ΠΑΝΕΠΙΣΤΗΜΙΟ ΔΥΤΙΚΗΣ ΑΤΤΙΚΗΣ

ΣΧΟΛΗ ΤΕΧΝΟΛΟΓΙΚΩΝ ΕΦΑΡΜΟΓΩΝ

ΤΜΗΜΑ ΑΥΤΟΜΑΤΙΣΜΟΥ

ΠΤΥΧΙΑΚΗ ΕΡΓΑΣΙΑ

Ανάπτυξη εφαρμογής με σκοπό την διαχείριση και επεξεργασία πολλαπλών αρχείων csv, με την χρήση της γλώσσας προγραμματισμού Python

Ονοματεπώνυμο: Ιωάννης Ηλιόπουλος

Αριθμός Μητρώου: 44582

Επιβλέπων Καθηγητής: κ. Χρήστος Δρόσος

Αιγάλεω, Νοέμβριος 2019

ΠΑΝΕΠΙΣΤΗΜΙΟ ΔΥΤΙΚΗΣ ΑΤΤΙΚΗΣ

ΔΗΛΩΣΗ ΣΥΓΓΡΑΦΕΑ ΠΤΥΧΙΑΚΗΣ ΕΡΓΑΣΙΑΣ

Ο κάτωθι υπογεγραμμένος ……Ηλιόπουλος…Ιωάννης…, του ……Γεωργίου.., φοιτητής του

Τμήματος .....Μηχανικών βιομηχανικής σχεδίασης και παραγωγής.... του Πανεπιστημίου Δυτικής Αττικής, πριν αναλάβω την εκπόνηση της Πτυχιακής Εργασίας μου, δηλώνω ότι ενημερώθηκα για τα παρακάτω:

«Η Πτυχιακή Εργασία (Π.Ε) αποτελεί προϊόν πνευματικής ιδιοκτησίας τόσο του συγγραφέα, όσο και του Ιδρύματος και θα πρέπει να έχει μοναδικό χαρακτήρα και πρωτότυπο περιεχόμενο.

Απαγορεύεται αυστηρά οποιοδήποτε κομμάτι κειμένου της να εμφανίζεται αυτούσιο ή μεταφρασμένο από κάποια άλλη δημοσιευμένη πηγή. Κάθε τέτοια πράξη αποτελεί προϊόν λογοκλοπής και εγείρει θέμα Ηθικής Τάξης για τα πνευματικά δικαιώματα του άλλου συγγραφέα. Αποκλειστικός υπεύθυνος είναι ο συγγραφέας της Π.Ε, ο οποίος φέρει και την ευθύνη των συνεπειών, ποινικών και άλλων, αυτής της πράξης.

Πέραν των όποιων ποινικών ευθυνών του συγγραφέα, σε περίπτωση που το Ίδρυμα τού έχει απονείμει Πτυχίο, αυτό ανακαλείται με απόφαση της Συνέλευσης του Τμήματος. Η Συνέλευση του Τμήματος με νέα απόφασή της, μετά από αίτηση του ενδιαφερόμενου, του αναθέτει εκ νέου την εκπόνηση Π.Ε με άλλο θέμα και διαφορετικό επιβλέποντα καθηγητή. Η εκπόνηση της εν λόγω Π.Ε πρέπει να ολοκληρωθεί εντός τουλάχιστον ενός ημερολογιακού 6μήνου από την ημερομηνία ανάθεσής της. Κατά τα λοιπά εφαρμόζονται τα προβλεπόμενα στο άρθρο 18. παρ.5 του ισχύοντος Εσωτερικού Κανονισμού».

Ο Δηλών

Ιωάννης Ηλιόπουλος

Ημερομηνία

13/11/2019

ΠΙΝΑΚΑΣ ΠΕΡΙΕΧΟΜΕΝΩΝ

Περίληψη

Όσο η τεχνολογία αναπτύσσεται, τόσο επεκτείνονται και οι δυνατότητες λειτουργίας και απόδοσης των συστημάτων και των εγκαταστάσεων (μηχανήματα κ.λπ.) που χρησιμοποιούνται στην σύγχρονη βιομηχανία.

Τα συστήματα αυτομάτου ελέγχου και οι αισθητήρες δεν αποτελούν πλέον ένα χαρακτηριστικό γνώρισμα, αλλά μια απαραίτητη προϋπόθεση που έχει ως στόχο τη βελτίωση των επιμέρους τμημάτων της παραγωγής. Μέσω αυτών των συστημάτων και των αισθητήρων γίνεται αλίευση δεδομένων που ενέχουν σημαντικές πληροφορίες για τη κατάσταση λειτουργίας αυτών των τμημάτων. Στη συνέχεια, αναλυτές δεδομένων εκμεταλλεύονται τα στοιχεία και έπειτα από κάποιες δοκιμές και συμπεράσματα, επιτυγχάνουν να προτείνουν μια ροή λειτουργίας, την πιο αποτελεσματική κατά περίπτωση (σχέση ποιότητας-απόδοσης-κόστους).

Πολλές φορές, αυτά τα δεδομένα αποθηκεύονται σε αρχεία .csv (comma separated values) και μπορεί να εμφανίσουν κάποιες ιδιοτροπίες στο περιεχόμενό τους. Αυτό καθιστά απαραίτητη την επεξεργασία αυτών των αρχείων πριν αξιοποιηθούν από τους αναλυτές δεδομένων. Επειδή ο αριθμός και το περιεχόμενο αυτών των αρχείων είναι ογκώδη, η ανάπτυξη μιας εφαρμογής, που θα διαχειρίζεται και θα επεξεργάζεται αυτόματα όλο αυτό το πλήθος αρχείων, θα αποτελέσει ένα βασικό εργαλείο χρήσης των αναλυτών δεδομένων. Η ανάδειξη της χρησιμότητας μιας τέτοιας εφαρμογής είναι και ο σκοπός αυτής της πτυχιακής εργασίας.

Abstract

As technology evolves, so do the capabilities and performance of the systems and installations (machines, etc.) used in modern industry. Automatic control systems and sensors are no longer a feature, but a necessary prerequisite to optimize the individual parts of production.

These systems and sensors extract data that provides important information on the operation status of these components. Then, data analysts take advantage of it and, after some testing and conclusions, manage to propose a workflow that will be the most effective (quality-performance-cost).

Many times this data is stored in .csv (comma separated values) files and may present some gimmicks in their content. This makes it necessary to edit these files before they can be exploited by data analysts. Because the numbers and contents of these files are large, developing an application that will automatically manage and process all of these files will be a key tool for data analysts to use. This is the purpose of this thesis.

ΕΙΣΑΓΩΓΗ

Σήμερα, οι οργανισμοί και οι επιχειρήσεις, προκειμένου να εξυπηρετήσουν τις οικονομικές και διαχειριστικές τους ανάγκες, συγκεντρώνουν μεγάλο όγκο δεδομένων από διαφορετικές πηγές σε καθημερινή βάση. Ωστόσο, η μετατροπή της μεγάλης ποσότητας δεδομένων σε γνώση και η δυνατότητα εκμετάλλευσης των δεδομένων ώστε να αποκτήσει κάποιος καλύτερη οπτική των καταστάσεων, παραμένει μία πρόκληση για τους περισσότερους οργανισμούς και επιχειρήσεις.

Η ανάλυση δεδομένων (data analytics) αποτελεί μια από τις μεγαλύτερες «επαναστάσεις» που έλαβαν χώρα τα τελευταία χρόνια σε επιχειρηματικό επίπεδο, δίνοντας τη δυνατότητα στους επιχειρηματίες να αποφύγουν τις επικίνδυνες εξελίξεις και να προστατέψουν καλύτερα τα συμφέροντά τους.

Τα data analytics είναι πολύ σημαντικά διότι βοηθούν τους οργανισμούς ή τις εταιρίες να αξιοποιήσουν τα δεδομένα τους και να τα χρησιμοποιήσουν για να εντοπίσουν τις κατάλληλες ευκαιρίες καθώς και ενδεχόμενους επιχειρηματικούς κινδύνους.

Το γεγονός αυτό, ακολούθως, οδηγεί σε έξυπνες επιχειρηματικές λύσεις, που κατά περίπτωση μπορεί να αποδειχθούν αποδοτικότερες, να φέρουν δηλαδή υψηλότερα κέρδη και να εξασφαλίσουν ένα μόνιμα ικανοποιημένο πελατολόγιο.

Το κλειδί για την επιτυχημένη ενσωμάτωση των data analytics στο υπάρχον επιχειρηματικό μοντέλο (business model) μιας οποιασδήποτε εταιρίας είναι η εκπαίδευση.

Τα big data ορίζουν μια εντελώς διαφορετική νοοτροπία στο επιχειρείν, ανεξαρτήτως μάλιστα κλάδου ή λοιπών ειδικών συνθηκών, και ως εκ τούτου οι εργαζόμενοι που είναι υπεύθυνοι για την IT (information technology) στρατηγική θα χρειαστούν επιπλέον γνώσεις και εργαλεία.

Σκοπός της πτυχιακής είναι να προσφέρει ένα χρήσιμο, εργονομικό εργαλείο στους αναλυτές δεδομένων.

ΚΕΦΑΛΑΙΟ 1: ΕΙΣΑΓΩΓΗ ΣΤΗΝ ΕΠΙΣΤΗΜΗ ΑΝΤΛΗΣΗΣ ΚΑΙ ΑΝΑΛΥΣΗΣ ΔΕΔΟΜΕΝΩΝ

Σήμερα, περισσότερο από ποτέ, κυρίως λόγω της έκρηξης των τεχνολογικών εφαρμογών, η επιστήμη των δεδομένων εξακολουθεί να εξελίσσεται ως μία από τις πιο απαιτητικές και επικερδείς διαδρομές σταδιοδρομίας για εξειδικευμένους επαγγελματίες. Στις μέρες μας, οι επιτυχημένοι επαγγελματίες δεδομένων αντιλαμβάνονται ότι είναι πρωταρχική ανάγκη να εκσυγχρονισθούν οι παραδοσιακές εφαρμογές άντλησης και ανάλυσης μεγάλων ποσοτήτων δεδομένων.

Κατά συνέπεια, ο εκσυγχρονισμός αυτός αφορά κυρίως τις δεξιότητες του προγραμματισμού. Προκειμένου να εξαχθούν χρήσιμες πληροφορίες για τους οργανισμούς-πελάτες τους, οι επιστήμονες δεδομένων θα πρέπει να γνωρίζουν επισταμένως το πλήρες φάσμα του κύκλου ζωής της επιστήμης τους, από εφαρμογής της και να διαθέτουν ένα επίπεδο ευελιξίας και προσαρμοστικότητας στις απαιτήσεις της εποχής ώστε να μεγιστοποιήσουν τις αποδόσεις σε κάθε φάση της διαδικασίας άντλησης και ανάλυσης.

Από τα πιο πάνω, φαίνεται καθαρά πως ένας αποτελεσματικός επιστήμονας δεδομένων αποτελεί, ιδιαίτερα κατά την τελευταία δεκαετία, ένα απαραίτητο περιουσιακό στοιχείο κάθε εταιρίας ή οργανισμού.

Οι καταρτισμένοι επιστήμονες του χώρου διαθέτουν τεχνικές δεξιότητες υψηλού επιπέδου και είναι σε θέση να σχεδιάσουν σύνθετους ποσοτικούς αλγόριθμους για να εκμαιεύσουν και να συνθέσουν μεγάλες ποσότητες πληροφοριών που χρησιμοποιούνται ως απάντηση σε ερωτήματα, με γνώμονα πάντοτε τον εντοπισμό της στρατηγικής για την ικανοποίηση των αναγκών του οργανισμού για τον οποίο εργάζονται.

Ωστόσο, η κατάρτιση μόνο δεν αρκεί. Ένας επιστήμονας του μεγέθους που αναφέραμε πιο πάνω θα πρέπει να διαθέτει την εμπειρία και το χάρισμα της επικοινωνίας προκειμένου να παρουσιάζει άρτια και με ηγετικό πνεύμα τα αποτελέσματά του και τις αναλύσεις του στις ενδιαφερόμενες εταιρίες, οργανισμούς ή επιχειρήσεις.

Η περιέργεια, η αέναη επιμόρφωση και η ικανότητα να επικοινωνεί τα ευρήματά του όχι μόνο στους τεχνικά καταρτισμένους ομολόγους του αλλά κυρίως στους μη γνώστες του αντικειμένου αποτελούν χαρακτηριστικά του σωστού επιστήμονα.

Επιπρόσθετα, η γνώση στατιστικής και γραμμικής άλγεβρας, οι δεξιότητες προγραμματισμού, ιδιαίτερα της αποθήκευσης δεδομένων, η άντληση και η δημιουργία και ανάλυση των αλγορίθμων, είναι χαρακτηριστικά απαραίτητα για τον επαγγελματία της επιστήμης των δεδομένων.

Η επίλυση σύνθετων προβλημάτων που ανακύπτουν κατά καιρούς, κυρίως στο πεδίο των ακατέργαστων δεδομένων, απαιτεί τη διαμόρφωση της κατάλληλης στρατηγικής που θα πρέπει να εφαρμοστεί κάθε φορά ώστε να οδηγηθεί ο επιστήμονας στη σωστή απόφαση με την βοήθεια της γνώσης και τη χρήση όλης της διαθέσιμης πληροφορίας που είναι αποθηκευμένη σε βάσεις δεδομένων και σε άλλες παρόμοιες πηγές.

Σε κάθε περίπτωση, όλα τα προβλήματα επιλύονται με την σωστή άντληση και ανάλυση των δεδομένων. Αυτό επιτυγχάνεται με την ανακάλυψη και αξιοποίηση προτύπων, δομών και μοντέλων, τάσεων και συσχετίσεων με έναν αυτοματοποιημένο τρόπο.

Κατά συνέπεια, η άρτια γνώση της επιστήμης των δεδομένων και η σωστή εκμετάλλευση αυτής της γνώσης είναι ζωτικής σημασίας για μια επιχείρηση καθώς αφορά στον σχεδιασμό και τη λήψη κρίσιμων αποφάσεων για την αύξηση της παραγωγικότητας σε στρατηγικά και επιχειρησιακά επίπεδα και ακολούθως τη διαμόρφωση της ανταγωνιστικότητάς της έναντι άλλων επιχειρήσεων.

