ΠΟΛΥΤΕΧΝΕΙΟ ΚΡΗΤΗΣ

ΤΜΗΜΑ ΗΛΕΚΡΟΛΟΓΩΝ ΚΑΙ ΜΗΧΑΝΙΚΩΝ ΥΠΟΛΟΓΙΣΤΩΝ

Ανάλυση Δεδομένων σε ένα Πέρασμα στο RapidMiner

One pass Data-Analytics using RapidMiner

ΠΟΛΥΤΕΧΝΕΙΟ
ΚΡΗΤΗΣ

Αντώνιος Κονταξάκης

Εξεταστική Επιτροπή:

Καθηγητής Αντώνιος Δεληγιαννάκης (Επιβλέπων)

Καθηγητής Μίνως Γαροφαλάκης

Καθηγητής Βασίλειος Σαμολαδάς

Χανιά, Οκτώβριος 2017

Abstract

Nowadays data are being produced at an inconceivable rate, and more and more businesses and organizations are trying to use all this data to make critical decisions. Data processing and analysis for decision-making has been converted from luxury to need. Data management requires the use of complex algorithms that most people cannot implement. This is why programs have been created to provide a user-friendly environment for the design of these algorithmic processes. However, most of them are implemented to run these algorithms on a single computer. But what happens when the volume of data is too large and more computational power is required? In that case, it is necessary to use a cluster of computers.

For the purpose of this thesis, we chose RapidMiner, a design program for data management as a use-case, and expanded it to enable its processes to run on a cluster of computers managed by Apache Spark.

Περίληψη

Στις μέρες μας δεδομένα παράγονται συνεχώς σε ασύλληπτους ρυθμούς και όλο και περισσότερες επιχειρήσεις και οργανισμοί προσπαθούν να χρησιμοποιήσουν όλα αυτά τα δεδομένα για να πάρουν κρίσιμες αποφάσεις. Η επεξεργασία και ανάλυση δεδομένων για την λήψη αποφάσεων έχει μετατραπεί από πολυτέλεια σε ανάγκη. Η διαχείριση δεδομένων απαιτεί την χρήση πολύπλοκων αλγορίθμων τους οποίους οι περισσότεροι άνθρωποι δεν μπορούν να υλοποιήσουν. Για αυτό έχουν δημιουργηθεί προγράμματα τα οποία προσφέρουν ένα φιλικό προς τον χρήστη περιβάλλον για σχεδίαση αυτών περιπλοκών αλγοριθμικών διαδικασιών . Όμως τα περισσότερα από αυτά είναι υλοποιημένα για να εκτελούν αυτούς τους αλγόριθμους σε ένα υπολογιστή. Τι γίνεται όμως όταν ο όγκος των δεδομένων είναι πολύ μεγάλος και απαιτείται μεγαλύτερη υπολογιστική ισχύς. Σε αυτήν την περίπτωση είναι αναγκαία η χρήση ενός συμπλέγματος από υπολογιστές.

Σε αυτήν την εργασία, επιλέξαμε το RapidMiner, ένα σχεδιαστικό πρόγραμμα για διαχείριση δεδομένων, ως use-case και το επεκτείναμε ώστε να έχει την δυνατότητα να εκτελούνται οι διαδικασίες του σε ένα σύμπλεγμα υπολογιστών που διαχειρίζεται από το Apache Spark.

Ευχαριστίες

Αρχικά θα ήθελα να ευχαριστήσω τον κ. Αντώνιο Δεληγιαννάκη, επιβλέπων καθηγητή αυτής της διπλωματικής εργασίας για την άμεση και ουσιαστική βοήθεια και καθοδήγηση του κατά τη διάρκεια της δουλειάς μου. Επίσης είμαι ευγνώμων στα υπόλοιπα μέλη της εξεταστικής επιτροπής, καθηγητές κ. Βασίλειο Σαμολαδά και κ. Μίνω Γαροφαλάκη για τις υποδείξεις τους.
Επίσης θα ήθελα να ευχαριστήσω όλα τα μέλη της οικογένειάς μου για την ηθική υποστήριξη και τη αμέριστη συμπαράσταση σε όλη τη διαδρομή της ζωής μου.

Τέλος δε θα μπορούσα να μην ευχαριστήσω όλα τα κοντινά μου πρόσωπα και κυρίως τους φίλους μου για την ηθική και ψυχολογική υποστήριξη και κατανόηση που έδειξαν.

Περιεχόμενα

Λίστα Εικόνων

Λίστα Πίνακες

1 Εισαγωγή

1.1 Περιγραφή και Συνεισφορά Εργασίας

Ένα κυρίαρχο αντικείμενο της εποχής μας είναι αυτό των μεγάλων Δεδομένων (Big Data). Σήμερα, τα δεδομένα παράγονται συνεχώς σε ασύλληπτους ρυθμούς, από διαφορετικές πηγές (μέσα-κοινωνικής δικτύωσης, αισθητήρες κλπ). Δεν είναι λίγοι, που χρησιμοποιούν όλα αυτά τα δεδομένα για να πάρουν κάποια χρήσιμη πληροφορία, με σκοπό να προβλέψουν γεγονότα, ώστε να πάρουν κρίσιμες αποφάσεις, είτε επαγγελματικές, είτε για θέματα υγείας. Η επιστήμη των δεδομένων (data science) έχει μεταφερθεί σε όλα τα πεδία της επιστήμης και έχει γίνει απαραίτητη. Αποτελεί τη συνέχεια επιστημών, όπως η στατιστική, η ανάλυση προγνωστικών (predictive analytics), η μηχανική μάθηση (machine learning) και η εξόρυξη δεδομένων (data mining), οι οποίες με τη βοήθεια των τεχνολογικών προόδων των δύο τελευταίων δεκαετιών, σε συνδυασμό, εν μέρη και με την έκρηξη του διαδικτύου, οδήγησαν στη δημιουργία της επιστήμης των δεδομένων. Επειδή δεν είναι εύκολο να δοθεί ένας ακριβής ορισμός που να εκφράζει απόλυτα ακόμα και σήμερα το τι ακριβώς είναι αυτή η περιβόητη επιστήμη των δεδομένων, παρατίθενται παρακάτω ένας ορισμός:
"Η επιστήμη δεδομένων είναι ένας επιστημονικός κλάδος που σχετίζεται με τις διαδικασίες και τα συστήματα μέσω των οποίων εξάγεται η γνώση ή επεξεργάζονται τα ίδια τα στοιχεία που προέκυψαν από τα δεδομένα σε διάφορες μορφές (είτε δομημένα είτε αδόμητα) και αποτελεί συνέχεια επιστημονικών κλάδων όπως η ανάλυση δεδομένων η στατιστική, η εξόρυξη δεδομένων και η ανάλυση των προβλέψεων".

Με την συνεχόμενη αύξηση της ανάγκης για τη διαχείριση των δεδομένων, αρκετοί βρήκαν την ευκαιρία να δημιουργήσουν πλατφόρμες σχεδιασμού διαδικασιών, με σκοπό την ανάλυση δεδομένων, όπως weka, Knime, AdvancedMiner κτλ., σε περιβάλλον όπου ο χρήστης δεν χρειάζεται να έχει γνώσεις αλγορίθμων, παρά μόνο βασικές γνώσεις της επιστήμης δεδομένων.
Παράλληλα, παρατηρείται ότι ο αριθμός των χρηστών που στρέφεται σε αυτές όλο και μεγαλώνει για δύο βασικούς λόγους:

- Είναι απαραίτητο να γίνει η διαχείριση των δεδομένων τους για να είναι ανταγωνιστικοί.
- Οι αλγόριθμοι που χρησιμοποιούνται γίνονται όλο και πιο περίπλοκοι.

Γεγονός είναι ότι με τη γρήγορη αύξηση του όγκου των δεδομένων, όλα αυτά τα προγράμματα χρειάζεται να έχουν έναν τρόπο, ώστε οι ροές που σχεδιάζονται, να εκτελούνται σε ένα μεγάλο σύνολο από υπολογιστές, και επομένως να προσφέρουν την εκτέλεση διεργασιών σε άλλες πλατφόρμες όπως είναι το Hadoop, το Storm, το Spark κτλ.

Ένα από αυτά, που προσφέρει την σχεδίαση των ροών εργασίας δεδομένων σε ένα φιλικό για τον χρήστη περιβάλλον, όπου με βασικές γνώσης στην επιστήμη δεδομένων μπορεί να δημιουργήσει περίπλοκα μοντέλα (νευρολογικά δίκτυα, δέντρα απόφασης), είναι το RapidMiner Studio. Σε αυτή την εργασία θα εξετάσουμε έναν αποδοτικό τρόπο εκτέλεσης των ροών διεργασίας, που σχεδιάζονται στο RapidMiner, σε ένα σύμπλεγμα υπολογιστών που διαχειρίζεται το Apache Spark.

1.2 Επισκόπηση εργασίας

- ✔ Στο Κεφάλαιο 2 θα δούμε τα βασικά προγράμματα που χρησιμοποιήσαμε για την υλοποίηση αυτής της εργασίας, τα οποία είναι το RapidMiner και το Apache Spark. Θα καλύψουμε μια σχετική δουλειά που έχει γίνει για σύνδεση του RapidMiner με το Hadoop, το Radoop.
- ✔ Στο Κεφάλαιο 3 παρατίθενται οι στόχοι και τα προβλήματα που αντιμετωπίσαμε.
- ✔ Στα Κεφάλαια 4 και 5 θα παρουσιάσουμε όλη την υλοποίηση μας, με παραδείγματα και ένα πρότζεκτ το οποίο έπαιξε καθοριστικό ρόλο, ώστε να πραγματοποιηθεί αυτή η εργασία στον Spark Job Server.
- ✔ Τέλος. στο Κεφάλαιο 6 κλείνουμε προτείνοντας μελλοντικές βελτιώσεις.

2 Γνωστικό Υπόβαθρο

2.1 RapidMiner Studio

Το RapidMiner Studio είναι μια πλατφόρμα λογισμικού για την επιστήμη των δεδομένων και παρέχει ένα ολοκληρωμένο περιβάλλον για την προετοιμασία των δεδομένων, την εκμάθηση μηχανών, τη βαθιά εκμάθηση, την εξόρυξη κειμένου και την ανάλυση προγνωστικών. Συνδυάζει την τεχνολογία και την εφαρμοσιμότητα ώστε να εξυπηρετεί μια φιλική προς το χρήστη ενσωμάτωση των τελευταίων και καθιερωμένων τεχνικών data mining. Ο καθορισμός διαδικασιών ανάλυσης με το RapidMiner Studio πραγματοποιείται με drag&drop operators, ρύθμιση παραμέτρων και συνδυασμό operators.

