# Όραση Υπολογιστών

# 3η Εργασία

## Περιεχόμενα



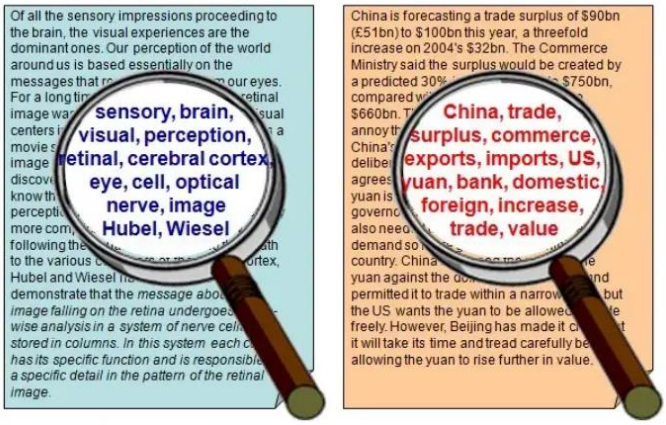
## 1. Εισαγωγή

Στην παρούσα εργασία διερευνάται το πρόβλημα της ταξινόμησης πολλαπλών κλάσεων (multi-class classification), με τη χρήση της βιβλιοθήκης OpenCV. Πιο συγκεκριμένα, παράγεται ένα οπτικό λεξικό με τη βοήθεια του αλγορίθμου K-Means και με τη χρήση ενός συνόλου εικόνων εκπαίδευσης (training images). Στη συνέχεια, γίνεται χρήση των ταξινομητών KNN (Κ πλησιέστεροι γείτονες) και SVM (Μηχανές υποστηρικτικών διανυσμάτων) για την ταξινόμηση ενός συνόλου εικόνων δοκιμής (testing images). Τέλος, γίνεται αξιολόγηση του συστήματος και διερευνώνται τυχόν αστοχίες του.

## 2. Σάκος οπτικών λέξεων (Bag of Visual Words)

Ο σάκος οπτικών λέξεων (BOVW) χρησιμοποιείται συνήθως στην ταξινόμηση εικόνων. Ο αλγόριθμος είναι εμπνευσμένος από τον αλγόριθμο του σάκου λέξεων (bag of words) που προέρχεται από τον τομέα της φυσικής επεξεργασίας γλώσσας (natural language processing).

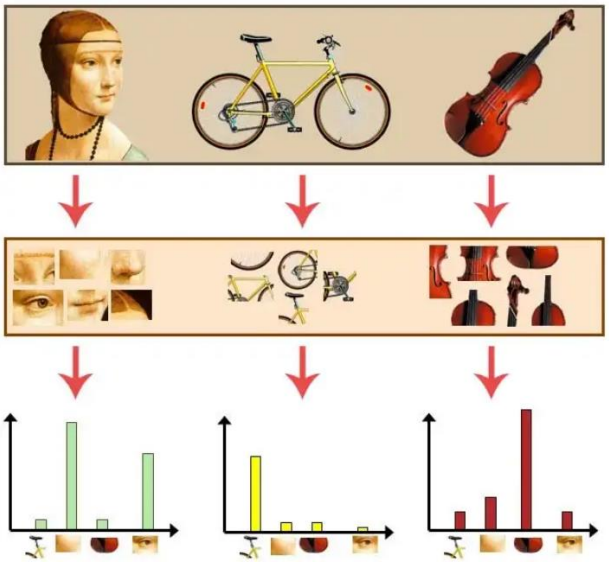
Στο σάκο λέξεων (BOW), μετράμε πόσες φορές εμφανίζεται κάθε λέξη σε ένα έγγραφο και χρησιμοποιούμε τη συχνότητα εμφάνισης κάθε λέξης ενδιαφέροντος (keyword) για να κάνουμε ένα ιστόγραμμα συχνότητας. Στο σάκο οπτικών λέξεων (BOVW) αντί για λέξεις, χρησιμοποιούμε διάφορα χαρακτηριστικά (features) των εικόνων.




Η γενική ιδέα του σάκου οπτικών λέξεων (BOVW) είναι να αναπαριστά μια εικόνα σαν ένα σύνολο χαρακτηριστικών (features). Τα χαρακτηριστικά είναι στην ουσία σημεία ενδιαφέροντος (keypoints) και περιγραφείς (descriptors). Τα σημεία ενδιαφέροντος είναι τα σημεία που χαρακτηρίζουν μια εικόνα, επομένως είναι ανεξάρτητα από την περιστροφή, τη συρρίκνωση, την μεγέθυνση ή οποιονδήποτε άλλο μετασχηματισμό της εικόνας. Οι περιγραφείς είναι διανύσματα που περιγράφουν τα σημεία ενδιαφέροντος.

Χρησιμοποιούμε τα σημεία ενδιαφέροντος και τους περιγραφείς για να παράξουμε λεξικά οπτικών λέξεων (vocabularies of visual words) και να αναπαραστήσουμε κάθε εικόνα σαν ένα ιστόγραμμα συχνότητας (frequency histogram) των χαρακτηριστικών που υπάρχουν στην εικόνα. Από το ιστόγραμμα συχνότητας, μπορούμε να βρούμε άλλες παρόμοιες εικόνες ή να προβλέψουμε την κατηγορία στην οποία ανήκει η εικόνα.

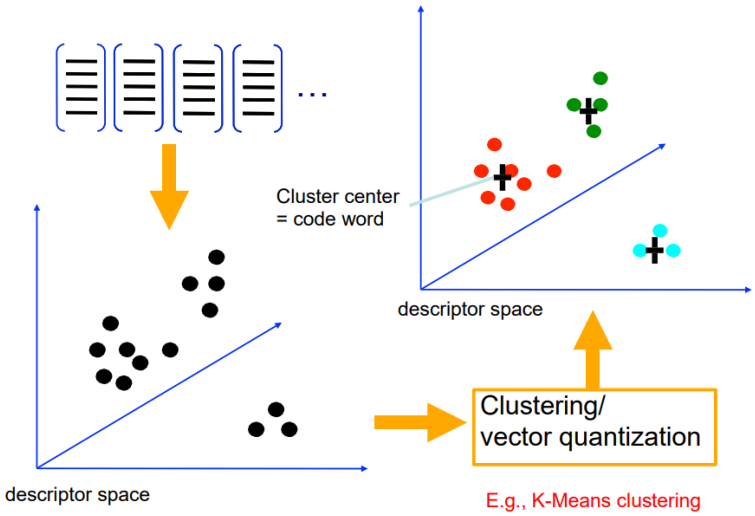
Παρακάτω φαίνεται ο τρόπος με τον οποίο γίνεται η εξαγωγή χαρακτηριστικών και η παραγωγή ιστογραμμάτων συχνότητας από κάποιες εικόνες.

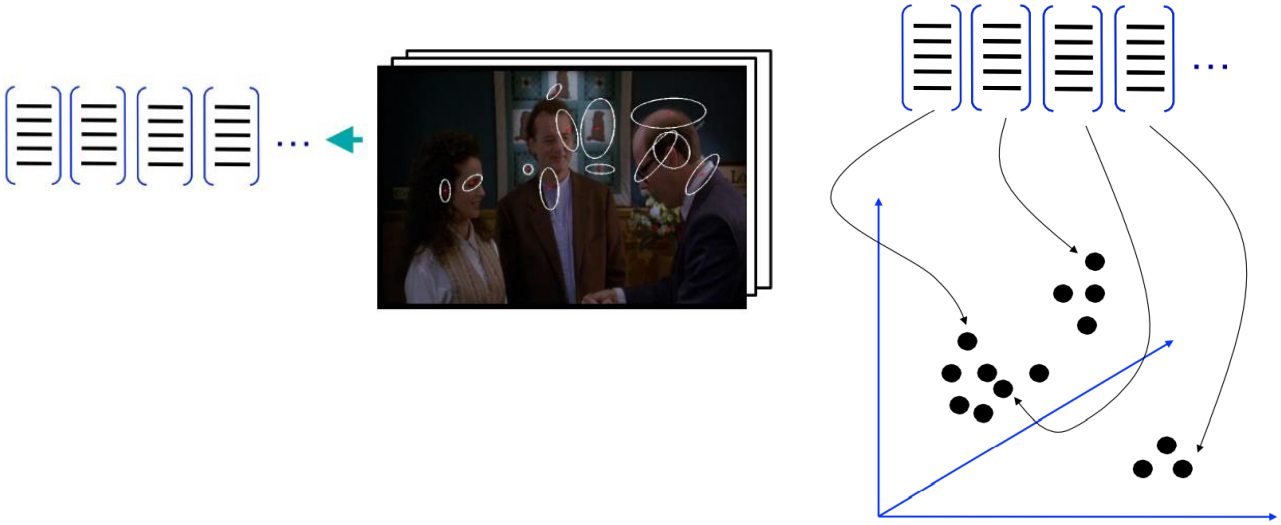


Για την παραγωγή λεξικού **ανιχνεύουμε σημεία ενδιαφέροντος στις εικόνες**, **εξάγουμε περιγραφείς** από κάθε εικόνα στο σύνολο δεδομένων εκπαίδευσης και δημιουργούμε ένα οπτικό λεξικό. Η ανίχνευση των σημείων ενδιαφέροντος και η εξαγωγή περιγραφών σε μια εικόνα μπορεί να γίνει χρησιμοποιώντας αλγόριθμους εξαγωγής χαρακτηριστικών (για παράδειγμα, SIFT, SURF, κ.λπ.).

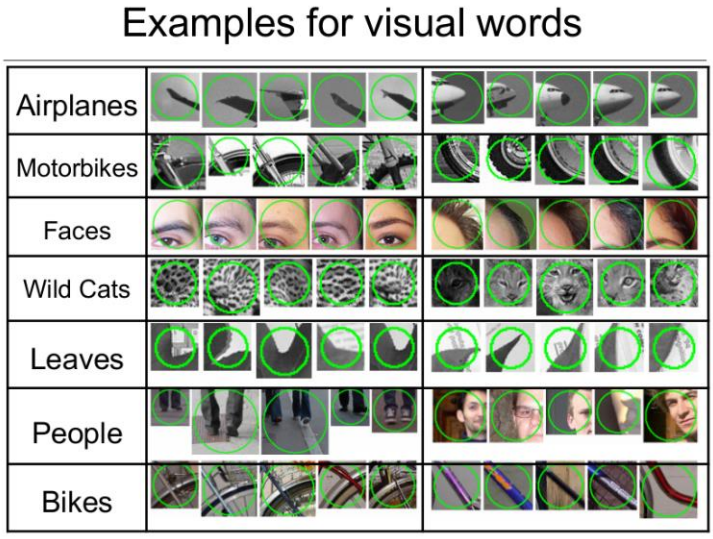

Στη συνέχεια, **φτιάχνουμε κλάσεις (clusters)** με τους περιγραφείς (μπορούμε να χρησιμοποιήσουμε K-Means ή κάποιον άλλο αλγόριθμο ομαδοποίησης). **Το κέντρο κάθε κλάσης θα χρησιμοποιηθεί ως λεξιλόγιο (σύνολο οπτικών λέξεων) του οπτικού λεξικού**.

Παρακάτω φαίνεται ο υπολογισμός ενός λεξικού χαρακτηριστικών από τις εικόνες εκπαίδευσης και η ομαδοποίηση των κλάσεων.

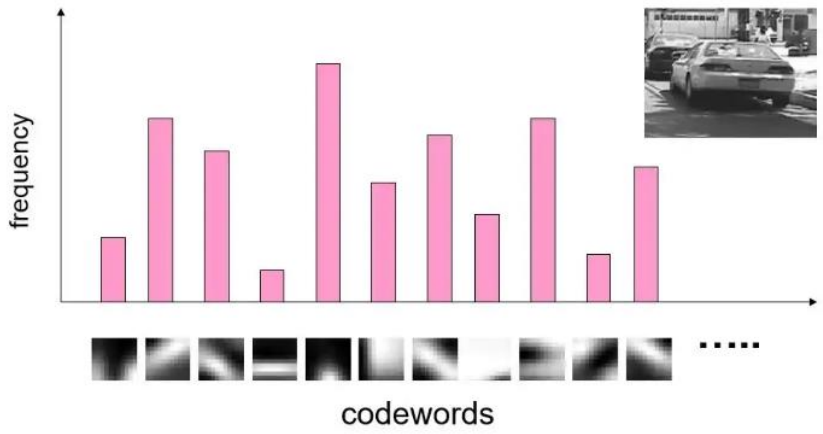


Παρακάτω φαίνεται ένα παράδειγμα οπτικού λεξικού.

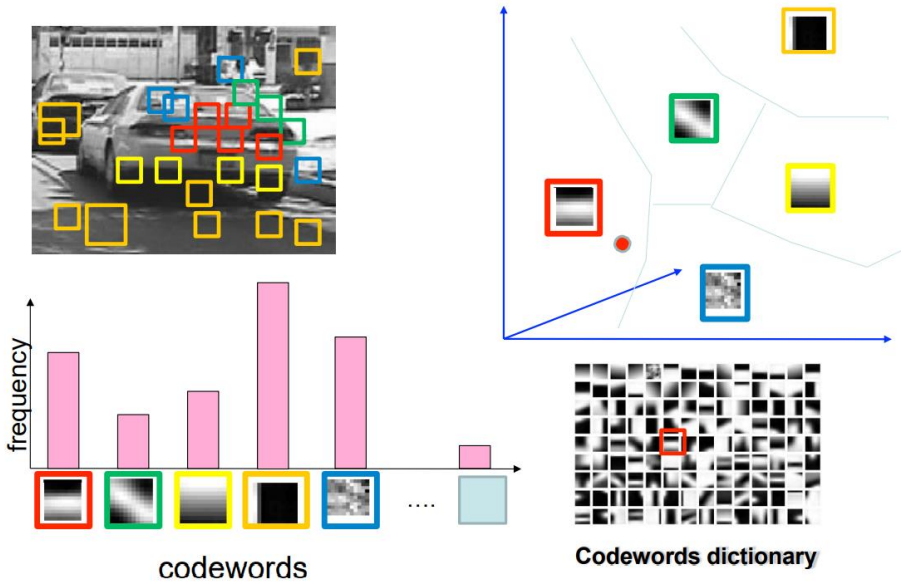


Τέλος, για κάθε εικόνα **φτιάχνουμε ιστόγραμμα συχνότητας** από το λεξιλόγιο και τη συχνότητα εμφάνισης των λέξεων της εικόνας. Από αυτό το ιστόγραμμα εξάγουμε ένα διάνυσμα, το οποίο περιέχει τη συχνότητα εμφάνισης της κάθε λέξης. Αυτό το διάνυσμα χαρακτηρίζει κάθε εικόνα. Το ιστόγραμμα είναι ο σάκος με τις οπτικές λέξεις (BOVW).