Παρά το γεγονός ότι υπάρχει μια γενικότερη αντίληψη ότι ο στόχος της άντλησης δεδομένων είναι η ανακάλυψη μιας νέας και χρήσιμης πληροφορίας σε βάσεις στοιχείων, τα μέσα για την επίτευξη του στόχου αυτού ποικίλουν σε μεγάλο βαθμό.

Η άντληση της γνώσης περιλαμβάνει ένα ευρύ πεδίο υπολογιστικών μεθόδων που μεταξύ άλλων περιλαμβάνουν, την στατιστική ανάλυση (statistical analysis), τα δένδρα αποφάσεων (decision trees), τα νευρωνικά δίκτυα (neural networks), την εξαγωγή κανόνων (rule induction) και την γραφική οπτικοποίηση (graphic visualization).

Τέτοιες μέθοδοι χρησιμοποιούνται για την εύρεση συσχετίσεων, προτύπων και δομών σε τεράστιες και διαρκώς αυξανόμενες βάσεις δεδομένων.

Ειδικά στο πεδίο της εύρεσης εργαλείων διαμορφώνεται ένα πολύ σημαντικό εξαγόμενο της άντλησης δεδομένων μέσω των σχέσεων που αναπτύσσονται ανάμεσα στα χαρακτηριστικά κάθε βάσης δεδομένων.

Η άντληση γνώσης βοηθά τις σύγχρονες εταιρείες να εστιάζουν στα πιο σημαντικά στοιχεία από τις αποθήκες δεδομένων τους. Με άλλα λόγια είναι η διαδικασία εφαρμογής μεθόδων ανάλυσης σε μεγάλο όγκο δεδομένων.

Ο χρήστης των εργαλείων άντλησης δεδομένων μπορεί να προβλέψει μελλοντικές συμπεριφορές και συνήθειες, ώστε οι εταιρίες να λαμβάνουν τις πιο επιτυχημένες αποφάσεις.

Είναι εμφανές πως οι τεχνικές άντλησης γνώσης αναπτύσσονται γρήγορα, δίχως αλλαγές στην υποδομή και με μοναδικό στόχο την αξιοποίηση των επεξεργασμένων δεδομένων.

Στη διεθνή βιβλιογραφία υπάρχει μια γενικότερη σύγχυση ανάμεσα στους όρους «Άντληση/Εξόρυξη Γνώσης» (Data mining) και «Ανεύρεση γνώσης στις βάσεις δεδομένων» (Knowledge discovery in databases, KDD).

Σε πολλές περιπτώσεις αξίζει να σημειωθεί ότι οι δύο αυτοί όροι ταυτίζονται, ενώ στην πραγματικότητα η άντληση δεδομένων αποτελεί τμήμα της ανεύρεσης της γνώσης και διαμορφώνει τον πυρήνα της γνώσης.

Προκειμένου να γίνει καλύτερα κατανοητή η άντληση δεδομένων, θα γίνει εδώ μια σύντομη αναφορά στη διαδικασία της ανεύρεσης γνώσης.

Η ανεύρεση γνώσης είναι μια επαναλαμβανόμενη διαδικασία μιας σειράς βημάτων, τα οποία οδηγούν από τη συλλογή των δεδομένων στην ανακάλυψη και εξαγωγή χρήσιμης πληροφορίας από αυτά.

Τα βήματα από τα οποία αποτελείται η διαδικασία ανεύρεσης γνώσης είναι τα ακόλουθα:

Καθαρισμός δεδομένων (Data cleaning): Στο βήμα αυτό γίνεται ένα ξεκαθάρισμα καθώς αφαιρούνται από τη βάση δεδομένων αυτά που παράγουν θόρυβο, δηλαδή όλα εκείνα τα στοιχεία που μπορούν να επηρεάσουν ή και να διαστρεβλώσουν το τελικό αποτέλεσμα.

Ενσωμάτωση δεδομένων (Data integration): Σε αυτό το στάδιο όλα τα δεδομένα τα οποία έχουν συλλεχθεί και που μπορεί να είναι ανομοιογενή ή/και από πολλές διαφορετικές πηγές, ενσωματώνονται σε μια κοινή βάση δεδομένων.

Επιλογή δεδομένων (Data selection): Από όλα εκείνα τα δεδομένα που έχουμε στη διάθεσή μας, επιλέγονται προσεκτικά τώρα εκείνα που είναι σχετικά και χρήσιμα για την ανάλυση που θα ακολουθήσει.

Τροποποίηση δεδομένων (Data transformation): Εδώ, τα δεδομένα που έχουμε επιλέξει δέχονται τις απαραίτητες τροποποιήσεις έτσι ώστε η μορφή τους να είναι κατάλληλη για την διαδικασία της άντλησης.

Άντληση/Εξόρυξη δεδομένων (Data mining): Αυτό είναι το σημαντικότερο από τα βήματα της διαδικασίας. Αυτό συμβαίνει επειδή στο συγκεκριμένο στάδιο, χρησιμοποιούνται διάφορες εξελιγμένες τεχνικές για την εξαγωγή δυνητικά χρήσιμων προτύπων.

Αξιολόγηση προτύπων (Pattern evaluation): Στο βήμα αυτό αναγνωρίζονται χρήσιμα πρότυπα που αναπαριστούν γνώση, βάσει συγκεκριμένων μέτρων αξιολόγησης (evaluation measures).

Αναπαράσταση γνώσης (Knowledge representation): Στο τελικό αυτό στάδιο, η γνώση, η οποία έχει ανακαλυφθεί, παρουσιάζεται στον χρήστη, βοηθώντας τον έτσι να κατανοήσει και να ερμηνεύσει τα αποτελέσματα της άντλησης δεδομένων.

Πολλές φορές, κάποια από τα παραπάνω βήματα μπορούν να συνδυαστούν μεταξύ τους προκειμένου να οδηγηθούμε στο καλύτερο δυνατό αποτέλεσμα.

Για παράδειγμα, τα βήματα του καθαρισμού και της ενσωμάτωσης των δεδομένων, μπορούν να υλοποιηθούν μαζί, με στόχο τη δημιουργία μια αποθήκης δεδομένων.

Με την ίδια λογική μπορούν να συνδυαστούν και τα βήματα της επιλογής και τροποποίησης των δεδομένων.

Από τα παραπάνω λοιπόν συμπεραίνουμε ότι η εξόρυξη/άντληση δεδομένων είναι μια πολύ σημαντική διαδικασία-κλειδί για την ανεύρεση γνώσης. Ωστόσο, παρά την σπουδαιότητά και την πολυπλοκότητά της, δεν καταλαμβάνει παρά μόνο ένα μικρό μέρος της όλης προσπάθειας.

Σε αυτό το σημείο, αξίζει να σημειωθεί ότι ο χρήστης, εκμεταλλευόμενος την επαναλαμβανόμενη μορφή της διαδικασίας ανεύρεσης γνώσης, έχει πάντοτε τη δυνατότητα να τροποποιήσει τα μέτρα αξιολόγησης, να τελειοποιήσει τη διαδικασία της άντλησης, να επιλέξει νέα δεδομένα, να τροποποιήσει περαιτέρω τα ήδη υπάρχοντα ή/και να ενσωματώσει στη βάση νέα, από καινούργιες πηγές.

Όλες αυτές οι ενέργειες έχουν ως τελικό στόχο την εξαγωγή διαφορετικών και ακόμη πιο κατάλληλων αποτελεσμάτων.

ΣΤΟΧΟΙ ΤΗΣ ΑΝΤΛΗΣΗΣ ΔΕΔΟΜΕΝΩΝ

Οι μέθοδοι άντλησης της γνώσης στοχεύουν πάντοτε στην ανακάλυψη στοιχείων που θα είναι χρήσιμα για τους οργανισμούς και τις επιχειρήσεις σε σχέση πάντοτε με την οργάνωση και τη δυναμική της κερδοφορίας τους, μέσα από κάποιες τυποποιημένες μορφές πληροφόρησης.

Για παράδειγμα, μέσω της άντλησης της γνώσης μπορούμε να γνωρίζουμε πως υπάρχουν πελάτες οι οποίοι θα ψωνίσουν περισσότερο από δύο φορές σε περίοδο εκπτώσεων ή προσφορών, ή ακόμα, πως είναι πιθανό να αγοράσουν τουλάχιστον μια φορά κατά τη διάρκεια των εορταστικών ημερών.

Ενημερωνόμαστε επίσης για συσχετίσεις αγορών, όπως στην περίπτωση ενός πελάτη ο οποίος αφού αγοράζει ένα dvd player είναι πολύ πιθανό να αγοράσει και κάποια άλλη ηλεκτρονική συσκευή.

Αυτές οι γνώσεις είναι δυνατόν να αποτελέσουν καθοριστικούς παράγοντες για την λήψη αποφάσεων, όσον αφορά τη λειτουργία μιας εμπορικής επιχείρησης.

Η εφαρμογή αυτών των γνώσεων θα μπορούσε να αφορά αποφάσεις σχετικές με το ωράριο, το ύψος και τη διάρκεια των εκπτώσεων, ακόμη και την εργονομική τοποθέτηση των προϊόντων μέσα στα καταστήματα.

Παράλληλα, τέτοιου είδους πληροφορίες χρησιμοποιούνται επίσης για τον προγραμματισμό χρήσης πρόσθετων αποθηκευτικών χώρων ή ακόμη και για τον σχεδιασμό διαφορετικών στρατηγικών μάρκετινγκ.

Τα στελέχη της επιχείρησης, τα οποία είναι υπεύθυνα για τη λήψη των αποφάσεων, εκμεταλλεύονται τις δυνατότητες της άντλησης γνώσης με κάθε τρόπο και μετατρέπουν αυτή την γνώση σε επιτυχή αποτελέσματα.

Παρακάτω περιγράφονται και αναλύονται οι στόχοι της άντλησης δεδομένων:

Οι βασικοί στόχοι της άντλησης των στοιχείων είναι, όπως αναφέρθηκε πιο πάνω, η εφαρμογή τεχνικών πρόβλεψης και συμπεριφοράς των τάσεων του αγοραστικού κοινού (prediction), η αναγνώριση και η περιγραφή τους (description) σε μεγάλες βάσεις δεδομένων, καθώς επίσης η ταξινόμηση και η βελτίωση των πόρων της επιχείρησης.

Ειδικότερα:

Πρόβλεψη: Περιλαμβάνει την χρήση μερικών μεταβλητών ή χαρακτηριστικών μιας βάσης δεδομένων για την πρόβλεψη άγνωστων ή μελλοντικών τιμών χρήσιμων μεταβλητών.

Με άλλα λόγια, οι διαδικασίες πρόβλεψης της άντλησης δεδομένων (predictive data mining tasks), προσπαθούν να κάνουν εκτιμήσεις βγάζοντας συμπεράσματα από τα διαθέσιμα δεδομένα.

Η προσπάθεια πρόβλεψης μελλοντικών συμπεριφορών έχει ως στόχο να ληφθούν αποφάσεις που να μεγιστοποιούν το κέρδος και να προλαμβάνουν δυσάρεστες καταστάσεις.

Τα αποτελέσματα της άντλησης στοιχείων μπορεί να είναι πληροφορίες σχετικές με το ύψος των πωλήσεων ενός καταστήματος για μια συγκεκριμένη χρονική περίοδο ή ακόμη και για το εάν το κλείσιμο μιας γραμμής παραγωγής θα είχε θετική επίδραση στις πωλήσεις.

Συγχρόνως, η διαδικασία της άντλησης αφορά ακόμη και στο ευρύτερο επιστημονικό πεδίο.

Για παράδειγμα, η μελέτη παλαιότερων σεισμικών φαινομένων, ενδεχομένως να οδηγούσε στην πρόβλεψη μιας σεισμικής δραστηριότητας.

Αναγνώριση: Σε αυτή τη φάση οι τυποποιημένες μορφές των δεδομένων χρησιμοποιούνται για να δηλώσουν την ύπαρξη μιας δραστηριότητας ή ενός γεγονότος.

Περιγραφή: Είναι η διαδικασία η οποία επικεντρώνεται στην ανακάλυψη προτύπων και αναπαριστά τα δεδομένα μιας πολύπλοκης βάσης δεδομένων με όσο το δυνατό πιο κατανοητό και αξιοποιήσιμο τρόπο.

Με άλλα λόγια, οι περιγραφικές διαδικασίες της άντλησης δεδομένων (descriptive data mining tasks) αντικατοπτρίζουν τις γενικές ιδιότητες των υπαρχόντων διαθέσιμων δεδομένων.

Ταξινόμηση: Σε αυτό το στάδιο έχουμε διαχωρισμό των στοιχείων, με αποτέλεσμα να προκύπτουν διαφορετικές κατηγορίες ή κλάσεις.

Για παράδειγμα, οι πελάτες μιας υπεραγοράς είναι δυνατόν να χωριστούν σε παρορμητικούς ή πιστούς ή αλλιώς, κανονικούς, σπάνιους ή ακόμα σε φίλους των εκπτώσεων και των προσφορών.

Κατά την ανάλυση των πωλήσεων, αυτή η κατηγοριοποίηση χρησιμοποιείται για να ληφθούν αποφάσεις, με στόχο την προσέλκυση περισσότερων πελατών, ανεξαρτήτως κατηγορίας.

Βελτιστοποίηση: Μεταξύ των άλλων, σκοπός της άντλησης γνώσης είναι η βέλτιστη χρήση κάποιων πόρων κάτω από περιορισμούς.

Τέτοιοι πόροι μπορεί να είναι ο χρόνος, ο χώρος, το χρήμα και η μεγιστοποίηση κάποιων μεγεθών, όπως είναι τα κέρδη είτε οι πωλήσεις. Σε αυτή την περίπτωση η άντληση γνώσης έχει κοινά σημεία με την επιχειρησιακή έρευνα (Marketing).