Οι διεργασίες μπορούν να παραχθούν από ένα μεγάλο αριθμό operators και τελικά να εκπροσωπούνται από ένα λεγόμενο process graph (flow design). Η δομή της διεργασίας περιγράφεται εσωτερικά από την XML και αναπτύσσεται μέσω ενός γραφικού user interface. Επίσης, το RapidMiner Studio ελέγχει συνεχώς τη διεργασία που βρίσκεται σε εξέλιξη για τη διόρθωση της σύνταξης και κάνει προτάσεις αυτόματα σε περίπτωση προβλημάτων. Αυτό, καθίσταται εφικτό από το λεγόμενο ''meta-data transformation'', ο οποίος μετατρέπει τα βασικά meta data στο στάδιο του σχεδιασμού με τέτοιο τρόπο ώστε η μορφή του αποτελέσματος να μπορεί ήδη να προβλεφθεί και να εντοπιστούν λύσεις σε περίπτωση ακατάλληλων operator combinations (quick fixes). Επιπλέον, το RapidMiner Studio προσφέρει τη δυνατότητα του καθορισμού των breakpoints και επομένως τον έλεγχο σχεδόν κάθε ενδιάμεσου αποτελέσματος. Οι επιτυχείς operator combinations μπορούν να συγκεντρωθούν σε δομικά στοιχεία και ως εκ τούτου είναι και πάλι διαθέσιμοι σε μεταγενέστερες διαδικασίες.

Το RapidMiner παρέχει ένα GUI για να σχεδιάσει και να εκτελέσει αναλυτικές ροές εργασιών. Αυτές οι ροές εργασίας ονομάζονται "Διαδικασίες" και αποτελούνται από πολλαπλούς "τελεστές". Κάθε τελεστής εκτελεί μία μόνο εργασία μέσα στη διαδικασία και η έξοδος κάθε τελεστή αποτελεί την είσοδο του επόμενου.
Εναλλακτικά, η RapidMiner engine μπορεί να καλείται από άλλα προγράμματα ή να χρησιμοποιηθεί ως API και μπορούν να καλούνται μεμονωμένες λειτουργίες από τη γραμμή εντολών.

2.1.1 Επεκτάσεις του RapidMiner

Το RapidMiner Studio περιλαμβάνει συνολικά περισσότερες από 1500 λειτουργίες για όλες τις εργασίες τύπου professional data analysis, από data partitioning, μέχρι και market-based analysis, και διαθέτει όλα τα εργαλεία που χρειάζεται κανείς για να διαχειριστεί τα δεδομένα. Ακόμα, διαθέτει μεθόδους text mining, web mining και sentiment analysis, καθώς επίσης και η ανάλυση και η πρόβλεψη χρονοσειρών είναι διαθέσιμη, όπου τα περισσότερα από αυτά διατίθενται ως επεκτάσεις της πλατφόρμας. Παρακάτω αναφέρονται οι πιο σημαντικές και δημοφιλείς επεκτάσεις τού.

- RapidMiner Radoop**:** ένα code-free environment για το σχεδιασμό προηγμένων αναλυτικών διεργασιών, οι οποίες ωθούν τους υπολογισμούς στο Hadoop cluster.Μεταφράζει τις ροές εργασίας με τα predictive analytics, που σχεδιάζονται στο RapidMiner Studio στη γλώσσα του Hadoop. Επίσης επικοινωνεί με native Hive, MapReduce, Spark, Pig και Mahout, εξασφαλίζοντας ότι κάθε βήμα της διαδικασίας των predictive analytics είναι σωστά ενσωματωμένο και εκτελεσμένο σε όλες τις βασικές τεχνολογίες Big Data.

- Text Processing: προσθέτει όλους τους operators που είναι απαραίτητοι για text analysis. Μπορεί να φορτώσει texts από πολλές διαφορετικές πηγές δεδομένων, να τα μετατρέψει με ένα σύνολο διαφορετικών τεχνικών φιλτραρίσματος και τέλος να αναλύσει τα δεδομένα.

- Τα Text Extensions: υποστηρίζουν αρκετά text formats, όπως το plain text, το HTML ή το PDF, καθώς και άλλες πηγές δεδομένων. Παρέχει τυποποιημένα φίλτρα για tokenization, stemming, stopword filtering και n-gram generation για να παρέχει όλα όσα είναι απαραίτητα για την προετοιμασία και την ανάλυση texts.

- Weka Extension: Όλες οι μέθοδοι μοντελοποίησης και οι μέθοδοι αξιολόγησης χαρακτηριστικών από τη Weka machine learning library είναι διαθέσιμες στο RapidMiner. Με αυτήν την επέκταση δίνεται πρόσβαση σε περίπου 100 πρόσθετα προγράμματα μοντελοποίησης, συμπεριλαμβανομένων πρόσθετων δέντρων αποφάσεων, rule learners και regression estimators.

- Web Mining: Παρέχει πρόσβαση σε διάφορες πηγές διαδικτύου, όπως ιστοσελίδες, ροές RSS και web services. Η επέκταση παρέχει επίσης συγκεκριμένους operators για το χειρισμό και τη μετατροπή του περιεχομένου ιστοσελίδων για να προετοιμαστεί για περαιτέρω επεξεργασία.

- Keras Extension: Επιτρέπει τη μόχλευση του Keras απευθείας από το RapidMiner. Το Keras είναι API υψηλού επιπέδου νευρωνικού δικτύου, και υποστηρίζει δημοφιλείς deep learning libraries όπως το Tensorflow, το Microsoft Cognitive Toolkit (CNTK) και το Theano ως computation backends.

- Series Extension: Παρέχει operators για την επεξεργασία των χρονοσειρών. Περιλαμβάνει μια τεράστια ποικιλία από βήματα προεπεξεργασίας για δεδομένα χρονοσειρών, όπως windowing, moving average, exponential smoothing, Wavelet και Fourier Transformation, καθώς και διάφορες μεθόδους για την εξαγωγή χαρακτηριστικών από σειρές.

- Anomaly Detection: Περιλαμβάνει τους πιο γνωστούς αλγόριθμους ανίχνευσης ανωμαλιών χωρίς επίβλεψη, με την ανάθεση βαθμολογιών ατομικής ανωμαλίας σε data rows από σύνολα παραδειγμάτων.

2.1.2 Τελεστές, τύποι δεδομένων, συντομογραφίες και σχεδιασμός workflow

Οι τελεστές του Rapidminer χωρίζονται στις παρακάτω κατηγορίες(στις παρενθέσεις αναφέρονται πόσοι τελεστές υπάρχουν ανά κατηγορία):

- Data Access(13): Τελεστές για εγγραφή και διάβασμα αρχείων, καθώς και ανάκτηση και αποθήκευση δεδομένων.
- Blending(77) : Τελεστές για μετονομασίες, μετατροπές τύπου δεδομένων κ.τ.λ.
- Cleansing(26): Τελεστές για ομαλοποίηση, αφαίρεση ακραίων τιμών, μετασχηματισμό fourier.
- Modeling(123): Τελεστές για δημιουργία μοντέλων πρόβλεψης (logistic Regression, Neural Net, naïve Bayes), κατάτμηση (segmentation) (K-means, K-Medoids, Expectation Maximazation Clustering), συσχετίσεων, associations, similarites, Feature Weights, Optimization.
- Validation(27): Υπολογίζει την επίδοση των παραπάνω μοντέλων.
- Utility(74): Χρήσιμα εργαλεία για ευκολότερο σχεδιασμό πιο περίπλοκων work-flows (loops, branchs, subprocess κ.τ.λ).

Τα πιο συχνά input και output ports των τελεστών και οι τύποι των δεδομένων εμφανίζονται στον Πίνακα 2.1 ενώ οι τύποι των δεδομένων εμφανίζονται στον Πίνακα 2.2.

Συντομογραφία	Πλήρες Όνομα
exa	example set
mod	model
tran	tranning set
ori	original
fre	frequent item sets
per	performance
tes	test result set
fil	file
arc	archive file
thr	through(data table)

Πίνακας 2.1:Συντομογραφίες στο RapidMiner

Τύπος Τιμής	RapidMiner	Χρήση
Nominal	nominal	Μη αριθμητικές τιμές που χρησιμοποιούνται συνήθως για πεπερασμένες ποσότητες διαφορετικών χαρακτηριστικών
Numerical values	numeric	Αριθμητικές τιμές
Integers	integer	Όλοι οι ακέραιοι, θετικοί και αρνητικοί
Real numbers	real	Πραγματικοί, θετικοί και αρνητικοί
Text	text	Τυχαίο ελεύθερο κείμενο χωρίς δομή
2-value nominal	binominal	Ειδική περίπτωση nominal όπου μόνο 2 τιμές επιτρέπονται
Multi-value nominal	polynominal	Ειδική περίπτωση nominal όπου περισσότερες από 2 τιμές επιτρέπονται
Date time	date_time	Ημερομηνία – ώρα ημέρας
Date	date	Μόνο ημερομηνία
Time	time	Μόνο ώρα ημέρας

Πίνακας 2.2 Τύποι δεδομένων στο RapidMiner

Παράδειγμα workflow

Έστω ότι θέλουμε να απαντήσουμε στο ερώτημα : *"Ποιοι είχαν τις περισσότερες πιθανότητες να επιβιώσουν από το ατύχημα στον Τιτανικό;"*

Χρειαζόμαστε δεδομένα για το ποιοι έζησαν στον Τιτανικό, τα οποία υπάρχουν στο repository του RapidMiner.

