ΤΕΧΝΙΚΕΣ ΜΗΧΑΝΙΚΗΣ ΜΑΘΗΣΗΣ
Διαχωρισμός Αστέρων σε Pulsar

ΜΑΡΙΑ ΒΡΑΝΑ · *mvrana.96@outlook.com*
ΠΡΟΗΓΜΕΝΑ ΣΥΣΤΗΜΑΤΑ ΥΠΟΛΟΓΙΣΤΩΝ ΚΑΙ ΕΠΙΚΟΙΝΩΝΙΩΝ
ΑΡΙΣΤΟΤΕΛΕΙΟ ΠΑΝΕΠΙΣΤΗΜΙΟ ΘΕΣΣΑΛΟΝΙΚΗΣ

Περιεχόμενα

1 Εισαγωγή

1.1 Τι είναι Pulsar;

Τα Pulsars είναι αστέρες νετρονίων με πολύ μεγάλη ταχύτητα περιστροφής γύρω από τον άξονα τους. Αποτελούν μαγνητικά δίπολα με εξαιρετικά ισχυρό μαγνητικό πεδίο και εκπέμπουν ηλεκτρομαγνητική ακτινοβολία σε μεγάλο εύρος συχνοτήτων. Η περιοδικότητα των ηλεκτρομαγνητικών παλμών που παράγουν τα καθιστά ¨κοσμικά ρολόγια¨ ή ¨κοσμικούς φάρους¨ και έτσι αποτελούν σημεία αναφοράς στη διερεύνυση κοντινών αστρικών αντικειμένων (πχ. μέτρηση απόστασης ενός αντικειμένου). Η διαταραχές των παλμικών σημάτων μπορεί να σημάνουν την ύπαρξη κάποιου συμβάντος στο σύμπαν. Επιπλέον, μπορούν να χρησιμοποιηθούν για τη μελέτη της ύλης σε καταστάσεις πολύ μεγάλης πίεσης καθώς είναι εξαιρετικά πυκνά αντικείμενα.

1.2 Το σύνολο δεδομένων

Το σύνολο δεδομένων υπάρχει διαθέσιμο στο Predicting Pulsar Star. Στόχος του αλγορίθμου είναι ο διαχωρισμός των αστέρων (υποψηφίων Pulsar) σε Pulsar (θετική κλάση) ή όχι (αρνητική κλάση). Το πρόβλημα είναι binary classification. Επιπλέον αποτελεί unbalanced πρόβλημα καθώς η θετική κλάση έχει σημαντικά λιγότερα δείγματα (1153 θετικά /11375 αρνητικά δείγματα) . Το σύνολο δεδομένων περιέχει 8 στήλες αριθμητικών δεδομένων (χαρακτηριστικά), 4 στήλες που αφορούν το integrated profile που προκύπτει από τα περιοδικά σήματα - "δακτυλικό αποτύπωμα του Pulsar", 4 στήλες με πληροφορίες για την DM-SNR καμπύλη (διασπορά - signal to noise ratio) και μία binary στήλη που αποτελεί τον στόχο (κλάση).

2 Μοντέλα χωρίς επιλογή χαρακτηριστικών

Αρχικά, επειδή τα χαρακτηριστικά είναι λίγα θα μελετηθούν μοντέλα που χρησιμοποιούν όλα τα χαρακτηριστικά. Σε αυτό το κεφάλαιο θα μελετηθεί πώς επηρεάζεται το accuracy και f1-score ενός μοντέλου σχετικά με τον τρόπο που αντιμετωπίζονται οι κενές τιμές NaN που βρίσκονται στο σύνολο δεδομένων. Θα μελετηθούν 3 μέθοδοι. Κατά τον πρώτο οι κενές τιμές θα αντικατασταθούν με το μέσο της στήλης που βρίσκονται, κατά τον δεύτερο θα αντικατασταθούν με τον median και στον τελευταίο τρόπο θα αφαιρεθούν οι γραμμές που περιέχουν κενές τιμές.

2.1 Αντικατάσταση NaN με mean

Η προεπεξεργασία που εφαρμόστηκε αποτελείται από:

- scaling χαρακτηριστικών (sklearn.preprocessing.MinMaxScaler).
- αντικατάσταση κενών τιμών με το μέσο όρο της στήλης που ανήκουν (sklearn.impute.SimpleImputer)
- διαχωρισμός σε σύνολο εκπαίδευσης (training set, 0.75) και σύνολο αξιολόγησης (test set, 0.25) με stratify split (sklearn.model_selection.train_test_split)
- δημιουργία ξεχωριστού συνόλου εκπαίδευσης με τη μέθοδο SMOTE ώστε να υπάρχει ισορροπία μεταξύ των δειγμάτων των κλάσεων (imblearn.over_sampling.SMOTE).

Μελετήθηκαν οι παρακάτω αλγόριθμοι στο αρχικό σύνολο εκπαίδευσης και στο σύνολο εκπαίδευσης με SMOTE:

- LDA, QDA με τις προεπιλεγμένες τιμές παραμέτρων
- Logistic regression με τις παραμέτρους που προκύπτουν από το grid search για τις παραμέτρους {'penalty': ['l2',None], 'solver': ['lbfgs'], 'C': range(1,11)} και επιπλέον ένα ακόμη grid search για τις ίδιες παραμέτρους με διαφορά το class_weight:'balanced'.
- Decision Tree με τις παραμέτρους που προκύπτουν από το grid search για τις παραμέτρους {'criterion': ('gini','entropy','log_loss'), 'ccp_alpha': ccpp.ccp_alphas}, όπου οι τιμές της παράμετρου ccp_alpha προκύπτουν από τον πίνακα τιμών του cost complexity pruning path. Επιπλέον πραγματοποιήθηκε ένα ακόμη grid search για τις ίδιες παραμέτρους με διαφορά το class_weight:'balanced'.
- SVM με τις παραμέτρους που προκύπτουν από δύο ξεχωριστά grid searches, το πρώτο για τις παραμέτρους {'kernel': ['poly'], 'C': range(1,11), 'degree': range(2,6)} και το δεύτερο για {'kernel': ('linear','sigmoid','rbf'), 'C': range(1,11)}.Ο λόγος που διαχωρίστηκαν τα δύο grid searches είναι επειδή, αν και η παράμετρος degree αγνοείται από τα kernels linear,sigmoid και rbf από το grid search δημιουργούνται μοντέλα για αυτά τα kernels για κάθε βαθμό και αξιολογούνται κανονικά, με αποτέλεμα να χρειάζεται αισθητά περισσότερος χρόνος. Αφού αξιολογηθούν τα μοντέλα από τα δύο grid searches επιλέγονται τα καλύτερα και συγκρίνονται για να βρεθεί τελικά το βέλτιστο. Επιπλέον πραγματοποιήθηκε ένα ακόμη σετ από grid searches για τις ίδιες παραμέτρους με διαφορά το class_weight:'balanced'.