Στη βιομηχανία χρησιμοποιούνται πολλές φορές αισθητήρες και συστήματα καταγραφής δεδομένων για παρακολούθηση τιμών κ.α. Σαν ο αριθμός των αρχείων που περιέχουν αυτά τα δεδομένα αποτελεί συνήθης μορφοποίηση csv και το πλήθος αυτών των αρχείων σαν μέγεθος είναι ογκώδης, στη παρούσα εργασία αναπτύσσεται μια εφαρμογή για να διευκολύνει έναν αναλυτή στην επεξεργασία που κάνει με τα αρχεία που προμηθεύεται με σκοπό να τα αναλύσει.

ΚΕΦΑΛΑΙΟ 2: ΕΡΓΑΛΕΙΑ ΠΟΥ ΧΡΗΣΙΜΟΠΟΙΗΘΗΚΑΝ ΓΙΑ ΤΗΝ ΕΡΓΑΣΙΑ

2.1 ΓΛΩΣΣΑ ΠΡΟΓΡΑΜΜΑΤΙΣΜΟΥ PYTHON

Η Python είναι μια ιδιαίτερα δημοφιλής γλώσσα γενικού προγραμματισμού με εφαρμογή κυρίως στην επιστήμη των δεδομένων. Οι εταιρείες την χρησιμοποιούν για να συλλέξουν πληροφορίες από τα δεδομένα τους και να αποκτήσουν ανταγωνιστικό πλεονέκτημα.

Περιλαμβάνει χρήσιμες βιβλιοθήκες για ανάλυση δεδομένων όπως η pandas που είναι ένα γρήγορο, ισχυρό, ευέλικτο και εύχρηστο εργαλείο ανάλυσης και χειρισμού δεδομένων ανοιχτού κώδικα.

Η Python είναι μία γλώσσα προγραμματισμού «υψηλού επιπέδου» (άλλα παραδείγματα τέτοιων γλωσσών είναι η FORTRAN, C, Java κ.λπ.). Ο κώδικας μίας τέτοιας γλώσσας πρέπει να μετατραπεί σε «γλώσσα μηχανής» ώστε να εκτελεστεί από τον ηλεκτρονικό υπολογιστή. Η επεξεργασία αυτή γίνεται από διερμηνευτές (interpreters) και μεταγλωττιστές (compilers). Στην περίπτωση της Python η επεξεργασία γίνεται από διερμηνευτή.

Η Python είναι μια γλώσσα προγραμματισμού με απλό συντακτικό, εξαιρετική αναγνωσιμότητα, φορητότητα (portability) και μοντέρνα χαρακτηριστικά που την κάνουν κατάλληλη ως πρώτη γλώσσα προγραμματισμού. Η επιτυχία της οφείλεται, σε μεγάλο βαθμό, στο γεγονός ότι είναι ερμηνευμένη γλώσσα (interpreted language). Αυτό σημαίνει ότι ο μεταγλωττιστής της Python παράγει ενδιάμεσο κώδικα (bytecode) ο οποίος μπορεί να εκτελεστεί από τον διερμηνέα (interpreter) σε πολλά διαφορετικά υπολογιστικά περιβάλλοντα. Διερμηνείς για τη γλώσσα Python υπάρχουν για όλα τα δημοφιλή λειτουργικά συστήματα (Windows, Linux, MacOS).

2.2 ΑΡΧΕΙΑ CSV

Χρησιμοποιήθηκε ένα πλήθος αρχείων csv για δοκιμές της εφαρμογής τα οποία περιλαμβάνουν δεκαδικές τιμές από αισθητήρες με τις χρονικές στιγμές τους (timestamps).

Ένα αρχείο csv είναι τα αρχικά των λέξεων Comma Separated Values που υποδηλώνει στην ουσία μία σειρά μεταβλητών με ενσωματωμένες τιμές, χωρισμένες απλά μεταξύ τους με ένα κόμμα.

Η διαφορά των αρχείων csv με τα αρχεία xls (Excel), είναι απλή:

Τα csv αρχεία σώζονται ως κανονικό κείμενο σε ansi μορφή, διαχωρίζοντας τα δεδομένα των πεδίων με ένα κόμμα και η επεξεργασία τους είναι δυνατή ακόμη και με έναν απλό text editor. Τα αρχεία δεν διατηρούν τις μορφοποιήσεις (χρώματα, bold, κ.λπ.), παρά μόνο τα δεδομένα.

Αντίθετα, τα αρχεία xls σώζονται σε δυαδική μορφή binary, διατηρώντας μέσα τους και όλες τις μορφοποιήσεις. Ωστόσο, είναι δυνατόν να επεξεργαστούν μόνο με ένα ειδικό πρόγραμμα για τέτοια αρχεία.

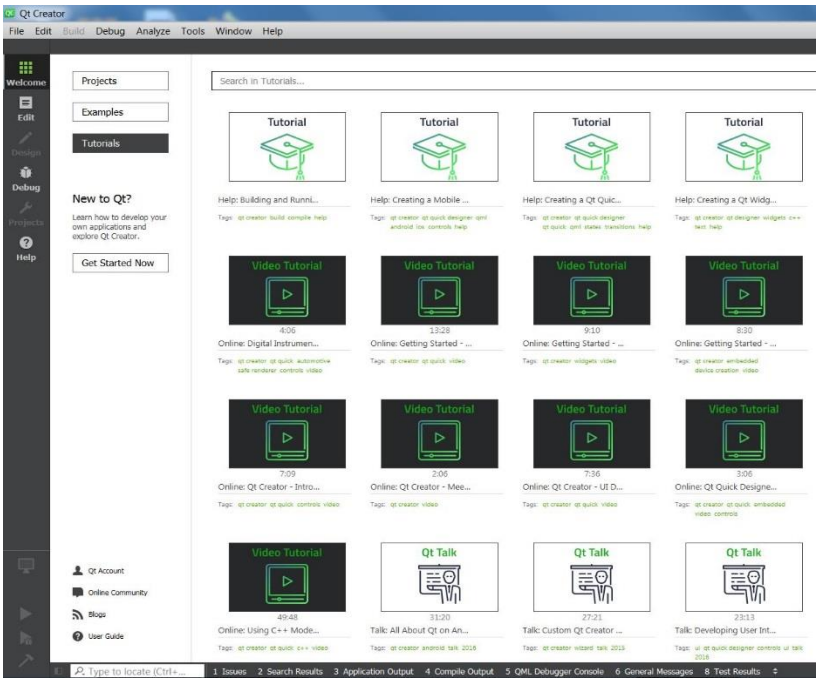
2.3 ΓΡΑΦΙΚΟ ΠΕΡΙΒΑΛΛΟΝ Qt Creator (IDE)

Για διευκόλυνση του χρήστη αξιοποιήθηκε και ένα GUI περιβάλλον.

Το αρκτικόλεξο GUI προέρχεται από τα αρχικά των αγγλικών λέξεων Graphical User Interface και στην πληροφορική καλείται ένα σύνολο εικονικών στοιχείων, τα οποία εμφανίζονται στην οθόνη μίας ψηφιακής συσκευής και χρησιμοποιούνται για να διευκολύνουν και να επιταχύνουν την αλληλεπίδραση μεταξύ του χρήστη και της συσκευής.

Σε αυτό το πλαίσιο, παρέχονται ενδείξεις και «εργαλεία» μέσω εικόνων, προκειμένου με απλές ενέργειες να εκτελεστούν συγκεκριμένες εργασίες. Τα εικονικά αυτά στοιχεία «αποδέχονται» ενέργειες του χρήστη και «αντιδρούν» ανάλογα στα συμβάντα που αυτός προκαλεί μέσω κάποιας συσκευής εισόδου (π.χ. πληκτρολόγιο, ποντίκι).

Κατά συνέπεια, το Qt Creator IDE (Integrated Development Environment) παρέχει ένα ολοκληρωμένο περιβάλλον ανάπτυξης για τους προγραμματιστές, όσο αφορά τη δημιουργία εφαρμογών για πολλαπλές πλατφόρμες υπολογιστών και κινητών συσκευών, όπως Android και iOS. Είναι επίσης διαθέσιμο για λειτουργικά συστήματα Linux, macOS και Windows.

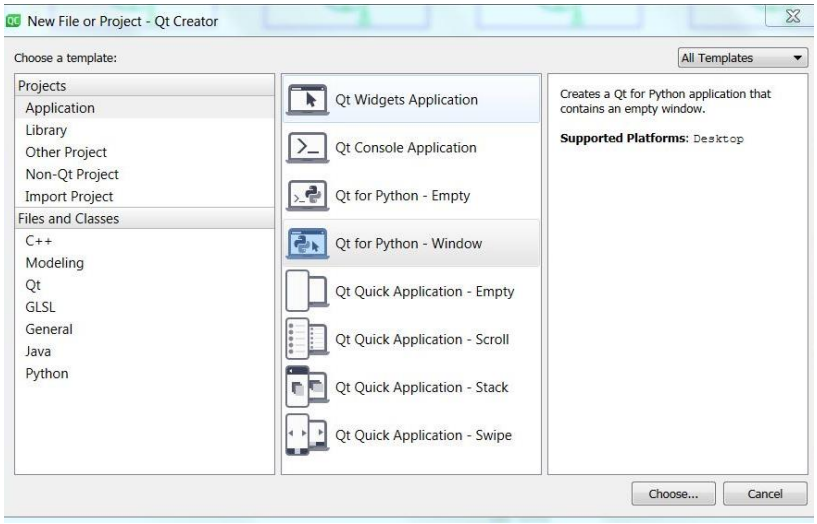


Εικόνα 1 *Γραφικό περιβάλλον του Qt Creator*

Για περισσότερες πληροφορίες σχετικά με το Qt Creator μπορεί κάποιος να ανατρέξει στην ιστοσελίδα αυτού του IDE και να διαβάσει το documentation που περιλαμβάνεται.

Στην παρούσα εργασία δε θα γίνει μεγάλη ανάλυση σχετικά με το Qt Creator καθώς δεν αποτελεί το κυρίως θέμα της εργασίας, όμως δίνεται μια σύντομη περιγραφή για τις δυνατότητες και τον τρόπο λειτουργίας του:

Όπως ισχύει και στις περισσότερες εφαρμογές υπάρχει η επιλογή για δημιουργία νέου project.

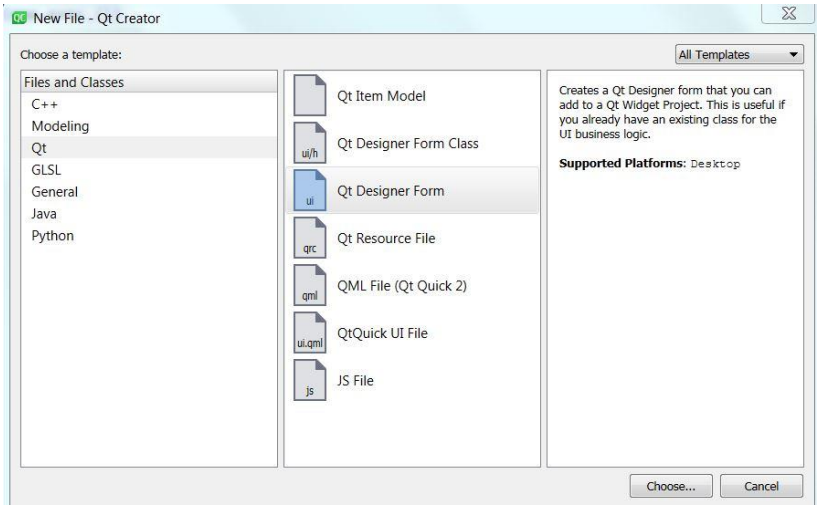


Εικόνα 2 *Δημιουργία νέου project*

Επιλέγεται από το μενού Application και συγκεκριμένα Qt for Python.

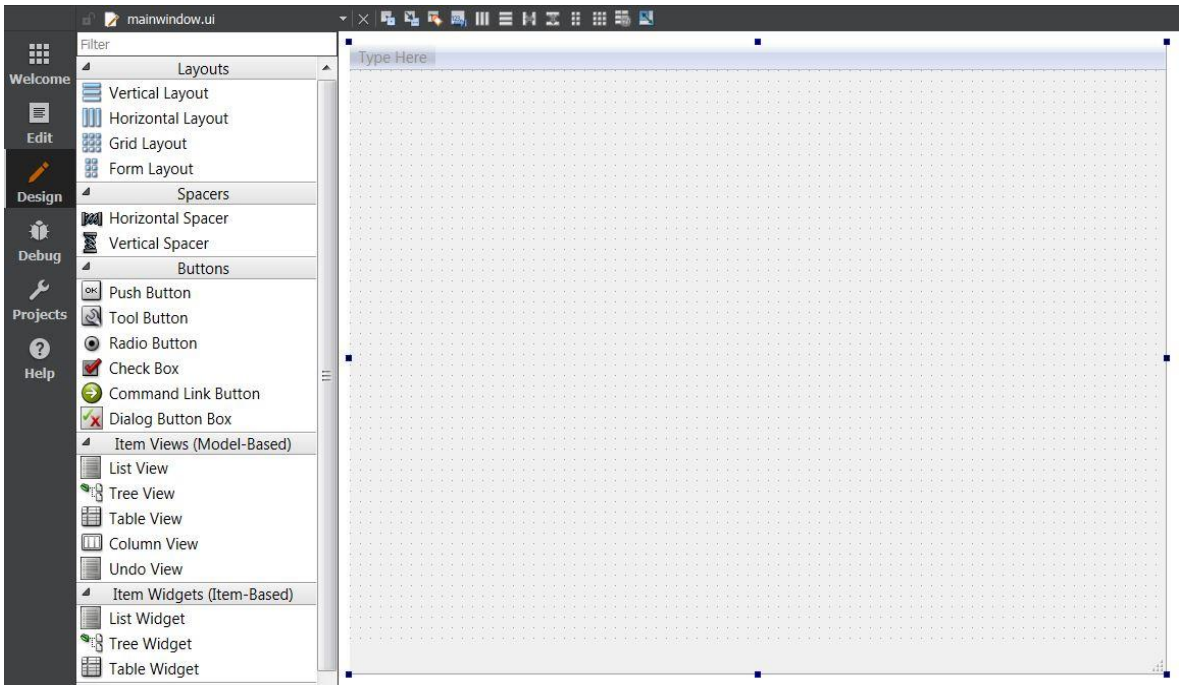
Στη συνέχεια προστίθεται η ονομασία του project και η διεύθυνση αποθήκευσής του.