Ανάλογα με τη μορφή του ιστογράμματος δημιουργούνται διαφορετικές κατηγορίες εικόνων με τα ίδια χαρακτηριστικά (παρόμοια διανύσματα), όσον αφορά το σύνολο των εικόνων εκπαίδευσης. Οπότε, πλέον μια νέα εικόνα από το σύνολο εικόνων δοκιμής, ανάλογα με τα χαρακτηριστικά που περιέχει και το ιστόγραμμα που εξάγεται, μπορεί να αντιστοιχηθεί σε μια από τις παραπάνω κατηγορίες, με τη χρήση αλγορίθμων ταξινόμησης (KNN, SVM).

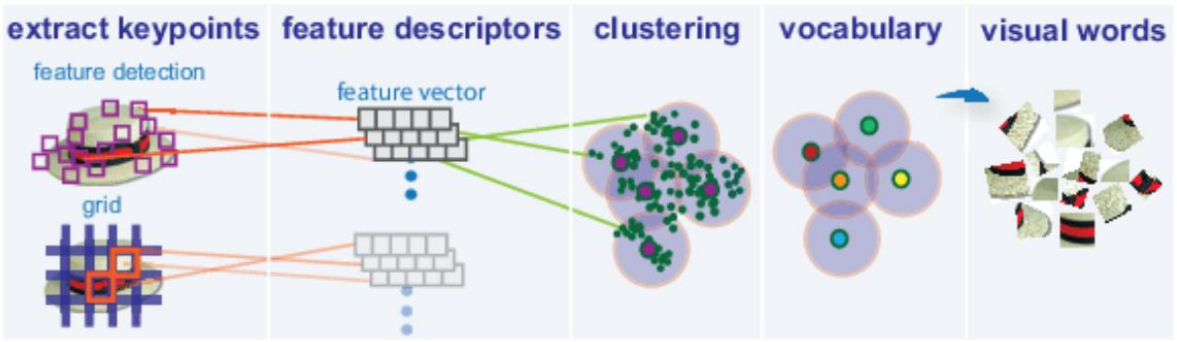
Το λεξικό είναι μια λίστα «σημαντικών» χαρακτηριστικών που εξάγονται από το σύνολο των χαρακτηριστικών σε όλες τις εικόνες εκπαίδευσης. Στόχος είναι το λεξικό να είναι αντιπροσωπευτικό του συνόλου των χαρακτηριστικών που έχουν εντοπισθεί.



Το οπτικό λεξικό δημιουργείται από τις εικόνες του συνόλου εκπαίδευσης (training set). Οι εικόνες του συνόλου δοκιμής (test set) χρησιμοποιούνται για την αξιολόγηση του συστήματος.


Παρακάτω περιγράφονται συνοπτικά τα βήματα για τη δημιουργία του λεξικού και την αναπαράσταση του σάκου οπτικών λέξεων.

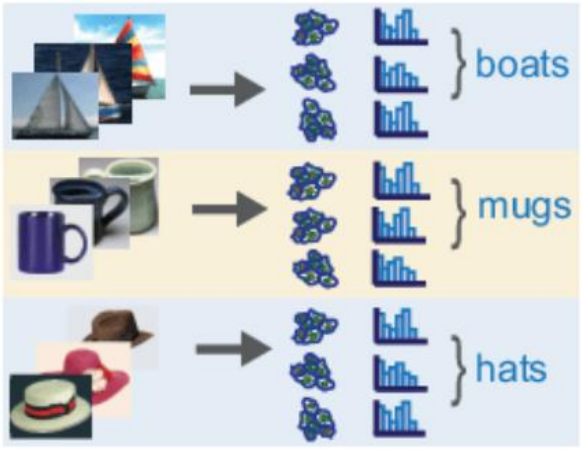
- Εξαγωγή σημείων ενδιαφέροντος και περιγραφέων στις εικόνες εκπαίδευσης
- Ομαδοποίηση περιγραφέων, δημιουργία λεξικού από οπτικές λέξεις (σάκος με οπτικές λέξεις)
- Δημιουργία ιστογράμματος και εξαγωγή διανύσματος για κάθε εικόνα



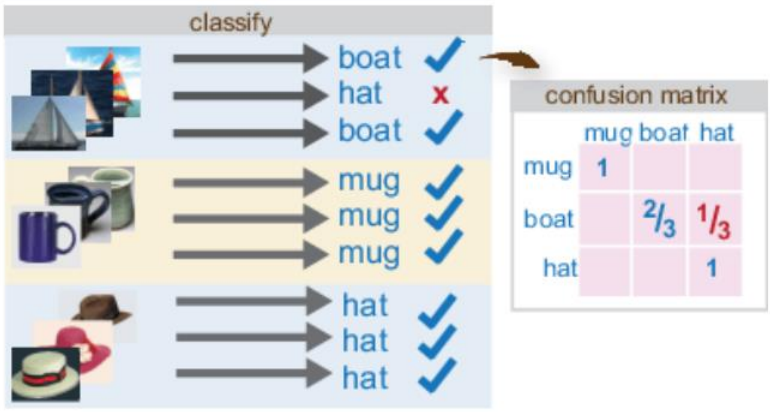


Στη συνέχεια, δημιουργείται για κάθε εικόνα εκπαίδευσης ένα ιστόγραμμα. Οι εικόνες των οποίων το ιστόγραμμα είναι παρόμοιο ανήκουν στην ίδια κατηγορία.

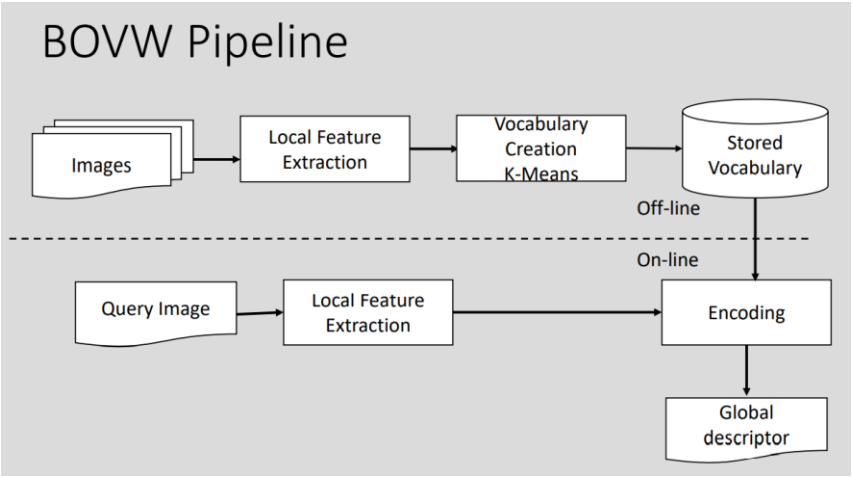
Οι εικόνες του συνόλου εκπαίδευσης χωρίζονται σε κατηγορίες, ανάλογα με τη μορφή των ιστογραμμάτων (κωδικοποίηση). Ο χωρισμός γίνεται με χρήση αλγορίθμων ταξινόμησης, όπως είναι για παράδειγμα ο KNN και ο SVM. Με τον τρόπο αυτό γίνεται η εκπαίδευση του ταξινομητή.



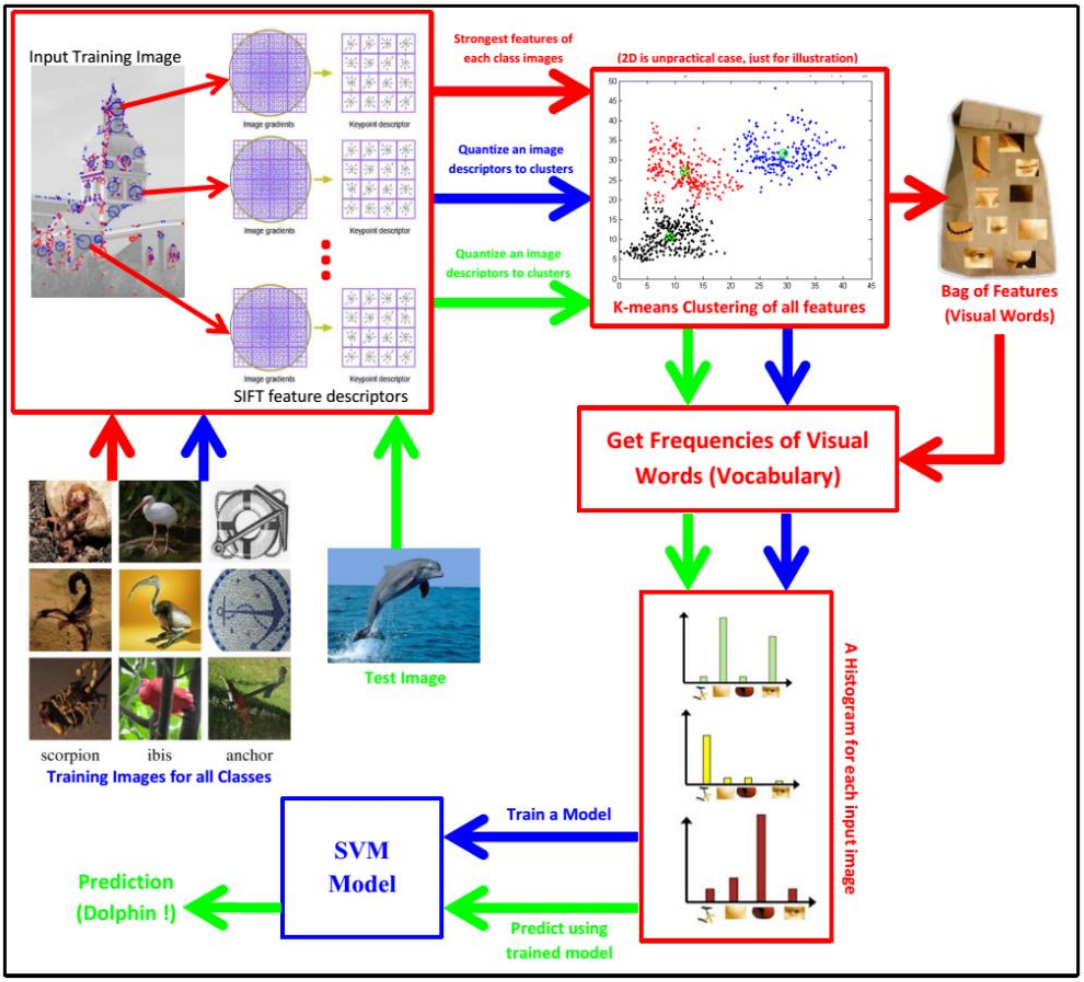

Αφού ολοκληρωθεί η εκπαίδευση, περνάμε στην αξιολόγηση του ταξινομητή. Παράγεται ένα ιστόγραμμα για κάθε εικόνα του συνόλου δοκιμής και με τη βοήθεια του ταξινομητή, αντιστοιχίζεται σε κάποια κατηγορία που έχει προκύψει από την εκπαίδευση. Η αντιστοίχιση γίνεται με βάση τη μορφή του ιστογράμματος. Η κατηγορία στην οποία γίνεται η αντιστοίχιση, περιέχει εικόνες των οποίων τα ιστογράμματα είναι παρόμοια με το ιστόγραμμα της εικόνας δοκιμής. Τέλος, η πραγματική κατηγορία στην οποία ανήκει η εικόνα δοκιμής είναι γνωστή. Είναι δηλαδή γνωστή η ετικέτα (label) της εικόνας, άρα μπορούμε να συγκρίνουμε την κατηγορία που πρόβλεψε ο ταξινομητής με την ετικέτα. Η παραπάνω διαδικασία γίνεται για όλες τις εικόνες του συνόλου δοκιμής και με κατάλληλες μετρικές γίνεται η αξιολόγηση του συστήματος.



Στο παρακάτω σχήμα φαίνονται τα βήματα που εξηγήσαμε προηγουμένως.




Παρακάτω παρουσιάζεται ένα ολοκληρωμένο διάγραμμα, το οποίο περιλαμβάνει όλα τα βήματα του αλγορίθμου ταξινόμησης.



