Aristotle University of Thessaloniki, Department of Electronics and Computer Science

Αναγνώριση Προτύπων

Εργασία 1 - Ταξινόμηση

Δημανίδης Αναστάσιος 7422 (dhmtasos@gmail.com)

Κλειτσιώτης Ιωάννης 7447 (ikleitsi@auth.gr)

Θεσσαλονίκη 21/11/2014

Το πρόβλημα

Στην εργασία αυτή μας δόθηκε ένα σύνολο δεδομένων (σετ εκπαίδευσης) το οποίο περιείχε εγγραφές επιβατών του τιτανικού. Πιο συγκεκριμένα, μας δόθηκαν διάφορες πληροφορίες για τους ίδιους (φύλο, ηλικία, κ.λ.π) αλλά και για την διαμονή τους στο πλοίο (τιμή εισιτηρίου, passenger class, καμπίνα κ.λ.π). Εμείς, αξιοποιώντας τις πληροφορίες αυτές και κυρίως αλγόριθμους ταξινόμησης, κληθήκαμε να δημιουργήσουμε και να αξιολογήσουμε μοντέλα τα οποία προβλέπουν αν ένας επιβάτης του τιτανικού έζησε ή όχι. Η διαδικασία που ακολουθήθηκε ήταν η εξής:

1. Επεξεργασία και κανονικοποίηση του συνόλου δεδομένων, ώστε να δημιουργηθούν διάφορα σετ εκπαίδευσης, με σκοπό την μεγιστοποίηση της αποτελεσματικότητας των αλγορίθμων ταξινόμησης.
2. Δοκιμή όλων των σετ εκπαίδευσης του βήματος 1 με μεγάλο πλήθος αλγορίθμων ταξινόμησης από διάφορες κατηγορίες (Δέντρα, Πιθανοτικούς, Νευρωνικά, SVM, KNN).
3. Επιλογή των καλύτερων σετ εκπαίδευσης για τον κάθε αλγόριθμο και έπειτα επιλογή των 2 καλύτερων αλγορίθμων από κάθε κατηγορία.
4. Αξιολόγηση των μοντέλων των καλύτερων αλγορίθμων με βάση τις μετρικές Accuracy, Precision, Recall, F-Measure.
5. Δημιουργία μετα-μοντέλων, όπου κρινόταν απαραίτητο, με την βοήθεια πινάκων κόστους και αξιολόγησή τους με τις ίδιες μετρικές.
6. Επιλογή του καλύτερου μοντέλου.

Ανάλυση

Για την εκπόνηση της εργασίας χρησιμοποιήθηκε το πρόγραμμα Weka (version 3-6). Σε κάποια σημεία χρησιμοποιήθηκε και MATLAB για data visualizations αλλά και η γλώσσα ruby για αυτοματοποιημένα data manipulations. Επίσης, αν και γενικώς δεν επισημαίνεται, αξιοποιήθηκε το πρόγραμμα Microsoft Excel για την καλύτερη κατανόηση (ταξινομήσεις) και διαχείριση (διαγραφή/ένωση στηλών, εφαρμογή formulas) των δεδομένων.

Να σημειωθεί ότι καθώς κάναμε ανάλυση των δεδομένων, το training set εξελίσσονταν. Προκειμένου να αξιολογήσουμε και τα σετ δεδομένων και να επιλέξουμε το καλύτερο, διατηρούσαμε όλες τις προηγούμενες μορφές του σετ δεδομένων έως και την τρέχουσα παραλλαγή του. Έτσι στο τέλος της εξερεύνησης των δεδομένων θα υπάρχει ένα μεγάλο πλήθος training set.

1. Εξερεύνηση – Καθαρισμός – Κανονικοποίηση Δεδομένων

Αρχικά κατεβάσαμε το αρχείο train.csv από την ιστοσελίδα του διαγωνισμού. Το αρχείο αυτό περιείχε πληροφορίες για τους επιβάτες του Τιτανικού. Συγκεκριμένα είχε την παρακάτω μορφή:

	A	B	C	D	E	F	G	H	I	J	K	L
1	Passenge	Survived	Pclass	Name	Sex	Age	SibSp	Parch	Ticket	Fare	Cabin	Embarked
2	1	0	3	Braund, Mr. Owen Harris	male	22	1	0	A/5 21171	7.25		S
3	2	1	1	Cumings, Mrs. John Brad	female	38	1	0	PC 17599	71.2833	C85	C
4	3	1	3	Heikkinen, Miss. Laina	female	26	0	0	STON/O2.	7.925		S
5	4	1	1	Futrelle, Mrs. Jacques He	female	35	1	0	113803	53.1	C123	S
6	5	0	3	Allen, Mr. William Henry	male	35	0	0	373450	8.05		S

Εικόνα 1 – Αρχικό σετ δεδομένων

[Προκειμένου να εισάγουμε το σετ δεδομένων στο Weka απαιτούνταν να αφαιρεθούν τα επιπλέον εισαγωγικά που υπήρχαν στην στήλη Name. Αυτό έγινε αυτοματοποιημένα με ένα απλό ruby script]

Με απλή επισκόπηση του αρχείο διαπιστώσαμε ότι οι στήλες PassengerId, Name, Ticket και Cabin δεν μπορούν να αξιοποιηθούν από μία διαδικασία εκπαίδευσης στην μορφή που δίνονται. Αυτό γιατί οι εγγραφές των στηλών αυτών είναι τύπου nominal και κατά κανόνα είναι διαφορετικές μεταξύ τους. Άρα σε μία διαδικασία πρόβλεψης με ένα καινούργιο test set, οι nominal αυτές μεταβλητές δεν θα υπήρχαν (π.χ. θα υπήρχαν διαφορετικά names επιβατών) και άρα το μοντέλο από την διαδικασία εκπαίδευσης θα στερούνταν νοήματος. Συνεπώς αποφασίστηκε να αφαιρεθούν αρχικά αυτές οι στήλες και να αξιοποιηθούν απλώς για άντληση πληροφορίας. Την πληροφορία αυτή θα την εισαγάγαμε ενδεχομένως

στο σετ δεδομένων σε ένα μετέπειτα στάδιο της προ επεξεργασίας. Το σετ δεδομένων ύστερα από αυτό το βήμα απέκτησε την παρακάτω μορφή:

	A	B	C	D	E	F	G	H	I	J	K	L
1	Pclass	Sex	Age	SibSp	Parch	Fare	Embarked	Survived				
2	3	male	22	1	0	7.25	S	0				
3	1	female	38	1	0	71.2833	C	1				
4	3	female	26	0	0	7.925	S	1				
5	1	female	35	1	0	53.1	S	1				
6	3	male	35	0	0	8.05	S	0				

Εικόνα 2 – Σετ δεδομένων με αφαίρεση στηλών

[Η στήλη survived μετακινήθηκε στο τέλος για ταχύτερη αναγνώρισή της ως στήλη-class από το Weka]

Εξετάζοντας περαιτέρω τα αρχικά δεδομένα (απλή ταξινόμηση ως προς την στήλη Ticket), διαπιστώσαμε ότι η στήλη Fare δεν περιέχει το ναύλο του κάθε επιβάτη ξεχωριστά. Πιο συγκεκριμένα, παρατηρήσαμε ότι κάποιοι επιβάτες ταξίδευαν μαζί. Σε κάθε ένα από αυτούς τους επιβάτες η αναγραφόμενη τιμή εισιτηρίου ήταν το συνολικό άθροισμα των εισιτηρίων των επιβατών μαζί με τους οποίους ταξίδευαν. Παραδείγματος χάρη οι επιβάτες 381, 558,701, 717 είχαν όλοι το ίδιο Ticket και το ίδιο Fare:

800	381	1	1	Bidois, Miss. Rosalie	female	42	0	0	PC 17757	227.525		C
801	558	0	1	Robbins, Mr. Victor	male		0	0	PC 17757	227.525		C
802	701	1	1	Astor, Mrs. John Jacob	female	18	1	0	PC 17757	227.525	C62 C64	C
803	717	1	1	Endres, Miss. Caroline	female	38	0	0	PC 17757	227.525	C45	C

Εικόνα 3 – Επιβάτες που ταξίδευαν μαζί

Η σκέψη λοιπόν ήταν, να αλλάξουμε την στήλη Fare ώστε να δείχνει το ναύλο του εισιτηρίου που αντιστοιχεί σε ένα άτομο (fare by person). Έτσι με την βοήθεια ενός ruby script (fbp.rb στα αρχεία της εργασίας) και αξιοποιώντας την στήλη Ticket, βρήκαμε τους επιβάτες που έχουν ίδιο Ticket και διαιρέσαμε την τιμή του εισιτηρίου τους με το πλήθος των επιβατών που είχαν ίδιο Ticket:

	A	B	C	D	E	F	G	H	I	J	K	L
1	Pclass	Sex	Age	SibSp	Parch	Fare	Embarked	Survived				
2	3	male	22	1	0	7.25	S	0				
3	1	female	38	1	0	71.2833	C	1				
4	3	female	26	0	0	7.925	S	1				
5	1	female	35	1	0	26.55	S	1				
6	3	male	35	0	0	8.05	S	0				

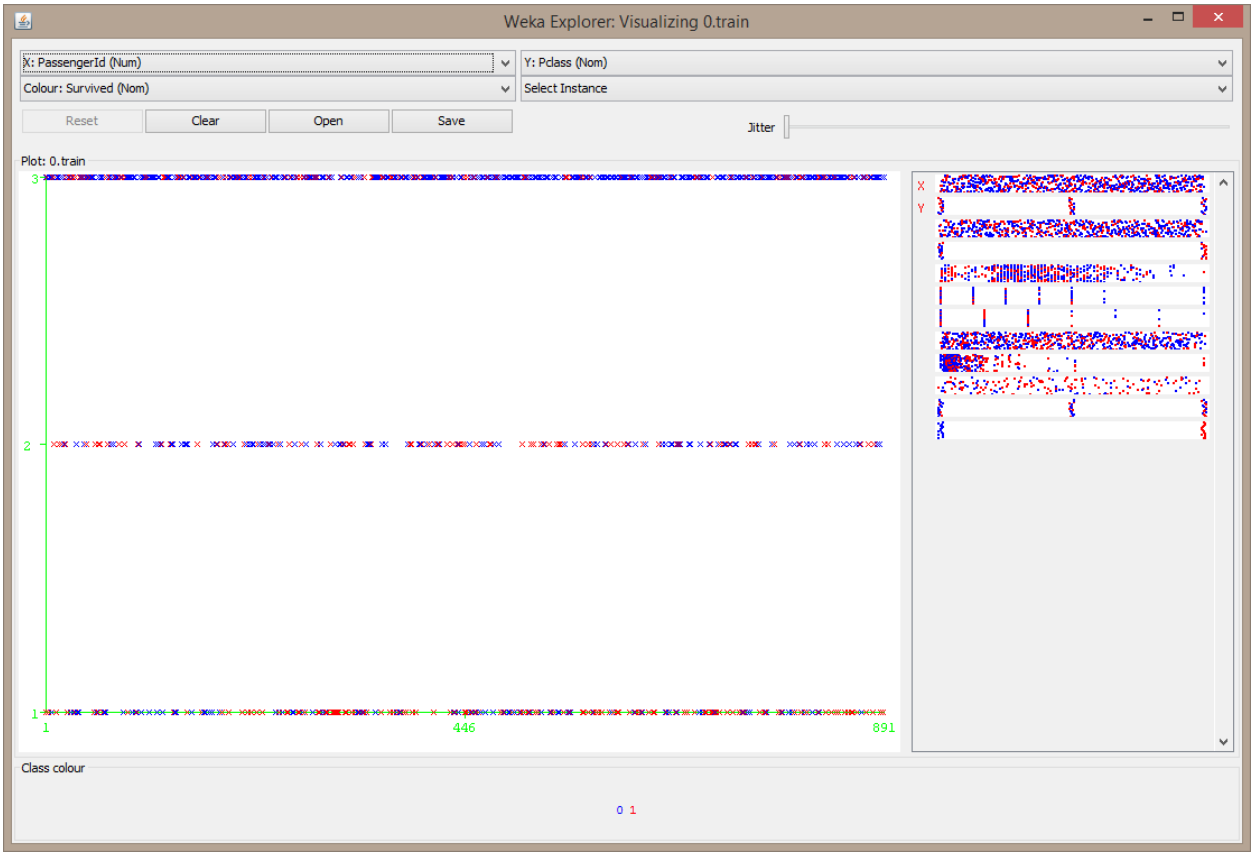
Εικόνα 4 – Μετατροπή της στήλης Fare

Πλέον λοιπόν στο σετ δεδομένων μας η στήλη Fare δείχνει το ναύλο ανά άτομο.