Αφού το project έχει δημιουργηθεί, είναι δυνατός ο εμπλουτισμός του με επιπλέον αρχεία, όπως διάφορες βιβλιοθήκες και ui αρχεία τα οποία περιλαμβάνουν και αποθηκεύουν τον κώδικα για την εμφάνιση – σχεδίαση της εφαρμογής. Προσθέτοντας ένα αρχείο ui στο project,



Εικόνα 3 *Προσθήκη ui αρχείου στο project*

ξεκλειδώνεται η δυνατότητα σχεδίασης της εφαρμογής.



Εικόνα 4 *Λειτουργία σχεδίασης στο Qt Creator*

Στο κεντρικό παράθυρο, όπως φαίνεται στην εικόνα 4, «κουμπώνουν» πάνω τα εργαλεία που δίνονται στην εργαλειοθήκη αριστερά.

Διατίθενται buttons, check boxes, list view κ.α. τα οποία θα συμπεριληφθούν στην εφαρμογή.

ΚΕΦΑΛΑΙΟ 3: ΣΥΝΔΕΣΗ ΚΑΙ ΕΠΙΚΟΙΝΩΝΙΑ ΤΩΝ ΕΠΙΜΕΡΟΥΣ ΑΡΧΕΙΩΝ ΤΗΣ ΕΦΑΡΜΟΓΗΣ

3.1 ΟΝΟΜΑΣΙΕΣ ΑΡΧΕΙΩΝ ΚΑΙ ΜΙΑ ΣΥΝΤΟΜΗ ΠΕΡΙΓΡΑΦΗ ΤΟΥΣ

Στην εφαρμογή χρησιμοποιήθηκαν τα εξής αρχεία:

- main.py *κύριο αρχείο python της εφαρμογής*
- mainwindow.ui *αρχείο με τον κώδικα απεικόνισης του UI (User Interface)*
- υποφάκελος project με:
    i. dialogs.py *περιλαμβάνει τον κώδικα για όλα τα αναδυόμενα παράθυρα και τις ειδοποιήσεις της εφαρμογής*
    ii. preferences.py *περιλαμβάνει τον κώδικα για τις επιλογές τροποποίησης από τον χρήστη για τα αρχεία csv που θα υποστούν επεξεργασία*

3.2.1 ΣΥΝΔΕΣΗ main.py ΜΕ mainwindow.ui

Στο αρχείο main.py, το κύριο δηλαδή αρχείο python που όταν τρέξει θα ανοίξει η εφαρμογή, φορτώνουμε το αρχείο mainwindow.ui ώστε να εμφανιστεί το κύριο παράθυρο του προγράμματος. Το παράθυρο αυτό περιλαμβάνει όλα τα κουμπιά και τις επιλογές που χρησιμοποιούνται.

```
# Importing necessairy modules
import sys
from PyQt5 import QtWidgets , uic
from project.dialogs import Dialog
from project.preferences import Preferences



class MainWindow(QtWidgets.QMainWindow):

#Loading of the main window form
 def __init__(self):
        super().__init__()
        uic.loadUi("mainwindow.ui", self)
```

Εικόνα 5 *Κώδικας για φόρτωση αρχείου ui και εισαγωγή των modules*

Εδώ δημιουργείται η κύρια class του προγράμματος στο αρχείο της python όπου φορτώνεται το αρχείο ui, εφόσον έχει προηγηθεί η καταχώριση των απαραίτητων βιβλιοθηκών και modules που θα χρησιμοποιηθούν.

```
#Running the application and showing main window form
app = QtWidgets.QApplication(sys.argv)
window = MainWindow()
window.show()
app.exec_()
```

Εικόνα 6 *Κώδικας για εκκίνηση της εφαρμογής*

Στη συνέχεια με τον κώδικα της εικόνας 6 τρέχει η εφαρμογή και απεικονίζεται το κεντρικό παράθυρο του προγράμματος.

3.2.2 ΣΥΝΔΕΣΗ dialogs.py ΚΑΙ preferences.py ΜΕ main.py

Τα αρχεία dialogs.py και preferences.py που βρίσκονται στον υποφάκελο project είναι βιβλιοθήκες που δημιουργήθηκαν και οι οποίες περιέχουν συναρτήσεις που χρησιμεύουν για τη σωστή λειτουργία του προγράμματος.

Η προσθήκη αυτών των βιβλιοθηκών στο κυρίως αρχείο της εφαρμογής, το main.py, γίνεται όπως και με κάθε άλλο class ή module.

```
from project.dialogs import Dialog
from project.preferences import Preferences
```

Εικόνα 7 *Προσθήκη βιβλιοθηκών στο κύριο πρόγραμμα*

3.2.3 ΣΥΝΔΕΣΗ dialogs.py ΚΑΙ preferences.py ΜΕ mainwindow.ui

Τα πιο πάνω αρχεία χρησιμοποιούν αντίστοιχα και άλλα modules με τα οποία τους δίνεται η δυνατότητα να αλληλοεπιδρούν με εργαλεία και πλήκτρα που εμφανίζονται στην εφαρμογή.

```
#Importing necessairy modules
from PyQt5.QtWidgets import QFileDialog , QListWidget , QDialog , QLabel , QPushButton
from os import listdir , path

#Class creation for dialog windows
class Dialog(QDialog):
```

Εικόνα 8 *Modules και classes που χρησιμοποιεί το αρχείο dialogs.py*

```
from PyQt5.QtWidgets import QListWidget , QPushButton , QLabel
import pandas
from os import path


class Preferences(QPushButton):
```

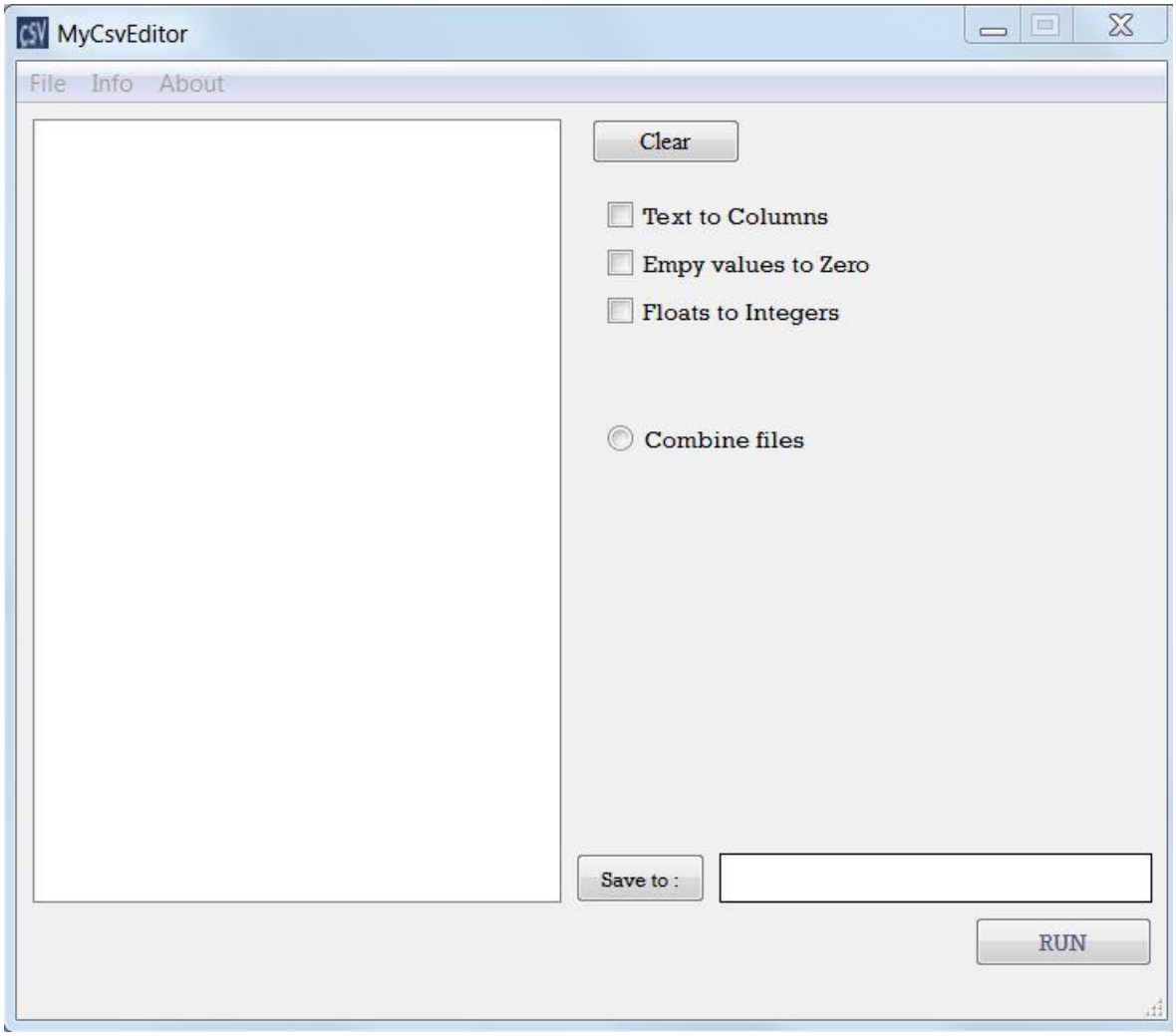
Εικόνα 9 *Modules και classes που χρησιμοποιεί το αρχείο preferences.py*

Η ανάλυση και επεξήγηση των modules αυτών δίνεται σε επόμενο κεφάλαιο μαζί με τον κώδικα που τα αξιοποιεί.

ΚΕΦΑΛΑΙΟ 4: ΟΔΗΓΙΕΣ ΧΡΗΣΗΣ ΤΗΣ ΕΦΑΡΜΟΓΗΣ

4.1 MyCsvEditor

Το όνομα της εφαρμογής είναι το MyCsvEditor και στην εικόνα 10 προβάλλεται το κεντρικό της παράθυρο.



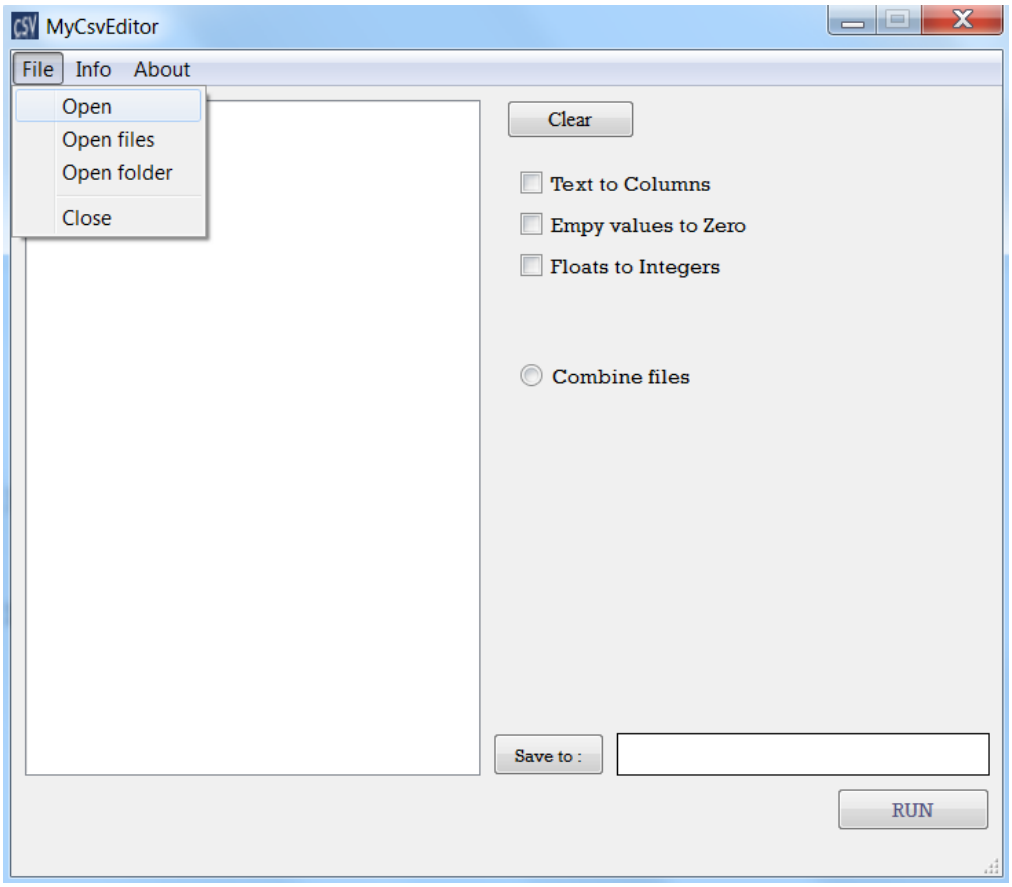
Εικόνα 10 *Κεντρικό παράθυρο του MyCsvEditor*

Παρακάτω αποδίδονται, με λεπτομερή περιγραφή, όλα τα στοιχεία που απεικονίζονται στο κύριο παράθυρο του MyCsvEditor, ένα προς ένα.