Οι μετρικές αξιολόγησης που μελετώνται είναι το accuracy και f1-score. Η επιλογή των βέλτιστων μοντέλων γίνεται με βάση το f1-score. Στον πινακα 1 φαίνονται τα αποτελέσματα της παραπάνω διαδικασίας. Τα validation scores προκύπτουν από 5 fold cross validation.

Πίνακας 1: Αποτελέσματα για τις επιλεγμένες μετρικές στο validation set και το test set.

method	**best parameters**	**validation accuracy**	**validation f1-score**	**test accuracy**	**test f1-score**
lda	-	0.9692	0.8049	0.9697	0.8141
lda (SMOTE)	-	0.9188	0.9154	0.9508	0.7638
qda	-	0.9616	0.8007	0.9658	0.8208
qda (SMOTE)	-	0.9121	0.9069	0.9633	0.8130
Logistic Regression	{'C': 1, 'penalty': None, 'random_state': 0, 'solver': 'lbfgs'}"	0.9769	0.8640	0.9757	0.8582
Logistic Regression (balanced)	{'C': 1, 'class_weight': 'balanced', 'penalty': None, 'random_state': 0, 'solver': 'lbfgs'}	0.9597	0.8024	0.9595	0.8025
Logistic Regression (SMOTE)	{'C': 1, 'penalty': None, 'random_state': 0, 'solver': 'lbfgs'}	0.9324	0.9300	0.9585	0.7988
Decision Tree	{'ccp_alpha': 0.0013271391766541355, 'criterion': 'entropy', 'random_state': 0}	0.9767	0.8671	0.9783	0.8794
Decision Tree (balanced)	{'ccp_alpha': 0.0, 'class_weight': 'balanced', 'criterion': 'entropy', 'random_state': 0}	0.9669	0.8194	0.9690	0.8336
Decision Tree (SMOTE)	{'ccp_alpha': 0.00019837154991298703, 'criterion': 'entropy', 'random_state': 0}	0.9601	0.9603	0.9515	0.7704
svm	{'C': 3, 'degree': 4, 'kernel': 'poly', 'random_state': 0}	0.9787	0.8752	0.9770	0.8672
svm (balanced)	{'C': 7, 'class_weight': 'balanced', 'degree': 4, 'kernel': 'poly', 'random_state': 0}	0.9713	0.8535	0.9642	0.8217
svm (SMOTE)	{'C': 10, 'degree': 5, 'kernel': 'poly', 'random_state': 0}	0.9497	0.9481	0.9649	0.8203

Καλύτερο accuracy και f1-score παρατηρείται για το decicion tree με υπερπαρμέτρους {'ccp_alpha': 0.0013271391766541355, 'criterion': 'entropy', 'random_state': 0}. Το random_state επιλέχθηκε ίσο με 0 για να είναι τα αποτελέσματα επαναλήψιμα και δεν είναι παράμετρος που προκύπτει από κάποιο grid search. validation και test f1-scores στα lda, qda, logistic regression, decision tree και svm με SMOTE παρατηρείται μεγάλη διαφορά, περίπου 10%-20% και αυτό πρέπει να οφείλεται σε κάποιου είδους υπερεκπαίδευση.

2.2 Αντικατάσταση NaN με median

Σε αυτή την περίπτωση ακολουθήθηκαν ακριβώς τα ίδια βήματα με το υποκεφάλαιο 2.1 με τη διαφορά ότι η αντικατάσταση γίνεται με το median και όχι το mean της στήλης. Στη συνέχεια μελετήθηκαν οι ίδιοι αλγόριθμοι (ίδιο grid search) και τα αποτελέσματα εμφανίζονται στον πίνακα 2.

Πίνακας 2: Αποτελέσματα για τις επιλεγμένες μετρικές στο validation set και το test set.

method	**best parameters**	**validation accuracy**	**validation f1 score**	**test accuracy**	**test f1 score**
lda	-	0.9691	0.8041	0.9697	0.8141
lda (SMOTE)	-	0.9165	0.9131	0.9511	0.7650
qda	-	0.9612	0.8002	0.9662	0.8233
qda (SMOTE)	-	0.9140	0.9093	0.9617	0.8071
Logistic Regression	{'C': 1, 'penalty': None, 'random_state': 0, 'solver': 'lbfgs'}	0.9765	0.8613	0.9751	0.8545
Logistic Regression (balanced)	{'C': 1, 'class_weight': 'balanced', 'penalty': None, 'random_state': 0, 'solver': 'lbfgs'}	0.9570	0.7921	0.9585	0.7981
Logistic Regression (SMOTE)	{'C': 1, 'penalty': None, 'random_state': 0, 'solver': 'lbfgs'}	0.9309	0.9285	0.9598	0.8037
Decision Tree	{'ccp_alpha': 0.0013271391766541355, 'criterion': 'entropy', 'random_state': 0}	0.9765	0.8651	0.9761	0.8687
Decision Tree (balanced)	{'ccp_alpha': 0.00010880548273516918, 'class_weight': 'balanced', 'criterion': 'gini', 'random_state': 0}	0.9658	0.8179	0.9649	0.8191
Decision Tree (SMOTE)	{'ccp_alpha': 0.00016959544338190086, 'criterion': 'entropy', 'random_state': 0}	0.9565	0.9567	0.9492	0.7565
svm	{'C': 7, 'degree': 4, 'kernel': 'poly', 'random_state': 0}	0.9789	0.8764	0.9767	0.8661
svm (balanced)	{'C': 7, 'class_weight': 'balanced', 'degree': 5, 'kernel': 'poly', 'random_state': 0}	0.9724	0.8564	0.9658	0.8249
svm (SMOTE)	{'C': 10, 'degree': 5, 'kernel': 'poly', 'random_state': 0}	0.9496	0.9480	0.9674	0.8328

Τα βέλτιστα μοντέλα που επιλέχθηκαν είναι ίδια με αυτά που επιλέχθηκαν στην πρώτη περίπτωση με μερικές εξαιρέσεις. Εδώ το καλύτερο f1-score παρατηρείται στο decision tree με υπερπαραμέτρους {'ccp_alpha': 0.0013271391766541355, 'criterion': 'entropy', 'random_state': 0} ενώ με μικρή διαφορά στο f1-score και το accuracy, το μεγαλύτερο accuracy συναντάται στον svm {'C': 3, 'degree': 4, 'kernel': 'poly', 'random_state': 0}. Επιπλέον, όπως και στην περίπτωση 2.1 όλοι οι αλγόριθμοι που εκπαιδεύτηκαν με τα δεδομένα που προκύπτουν από SMOTE παρουσιάζουν αισθητά χαμηλότερο f1-score στο test set από ότι στο validation set.