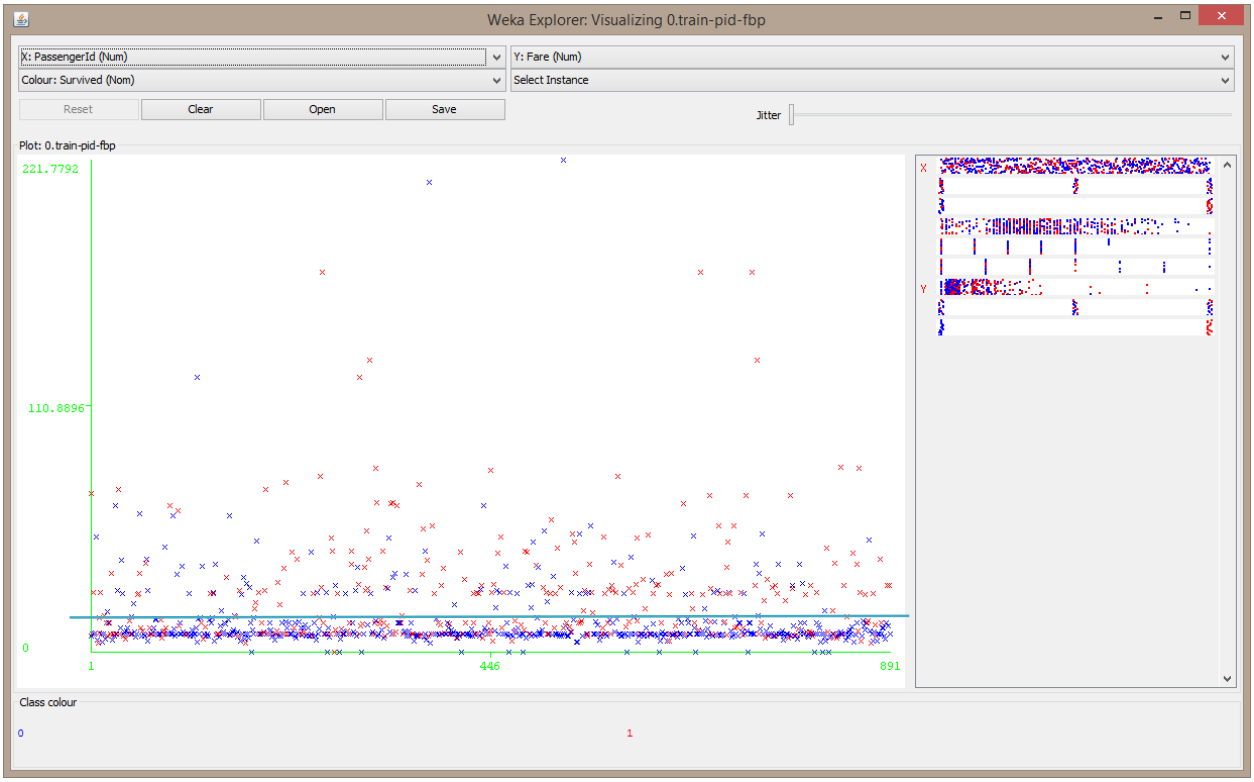
A. Χαρακτηριστικό Type

Το επόμενο βήμα ήταν να διερευνήσουμε, ποια χαρακτηριστικά επιβατών έπαιξαν σημαντικότερο ρόλο στην επιβίωσή τους. Για τον λόγο αυτό ανατρέξαμε σε απεικονίσεις δεδομένων. Η αρχική διαίσθησή μας ήταν ότι σημαντικό ρόλο θα έπαιζαν τα κοινωνικό-οικονομικά χαρακτηριστικά των επιβατών. Δηλαδή οι στήλες Pclass και Fare. Απεικονίσαμε λοιπόν τα δύο αυτά χαρακτηριστικά στο Weka:

[Σε όλα τα διαγράμματα στο Weka, το κόκκινο χρώμα αντιστοιχεί στους survived (1), και το μπλε στους not survived (0)]

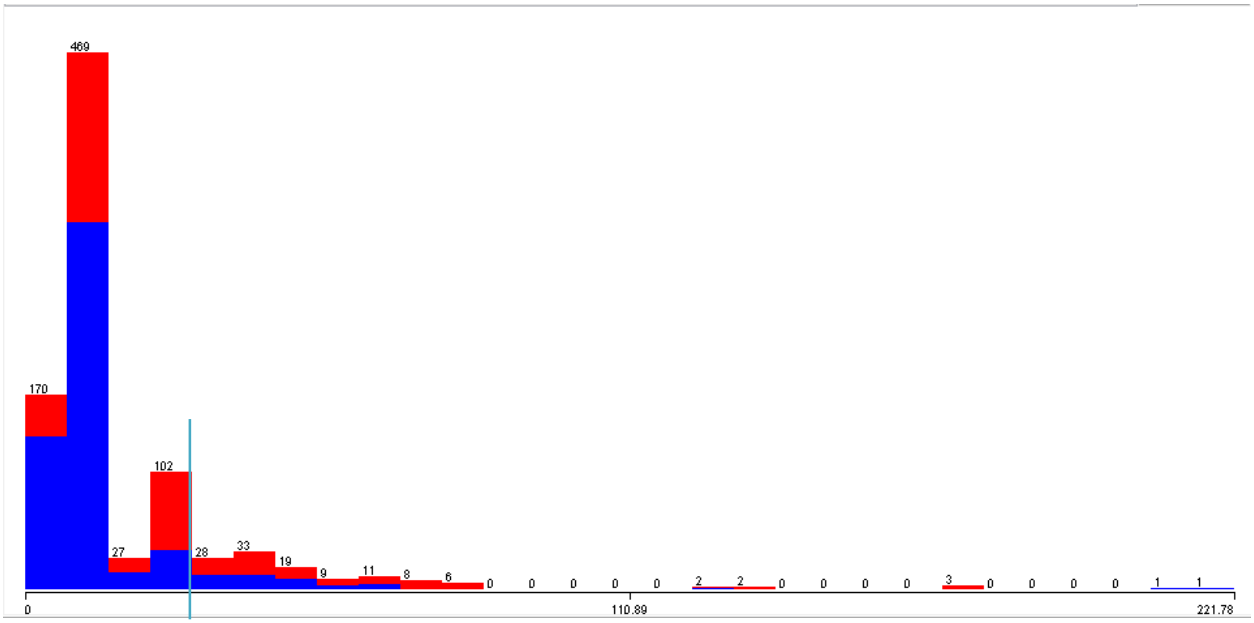


Διάγραμμα 1 - PassengerId – Pclass



Διάγραμμα 2 - PassengerId – Fare (by person)

Το πρώτο διάγραμμα δείχνει αρκετά καθαρά ότι, οι επιβάτες που πέθαναν και ανήκαν στο Pclass 3, ήταν πολύ περισσότεροι από τους επιβάτες που πέθαναν σε κάθε ένα από τα άλλα 2 passenger class. Το δεύτερο διάγραμμα από την άλλη, δεν δίνει επιπλέον πληροφορία, για το πως επηρέασε το Fare στην επιβίωση. Κυρίως μας δείχνει ότι οι επιβάτες που πλήρωσαν υψηλό εισιτήριο, έχουν μεγαλύτερο ποσοστό επιβίωσης, το οποίο φαίνεται και από το πρώτο διάγραμμα (με την λογική ότι τα pclass 1 και 2 είχαν ακριβότερο εισιτήριο από το 3). Επίσης μας δείχνει ότι το μεγαλύτερο πλήθος των επιβατών, δεν πλήρωσε υψηλό εισιτήριο (γαλάζια γραμμή). Αυτό επιβεβαιώνεται και από την κατανομή της τιμής του εισιτηρίου ως προς τους επιβάτες:



Διάγραμμα 3 - Fare (by person) - PassengerId

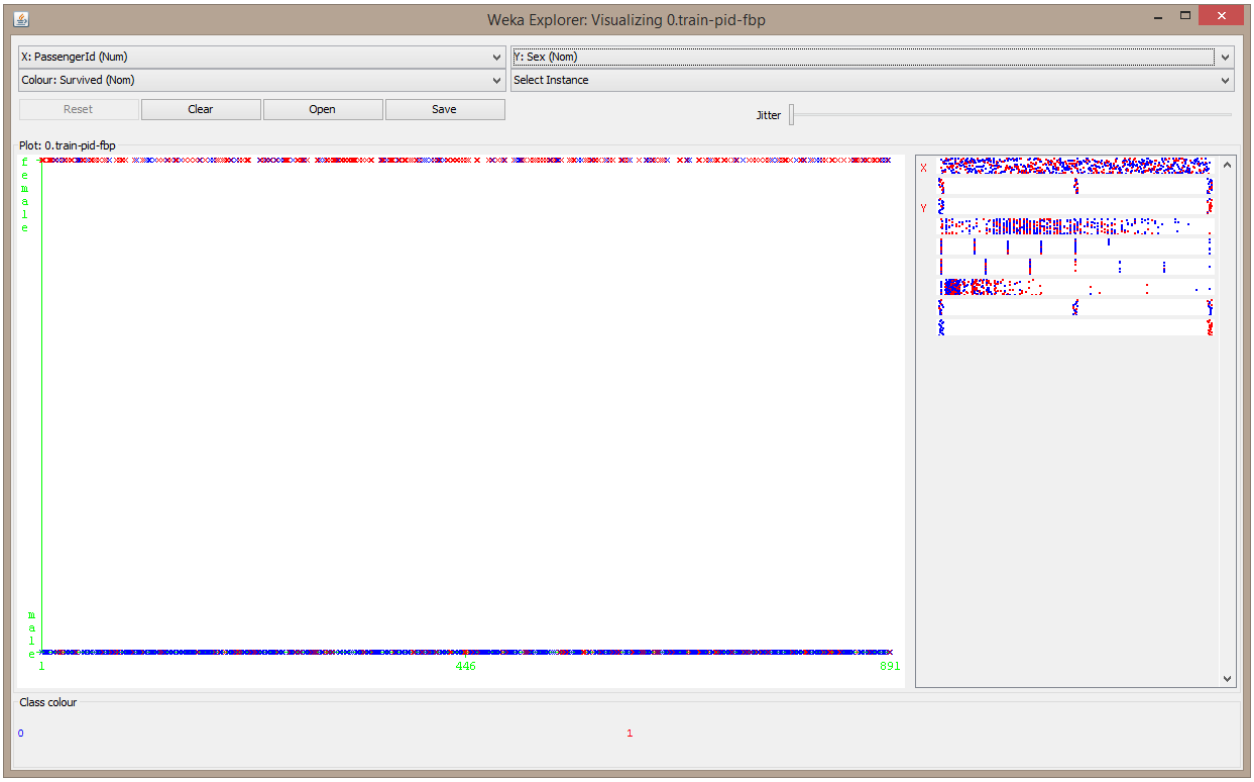
Που σημαίνει ότι για ένα μεγάλο ποσοστό επιβατών, χρειάζεται να διερευνήσουμε και άλλα χαρακτηριστικά, για να αποκτήσουμε χρήσιμη πληροφορία.

Αναζητώντας λοιπόν και άλλα σημαντικά χαρακτηριστικά, βρήκαμε ότι όσο βούλιαζε το πλοίο, υπήρχε οδηγία από τον καπετάνιο να δίνεται προτεραιότητα στο σώσιμο των γυναικών και των παιδιών.

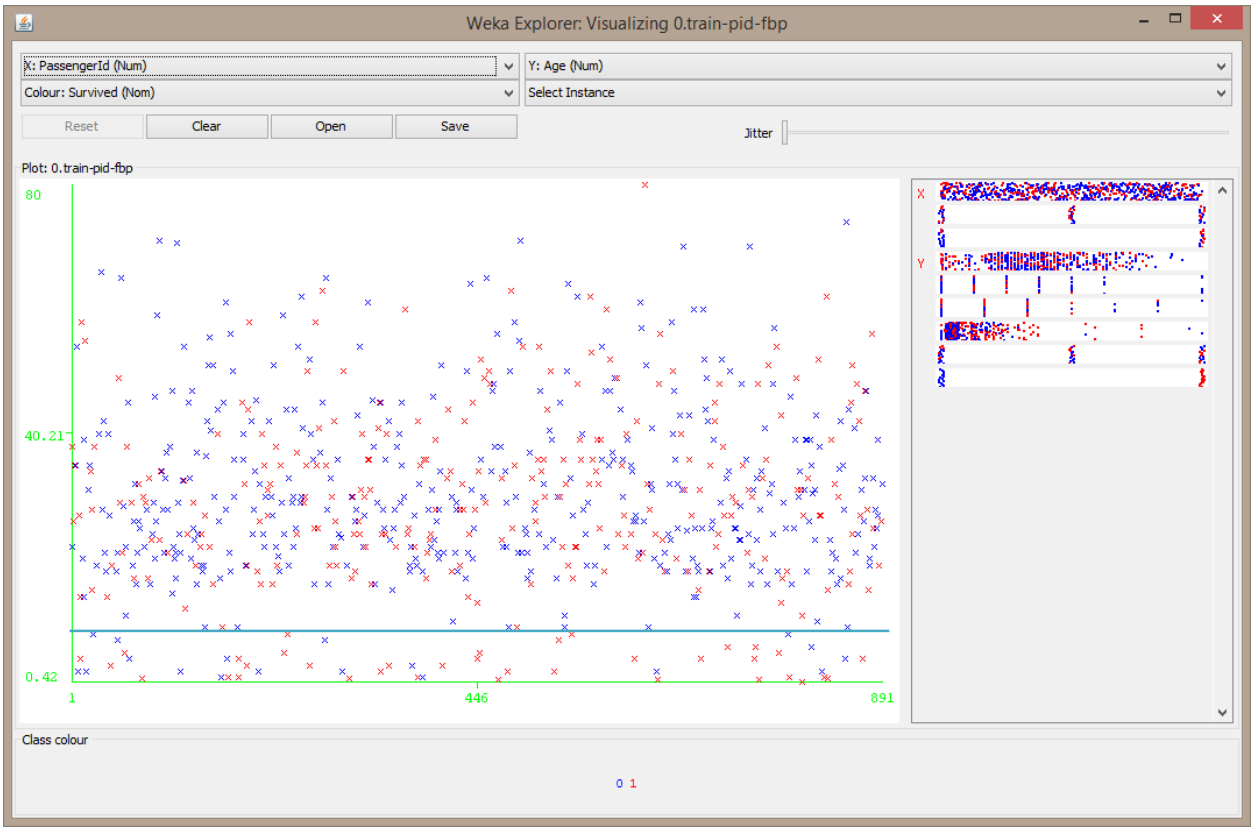
«Ένας δυσανάλογος αριθμός ανδρών έμεινε στο πλοίο επειδή ακολουθήθηκε ένα πρωτόκολλο "πρώτα γυναίκες και παιδιά" από μερικούς αξιωματικούς που φόρτωναν τις βάρκες[3]*»*