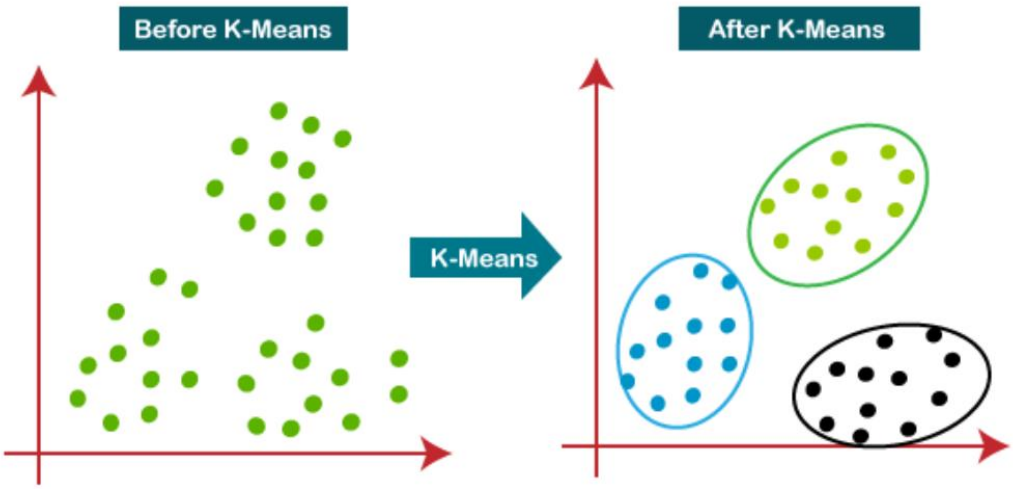
## 3. Ομαδοποίηση των Κ μέσων (K-Means Clustering)

Ο αλγόριθμος ομαδοποίησης Κ μέσων, είναι ένας αλγόριθμος μη επιτηρούμενης μάθησης (unsupervised learning) που χρησιμοποιείται για την επίλυση προβλημάτων ομαδοποίησης. Το Κ ορίζει τον αριθμό των προκαθορισμένων συστάδων (clusters) που πρέπει να δημιουργηθούν κατά τη διαδικασία, για παράδειγμα εάν Κ=2 θα προκύψουν δύο συστάδες και για Κ=3 θα προκύψουν τρεις συστάδες κ.λπ.

Είναι στην ουσία ένας επαναληπτικός αλγόριθμος που διαιρεί το σύνολο δεδομένων χωρίς ετικέτα σε Κ διαφορετικές συστάδες, με τέτοιο τρόπο ώστε κάθε συστάδα να περιέχει ένα σύνολο δεδομένων, στο οποίο όλα τα στοιχεία έχουν παρόμοιες ιδιότητες. Μας επιτρέπει να ομαδοποιήσουμε τα δεδομένα σε διαφορετικές ομάδες και να ανακαλύψουμε τις κατηγορίες των ομάδων στο μη επισημασμένο σύνολο δεδομένων, χωρίς να υπάρχει η ανάγκη εκπαίδευσης.

Κάθε συστάδα σχετίζεται με ένα κέντρο ή αλλιώς κεντροειδές. Ο κύριος στόχος του αλγορίθμου είναι να ελαχιστοποιήσει το άθροισμα των αποστάσεων μεταξύ των στοιχείων του συνόλου δεδομένων και των αντίστοιχων κεντροειδών. Ο αλγόριθμος αντιστοιχίζει κάθε στοιχείο των δεδομένων στο πλησιέστερο κεντροειδές. Τα σημεία που βρίσκονται κοντά στο συγκεκριμένο κεντροειδές Κ, δημιουργούν μια συστάδα.

Στο παρακάτω διάγραμμα εξηγείται η λειτουργία του αλγόριθμου ομαδοποίησης Κ μέσων με Κ=3:



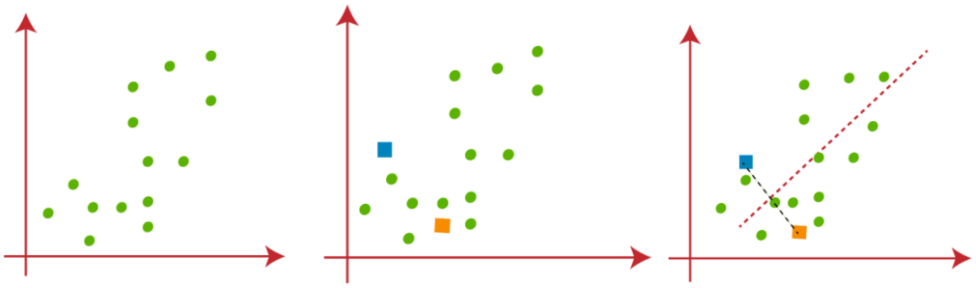

Τα βήματα του αλγορίθμου είναι τα παρακάτω:

1. Επιλέγεται ένας προκαθορισμένος αριθμός Κ των συστάδων.
2. Επιλέγονται Κ **τυχαία** σημεία, τα οποία ονομάζονται κεντροειδή και θα αποτελούν το κέντρο της κάθε συστάδας..
3. Αντιστοιχίζεται κάθε σημείο των δεδομένων στο πλησιέστερο κεντροειδές και έτσι σχηματίζονται Κ τα προκαθορισμένες συστάδες.
4. Το κεντροειδές μετακινείται στο κέντρο κάθε συστάδας.

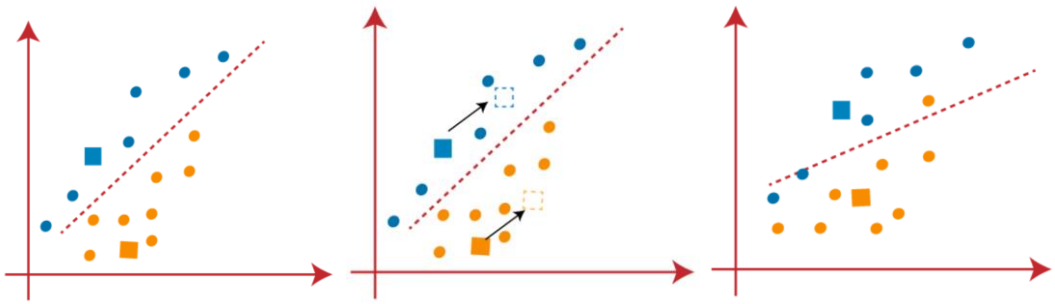
5. Αντιστοιχίζεται ξανά κάθε σημείο των δεδομένων στο πλησιέστερο κεντροειδές.
6. Τα βήματα 3 με 5 επαναλαμβάνονται, μέχρι να μην αλλάζει κανένα σημείο το κεντροειδές στο οποίο ανήκει.

Με το παρακάτω παράδειγμα γίνεται περισσότερο κατανοητή η λειτουργία του αλγορίθμου.

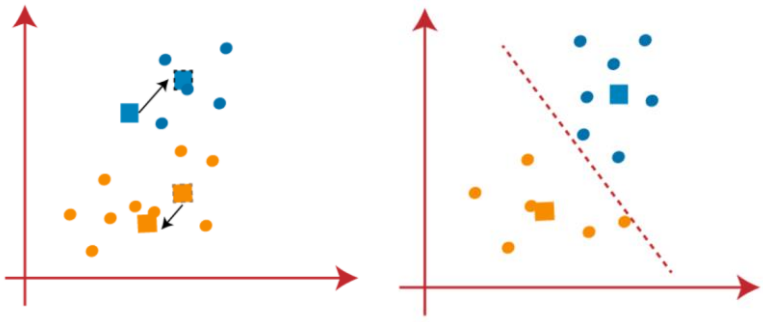
Αρχικά, επιλέγουμε K=2 και οι θέσεις των δύο κεντροειδών επιλέγονται τυχαία. Αντιστοιχίζεται κάθε σημείο των δεδομένων στο πλησιέστερο κεντροειδές και έτσι σχηματίζονται δύο συστάδες. Οι κουκίδες στις εικόνες είναι τα δεδομένα και τα δύο τετράγωνα είναι τα κεντροειδή.



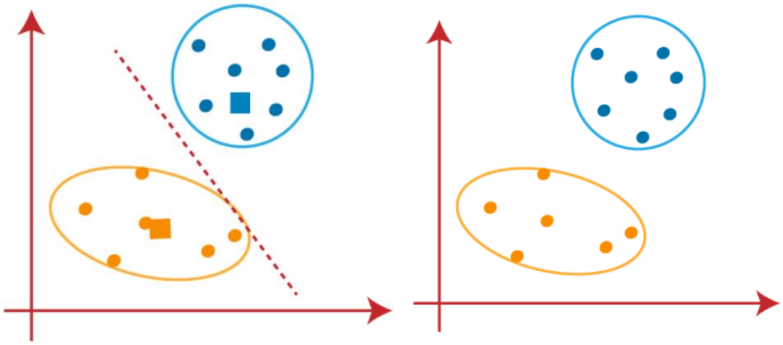
Παρακάτω φαίνονται οι δύο συστάδες με μπλε και πορτοκαλί χρώμα. Τα κεντροειδή μετακινούνται στο κέντρο της κάθε συστάδας και τα δεδομένα αντιστοιχίζονται ξανά στο πλησιέστερο κεντροειδές, με αποτέλεσμα κάποια σημεία να αλλάζουν συστάδα.



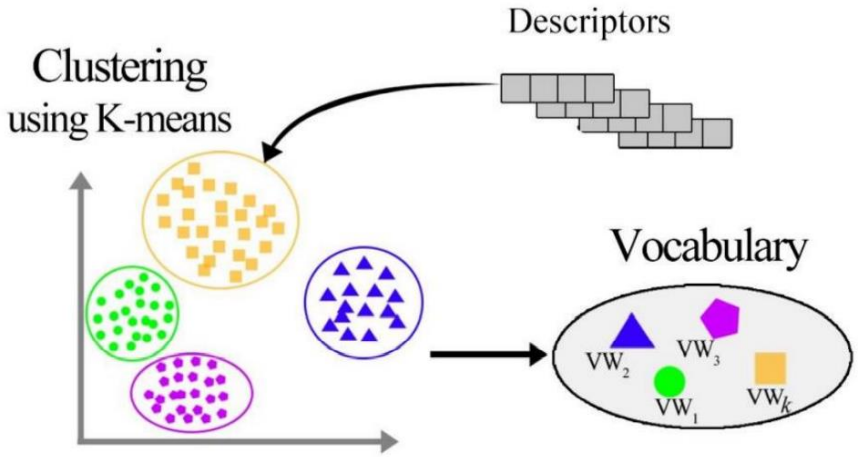
Τα κεντροειδή μετακινούνται ξανά στο κέντρο της κάθε συστάδας και τα δεδομένα αντιστοιχίζονται ξανά στο πλησιέστερο κεντροειδές.



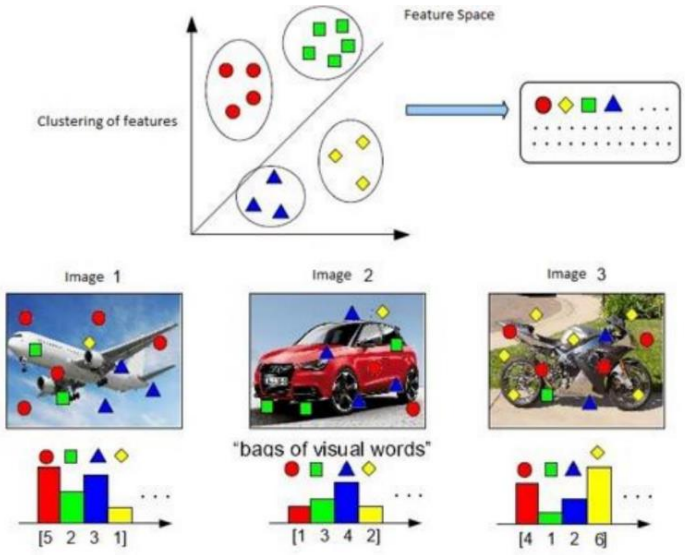
Στη συνέχεια, μετακινούνται πάλι τα κεντροειδή στο κέντρο της κάθε συστάδας και τα δεδομένα αντιστοιχίζονται ξανά στο πλησιέστερο κεντροειδές. Εφόσον, κανένα σημείο δεν αλλάζει συστάδα, έχουμε βρει τις τελικές συστάδες που φαίνονται παρακάτω.



Ο αλγόριθμος ομαδοποίησης Κ μέσων χρησιμοποιείται στο σάκο οπτικών λέξεων, για την ομαδοποίηση των περιγραφέων. Αρχικά, επιλέγουμε πόσες λέξεις θέλουμε να περιέχει το λεξικό οπτικών λέξεων, δηλαδή επιλέγουμε Κ αριθμό λέξεων. Έπειτα, οι περιγραφείς που εξάγονται από τις εικόνες εκπαίδευσης, ομαδοποιούνται σε Κ συστάδες. Με τον τρόπο αυτό δημιουργείται το λεξικό οπτικών λέξεων.




Η κωδικοποίηση της εικόνας γίνεται μετά τη δημιουργία του λεξικού. Εξάγονται για κάθε εικόνα κάποιοι περιγραφείς και αντιστοιχίζονται στην κοντινότερη λέξη του λεξικού. Δημιουργείται ιστόγραμμα με τη συχνότητα εμφάνισης κάθε λέξης μέσα στην εικόνα. Στη συνέχεια, εξάγεται για κάθε εικόνα ένα διάνυσμα από το ιστόγραμμα. Με τη χρήση αυτού του διανύσματος μπορούμε πλέον να εκπαιδεύσουμε κατάλληλους ταξινομητές στο σύνολο δεδομένων εκπαίδευσης και να τους αξιολογήσουμε στο σύνολο δεδομένων δοκιμής.

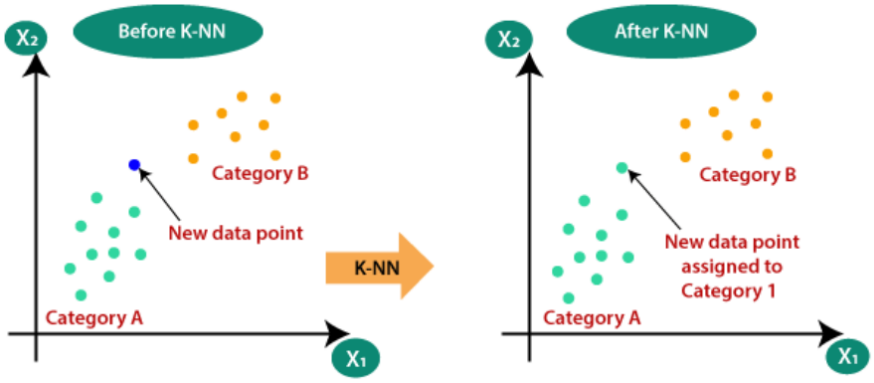


## 4. Κ πλησιέστεροι γείτονες (K-Nearest Neighbours)

Ο αλγόριθμος των Κ πλησιέστερων γειτόνων (KNN) είναι ένας από τους απλούστερους αλγόριθμους μηχανικής μάθησης που βασίζεται στην τεχνική εποπτευόμενης μάθησης (supervised learning). Χρησιμοποιείται κυρίως για προβλήματα ταξινόμησης.