2.3 Αφαίρεση γραμμών με NaN

Η βιβλιοθήκη pandas της python περιέχει μία εντολή για την αφαίρεση των κενών τιμών σε ένα data frame. Η pandas.DataFrame.dropna χρησιμοποιήθηκε για τον σκοπό αυτό και το νέο σύνολο δεδομένων περιέχει πλέον 9273 εγγραφές (από τις 12528). Στη συνέχεια εφαρμόστηκε η ίδια προεπεξεργασία (εκτός του γεμίσματος των κενών τιμών) και μελετήθηκαν τα ίδια μοντέλα με το υποκεφάλαιο 2.1. Τα αποτελέσματα φαίνονται στον πίνακα

Πίνακας 3: Αποτελέσματα για τις επιλεγμένες μετρικές στο validation set και το test set.

method	**best parameters**	**validation accuracy**	**validation f1 score**	**test accuracy**	**test f1 score**
lda	-	0.9740	0.8422	0.9802	0.8832
lda (SMOTE)	-	0.9259	0.9215	0.9789	0.8879
qda	-	0.9664	0.8213	0.9737	0.8598
qda (SMOTE)	-	0.9193	0.9146	0.9715	0.8520
Logistic Regression	{'C': 1, 'penalty': None, 'random_state': 0, 'solver': 'lbfgs'}	0.9784	0.8742	0.9819	0.8960
Logistic Regression (balanced)	{'C': 1, 'class_weight': 'balanced', 'penalty': None, 'random_state': 0, 'solver': 'lbfgs'}	0.9648	0.8238	0.9672	0.8397
Logistic Regression (SMOTE)	{'C': 1, 'penalty': None, 'random_state': 0, 'solver': 'lbfgs'}	0.9364	0.9340	0.9668	0.8379
Decision Tree	{'ccp_alpha': 0.0013504197380731654, 'criterion': 'entropy', 'random_state': 0}	0.9791	0.8824	0.9815	0.8969
Decision Tree (balanced)	{'ccp_alpha': 0.00024121518079038668, 'class_weight': 'balanced', 'criterion': 'gini', 'random_state': 0}	0.9656	0.8228	0.9664	0.8319
Decision Tree (SMOTE)	{'ccp_alpha': 0.00019787873990818427, 'criterion': 'entropy', 'random_state': 0}	0.9600	0.9602	0.9414	0.7364
svm	{'C': 10, 'kernel': 'rbf', 'random_state': 0}	0.9780	0.8723	0.9815	0.8949
svm (balanced)	{'C': 1, 'class_weight': 'balanced', 'degree': 2, 'kernel': 'poly', 'random_state': 0}	0.9740	0.8635	0.9776	0.8839
svm (SMOTE)	{'C': 9, 'degree': 5, 'kernel': 'poly', 'random_state': 0}	0.9448	0.9429	0.9754	0.8736

Εδώ παρατηρούνται λίγο πιο διαφοροποιημένες τιμές υπερπαραμέτρων από τις προηγούμενες περιπτώσεις. Το μεγαλύτερο accuracy στο test set εμφανίζεται για δύο διαφορετικούς αλγορίθμους decision tree 'ccp_alpha': 0.0013504197380731654, 'criterion': 'entropy', 'random_state': 0} και svm {'C': 10, 'kernel': 'rbf', 'random_state': 0}, όμως με μικρή διαφορά το μεγαλύτερο f1-score προκύπτει στον πρώτο. Επιπλέον, για τους αλγορίθμους που εκπαιδεύτηκαν με το SMOTE σύνολο εκπαίδευσης υπάρχει ακόμα η διαφορά στα f1-scores ανάμεσα στο test και validation set αν και είναι μικρότερη (μέχρι περίπου 10%) με εξαίρεση τον decision tree αλγόριθμο. Αυτό ίσως να συμβαίνει γιατί με την αφαίρεση των κενών τιμών τα επιπλέον δείγματα που προστίθενται από το SMOTE είναι πιο αντιπροσωπευτικά της θετικής κλάσης. Ο μέσος όρος και ο median τείνουν να παίρνουν τιμές πιο κοντά στην αρνητική κλάση καθώς αυτή είναι η επικρατούσα με αποτέλεσμα ο SMOTE να γεννά δείγματα που δεν είναι αρκετά αντιπροσωπευτικά της θετικής κλάσης.