Γνωρίζοντας αυτήν την λεπτομέρεια προχωρήσαμε στις απεικονίσεις κατά Sex και Age:

[3] http://el.wikipedia.org/wiki/Τιτανικός



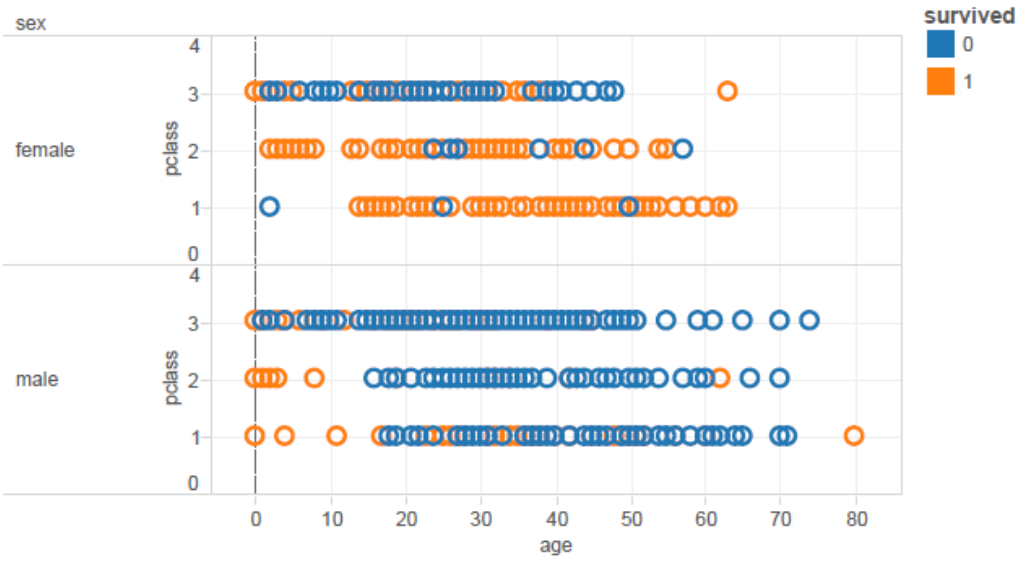
Διάγραμμα 4 - PassengerId – Sex



Διάγραμμα 5 - PassengerId – Age

Από το διάγραμμα 4 φαίνεται ξεκάθαρα ότι το φύλο έπαιξε καθοριστικό ρόλο στην επιβίωση των επιβατών. Στο διάγραμμα 5 από την άλλη η πληροφορία δεν είναι καθαρή, αλλά εστιάζοντας λίγο περισσότερο παρατηρούμε ότι οι επιβάτες κάτω από την γαλάζια γραμμή έχουν ικανοποιητικό ρυθμό επιβίωσης. Οι επιβάτες αυτοί αντιστοιχούν σε μικρές ηλικίες, δηλαδή σε παιδιά. Πράγματι λοιπόν η οδηγία του καπετάνιου για την προτεραιότητα των γυναικόπαιδων, φαίνεται στις απεικονίσεις των δεδομένων μας.

Συνοψίζοντας τις πληροφορίες από την παραπάνω ανάλυση σε ένα διάγραμμα:



Διάγραμμα 6.1 – Επιβίωση επιβατών βάσει φύλου, ηλικίας και passenger class[4]

Και μία άλλη απεικόνιση που δείχνει καθαρά ότι τα χαρακτηριστικά αυτά παίζουν μεγάλο ρόλο:

Διάγραμμα 6.2 – Επιβίωση επιβατών βάσει φύλου, ηλικίας και passenger class[5]

[4] https://www.kaggle.com/c/titanic-gettingStarted/prospector#1234
[5] http://nbviewer.ipython.org/gist/mwaskom/8224591 - Plot [37]

Τώρα από απλή ταξινόμηση της στήλης Age στο Excel διαπιστώσαμε ότι όλοι οι επιβάτες κάτω από 1 χρονών έζησαν. Διαπιστώσαμε επίσης όμως ότι υπήρχαν επιβάτες χωρίς αναγραφόμενη ηλικία. Στο σημείο αυτό σκεφτήκαμε να αξιοποιήσουμε την στήλη Names στην οποία πέρα από τα ονόματα, αναγράφονται και οι τίτλοι των επιβατών (Mr., Master, Miss, Mrs., κλπ.). Από τους τίτλους αυτούς μπορεί να βγει ένα συμπέρασμα για τις ηλικίες των ανδρών (μιας και Miss είναι απλώς ανύπαντρη γυναίκα). Θεωρήσαμε δηλαδή τους επιβάτες με τίτλο Master ως παιδιά και τους επιβάτες με τίτλο Mr. ως ενήλικες άντρες. Για τις γυναίκες των οποίων η ηλικία έλειπε, ανεχτήκαμε ένα βαθμό ανακρίβειας μιας και

1. Ο τίτλος της γυναίκας δεν δίνει σαφή πληροφορία για την ηλικία
2. Ένας επιβάτης είτε ήταν ενήλικη γυναίκα είτε μικρό κορίτσι, άνηκε στον κανόνα «γυναίκες και παιδιά πρώτα».

Συνεπώς αποφασίσαμε ότι για τις γυναίκες των οποίων η ηλικία έλειπε, δεν θα κάναμε τον διαχωρισμό γυναίκα – παιδί. Επίσης θεωρήσαμε, με ένα βαθμό προσωπικής εκτίμησης, ότι παιδί θεωρείται ένας επιβάτης κάτω από 15 χρονών (να φαίνεται μικρός/ή δηλαδή). Να σημειώσουμε εδώ ότι η εξαγωγή των τίτλων των ανδρών έγινε με τη βοήθεια ενός ruby script (titles.rb στα αρχεία της εργασίας). Βασιζόμενοι στα παραπάνω δημιουργήσαμε ένα καινούργιο χαρακτηριστικό, το χαρακτηριστικό Type. Στο χαρακτηριστικό αυτό συνοψίζεται η ωφέλιμη (κατά την ανάλυσή μας) πληροφορία των χαρακτηριστικών Sex, Age και Name. Το χαρακτηριστικό Type περιέχει τις εξής nominal τιμές:

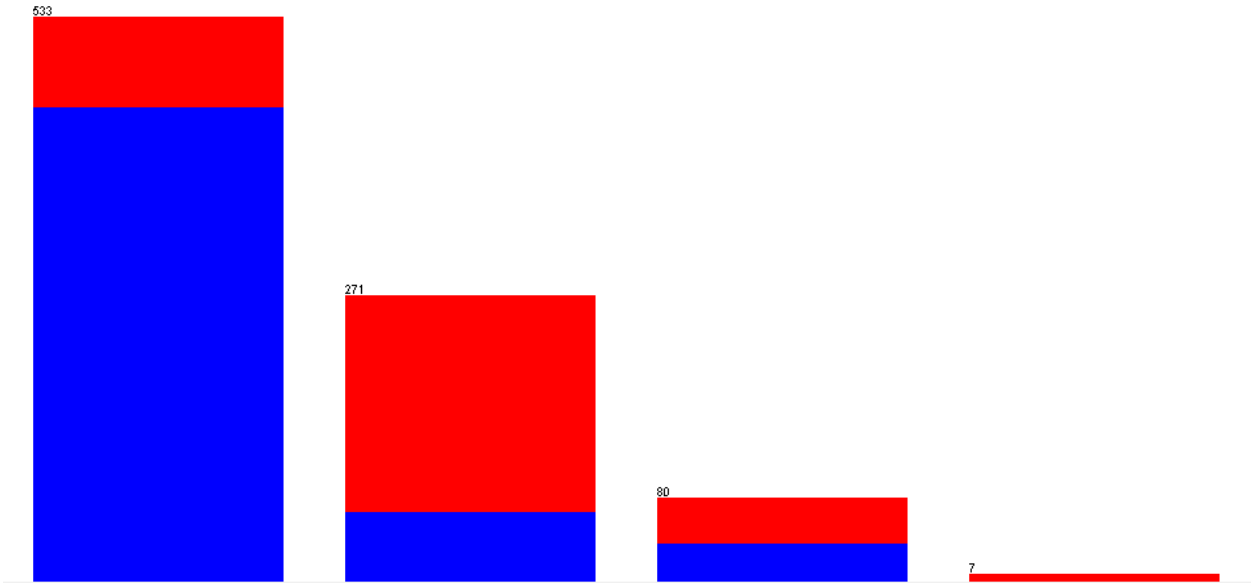
- adult-male
- child
- female (κυρίως adult-female)
- newborn

Πλέον το σετ δεδομένων είναι κάπως έτσι:

	A	B	C	D	E	F	G	H	I	J	K
1	Pclass	Type	SibSp	Parch	Fare	Embarked	Survived				
2	3	adult-mal	1	0	7.25	S	0				
3	1	female	1	0	71.2833	C	1				
4	3	female	0	0	7.925	S	1				
5	1	female	1	0	26.55	S	1				
6	3	adult-mal	0	0	8.05	S	0				

Εικόνα 5 – Συνένωση των στηλών Age, Sex και πληροφορίας από την Name στην στήλη Type

Ενώ η πιθανότητα επιβίωσης των τιμών του χαρακτηριστικού Type:

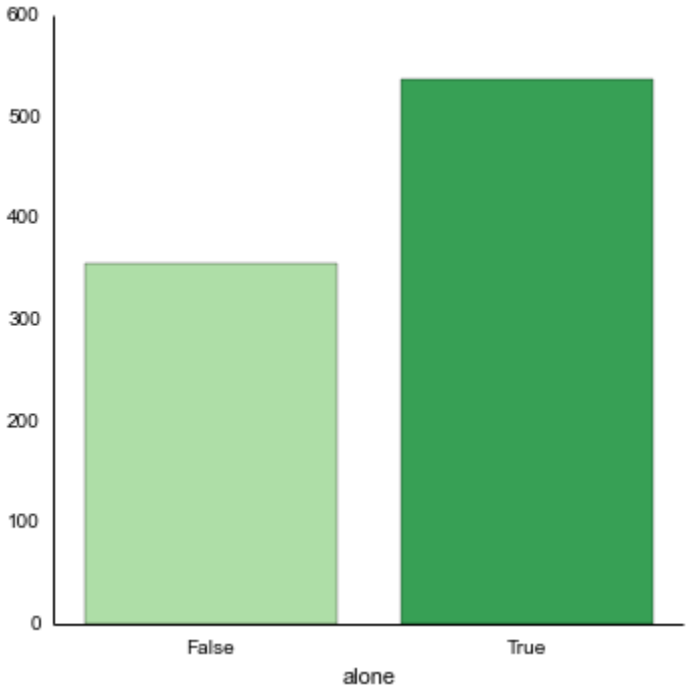


Διάγραμμα 7 – Type (Adult male, female, child, newborn) – PassengerId

Β. Χαρακτηριστικό Company

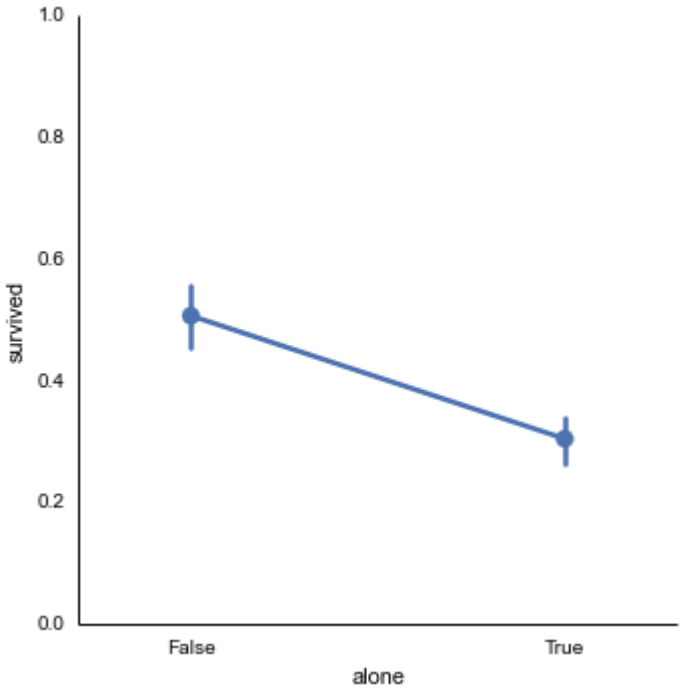
Στη συνέχεια, διερευνήσαμε τα χαρακτηριστικά SibSp και ParCh, τα οποία περιέχουν τον αριθμό αδελφών-συζύγων και γονέων-τέκνων αντίστοιχα. Εξαρχής φαίνεται ότι η πληροφορία που δίνουν αυτά τα attributes δεν είναι καλά κωδικοποιημένη, αφού κάθε χαρακτηριστικό είναι το άθροισμα δύο αρκετά διαφορετικών (για το πρόβλημά μας) επιμέρους χαρακτηριστικών (π.χ για το ParCh - το να έχει κάποιος 2 γονείς περιέχει διαφορετική βαρύτητα από το να έχει 2 παιδιά επιβάτες, αλλά το attribute δεν τα διαχωρίζει). Επίσης δύσκολα θα μπορούσαμε να εξαγάγουμε σαφείς πληροφορίες για την οικογενειακή κατάσταση κάποιου βασιζόμενοι μόνο σε αυτά τα χαρακτηριστικά (ParCh = 2 σημαίνει ότι κάποιος ήταν παιδί με δύο γονείς, γονέας με δύο παιδιά κτλ.), οπότε μόνο σε συνδυασμό με άλλα attributes, και με ευρετικό τρόπο, θα μπορούσαμε να βρούμε λεπτομέρειες για τις οικογένειες των επιβατών.