4.2 ΤΟ ΚΕΝΤΡΙΚΟ ΜΕΝΟΥ

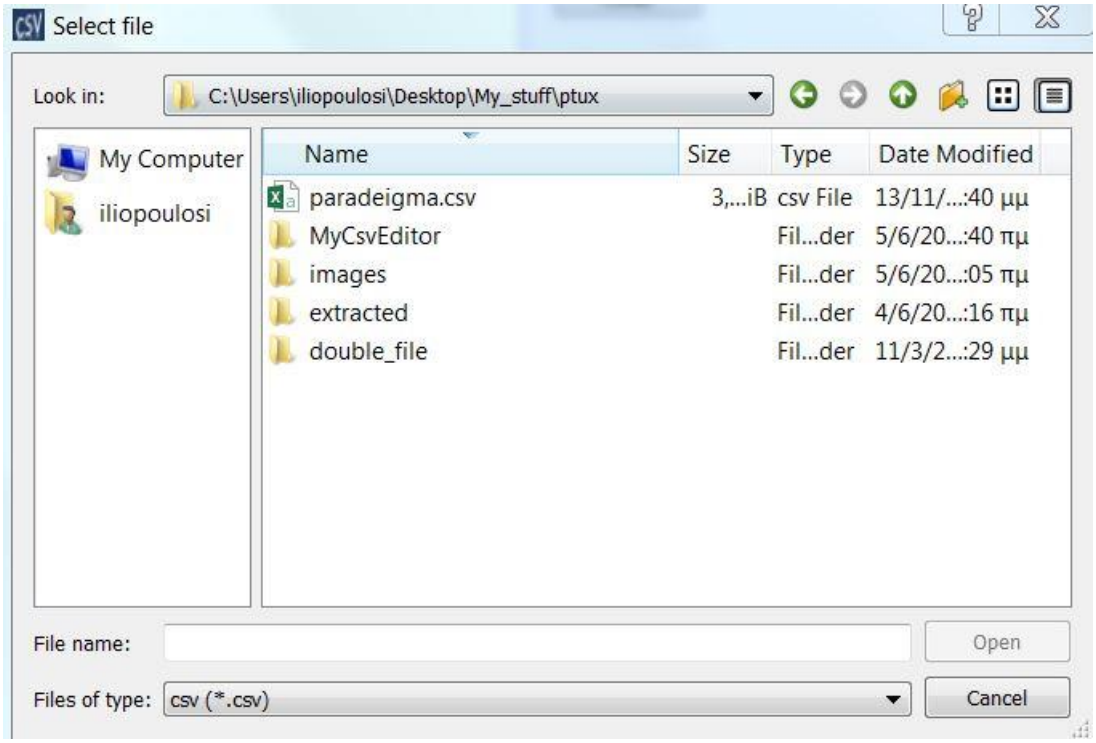
Στο κεντρικό μενού της εφαρμογής υπάρχουν τέσσερις επιλογές:

- Open
- Open files
- Open folder
- Close



Εικόνα 11 *Κεντρικό μενού*

Επιλέγοντας **Open** θα αναδυθεί ένα παράθυρο στο οποίο ο χρήστης καλείται να επιλέξει ένα αρχείο στον υπολογιστή του ώστε να προστεθεί στη λίστα του προγράμματος προς επεξεργασία. Το αρχείο θα πρέπει οπωσδήποτε να είναι csv αρχείο. Αν ο χρήστης επιλέξει μία άλλη μορφή αρχείου, τότε το αρχείο δεν θα προστεθεί στη λίστα του προγράμματος.



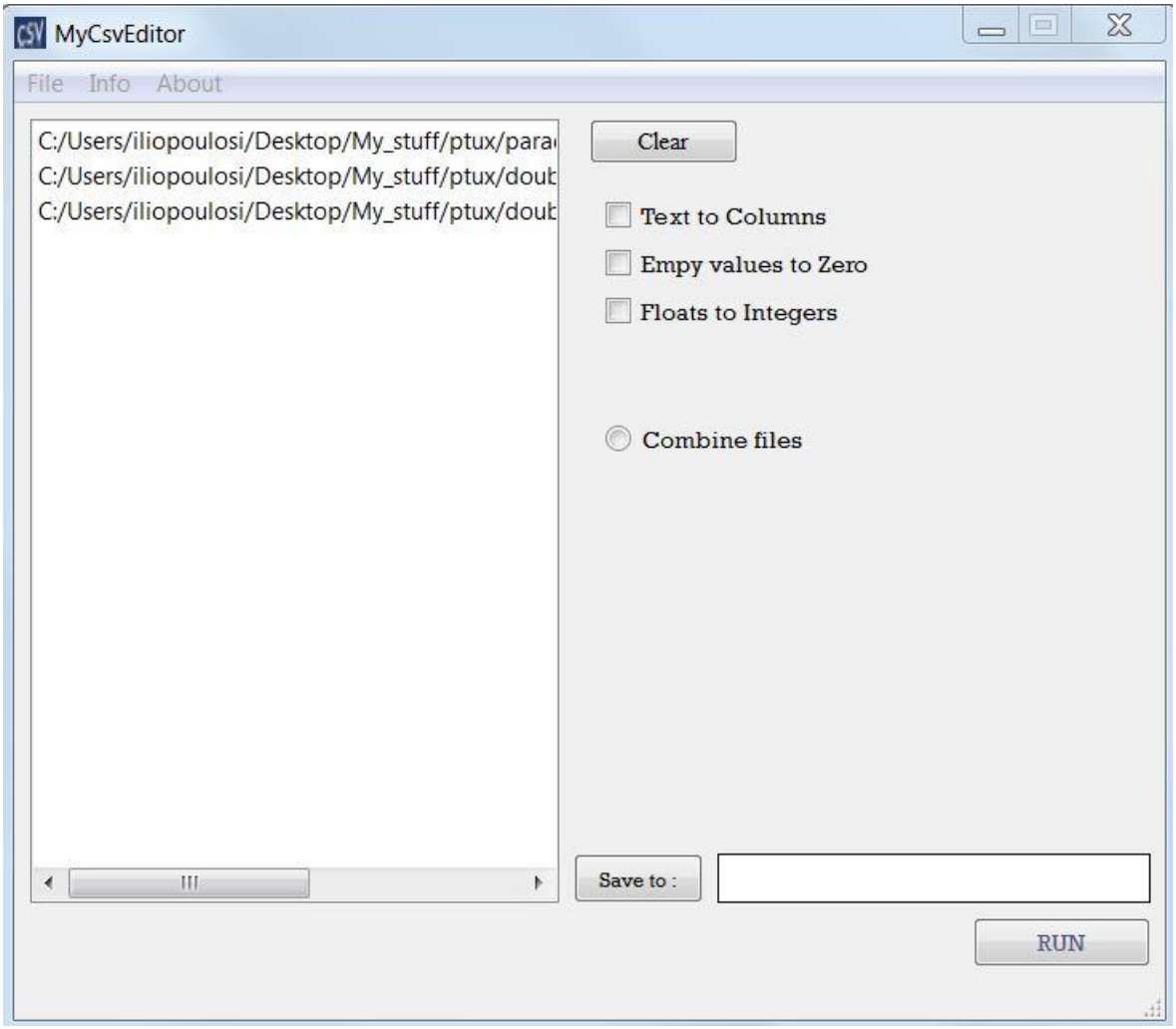
Εικόνα 12 *Αναδυόμενο παράθυρο επιλογής αρχείου*

Οι επιλογές **Open files** και **Open folder** λειτουργούν με παρόμοιο τρόπο. Στην πρώτη περίπτωση καλείται ο χρήστης να επιλέξει πολλαπλά αρχεία csv, και στη δεύτερη έναν φάκελο που περιλαμβάνει csv αρχεία.

Με την επιλογή **Close** η εφαρμογή τερματίζεται.

4.3 Η ΛΙΣΤΑ ΜΕ ΤΑ ΑΡΧΕΙΑ csv ΠΡΟΣ ΕΠΕΞΕΡΓΑΣΙΑ

Εφόσον ο χρήστης έχει επιλέξει ποια αρχεία θέλει να υποστούν επεξεργασία μέσα από τις δυνατότητες του κεντρικού μενού, τα αρχεία που επέλεξε προστίθενται σε μια λίστα, η οποία απεικονίζεται στο κεντρικό παράθυρο της εφαρμογής και προβάλλει ολοκληρωμένα τις διευθύνσεις των αρχείων αυτών στον υπολογιστή στον οποίο τρέχει η εφαρμογή.



Εικόνα 13 *Λίστα όπου αποθηκεύονται τα επιλεγμένα αρχεία csv*

4.4 ΤΑ ΑΠΕΙΚΟΝΙΖΟΜΕΝΑ ΠΛΗΚΤΡΑ

Στο κεντρικό παράθυρο της εφαρμογής φαίνονται τρία πλήκτρα:

- Clear
- Save to:
- RUN

Το πλήκτρο **Clear** αφαιρεί τα αρχεία που πρόσθεσε ο χρήστης στη λίστα και την αφήνει κενή.

Το πλήκτρο **Save to:** εμφανίζει ένα νέο παράθυρο όπου ο χρήστης καλείται να διαλέξει σε ποιο φάκελο θα αποθηκευτούν τα νέα παραγόμενα αρχεία από τα παλιά που υπέστησαν επεξεργασία

Το πλήκτρο **RUN** ξεκινά τη διαδικασία επεξεργασίας των αρχείων csv που βρίσκονται στη λίστα, με βάση τις επιλογές τροποποίησης που έκανε ο χρήστης.

4.5 ΕΠΙΛΟΓΕΣ ΤΡΟΠΟΠΟΙΗΣΗΣ ΕΙΣΑΓΩΜΕΝΩΝ ΑΡΧΕΙΩΝ

Δεξιά από τη λίστα με τα αρχεία που προσθέτει ο χρήστης υπάρχουν κάποια σημεία ελέγχου, checkbox, τα οποία μπορεί ο τελευταίος να επιλέξει ανάλογα με το αποτέλεσμα στο οποίο ο ίδιος αποσκοπεί:

- Text to Columns
- Empty values to Zero
- Floats to Integers
- Combine files

Επιλέγοντας **Text to Columns** στο νέο παραγόμενο αρχείο, όταν αυτό ανοιχτεί με το πρόγραμμα Excel, όλες οι τιμές διαχωρίζονται με στήλες και δεν εμφανίζονται χωρισμένες με κόμμα στην ίδια στήλη.

Επιλέγοντας **Empty values to Zero** το πρόγραμμα εντοπίζει στα αρχεία της λίστας κενές τιμές (NaN, n.a.) και τις αντικαθιστά με την τιμή μηδέν 0.

Επιλέγοντας **Floats to Integers** όλες οι δεκαδικές τιμές στα αρχεία θα μετατραπούν σε ακέραιες.

Τέλος, αν ο χρήστης θελήσει να ενώσει αρχεία csv μεταξύ τους μπορεί να το πετύχει πολύ εύκολα με την επιλογή **Combine files.**

ΚΕΦΑΛΑΙΟ 5: Ο ΚΩΔΙΚΑΣ ΣΤΗΝ PYTHON

5.1 ΚΩΔΙΚΑΣ ΓΙΑ ΤΟ ΚΕΝΤΡΙΚΟ ΜΕΝΟΥ

Στην python έχει αναπτυχθεί μια βιβλιοθήκη, η PyQt5, η οποία συμπεριλαμβάνει όλα τα αντικείμενα (objects) που χρησιμοποιούνται στην εφαρμογή. Τα objects αυτά χαρακτηρίζονται από το πρώτο γράμμα στο όνομα τους, το «Q». Τα object που περιλαμβάνονται στην εφαρμογή είναι τα εξής:

- Το QFileDialog είναι ένα module που χρησιμοποιείται σε περιπτώσεις κατά τις οποίες ο χρήστης καλείται να ανοίξει ένα αρχείο ή να αποθηκεύσει ένα αρχείο όπου εμφανίζει ένα παράθυρο με τις παρούσες διευθύνσεις του υπολογιστή
- Το QListWidget αντιπροσωπεύει ένα αντικείμενο λίστας
- Το QDialog χρησιμοποιείται για αναδυόμενα παράθυρα
- Το QLabel αντιπροσωπεύει μία ετικέτα στο πρόγραμμα
- Το QPushButton περιλαμβάνει τα πλήκτρα (buttons) της εφαρμογής
- Το QCheckBox και το QRadioButton αντιστοιχεί σε κουτάκια τσεκαρίσματος (σημεία ελέγχου)

Τα πιο πάνω objects αντιστοιχούν σε όλα τα εικονιζόμενα σχέδια του κεντρικού παραθύρου της εφαρμογής.

- Το πλήκτρο **Open** από το πτυσσόμενο μενού καταχωρείται στον κώδικα με τις εντολές:

```
open = self.findChild(QtWidgets.QAction, 'actionOpen')
open.triggered.connect(self.fileopener)
```

Εικόνα 14 *Εντολές για την επιλογή open*

Το όνομα του πιο πάνω αντικειμένου είναι ‘actionOpen’. Στην πρώτη εντολή αποθηκεύεται στη μεταβλητή open και στη δεύτερη εντολή καλείται η συνάρτηση fileopener η οποία ενεργοποιείται εάν πατηθεί η επιλογή **Open**. Στη συνέχεια η συνάρτηση fileopener

```
#Call open dialog for a single file
 def fileopener(self):
     Dialog.openFileNameDialog(self)
```

Εικόνα 15 *Συνάρτηση fileopener*

συνδέεται στην συνάρτηση openFileNameDialog του αρχείου dialogs.py.

```
#Open single file dialog function
   def openFileNameDialog(self):
       options = QFileDialog.Options()
       options |= QFileDialog.DontUseNativeDialog
       fileName, _ = QFileDialog.getOpenFileName(self, "Select file", "","csv (*.csv)", options=options)
       list = self.findChild(QListWidget, 'listWidget')
       QListWidget.addItem(list, fileName)
```

Εικόνα 16 *Συνάρτηση openFileNameDialog*

Στην μεταβλητή fileName εισάγεται το αρχείο που επιλέγει ο χρήστης το οποίο επιτρέπει μόνο την εισαγωγή αρχείων csv.

Το όνομα της λίστας σαν αντικείμενο του προγράμματος είναι το 'listWidget'. Αυτή αποθηκεύεται στην μεταβλητή list και τέλος, με την τελευταία γραμμή κώδικα της εικόνας 16, προστίθεται το αρχείο που επέλεξε ο χρήστης στη λίστα.