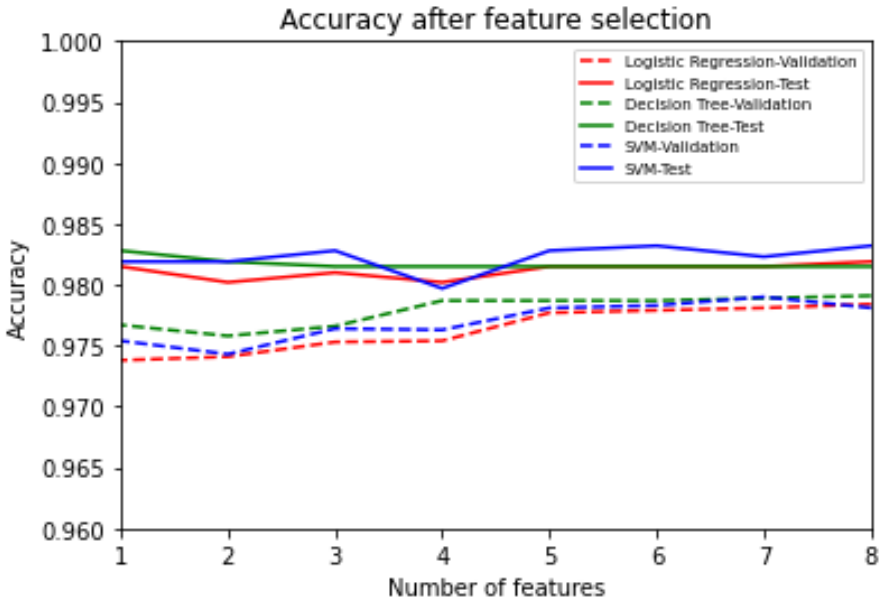
3 Μοντέλα με επιλογή χαρακτηριστικών

Για την επιλογή χαρακτηριστικών χρησιμοποιήθηκε ο sklearn.feature_selection.SelectKBest στο σύνολο εκπαίδευσης στο οποίο έχουν αφαιρεθεί τα NaN (με την προεπεξεργασία που έγινε στο υποκεφάλαιο 2.3). Αρχικά θα μελετηθούν οι βελτιστοποιημένοι αλγόριθμοι logistic regression, decision tree και svm στο σύνολο δεδομένων για μειούμενο αριθμό χαρακτηριστικών (από 8 μέχρι 1), και στη συνέχεια θα ακολουθηθεί η ίδια διαδικασία για το σύνολο δεδομένων με SMOTE.

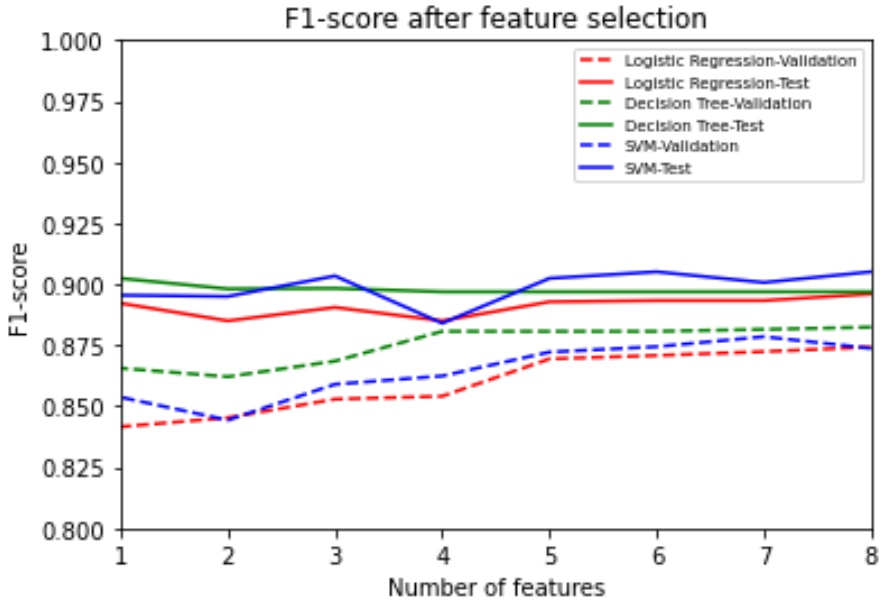
3.1 Επιλογή χαρακτηριστικών

Η διαδικασία που ακολουθήθηκε για την εξαγωγή των σχημάτων 1,2 αποτελείται από τα παρακάτω βήματα:

1. Προεπεξεργασία σύμφωνα με το υποκεφάλαιο 2.3.
2. Επιλογή χαρακτηριστικών μέσω του SelectKBest, (αρχικά 8 χαρακτηριστικά)
3. Επιλογή υπερπαραμέτρων που δίνουν το καλύτερο f1-score για τους αλγόριθμους logistic regression, decision tree και svm και καταγραφή αποτελεσμάτων σε πίνακα (δες τον πίνακα στο 5.1)
4. Μείωση του αριθμού των χαρακτηριστικών κατά 1 και επανάληψη βημάτων 2-4 μέχρι ο αριθμός των χαρακτηριστικών να γίνει 1.
5. Αποθήκευση πίνακα και σχεδιασμός των σχημάτων 1,2.



Σχήμα 1: Accuracy σε συνάρτηση με τον αριθμό των καλύτερων χαρακτηριστικών.

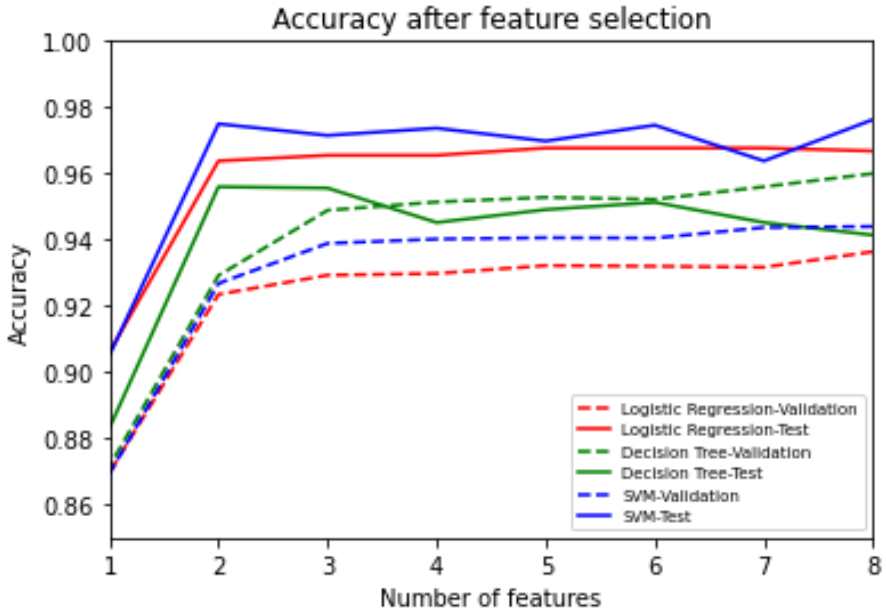


Σχήμα 2: F1-score σε συνάρτηση με τον αριθμό των καλύτερων χαρακτηριστικών.