Η πρώτη μας προσέγγιση ήταν να συγχωνεύσουμε την πληροφορία από τα δύο αυτά χαρακτηριστικά σε ένα, το οποίο απλά θα δήλωνε το αν κάποιος επιβάτης ήταν μόνος του ή όχι επάνω στο πλοίο.



Διάγραμμα 8 – Alone - PassengerId[6]

[6] http://nbviewer.ipython.org/gist/mwaskom/8224591 - Plot [33]

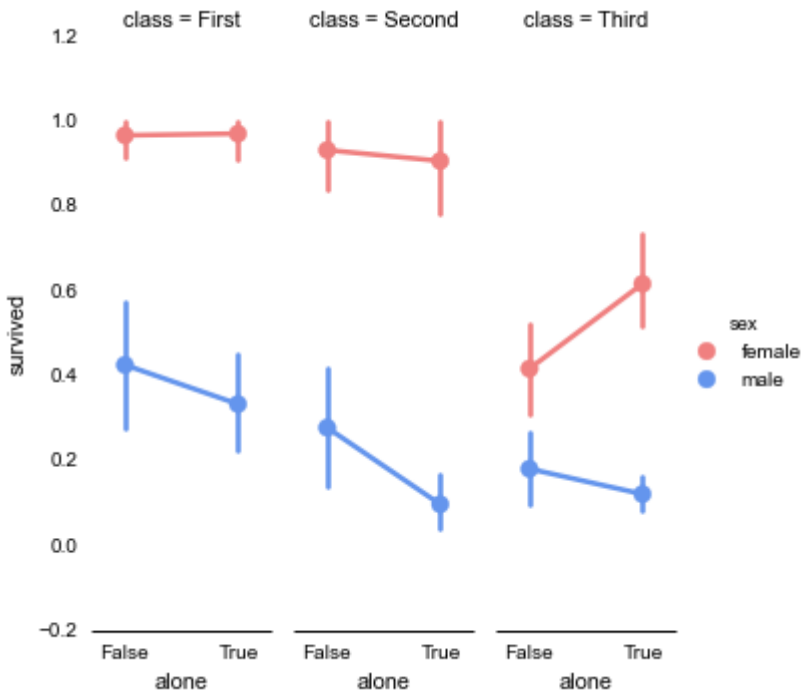


Διάγραμμα 9 - Alone - Survived[7]

Από τα παραπάνω δύο διαγράμματα συμπεραίνουμε ότι:

1. Κανένα από τα ενδεχόμενα alone-not alone δεν έχει αμελητέα πιθανότητα πραγματοποίησης
2. Οι επιβάτες που ταξίδευαν με οικογένεια είχαν ελαφρώς μεγαλύτερη πιθανότητα επιβίωσης. Οπότε έχει νόημα αυτού του είδους η συγχώνευση. Ως επαλήθευση, εξετάζουμε ένα διάγραμμα με περισσότερο «συνδυαστική» πληροφορία, ώστε να είμαστε σίγουροι ότι το survivability δεν προέκυψε απλά από τον συνδυασμό άλλων χαρακτηριστικών.

[7] http://nbviewer.ipython.org/gist/mwaskom/8224591 - Plot [51]



Διάγραμμα 10 – Alone - Survived για κάθε class/sex[8]

Στο αυτό το διάγραμμα 10 βλέπουμε πάλι, ότι αν εξαιρέσουμε τις γυναίκες της τρίτης κλάσης, οι alone επιβάτες είχαν μεγαλύτερη θνησιμότητα. Έτσι δημιουργήσαμε ένα νέο σετ δεδομένων στο οποίο συγχωνεύτηκαν τα χαρακτηριστικά Sibsp και Parch στο χαρακτηριστικό Company:

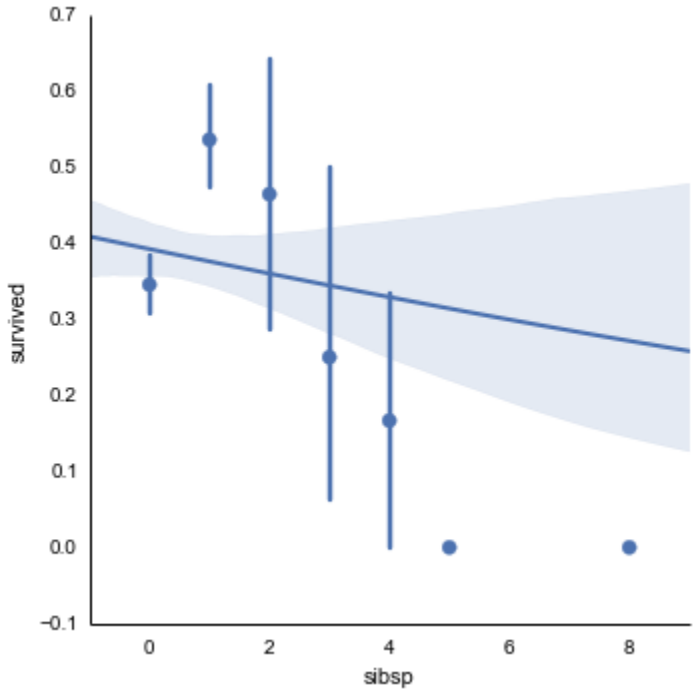
	A	B	C	D	E	F	G	H	I	J
1	Pclass	Type	Fare	Embarked	Company	Survived				
2	3	adult-mal	7.25	S	not-alone	0				
3	1	female	71.2833	C	not-alone	1				
4	3	female	7.925	S	alone	1				
5	1	female	26.55	S	not-alone	1				
6	3	adult-mal	8.05	S	alone	0				

Εικόνα 6 – Συγχώνευση των στηλών Sibsp και Parch στην στήλη Company

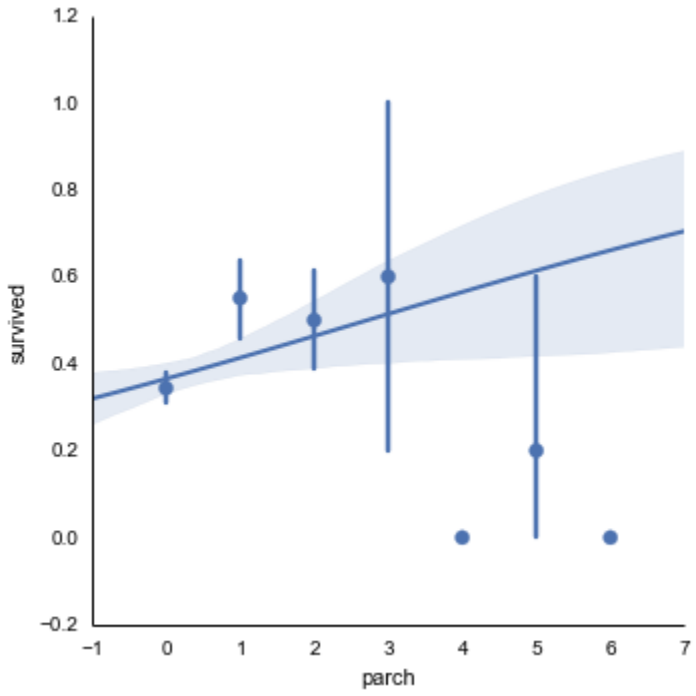
Γ. Sibsp nominal

Η δεύτερη προσέγγιση προέκυψε από την εξέταση των παρακάτω γραφημάτων:

[8] http://nbviewer.ipython.org/gist/mwaskom/8224591 - Plot [55]



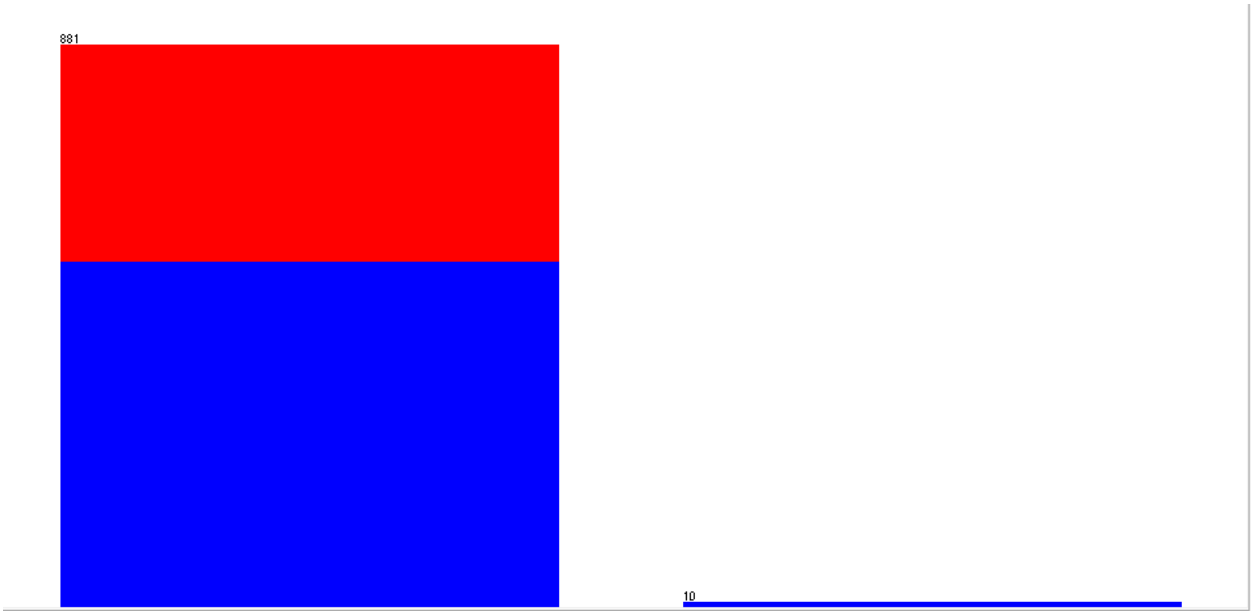
Διάγραμμα 11 – Sibsp - Survived[9]



Διάγραμμα 12 – Parch - Survived[10]

[9] http://nbviewer.ipython.org/gist/mwaskom/8224591 - Plot [50]
[10] http://nbviewer.ipython.org/gist/mwaskom/8224591 - Plot [48]

Από το πρώτο φαίνεται ότι οι επιβάτες που το άθροισμα αδερφών-συζύγων τους ήταν έως και 2, είχαν σχετικά μικρότερη θνησιμότητα. Το ίδιο ισχύει και για αυτούς που είχαν το parch έως και 3. Για το parch όμως αν το χωρίσουμε στην τιμή 3 έχουμε την εξής κατανομή επιβατών:



Διάγραμμα 13 – Parch (nominal) - PassengerId

Προφανώς δεν έχει νόημα να χωρίσουμε το parch αφού επηρεάζει αμελητέο αριθμό εγγραφών. Συνεπώς το parch στην numerical μορφή του χρησιμοποιήθηκε ως brute force χαρακτηριστικό.

Έτσι χωρίσαμε μόνο το χαρακτηριστικό sibsp σε 2 bins:

- less (<=2)
- more (>2)

Έτσι μία παραλλαγή του σετ δεδομένων με το company είναι κάπως έτσι:

	A	B	C	D	E	F	G	H	I	J
1	Pclass	Type	Fare	Embarked	SibSp	Parch	Survived			
2	THREE	adult-mal	7.25	S	less	0	FALSE			
3	ONE	female	71.2833	C	less	0	TRUE			
4	THREE	female	7.925	S	less	0	TRUE			
5	ONE	female	26.55	S	less	0	TRUE			

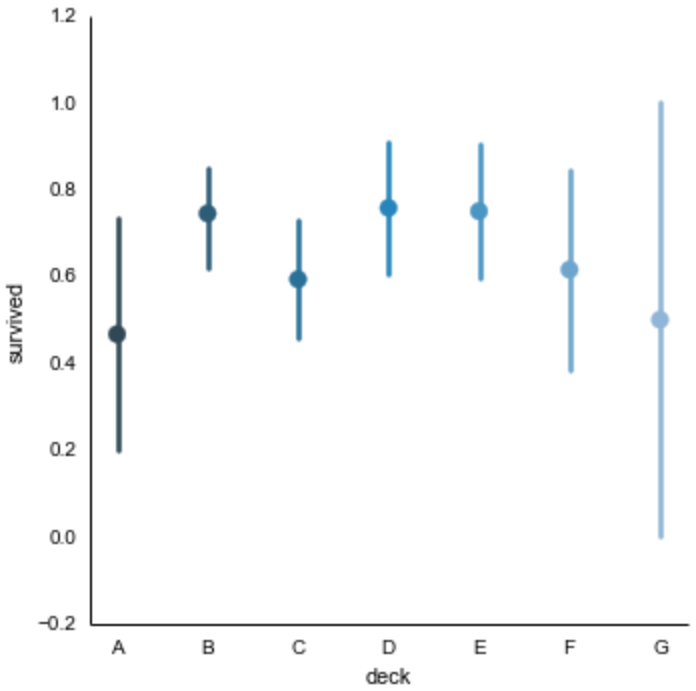
Εικόνα 7 - Μετατροπή των μεταβλητών των στηλων Sibsp και Parch σε nominal

Δ. Χαρακτηριστικό Deck

Στο επόμενο στάδιο της ανάλυσης ασχοληθήκαμε με το χαρακτηριστικό Cabin. Αρχικά διαπιστώσαμε ότι το χαρακτηριστικό αυτό δύσκολα θα μπορούσε να χρησιμοποιηθεί ως έχει. Αυτό γιατί

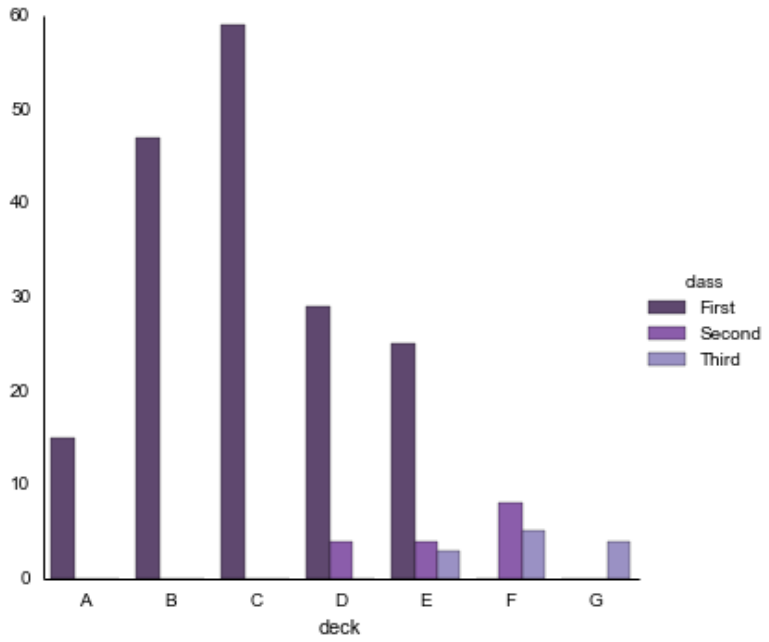
1. Οι καμπίνες κάθε εγγραφής είναι μοναδικές (όπως τα names) και άρα δεν μπορούν να χρησιμοποιηθούν όπως είναι σε μία διαδικασία πρόβλεψης.
2. Για τους περισσότερους επιβάτες δεν αναγράφεται καμπίνα (687/891=77%).