Ο αλγόριθμος KNN στη φάση εκπαίδευσης απλώς αποθηκεύει το σύνολο δεδομένων, το οποίο αποτελείται από διάφορες κατηγορίες, των οποίων τα δεδομένα έχουν παρόμοια χαρακτηριστικά. Όταν λαμβάνει νέα δεδομένα, τότε τα ταξινομεί στην κατηγορία με την οποία παρουσιάζουν την μεγαλύτερη ομοιότητα.

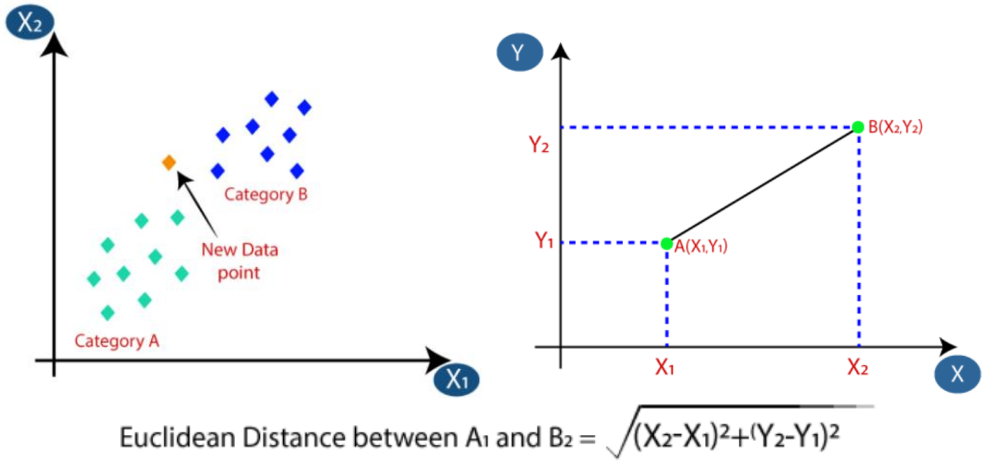
Ας υποθέσουμε ότι υπάρχουν δύο κατηγορίες, η κατηγορία Α και η κατηγορία Β και έχουμε ένα νέο δεδομένο x1. Αυτό θα βρίσκεται σε μία από αυτές τις δύο κατηγορίες. Με τη βοήθεια του KNN, μπορούμε να αναγνωρίσουμε την κατηγορία στην οποία ανήκει το νέο σημείο.




Η λειτουργία του KNN μπορεί να εξηγηθεί με βάση τον παρακάτω αλγόριθμο:

1. Επιλέγεται ο αριθμός Κ των γειτόνων.
2. Για ένα νέο σημείο που θέλουμε να ταξινομηθεί υπολογίζεται η Ευκλείδεια απόσταση του σημείου από κάθε σημείο του συνόλου εκπαίδευσης.
3. Παίρνουμε τους Κ γείτονες, δηλαδή τα Κ σημεία με τις Κ ελάχιστες αποστάσεις.
4. Για τους Κ γείτονες, βλέπουμε σε ποια κατηγορία ανήκουν. Η κατηγορία στην οποία ανήκουν οι περισσότεροι πλησιέστεροι γείτονες, είναι η κατηγορία στην οποία θα αναθέσουμε το νέο σημείο.
5. Το νέο σημείο εκχωρείται στην κατηγορία που ανήκουν οι πλησιέστεροι γείτονες.

Για Κ = 5 γείτονες υπολογίζεται η Ευκλείδεια απόσταση από κάθε γείτονα και κρατάμε τους 5 πλησιέστερους γείτονες, δηλαδή τους γείτονες με τις 5 μικρότερες Ευκλείδειες αποστάσεις.

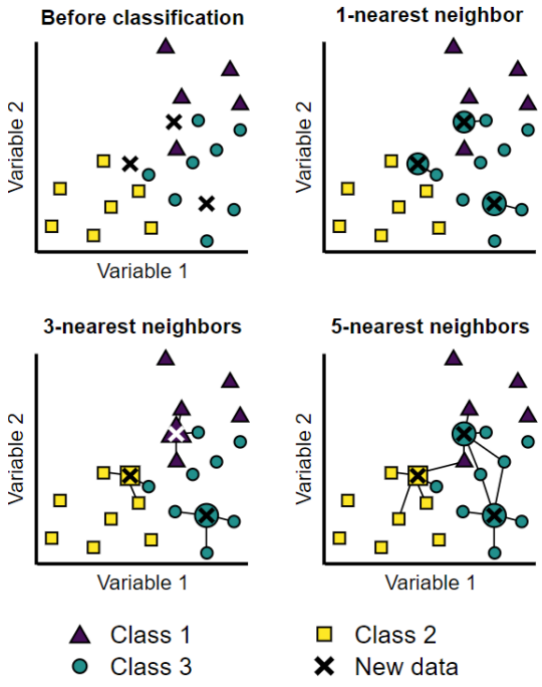



Εφόσον, οι 3 γείτονες ανήκουν στην κατηγορία Α και οι 2 ανήκουν στην κατηγορία Β, το νέο σημείο θα ανατεθεί στην κατηγορία Α.


Ανάλογα με τον αριθμό Κ των γειτόνων που επιλέγεται, μπορεί να αλλάζει και η κατηγορία στην οποία ανατίθεται το νέο σημείο. Οπότε, αλλάζει και η ακρίβεια των προβλέψεων.

Με δοκιμές μπορούμε να βρούμε τον βέλτιστο αριθμό Κ γειτόνων που μπορούν να μεγιστοποιήσουν την ακρίβεια του συστήματος ταξινόμησης.

Όπως φαίνεται στη διπλανή εικόνα, εξετάζεται η κατηγορία στην οποία ανατίθενται 3 νέα σημεία (τα σημεία με το Χ), για Κ = 1, Κ = 3, Κ = 5. Η κατηγορίες αλλάζουν, καθώς αλλάζει το Κ.



Εφόσον, έχει δημιουργηθεί το λεξικό είναι απαραίτητο να γίνει η ταξινόμηση των εικόνων δοκιμής. Ο αλγόριθμος KNN δεν περιλαμβάνει κάποιο επιπλέον στάδιο εκπαίδευσης. Οπότε, περνάμε κατευθείαν στο στάδιο των προβλέψεων.

Έχουμε διαθέσιμο ένα σύνολο διανυσμάτων (ένα διάνυσμα για κάθε εικόνα εκπαίδευσης) από τον σάκο οπτικών λέξεων με διαστάσεις 1 X αριθμός οπτικών λέξεων που έχουν επιλεγεί κατά τη δημιουργία του λεξικού. Εξάγουμε για κάθε εικόνα δοκιμής ένα περιγραφέα. Για μία εικόνα δοκιμής υπολογίζουμε την απόσταση του περιγραφέα της από τα διανύσματα του σάκου οπτικών λέξεων. Κρατάμε τα K διανύσματα με τις μικρότερες αποστάσεις. Βρίσκουμε σε ποια κατηγορία ανήκουν αυτά τα K διανύσματα, με βάση την ετικέτα τους. Η εικόνα δοκιμής ανατίθεται στην κατηγορία, στην οποία ανήκουν τα περισσότερα από αυτά τα K διανύσματα. Αυτή η κατηγορία αποτελεί την πρόβλεψή μας.

Η παραπάνω διαδικασία γίνεται για όλες τις εικόνες δοκιμής. Αν γνωρίζουμε τις ετικέτες των εικόνων δοκιμής (επιτηρούμενη μάθηση) μπορούμε με τη χρήση κατάλληλων μετρικών να αξιολογήσουμε το σύστημα και ανάλογα με την απόδοσή του, να το χρησιμοποιήσουμε για προβλέψεις σε άγνωστες εικόνες.

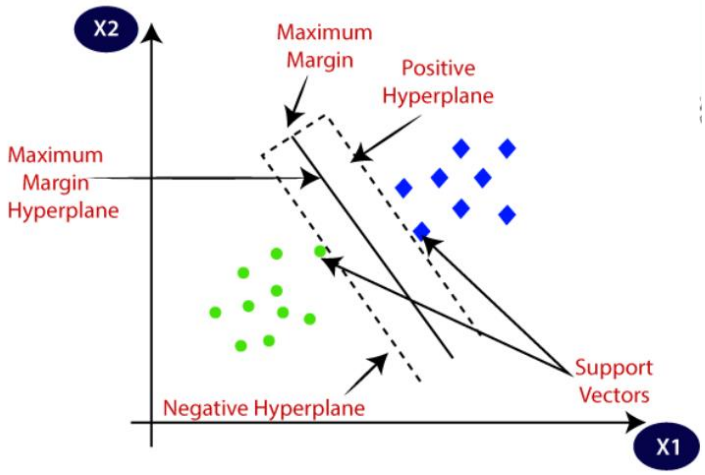
## 5. Μηχανές υποστηρικτικών διανυσμάτων (Support Vector Machines)

Οι μηχανές υποστηρικτικών διανυσμάτων (SVM) είναι ένας από τους πιο δημοφιλείς αλγόριθμους εποπτευόμενης μάθησης (supervised learning), ο οποίος χρησιμοποιείται κυρίως για προβλήματα ταξινόμησης.

Στόχος του αλγόριθμου SVM είναι να δημιουργήσει την καλύτερη γραμμή (line) ή όριο απόφασης (decision boundary) που μπορεί να διαχωρίσει το χώρο n-διαστάσεων σε κλάσεις (κατηγορίες), ώστε να μπορούμε εύκολα να βάλουμε το νέο δεδομένο στη σωστή κατηγορία. Αυτό το όριο καλύτερης απόφασης ονομάζεται υπερεπίπεδο (hyperplane).

Το SVM επιλέγει τα ακραία σημεία/διανύσματα που βοηθούν στη δημιουργία του υπερεπίπεδου. Αυτές οι ακραίες περιπτώσεις ονομάζονται διανύσματα υποστήριξης (support vectors)και ως εκ τούτου ο αλγόριθμος ονομάζεται μηχανές υποστηρικτικών διανυσμάτων (Support Vector Machines).

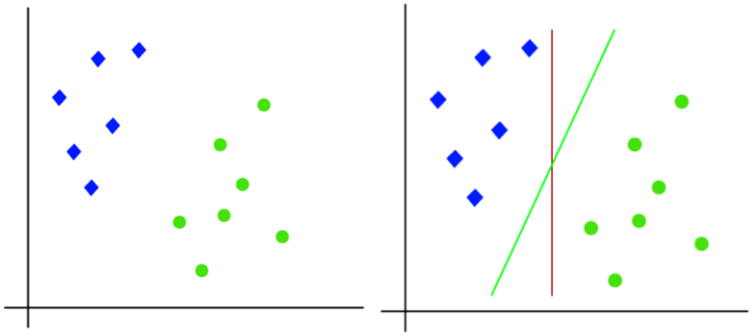
Στο παρακάτω διάγραμμα υπάρχουν δύο διαφορετικές κατηγορίες που ταξινομούνται χρησιμοποιώντας ένα υπερεπίπεδο:



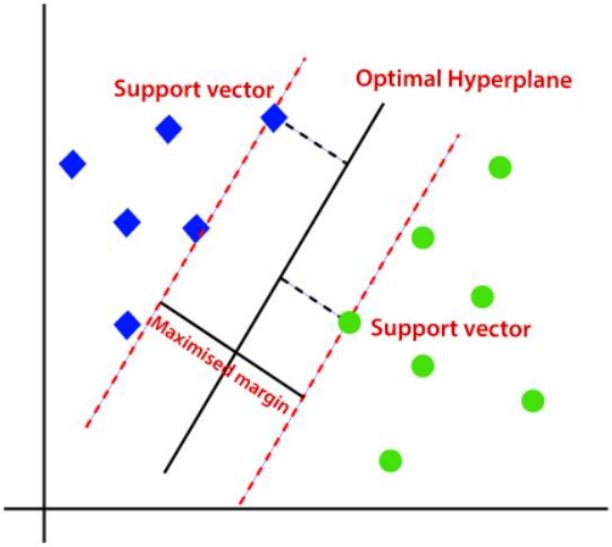

Το υπερεπίπεδο, διαχωρίζει τις δύο κατηγορίες.

Η λειτουργία του αλγορίθμου SVM μπορεί να γίνει κατανοητή χρησιμοποιώντας ένα παράδειγμα. Ας υποθέσουμε ότι έχουμε ένα σύνολο δεδομένων που έχει δύο ετικέτες. Θέλουμε έναν ταξινομητή που να μπορεί να ταξινομήσει ένα νέο σημείο είτε στην κατηγορία με πράσινο είτε με μπλε χρώμα.

Ο διαχωρισμός των δύο κλάσεων μπορεί να γίνει με μια ευθεία γραμμή. Όμως, υπάρχουν πολλές γραμμές που μπορούν να διαχωρίσουν αυτές τις κλάσεις. Όπως φαίνεται στο παρακάτω σχήμα και η κόκκινη και η πράσινη γραμμή μπορούν να διαχωρίσουν τις δύο κλάσεις.