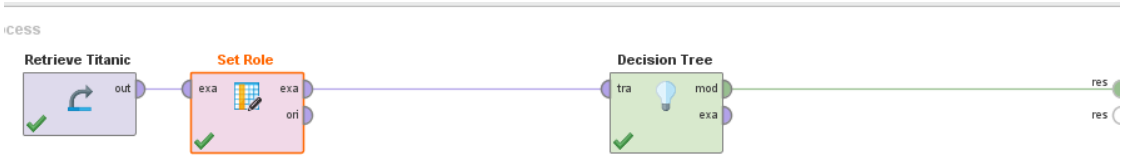
ExampleSet (Retrieve Titanic)

ExampleSet (1309 examples, 0 special attributes, 12 regular attributes)

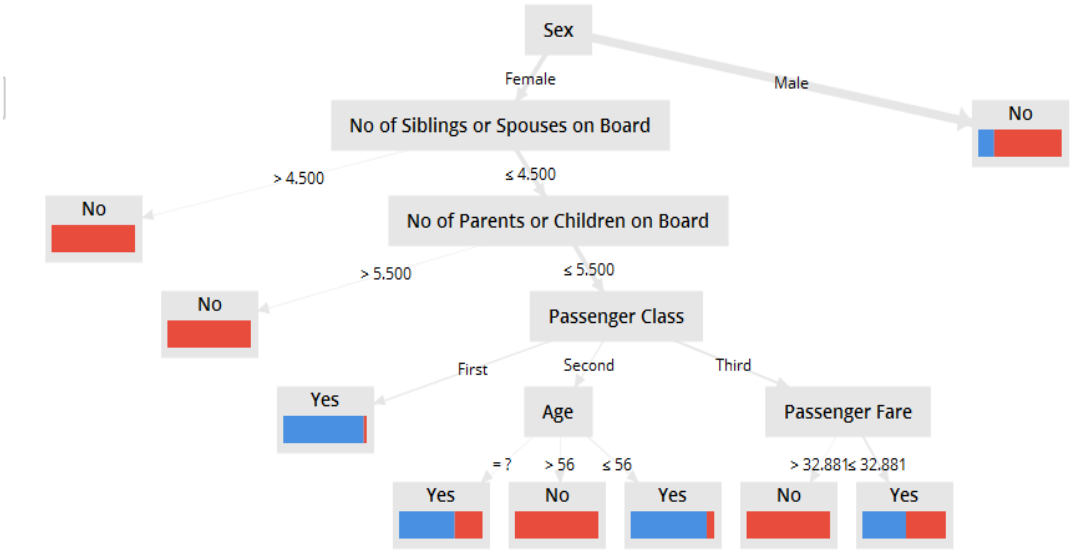
Row No.	Passenger ...	Name	Sex	Age	No of Sibling...	No of Parent...	Ticket Numb...	Passenger F...	Cabin	Port of Emb...	Life Boat	Survived
1	First	Allen, Miss. E...	Female	29	0	0	24160	211.338	B5	Southampton	2	Yes
2	First	Allison, Mast...	Male	0.917	1	2	113781	151.550	C22 C26	Southampton	11	Yes
3	First	Allison, Miss. ...	Female	2	1	2	113781	151.550	C22 C26	Southampton	?	No
4	First	Allison, Mr. H...	Male	30	1	2	113781	151.550	C22 C26	Southampton	?	No
5	First	Allison, Mrs. ...	Female	25	1	2	113781	151.550	C22 C26	Southampton	?	No
6	First	Anderson, Mr....	Male	48	0	0	19952	26.550	E12	Southampton	3	Yes
7	First	Andrews, Mis...	Female	63	1	0	13502	77.958	D7	Southampton	10	Yes
8	First	Andrews, Mr. ...	Male	39	0	0	112050	0	A36	Southampton	?	No
9	First	Appleton, Mrs...	Female	53	2	0	11769	51.479	C101	Southampton	D	Yes
10	First	Artagaveytia, ...	Male	71	0	0	PC 17609	49.504	?	Cherbourg	?	No
11	First	Astor, Col. Jo...	Male	47	1	0	PC 17757	227.525	C62 C64	Cherbourg	?	No
12	First	Astor, Mrs. Jo...	Female	18	1	0	PC 17757	227.525	C62 C64	Cherbourg	4	Yes
13	First	Aubart, Mme. ...	Female	24	0	0	PC 17477	69.300	B35	Cherbourg	9	Yes
14	First	Barber, Miss. ...	Female	26	0	0	19877	78.850	?	Southampton	6	Yes
15	First	Barkworth, Mr...	Male	80	0	0	27042	30	A23	Southampton	B	Yes
16	First	Baumann, Mr...	Male	?	0	0	PC 17318	25.925	?	Southampton	?	No
17	First	Baxter, Mr. Qu...	Male	24	0	1	PC 17558	247.521	B58 B60	Cherbourg	?	No
18	First	Baxter, Mrs. J...	Female	50	0	1	PC 17558	247.521	B58 B60	Cherbourg	6	Yes
19	First	Bazzani, Miss...	Female	32	0	0	11813	76.292	D15	Cherbourg	8	Yes
20	First	Beattie, Mr. T...	Male	36	0	0	13050	75.242	C6	Cherbourg	A	No
21	First	Beckwith, Mr. ...	Male	37	1	1	11751	52.554	D35	Southampton	5	Yes
22	First	Beckwith, Mrs...	Female	47	1	1	11751	52.554	D35	Southampton	5	Yes

Εικόνα 1: Παράδειγμα ενός RapidMiner Table

Παρατηρούμε ότι υπάρχει το attribute survived, που μας δίνει την πληροφορία που χρειαζόμαστε, ώστε να εκπαιδεύσουμε κάποιο μοντέλο απόφασης. Στην περίπτωση μας χρησιμοποιούμε ένα δέντρο απόφασης (Decision Tree). Με το set role επιλέγουμε το attribute που θέλουμε ως label και τρέχουμε το παρακάτω παράδειγμα, για να δούμε ποια ήταν τα κύρια χαρακτηριστικά που έπαιξαν το μεγαλύτερο ρόλο στο αν έζησε κάποιος.



Εικόνα 2: Παράδειγμα Workflow



Εικόνα 3: decision Tree

Από το σχήμα, παρατηρούμε ότι το κύριο χαρακτηριστικό που είχε καθοριστικό ρόλο στο αν κάποιος επιβίωσε από τον Τιτανικό, ήταν κατά κύριο λόγο το φύλο και ως δευτερεύον το Passenger Class. Γίνεται προφανές, λοιπόν, πόσο εύκολο είναι να υπολογίσει κανείς δέντρα απόφασης (ένα σχετικά περίπλοκο αλγοριθμικά μοντέλο) στο user interface του RapidMiner.

2.2 Apache Spark

Χωρίς αμφιβολία, η εμφάνιση του Hadoop και του οικοσυστήματος του παρείχε μια νέα αρχιτεκτονική για την επίλυση μεγάλων προβλημάτων δεδομένων. Παρέχει μια χαμηλού κόστους και κλιμακούμενη λύση, η οποία μπορεί να επεξεργαστεί terabyte δεδομένων σε λίγες ώρες, τα οποία νωρίτερα θα μπορούσαν να πάρουν ημέρες. Αυτή ήταν μόνο μία πλευρά του κέρματος όπου το Hadoop προοριζόταν για διαδικασίες παρτίδας, ενώ υπήρχαν άλλες περιπτώσεις επιχειρηματικής χρήσης, οι οποίες απαιτούσαν την παραγωγή επιχειρηματικών γνώσεων σε πραγματικό ή σχεδόν πραγματικό χρόνο (δευτερόλεπτα SLA). Αυτό ονομάστηκε ως "γρήγορα δεδομένα", όπου συνεπάγεται την ικανότητα λήψης αποφάσεων σε πραγματικό χρόνο και τη διευκόλυνση των τάσεων μεγέθους του χρόνου που πέρασε στις αποφάσεις για τις επιχειρήσεις. Έχουν αναπτυχθεί αρκετές ισχυρές, εύχρηστες πλατφόρμες ανοιχτού κώδικα για την επίλυση αυτών των επιχειρηματικών περιπτώσεων χρήσης σε πραγματικό χρόνο. Δύο από τις πιο αξιοσημείωτες είναι οι Apache Storm και Apache Spark, οι οποίες προσφέρουν δυνατότητες επεξεργασίας σε πραγματικό χρόνο σε ένα ευρύτερο φάσμα πιθανών χρηστών.

Και τα δύο προγράμματα είναι μέρος του Apache Software Foundation και ενώ τα δύο εργαλεία παρέχουν επικαλυπτόμενες δυνατότητες, καθένα από αυτά έχει ξεχωριστά χαρακτηριστικά και ρόλους για να παίξει. Το Apache Storm είναι ένα εξαιρετικό πλαίσιο για την αξιόπιστη επεξεργασία κατανεμημένων ροών. Εργάστηκε για την πλειοψηφία των περιπτώσεων χρήσης σε πραγματικό χρόνο, αλλά δεν κατάφερε να δώσει απαντήσεις στις ερωτήσεις όπως:

- *"Για να απαντηθούν ερωτήματα που απαιτούν επεξεργασία τόσο ροών δεδομένων, όσο και μαζική επεξεργασία ιστορικών δεδομένων, το Storm δεν επαρκεί. Χρειάζεται όμως απαραίτητα να αναπτυχθούν δύο διαφορετικά πλαίσια (Hadoop και Storm); "*
- *"Τι συμβαίνει με τη συγχώνευση ρευμάτων που προέρχονται από δύο διαφορετικές πηγές δεδομένων;"*
- *"Εκτός από την Java, μπορώ να χρησιμοποιήσω μια άλλη γλώσσα προγραμματισμού; "*
- *"Μπορούμε να ενσωματώσουμε ροές κοντά σε πραγματικό χρόνο με άλλα συστήματα όπως γραφήματα, SQL, κυψέλη και ούτω καθεξής;"*

Το Apache Spark ήταν η απάντηση σε όλες τις προηγούμενες ερωτήσεις. Δεν διατήρησε μόνο τα οφέλη του Hadoop και του Storm, αλλά ταυτόχρονα έδωσε ένα ενοποιημένο πλαίσιο όπου μπορεί ο χρήστης να γράψει τον κώδικα του σε μια ποικιλία γλωσσών προγραμματισμού όπως η Python, η Java ή η Scala και να επαναχρησιμοποιήσει το ίδιο κομμάτι κώδικα στις περιπτώσεις χρήσης ροής και παρτίδας. Επιπλέον, το Spark είναι συμβατό με σχεδόν οποιοδήποτε σύστημα αποθήκευσης που υποστηρίζεται από το Hadoop, και μπορεί να διαβάζει δεδομένα ή να γράφει δεδομένα από αυτά. Με αυτό τον τρόπο είναι συμβατό με τις μορφές που χρησιμοποιούνται τακτικά για την αποθήκευση δεδομένων για το Hadoop, όπως το Avro και το CSV, καθώς επίσης και με βάσεις δεδομένων NoSQL όπως η HBase και η Cassandra. Υποστηρίζει επίσης πολλές διαφορετικές πηγές δεδομένων

όπως Hive tables, Parquet και JSON. Επίσης, το Spark SQL υποστηρίζει μία διεπαφή SQL και επιπλέον, επιτρέπει στους προγραμματιστές να συνδυάζουν ερωτήματα SQL με τους προγραμματισμένους χειρισμούς δεδομένων που υποστηρίζονται από τα Resilient Distributed Datasets (RDDs), όλα μέσα σε μία ενιαία εφαρμογή, ενοποιώντας έτσι SQL με σύνθετα analytics.