- Το πλήκτρο **Open files** από το πτυσσόμενο μενού καταχωρείται στον κώδικα με τις εντολές:

```
openfiles = self.findChild(QtWidgets.QAction, 'actionOpen_files')
openfiles.triggered.connect(self.filesopener)
```

Εικόνα 17 *Εντολές για την επιλογή Open files*

Το όνομα του αντικειμένου είναι 'actionOpen_files'. Στην πρώτη εντολή αποθηκεύεται στη μεταβλητή openfiles και στη δεύτερη εντολή καλείται η συνάρτηση filesopener η οποία ενεργοποιείται εάν πατηθεί η επιλογή **Open files**. Στη συνέχεια η συνάρτηση filesopener

```
#Call open dialog for multiple files
 def filesopener(self):
     Dialog.openFileNamesDialog(self)
```

Εικόνα 18 *Συνάρτηση filesopener*

συνδέεται στην συνάρτηση openFileNamesDialog του αρχείου dialogs.py.

```
#Open multiple files dialog function
   def openFileNamesDialog(self):
       options = QFileDialog.Options()
       options |= QFileDialog.DontUseNativeDialog
       files, _ = QFileDialog.getOpenFileNames(self,"Select files", "","csv (*.csv)", options=options)
       for filename in files:
           if filename.endswith(".csv"):
             list = self.findChild(QListWidget, 'listWidget')
             QListWidget.addItem(list, filename)
```

Εικόνα 19 *Συνάρτηση openFileNamesDialog*

Στην μεταβλητή files εισάγονται τα αρχεία που επιλέγει ο χρήστης.

Με την επανάληψη for γίνεται έλεγχος για κάθε αρχείο και εάν είναι csv αρχείο προστίθεται στη συνέχεια στη λίστα.

- Το πλήκτρο **Open folder** από το πτυσσόμενο μενού καταχωρείται στον κώδικα με τις εντολές:

```
openfolder = self.findChild(QtWidgets.QAction, 'actionOpen_folder')
openfolder.triggered.connect(self.folderopener)
```

Εικόνα 20 *Εντολές για την επιλογή Open folder*

Το όνομα του αντικειμένου εδώ είναι 'actionOpen_folder'. Στην πρώτη εντολή αποθηκεύεται στη μεταβλητή openfolder και στη δεύτερη εντολή καλείται η συνάρτηση folderopener η οποία ενεργοποιείται εάν πατηθεί η επιλογή **Open folder**. Στη συνέχεια η συνάρτηση folderopener

```
#Call open dialog for folder
 def folderopener(self):
     Dialog.openFolderNameDialog(self)
```

Εικόνα 21 *Συνάρτηση folderopener*

συνδέεται στην συνάρτηση openFolderNameDialog του αρχείου dialogs.py.

```
#Open folder dialog function
   def openFolderNameDialog(self):
       options = QFileDialog.Options()
       options |= QFileDialog.DontUseNativeDialog
       folder = QFileDialog.getExistingDirectory(self,"Select folder", "", options=options)
       filenames = listdir(folder)
       for filename in filenames:
           if filename.endswith('.csv'):
              pof = path.abspath(filename)
              pofl = pof.replace("\\","/")
              list = self.findChild(QListWidget, 'listWidget')
              QListWidget.addItem(list, pofl)
```

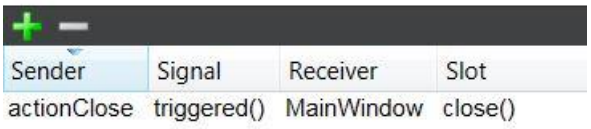
Εικόνα 22 *Συνάρτηση openFolderNameDialog*

Στην μεταβλητή folder εισάγεται η διεύθυνση του φακέλου που επιλέγει ο χρήστης και στην μεταβλητή filenames εισάγονται τα ονόματα των αρχείων που συμπεριλαμβάνονται στον φάκελο που επέλεξε.

Στην συνέχεια, για κάθε αρχείο γίνεται έλεγχος εάν είναι csv αρχείο και έπειτα από μια μικρή τροποποίηση στο όνομα της διεύθυνσης για κάθε αρχείο του φακέλου, προστίθενται στη λίστα όλες οι διευθύνσεις των αρχείων csv.

❖ Το πλήκτρο **Close** από το πτυσσόμενο μενού ωστόσο, καταχωρείται στον κώδικα με διαφορετικό τρόπο από τους προηγούμενους.

Στο πλαίσιο εργασίας για το σχέδιο του κεντρικού παραθύρου στο Qt Creator δίνεται η δυνατότητα, σε περιπτώσεις που μια δράση (action) δεν είναι σύνθετη, να καταχωρηθεί με έναν πολύ απλό και εύκολο τρόπο:

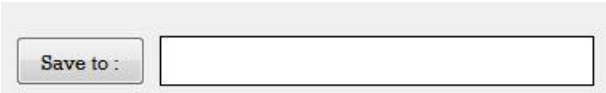


Εικόνα 23 *Επιλογή Close*

Το όνομα του αντικειμένου είναι ‘actionClose’ και όταν ενεργοποιηθεί (δηλαδή πατηθεί το κουμπί **Close**) το κεντρικό παράθυρο (MainWindow) του προγράμματος θα κλείσει και έτσι θα τερματιστεί η εφαρμογή.

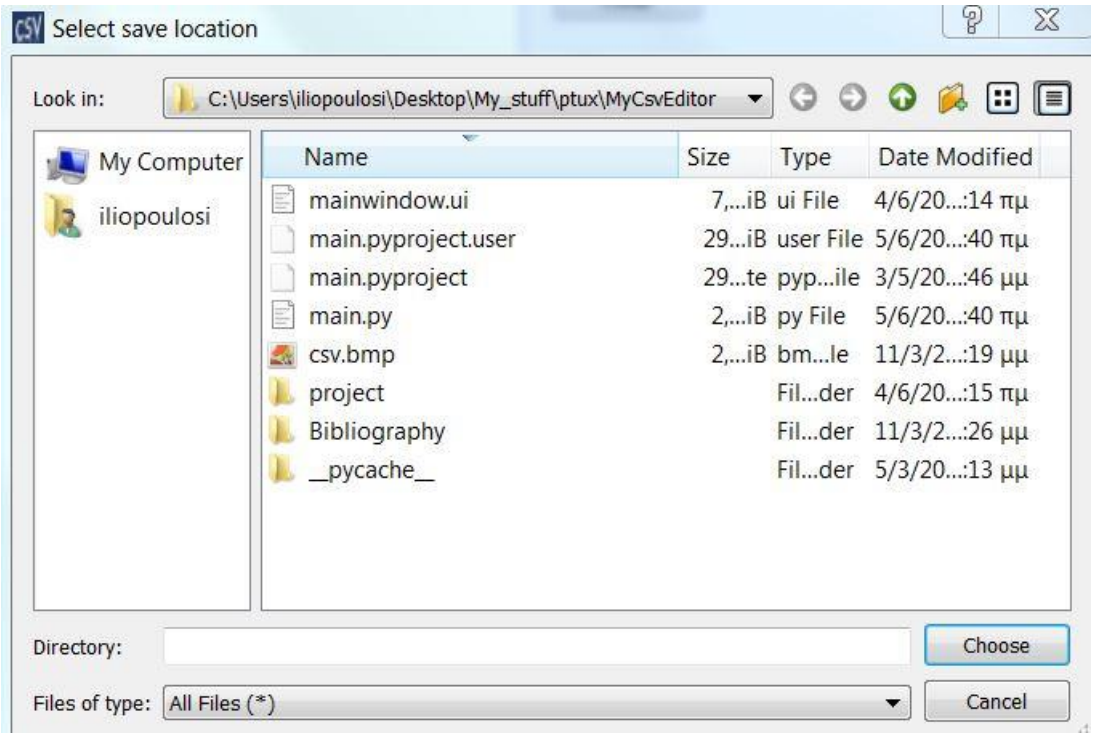
5.2 ΚΩΔΙΚΑΣ ΓΙΑ ΤΗΝ ΑΠΟΘΗΚΕΥΣΗ ΤΩΝ ΑΡΧΕΙΩΝ

Για την τοποθεσία αποθήκευσης των αρχείων, μετά την διαμόρφωσή τους, είναι διαθέσιμο το πλήκτρο **save to:**



Εικόνα 24 *Button για επιλογή τοποθεσίας που θα αποθηκευτεί το αρχείο*

το οποίο, όταν ενεργοποιηθεί από τον χρήστη, ανοίγει ένα καινούριο παράθυρο της εφαρμογής όπου και επιλέγεται η διεύθυνση αποθήκευσης και η ονομασία του παραγόμενου αρχείου στον υπολογιστή.



Εικόνα 25 *Παράθυρο για επιλογή διεύθυνσης αποθήκευσης*

Η διεύθυνση αποθήκευσης εμφανίζεται στην ετικέτα δίπλα από το πλήκτρο **save to:** .

```
saveto = self.findChild(QtWidgets.QPushButton, 'pushButton_2')
saveto.clicked.connect(self.saveloc)
```

Εικόνα 26 *Εντολές για το κουμπί save to:*

Το πλήκτρο **save to:** στο πρόγραμμα ονομάζεται ‘pushButton_2’ και αποθηκεύεται στη μεταβλητή saveto. Όταν πατηθεί το πλήκτρο, καλείται η συνάρτηση saveloc

```
#Call open dialog to select save location
 def saveloc(self):
     Dialog.saveFileDialog(self)
```

Εικόνα 27 *Συνάρτηση saveloc*

η οποία στη συνέχεια τρέχει τη συνάρτηση saveFileDialog του αρχείου dialogs.py.

```
#Open save dialog function
   def saveFileDialog(self):
        options = QFileDialog.Options()
        options |= QFileDialog.DontUseNativeDialog
        directory = QFileDialog.getExistingDirectory(self,"Select save location","", options=options)
        location = self.findChild(QLabel , 'label')
        location.setText(directory)
```

Εικόνα 28 *Συνάρτηση saveFileDialog*

Στην μεταβλητή directory καταχωρείται η διεύθυνση που επέλεξε ο χρήστης για να αποθηκευτεί το τελικό αρχείο ή αρχεία μετά την επεξεργασία.

Στην μεταβλητή location καταχωρείται το αντικείμενο της ετικέτας που φαίνεται στην εφαρμογή, το οποίο έχει την ονομασία ‘label’.

Με την εντολή location.setText(directory) το κείμενο της ετικέτας αλλάζει στο όνομα της διεύθυνσης αποθήκευσης.

5.3 ΚΩΔΙΚΑΣ ΤΩΝ ΕΙΔΟΠΟΙΗΣΕΩΝ

Όπως αναφέρθηκε στο 3ο κεφάλαιο, το πλήκτρο **RUN** θα τρέξει τη διαδικασία επεξεργασίας των αρχείων της λίστας, με βάση τις προεπιλογές που έχει κάνει ο χρήστης. Τι συμβαίνει ωστόσο, εάν ο χρήστης πατήσει το πλήκτρο χωρίς να έχει προσθέσει κάποιο αρχείο ή να έχει βάλει διεύθυνση αποθήκευσης για το αρχείο που θα παραχθεί στη συνέχεια;

Σε αυτές τις περιπτώσεις, θα αναδύονται κάποια παράθυρα που θα ενημερώνουν τον χρήστη για την παράλειψή του.

```
run = self.findChild(QtWidgets.QPushButton, 'pushButton')
run.clicked.connect(self.execute)
```

Εικόνα 29 *Κώδικας για το κουμπί RUN*

Το πλήκτρο **RUN** που έχει σαν όνομα αντικειμένου το 'pushButton' αποθηκεύεται στην μεταβλητή run και όταν ενεργοποιηθεί οδηγεί στην συνάρτηση execute.

```
#Button RUN code
 def execute(self):
     list = self.findChild(QtWidgets.QListWidget, 'listWidget')
     location = self.findChild(QtWidgets.QLabel , 'label')
     hasitem = list.count()
     hasloc = location.text()
     if hasloc == '' and hasitem == 0:
         Dialog.noitem_nosave(self)
         #Please add files to the list and select save location
     elif hasloc == '' and hasitem != 0:
         Dialog.nosave(self)
         #Please select save file location
     elif hasloc != '' and hasitem == 0:
         Dialog.noitem(self)
         #Please add files to the list
```

Εικόνα 30 *Πρώτο σκέλος συνάρτησης execute*

Όπως αναφέρθηκε προηγουμένως, η λίστα με όνομα αντικειμένου 'listWidget' αποθηκεύεται στη μεταβλητή list και η ετικέτα με το όνομα αντικειμένου 'label' στη μεταβλητή location.

Με την εντολή hasitem = list.count() καταχωρείται ο αριθμός των αντικειμένων της λίστας στη μεταβλητή hasitem ενώ με την εντολή hasloc = location.text() το κείμενο της ετικέτας καταχωρείται στη μεταβλητή hasloc.

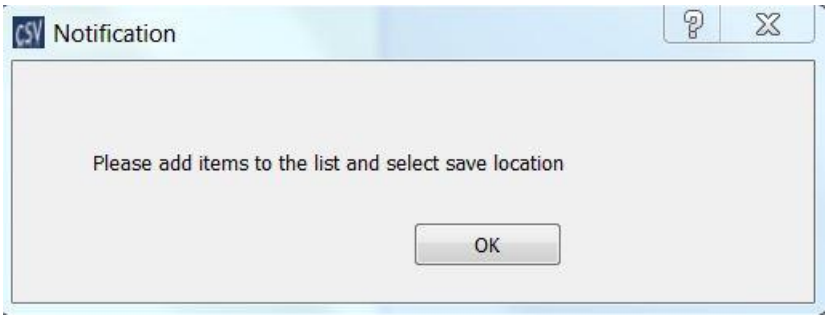
Στη συνέχεια με τη χρήση μιας συνθήκης if (εάν… τότε..) ελέγχεται ο αριθμός των αντικειμένων της λίστας και το κείμενο της ετικέτας για τη διεύθυνση αποθήκευσης.

Εάν ο αριθμός των αντικειμένων της λίστας είναι μηδέν και το κείμενο της ετικέτας κενό, τότε καλείται η συνάρτηση noitem_nosave του αρχείου dialogs.py.

```
#Notify when no item no save has been registered
   def noitem_nosave(self):
        dlg = QDialog(self)
        dlg.resize(500,150)
        dlg.setWindowTitle("Notification")
        l1 = QLabel("Please add items to the list and select save location", dlg)
        l1.move(50,55)
        b1 = QPushButton("OK",dlg)
        b1.move(250,100)
        b1.clicked.connect(dlg.close)
        dlg.exec_()
```

Εικόνα 31 *Συνάρτηση noitem_nosave*

Η συνάρτηση noitem_nosave δημιουργεί ένα παράθυρο με κείμενο “Please add items to the list and select save location” το οποίο σημαίνει «Παρακαλώ προσθέστε αντικείμενα (αρχεία) στη λίστα και επιλέξτε διεύθυνση αποθήκευσης». Το παράθυρο παρέχει και το κουμπί OK στον χρήστη για να μπορεί να κλείσει το παράθυρο εφόσον το διαβάσει.