Ο αλγόριθμος SVM βοηθά στην εύρεση της βέλτιστης γραμμής, η οποία ονομάζεται υπερεπίπεδο. Ο αλγόριθμος βρίσκει το πλησιέστερο σημείο των γραμμών και από τις δύο κλάσεις. Αυτά τα σημεία ονομάζονται διανύσματα υποστήριξης (support vectors). Η απόσταση μεταξύ των διανυσμάτων υποστήριξης και του υπερεπίπεδου ονομάζεται περιθώριο (margin). Στόχος του SVM είναι να μεγιστοποιήσει αυτό το περιθώριο. Το υπερεπίπεδο με μέγιστο περιθώριο ονομάζεται βέλτιστο υπερεπίπεδο και διαχωρίζει τις δύο κλάσεις με το βέλτιστο τρόπο.

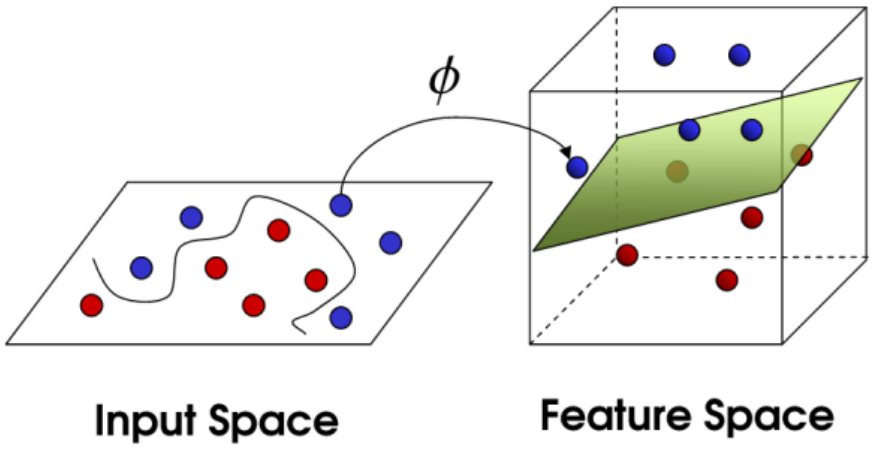



Όταν τα δεδομένα μας είναι διδιάστατα, τότε μιλάμε για διαχωρισμό τους από μια γραμμή, ενώ όταν είναι περισσότερων διαστάσεων μιλάμε για υπερεπίπεδο.

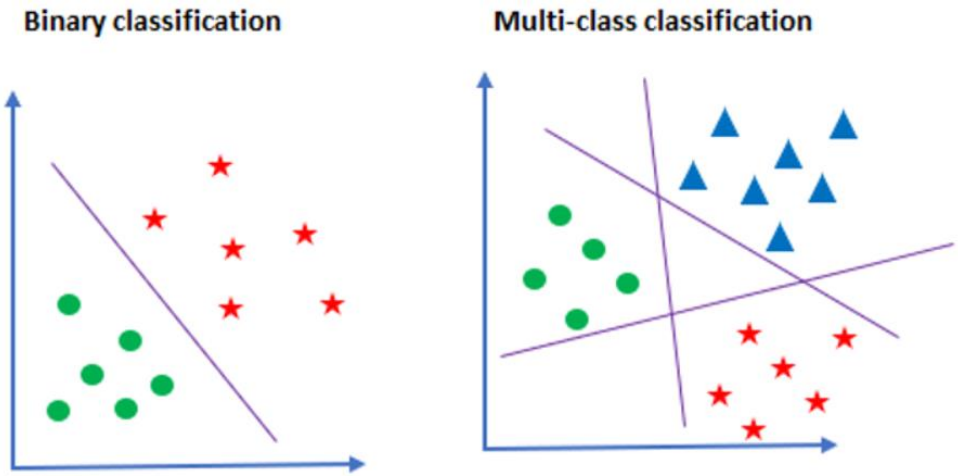
Ο αλγόριθμος SVM χρησιμοποιείται για προβλήματα δυαδικής ταξινόμησης (binary classification), προβλήματα δηλαδή που έχουμε μόνο δύο κατηγορίες.

Επίσης, για να είναι αποτελεσματικός ο SVM πρέπει τα δεδομένα να είναι γραμμικώς διαχωρίσιμα (linearly separable). Σε περίπτωση που τα δεδομένα δεν είναι γραμμικώς διαχωρίσιμα, δηλαδή δεν μπορούν να διαχωριστούν από μια ευθεία γραμμή, τότε μπορεί να εφαρμοστεί το επονομαζόμενο **SVM Kernel Trick**.

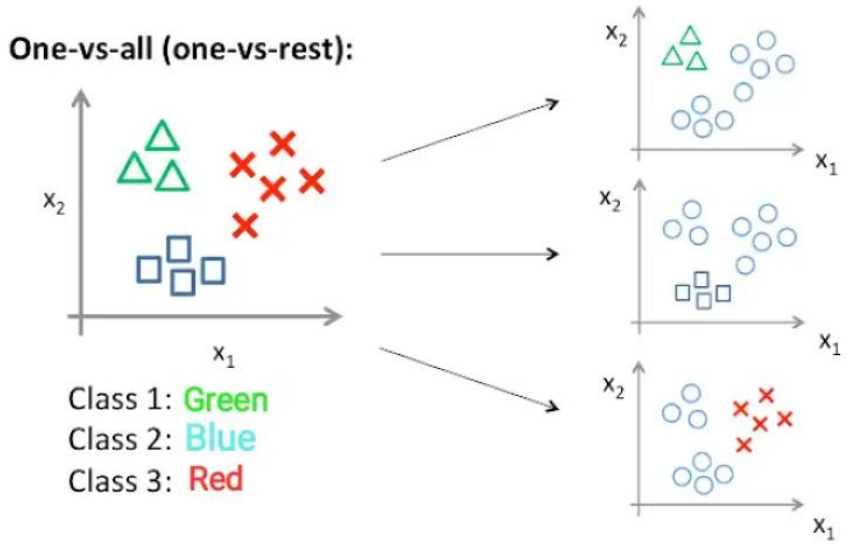
Τα δεδομένα προβάλλονται σε έναν χώρο περισσότερων διαστάσεων (προσθέτοντας χαρακτηριστικά). Ο γραμμικός διαχωρισμός στον επαυξημένο χώρο αντιστοιχεί σε μη γραμμικό διαχωρισμό στον αρχικό. Οπότε πλέον τα δεδομένα μπορούν να διαχωριστούν από ένα υπερεπίπεδο.



Σε περίπτωση που θέλουμε να χρησιμοποιήσουμε τον SVM σε προβλήματα ταξινόμησης πολλαπλών κλάσεων (multiclass classification) πρέπει να χρησιμοποιήσουμε την προσέγγιση ενός εναντίον όλων (one versus all approach).

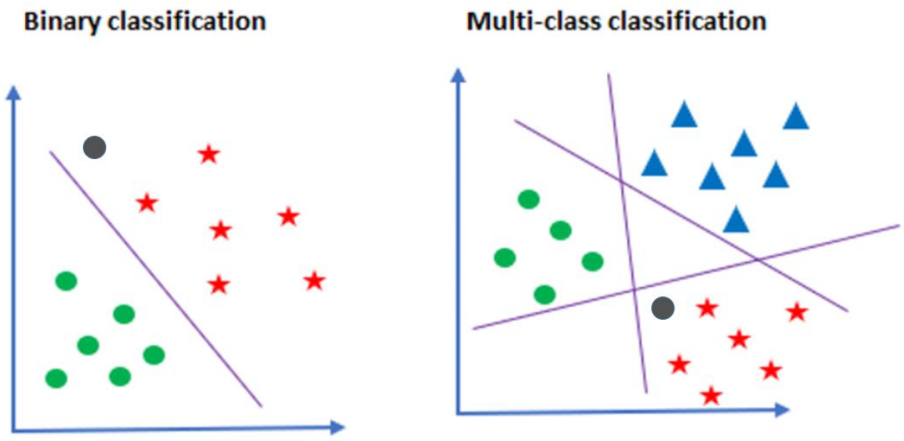



Στην προσέγγιση ενός εναντίον όλων, πρέπει να δημιουργήσουμε τόσους ταξινομητές SVM όσες και οι κλάσεις. Στην ουσία γίνεται δυαδική ταξινόμηση μιας κλάσης με όλες τις υπόλοιπες. Αυτό γίνεται για κάθε κλάση. Οπότε στο τέλος, αν έχουμε Ν κλάσεις, θα βρούμε Ν υπερεπίπεδα, από τα οποία το καθένα θα διαχωρίζει την κάθε κλάση από τις υπόλοιπες. Άρα, πλέον έχουν διαχωριστεί όλες οι κλάσεις.




Όπως φαίνεται στην παραπάνω εικόνα γίνονται 3 δυαδικές ταξινομήσεις (εύρεση 3 υπερεπιπέδων), οι οποίες στο τέλος συνενώνονται σε μία.

Κατά την αξιολόγηση του συστήματος, για να βρούμε σε ποια κατηγορία ανήκει μια εικόνα δοκιμής, πρέπει να δούμε σε ποια μεριά του υπερεπιπέδου βρίσκεται η εικόνα. Το υπερεπίπεδο χωρίζει τα δεδομένα σε δύο επίπεδα, οπότε σε όποια πλευρά βρίσκεται το νέο σημείο, σε αυτήν την κατηγορία θα ανατεθεί.




Στην παραπάνω εικόνα η μαύρη κουκίδα, αντιπροσωπεύει ένα νέο σημείο από το σύνολο δοκιμής. Και στις δύο παραπάνω περιπτώσεις το νέο σημείο ταξινομείται στην κατηγορία με τα κόκκινα αστέρια.

Από τα διανύσματα που έχουμε εξάγει, από τον σάκο λέξεων για το σύνολο των εικόνων εκπαίδευσης, εκπαιδεύουμε έναν ταξινομητή SVM, βρίσκουμε δηλαδή Ν βέλτιστα υπερεπίπεδα (έχουμε ταξινόμηση πολλαπλών κλάσεων). Στη συνέχεια, χρησιμοποιούμε τα διανύσματα από το σύνολο των εικόνων δοκιμής για να προβλέψουμε σε ποια κατηγορία ανήκουν οι εικόνες. Όπως και στον KNN μπορούμε να αξιολογήσουμε το σύστημα με κατάλληλες μετρικές.

# 6. Κώδικας για την υλοποίηση του συστήματος ταξινόμησης

## Λεξικό και κωδικοποίηση εικόνων

Αρχικά στο αρχείο HW3_vocabulary_index.py έχουμε βάλει να παράγονται 10 διαφορετικά λεξικά και να αποθηκεύονται στο φάκελο vocabulary_files. Επίσης, εξάγονται τα διανύσματα από την κωδικοποίηση των εικόνων εκπαίδευσης και δοκιμής και αποθηκεύονται στον ίδιο φάκελο, μαζί με τα paths των εικόνων.

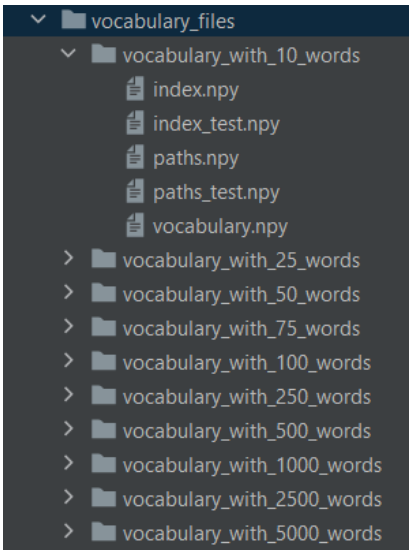
Η παραπάνω διαδικασία γίνεται για να μπορούμε να δοκιμάσουμε στον κώδικά μας παρακάτω και να αξιολογήσουμε την απόδοση του συστήματος για διαφορετικό πλήθος λέξεων.

Παρακάτω φαίνεται ο αριθμός λέξεων που δοκιμάστηκε για τα 10 λεξικά, τον οποίο παίρνουμε από μια λίστα.

words = [10, 25, 50, 75, 100, 250, 500, 1000, 2500, 5000]

Επίσης τα αρχεία για το λεξικό, τις κωδικοποιημένες εικόνες και τα paths, αποθηκεύονται με τα ίδια ονόματα κάθε φορά σε διαφορετικό όμως φάκελο μέσα στο φάκελο vocabulary_files.

Όπως φαίνεται στην παρακάτω εικόνα, ανάλογα με τον αριθμό λέξεων που θέλουμε να έχουμε, πρέπει να χρησιμοποιήσουμε τα αρχεία του αντίστοιχου φακέλου.




Όπως εξηγήσαμε παραπάνω για τη δημιουργία του λεξικού, αρχικά εξάγουμε τα σημεία ενδιαφέροντος και τους περιγραφείς από τις εικόνες με τη χρήση του αλγορίθμου SIFT. Στη συνέχεια με τον αλγόριθμο ομαδοποίησης Κ μέσων (K Means) ομαδοποιούνται οι περιγραφείς κάθε εικόνας εκπαίδευσης σε τόσες κατηγορίες όσες και οι λέξεις που επιλέγουμε κάθε φορά και παράγεται το λεξικό οπτικών λέξεων. Οι εικόνες εκπαίδευσης και δοκιμής κωδικοποιούνται (δημιουργείται από το ιστόγραμμα ένα διάνυσμα για κάθε εικόνα) με βάση το λεξικό.

Αλγόριθμος KNN

Παρακάτω φαίνονται οι συναρτήσεις που χρησιμοποιήθηκαν για την υλοποίηση του αλγορίθμου KNN, χωρίς τη χρήση της σχετικής OpenCV συνάρτησης (cv.ml.KNearest_create()).