Το Spark είναι ένα καινοτόμο framework συμπλέγματος υπολογιστών που είναι σε θέση να εκτελεί προγράμματα μέχρι και 40 φορές ταχύτερα από το Hadoop, ενώ παράλληλα διατηρεί τη γραμμική κλιμάκωση και την ανοχή σφάλματος του MapReduce. Επιπλέον, επεκτείνεται με πολλούς σημαντικούς τρόπους.

Αξίζει ακόμα να αναφερθεί πως το Spark βελτιώνει τα δεδομένα των χρηστών του με επεξεργασία μέσα στην μνήμη. Τα RDDs παρέχουν στους προγραμματιστές τη δυνατότητα να επεξεργαστούν οποιοδήποτε σημείο του pipeline στη μνήμη του συμπλέγματος, με αποτέλεσμα οποιεσδήποτε μελλοντικές ενέργειες που πρέπει να χρησιμοποιούν τα ίδια δεδομένα, να μη χρειαστεί να επαναπροσδιορίσουν τιμές ή να τα ανακτήσουν από το δίσκο. Με αυτό τον τρόπο, το Spark χειρίζεται μια σειρά σεναρίων που οι κατανεμημένες μηχανές επεξεργασίας δεν μπορούσαν να αντιμετωπίσουν στο παρελθόν.
Στον πυρήνα του Spark έχουν υλοποιηθεί διάφορες βιβλιοθήκες και επεκτάσεις, οι οποίες απεικονίζονται στην Εικόνα 4 και αναγράφονται παρακάτω:

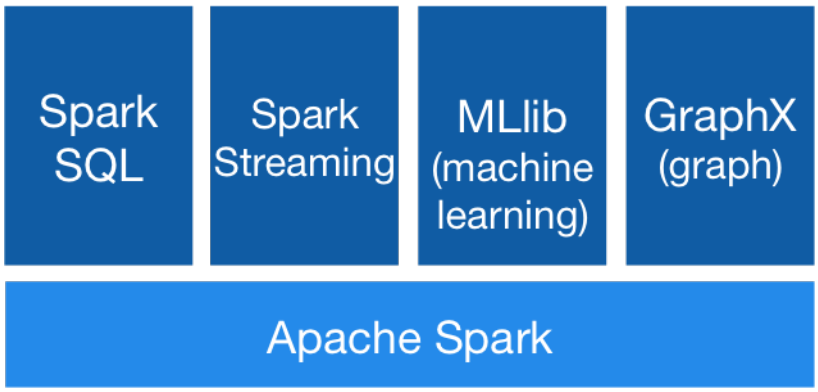
- Spark GraphX: Για την ανάπτυξη γραφημάτων Spark DataFrames.
- SQL: Εκτέλεση ερωτημάτων SQL.
- Spark MLlib: Εκτέλεση αλγορίθμων μηχανικής μάθησης για συστάσεις, ομαδοποίηση και ταξινομήσεις.
- Spark Streaming: Για χειρισμό δεδομένων ροής σε σχεδόν πραγματικό χρόνο, η οποία έχει τη δυνατότητα να αποκτά πληροφορίες συνεχώς από διάφορα frameworks, όπως το Flume και Kafka.

Ένα από τα αξιοσημείωτα χαρακτηριστικά του Apache Spark ήταν η διαλειτουργικότητα όλων αυτών των βιβλιοθηκών και επεκτάσεων. Για παράδειγμα, τα δεδομένα που λαμβάνονται από ρεύματα σχεδόν σε πραγματικό χρόνο μπορούν να μετατραπούν σε γραφήματα ή να αναλυθούν χρησιμοποιώντας SQL, ή μπορούμε να εκτελέσουμε αλγόριθμους μηχανικής μάθησης για την παροχή συστάσεων, ομαδοποίησης ή ταξινομήσεων.

Όλες αυτές οι μέθοδοι έχουν σχεδιαστεί για να κλιμακώνονται σε ένα σύμπλεγμα υπολογιστών, εφαρμόζοντας πολλές κοινές μεθόδους μηχανικής μάθησης και στατιστικούς αλγόριθμους για την απλοποίηση μεγάλων pipeline μηχανικής μάθησης. Για παράδειγμα, το GraphX είναι μια βιβλιοθήκη για τον χειρισμό γραφικών παραστάσεων και την εκτέλεση παράλληλων υπολογισμών. Όπως το Spark Streaming και το Spark SQL, το GraphX επεκτείνει το Spark RDD API, επιτρέποντας στο χρήστη να δημιουργεί ένα κατευθυνόμενο γράφημα με αυθαίρετες ιδιότητες, προσαρτημένες σε κάθε κορυφή και άκρη. Το GraphX

παρέχει, επίσης, διάφορους τελεστές για το χειρισμό γραφικών παραστάσεων και μια βιβλιοθήκη κοινών αλγόριθμων γραφημάτων.

Το Spark θα μπορούσε να θεωρηθεί μια πλατφόρμα κατανεμημένων υπολογιστών γενικής χρήσης,που υποστηρίζει τόσο παρτίδες όσο και επεξεργασία δεδομένων σε σχεδόν πραγματικό χρόνο.



Εικόνα 4: Apache Spark

2.3 Data Streaming Processing

Όταν αναφερόμαστε σε επεξεργασία δεδομένων σε πραγματικό χρόνο θεωρούμε πως τα δεδομένα μας συνεχώς μεταβάλλονται, τα δεδομένα αυτά μπορεί για παράδειγμα να αφορούν πληροφορίες για τον έλεγχο της εναέριας κυκλοφορίας, travel booking systems κ.τ.λ., και η επεξεργασία γίνεται με τέτοιο ρυθμό ώστε να ελέγχεται η πηγή των δεδομένων. Ο χρόνος απόκρισης για την επεξεργασία δεδομένων σε πραγματικό χρόνο είναι άμεσος και αναμένεται να είναι της τάξης των milliseconds (μερικές φορές ακόμα και microseconds).

Ένα σύστημα ορίζεται ως σύστημα επεξεργασίας δεδομένων σε πραγματικό χρόνο μόνο εάν παράγει τα λογικά και σωστά αποτελέσματα μέσα στο δεδομένο χρονικό περιθώριο (milliseconds). Τα συστήματα πραγματικού χρόνου συχνά χρησιμοποιούν τον όρο ‘χρόνος καθυστέρησης’, ο οποίος αναφέρεται στη χρονική διαφορά ανάμεσα στη στιγμή που έφθασαν τα δεδομένα και τη στιγμή που δημιουργήθηκε η απάντηση. Ακολουθούν μερικά παραδείγματα συστημάτων πραγματικού χρόνου τα οποία δέχονται δεδομένα σε πραγματικό χρόνο, τα επεξεργάζονται και επιστρέφουν τα αποτελέσματα :

- ATM τραπεζών: Δέχονται δεδομένα ως είσοδο από το χρήστη και άμεσα ενημερώνουν το κεντρικό σύστημα για τις συναλλαγές (αναλήψεις ή άλλες αιτήσεις).
- Real-time monitoring: Ανάλυση δεδομένων τα οποία μπορεί να προέρχονται από διάφορες πηγές, όπως αισθητήρες, ζωντανές μεταδόσεις κ.τ.λ.
- POS συστήματα: Ενημέρωση αποθέματος, παροχή ιστορικού αποθέματος και

πωλήσεις συγκεκριμένου στοιχείου επιτρέποντας σε έναν οργανισμό να εκτελεί πληρωμές σε πραγματικό χρόνο.

- Assembly lines: Επεξεργάζονται δεδομένα σε πραγματικό χρόνο, για την ελαχιστοποίηση του χρόνου απόκρισης και σφαλμάτων. Τα σφάλματα ανιχνεύονται αμέσως και αντιμετωπίζονται χωρίς καμία καθυστέρηση, που σε διαφορετική περίπτωση θα είχε οδηγήσει στην παραγωγή ελαττωματικών η χαμηλής ποιότητας αποτελεσμάτων.

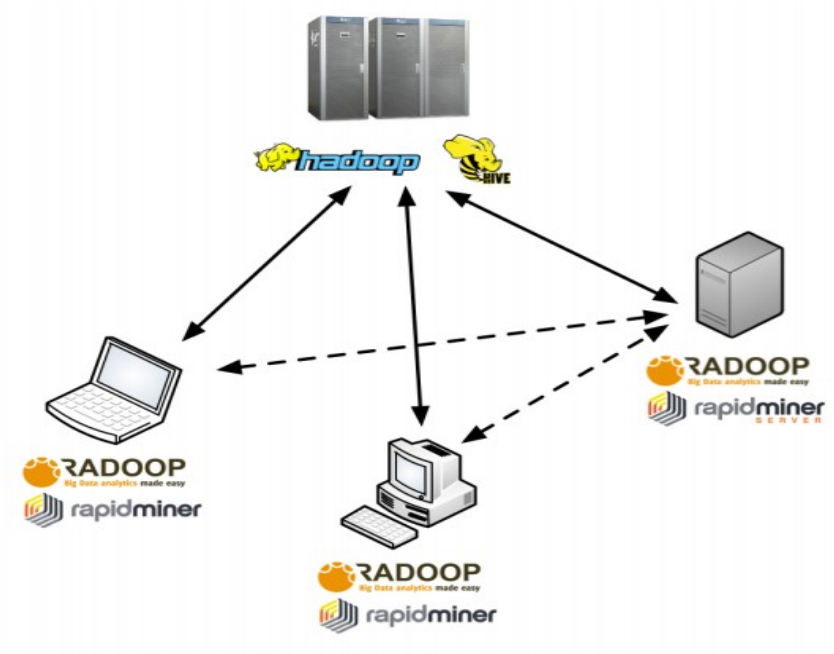
Λίγα λόγια όσον αφορά τις προκλήσεις των συστημάτων πραγματικού χρόνου:

- Απόκριση του συστήματος: η απαίτηση των χρηστών συστημάτων πραγματικού χρόνου είναι, η επεξεργασία των δεδομένων να γίνεται σε microsecond ή milisecond από τη στιγμή της άφιξής τους έτσι ώστε να μην εισάγεται καθόλου καθυστέρηση στη ροή των δεδομένων.
- Αξιοπιστία: Τα σφάλματα είναι αναπόφευκτα, αλλά τα συστήματα πραγματικού χρόνου δεν έχουν την πολυτέλεια να χάσουν ούτε ένα event.
- Δυνατότητα κλιμάκωσης: Είναι ανάγκη να έχουμε μια αρχιτεκτονική με δυνατότητα κλιμάκωσης έτσι ώστε να μπορεί να ανταποκριθεί στις απαιτήσεις των αυξανόμενων δεδομένων με την προσθήκη περισσότερων υπολογιστικών πόρων χωρίς να είναι αναγκαία η εκ νέου σχεδίαση ολόκληρου του συστήματος.
- In memory: Τα συστήματα σε πραγματικό χρόνο δεν έχουν τη δυνατότητα να διαβάζουν από δίσκους, επομένως η επεξεργασία δεδομένων πρέπει να γίνεται μέσα στην ίδια τη μνήμη. Επομένως, τα συστήματα πρέπει να εξασφαλίζουν επαρκή μνήμη για την αποθήκευση των δεδομένων εισόδου στη μνήμη του συστήματος.