Στην συνέχεια εξετάσαμε τα παρακάτω διαγράμματα, τα οποία ασχολούνται μόνο με τα καταστρώματα των καμπινών:



Διάγραμμα 14 – Deck - Survived[11]

[11] http://nbviewer.ipython.org/gist/mwaskom/8224591 - Plot [46]



Διάγγραμα 15 – Deck – Passengers - Pclass[12]

Στο πρώτο διάγραμμα βλέπουμε ότι η θνησιμότητα των επιβατών δεν ήταν η ίδια για κάθε κατάστρωμα. Παρ' όλ' αυτά, πολλοί παράγοντες θα μπορούσαν να παίζουν ρόλο σε αυτό.

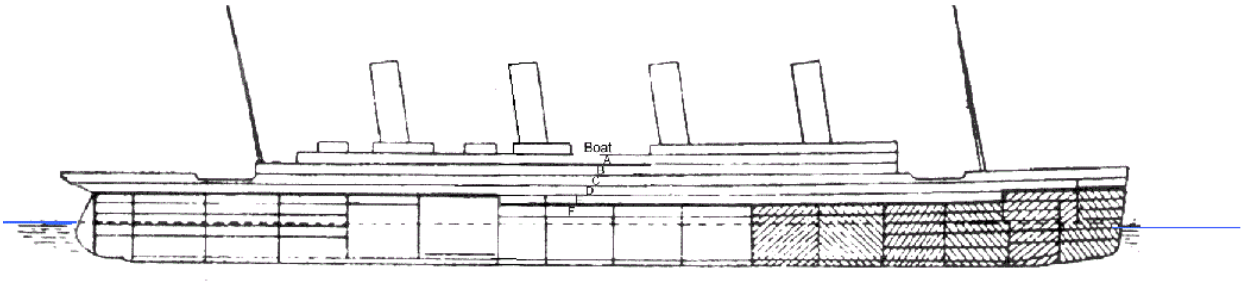
Στο δεύτερο διάγραμμα βλέπουμε απλώς την κατανομή των επιβατών ανάλογα με το pclass στα decks. Αυτό το διάγραμμα θα μπορούσε να αξιοποιηθεί για να βρεθεί έστω το deck με βάση το Pclass. Διερωτηθήκαμε όμως τι νόημα θα είχε κάτι τέτοιο. Να βρούμε δηλαδή κάποιο λόγο για τον οποίο το κατάστρωμα θα επηρέαζε την πιθανότητα επιβίωσης του επιβάτη (μέσω τις θέσης του στο πλοίο). Αναζητώντας διάφορες εγκυκλοπαιδικές πηγές βρήκαμε ότι υπήρχαν επιβάτες των οποίων η θέση μέσα στο πλοίο, όσο αυτό βούλιαζε, ενδεχομένως να έπαιξε κάποιο ρόλο. Συγκεκριμένα:

1. Κάποιοι επιβάτες του τρίτου Pclass βρισκόταν μέσα στο πλοίο ενώ αυτό βούλιαζε.[13]
2. Πολλές από τις γυναίκες που πέθαναν βρισκόταν στο steerage.[14]
3. Οι άντρες ήταν περισσότεροι από τις γυναίκες στις σωστικές λέμβους του starboard.[15]

Ταυτόχρονα, ο τρόπος με τον οποίο γέμισε με νερό το πλοίο και άρα ο τρόπος με τον

[12] http://nbviewer.ipython.org/gist/mwaskom/8224591 - Plot [20]
[13] http://www.encyclopedia-titanica.org/csi-titanic-who-died-how.html - Location of Bodies
[14] http://www.icyousee.org/titanic.html - Women and Children First? [8]
[15] http://www.icyousee.org/titanic.html - Women and Children First? [2]

οποίο βυθίστηκε, σε σχέση με την θέση των επιβατών μπορεί επίσης να επηρέασε την πιθανότητα επιβίωσης (πρώτα γέμισε νερό η πλώρη). Παρ' όλ' αυτά βρήκαμε ότι τα καταστρώματα είναι γενικώς οριζόντια. Βλέποντας το πλοίο σε πλάγια όψη:



Εικόνα 8 – Πλάγια όψη του Τιτανικού με τα γράμματα των καταστρωμάτων

Έτσι η πληροφορία για την θέση ενός επιβάτη από το κατάστρωμα στο οποίο αυτός διέμενε, δεν είναι όσο ακριβείς απαιτούν τα παραπάνω ευρήματα. Ακόμη η ανάλυση κατά την οποία βρίσκουμε μέσω του Pclass σε ποια καταστρώματα βρίσκονται οι επιβάτες έχει αρκετές ανακρίβειες στατιστικά (πολύ λίγα δείγματα για γενίκευση, τα δείγματα δεν έχουν ξεκάθαρη κατανομή σε σχέση με τις κλάσεις – Διάγραμμα 15). Οπότε τελικά αποφασίσαμε ότι η θέση ενός επιβάτη στο πλοίο έχει ένα ενδεχόμενο ενδιαφέρον, αλλά είναι απίθανο να προκύψει από τα καταστρώματα.

Για τον λόγο αυτό αρκεστήκαμε στο να μετατρέψουμε τις υπάρχουσες καμπίνες σε καταστρώματα μέσω ενός απλού ruby script (decks.rb στα αρχεία της εργασίας). Έτσι δημιουργήθηκε ένα καινούργιο χαρακτηριστικό στην θέση του Cabin, το χαρακτηριστικό Deck. Το σετ δεδομένων πλέον έχει την παρακάτω μορφή:

	A	B	C	D	E	F	G	H	I	J	K	L
1	Pclass	Type	SibSp	Parch	Fare	Deck	Embarked	Survived				
2	THREE	adult-mal	less	0	7.25	C	S	FALSE				
3	ONE	female	less	0	71.2833		C	TRUE				
4	THREE	female	less	0	7.925	C	S	TRUE				
5	ONE	female	less	0	26.55		S	TRUE				
6	THREE	adult-mal	less	0	8.05		S	FALSE				

Εικόνα 9 – Μετατροπή στήλης Cabins σε Deck

Συνοψίζοντας την παραπάνω ανάλυση λοιπόν, δημιουργήθηκαν σετ δεδομένων τα οποία περιστρέφονταν γύρω από τα παρακάτω χαρακτηριστικά με την σειρά που περιγράφηκαν:

1. Type
2. Company
3. Sibsp (nominal)

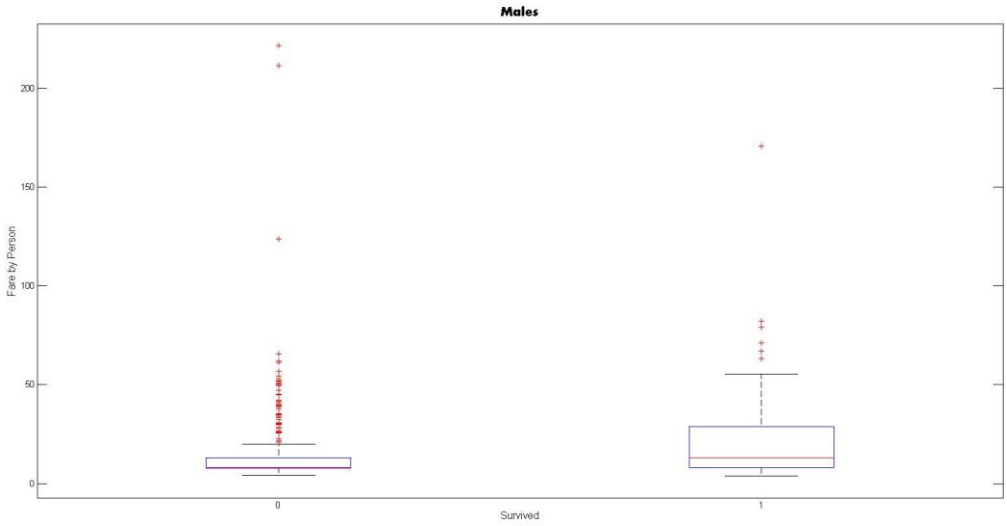
4. Deck

E. Fare και Embarked

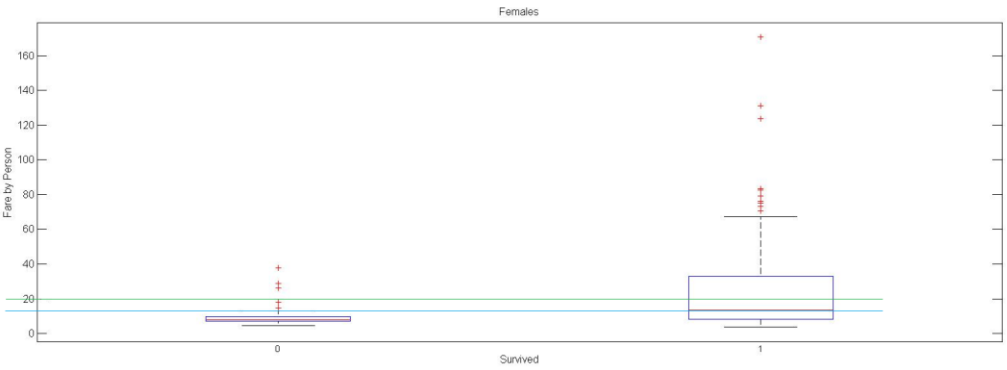
Υπάρχουν όμως και δύο χαρακτηριστικά τα οποία δεν έχουμε αξιοποιήσει πλήρως. Αυτά είναι το Fare και το Embarked.

Για το Embarked δεν μπορέσαμε να βρούμε ξεκάθαρη ωφέλιμη πληροφορία και δεδομένου ότι ήταν ήδη σε nominal μορφή, αποφασίσαμε να το αξιοποιήσουμε με την λογική του brute force training.

Για το Fare από την άλλη διαπιστώσαμε ότι ένα σημαντικό κομμάτι της πληροφορίας που μας δίνει συμπεριλαμβάνεται ήδη στο Pclass. Προσπαθώντας λοιπόν να πάρουμε κάποια πιο ειδική πληροφορία από το χαρακτηριστικό αυτό, αναπαραστήσαμε στην MATLAB, σε box plot, το Fare by person για τα Males και Females:



Διάγραμμα 16 - Box Plot του Fare (by person) των Males, για Survived=0,1

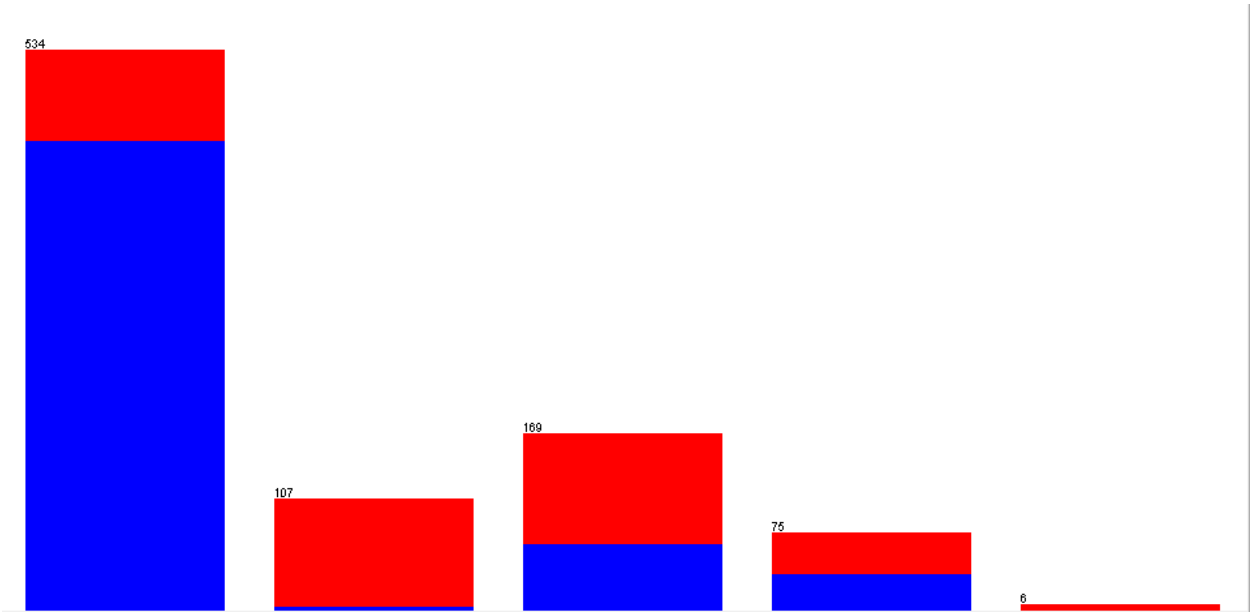


Διάγραμμα 17 - Box Plot του Fare (by person) των Female, για Survived=0,1

Στο πρώτο διάγραμμα παρατηρούμε ότι η κατανομή του ναύλου για τους άντρες δεν μας δίνει ιδιαίτερη πληροφορία για την θνησιμότητά τους. Για παράδειγμα φαίνεται ότι το 90% των αντρών που απεβίωσαν είχαν πληρώσει το πολύ τόσο, όσο το 75% αυτών που επιβίωσαν. Επίσης στην κλάση αυτών που απεβίωσαν υπάρχουν πάρα πολλά outliers μέχρι το άκρο 90% αυτών που που επιβίωσαν. Άρα σαφές συμπέρασμα για τους άντρες δεν μπορεί να βγει.