Η πρώτη συνάρτηση υλοποίει τον αλγόριθμο για μία εικόνα δοκιμής και επιστρέφει έναν αριθμό, που αντιστοιχεί στην κατηγορία η οποία προβλέπεται ότι ανήκει η εικόνα. Η δεύτερη συνάρτηση καλεί τη πρώτη για όλες τις εικόνες δοκιμής και επιστρέφει μία λίστα με αριθμούς που αντιστοιχούν στις κατηγορίες που προβλέφθηκαν για τις εικόνες δοκιμής.

```
# For one image
def knn(bow_desc, labels, bow_descs, neigh_num):
    distances = np.sum((bow_desc - bow_descs) ** 2, axis=1)
    # distances = np.sqrt(np.sum((bow_desc - bow_descs) ** 2,
axis=1))  # euclidian distance
    # distances = np.sum(np.abs(bow_desc - bow_descs), axis=1) #
manhattan distance
    retrieved_ids = np.argsort(distances)
    ids = retrieved_ids.tolist()

    categories = [0, 0, 0, 0, 0]
    for i in range(neigh_num):
        categories[labels[ids[i]]] += 1
    result = categories.index(max(categories))

    return result


# For many images
def knn_test(test_descs, labels, bow_descs, neigh_num):
    results = []
    for i in range(len(test_descs)):
        result = knn(test_descs[i], labels, bow_descs, neigh_num)
        results.append(result)
    results = np.array(results, np.int32)

    return results
```

Επίσης, παράγουμε τις πραγματικές ετικέτες των εικόνων δοκιμής από το path τους.

Η παρακάτω συνάρτηση **accuracy(labels_test, predictions, show_all = True)** βρίσκει το ποσοστό των εικόνων που προβλέφθηκαν σωστά, συγκρίνοντας τις προβλέψεις μας με τις πραγματικές ετικέτες των εικόνων, οπότε μας επιστρέφει την ακρίβεια του συστήματος μας.

$$\text{Accuracy} = \frac{\text{Number of correct predictions}}{\text{Total number of predictions}}$$

Επίσης, μας τυπώνει την ακρίβεια του συστήματος και την ακρίβεια ανά κλάση.

Σε περίπτωση που δε θέλουμε να μας τυπώσει κάτι θέτουμε στο όρισμα της συνάρτησης show_all = False.

Η συνάρτηση **show_img_and_prediction(results, test_descs, img_paths_test)** προβάλει τις εικόνες κάθε κλάσης σε διαφορετικό παράθυρο μία μία και τυπώνει την πρόβλεψη που έκανε ο αλγόριθμος, μαζί με τον αύξοντα αριθμό της εικόνας

δοκιμής. Οπότε, ανάλογα με το τι τυπώθηκε και με την εικόνα που βλέπουμε μπορούμε να δούμε ακριβώς σε ποια εικόνα έκανε λάθος πρόβλεψη ο αλγόριθμος.

```
def show_img_and_prediction(results, test_descs, img_paths_test):
    classes = ["motorbike", "school-bus", "touring-bike", "airplane",
"car-side"]
    num_cl = [0, 0, 0, 0, 0]
    n = 5
    for i in range(n):
        for j in range(len(labels_test)):
            if labels_test[j] == i:
                num_cl[i] += 1

    for c in range(len(num_cl)):
        name = "\nClass" + str(c)
        print(name)

        cv.namedWindow(name, cv.WINDOW_NORMAL)
        for i in range(len(labels_test)):
            if labels_test[i] == c:
                prediction = classes[results[i]]
                print("Test image " + str(i+1) + ": " + "It is a " +
prediction)
                test_img = cv.imread(img_paths_test[i])
                cv.imshow(name, test_img)
                cv.waitKey(0)


def accuracy(labels_test, predictions, show_all = True):
    # Total
    matches = np.count_nonzero(labels_test == predictions)
    total_acc = matches / len(labels_test) * 100
    total_acc = round(total_acc, 2)
    if (show_all):
        print("Correctly predicted: " + str(matches) + " images out
of " + str(len(labels_test)) + " images")
        print("Total accuracy: " + str(total_acc) + "%\n")

    # For every class
    num_cl = [0, 0, 0, 0, 0]
    n = 5
    matches = [0, 0, 0, 0, 0]
    for i in range(n):
        for j in range(len(labels_test)):
            if labels_test[j] == i:
                if labels_test[j] == predictions[j]:
                    matches[i] += 1
                num_cl[i] += 1

    acc = []
    for i in range(n):
        acc.append(round(matches[i] / num_cl[i] * 100, 2))

    if (show_all):
        for i in range(n):
            print("Correctly predicted images for class" + str(i) +
": " + str(matches[i])+ " out of " + str(num_cl[i]) + " images")
            print("Accuracy for class"+ str(i) + " is: " +
str(acc[i]) + "%\n")

    return total_acc
```

Στο αρχείο HW3_KNN.py τρέχουμε τον αλγόριθμο KNN για τα 10 διαφορετικά λεξικά. Για κάθε λεξικό ο αλγόριθμος τρέχει 150 φορές, για αριθμό γειτόνων 1 έως 150. Κρατάμε την ακρίβεια για κάθε αριθμό γειτόνων σε μια λίστα και στο τέλος επιλέγουμε την μέγιστη ακρίβεια και βρίσκουμε τον αριθμό γειτόνων για τον οποίο προέκυψε αυτή.

Στο τέλος, έχουμε κρατήσει 10 ακρίβειες (με τον αριθμό γειτόνων για τις οποίες προέκυψαν τον έχουμε κρατήσει σε μια άλλη λίστα). Οπότε πλέον επιλέγουμε τη μέγιστη ακρίβεια με τον αριθμό γειτόνων και τις λέξεις για τις οποίες προέκυψε.

Άρα, έχουμε διερευνήσει και βρει τη μέγιστη ακρίβεια όσον αφορά τον αριθμό λέξεων του λεξικού και τον αριθμό γειτόνων.

Αν η ακρίβεια για δύο διαφορετικούς αριθμούς γειτόνων είναι ίδια, τότε κρατάμε αυτή με το μικρότερο αριθμό γειτόνων. Αν η ακρίβεια για δύο διαφορετικούς αριθμούς λέξεων είναι ίδια, τότε κρατάμε αυτή με το μικρότερο αριθμό λέξεων. Με τον τρόπο αυτό κρατάμε την μέγιστη ακρίβεια που προκύπτει με το ελάχιστο υπολογιστικό κόστος.

Αλγόριθμος SVM

Με το παρακάτω κομμάτι κώδικα υλοποιείται η προσέγγιση ενός εναντίον όλων για την εκπαίδευση Ν ταξινομητών. Έχουμε 5 κλάσεις άρα εκπαιδεύουμε 5 ταξινομητές.

Πρέπει να προσέξουμε ότι οι ετικέτες των εικόνων εκπαίδευσης δεν μπορούν να δοθούν αυτούσιες στον αλγόριθμο SVM. Πρέπει κάθε φορά οι ετικέτες μιας κλάσης να είναι 1 και όλων των υπολοίπων 0 και όχι να παίρνουν οι ετικέτες αριθμούς 0 έως 4 ανάλογα με την κλάση. Οπότε γίνεται η απαραίτητη επεξεργασία των ετικετών.

```
# Train SVM one versus all approach
for i in range(n):
    print('Training SVM' + str(i))
    svm = cv.ml.SVM_create()
    svm.setType(cv.ml.SVM_C_SVC)
    svm.setKernel(value)
    svm.setTermCriteria((cv.TERM_CRITERIA_COUNT, 100, 1.e-06))

    svm.trainAuto(bow_descs, cv.ml.ROW_SAMPLE, labels_train_svm[i])
```

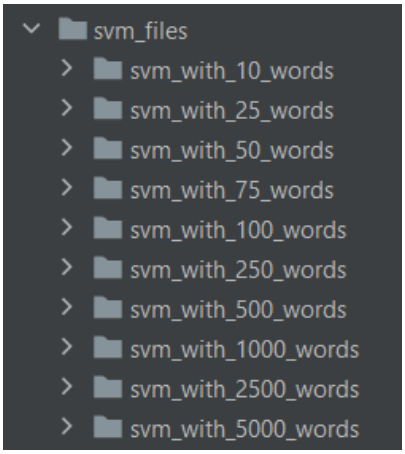
Για βελτίωση της ακρίβειας σκεφτήκαμε να αλλάξουμε τύπο kernel. Οπότε, τρέχουμε τον αλγόριθμο για τα 5 παρακάτω διαφορετικά kernel για κάθε λεξικό.

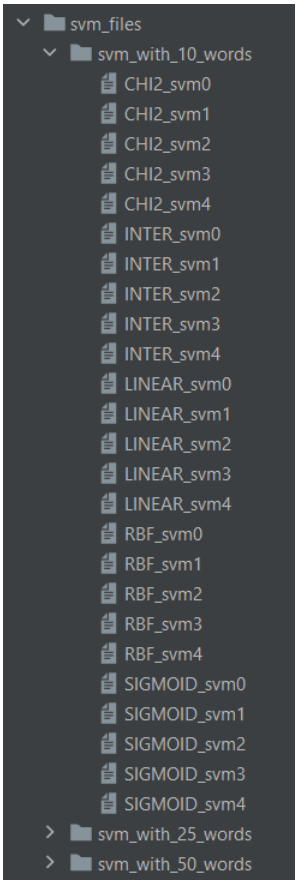
```
kernel_types = {"RBF":cv.ml.SVM_RBF, "LINEAR":cv.ml.SVM_LINEAR,
"SIGMOID":cv.ml.SVM_SIGMOID, "CHI2":cv.ml.SVM_CHI2,
"INTER":cv.ml.SVM_INTER}
```

Οπότε για κάθε SVM one versus all εκπαιδεύουμε 5 ταξινομητές.

Εκτελούμε SVM one versus all 5 φορές για τα διαφορετικά kernel και για τα 10 λεξικά. Οπότε σύνολο εκτελείται SVM one versus all 50 φορές. Άρα εκπαιδεύουμε

50 * 5 = 250 ταξινομητές σύνολο, τους οποίους αποθηκεύουμε στον φάκελο svm_files.




Οι ταξινομητές για κάθε λεξικό αποθηκεύονται με τα ίδια ονόματα σε διαφορετικούς φακέλους.

Οι ταξινομητές για διαφορετικό τύπο kernel για το ίδιο λεξικό αποθηκεύονται στον ίδιο φάκελο με διαφορετικό όνομα.

Η τιμή 0, 1, 2, 3, 4 στο όνομα των αρχείων αναφέρεται στην κλάση one versus all του ταξινομητή.

Για παράδειγμα το αρχείο CHI2_svm0 καταλαβαίνουμε ότι αναφέρεται σε τύπο kernel CHI2 και στον ταξινομητή της κλάσης 0 με όλες τις υπόλοιπες.

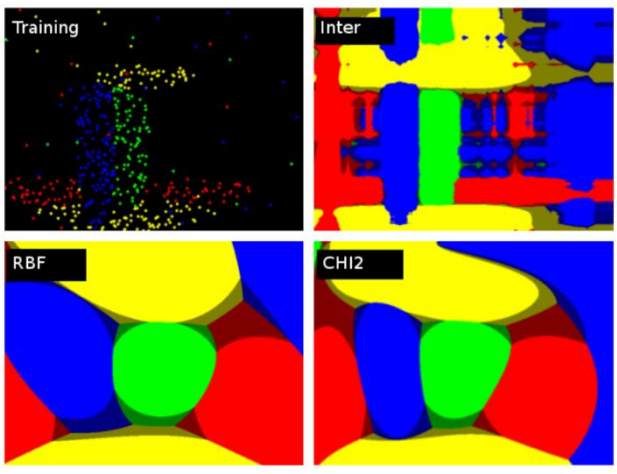
Όλα τα παραπάνω γίνονται στο αρχείο HW3_SVM_TRAIN.py.

Παρακάτω μπορούμε να δούμε τους διαφορετικούς τύπους kernel που μπορούν να χρησιμοποιηθούν στην opencv.

| | |
|---|---|
| LINEAR | Linear kernel. No mapping is done, linear discrimination (or regression) is done in the original feature space. It is the fastest option. $K(x_i, x_j) = x_i^T x_j$. |
| POLY | Polynomial kernel: $K(x_i, x_j) = (\gamma x_i^T x_j + coef0)^{degree}, \gamma > 0$. |
| RBF | Radial basis function (RBF), a good choice in most cases. $K(x_i, x_j) = e^{-\gamma \|x_i - x_j\|^2}, \gamma > 0$. |
| SIGMOID | Sigmoid kernel: $K(x_i, x_j) = \tanh(\gamma x_i^T x_j + coef0)$. |
| CHI2 | Exponential Chi2 kernel, similar to the RBF kernel: $K(x_i, x_j) = e^{-\gamma \chi^2(x_i, x_j)}, \chi^2(x_i, x_j) = (x_i - x_j)^2/(x_i + x_j), \gamma > 0$. |
| INTER | Histogram intersection kernel. A fast kernel. $K(x_i, x_j) = min(x_i, x_j)$. |

Μια σύγκριση διαφορετικών kernel σε 2D διάσταση φαίνεται στη διπλανή εικόνα.

Το χρώμα απεικονίζει την κλάση με το μέγιστο σκορ. Φωτεινό χρώμα σημαίνει μέγιστο σκορ > 0, σκούρο χρώμα σημαίνει μέγιστο σκορ < 0.