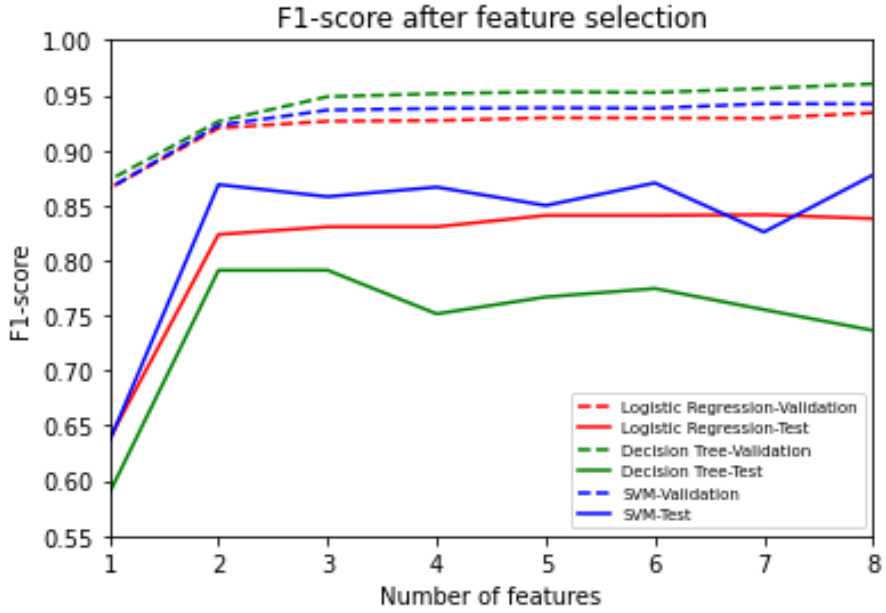
΄Οπως φαίνεται από τα σχήματα 1,2 το accuracy και f1-score δεν μεταβάλλεται ιδιαίτερα μειώνοντας τα χαρακτηριστικά. Αυτό σημαίνει ότι οι δύο κλάσεις μπορούν να περιγραφούν πολύ καλά με χρήση ενός μονο χαρακτηριστικού. Σύμφωνα με τον πίνακα στο παράρτημα A 5.1, το χαρακτηριστικό που περιέχει τη σημαντικότερη πληροφορία για το αν ένας αστέρας είναι pulsar ή όχι, είναι το excess kurtosis of the integrated profile, με καλύτερο accuracy και f1-score στο validation και test set με το decision tree {'ccp_alpha': 0.0016271480812315078, 'criterion':'gini', 'random_state': 0}.

3.2 Επιλογή χαρακτηριστικών με SMOTE

Ακολουθήθηκε η ίδια διαδικασία με το υποκεφάλαιο 3.1 με τη διαφορά ότι το σύνολο εκπαίδευσης προκύπτει από το SMOTE. Στη συνέχεια, σχεδιάστηκαν τα σχήματα 5.2,5.2. Τα αποτελέσματα φαίνονται αναλυτικότερα και στον πίνακα του παραρτήματος B 5.2.



Σχήμα 3: Accuracy σε συνάρτηση με τον αριθμό των καλύτερων χαρακτηριστικών.



Σχήμα 4: F1-score σε συνάρτηση με τον αριθμό των καλύτερων χαρακτηριστικών.

Σε αυτή την περίπτωση φαίνεται ξεκάθαρα ότι για ένα χαρακτηριστικό έχουμε υπερεκπαίδευση καθώς τα f1-scores είναι σημαντικά πιο χαμηλά στο test set. Στο δεύτερο χαρακτηριστικό φαίνεται να δημιουργείται ένα ¨γόνατο¨ και οι τιμές των δύο μετρικών αυξάνουν. Το χαρακτηριστικό που επιβιώνει ως ένα είναι το mean of the integrated profile που είναι διαφορετικό από το χαρακτηριστικό που επιλέχθηκε στην περίπτωση του υποκεφαλαίου 3.1. Το δεύτερο χαρακτηριστικό που είναι το excess kurtosis of the integrated profile που παρατηρήθηκε ότι περιέχει αρκετές πληροφορίες για τη δημιουργία ενός ικανοποιητικού μοντέλου.

4 Σύνοψη

Αρχικά μελετήθηκαν μοντέλα χωρίς επιλογή χαρακτηριστικών. Γενικά, όλοι οι αλγόριθμοι παρουσιάζουν καλύτερα αποτελέσματα για την unbalanced εκδοχή του συνόλου εκπαίδευσης, λιγότερο καλά για τις balanced (όσοι αλγόριθμοι υποστηρίζουν αυτή τη λειτουργία) και τα χειρότερα αποτελέσματα προκύπτουν απο SMOTE. Επιπλέον, για την περίπτωση του SMOTE εμφανίζεται υπερεκπαίδευση.

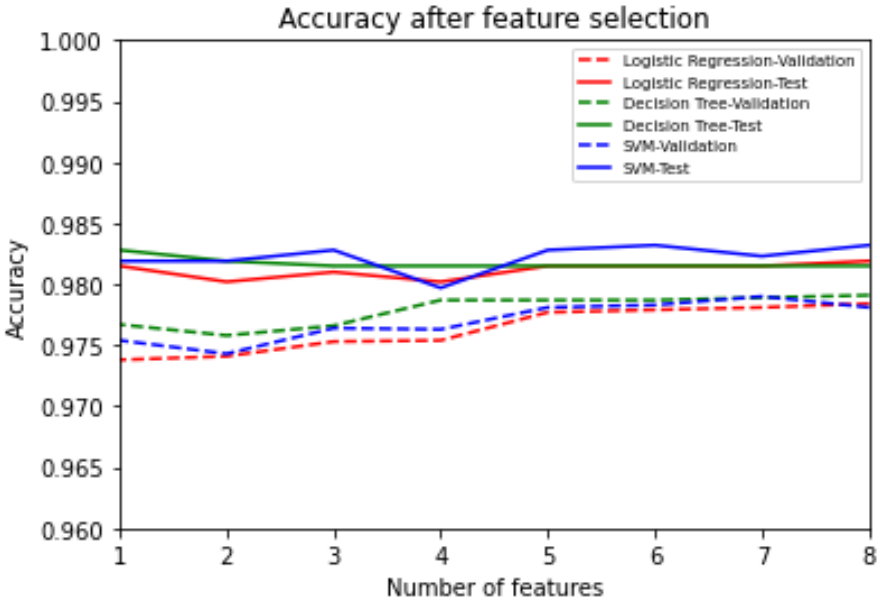
Οι βελτιστοποιημένοι αλγόριθμοι καθώς και οι lda και qda δεν παρουσιάζουν μεγάλες διαφορές στις επιδόσεις τους με εξαίρεση το SMOTE. Υψηλότερες τιμές μετρικών παρατηρούνται για το σύνολο δεδομένων που έχουν αφαιρεθεί οι κενές τιμές με μικρή διαφορά.