Η αναγκαιότητα είναι η μητέρα της εφεύρεσης, και αυτό συνέβη. Το Apache Spark αναπτύχθηκε ως ένα πλαίσιο επόμενης γενιάς και μια ενιαία λύση για όλες τις περιπτώσεις χρήσης, ανεξάρτητα από το αν έπρεπε να επεξεργαστούν σε παρτίδες ή σε πραγματικό χρόνο.

2.4 RapidMiner Radoop extension

Το RapidMiner Radoop παρέχει μια εύχρηστη γραφική διεπαφή για την ανάλυση δεδομένων σε ένα Hadoop cluster. Απαραίτητη προϋπόθεση είναι ο Hadoop cluster να τρέχει έναν Hive server.
Το RapidMiner Radoop είναι ένα λογισμικό πελάτη(client software) που συνδέεται με ένα Hadoop cluster και εκτελεί διαδικασίες που δημιουργήθηκαν σε ένα διαισθητικό γραφικό περιβάλλον εργασίας .Εκτελεί, προγραμματίζει, διαχειρίζεται διαδικασίες που δημιουργούνται με τον client και παρέχει πρόσθετες λειτουργίες συνεργασίας. Παρέχει λοιπόν στο RapidMiner Studio έναν εύκολο τρόπο να



Εικόνα 5: Architecture diagram of the RapidMiner Radoop client

σχεδιάζονται διεργασίες και να τρέχουν σε ένα σύμπλεγμα υπολογιστών που διαχειρίζεται το Hadoop. Επιπλέον δίνει την δυνατότητα οι διεργασίες, που σχεδιάζονται να μεταφραστούν και εκτελέσουν στο Hive ή στο Spark MLLib. Το Spark MLLib όπως αναφέραμε και προηγουμένως είναι μια επέκταση του Spark και δεν συμπεριλαμβάνει την διαχείριση ροών δεδομένων που είναι ο κύριος στόχος μας. Το μεγάλο πλεονέκτημα του Radoop είναι ότι προσφέρει όλη την λειτουργικότητα σχεδίασης διεργασιών του RapidMiner Studio, αλλά η εκτέλεση γίνεται κατανεμημένα σε πολλούς υπολογιστές.

3 Στόχοι και Προβλήματα που Αντιμετωπίσαμε

3.1 Στόχοι

Αυτή η εργασία προσπαθεί να δώσει στο RapidMiner Studio τη δυνατότητα να χρησιμοποιείται για μηχανή εκτέλεσης των διεργασιών που σχεδιάζονται, το Apache Spark με όλες τις δυνατότητες που προσφέρει, όπως SQL, Streaming, complex analytics. Κύρια λειτουργικότητα που επιθυμούμε, είναι να μπορούμε να σχεδιάζουμε streaming jobs που εκτελούνται κατανεμημένα σε συμπλέγματα υπολογιστών που διαχειρίζονται από το Spark. Επίσης, θέλουμε αυτό να μπορεί να συμβεί χωρίς να χρειάζεται να έχουν στηθεί άλλα προγράμματα, δηλαδή η υλοποίηση μας να μπορεί να επικοινωνεί με το Spark, όπου και να τρέχει αυτό, σε Hadoop ή mesos ή ακόμα και standalone.Τέλος, να ελαχιστοποιήσουμε όσο το δυνατόν περισσότερο την επικοινωνία RapidMiner – Spark και την επιπλέον πληροφορία που μεταφέρεται, καθώς και τον επιπρόσθετο χώρο που χρειάζεται.

3.2 Προβλήματα με το Radoop

Η πρώτη μας προσέγγιση ήταν να επεκτείνουμε το Radoop, ώστε να μπορούν να σχεδιάζονται streaming διεργασίες, δεδομένου ότι μπορεί να χρησιμοποιεί το Spark ως μηχανή εκτέλεσης διεργασιών και πιθανότατα να μπορεί να το χρησιμοποιεί και για εκτέλεση streaming διαδικασιών. Αυτή η προσπάθεια μας έχει ένα μεγάλο εμπόδιο. Οι τελεστές του RapidMiner και αντίστοιχα και του Radoop, δεν έχουν σχεδιαστεί για να εκτελούνται με pipeline μορφή διαδικασιών και τα δεδομένα δεν ρέουν σε όλο το σχεδιαστικό πλάνο. Ξεκινάει ο πρώτος τελεστής, τελειώνει την δουλειά του και μετά ξεκινάει ο επόμενος. Ο κύριος λόγος που το RapidMiner χρησιμοποιεί αυτήν την προσέγγιση, είναι ότι θέλει να επιτρέπει breakpoints ώστε να μπορεί ο χρήστης να σταματάει τη διαδικασία μέχρι ένα σημείο, και να αναλύει τα αποτελέσματα, το οποίο μάλλον είναι “απίθανο” να υλοποιηθεί αν έχουμε συνεχόμενη ροή δεδομένων σε όλους τους τελεστές.Επιπλέον για να γίνει η σύνδεση του RapidMiner με το Hadoop, είναι απαραίτητο να υπάρχει ένας Hive Server, για την αποθήκευση των μεταδεδομένων του workflow, το οποίο δεν είναι πάντα επιθυμητό.
Τέλος, το Radoop δημιουργεί προσωρινά αρχεία στο Hadoop με την εκτέλεση κάθε τελεστή και επίσης απαιτεί επικοινωνία στην αρχή και στο τέλος κάθε τελεστή.

3.3 Προβλήματα επικοινωνίας με Spark Cluster

Το Spark δεν μας προσφέρει κάποιον εύκολο τρόπο να κάνουμε submit jobs στον cluster Jobs.Όταν αναζητούσαμε μια μέθοδο για την επικοινωνία της εφαρμογής μας, ανακαλύψαμε ότι υπήρχαν αρκετές επιλογές για να επικοινωνήσουμε με ένα Spark cluster, αλλά καμία από αυτές δεν παρείχε την ευελιξία που χρειαζόμασταν συνδυασμένη με ένα εύχρηστο API.

- Το Spark Submit ήταν η κύρια υποστηριζόμενη προσέγγιση για Spark job submission. Ωστόσο, το να κάνεις fork μια διαδικασία για την εκτέλεση του shell script αποδείχθηκε τόσο δυσκίνητο όσο και αργό. Τα αποτελέσματα από το job execution χρειάζεται να γραφτούν σε μια εξωτερική βάση δεδομένων, στην οποία θα έχει τότε απευθείας πρόσβαση η εφαρμογή μας.

- Το JDBC παρέχει πιο άμεση πρόσβαση σε ένα cluster σε σύγκριση με το Spark Submit, αλλά περιορίστηκε στη Spark SQL και δεν χρησιμοποιούσε εύκολα άλλα Spark components όπως το Spark Streaming.

- Το Spark Shell προσφέρει την μεγαλύτερη ευελιξία μέσω της υποστήριξης της εκτέλεσης code snippets, επιτρέποντας μας με αυτόν τον τρόπο να ελέγχουμε με ακρίβεια και δυναμικά τις διεργασίες που υποβάλλονται σε ένα Spark cluster. Δυστυχώς, το shell δεν ήταν μια αναλώσιμη υπηρεσία (consumable service) που θα μπορούσαμε να χρησιμοποιήσουμε στις εφαρμογές μας.

4 Η Προσέγγιση μας

Λαμβάνοντας υπόψιν όλα τα προαναφερόμενα, καταλήξαμε στο να υιοθετήσουμε μια τελείως διαφορετική προσέγγιση στο πρόβλημα. Χρησιμοποιούμε το RapidMiner για τον σχεδιασμό κώδικα και όχι για την άμεση εκτέλεση τελεστών. Αυτό μας δίνει τη δυνατότητα να σχεδιάσουμε κάθε μορφής ροή διαδικασιών (work flow). Με σωστή υλοποίηση των τελεστών μας, μπορούμε να σχεδιάσουμε κάθε τύπου διεργασία, αφού σχεδιάζουμε κώδικα. Ωστόσο, δεν μας επιτρέπει τη χρήση breakpoints άρα και το να έχουμε πρόσβαση σε ενδιάμεσα αποτελέσματα.
Ένα μεγάλο πλεονέκτημα αυτής της υλοποίησης, είναι ότι ο σχεδιασμός γίνεται σχεδόν ακαριαία, καθώς απλά σχεδιάζουμε κώδικα και ο χρόνος εκτέλεσης είναι τελείως ανεξάρτητος από τον όγκο των δεδομένων. Συνεπώς, ο σχεδιασμός μπορεί να γίνει σε υπολογιστή με την ελάχιστη υπολογιστική ισχύ και χώρο μνήμης. Επίσης, η υλοποίηση μας μπορεί να κάνει compile (μετάφραση) σε πραγματικό χρόνο java κώδικα, το οποίο μας δίνει τη δυνατότητα όχι μόνο να χρησιμοποιούμε pre-compiled κώδικα, αλλά δυναμικά να αλλάζουμε συναρτήσεις και τον τρόπο εκτέλεσης των διαδικασιών.
Τέλος, προσφέρει το σχεδιασμό δύο ειδών σχεδίασης εργασιών, τις σύγχρονες εργασίες και τις ασύγχρονες.