Η πρώτη συνάρτηση υλοποίει τον αλγόριθμο για μία εικόνα δοκιμής και επιστρέφει έναν αριθμό, που αντιστοιχεί στην κατηγορία η οποία προβλέπεται ότι ανήκει η εικόνα. Η δεύτερη συνάρτηση καλεί τη πρώτη για όλες τις εικόνες δοκιμής και επιστρέφει μία λίστα με αριθμούς που αντιστοιχούν στις κατηγορίες που προβλέφθηκαν για τις εικόνες δοκιμής.

```
# For one image
def svm_func(bow_desc, labels, bow_descs, svm_load):
    response = []
    for i in range(len(svm_load)):
        r = svm_load[i].predict(bow_desc,
flags=cv.ml.STAT_MODEL_RAW_OUTPUT)
        response.append(r[1])

    result = np.argmin(response)

    return result


# For many images
def svm_test(test_descs, labels, bow_descs, svm_load):
    results = []
    for i in range(len(test_descs)):
        test = np.resize(test_descs[i], (1, test_descs[i].shape[0]))
        result = svm_func(test, labels, bow_descs, svm_load)
        results.append(result)
    results = np.array(results, np.int32)

    return results
```

Στο αρχείο HW3_SVM_TEST.py τρέχουμε τον αλγόριθμο SVM για τα 10 διαφορετικά λεξικά. Για κάθε λεξικό ο αλγόριθμος τρέχει 5 φορές, μία για κάθε kernel.

Για κάθε λεξικό οπτικών λέξεων τυπώνουμε το kernel που χρησιμοποιήθηκε και την ακρίβεια που προέκυψε. Στη συνέχεια κρατάμε ένα λεξικό με κλειδιά τους τύπους kernel που χρησιμοποιήθηκαν και τιμές την ακρίβεια τους. Στο τέλος βρίσκουμε τη μέγιστη ακρίβεια και το αντίστοιχο kernel και τα αποθηκεύουμε σε δύο διαφορετικές λίστες.

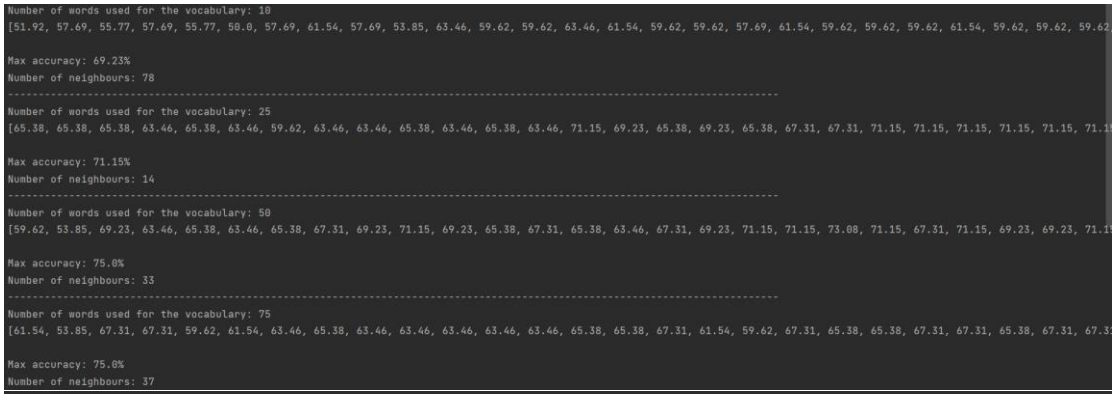
Στο τέλος, έχουμε κρατήσει 10 ακρίβειες και 10 kernel. Οπότε πλέον επιλέγουμε τη μέγιστη ακρίβεια με το αντίστοιχο kernel και τις λέξεις για τις οποίες προέκυψε.

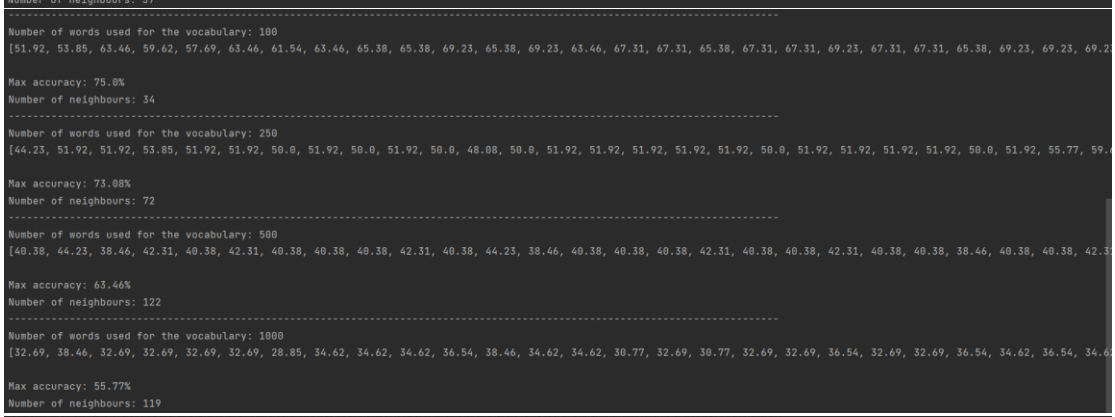
Άρα, έχουμε διερευνήσει και βρει τη μέγιστη ακρίβεια όσον αφορά τον αριθμό λέξεων του λεξικού και τον τύπο kernel.

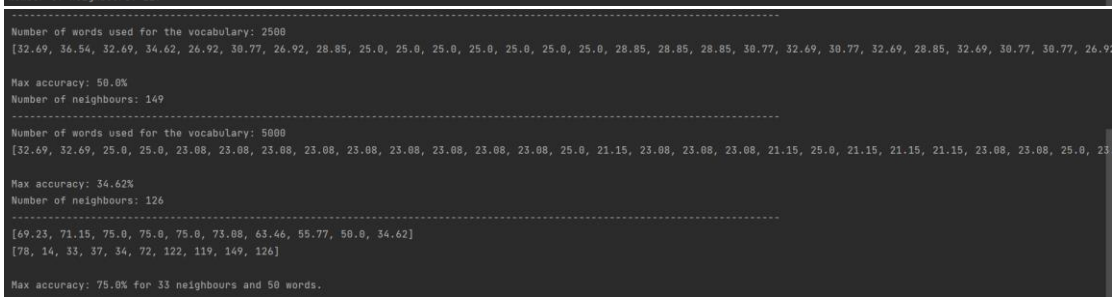
## 7. Αξιολόγηση του συστήματος

Αποτελέσματα KNN

Παρακάτω βλέπουμε τα αποτελέσματα που τυπώνονται.








```
Best accuracy
Correctly predicted: 39 images out of 52 images
Total accuracy: 75.0%

Correctly predicted images for class0: 3 out of 10 images
Accuracy for class0 is: 30.0%

Correctly predicted images for class1: 7 out of 9 images
Accuracy for class1 is: 77.78%

Correctly predicted images for class2: 10 out of 11 images
Accuracy for class2 is: 90.91%

Correctly predicted images for class3: 11 out of 11 images
Accuracy for class3 is: 100.0%

Correctly predicted images for class4: 8 out of 11 images
Accuracy for class4 is: 72.73%
```

Η μέγιστη ακρίβεια είναι **75%** και προκύπτει για **50 οπτικές λέξεις** και για **33 γείτονες**.

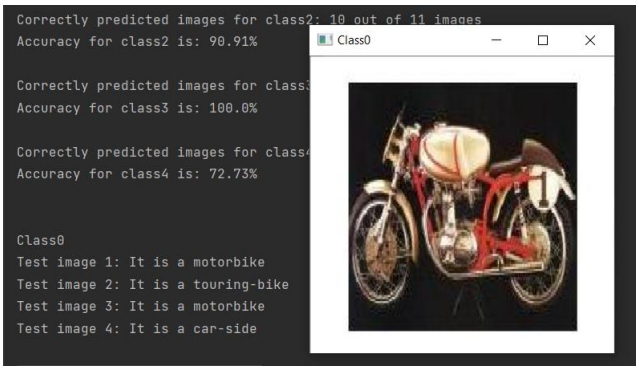
Οι παρακάτω εκτυπώσεις γίνονται μόνο για την μέγιστη ακρίβεια που έχουμε βρει.

| Οπτικές λέξεις = 50, Γείτονες = 33 | |
|---|---|
| Συνολική ακρίβεια | 75% |
| Κλάση 0 (motorbike) | 30% |
| Κλάση 1 (school-bus) | 77.78% |
| Κλάση 2 (touring-bike) | 90.91% |
| Κλάση 3 (airplane) | 100.0% |
| Κλάση 4 (car-side) | 72.73% |

Η μεγαλύτερη αποτυχία γίνεται στην κλάση 0, δηλαδή την κλάση με τις μοτοσυκλέτες.

Για κάθε εικόνα δοκιμής μπορούμε να δούμε την εικόνα και να τυπώσουμε την πρόβλεψη, ώστε να δούμε σε ποιες εικόνες έχουμε κάνει λάθος πρόβλεψη.

Παρακάτω βλέπουμε ότι στην εικόνα 1 έχει γίνει σωστή πρόβλεψη, ενώ η εικόνα 4 έχει αναγνωριστεί λανθασμένα σαν αυτοκίνητο.

Παρακάτω βλέπουμε όλα τα αποτελέσματα της εκτύπωσης.

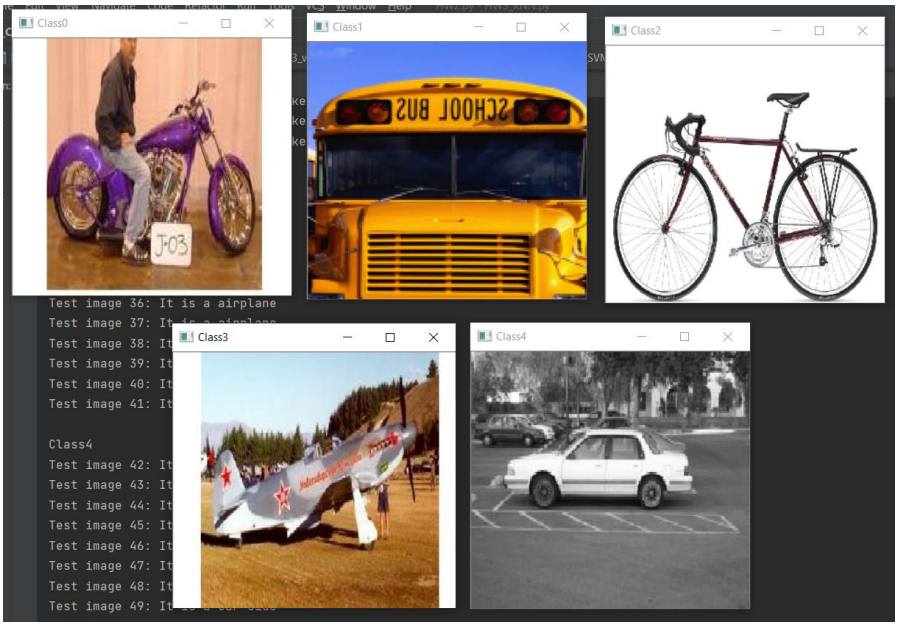
```
Class0
Test image 1: It is a motorbike
Test image 2: It is a touring-bike
Test image 3: It is a motorbike
Test image 4: It is a car-side
Test image 5: It is a motorbike
Test image 6: It is a touring-bike
Test image 7: It is a touring-bike
Test image 8: It is a school-bus
Test image 9: It is a airplane
Test image 10: It is a school-bus

Class1
Test image 11: It is a school-bus
Test image 12: It is a school-bus
Test image 13: It is a school-bus
Test image 14: It is a car-side
Test image 15: It is a school-bus
Test image 16: It is a school-bus
Test image 17: It is a school-bus
Test image 18: It is a car-side
Test image 19: It is a school-bus
```

```
Class2
Test image 20: It is a touring-bike
Test image 21: It is a touring-bike
Test image 22: It is a touring-bike
Test image 23: It is a touring-bike
Test image 24: It is a touring-bike
Test image 25: It is a touring-bike
Test image 26: It is a car-side
Test image 27: It is a touring-bike
Test image 28: It is a touring-bike
Test image 29: It is a touring-bike
Test image 30: It is a touring-bike

Class3
Test image 31: It is a airplane
Test image 32: It is a airplane
Test image 33: It is a airplane
Test image 34: It is a airplane
Test image 35: It is a airplane
Test image 36: It is a airplane
Test image 37: It is a airplane
Test image 38: It is a airplane
Test image 39: It is a airplane
Test image 40: It is a airplane
Test image 41: It is a airplane
```

```
Class4
Test image 42: It is a car-side
Test image 43: It is a car-side
Test image 44: It is a car-side
Test image 45: It is a car-side
Test image 46: It is a car-side
Test image 47: It is a school-bus
Test image 48: It is a car-side
Test image 49: It is a car-side
Test image 50: It is a school-bus
Test image 51: It is a airplane
Test image 52: It is a car-side
```



