Με βάση το αποθετήριο GitHub για την Εφαρμογή Εξόρυξης και Ανάλυσης Δεδομένων, ακολουθεί μια λεπτομερής επισκόπηση του σχεδιασμού και της υλοποίησης της εφαρμογής:

Σχεδιασμός της εφαρμογής

1. Επισκόπηση της αρχιτεκτονικής
Η εφαρμογή έχει σχεδιαστεί ως διαδικτυακή πλατφόρμα με χρήση του Streamlit, το οποίο επιτρέπει την ταχεία ανάπτυξη διαδραστικών εφαρμογών δεδομένων. Η αρχιτεκτονική μπορεί να αναλυθεί σε διάφορα βασικά στοιχεία:

Frontend: Η διεπαφή χρήστη έχει κατασκευαστεί με τη χρήση του Streamlit, παρέχοντας μια διαισθητική και διαδραστική εμπειρία για τους χρήστες για τη μεταφόρτωση δεδομένων, την οπτικοποίηση των αποτελεσμάτων και την εκτέλεση αναλύσεων.
Backend: Το backend αποτελείται από την Python που χειρίζεται τις εργασίες επεξεργασίας δεδομένων, ανάλυσης και μηχανικής μάθησης. Αυτό περιλαμβάνει τη φόρτωση δεδομένων, την εκτέλεση διερευνητικής ανάλυσης δεδομένων (EDA) και την εκτέλεση αλγορίθμων μηχανικής μάθησης.
Αποθήκευση δεδομένων: Η εφαρμογή υποστηρίζει τη φόρτωση δεδομένων από διάφορες μορφές, όπως CSV, Excel και TSV, επιτρέποντας στους χρήστες να εργάζονται απρόσκοπτα με διαφορετικά σύνολα δεδομένων.

2. Βασικά χαρακτηριστικά
Η εφαρμογή περιλαμβάνει διάφορα σημαντικά χαρακτηριστικά:
Φόρτωση δεδομένων**: Οι χρήστες μπορούν να φορτώνουν αρχεία δεδομένων σε μορφή πίνακα (CSV, Excel, TSV) μέσω της διεπαφής.
Οπτικοποίηση δεδομένων**: Η εφαρμογή παρέχει τρισδιάστατες απεικονίσεις με τη χρήση τεχνικών όπως η PCA (Ανάλυση Κύριων Συνιστωσών) και η UMAP (Ομοιόμορφη Προσέγγιση και Προβολή Πολλαπλότητας) για να βοηθήσει τους χρήστες να κατανοήσουν πολύπλοκα σύνολα δεδομένων.
Διερευνητική ανάλυση δεδομένων (EDA)**: Οι χρήστες μπορούν να δημιουργήσουν διάφορα διαγράμματα EDA για να διερευνήσουν τις κατανομές και τις σχέσεις των δεδομένων.
Επιλογή χαρακτηριστικών: Η εφαρμογή περιλαμβάνει λειτουργικότητα για την επιλογή σχετικών χαρακτηριστικών για μοντέλα μηχανικής μάθησης.
Ταξινόμηση μηχανικής μάθησης: Οι χρήστες μπορούν να εφαρμόζουν αλγορίθμους ταξινόμησης, όπως K-Nearest Neighbors (KNN) και Support Vector Machines (SVM) στα σύνολα δεδομένων τους.
Σύγκριση επιδόσεων**: Η εφαρμογή επιτρέπει στους χρήστες να συγκρίνουν την απόδοση του μοντέλου πριν και μετά την επιλογή χαρακτηριστικών.

3. Σχεδιασμός διεπαφής χρήστη
Η διεπαφή χρήστη έχει σχεδιαστεί ώστε να είναι φιλική προς τον χρήστη και διαισθητική. Τα βασικά στοιχεία περιλαμβάνουν:
Widget φόρτωσης αρχείων**: Μια απλή περιοχή drag-and-drop για να ανεβάζουν οι χρήστες τα αρχεία δεδομένων τους.
Πλοήγηση στην πλαϊνή γραμμή**: Μια πλαϊνή μπάρα που επιτρέπει στους χρήστες να πλοηγούνται μεταξύ διαφορετικών λειτουργιών (π.χ. φόρτωση δεδομένων, οπτικοποίηση, ανάλυση).
Διαδραστικά διαγράμματα**: Οπτικοποιήσεις με τις οποίες οι χρήστες μπορούν να αλληλεπιδράσουν για να αποκτήσουν γνώσεις σχετικά με τα δεδομένα τους.
Εφαρμογή της εφαρμογής

1. Δομή αποθετηρίου
Το αποθετήριο είναι οργανωμένο σε διάφορους καταλόγους:

src/**: Περιέχει τον κύριο κώδικα της εφαρμογής.
lib/**: Περιλαμβάνει βιβλιοθήκες και βοηθητικές λειτουργίες.
functions/**: Περιέχει ειδικές συναρτήσεις για την επεξεργασία και την ανάλυση δεδομένων.
pages/**: Περιέχει διάφορες σελίδες της εφαρμογής Streamlit.
Dockerfile**: Χρησιμοποιείται για τη δημιουργία εμπορευματοκιβωτίων της εφαρμογής για εύκολη ανάπτυξη.