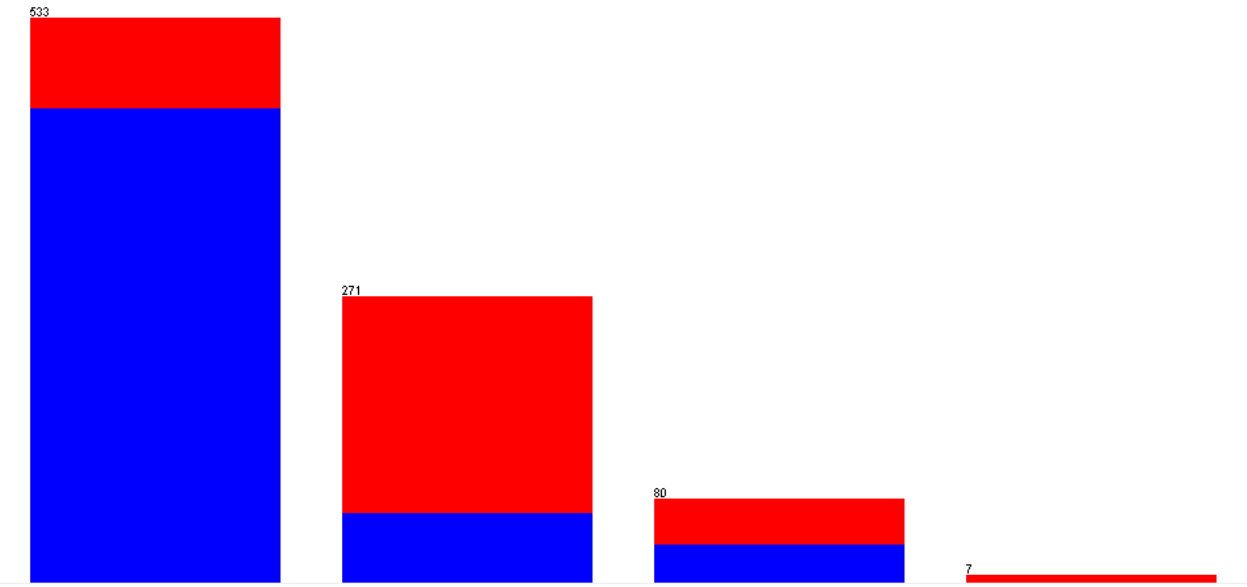
Στο δεύτερο διάγραμμα από την άλλη παρατηρούμε στο αριστερό box ότι πάνω από το 90% των γυναικών πλήρωσε εισιτήριο κάτω από 13 pounds και απεβίωσε (γαλάζια γραμμή). Θέλοντας να εξακριβώσουμε την παρατήρηση αυτή, ανατρέξαμε στο Excel και κάναμε ταξινόμηση και καταμέτρηση των γυναικών με εισιτήριο πάνω και κάτω από 13. Έτσι διαπιστώσαμε ότι από τις 124 γυναίκες με εισιτήριο πάνω από 13, μόνο οι 7 απεβίωσαν. Άρα γι' αυτές τις 124 «πλουσιότερες» γυναίκες, η πιθανότητα μία να επιβιώσει ήταν περίπου 94%. Μάλιστα ύστερα από μερικούς πειραματισμούς και χωρίζοντας τις γυναίκες στο fare 20 αντί για 13 (πράσινη γραμμή) είχαμε βελτίωση της πιθανότητας επιβίωσης κατά 2% δηλαδή 96%. Αντιθέτως, αν μια γυναίκα είχε εισιτήριο κάτω από 20, η πιθανότητα να επιβιώσει ήταν 131/208 = 63%.

Έτσι λοιπόν στο χαρακτηριστικό Type, χωρίσαμε τις γυναίκες σε females και σε rich-females με βάση την τιμή του εισιτηρίου (20) και δημιουργήσαμε το χαρακτηριστικό Type2. Να σημειωθεί ότι, τα female παιδιά που είχαν εισιτήριο πάνω από 20 επίσης αλλάχθηκαν σε rich-female (ο συνδυασμός χαρακτηριστικών rich-female είναι πιο ισχυρός από το χαρακτηριστικό child). Ενδιαφέρον εδώ έχει να δούμε την πιθανότητα επιβίωσης των τιμών του χαρακτηριστικού Type2:



Διάγραμμα 18 – Type2 (Adult male, rich-female, female, child, newborn) – PassengerId

το οποίο συγκρίνουμε με το αντίστοιχο διάγγραμμα 6 για το χαρακτηριστικό Type το οποίο παραθέτουμε ξανά εδώ:



Διάγραμμα 6 – Type (Adult male, female, child, newborn) – PassengerId

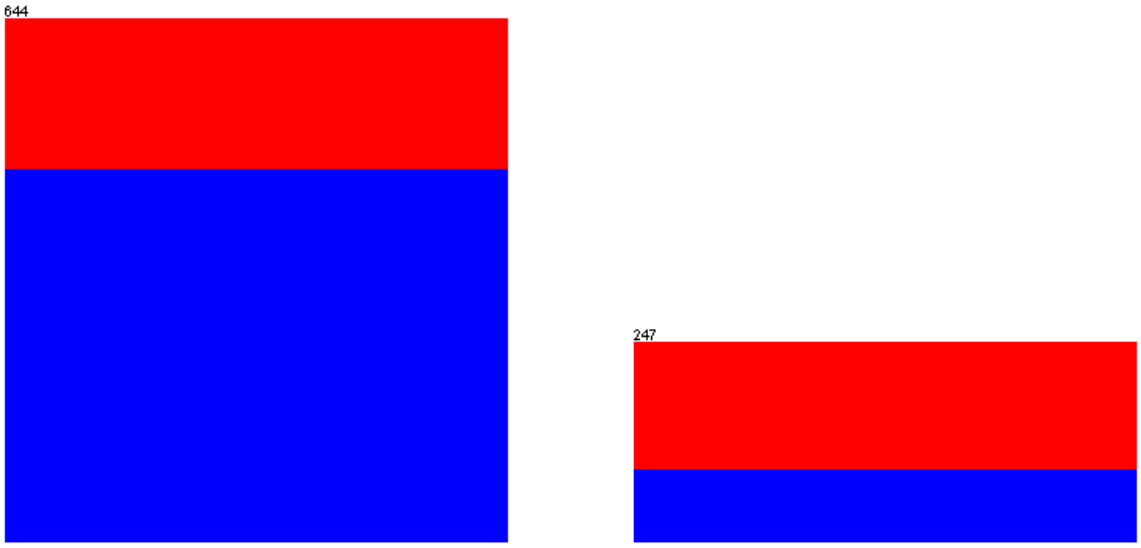
Ουσιαστικά βλέπουμε ότι έχουμε θυσιάσει κάποιο ποσοστό επιβίωσης από την τιμή female του Type για να δημιουργήσουμε μία νέα πολύ πολωμένη τιμή (rich-female) στο Type2. Το αν άξιζε αυτό το trade-off θα φανεί στο στάδιο των πειραμάτων.

Το σετ δεδομένων πλέον είναι κάπως έτσι:

	A	B	C	D	E	F	G	H	I	J
1	Pclass	Type	SibSp	Parch	Embarked	Survived				
2	THREE	adult-mal	1	0	S	FALSE				
3	ONE	rich-fema	1	0	C	TRUE				
4	THREE	female	0	0	S	TRUE				
5	ONE	rich-fema	1	0	S	TRUE				
6	THREE	adult-mal	0	0	S	FALSE				

Εικόνα 10 – Προσθήκη rich-females στο Type

Σαν μία τελευταία προσέγγιση δοκιμάσαμε να χρησιμοποιήσουμε το Fare και ως nominal μεταβλητή. Αυτό το κάναμε μέσω του φίλτρου Discretize στο Weka, το οποίο χώρισε το Fare σε 2 bins, πάνω και κάτω από 15.65. Σύμφωνα με το αντίστοιχο διάγραμμα:



Διάγραμμα 19 – Fare (by person) – PassengerId

φαίνεται ότι ο διαχωρισμός στο 15.65 είναι ικανοποιητικός και αποφασίσαμε να τον κρατήσουμε. Το dataset με αυτή την παραλλαγή απέκτησε την μορφή:

	A	B	C	D	E	F	G	H	I	J	K	L
1	Pclass	Sex	Age	SibSp	Parch	Fare	Embarked	Survived				
2	THREE	male	22	1	0	'\'(-inf-15.	S	FALSE				
3	ONE	female	38	1	0	'\'(15.65-i	C	TRUE				
4	THREE	female	26	0	0	'\'(-inf-15.	S	TRUE				
5	ONE	female	35	1	0	'\'(15.65-i	S	TRUE				
6	THREE	male	35	0	0	'\'(-inf-15.	S	FALSE				

Εικόνα 11 – Διαχωρισμός του Fare σε bins

Συνοψίζοντας για το Fare (by person), το χρησιμοποιήσαμε με τους εξής τρεις τρόπους:

1. Με brute force για την numerical μορφή του
2. Ενσωματωμένο στο Type2 για τα rich-females
3. Σε bins με φίλτρο από το Weka

Περί brute forcing:

- Για το χαρακτηριστικά Embarked και Parch δεν βρέθηκε κάτι χρήσιμο όπως αναφέρθηκε και άρα χρησιμοποιήθηκαν σε brute forcing.
- Το χαρακτηριστικό Fare είχε ενδιαφέρον οπότε το χρησιμοποιήσαμε και αυτό σε brute forcing (πέρα από τις αναλύσεις που κάναμε σχετικά με αυτό).
- Το χαρακτηριστικό Sibsp χρησιμοποιήθηκε σε brute force όταν ήταν σε numerical μορφή. Ακόμη συμφωνήσαμε ότι το Sibsp και το Parch δίνουν ένα είδος πληροφορίας και άρα αποτελούν ένα συνδυασμό brute forcing (όταν και τα 2 είναι numerical).

Τα σετ δεδομένων που προέκυψαν (42) βρίσκονται στους φακέλους της εργασίας 0.Datasets, 1.Datasets-type2, 2.Datasets-binfbp.

0.Datasets:

Περιλαμβάνει όλα τα σετ δεδομένων με βάση την αρχική ανάλυση.

- Brute Force χαρακτηριστικά: Fare (numerical), Embarked (nominal), SibSp (numerical), Parch (numerical)

1.Datasets-type2:

Περιλαμβάνει όλα τα σετ δεδομένων από το 0.Datasets, τα οποία δεν είχαν το Fare. Το χαρακτηριστικό Fare έχει ενσωματωθεί στο Type2 (rich females).

- Brute Force χαρακτηριστικά: Embarked (nominal), SibpSp (numerical), Parch (numerical)

2.Datasets-binfp

Περιλαμβάνει όλα τα σετ δεδομένων από το 0.Datasets, τα οποία είχαν το Fare. Το χαρακτηριστικό Fare είναι σε nominal μορφή (bins).