Αποτελέσματα SVM

Παρακάτω βλέπουμε τα αποτελέσματα που τυπώνονται.

```
Number of words used for the vocabulary: 10

Kernel: RBF
Accuracy: 50.0%

Kernel: LINEAR
Accuracy: 57.69%

Kernel: SIGMOID
Accuracy: 21.15%

Kernel: CHI2
Accuracy: 59.62%

Kernel: INTER
Accuracy: 53.85%

{'RBF': 50.0, 'LINEAR': 57.69, 'SIGMOID': 21.15, 'CHI2': 59.62, 'INTER': 53.85}

Max accuracy: 59.62%
Kernel: CHI2
--------------------------------------------------------------------------------------------------------------------------------
```

```
Number of words used for the vocabulary: 25

Kernel: RBF
Accuracy: 75.0%

Kernel: LINEAR
Accuracy: 73.08%

Kernel: SIGMOID
Accuracy: 21.15%

Kernel: CHI2
Accuracy: 76.92%

Kernel: INTER
Accuracy: 78.85%

{'RBF': 75.0, 'LINEAR': 73.08, 'SIGMOID': 21.15, 'CHI2': 76.92, 'INTER': 78.85}

Max accuracy: 78.85%
Kernel: INTER
--------------------------------------------------------------------------------------------------------------------------------
```

```
Number of words used for the vocabulary: 50

Kernel: RBF
Accuracy: 78.85%

Kernel: LINEAR
Accuracy: 76.92%

Kernel: SIGMOID
Accuracy: 21.15%

Kernel: CHI2
Accuracy: 82.69%

Kernel: INTER
Accuracy: 86.54%

{'RBF': 78.85, 'LINEAR': 76.92, 'SIGMOID': 21.15, 'CHI2': 82.69, 'INTER': 86.54}

Max accuracy: 86.54%
Kernel: INTER
--------------------------------------------------------------------------------------------------------------------------------
```

```
Number of words used for the vocabulary: 75

Kernel: RBF
Accuracy: 76.92%

Kernel: LINEAR
Accuracy: 67.31%

Kernel: SIGMOID
Accuracy: 21.15%

Kernel: CHI2
Accuracy: 80.77%

Kernel: INTER
Accuracy: 78.85%

{'RBF': 76.92, 'LINEAR': 67.31, 'SIGMOID': 21.15, 'CHI2': 80.77, 'INTER': 78.85}

Max accuracy: 80.77%
Kernel: CHI2
--------------------------------------------------------------------------------------------------------------------------------
```

```
Number of words used for the vocabulary: 100

Kernel: RBF
Accuracy: 88.46%

Kernel: LINEAR
Accuracy: 88.46%

Kernel: SIGMOID
Accuracy: 21.15%

Kernel: CHI2
Accuracy: 86.54%

Kernel: INTER
Accuracy: 86.54%

{'RBF': 88.46, 'LINEAR': 88.46, 'SIGMOID': 21.15, 'CHI2': 86.54, 'INTER': 86.54}

Max accuracy: 88.46%
Kernel: RBF
--------------------------------------------------------------------------------------------------------------------------------
```

```
Number of words used for the vocabulary: 250

Kernel: RBF
Accuracy: 92.31%

Kernel: LINEAR
Accuracy: 92.31%

Kernel: SIGMOID
Accuracy: 21.15%

Kernel: CHI2
Accuracy: 88.46%

Kernel: INTER
Accuracy: 90.38%

{'RBF': 92.31, 'LINEAR': 92.31, 'SIGMOID': 21.15, 'CHI2': 88.46, 'INTER': 90.38}

Max accuracy: 92.31%
Kernel: RBF
--------------------------------------------------------------------------------------------------------------------------------
```

```
Number of words used for the vocabulary: 500

Kernel: RBF
Accuracy: 86.54%

Kernel: LINEAR
Accuracy: 88.46%

Kernel: SIGMOID
Accuracy: 21.15%

Kernel: CHI2
Accuracy: 88.46%

Kernel: INTER
Accuracy: 88.46%

{'RBF': 86.54, 'LINEAR': 88.46, 'SIGMOID': 21.15, 'CHI2': 88.46, 'INTER': 88.46}

Max accuracy: 88.46%
Kernel: LINEAR
```

```
Number of words used for the vocabulary: 1000

Kernel: RBF
Accuracy: 88.46%

Kernel: LINEAR
Accuracy: 88.46%

Kernel: SIGMOID
Accuracy: 21.15%

Kernel: CHI2
Accuracy: 92.31%

Kernel: INTER
Accuracy: 92.31%

{'RBF': 88.46, 'LINEAR': 88.46, 'SIGMOID': 21.15, 'CHI2': 92.31, 'INTER': 92.31}

Max accuracy: 92.31%
Kernel: CHI2
```

```
Number of words used for the vocabulary: 2500

Kernel: RBF
Accuracy: 88.46%

Kernel: LINEAR
Accuracy: 88.46%

Kernel: SIGMOID
Accuracy: 21.15%

Kernel: CHI2
Accuracy: 90.38%

Kernel: INTER
Accuracy: 88.46%

{'RBF': 88.46, 'LINEAR': 88.46, 'SIGMOID': 21.15, 'CHI2': 90.38, 'INTER': 88.46}

Max accuracy: 90.38%
Kernel: CHI2
```

```
Number of words used for the vocabulary: 5000

Kernel: RBF
Accuracy: 90.38%

Kernel: LINEAR
Accuracy: 88.46%

Kernel: SIGMOID
Accuracy: 21.15%

Kernel: CHI2
Accuracy: 90.38%

Kernel: INTER
Accuracy: 92.31%

{'RBF': 90.38, 'LINEAR': 88.46, 'SIGMOID': 21.15, 'CHI2': 90.38, 'INTER': 92.31}

Max accuracy: 92.31%
Kernel: INTER
```

```
[59.62, 78.85, 86.54, 80.77, 88.46, 92.31, 88.46, 92.31, 90.38, 92.31]
['CHI2', 'INTER', 'INTER', 'CHI2', 'RBF', 'RBF', 'LINEAR', 'CHI2', 'CHI2', 'INTER']
Max accuracy: 92.31% for kernel RBF and 250 words.
```

```
Best accuracy
Correctly predicted: 48 images out of 52 images
Total accuracy: 92.31%

Correctly predicted images for class0: 9 out of 10 images
Accuracy for class0 is: 90.0%

Correctly predicted images for class1: 9 out of 9 images
Accuracy for class1 is: 100.0%

Correctly predicted images for class2: 10 out of 11 images
Accuracy for class2 is: 90.91%

Correctly predicted images for class3: 10 out of 11 images
Accuracy for class3 is: 90.91%

Correctly predicted images for class4: 10 out of 11 images
Accuracy for class4 is: 90.91%
```

Η μέγιστη ακρίβεια είναι **92.31%** και προκύπτει για **250 οπτικές λέξεις** και για **RBF τύπο kernel**. Ακρίβεια 92.31% προκύπτει επίσης για 1000 και 5000, όμως προτιμάμε τις λιγότερες λέξεις, με το ελάχιστο υπολογιστικό κόστος.

Οι παρακάτω εκτυπώσεις γίνονται μόνο για την μέγιστη ακρίβεια που έχουμε βρει.

| **Οπτικές λέξεις = 250, Kernel = RBF** | |
|---|---|
| **Συνολική ακρίβεια** | 92.31% |
| **Κλάση 0 (motorbike)** | 90.0% |
| **Κλάση 1 (school-bus)** | 100.0% |
| **Κλάση 2 (touring-bike)** | 90.91% |
| **Κλάση 3 (airplane)** | 90.91% |
| **Κλάση 4 (car-side)** | 90.91% |

Για κάθε εικόνα δοκιμής μπορούμε να δούμε την εικόνα και να τυπώσουμε την πρόβλεψη, ώστε να δούμε σε ποιες εικόνες έχουμε κάνει λάθος πρόβλεψη.

Παρακάτω βλέπουμε όλα τα αποτελέσματα της εκτύπωσης.

```
Class0
Test image 1: It is a motorbike
Test image 2: It is a motorbike
Test image 3: It is a motorbike
Test image 4: It is a motorbike
Test image 5: It is a motorbike
Test image 6: It is a motorbike
Test image 7: It is a motorbike
Test image 8: It is a motorbike
Test image 9: It is a airplane
Test image 10: It is a motorbike

Class1
Test image 11: It is a school-bus
Test image 12: It is a school-bus
Test image 13: It is a school-bus
Test image 14: It is a school-bus
Test image 15: It is a school-bus
Test image 16: It is a school-bus
Test image 17: It is a school-bus
Test image 18: It is a school-bus
Test image 19: It is a school-bus
```

```
Class2
Test image 20: It is a touring-bike
Test image 21: It is a touring-bike
Test image 22: It is a touring-bike
Test image 23: It is a touring-bike
Test image 24: It is a touring-bike
Test image 25: It is a touring-bike
Test image 26: It is a car-side
Test image 27: It is a touring-bike
Test image 28: It is a touring-bike
Test image 29: It is a touring-bike
Test image 30: It is a touring-bike

Class3
Test image 31: It is a airplane
Test image 32: It is a airplane
Test image 33: It is a airplane
Test image 34: It is a airplane
Test image 35: It is a school-bus
Test image 36: It is a airplane
Test image 37: It is a airplane
Test image 38: It is a airplane
Test image 39: It is a airplane
Test image 40: It is a airplane
Test image 41: It is a airplane
```

```
Class4
Test image 42: It is a car-side
Test image 43: It is a car-side
Test image 44: It is a car-side
Test image 45: It is a car-side
Test image 46: It is a car-side
Test image 47: It is a car-side
Test image 48: It is a car-side
Test image 49: It is a car-side
Test image 50: It is a car-side
Test image 51: It is a airplane
Test image 52: It is a car-side
```

## 8. Επίδραση διαφόρων παραμέτρων στην ακρίβεια του συστήματος

Εφόσον έχουμε βρει τις βέλτιστες παραμέτρους για τους δύο αλγορίθμους, τις αποθηκεύουμε σε αρχεία txt στον φάκελο best_parameters και διαβάζουμε αυτές τις παραμέτρους στα αρχεία KNN_best.py και SVM_best.py. Οπότε πλέον αυτά τα αρχεία δεν περιλαμβάνουν τη διερεύνηση των παραμέτρων και υπολογίζουν κατευθείαν τις προβλέψεις για τις εικόνες δοκιμής.

<u>KNN</u>

Παρατηρούμε ότι για μικρό αριθμό οπτικών λέξεων (10 λέξεις) η ακρίβεια είναι σχετικά χαμηλή. Όσο οι λέξεις αυξάνονται, η ακρίβεια αυξάνεται (25, 50, 75, 100, 250 λέξεις). Όμως αν αυξηθεί πολύ ο αριθμός των λέξεων (500, 1000, 2500, 5000 λέξεις) η ακρίβεια μειώνεται.

Όσον αφορά τον αριθμό γειτόνων όταν έχουμε πολύ μεγάλο ή πολύ μικρό αριθμό η ακρίβεια μειώνεται.

Βέβαια, όσο αυξάνονται οι οπτικές λέξεις, τόσο αυξάνεται ο αριθμός των γειτόνων για τον οποίο προκύπτει η μέγιστη ακρίβεια.

| Οπτικές λέξεις | Ακρίβεια | Γείτονες |
|---|---|---|
| 10 | 69.23% | 78 |
| 25 | 71.15% | 14 |
| **50** | **75.0%** | **33** |
| 75 | 75.0% | 37 |
| 100 | 75.0% | 34 |
| 250 | 73.08% | 72 |
| 500 | 63.46% | 122 |
| 1000 | 55.77% | 119 |
| 2500 | 50.0% | 149 |
| 5000 | 34.62% | 126 |

SVM

Παρατηρούμε ότι όσο οι λέξεις αυξάνονται, η ακρίβεια αυξάνεται, γιατί ο αλγόριθμος εκπαιδεύεται περισσότερο (αυξάνεται η πολυπλοκότητα).

| Οπτικές λέξεις | Ακρίβεια | Kernel |
|---|---|---|
| 10 | 59.62% | CHI2 |
| 25 | 78.85% | INTER |
| 50 | 86.54% | INTER |
| 75 | 80.77% | CHI2 |
| 100 | 88.46% | RBF |
| **250** | **92.31%** | **RBF** |
| 500 | 88.46% | LINEAR |
| 1000 | 92.31% | CHI2 |
| 2500 | 90.38% | CHI2 |
| 5000 | 92.31% | INTER |

## 9. Αστοχίες του συστήματος

KNN

Η μεγαλύτερη αποτυχία γίνεται στην κλάση 0, δηλαδή την κλάση με τις μοτοσυκλέτες (30.0% ακρίβεια). Αυτό συμβαίνει επειδή πολλές μηχανές μοιάζουν με ποδήλατα, με αποτέλεσμα ο αλγόριθμος να τις αναγνωρίζει λανθασμένα σαν ποδήλατα (3 εικόνες). Επίσης, σε όλες τις εικόνες που υπάρχει κάποιος άνθρωπος δίπλα ή πάνω στην μηχανή, ο αλγόριθμος τις αναγνωρίζει σαν σχολικά λεωφορεία, πιθανότατα επειδή τα σχολικά λεωφορεία έχουν μεγάλο τζάμι και φαίνεται λίγο ο οδηγός ή και οι επιβάτες (2 εικόνες). Μία εικόνα αναγνωρίζεται λανθασμένα σαν αεροπλάνο, μάλλον επειδή υπάρχει στο φόντο η θάλασσα που είναι μπλε και ο αλγόριθμος νομίζει ότι είναι ουρανός. Τέλος, μία εικόνα αναγνωρίζεται λανθασμένα σαν αυτοκίνητο, ίσως επειδή υπάρχει ένα κράνος πάνω στη σέλα της μηχανής που καλύπτει κάποιο μέρος της.

SVM

Δεν υπάρχει κάποια κλάση με ιδιαίτερη αποτυχία. Οπότε οι λανθασμένες προβλέψεις μπορεί να οφείλονται σε κάποιες ομοιότητες μεταξύ των εικόνων.

# 10. Βιβλιογραφία


1. https://towardsdatascience.com/bag-of-visual-words-in-a-nutshell-9ceea97ce0fb
2. https://www.mathworks.com/help/vision/ug/image-classification-with-bag-of-visual-words.html
3. https://www.mathworks.com/matlabcentral/mlc-downloads/downloads/submissions/62217/versions/2/screenshot.png
4. https://www.javatpoint.com/k-means-clustering-algorithm-in-machine-learning
5. https://www.javatpoint.com/k-nearest-neighbor-algorithm-for-machine-learning
6. https://www.javatpoint.com/machine-learning-support-vector-machine-algorithm
7. https://towardsdatascience.com/multi-class-classification-one-vs-all-one-vs-one-94daed32a87b
8. https://docs.opencv.org/3.4/d1/d2d/classcv_1_1ml_1_1SVM.html
9. Διαφάνειες μαθήματος “ Όραση Υπολογιστών”, Δημοκρίτειο Πανεπιστήμιο Θράκης