1. Σύγχρονες εργασίες: ξεκινάμε μια εργασία στο σύμπλεγμα υπολογιστών και περιμένουμε να πάρουμε απάντηση με τα δεδομένα. Γίνεται για εργασίες που έχουν μικρό χρόνο εκτέλεσης ή γνωρίζουμε ότι θα τελειώσουν.
2. Ασύγχρονες εργασίες: σχεδιάζουμε μια εργασία και τελειώνει μόλις την στείλουμε στο σύμπλεγμα. Είναι απαραίτητες για τον σχεδιασμό streaming εργασιών, οι οποίες δεν ξέρουμε πότε θα τελειώσουν και μπορεί να τρέχουν για μέρες. Αυτό απαιτεί την υλοποίηση διαφορετικών εργασιών για να πάρουμε τα αποτελέσματα.

4.1 Spark Job server

Δεδομένου ότι καμία από τις διαθέσιμες επιλογές επικοινωνίας με το Apache Spark δεν ανταποκρινόταν στις ανάγκες μας για μια διαδραστική εφαρμογή, αποφασίσαμε να χρησιμοποιήσουμε ένα άλλο εργαλείο :τον Spark Job Server. Ο Spark Job Server χρησιμοποιείται ως μεσάζων (middleware) μεταξύ της εφαρμογής μας και του cluster Spark.
Επειδή η εφαρμογή μας επικεντρώθηκε στην αλληλεπίδραση, χρειαζόμασταν ένα τρόπο να διαχειριζόμαστε τις εφαρμογές, τις εργασίες και τα context του Spark. Ο Spark Job Server με το REST api που μας παρέχει μια πιο ευέλικτη και δυναμική επικοινωνία με το Spark.

4.1.1 Spark Job Server REST API

Παρακάτω παραθέτουμε τις κύριες εντολές του REST API του Spark Job Server, που κάνουν δυνατή την δυναμική διαχείριση εργασιών και εφαρμογών του Spark με χρήση οποιαδήποτε προγραμματιστικής πλατφόρμας.

GET /jars - lists all the jars and the last upload timestamp
POST /jars/<appName> - uploads a new jar under <appName>

GET /contexts - lists all current contexts
GET /contexts/<name> - gets info about a context, such as the spark UI url
POST /contexts/<name> - creates a new context
DELETE /contexts/<name> - stops a context and all jobs running in it
PUT /contexts?reset=reboot - shuts down all contexts

Use ?sync=false to execute asynchronously.

GET /jobs - Lists the last N jobs
POST /jobs - Starts a new job, use ?sync=true to wait for results
GET /jobs/<jobId> - Gets the result or status of a specific job
DELETE /jobs/<jobId> - Kills the specified job
GET /jobs/<jobId>/config - Gets the job configuration

4.2 Επικοινωνία με τον Job Server

Το REST api που μας προσφέρει ο Spark Job Server, έκανε εφικτή την επικοινωνία του RapidMiner με τον Spark Cluster. Το RapidMiner δημιουργεί έναν HttpClient για κάθε εργασία που θέλει να τρέξει στον Spark Cluster. Ο Client στέλνει όλη την πληροφορία που χρειάζεται στον job Server για να εκτελεστεί η εργασία, και είτε περιμένει απάντηση με τα αποτελέσματα (σύγχρονη επικοινωνία), είτε απλά στέλνει το job για να εκτελεστεί, χωρίς να περιμένει για τα αποτελέσματα (ασύγχρονη επικοινωνία). Αφού τελειώσει η παραπάνω διαδικασία, ο HttpClient κλείνει και όταν σχεδιαστεί η επόμενη διεργασία δημιουργείται ξανά.
Η διαδικασία που αναφέραμε, απαιτεί συνολικά τρία μηνύματα προς τον Job Server και αντίστοιχα τρεις απαντήσεις από τον Server στον client που αναφέρονται παρακάτω.

1. SparkJobServerClientFactory.createSparkJobServerClient("HostName:port");
   Ξεκινάει μια καινούρια σύνδεση με τον JobServer, ο οποίος έχει **Internet Protocol address** (IP adress) = HostName και ακούει στην θύρα = port.
2. client.uploadSparkJobJar(**new** File("jarName.jar"), "appName");
   Ο client στέλνει στον jobServer το .jar file που έχει δημιουργηθεί και του δίνει το όνομα appName.

3. client.startJob("input.string", params);
   Ξεκινάει ένα καινούριο job στον Server δίνοντας του μια συμβολοσειρά για την είσοδο του, άν αυτό απαιτείται, και μια λίστα από μερικές απαραίτητες και μερικές προαιρετικές παραμέτρους, που θα δούμε στους παρακάτω πίνακες 5.1 και 5.2.

Όνομα Απαραίτητων Παραμέτρων	Χρήση
PARAM_APP_NAME	*Όνομα της εφαρμογής που θέλουμε να τρέξουμε*
PARAM_CLASS_PATH	*Η κλάση η οποία περιέχει την μέθοδο Run*
PARAM_CONTEXT	*το όνομα του context που θέλουμε η εφαρμογή μας να τρέξει (Streaming Context, JavaContext)*

Πίνακας 4.1: Απαραίτητοι Παράμετροι

Όνομα Προαιρετικών Παραμέτρων	Χρήση
PARAM_SYNC	*Αληθής αν θέλουμε σύγχρονη κλήση, ψευδής αν θέλουμε ασύγχρονη. Αν παραληφθεί έχουμε ασύγχρονη κλήση*
PARAM_MEM_PER_NODE	*αριθμός των κόμβων που θέλουμε να χρησιμοποιήσουμε*
PARAM_NUM_CPU_CORES	*αριθμός των κόμβων πυρήνων*
PARAM_SPARK_EXECUTOR_MEMORY	*μέγεθος μνήμης που απαιτείται*

Πίνακας 4.2:Προαρετικοι Παράμετροι

Η διαχείριση όλων αυτών των παραμέτρων από το εικονικό περιβάλλον του RapidMiner δεν είναι ακόμα εφικτή, αλλά με μικρές αλλαγές στο σχεδιασμό των τελεστών μπορεί να γίνει σε μικρό χρονικό διάστημα. Παρακάτω θα αναφερθούν οι προκαθορισμένες τιμές αυτών τον παραμέτρων ώστε να γίνει εφικτή η επικοινωνία με τον Job Server.

4.3 Οι Τελεστές μας

Για την πραγματοποίηση της προσέγγισής μας, υλοποιήσαμε δύο βασικούς τύπους τελεστών:

1. **Απλοί Τελεστές** υπεύθυνοι για την σχεδίαση των διεργασιών που θέλουμε να εκτελέσουμε. Μπορούν να έχουν είσοδο Table με ένα attribute (source) τύπου nominal και να προσθέσουν στην στήλη την διεργασία που θέλουν να εκτελεστεί ή να μην έχουν είσοδο και να δημιουργήσουν το Table που αναφέραμε (attribute(source)) με ένα tuple. Επίσης, έχουν υποχρεωτικά μία έξοδο το Table που πήραν ως είσοδο ή αυτό που δημιούργησαν.
2. **Τελεστές επικοινωνίας** για την εκτέλεση των διεργασιών που σχεδιάστηκαν, παίρνουν υποχρεωτικά μια είσοδο ένα Table με όλες τις διεργασίες που χρειάζεται να εκτελεστούν και πραγματοποιούν τα παρακάτω:

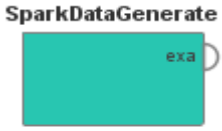
    - Δημιουργούν έναν κώδικα σε JAVA (αρχείο .java).
    - Κάνουν compile τον κώδικα (ένα αρχείο .class).
    - Βρίσκουν όλες τις απαραίτητες κλάσεις και δημιουργούν ένα .jar με όλα όσα χρειάζονται για την εκτέλεση της συνολικής διεργασίας που σχεδιάστηκε.
    - Δημιουργούν ένα HttpClient και συνδέονται στον jobServer.
    - Στέλνουν το .jar που δημιουργήθηκε στον JobServer.
    - Ξεκινάνε τη διεργασία που έστειλαν σύγχρονα ή ασύγχρονα.

    Κάθε διεργασία είναι απαραίτητο να έχει τουλάχιστον έναν τελεστή τύπου *απλός τελεστής* και έναν τελεστή επικοινωνίας. Ο αριθμός των διεργασιών που δημιουργούνται είναι ίσος με τον αριθμό των τελεστών επικοινωνίας που χρησιμοποιούνται στη σχεδίαση.

Οι τελεστές σχεδίασης χωρίζονται ανάλογα με το αν οι διεργασίες τους απαιτούν Spark Context ή Spark Streaming Context σε Spark τελεστές ή Spark Streaming τελεστές.

4.3.1 Spark Τελεστές

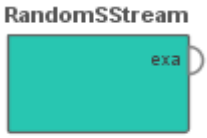
1. Δημιουργεί ένα καινούριο RDD με τυχαίες τιμές



2. Προσθέτει τυχαία δεδομένα στο RDD

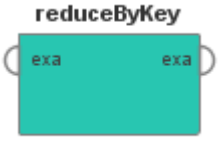
4.3.2 Spark Streaming Τελεστές

1. Δημιουργεί ένα καινούριο Streaming RDD με τυχαίες τιμές



2. Κάνει map όλους τους αριθμούς σε (αριθμός %100 , 1)

3. Προσθέτει όλα τα ίδια κλειδιά και έχει έξοδο (κλειδί, άθροισμα εμφάνισης κάθε κλειδιού)



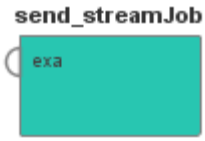
4.3.3 Τελεστές επικοινωνίας (Περίπλοκοι τελεστές)

Δέχονται ως είσοδο το Table που αναφέραμε παραπάνω κάνουν τις εργασίες που προαναφέραμε και θέτουν παραμέτρους πριν στείλουν την εργασίες στον JobServer.

