος που μόλις αναλύθηκε δεν παρουσιάζει false negatives.

Πέραν του προαναφερθέντος αλγορίθμου, εφαρμόζει και αναλύει τους αλγορίθμους για την εύρεση συσχετίσεων μέσω Mutual Information και Transfer Entropy:

- Mutual Information: Σύμφωνα με τη θεωρία των πιθανοτήτων και των πληροφοριών, πρόκειται για το μέτρο της αμοιβαίας εξάρτησης δύο τυχαίων μεταβλητών. Σε αντίθεση με τον Pearson correlation coefficient, δεν περιορίζεται σε τυχαίες πραγματικές μεταβλητές και στην ανίχνευση γραμμικών σχέσεων.
- Transfer Entropy: Είναι ένα χρήσιμο μέτρο για asymmetric information transfer καθώς μπορεί να μας κάνει γνωστά τόσο την ποσότητα της κοινής πληροφορίας μεταξύ δυο διανυσμάτων, όσο και την κατεύθυνση εξάρτησής τους (ποιο από τα δύο ακολουθεί το άλλο).

8 Συμπεράσματα

Στην εργασία αυτή μελετήσαμε το πρόβλημα της πρόγνωσης μετοχών σε πραγματικό χρόνο. Πιο συγκεκριμένα εστιάσαμε στην εύρεση ζευγών μετοχών που παρουσιάζουν χρόνο-καθυστερημένη συσχέτιση στις ροπές τους (άνοδος - κάθοδος) . Με αυτό τον τρόπο μπορούμε να βγάλουμε συμπεράσματα για συμπεριφορές leader-follower μεταξύ των μετοχών, δηλαδή για μετοχές οι κινήσεις (ανοδικές-καθοδικές) των οποίων επηρεάζουν με αντίστοιχο τρόπο, αφού παρέλθει κάποιο χρονικό διάστημα, τις κινήσεις άλλων μετοχών. Έτσι εκμεταλλευόμενοι αυτές τις χρόνο-καθυστερημένες συσχετίσεις, μπορούμε να κάνουμε προβλέψεις από τους leaders για τους followers.

Ο naive αλγόριθμος για την επίλυση του προβλήματος παρουσιάζει τετραγωνικό αλγοριθμικό κόστος, καθώς πρέπει να ελέγξουμε όλους τους πιθανούς συνδυασμούς ζευγών για ύπαρξη συσχέτισης. Η προσέγγιση αυτή όμως καθιστά ανέφικτη την έγκαιρη εξαγωγή συμπερασμάτων που θα οδηγήσουν στην πρόβλεψη, καθώς κάνει το σύστημά μας να μη κλιμακώνεται, αφού για να ανταπεξέλθει σε μια σχετικά μικρή αύξηση του όγκου δεδομένων χρειάζεται πολύ μεγάλη αύξηση υπολογιστικής ισχύος. Αυτό το πρόβλημα μας ανάγκασε να στραφούμε σε κάποιο αλγόριθμο με μικρή πολυπλοκότητα. Έτσι επιλέξαμε τον αλγόριθμο CLSH χωρίς false negatives και τον εφαρμόσαμε πάνω στο κατανεμημένο σύστημα Apache Storm το οποίο μας προσφέρει τη δυνατότητα επεξεργασίας μεγάλων δεδομένων σε πραγματικό χρόνο και κατανεμημένα.

Τα πειραματικά αποτελέσματα της υλοποίησης έδειξαν ότι οι δυνατότητες παραμετροποίησης του αλγορίθμου, μας επιτρέπουν να κυμανθούμε μεταξύ ποσότητας και ποιότητας (accuracy) των προβλέψεων. Επίσης ο αλγόριθμος μας έχει ικανοποιητικές επιδόσεις, όσον αφορά την κλιμακωσιμότητα, κατά την αύξηση του εισερχομένου όγκου των δεδομένων. Αυτό οφείλεται αφενός στις δυνατότητες παραλληλοποίησης και το αλγοριθμικό κόστος του αλγορίθμου και αφετέρου στο μικρό πλήθος false positives που μας επιστρέφει.

Βιβλιογραφία

[1] Jure Leskovec, Anand Rajaraman, Jeffrey D. Ullman, Mining of Massive Datasets, 2014
http://infolab.stanford.edu/~ullman/mmds/book.pdf

[2] Aristides Gionis, Data Mining, 2016
http://aris.me/contents/teaching/data-mining-2016/slides/lsh.pdf

[3] Rasmus Pagh, CoveringLSH: Locality-sensitive Hashing without False Negatives, 2016
http://delivery.acm.org/10.1145/3160000/3155300/a29-pagh.html?ip=147.27.115.76&id=3155300&acc=ACTIVE%20SERVICE&key=5641A0C343C36AC1%2E6FEFED88C40FDD50%2E4D4702B0C3E38B35%2E4D4702B0C3E38B35&__acm__=1554671022_29ec5d92e590b145957b1fbf640abd02

[4] Stanislav Kozlovski, A Thorough Introduction to Distributed Systems, 2018
https://medium.freecodecamp.org/a-thorough-introduction-to-distributed-systems-3b91562c9b3c

[5] Apache Storm – TutorialsPoint
https://www.tutorialspoint.com/apache_storm/apache_storm_introduction.htm

[6] Quinton Anderson, Storm real-time processing cookbook, 2013

[7] Jonathan Leibiusky, Gabriel Eisbruch,and Dario Simonassi, Getting Started with Storm, 2012

[8] Apache Storm
http://storm.apache.org/index.html

[9] Rasmus Pagh, Ninh Pham, Scalability and Total Recall with Fast CoveringLSH, 2016
https://arxiv.org/abs/1602.02620

[10] Rohit Choudhry, Kumkum Garg , A Hybrid Machine Learning System for Stock Market Forecasting, 2008
http://waset.org/publications/8952

[11] Shunrong Shen, Haomiao Jiang, Tongda Zhang, Stock Market Forecasting Using Machine Learning Algorithms
https://pdfs.semanticscholar.org/b68e/8d2f4d2c709bb5919b82effcb6a7bbd3db37.pdf

[12] Thien Hai Nguyena, Kiyoaki Shirai, Julien Velcin, Sentiment Analysis on Social Media for Stock Movement Prediction, 2015
https://dl.acm.org/citation.cfm?id=2837518

[13] Nikolaos Pavlakis, Scaling out streaming time series analytics on Storm, 2017
https://dias.library.tuc.gr/view/67653

[14] Antonios Vogiatzis, Reinforcement Learning for Financial Portfolio Management, 2019