2. Εγκατάσταση και ρύθμιση
Για να εγκαταστήσουν την εφαρμογή τοπικά, οι χρήστες μπορούν να ακολουθήσουν τα εξής βήματα:
Κλωνοποιήστε το αποθετήριο:

```
git clone https://github.com/sudo455/texnologia_logismikoy_sptember_2024.git
cd texnologia_logismikoy_sptember_2024/src
```

Δημιουργήστε και ενεργοποιήστε ένα εικονικό περιβάλλον:

```
python -m venv venv
source venv/bin/activate # Στα Windows, χρησιμοποιήστε venv\Scripts\activate
```

Εγκαταστήστε τα απαιτούμενα πακέτα:

```
pip install -r requirements.txt
```

Εκτελέστε την εφαρμογή Streamlit:
Ανοίξτε το πρόγραμμα περιήγησής σας και μεταβείτε στη διεύθυνση http://localhost:8501.

3. Βασικές λεπτομέρειες εφαρμογής
Φόρτωση δεδομένων**: Η εφαρμογή χρησιμοποιεί το Pandas για την ανάγνωση αρχείων δεδομένων και τη μετατροπή τους σε DataFrames για ανάλυση.
Οπτικοποίηση**: Χρησιμοποιούνται βιβλιοθήκες όπως οι Matplotlib και Plotly για τη δημιουργία διαδραστικών οπτικοποιήσεων.
Μηχανική μάθηση**: Το Scikit-learn χρησιμοποιείται για την υλοποίηση αλγορίθμων μηχανικής μάθησης, επιτρέποντας στους χρήστες να εκπαιδεύουν και να αξιολογούν μοντέλα εύκολα.
Επιλογή χαρακτηριστικών**: Τεχνικές όπως η Recursive Feature Elimination (RFE) μπορούν να εφαρμοστούν για να βοηθήσουν τους χρήστες να επιλέξουν τα πιο σχετικά χαρακτηριστικά.

4. Ανάπτυξη
Η εφαρμογή μπορεί να αναπτυχθεί με τη χρήση του Docker για συνοχή σε διάφορα περιβάλλοντα. Οι χρήστες μπορούν να δημιουργήσουν και να εκτελέσουν την εικόνα Docker με τις ακόλουθες εντολές:
Κατασκευή της εικόνας Docker

```
sudo docker build -t texnologia_logismikoy:latest .
```

Εκτελέστε το δοχείο Docker

```
sudo docker run -d -p 8501:80 --name texnologia_logismikoy
ghcr.io/sudo455/texnologia_logismikoy:latest
```

Συμπέρασμα
Η εφαρμογή εξόρυξης και ανάλυσης δεδομένων έχει σχεδιαστεί για να παρέχει στους χρήστες ένα ολοκληρωμένο εργαλείο για την ανάλυση δεδομένων και τη μηχανική μάθηση. Η αρχιτεκτονική της αξιοποιεί σύγχρονες τεχνολογίες ιστού και βιβλιοθήκες Python για να προσφέρει μια διαδραστική και φιλική προς τον χρήστη εμπειρία. Οι λεπτομέρειες υλοποίησης διασφαλίζουν ότι οι χρήστες μπορούν εύκολα να ρυθμίσουν, να εκτελέσουν και να αξιοποιήσουν την εφαρμογή για τις ανάγκες τους σε ανάλυση δεδομένων.

Συνεισφορές των μελών της ομάδας
1. Άγγελος Μωραΐτης (inf2021163)

Φόρτωση δεδομένων: Ανάπτυξη της λειτουργικότητας για τη μεταφόρτωση και την ανάγνωση διαφόρων μορφών δεδομένων (CSV, Excel, TSV) στην εφαρμογή. Αυτό περιελάμβανε τη χρήση βιβλιοθηκών όπως η Pandas για να εξασφαλιστεί η απρόσκοπτη ενσωμάτωση των δεδομένων των χρηστών στην εφαρμογή για ανάλυση.
Προεπεξεργασία:
Εφαρμογή βημάτων προεπεξεργασίας δεδομένων, συμπεριλαμβανομένου του καθαρισμού, της κανονικοποίησης και του μετασχηματισμού δεδομένων. Αυτό διασφαλίζει ότι τα δεδομένα βρίσκονται σε κατάλληλη μορφή για την ανάλυση και τις εργασίες μηχανικής μάθησης.
Επιλογή χαρακτηριστικών:
Σχεδίασε και υλοποίησε τεχνικές επιλογής χαρακτηριστικών για να βοηθήσει τους χρήστες να εντοπίσουν τα πιο σχετικά χαρακτηριστικά για τα μοντέλα τους. Αυτό μπορεί να περιλαμβάνει μεθόδους όπως η Recursive Feature Elimination (RFE) ή άλλες στατιστικές τεχνικές για τη βελτίωση της απόδοσης των μοντέλων.
Αλγόριθμοι ταξινόμησης:
Ενσωμάτωση αλγορίθμων ταξινόμησης μηχανικής μάθησης, συγκεκριμένα K-Nearest Neighbors (KNN) και Support Vector Machines (SVM). Αυτό περιελάμβανε την κωδικοποίηση της λογικής για την εκπαίδευση, την επικύρωση και τη δοκιμή αυτών των μοντέλων, καθώς και την παροχή στους χρήστες μετρικών επιδόσεων.