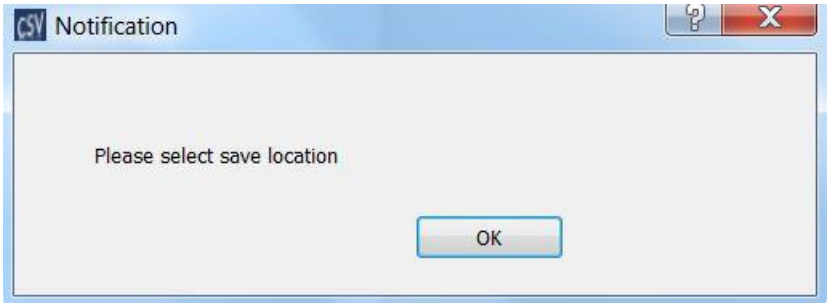
Εικόνα 32 *Παράθυρο noitem_nosave ειδοποίησης*

Εάν ο αριθμός των αντικειμένων της λίστας δεν είναι μηδέν, αλλά το κείμενο της ετικέτας για τη διεύθυνση αποθήκευσης κενό, τότε καλείται η συνάρτηση nosave του αρχείου dialogs.py.

```
#Notify when no save has been registered
   def nosave(self):
            dlg = QDialog(self)
            dlg.resize(500,150)
            dlg.setWindowTitle("Notification")
            l1 = QLabel("Please select save location", dlg)
            l1.move(50,55)
            b1 = QPushButton("OK",dlg)
            b1.move(250,100)
            b1.clicked.connect(dlg.close)
            dlg.exec_()
```

Εικόνα 33 *Συνάρτηση nosave*

Η συνάρτηση nosave δημιουργεί ένα παράθυρο με κείμενο “Please select save location” το οποίο σημαίνει «Παρακαλώ επιλέξτε τοποθεσία αποθήκευσης». Το παράθυρο παρέχει και το κουμπί OK στον χρήστη για να μπορεί να κλείσει το παράθυρο εφόσον το διαβάσει.



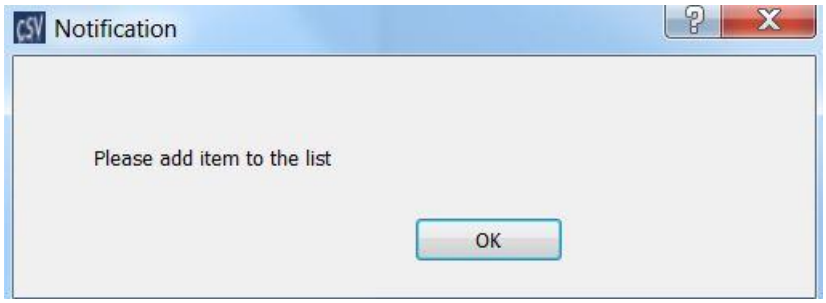
Εικόνα 34 *Παράθυρο nosave ειδοποίησης*

Εάν ο αριθμός των αντικειμένων της λίστας είναι μηδέν, όμως το κείμενο της ετικέτας με την διεύθυνση αποθήκευσης δεν είναι κενό, τότε καλείται η συνάρτηση noitem του αρχείου dialogs.py.

```
#Notify when no item has been added
   def noitem(self):
                            dlg = QDialog(self)
                            dlg.resize(500,150)
                            dlg.setWindowTitle("Notification")
                            l1 = QLabel("Please add item to the list", dlg)
                            l1.move(50,55)
                            b1 = QPushButton("OK",dlg)
                            b1.move(250,100)
                            b1.clicked.connect(dlg.close)
                            dlg.exec_()
```

Εικόνα 35 *Συνάρτηση noitem*

Η συνάρτηση noitem δημιουργεί ένα παράθυρο με κείμενο “Please add item to the list” το οποίο σημαίνει «Παρακαλώ προσθέστε αντικείμενο στη λίστα». Το παράθυρο παρέχει και το κουμπί OK στον χρήστη για να μπορεί να κλείσει το παράθυρο εφόσον το διαβάσει.



Εικόνα 36 *Παράθυρο noitem ειδοποίησης*

Με αυτό τον τρόπο γίνεται έλεγχος, πριν τρέξει η επεξεργασία των αρχείων ή του αρχείου εάν είναι μόνο ένα, για να γίνει ομαλά η διεργασία.

5.4 ΚΩΔΙΚΑΣ ΤΟΥ ΚΟΥΜΠΙΟΥ Clear

Το πλήκτρο **clear** αδειάζει τη λίστα με τα αρχεία,

```
clear = self.findChild(QtWidgets.QPushButton, 'pushButton_3')
clear.clicked.connect(self.clearlist)
```

Εικόνα 37 *Κώδικας για το κουμπί clear*

έχει σαν όνομα αντικειμένου το “pushButton_3” το οποίο καταχωρείται στη μεταβλητή clear και όταν πατηθεί καλείται η συνάρτηση clearlist,

```
#Clear list
 def clearlist(self):
     Preferences.clearthelist(self)
```

Εικόνα 38 *Συνάρτηση clearlist*

η οποία στη συνέχεια καλεί τη συνάρτηση clearthelist του αρχείου preferences.py.

```
class Preferences(QPushButton):
#0)clear the list items
    def clearthelist(self):
        list = self.findChild(QListWidget, 'listWidget')
        list.clear()
```

Εικόνα 39 *Συνάρτηση clearthelist*

Η λίστα με όνομα αντικειμένου 'listWidget' καταχωρείται στη μεταβλητή list και με την εντολή list.clear() η λίστα αδειάζει και μένει κενή.

5.5 ΚΩΔΙΚΑΣ ΤΩΝ ΕΠΙΛΟΓΩΝ ΔΙΑΜΟΡΦΩΣΗΣ

Σε αυτό το κεφάλαιο βρίσκεται το κύριο τμήμα του κώδικα της εφαρμογής και για αυτό το λόγο δίνεται πιο λεπτομερής επεξήγηση στις εντολές που χρησιμοποιήθηκαν.

Ο χρήστης μπορεί να επιλέξει ανάμεσα σε τέσσερις επιλογές τροποποίησης των αρχείων προς επεξεργασία. Επιπλέον, έχει τη δυνατότητα να επιλέξει παραπάνω από μία επιλογή ταυτόχρονα, προκειμένου να επεξεργασθούν τα αρχεία του.

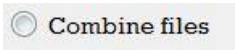
Αυτό γίνεται με τη χρήση πολλαπλής συνθήκης if (εάν… τότε…)

```
else:
    dtm = self.findChild(QtWidgets.QRadioButton, 'radioButton')
    ttc = self.findChild(QtWidgets.QCheckBox, 'checkBox')
    etz = self.findChild(QtWidgets.QCheckBox, 'checkBox_2')
    fti = self.findChild(QtWidgets.QCheckBox, 'checkBox_3')
    if ttc.isChecked() and not dtm.isChecked():
        Preferences.makecolumns(self)
    elif dtm.isChecked() and ttc.isChecked():
        Preferences.concatenate(self)
    elif dtm.isChecked() and not ttc.isChecked():
        Preferences.onlyconc(self)
```

Εικόνα 40 *Δεύτερο σκέλος συνάρτησης execute*

Τα αντικείμενα των σημείων ελέγχου καταχωρούνται σε μεταβλητές και στη συνέχεια χρησιμοποιούνται στον κώδικα. Αναλυτικότερα:

Το αντικείμενο με ονομασία 'radioButton' καταχωρείται στη μεταβλητή dtm και στη συνέχεια, μέσα στη συνθήκη if, ελέγχεται η κατάστασή της (εάν επέλεξε ή όχι ο χρήστης την επιλογή με το radioButton).



Εικόνα 41 *Ραδιοκουμπί συνδυασμού αρχείων*

Εάν έχει επιλεχθεί από τον χρήστη, τότε στόχος είναι να ενωθούν όλα τα αρχεία csv της λίστας στην οποία πρόσθεσε ο χρήστης. Αυτό επιτυγχάνεται με την συνάρτηση onlyconc του αρχείου preferences.py.

```
def onlyconc(self):
      list = self.findChild(QListWidget, 'listWidget')
      location = self.findChild(QLabel , 'label')
      items = []
      li= []
      for index in range(list.count()):
          items.append(list.item(index))
      for item in items:
          nam = item.text()
          myfile = 'allinone.csv'
          pathof = location.text()
          final = pathof + '\\' + myfile
          df = pandas.read_csv(nam, sep =',' , encoding="utf-16")
          li.append(df)
          frame = pandas.concat(li, axis=0, ignore_index=True)
          frame.to_csv(final)
```

Εικόνα 42 *Συνάρτηση onlyconc*

Στην αρχή της συνάρτησης καταχωρούνται τα αντικείμενα της «λίστας των αντικειμένων» και της «ετικέτας διεύθυνσης αποθήκευσης» στις μεταβλητές list και location αντίστοιχα.

Κατόπιν δημιουργούνται δύο κενές λίστες, η items[] και η li[].

Με την πρώτη επανάληψη for (συνθήκη επανάληψης Για..) προσθέτουμε κάθε αρχείο της λίστας αντικειμένων στη λίστα items[].

Στην δεύτερη επανάληψη for καταχωρείται το όνομα του αρχείου στη μεταβλητή nam. Ακολούθως, στη μεταβλητή myfile αναφέρεται το όνομα του τελικού αρχείου και στη συνέχεια στη μεταβλητή pathof αναφέρεται το κείμενο της ετικέτας διεύθυνσης αποθήκευσης.

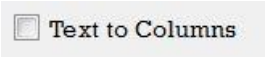
Η τελική διεύθυνση, μαζί με το όνομα του παραγόμενου αρχείου, καταχωρείται στη μεταβλητή final.

Με τη χρήση της βιβλιοθήκης της python, την pandas, διαβάζεται το αρχείο csv και προστίθεται στη λίστα li[] με την εντολή li.append(df).

Τέλος δημιουργείται μια νέα μεταβλητή, η frame, στην οποία γίνεται η συγχώνευση των αρχείων csv με την εντολή pandas.concat(li, axis = 0, ignore_index = True)

και ολοκληρώνοντας, φτιάχνεται το τελικό αρχείο csv με την εντολή frame.to_csv(final).

Το αντικείμενο με ονομασία ‘checkBox’ καταχωρείται στη μεταβλητή ttc και στη συνέχεια, μέσα στη συνθήκη if, ελέγχεται η κατάστασή της (εάν επέλεξε δηλαδή ή όχι ο χρήστης την επιλογή με το checkBox).



Εικόνα 43 *Κουμπί τσεκαρίσματος για μετατροπή σε στήλες*

Εάν η μεταβλητή έχει επιλεχθεί από το χρήστη, τότε ο στόχος είναι να μετατραπούν σε στήλες οι τιμές από όλα τα αρχεία csv της λίστας στην οποία πρόσθεσε ο χρήστης. Αυτό επιτυγχάνεται με την συνάρτηση makecolumns του αρχείου preferences.py.

```
def makecolumns(self):
    list = self.findChild(QListWidget, 'listWidget')
    location = self.findChild(QLabel , 'label')
    items = []
    for index in range(list.count()):
        items.append(list.item(index))
    for item in items:
        nam = item.text()
        myfile = path.basename(nam)
        pathof = location.text()
        final = pathof + '\\' + myfile
        df=pandas.read_csv(nam,sep =',' , encoding="utf-16")
        df.to_csv(final, sep= ';')
```

Εικόνα 44 *Συνάρτηση makecolumns*

Στην αρχή της συνάρτησης καταχωρούνται τα αντικείμενα της «λίστας των αντικειμένων» και της «ετικέτας διεύθυνσης αποθήκευσης» στις μεταβλητές list και location αντίστοιχα.

Κατόπιν δημιουργείται μία κενή λίστα, η items[].

Με την πρώτη επανάληψη for (συνθήκη επανάληψης Για..) προσθέτουμε κάθε αρχείο της λίστας αντικειμένων στη λίστα items[].

Στην δεύτερη επανάληψη for καταχωρείται το όνομα του αρχείου στη μεταβλητή nam. Κατόπιν, στη μεταβλητή myfile αναφέρεται το όνομα του τελικού αρχείου και στη συνέχεια στη μεταβλητή pathof το κείμενο της ετικέτας διεύθυνσης αποθήκευσης.

Η τελική διεύθυνση, μαζί με το όνομα του παραγόμενου αρχείου, καταχωρείται στη μεταβλητή final.

Με τη χρήση της βιβλιοθήκης της python, την pandas, διαβάζεται το αρχείο csv με την εντολή pandas.read_csv() όπου και καταχωρείται στη μεταβλητή df.

Τέλος, φτιάχνεται το τελικό αρχείο csv με την εντολή df.to_csv(final, sep = ';'). Το sep είναι ο διαχωριστής για τις τιμές και θέτοντάς τον ως ";" μετατρέπονται τα κόμματα σε στήλες για το Excel.

Εάν, για παράδειγμα, επιλεχθεί και το checkBox και το radioButton -δηλαδή ο χρήστης θέλει να μετατραπούν οι τιμές των αρχείων του σε στήλες αλλά και να συγχωνευθούν- τότε, μέσω της πολλαπλής συνθήκης if, θα κληθεί η συνάρτηση concatenate του αρχείου preferences.py.

```
def concatenate(self):
    list = self.findChild(QListWidget, 'listWidget')
    location = self.findChild(QLabel , 'label')
    items = []
    li= []
    for index in range(list.count()):
        items.append(list.item(index))
    for item in items:
        nam = item.text()
        myfile = 'allinone.csv'
        pathof = location.text()
        final = pathof + '\\' + myfile
        df = pandas.read_csv(nam, sep =',' , encoding="utf-16")
        li.append(df)
        frame = pandas.concat(li, axis=0, ignore_index=True)
        frame.to_csv(final, sep= ';')
```

Εικόνα 45 *Συνάρτηση concatenate*

Στην αρχή της συνάρτησης καταχωρούνται τα αντικείμενα της «λίστας των αντικειμένων» και της «ετικέτας διεύθυνσης αποθήκευσης» στις μεταβλητές list και location αντίστοιχα.