Η αφαίρεση χαρακτηριστικών από το σύνολο δεδομένων δε φαίνεται να επηρεάζει σημαντικά τις μετρικές αξιολόγησης στην περίπτωση του υποκεφαλαίου 3.1 καθώς παραμένουν σε σταθερά επίπεδα μέχρι και να επιλεχθεί ένα χαρακτηριστικό. Στην περίπτωση του υποκεφαλαίου 3.2 παρατηρείται απότομη πτώση στις τιμές των μετρικών, ιδιαίτερα στο test set. Αυτό υποδεικύει ότι η επιλογή του ενός χαρακτηριστικού είναι λανθασμένη ή ότι εξαρτάται και από άλλα χαρακτηριστικά σε συνδυασμό. Εαν συνδυαστουν τα δύο αποτελέσματα από τις διαφορετικές επιλογές χαρακτηριστικών, εύκολα μπορεί κάποιος να καταλήξει ότι το χαρακτηριστικό που δίνει την περισσότερη πληροφορία για το αν ένας αστέρας είναι pulsar είναι το excess kurtosis of the integrated profile.

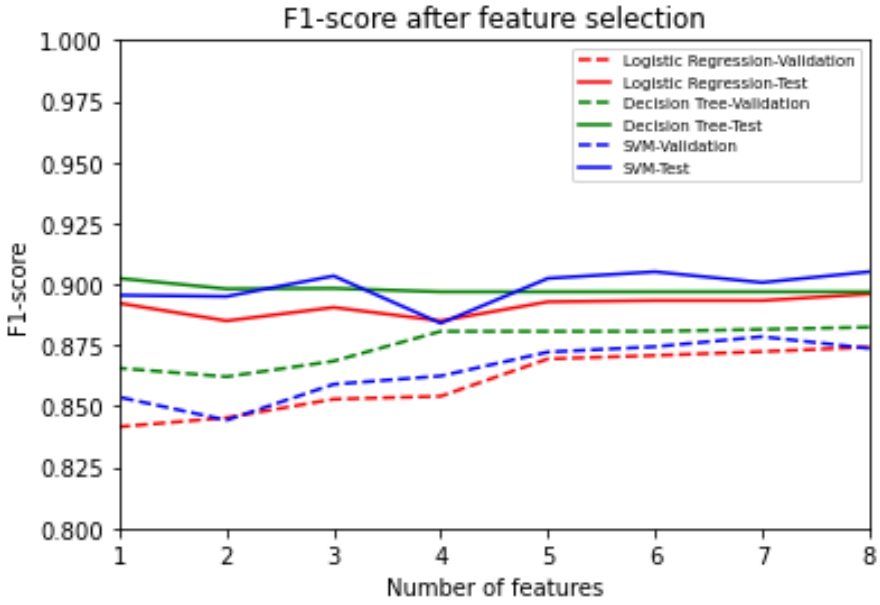
5 Παραρτήματα

5.1 Παραρτημα Α

Στην επόμενη σελίδα βρίσκεται ο πίνακας που προκύπτει από το feature selection στο σύνολο εκπαίδευσης που έχουν αφαιρεθεί οι κενές τιμές.



Accuracy σε συνάρτηση με τον αριθμό των καλύτερων χαρακτηριστικών.



F1-score σε συνάρτηση με τον αριθμό των καλύτερων χαρακτηριστικών.

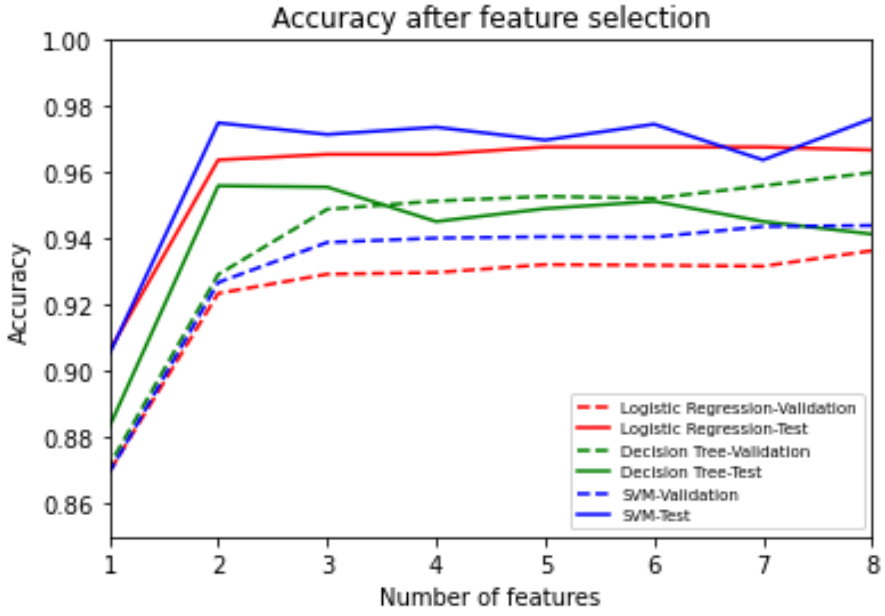
Πίνακας που προκύπτει από feature selection