1. Υπεύθυνος τελεστής για διαδικασίες που εκτελούνται με SparkContext
   - PARAM_APP_NAME = tjar
   - PARAM_CLASS_PATH = SparkJob
   - PARAM_CONTEXT = jcontext
   - PARAM_SYNC = true

   Παρατηρούμε ότι έχει την παράμετρο συγχρονισμού αληθής άρα περιμένει απάντηση με τα δεδομένα από τον JobServer.Επίσης η έξοδος του είναι ένα Table με ένα attribute(“result”) τύπου Numerical και είναι συμβατή με όλους τους υπόλοιπους τελεστές του RapidMiner.

2. Υπεύθυνος τελεστής για διεργασίες που εκτελούνται με StreamingContext
   - PARAM_APP_NAME = StreamingContext
   - *PARAM_CLASS_PATH = JavaStreamingjob*
   - PARAM_CONTEXT = StreamingContext



   Ο τελεστής αυτός αφού διαχειρίζεται το StreamingContext είναι απαραίτητο να ξεκινάει ένα ασύγχρονο job γιατί τα streaming jobs δεν σταματάν να τρέχουν συνεχόμενα μέχρι να διαγραφεί το context τους.

4.4 Από ροές εργασίας σε εργασίες του Spark

Με τη χρήση των τελεστών σχεδίασης δημιουργούμε έναν πίνακα από όλες τις μεθόδους που θέλουμε να εκτελεστούν στο Spark. Ο πίνακας αυτός είναι η είσοδος στους τελεστές μετάδοσης, ο οποίος με τη χρήση του αλγορίθμου που θα περιγράψουμε παρακάτω, μετατρέπει τον πίνακα σε γραμμές JAVA κώδικα. Έπειτα, ο κώδικας που δημιουργήθηκε προστίθεται στον default-κώδικα που έχουμε γράψει (ουσιαστικά προτίθεται στον κώδικα που είναι κοινός για κάθε είδους ίδιου τύπου διεργασία) και ο κώδικας είναι έτοιμος για να γίνει compile.

Αλγόριθμος δημιουργίας Κώδικα.

```
String CreateString(String classes[] ){
        String str
        for ( int i = 0 ; i < classes.length ; i++)
        {
                str.append(classes[classes.length-1];
                str.append(".call(ssc,");
```

```
            }
            str.append("null");
            for ( int i = 0 ; i < classes.length ; i++)
            {
            str.append(")");
            }
            return str;
    }
```

Ο αλγόριθμος μας δέχεται τις μεθόδους που θέλουμε να καλεστούν και τις τοποθετεί σε αντίθετη σειρά με αυτήν που βρίσκονται στον πίνακα προσθέτοντας σε κάθε μια την συμβολοσειρά ".call(ssc,". Παρακάτω παρατίθενται παραδείγματα εκτέλεσης του αλγορίθμου στην περίπτωση που έχουν σχεδιαστεί οι δύο πίνακες της Εικόνας 6.

Αποτελέσματα (παραγόμενος κώδικας κλήσης τελεστών) αλγόριθμου.
1ου πίνακα : addData.call(ssc, addData.call(ssc,GenerateRandom.call(ssc,null)));
2ου πίνακα: REDUCE_BY_KEY.call(ssc,MAP_TO_PAIR.call(ssc,CreateQ.call(ssc,null)));

Row No.	Source
1	GenerateRandom
2	addData
3	addData

Row No.	Source
1	CreateQ
2	MAP_TO_PAIR
3	REDUCE_BY_KEY

Εικόνα 6: Workflows στην υλοποίηση μας

Τέλος ο παραγόμενος κώδικα που γίνεται compile και στέλνουμε στον jobServer για τις περιπτώσεις που έχουν σχεδιαστεί τα workflows της Εικόνας 6 παρατίθεται παρακάτω.

Παράδειγμα τελικός Κώδικας για SparkContextJob

```
public class SparkJobimplements JSparkJob<String> {

   public String run(JavaSparkContext ssc, JobEnvironment runtime, Config data) {
        char k = '!';

        JavaRDD<Double> randomDRDD = addData.call(ssc,
addData.call(ssc,GenerateRandom.call(ssc,null)));

        StringBuilder sb = new StringBuilder();
```

```
        for(Double row:randomDRDD.collect())
         {
            sb.append(row.toString()).append(k);
        }
        return(sb.toString());

        }

public Config verify(JavaSparkContext sc, JobEnvironment runtime, Config config) {

                        return ConfigFactory.empty();

                        }
        }
```

Παράδειγμα τελικού κώδικα για Spark Streaming Context.

```
public class JavaStreamJob implements JStreamingJob<Integer> {

   public Integer run(StreamingContext context, JobEnvironment jEnv, Config data) {

          final JavaStreamingContext ssc = new JavaStreamingContext(context);

          JavaPairDStream<Integer, Integer> rS = REDUCE_BY_KEY.call(ssc,
MAP_TO_PAIR.call(ssc,CreateQ.call(ssc,null)));

      rS.print();

      ssc.start();

      try {

        ssc.awaitTermination();

      } catch (InterruptedException e) {

        return -1;

      }

      return 1;

    }
```

```
public Config verify(StreamingContext context, JobEnvironment jEnv, Config cfg)  {
    return cfg;
 }
}
```

4.5 Τελεστές που υποστηρίζει η υλοποίηση μας

Όλοι οι τελεστές που είδαμε δημιουργήθηκαν για πειραματικούς σκοπούς και δεν προσφέρουν καμία ουσιαστική επεξεργασία πάνω στα δεδομένα. Όμως, μπορούμε εύκολα να δούμε δύο μεγάλες "ομάδες" διαδικασιών-λειτουργικότητας που είναι πολύ εύκολο με την ήδη υπάρχουσα γνώση να υλοποιηθούν.
Υποστηρίζουμε λοιπόν κάθε μέθοδο της μορφής:

1. JavaRDD call (JavaSparkContext ssc, JavaRDD rdd);
   Έχουν ως παραμέτρους το SparkContext και ένα RDD, το οποίο προφανώς μπορεί να είναι και NULL όπως στον τελεστή SparkDataGenarate, ο οποίος απλά δημιουργεί RDD και δεν παίρνει κάποιο ως είσοδο.

2. JavaDStream call (JavaStreamingContext ssc, JavaDStream rdd);
   Αντίστοιχα, για StreamingContext.

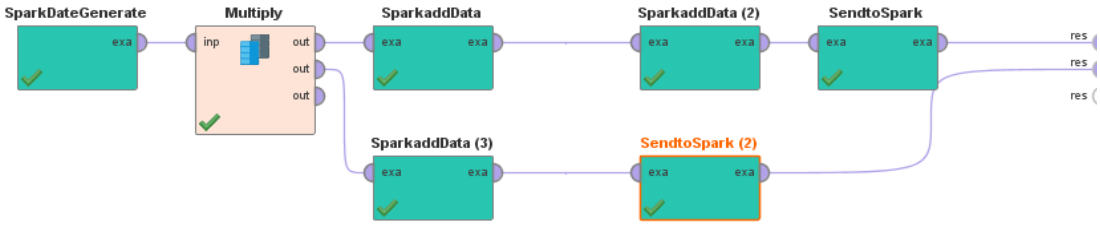
Με αυτές τις δύο μορφές μπορούμε να διαβάσουμε και να αποθηκεύσουμε δεδομένα από διαφορετικές πηγές ως RDDs ή Streams, να κάνουμε οποιαδήποτε μετατροπή θέλουμε πάνω στα δεδομένα και να υπολογίσουμε συναθροιστικές συναρτήσεις και όποια άλλη επεξεργασία επιθυμούμε.

Οι μέθοδοι που δεν μπορούν να υλοποιηθούν με τις μορφές αυτές είναι joins και αλγόριθμοι, που χρειάζονται δύο ή περισσότερα RDDs για να υπολογίσουνε την έξοδο τους, όπως για παράδειγμα υπολογισμός συσχέτισης δύο ή περισσοτέρων RDDs. Οι αλλαγές που θα χρειαζόταν ο κώδικας για να υποστηρίζει joins δεν θα ήταν μεγάλες, αλλά θα ήταν απαραίτητο να γίνουν αλλαγές στον τρόπο σχεδίασης των διεργασιών.

5 Παραδείγματα Υλοποίησης

5.1 Spark Context Job

Σχεδίαση ενός work-flow με τους τελεστές που επεξηγήσαμε. Παρατηρούμε ότι έχουμε δύο sendtoSpark τελεστές, άρα περιμένουμε να δημιουργηθούν δύο SparkJobs.



Εικόνα 7: SparkContextWorkflow

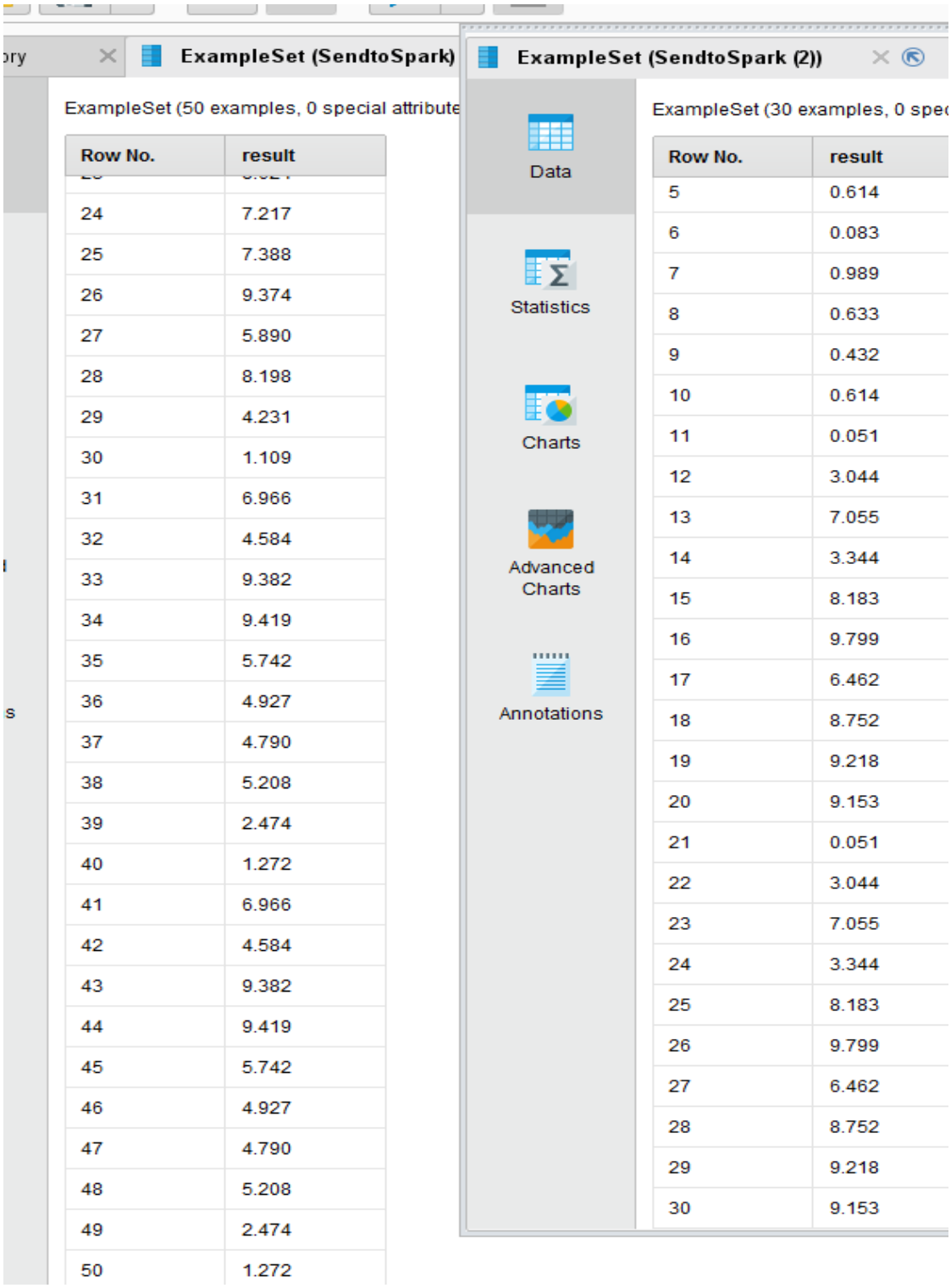
Δύο jobs εμφανίζονται στο user-interface του JobServer με το ίδιο context = jcontext και classpath+"RandomInt"