Κατόπιν δημιουργούνται δύο κενές λίστες, η items[] και η li[].

Με την πρώτη επανάληψη for (συνθήκη επανάληψης Για..) προστίθεται κάθε αρχείο της λίστας αντικειμένων στη λίστα items[].

Στην δεύτερη επανάληψη for καταχωρείται το όνομα του αρχείου στη μεταβλητή nam. Κατόπιν, στη μεταβλητή myfile αναφέρεται το όνομα του τελικού αρχείου και στη συνέχεια στη μεταβλητή pathof το κείμενο της ετικέτας διεύθυνσης αποθήκευσης.

Η τελική διεύθυνση μαζί με το όνομα του παραγόμενου αρχείου καταχωρείται στη μεταβλητή final.

Με τη χρήση της βιβλιοθήκης της python, την pandas, διαβάζεται το αρχείο csv και προστίθεται στη λίστα li[] με την εντολή li.append(df).

Τέλος, δημιουργείται μια νέα μεταβλητή, η frame, στην οποία γίνεται η συγχώνευση των αρχείων csv με την εντολή pandas.concat(li, axis = 0, ignore_index = True).

Ολοκληρώνοντας, φτιάχνεται το τελικό αρχείο csv με την εντολή frame.to_csv(final, sep = ‘;’) έτσι ώστε ο διαχωριστής του συγχωνευμένου αρχείου να είναι το ελληνικό ερωτηματικό (;) και τα κόμματα να μετατραπούν σε στήλες.

Με παρόμοιο τρόπο λειτουργεί ο κώδικας όταν ο χρήστης επιλέγει παραπάνω από μία επιλογή διαμόρφωσης και τρέχει τη συνάρτηση που χρειάζεται αναλόγως.

Το αντικείμενο με ονομασία ‘checkBox_2’ καταχωρείται στη μεταβλητή etz. Στην συνέχεια, μέσα στη συνθήκη if, ελέγχεται η κατάστασή της (εάν δηλαδή επέλεξε ή όχι ο χρήστης την επιλογή με το checkBox_2).

Εικόνα 46 *Κουμπί τσεκαρίσματος για μετατροπή κενών τιμών σε μηδέν*

Εάν έχει επιλεχθεί από το χρήστη, τότε στόχος είναι να μετατραπούν οι κενές ή άδειες τιμές στην τιμή μηδέν σε όλα τα αρχεία csv της λίστας στην οποία πρόσθεσε αντικείμενα ο χρήστης. Αυτό επιτυγχάνεται με την συνάρτηση emptozero του αρχείου preferences.py.

```
def emptozero(self):
    list = self.findChild(QListWidget, 'listWidget')
    location = self.findChild(QLabel , 'label')
    items = []
    for index in range(list.count()):
        items.append(list.item(index))
    for item in items:
        nam = item.text()
        myfile = 'allinone.csv'
        pathof = location.text()
        final = pathof + '\\' + myfile
        df = pandas.read_csv(nam, sep =',' , encoding="utf-16")
        c = df.fillna(0, inplace=True)
        c.to_csv(final, sep= ';')
```

Εικόνα 47 *Συνάρτηση emptozero*

Στην αρχή της συνάρτησης καταχωρούνται τα αντικείμενα της «λίστας των αντικειμένων» και της «ετικέτας διεύθυνσης αποθήκευσης» στις μεταβλητές list και location αντίστοιχα.

Κατόπιν δημιουργείται μία κενή λίστα, η items[].

Με την πρώτη επανάληψη for (συνθήκη επανάληψης Για..) προστίθεται κάθε αρχείο της λίστας αντικειμένων στη λίστα items[].

Στην δεύτερη επανάληψη for καταχωρείται το όνομα του αρχείου στη μεταβλητή nam. Κατόπιν, στη μεταβλητή myfile αναφέρεται το όνομα του τελικού αρχείου και στη συνέχεια στη μεταβλητή pathof αναφέρεται το κείμενο της ετικέτας διεύθυνσης αποθήκευσης.

Η τελική διεύθυνση μαζί με το όνομα του παραγόμενου αρχείου καταχωρείται στη μεταβλητή final.

Με τη χρήση της βιβλιοθήκης της python, την pandas, διαβάζεται το αρχείο csv με την εντολή pandas.read_csv() όπου και καταχωρείται στη μεταβλητή df.

Με την εντολή c = df.fillna(0, inplace = True) αντικαθίστανται όλες οι άδειες ή κενές τιμές με τον αριθμό μηδέν.

Τέλος, το αρχείο εξάγεται στην προκαθορισμένη διεύθυνση αποθήκευσης.

Το αντικείμενο με ονομασία ‘checkBox_3’ καταχωρείται στη μεταβλητή fti και στη συνέχεια μέσα στη συνθήκη if ελέγχεται η κατάστασή της (εάν επέλεξε δηλαδή ή όχι ο χρήστης την επιλογή με το checkBox_3).

Εικόνα 48 *Κουμπί τσεκαρίσματος για μετατροπή δεκαδικών σε ακέραιους*

Εάν έχει επιλεχθεί από τον χρήστη, τότε στόχος είναι να μετατραπούν οι δεκαδικές τιμές σε ακέραιες σε όλα τα αρχεία csv της λίστας στην οποία πρόσθεσε αντικείμενα ο χρήστης και αυτό επιτυγχάνεται με την συνάρτηση fltointe του αρχείου preferences.py.

```
def fltointe(self):
    list = self.findChild(QListWidget, 'listWidget')
    location = self.findChild(QLabel , 'label')
    items = []
    for index in range(list.count()):
        items.append(list.item(index))
    for item in items:
        nam = item.text()
        myfile = 'allinone.csv'
        pathof = location.text()
        final = pathof + '\\' + myfile
        df = pandas.read_csv(nam, sep =',' , encoding="utf-16")
        c = df.columns[df.dtypes == float]
        df[c] = df[c].astype(int)
        df[c].to_csv(final, sep= ';')
```

Εικόνα 49 *Συνάρτηση fltointe*

Στην αρχή της συνάρτησης καταχωρούνται τα αντικείμενα της «λίστας των αντικειμένων» και της «ετικέτας διεύθυνσης αποθήκευσης» στις μεταβλητές list και location αντίστοιχα.

Κατόπιν δημιουργείται μία κενή λίστα, η items[].

Με την πρώτη επανάληψη for (συνθήκη επανάληψης Για..) προστίθεται κάθε αρχείο της λίστας αντικειμένων στη λίστα items[].

Στην δεύτερη επανάληψη for καταχωρείται το όνομα του αρχείου στη μεταβλητή nam. Κατόπιν, στη μεταβλητή myfile αναφέρεται το όνομα του τελικού αρχείου και στη συνέχεια στη μεταβλητή pathof αναφέρεται το κείμενο της ετικέτας διεύθυνσης αποθήκευσης.

Η τελική διεύθυνση, μαζί με το όνομα του παραγόμενου αρχείου, καταχωρείται στη μεταβλητή final.

Με τη χρήση της βιβλιοθήκης της python, την pandas, διαβάζεται το αρχείο csv με την εντολή pandas.read_csv() όπου και καταχωρείται στη μεταβλητή df.

Με την εντολή c = df.columns[df.dtypes == float] και την εντολή df[c] = df[c].astype(int) αντικαθίστανται όλες οι δεκαδικές τιμές με τις αντίστοιχες ακέραιες.

Τέλος, εξάγεται το αρχείο στην προκαθορισμένη διεύθυνση αποθήκευσης.

ΣΥΜΠΕΡΑΣΜΑΤΑ

Η ανάπτυξη της συγκεκριμένης εφαρμογής έχει ως κύριο σκοπό να διευκολύνει την λειτουργία της επιχείρησης και να αποτελέσει ένα χρήσιμο εργαλείο για τον αναλυτή δεδομένων, καθώς και οποιονδήποτε θα θελήσει να διαχειρισθεί, μέσω κάποιων λειτουργιών, πολλά αρχεία csv ταυτόχρονα.

Στην προσπάθεια δημιουργίας της εφαρμογής, χρησιμοποιήθηκαν ως γλώσσα προγραμματισμού, η γλώσσα python καθώς και ένα γραφικό περιβάλλον σχεδιασμού και ανάπτυξης εφαρμογών, το QtCreator.

Κρίθηκε απαραίτητη τόσο η παροχή ενός μεγάλου μεγέθους αρχείων όσο και η παροχή αρχείων με συγκεκριμένες ιδιομορφίες όπως η έλλειψη τιμών και η παρουσία πολλαπλών δεκαδικών ψηφίων στις τιμές που περιείχαν.

Σημαντική εξίσου λεπτομέρεια ήταν το περιβάλλον χρήστη ώστε να μπορεί να μάθει κανείς την εφαρμογή και το πώς να την χρησιμοποιεί γρήγορα και εύκολα χωρίς πολλούς αντιπερισπασμούς στην οθόνη απεικόνισης της εφαρμογής.

Μέσω της εφαρμογής επιτεύχθηκε η δυνατότητα να μπορεί ένας επιστήμονας (data scientist) ή αναλυτής δεδομένων (data analyst) να κάνει κάποιες βασικές τροποποιήσεις πλήθους αρχείων csv σύντομα και με μεγάλη ευκολία.

Οι τροποποιήσεις αυτές περιλαμβάνουν ένωση αρχείων, χωρισμό τιμών από διαχωριστή κόμμα σε στήλες σε αρχεία excel, αντικατάσταση κενών τιμών με το 0 (μηδέν), στρογγυλοποίηση τιμών κ.α.

Η εφαρμογή θα μπορούσε να αναβαθμιστεί και να επεκταθεί ποικιλοτρόπως. Θα αποτελούσε έτσι ένα πολύ-λειτουργικό πρόγραμμα στο οποίο θα υπάρχουν επιπλέον δυνατότητες διαμόρφωσης όπως είναι η επεξεργασία ημερομηνίας και ώρας στα αρχεία, η αλλαγή του διαχωριστή, ανίχνευση συγκεκριμένης τιμής, αναζήτηση εύρους τιμών βάσει σύγκρισης.

Ενδεχομένως, ένας αναλυτής δεδομένων θα έδινε επιπλέον χρήσιμες προτάσεις αναλόγως το περιβάλλον εργασίας και τον τύπο αρχείων που συνήθως καλείται να τροποποιήσει και εκμεταλλευτεί.

Βιβλιογραφία - Ιστοσελίδες

1. Hall A. M., Frank E., Witten H. I (2011): Data Mining, Practical Machine Learning Tools and Techniques

2. Kumar V., Steinbach M. (2006): Introduction to Data Mining, Addison Wesley

3. Freitas, A.A., (1998): A Survey of Parallel Data Mining, in Proceedings of 2nd International Conference on the Practical Applications of Knowledge Discovery and Data Mining

4. Κ. Μαγκούτης, Χ. Νικολάου, (2005): ΕΙΣΑΓΩΓΗ ΣΤΟΝ ΑΝΤΙΚΕΙΜΕΝΟΣΤΡΑΦΗ ΠΡΟΓΡΑΜΜΑΤΙΣΜΟ ΜΕ ΡΥΤΗΟΝ

5. ΜΑΝΗΣ ΓΕΩΡΓΙΟΣ, (2005): ΕΙΣΑΓΩΓΗ ΣΤΟΝ ΠΡΟΓΡΑΜΜΑΤΙΣΜΟ ΜΕ ΑΡΩΓΟ ΤΗ ΓΛΩΣΣΑ ΡΥΤΗΟΝ

6. Νικόλαος Σαμαράς, Κωνσταντίνος Τσιπλίδης, (2019): ΤΟ ΒΙΒΛΙΟ ΤΗΣ ΡΥΤΗΟΝ

7. Microsoft Press, Microsoft Corporation Staff, (1994): THE WINDOWS INTERFACE GUIDELINES FOR SOFTWARE DESIGN

8. Benjamin Baka, (2019): Getting started with Qt 5

9. Marek Krajewski, (2019): Hands-On High Performance Programming with Qt 5

10. Guilaume Lazar, Robin Penea, (2018): Mastering Qt 5

11. Everett N. McKay, (2013): UI is Communication: How to Design Intuitive, User Centered Interfaces by Focusing on Effective Communication

12. Eric Matthes, (2019): Python Crash Course, 2nd Edition: A Hands-On, Project-Based Introduction to Programming

13. Al Sweigart (2019): Automate the Boring Stuff with Python, 2nd Edition: Practical Programmig for Total Beginners

14. Wes McKinney, (2017): Python for Data Analysis: Data Wrangling with Pandas, NumPy, and IPython

15. R. Lyman Ott and Micheal T. Longnecker, (2015): An Introduction to Statistical Methods and Data Analysis

16. https://www.qt.io/

17. https://www.riverbankcomputing.com/static/Docs/PyQt5/

18. https://pythonspot.com/pyqt5-file-dialog

19. https://www.shanelynn.ie/python-pandas-read_csv-load-data-from-csv-files/

20. https://stackoverflow.com/questions/9234560/find-all-csv-files-in-a-directory-using-python/12280052

21. https://www.geeksforgeeks.org/python-os-path-basename-method/

22. https://www.geeksforgeeks.org/python-pandas-split-strings-into-two-list-columns-using-str-split/

23. https://stackoverflow.com/questions/36606771/pyqt-how-do-i-update-a-label

24. https://www.tutorialspoint.com/pyqt/pyqt_qlabel_widget.htm

25. https://docs.python.org/3/library/csv.html

26. https://www.w3schools.com/python/python_intro.asp

27. https://www.computerhope.com/jargon/g/gui.htm

28. https://www.rejoin.gr/blog-news/4783-rejoin-h-shmasia-twn-data-analytics-stis-epixeiriseis

Πίνακας Διαγραμμάτων