method	best parameters	num features	features	validation accuracy	validation f1 score	test accuracy	test f1 score
Logistic Regression	{'C': 1, 'penalty': None, 'random_state': 0, 'solver': 'lbfgs'}	8	' Mean of the integrated profile', ' Standard deviation of the integrated profile', ' Excess kurtosis of the integrated profile', ' Skewness of the integrated profile', ' Mean of the DM-SNR curve', ' Standard deviation of the DM-SNR curve', ' Excess kurtosis of the DM-SNR curve', ' Skewness of the DM-SNR curve'	0.9784	0.8742	0.9819	0.896
Decision Tree	{'ccp_alpha': 0.00135041973807 31654, 'criterion': 'entropy', 'random_state': 0}	8		0.9791	0.8824	0.9815	0.8969
svm	{'C': 8, 'degree': 4, 'kernel': 'poly', 'max_iter': 10000, 'random_state': 0}	8		0.9781	0.8737	0.9832	0.9051
Logistic Regression	{'C': 1, 'penalty': None, 'random_state': 0, 'solver': 'lbfgs'}	7	' Mean of the integrated profile', ' Standard deviation of the integrated profile', ' Excess kurtosis of the integrated profile', ' Skewness of the integrated profile', ' Mean of the DM-SNR curve', ' Standard deviation of the DM-SNR curve', ' Excess kurtosis of the DM-SNR curve'	0.9781	0.8724	0.9815	0.8933
Decision Tree	{'ccp_alpha': 0.00175633535450 8514, 'criterion': 'entropy', 'random_state': 0}	7		0.9789	0.8815	0.9815	0.8969
svm	{'C': 6, 'degree': 4, 'kernel': 'poly', 'max_iter': 10000, 'random_state': 0}	7		0.979	0.8785	0.9823	0.9007
Logistic Regression	{'C': 1, 'penalty': None, 'random_state': 0, 'solver': 'lbfgs'}	6	' Mean of the integrated profile', ' Excess kurtosis of the integrated profile', ' Skewness of the integrated profile', ' Mean of the DM-SNR curve', ' Standard deviation of the DM-SNR curve', ' Excess kurtosis of the DM-SNR curve'	0.9779	0.8708	0.9815	0.8933
Decision Tree	{'ccp_alpha': 0.00175633535450 85106, 'criterion': 'entropy', 'random_state': 0}	6		0.9787	0.8807	0.9815	0.8969
svm	{'C': 7, 'degree': 4, 'kernel': 'poly', 'max_iter': 10000, 'random_state': 0}	6		0.9783	0.8744	0.9832	0.9051
Logistic Regression	{'C': 1, 'penalty': None, 'random_state': 0, 'solver': 'lbfgs'}	5	' Mean of the integrated profile', ' Excess kurtosis of the integrated profile', ' Skewness of the integrated profile', ' Mean of the DM-SNR curve', ' Standard deviation of the DM-SNR curve'	0.9777	0.8694	0.9815	0.8928
Decision Tree	{'ccp_alpha': 0.00175633535450 85106, 'criterion': 'entropy', 'random_state': 0}	5		0.9787	0.8807	0.9815	0.8969
svm	{'C': 10, 'degree': 2, 'kernel': 'poly', 'max_iter': 10000, 'random_state': 0}	5		0.9781	0.8722	0.9828	0.9024
Logistic Regression	{'C': 1, 'penalty': None, 'random_state': 0, 'solver': 'lbfgs'}	4	' Mean of the integrated profile', ' Excess kurtosis of the integrated profile', ' Skewness of the integrated profile',	0.9754	0.8541	0.9802	0.885
Decision Tree	{'ccp_alpha': 0.00175633535450	4		0.9787	0.8807	0.9815	0.8969

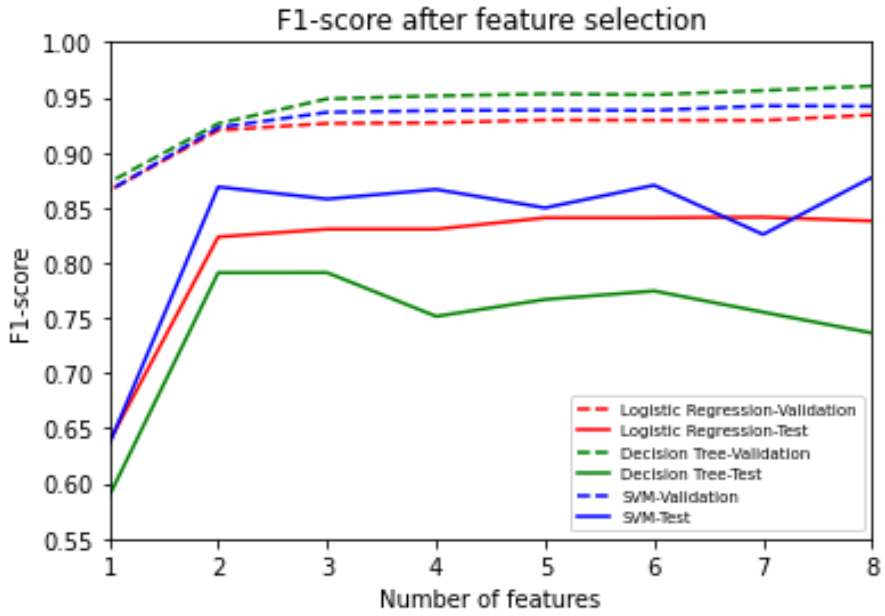
	85106, 'criterion': 'entropy', 'random_state': 0}		' Standard deviation of the DM-SNR curve'				
svm	{'C': 8, 'degree': 4, 'kernel': 'poly', 'max_iter': 10000, 'random_state': 0}	4		0.9763	0.8624	0.9797	0.884
Logistic Regression	{'C': 1, 'penalty': None, 'random_state': 0, 'solver': 'lbfgs'}	3	' Mean of the integrated profile', ' Excess kurtosis of the integrated profile', ' Skewness of the integrated profile'	0.9753	0.8529	0.981	0.8905
Decision Tree	{'ccp_alpha': 0.00048436081425055465, 'criterion': 'gini', 'random_state': 0}	3		0.9766	0.8685	0.9815	0.8983
svm	{'C': 9, 'degree': 5, 'kernel': 'poly', 'max_iter': 10000, 'random_state': 0}	3		0.9764	0.859	0.9828	0.9034
Logistic Regression	{'C': 1, 'penalty': None, 'random_state': 0, 'solver': 'lbfgs'}	2	' Excess kurtosis of the integrated profile', ' Skewness of the integrated prof	0.9741	0.8453	0.9802	0.885
Decision Tree	{'ccp_alpha': 0.12227217053584244, 'criterion': 'entropy', 'random_state': 0}	2		0.9758	0.8621	0.9819	0.8981
svm	{'C': 8, 'kernel': 'rbf', 'random_state': 0}	2		0.9743	0.8443	0.9819	0.895
Logistic Regression	{'C': 1, 'penalty': None, 'random_state': 0, 'solver': 'lbfgs'}	1	' Excess kurtosis of the integrated prof	0.9738	0.8416	0.9815	0.8922
Decision Tree	{'ccp_alpha': 0.0016271480812315078, 'criterion': 'gini', 'random_state': 0}	1		0.9767	0.8656	0.9828	0.9024
svm	{'C': 9, 'kernel': 'rbf', 'random_state': 0}	1		0.9754	0.8537	0.9819	0.8955

5.2 Παραρτημα Β

Στην επόμενη σελίδα βρίσκεται ο πίνακας που προκύπτει από το feature selection στο σύνολο εκπαίδευσης που έχουν αφαιρεθεί οι κενές τιμές και έχει εφαρμοστεί SMOTE.