Completed Jobs

Id	Classpath	Context	Start Time	Duration
e7cdf467-1b5b-4d08-b728-67a6fedc621b (C)	SparkJob9303	jcontext	2017-09-30T23:25:14.625+03:00	0.395 secs
43cc50f7-5f6b-441f-909f-24da4e3759df (C)	SparkJob6769	jcontext	2017-09-30T23:25:10.903+03:00	1.134 secs

Εικόνα 8: SparkContext Server UI

Παρατηρούμε στην Εικόνα 9 στα αποτελέσματα το SendtoSpark να έχει μεγαλύτερο αριθμό αποτελεσμάτων αφού στο πρώτο job έχει χρησιμοποιηθεί μια φορά παραπάνω το addData από ότι στο δεύτερο.



ExampleSet (SendtoSpark)

ExampleSet (50 examples, 0 special attribute

Row No.	result
24	7.217
25	7.388
26	9.374
27	5.890
28	8.198
29	4.231
30	1.109
31	6.966
32	4.584
33	9.382
34	9.419
35	5.742
36	4.927
37	4.790
38	5.208
39	2.474
40	1.272
41	6.966
42	4.584
43	9.382
44	9.419
45	5.742
46	4.927
47	4.790
48	5.208
49	2.474
50	1.272

ExampleSet (SendtoSpark (2))

Data

Statistics

Charts

Advanced Charts

Annotations

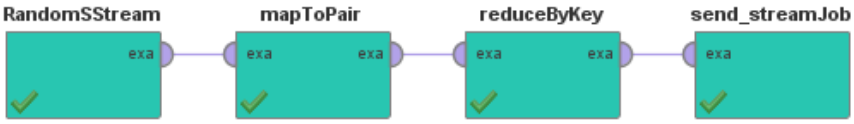
ExampleSet (30 examples, 0 spec

Row No.	result
5	0.614
6	0.083
7	0.989
8	0.633
9	0.432
10	0.614
11	0.051
12	3.044
13	7.055
14	3.344
15	8.183
16	9.799
17	6.462
18	8.752
19	9.218
20	9.153
21	0.051
22	3.044
23	7.055
24	3.344
25	8.183
26	9.799
27	6.462
28	8.752
29	9.218
30	9.153

Εικόνα 9: SparkContextResults

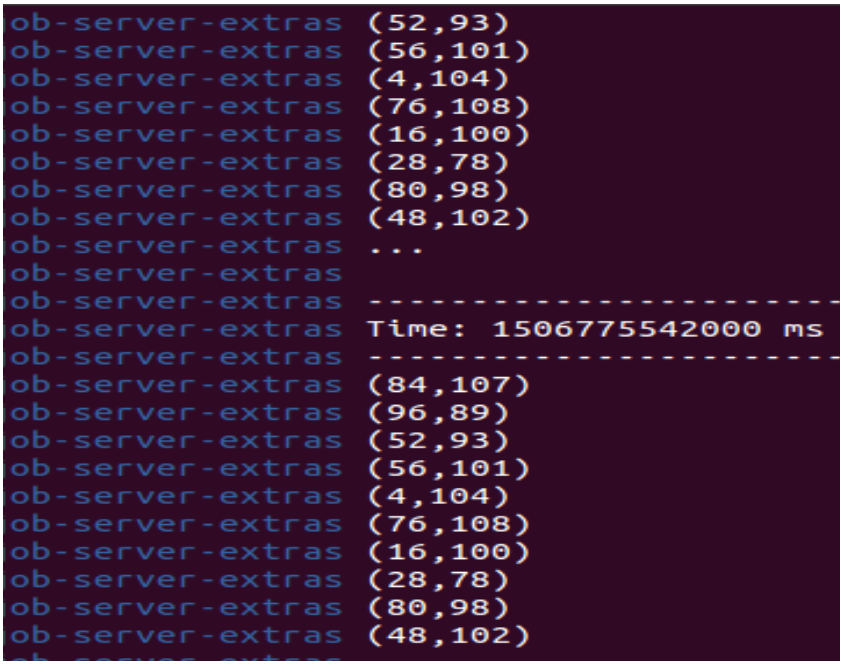
5.2 Spark Streaming Context job

Και ένα streaming Context workflow



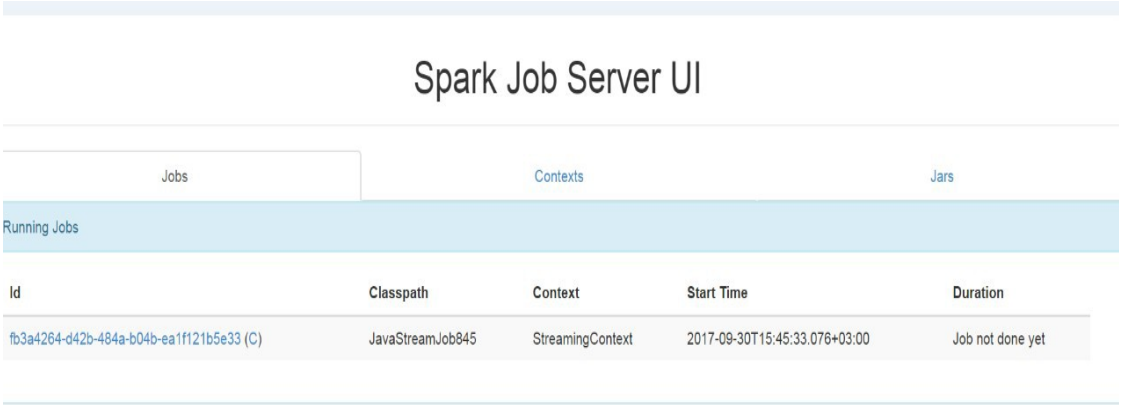
Εικόνες 10: StreamingJobWorkflow

Τα αποτελέσματα που εμφανίζονται στο terminal του JobServer.



Εικόνα 11: SteamingJobResults

Και τέλος το job να τρέχει με StreamingContext και όνομα JavaStreamJob+"Int"



Εικόνα 12: StreamingJob Server UI

6 Συμπεράσματα και μελλοντικές επεκτάσεις

6.1 Συμπεράσματα

Η επέκταση που υλοποιήσαμε, για το RapidMiner Studio είχε ως απώτερο στόχο να προσφέρει, όσον των δυνατών πιο αποδοτικά όλες τις δυνατότητες του Apache Spark στους χρήστες του. Μπορεί να μην υποστηρίζουμε όλες τις λειτουργικότητες του RapidMiner Studio αλλά δεδομένου ότι το χρησιμοποιούμε μόνο για τον σχεδιασμό του κώδικα όλοι οι περιορισμοί που δημιούργησε η εποχή των Μεγάλων Δεδομένων (Big Data) μεταφέρονται στο σύμπλεγμα των Υπολογιστών και δεν επιβαρύνουν καθόλου τον υπολογιστή του χρήστη.

6.2 Μελλοντικές Επεκτάσεις

Η επέκταση για το RapidMiner Studio που αναπτύξαμε έχει την δυνατότητα στο μέλλον με συγκεκριμένες επεκτάσεις, οι οποίες επισημαίνονται συνοπτικά παρακάτω να γίνει περισσότερο λειτουργική και αποδοτική.

- Δημιουργία περισσοτέρων τελεστών με τον τρόπο που αναφέραμε στην ενότητα 5.5 (όσοι είναι οι τελεστές τόση είναι και η λειτουργικότητα που παρέχουμε στον χρήστη).

- Σχεδίαση τελεστών για να μπορούμε να διαβάζουμε και να αποθηκεύουμε δεδομένα από διάφορες πηγές (Sockets, Hive, HDFS, Kafka, Cassandra κ.τ.λ.).

- Υλοποίηση τελεστών για την καλύτερη επικοινωνία των τελεστών που υλοποιούμε με τους ήδη υπάρχοντες τελεστές του RapidMiner Studio.

- Μελετώντας τον τρόπο επικοινωνίας του Spark Job Server με τον Spark cluster να ανακαλύψουμε καινούριους τρόπους επικοινωνίας με τον cluster και να επεκτείνουμε την εφαρμογή μας ώστε να μην είναι απαραίτητη η χρήση του Spark Job Server ως middleware.

7 Βιβλιογραφία

[1] https://rapidminer.com/

[2] https://spark.apache.org/

[3] https://github.com/spark-jobserver/spark-jobserver

[4] https://sfb876.de/streams/doc/rapidminer.html

[5] https://github.com/wsldl123292/jobServerClient

[6] https://github.com/rapidminer/rapidminer-studio

[7] https://mvnrepository.com/