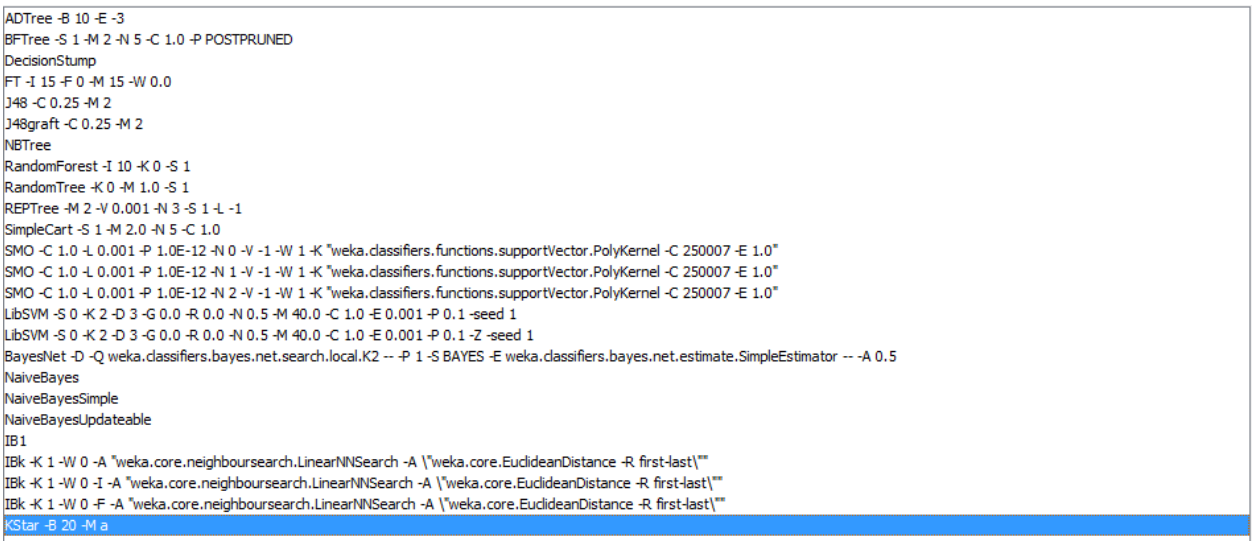
- Brute Force χαρακτηριστικά: Embarked (nominal), SibpSp (numerical), Parch (numerical)

2. Πειράματα

Για τα πειράματα αρχικά αποφασίσαμε να εξετάσουμε τις εξής 4 κατηγορίες αλγορίθμων:

1. Δέντρα
2. Πιθανοτικούς
3. Support Vector Machines
4. NNs

Ύστερα από πολλές δοκιμές αλγορίθμων στον Experimenter του Weka καταλήξαμε στην εξής λίστα αλγορίθμων:



Εικόνα 12 – Αλγόριθμοι δοκιμών

Τους αλγόριθμους αυτούς τους τρέξαμε και στις 3 ομάδες σετ δεδομένων (0,1,2) με την μέθοδο εκπαίδευσης cross-validation για 10 folds. Επίσης επιλέξαμε 10 iterations.

Να αναφερθεί σε αυτό το σημείο ότι κανονικοποίηση των numerical attributes (η οποία έχει νόημα σε classifiers τύπου KNN, SVM) έγινε στους αλγορίθμους μέσω του αντίστοιχου parameter που προσφέρει το Weka. Στους συγκεκριμένους αλγορίθμους δοκιμάσαμε και να αφήσουμε τα αριθμητικά δεδομένα στην αρχική μορφή τους, ώστε να δούμε την συμπεριφορά των classifiers και στις δύο περιπτώσεις.

Επίσης, δεν πραγματοποιήθηκε κανενός άλλου είδους μετασχηματισμός στα numerical attributes των datasets (π.χ. $log(Fare)$), επειδή δεν θεωρήθηκε ότι υπάρχει κάποιος σοβαρός λόγος (π.χ. υπερβολικά μεγάλη διακύμανση των τιμών), ώστε να αλλάξει τόσο ριζικά η δομή των εν λόγω χαρακτηριστικών.

Τέλος, πειραματιστήκαμε με τους διάφορους τρόπους ανάθεσης βαρών στον KNN, όπως και με τον αριθμό των neighbors, πάλι μέσω των παραμέτρων.

Τα αποτελέσματα συνοψίζονται στα csv αρχεία μέσα στον φάκελο Results (Το αρχικό dataset –Εικόνα 4– είναι το πρώτο της λίστας με τα σετ δεδομένων). Το αρχείο με σημασία είναι το accuracy.csv βάσει του οποίου βρήκαμε τους καλύτερους συνδυασμούς αλγορίθμων-σετ δεδομένων:

Δέντρα

Algorithm	Accuracy	Dataset Folder	Dataset
ADTree	83.37%	2.Datasets-binfp	4.3-type-binfbp-sibsp-noparch.csv
ADTree	83.35%	0.Datasets	4.5-type-sibsp-nofbp-noparch.csv
ADTree	83.23%	1.Datasets-type2	4.4-type2-sibsp-noparch.csv
BFTree	83.1%	0.Datasets	4.1-type-sibsp-parch-nofbp.csv

SVM

Algorithm	Accuracy	Dataset Folder	Dataset
LibSVM	83.07%	0.Datasets	2.4-type-nofbp.csv
SMO	82.22%	0.Datasets	2.4-type-nofbp.csv

Πιθανοτικοί

Algorithm	Accuracy	Dataset Folder	Dataset
NaiveBayesSimple	80.76%	0.Datasets-type2	4.3-type-sibsp-parch-nofb.csv
BayesNet	80.7%	0.Datasets-type2	4.7-type-sibsp-nofbp-noem.csv

K-NN

Algorithm	Accuracy	Dataset Folder	Dataset
Ibk (k=8)	83.85%	0.Datasets	4.2-type-fbp-sibsp-noemba.csv
Kstar	82.68%	0.Datasets	2.10-type-nofbp-noparch.csv

Αξιολόγηση και συμπεράσματα

Α. Αξιολόγηση αλγορίθμων συνολικά

Αφού βρήκαμε για κάθε μία από τις 4 τεχνικές ταξινόμησης τους 2 καλύτερους αλγορίθμους, με κριτήριο αξιολόγησης αποκλειστικά το Accuracy, συνεχίζουμε την διερεύνησή μας, αυτή την φορά με 3 επιπλέον μετρικές, τα F-measure, recall και precision.

Σημειώνουμε ότι οι τρεις τελευταίες μετρικές δεν αποτελούν "συνολική" αξιολόγηση του αλγορίθμου (δηλαδή και για τις 2 κλάσεις Survived=TRUE, FALSE) όπως το Accuracy, αλλά σχετίζονται με την ικανότητα του αλγορίθμου να προβλέπει ικανοποιητικά μία συγκεκριμένη κλάση. Έτσι έχουμε π.χ. F-measure και για τις δύο δυνατές τιμές του Survived. Φυσικά, είναι δυνατό να πάρουμε την (στατιστική) μέση τιμή αυτών των μετρικών ως προς τις δύο κλάσεις και τελικά με αυτήν ασχοληθήκαμε.

Για μια κλάση +:

- recall(+) = Pr(εκτιμήθηκε+|δόθηκε+)
- precision(+) = Pr(δόθηκε+|εκτιμήθηκε+)

F-measure(+) = αρμονικός μέσος των recall(+), precision(+). Ο αρμονικός μέσος 2 τιμών είναι, όπως δηλώνει το όνομά του, μία μορφή μέσης τιμής, όπως ακριβώς είναι και ο αριθμητικός μέσος. Σε αντίθεση όμως με τον αριθμητικό μέσο, ο αρμονικός μέσος είναι πολωμένος προς την μικρότερη τιμή (όπως ακριβώς και η συνολική αντίσταση 2 παραλλήλων αντιστάσεων). Έτσι, το F-measure(+) θα είναι ένας αριθμός ανάμεσα στους recall(+) και precision(+), και θα βρίσκεται πιο κοντά στον μικρότερο.

Η ιδιότητα του F-measure να δίνει μια worst-case μέση τιμή τον καθιστά ιδανικό για την συνοπτική εξέταση της συμπεριφοράς ενός αλγορίθμου, χωρίς να χρειάζεται να εξετάζουμε 2 τιμές (recall/precision).

	Accuracy	F-measure	Max F-measure
Ibk (k=8)	83.85%	0.84	0.84
ADTree	83.37%	0.83	0.83
BFTree	83.1%	0.82	0.83
LibSVM	83.07%	0.82	0.82
Kstar	82.68%	0.82	0.82
SMO	82.22%	0.83	0.83
NaiveBayesSimple	80.76%	0.8	0.8
BayesNet	80.7%	0.8	0.8

Στον παραπάνω πίνακα παρουσιάζονται οι 8 καλύτεροι classifiers, ταξινομημένοι ως προς την μετρική accuracy, κατά φθίνουσα σειρά. Η στήλη F-measure δείχνει το F-measure ενός αλγορίθμου για το αναδεικνυόμενο accuracy. Η Max F-measure από την άλλη δείχνει την συνολικά μέγιστη για τον αλγόριθμο, για όλα τα dataset. Επίσης να σημειωθεί ότι τα F-measure αυτά είναι weighted. Από την σύγκριση των δύο στηλών F-measure φαίνεται ότι το dataset που πέτυχε το καλύτερο accuracy για κάποιον αλγόριθμο, πέτυχε και το καλύτερο F-measure,εκτός από την περίπτωση του BFTree, η οποία όμως είναι μοναδική και αμελητέα. Γενικώς δηλαδή ισχύει ο κανόνας «Μεγαλύτερο accuracy, μεγαλύτερο F-measure». Το τελικό συμπέρασμα λοιπόν είναι ότι ο Ibk (k=8) ήταν ο καλύτερος αλγόριθμος, βασιζόμενοι μόνο σε αυτές τις δύο μετρικές αξιολόγησης. Οι πιθανοτικοί αλγόριθμοι είχαν τις χειρότερες επιδόσεις, ενώ τα Trees και SVM είχαν συγκρίσιμες επιδόσεις ως προς το F-measure, με τα SVM να υπερτερούν ως προς το Accuracy.

Ας εξετάσουμε τώρα τις μετρικές recall και precision:

Ibk(k=8)

	Recall	Precision
Dead	0.905	0.837
Survived	0.716	0.825
Weighted	0.84	0.84

ADTree

	Recall	Precision
Dead	0.94	0.813
Survived	0.652	0.871
Weighted	0.81	0.81

BFTree

	Recall	Precision
Dead	0.938	0.812
Survived	0.652	0.868
Weighted	0.82	0.83

LibSVM

	Recall	Precision
Dead	0.889	0.843
Survived	0.734	0.804
Weighted	0.83	0.83

Kstar

	Recall	Precision
Dead	0.942	0.804
Survived	0.632	0.871
Weighted	0.83	0.84

SMO

	Recall	Precision
Dead	0.876	0.841
Survived	0.734	0.787

Weighted	0.82	0.82

NaiveBayesSimple

	Recall	Precision
Dead	0.898	0.814
Survived	0.67	0.804
Weighted	0.81	0.81

BayesNet

	Recall	Precision
Dead	0.891	0.814
Survived	0.673	0.793
Weighted	0.81	0.81

Σύμφωνα με τους παραπάνω πίνακες, παρατηρούμε ότι το precision γενικώς δεν έχει μεγάλες αποκλίσεις από το weighted average του. Παρατηρούμε απλώς ότι μερικές φορές μεγαλύτερο accuracy δεν σημαίνει μεγαλύτερο weighted average precision.

Για το recall από την άλλη όμως παρατηρούμε ότι σε αρκετές περιπτώσεις είναι σημαντικά μικρότερο για τους Survived από τους μη. Θέλοντας να εξετάσουμε αυτό το φαινόμενο περαιτέρω λάβαμε υπ' όψη την υποσημείωση της εργασίας, η οποία κάνει νύξη για την μετρική κόστους σε περίπτωση που υπάρχει class imbalance.

Class imbalance υπάρχει όταν το πλήθος των εγγραφών μιας κλάσης είναι συγκριτικά πολύ μεγαλύτερο από το πλήθος των εγγραφών της άλλης (για πρόβλημα 2 κλάσεων). Εξετάζοντας αρχικά τα δεδομένα μας, **θεωρήσαμε ότι *δεν* υπάρχει class imbalance**: 549 νεκροί, 342 επιζώντες είναι μια σχετικά ισορροπημένη κατανομή. Αποφασίσαμε όμως να υποθέσουμε ότι υπάρχει θέμα class imbalance και μάλιστα κατά

$$\frac{549}{342} = 1.605$$

Δηλαδή περίπου 60%. Με την παραδοχή αυτή θα θέλαμε να κάνουμε τα classes balanced. Για να είναι balanced τα classes εισάγουμε την μετρική κόστους. Σύμφωνα με αυτήν, κόστος έχουμε όταν γίνεται λάθος πρόβλεψη. Δηλαδή εξ' ορισμού ο πίνακας κόστους για κάθε αλγόριθμο είναι ο εξής:

0	1
1	0

και κατ' αυτόν η μετρική κόστους απαντάει στο ερώτημα «Πόσο κοστίζει μία λάθος πρόβλεψη;». Αν όμως θεωρήσουμε ότι τα classes είναι imbalanced, τότε βλέπουμε ότι το να κάνει ο αλγόριθμος λάθος πρόβλεψη «κοστίζει» το ίδιο για κάθε κλάση. Αυτό σημαίνει όμως ότι τα λάθη της κλάσης με τις λιγότερες εγγραφές (εδώ Survived) θα ήταν πιο

«φτηνά», αφού η κλάση έχει συνολικά λιγότερες τιμές από την άλλη. Η συνολική συνεισφορά των λαθών δηλαδή για την κλάση με τις λιγότερες εγγραφές, θα είχε μικρότερη βαρύτητα στην εξίσωση κόστους από την συνεισφορά της άλλης.