Accuracy σε συνάρτηση με τον αριθμό των καλύτερων χαρακτηριστικών.



F1-score σε συνάρτηση με τον αριθμό των καλύτερων χαρακτηριστικών.

Πίνακας που προκύπτει από feature selection SMOTE

method	best parameters	num features	features	validation accuracy	validation f1 score	test accuracy	test f1 score
Logistic Regression	{'C': 1, 'penalty': None, 'random_state': 0, 'solver': 'lbfgs'}	8	' Mean of the integrated profile', ' Standard deviation of the integrated profile', ' Excess kurtosis of the integrated profile', ' Skewness of the integrated profile', ' Mean of the DM-SNR curve', ' Standard deviation of the DM-SNR curve', ' Excess kurtosis of the DM-SNR curve', ' Skewness of the DM-SNR curve'	0.9364	0.934	0.9668	0.8379
Decision Tree	{'ccp_alpha': 0.00019787873990818427, 'criterion': 'entropy', 'random_state': 0}	8		0.96	0.9602	0.9414	0.7364
svm	{'C': 10, 'kernel': 'rbf', 'random_state': 0}	8		0.944	0.9419	0.9763	0.8775
Logistic Regression	{'C': 1, 'penalty': None, 'random_state': 0, 'solver': 'lbfgs'}	7	' Mean of the integrated profile', ' Standard deviation of the integrated profile', ' Excess kurtosis of the integrated profile', ' Skewness of the integrated profile', ' Standard deviation of the DM-SNR curve', ' Excess kurtosis of the DM-SNR curve', ' Skewness of the DM-SNR curve'	0.9317	0.929	0.9677	0.8414
Decision Tree	{'ccp_alpha': 0.00029428335333279857, 'criterion': 'gini', 'random_state': 0}	7		0.956	0.9561	0.9452	0.7553
svm	{'C': 1, 'degree': 5, 'kernel': 'poly', 'max_iter': 10000, 'random_state': 0}	7		0.9437	0.9422	0.9638	0.8257
Logistic Regression	{'C': 1, 'penalty': None, 'random_state': 0, 'solver': 'lbfgs'}	6	' Mean of the integrated profile', ' Excess kurtosis of the integrated profile', ' Skewness of the integrated profile', ' Standard deviation of the DM-SNR curve', ' Excess kurtosis of the DM-SNR curve', ' Skewness of the DM-SNR curve'	0.932	0.9292	0.9677	0.8408
Decision Tree	{'ccp_alpha': 0.00015755571890960084, 'criterion': 'gini', 'random_state': 0}	6		0.9522	0.9523	0.9513	0.7745
svm	{'C': 10, 'kernel': 'rbf', 'random_state': 0}	6		0.9405	0.9381	0.9746	0.8703
Logistic Regression	{'C': 1, 'penalty': None, 'random_state': 0, 'solver': 'lbfgs'}	5	' Mean of the integrated profile', ' Excess kurtosis of the integrated profile', ' Skewness of the integrated profile', ' Standard deviation of the DM-SNR curve', ' Excess kurtosis of the DM-SNR curve'	0.9322	0.9294	0.9677	0.8408
Decision Tree	{'ccp_alpha': 0.000211681772652148, 'criterion': 'gini', 'random_state': 0}	5		0.9528	0.953	0.9491	0.7668
svm	{'C': 9, 'degree': 3, 'kernel': 'poly', 'max_iter': 10000, 'random_state': 0}	5		0.9406	0.9384	0.9698	0.8498
Logistic Regression	{'C': 1, 'penalty': None, 'random_state': 0, 'solver': 'lbfgs'}	4	' Mean of the integrated profile', ' Excess kurtosis of the integrated profile', ' Standard deviation of the DM-SNR curve',	0.9298	0.9269	0.9655	0.8305
Decision Tree	{'ccp_alpha': 0.00022932112614	4		0.9514	0.9513	0.9452	0.7515

	923423, 'criterion': 'gini', 'random_state': 0}		' Excess kurtosis of the DM-SNR curve'				
svm	{'C': 10, 'kernel': 'rbf', 'random_state': 0}	4		0.9402	0.9379	0.9737	0.8665
Logistic Regression	{'C': 1, 'penalty': None, 'random_state': 0, 'solver': 'lbfgs'}	3	' Mean of the integrated profile', ' Excess kurtosis of the integrated profile', ' Standard deviation of the DM-SNR curve'	0.9293	0.9263	0.9655	0.8305
Decision Tree	{'ccp_alpha': 0.00026383831987 75789, 'criterion': 'gini', 'random_state': 0}	3		0.9489	0.9486	0.9556	0.7911
svm	{'C': 10, 'kernel': 'rbf', 'random_state': 0}	3		0.9389	0.9364	0.9715	0.8578
Logistic Regression	{'C': 1, 'penalty': None, 'random_state': 0, 'solver': 'lbfgs'}	2	' Mean of the integrated profile', ' Excess kurtosis of the integrated prof	0.9235	0.9201	0.9638	0.8235
Decision Tree	{'ccp_alpha': 0.00063629597693 96952, 'criterion': 'entropy', 'random_state': 0}	2		0.9292	0.9261	0.956	0.791
svm	{'C': 3, 'kernel': 'rbf', 'random_state': 0}	2		0.9268	0.9225	0.975	0.8688
Logistic Regression	{'C': 3, 'penalty': 'l2', 'random_state': 0, 'solver': 'lbfgs'}	1	' Mean of the integrated prof	0.8696	0.8655	0.906	0.6379
Decision Tree	{'ccp_alpha': 0.00048905051734 62574, 'criterion': 'gini', 'random_state': 0}	1		0.8715	0.8729	0.8831	0.5888
svm	{'C': 1, 'degree': 2, 'kernel': 'poly', 'max_iter': 10000, 'random_state': 0}	1		0.8693	0.8654	0.9051	0.6358