Σκοπός λοιπόν είναι τα λάθη κάθε κλάσης, να συνεισφέρουν το ίδιο στο κόστος. Δεδομένου ότι στο πρόβλημά μας οι νεκροί επιβάτες είναι κατά 60% περισσότεροι από τους επιζώντες, πολλαπλασιάσαμε την 2η γραμμή στο cost matrix με τον συντελεστή 1.6. Αυτό έγινε ώστε στην εξίσωση

$$Cost = q(a + d) + p(b + c)$$

στο δεύτερο σκέλος που είναι τα λάθη, το c να πολλαπλασιαστεί με 1.6 και άρα

$$d + 1.6 * c = a + b = 549$$

Με τον τρόπο αυτό η λάθος πρόβλεψη για survived, θα κοστίζει όσο η λάθος πρόβλεψη για died. Το cost matrix πλέον είναι έτσι:

0	1
1.6	0

Χρησιμοποιώντας λοιπόν το αρχικό cost matrix και το καινούργιο, κάναμε cost sensitive evaluation κατά το οποίο δημιουργήθηκε ο παρακάτω πίνακας:

	Accuracy	Cost	Class-Balanced Cost
ADTree	83.37	155	211.4
BFTree	83.1	153	224.4
LibSVM	83.07	152	206.6
SMO	82.22	159	213.6
NaiveBayesSimple	80.76	169	236.8
BayesNet	80.7	172	239.2
Ibk	83.85	149	207.2
Kstar	82.68	158	233.6

Επίσης τα αντίστοιχα confusion matrix των μοντέλων είναι τα εξής:

ADTree

	Predicted = Died	Predicted = Survived
Actual = Died	516	33
Actual = Survived	119	223

BFTree

	Predicted = Died	Predicted = Survived
Actual = Died	515	34
Actual = Survived	119	223

LibSVM

	Predicted = Died	Predicted = Survived
Actual = Died	488	61
Actual = Survived	91	251

SMO

	Predicted = Died	Predicted = Survived
Actual = Died	481	68
Actual = Survived	91	251

NaiveBayesSimple

	Predicted = Died	Predicted = Survived
Actual = Died	493	56
Actual = Survived	113	229

BayesNet

	Predicted = Died	Predicted = Survived
Actual = Died	489	60
Actual = Survived	112	230

Ibk

	Predicted = Died	Predicted = Survived
Actual = Died	497	52
Actual = Survived	97	245

Kstar

	Predicted = Died	Predicted = Survived
Actual = Died	517	32
Actual = Survived	126	216

Στο σημείο αυτό θέλουμε να αξιολογήσουμε τους αλγορίθμους με βάση τα λάθη που πραγματοποίησαν. Αρχικά παρατηρούμε ότι ο Ibk, που αναδείχθηκε ως ο καλύτερος στην ανάλυση Accuracy – F-Measure, έχει και το μικρότερο κόστος πριν και μετά το balancing. Το ίδιο ισχύει και για τους δύο χειρότερους αλγορίθμους, τους Πιθανοτικούς (έχουν το μεγαλύτερο κόστος πριν και μετά).

Ενδιαφέρον εδώ έχει να εξετάσουμε τα ζευγάρια αλγορίθμων, στα οποία ο ένας έχει μεγαλύτερο accuracy αλλά και μεγαλύτερο class-balanced κόστος από τον άλλον. Συγκεκριμένα πρόκειται για τα ζευγάρια

- BFTree – LibSVM
- KStar - SMO

Στα δύο αυτά ζευγάρια ο BFTree και ο KStar έχουν μεγαλύτερο accuracy από τους LibSVM και SMO αντίστοιχα. Οι τελευταίοι όμως έχουν μικρότερο κόστος. Αν εξετάσουμε την απόλυτη διαφορά κόστους για κάθε ζευγάρι πριν και μετά το class balancing έχουμε το εξής αποτέλεσμα:

	Διαφορά κόστους πριν	Διαφορά κόστους μετά
BFTree - LibSVM	1	17.8
KStar - SMO	1	20

Εξετάζοντας και τα αντίστοιχα confusion matrix βλέπουμε ότι οι αλγόριθμοι με μικρότερο κόστος βρήκαν περισσότερους ζωντανούς και λιγότερους νεκρούς από αυτούς με μεγαλύτερο κόστος. Αυτό «αναδείχτηκε» από το νέο Cost Matrix και τις νέες διαφορές κόστους. Συνεπώς αν δεχτούμε ότι υπάρχει θέμα class imbalance, και άρα ότι θέλουμε τα λάθη να κοστίζουν το ίδιο, τότε ο SMO και ο LibSVM είναι καλύτεροι από τους άλλους δύο. Η διαφορά στο accuracy δηλαδή χάνει βαρύτητα, μιας και είναι και τόσο μικρή, όταν εμπλακεί και η ανάλυση κόστους (κατά την κρίση μας). Τέλος ο BFTree έχει και μικρότερο κόστος και μεγαλύτερο accuracy από τον KStar, όπως και ο LibSVM από τον SMO.

Έτσι τελική η κατάταξη αλγορίθμων σε φθίνουσα σειρά (καλύτερο – χειρότερο) έχει ως εξής:

Ibk για 8 neighbors
ADTree
LibSVM
SMO
BFTree
KStar
NaiveBayesSimple
BayesNet

Το τελευταίο στάδιο για την συνολική αξιολόγηση των αλγορίθμων, ήταν να δημιουργήσουμε μετα-μοντέλα με βάση τον πίνακα κόστους που ισορροπεί τα classes (δηλαδή να κάνουμε cost prediction). Έτσι τρέξαμε τον αλγόριθμο MetaCost στο Weka και του δώσαμε ως παραμέτρους τον πίνακα κόστους και τον εκάστοτε αλγόριθμο προς αξιολόγηση. Τα αποτελέσματα που προέκυψαν έχουν ως εξής:

	Accuracy	Precision	Recall	F-Measure
Ibk	81.93	0.819	0.819	0.819
ADTree	78.67	0.804	0.787	0.789
LibSVM	82.49	0.824	0.825	0.823
SMO	81.59	0.815	0.816	0.815
BFTree	80.35	0.803	0.804	0.803
KStar	80.58	0.804	0.806	0.804
NaiveBayesSimple	78.78	0.787	0.788	0.787
BayesNet	79	0.789	0.79	0.789

Αρχικά παρατηρούμε ότι όλες οι μετρικές έπεσαν (σε σύγκριση με τα απλά, αρχικά μοντέλα). Συνεπώς η τελική κατάταξη παραμένει. Αν θέλαμε να σχολιάσουμε τα παραπάνω αποτελέσματα, βλέπουμε ότι όλες οι μετρικές είναι πολύ κοντά, το οποίο υποδεικνύει ισορροπία. Επίσης οι αλγόριθμοι οι οποίοι έβρισκαν πολύ περισσότερους νεκρούς από ζωντανούς (ADTree, BFTree, Kstar) έχασαν πάνω από 3% accuracy στο Meta Classification, το οποίο είναι λογικό.

Εν τέλει το καλύτερο μοντέλο είναι το απλό του Ibk για 8 Neighbors.

Β. Αξιολόγηση αλγορίθμων ως προς μία κλάση

Στο κομμάτι αυτό θέλουμε να δοθεί βαρύτητα στην αναγνώριση των ατόμων που είναι πιθανόν να σωθούν. Συνεπώς εδώ κρίναμε ότι μας ενδιαφέρουν μετρικές που προσφέρουν καλύτερη αξιολόγηση ανά κλάση (αντί συνολικά). Δηλαδή στην περίπτωσή μας οι μετρικές Recall και Precision και F-measure ανά κλάση. Συγκεκριμένα λοιπόν θα αξιολογήσουμε τους αλγορίθμους ως προς:

1. Την ικανότητά τους να εντοπίσουν έναν επιβάτη που έζησε, δεδομένου ότι αυτός όντως έζησε – Recall
2. Την αξιοπιστία τους ως προς το πόσο πιθανόν είναι να έζησε ένα άτομο, όταν ο αλγόριθμος προβλέπει ότι έζησε – Precision
3. Αρμονικός μέσος των δύο παραπάνω - F-Measure

Ανατρέχοντας στους πίνακες recall και precision, F-Measure (για Class = Survived) προκύπτουν οι εξής ταξινομημένοι πίνακες για τις τρεις μετρικές:

	Precision
ADTree	0.871
KStar	0.871
BFTree	0.868
Ibk	0.825
LibSVM	0.804
NaiveBayesSimple	0.804
BayesNet	0.793
SMO	0.787

	Recall
LibSVM	0.734
SMO	0.734
Ibk	0.716
BayesNet	0.673
NaiveBayesSimple	0.67
ADTree	0.652
BFTree	0.652
KStar	0.632

	F-Measure
LibSVM	0.768
Ibk	0.767
ADTree	0.762
SMO	0.759
BFTree	0.745
KStar	0.732
NaiveBayesSimple	0.73
BayesNet	0.728

Αν θέλαμε να θεωρήσουμε μία μετρική πιο σημαντική από την άλλη, μεταξύ precision και recall, αυτό κατ' αρχήν εξαρτάται από την εφαρμογή.

Παραδείγματος χάρη, ένα τραπεζικό σύστημα ενδιαφέρεται να εντοπίζει transactions τα οποία φαίνονται πλαστά. Για το σκοπό αυτό θα χρησιμοποιήσει αναγνώριση προτύπων. Σε τι βαθμό όμως επιθυμεί το τραπεζικό σύστημα να εντοπίζει ένα πλαστό transaction είναι υποκειμενικό. Μπορεί να επιθυμεί να εντοπίζονται όλα τα transactions τα οποία μοιάζουν έστω και λίγο με πλαστά. Να αυξήσει δηλαδή την ευαισθησία του αλγορίθμου ως προς το Class = FakeTransaction (μεγάλο recall).

Από την άλλη, ένα ιατρικό σύστημα που θα εντόπιζε τον καρκίνο, θα θέλαμε να έχει πολύ υψηλό precision. Να είμαστε σίγουροι δηλαδή ότι η πρόβλεψη είναι «αληθινή», και άρα να προχωρήσουμε σε θεραπείες. Δεν θέλουμε να βρίσκουμε ασθενείς με καρκίνο, ενώ δεν έχουν καρκίνο. Το precision από την άλλη είναι κάτι που δεν θα ενδιέφερε το τραπεζικό σύστημα τόσο πολύ. Δεν τον νοιάζει αν μία πρόβλεψη για ένα FakeTransaction είναι false alarm, αυτό ενδιαφέρεται να βρίσκει κάθε **πιθανά** (έστω και λίγο) FakeTransaction για να μην προκληθεί οικονομική ζημιά.

Προφανώς στο ιατρικό σύστημα θα μας ένοιαζε επίσης και πάρα πολύ υψηλό accuracy, και άρα και recall, απλώς χρησιμοποιήθηκε ως παράδειγμα για την σύγκριση των μετρικών.

Στο δικό μας πρόβλημα τώρα μιας και δεν «προτιμάμε» κάποια μετρική, επιλέγουμε αρχικά να αξιολογήσουμε τους αλγορίθμους ως προς το F-Measure, το οποίο δίνει μία μέση πληροφορία και για τις δύο. Αν θέλαμε να επιλέξουμε μία μετρική έναντι άλλης, θα επιλέγαμε το recall. Αυτό με την λογική ότι σε ένα πρόβλημα αναγνώρισης προτύπων, μας ενδιαφέρει ενδεχομένως *λίγο* περισσότερο **για μια κλάση** να **αναγνωρίζεται το πρότυπο** (recall) από το να είναι η πρόβλεψη ακριβής (precision). Η μετρική F-Measure συνδέεται καλά με αυτή την λογική στο πρόβλημά μας μιας και είναι πολωμένη ως προς την μικρότερη τιμή, δηλαδή εδώ το recall. Ταυτόχρονα λαμβάνει υπ' όψη και το precision.

Συνεπώς αξιολογούμε τους αλγόριθμους με βάση το F-Measure και άρα καλύτερο μοντέλο είναι ο LibSVM (που τυχαίνει να έχει και το μεγαλύτερο recall).

Εξετάζουμε τώρα πάλι τις ίδιες μετρικές για το Meta Classification με τον πίνακα κόστους για το Class Imbalance. Τα αποτελέσματα έχουν ώς εξής (Ταξινομημένα, Class = Survived):

	Precision (meta)
LibSVM	0.798
KStar	0.772
SMO	0.771
Ibk(k=8)	0.77
BFTree	0.749
BayesNet	0.738
NaiveBayesSimple	0.731
ADTree	0.682

	Recall (meta)
ADTree	0.833
Ibk(k=8)	0.754
SMO	0.74
BFTree	0.734
LibSVM	0.728
NaiveBayesSimple	0.708
KStar	0.702
BayesNet	0.702

	F-Measure (meta)
Ibk	0.762
LibSVM	0.761
SMO	0.755
ADTree	0.75
BFTree	0.742
KStar	0.735
BayesNet	0.72
NaiveBayesSimple	0.719

Εδώ πάλι με την λογική της προηγούμενης ανάλυσης, ο Ibk είναι ο καλύτερος αλγόριθμος αλλά ο LibSVM παραμένει καλύτερος γενικά στην ανά κλάση αξιολόγηση μιας και έχει

μεγαλύτερο F-Measure. Σαν σχόλιο εδώ να πούμε ότι το Meta-Recall είναι για όλους τους αλγορίθμους καλύτερο. Αυτό είναι απολύτως λογικό μιας και το class το οποίο κάνουμε balance είναι οι Survived, και άρα στα μετα-μοντέλα δίνεται βαρύτητα σε αυτούς. Από την άλλη παρατηρούμε ότι το precision και το F-measure έχουν πέσει και άρα κατά την άποψή μας τα μοντέλα της μετα-ανάλυσης είναι χειρότερα (παρ' όλο το μεγαλύτερο recall).

Να πούμε επίσης ότι, αν θέλαμε να πετύχουμε μεγάλο recall, και άρα μεγάλο TPR (TPR, όπου στο confusion matrix το πρώτο class θα ήταν οι survived) θα κάναμε μία ανάλυση roc. Θα κοιτούσαμε δηλαδή για το δοθέν TPR που μας ενδιέφερε (και άρα δοθέν FPR), το threshold που αντιστοιχεί στην καμπύλη και με αυτό θα υπολογίζαμε τον πίνακα κόστους δίνοντας βαρύτητα στην κλάση survived. Στην συνέχεια θα κάναμε meta-classification και θα πέρναμε το μοντέλο που επιθυμούμε.

Τελικά

Καλύτερο Μοντέλο:

Ibk με 8 neighbors

Καλύτερο Μοντέλο για Class = Survived:

LibSVM