2. Θεοχάρης Παρίσης (Π2017162)

Οπτικοποίηση: Ανάπτυξη των στοιχείων οπτικοποίησης της εφαρμογής, με έμφαση στη δημιουργία διαδραστικών γραφημάτων και διαγραμμάτων που επιτρέπουν στους χρήστες να εξερευνήσουν οπτικά τα δεδομένα τους. Αυτό περιλαμβάνει τη χρήση βιβλιοθηκών όπως η Matplotlib και η Plotly για την αποτελεσματική αναπαράσταση δεδομένων.
Μείωση διαστάσεων:
Εφαρμογή τεχνικών όπως η ανάλυση κύριων συνιστωσών (PCA) και η ομοιόμορφη προσέγγιση και προβολή πολλαπλών στοιχείων (UMAP) για τη μείωση της διαστατικότητας. Αυτό βοηθά τους χρήστες να απεικονίσουν δεδομένα υψηλών διαστάσεων σε μια πιο ερμηνεύσιμη τρισδιάστατη μορφή.
Σχεδιασμός UI:
Ηγήθηκε του σχεδιασμού της διεπαφής χρήστη, διασφαλίζοντας ότι είναι διαισθητική και φιλική προς τον χρήστη. Αυτό περιελάμβανε τη δημιουργία μιας διάταξης που διευκολύνει την εύκολη πλοήγηση και αλληλεπίδραση με τα χαρακτηριστικά της εφαρμογής.
Ενσωμάτωση:
Ασχολήθηκε με την ενσωμάτωση των frontend και backend στοιχείων της εφαρμογής, διασφαλίζοντας την ομαλή ροή των δεδομένων μεταξύ της διεπαφής χρήστη και της υποκείμενης λογικής επεξεργασίας. Αυτό περιλαμβάνει τη σύνδεση των εισόδων του χρήστη με τις κατάλληλες λειτουργίες επεξεργασίας και οπτικοποίησης δεδομένων.

Για τον κύκλο ζωής έκδοσης λογισμικού προσαρμοσμένο για τη διάθεση της «Εφαρμογής Εξόρυξης και Ανάλυσης Δεδομένων» στο ευρύ κοινό, μπορούμε να χρησιμοποιήσουμε το μοντέλο DevOps. Αυτή η προσέγγιση δίνει έμφαση στη συνεργασία μεταξύ των ομάδων ανάπτυξης και λειτουργίας, εξασφαλίζοντας μια ομαλή και αποτελεσματική διαδικασία έκδοσης. Ο τρόπος με τον οποίο μπορεί να δομηθεί ο κύκλος ζωής της έκδοσης λογισμικού είναι με τα εξής βήματα:

1. Σχεδιασμός
Συλλογή απαιτήσεων: Συλλογή ανατροφοδότηση από τους χρήστες για να κατανοηθούν οι ανάγκες και οι προσδοκίες τους.

Ανάπτυξη πλάνου πορείας: Δημιουργία ενός πλάνου πορείας που περιγράφει τα χαρακτηριστικά, τις βελτιώσεις και τα χρονοδιαγράμματα της εφαρμογής.

2. Ανάπτυξη

Ευέλικτη μεθοδολογία: Χρησιμοποίηση ευέλικτες πρακτικές για την ανάπτυξη της εφαρμογής σε επαναληπτικούς κύκλους (sprints), επιτρέποντας τη συνεχή ανατροφοδότηση και βελτίωση.

Έλεγχος εκδόσεων: Χρήση του Git για τον έλεγχο εκδόσεων για τη διαχείριση των αλλαγών στον κώδικα και την αποτελεσματική συνεργασία μεταξύ των μελών της ομάδας.

3. Συνεχής ολοκλήρωση (CI)

Αυτοματοποιημένη δοκιμή: Εφαρμογή αυτοματοποιημένων δοκιμών για τη διασφάλιση της ποιότητας και της λειτουργικότητας του κώδικα. Αυτό περιλαμβάνει δοκιμές μονάδας, δοκιμές ολοκλήρωσης και δοκιμές αποδοχής από τον χρήστη.

Αυτοματοποίηση κατασκευής: Χρησιμοποίηση εργαλείων CI (π.χ. Jenkins, GitHub Actions) για την αυτοματοποίηση της διαδικασίας κατασκευής, διασφαλίζοντας ότι η εφαρμογή βρίσκεται πάντα σε κατάσταση ανάπτυξης.

4. Συνεχής παράδοση (CD)

Περιβάλλον σταδιοποίησης: Ανάπτυξη της εφαρμογής σε ένα περιβάλλον σταδιοποίησης που μιμείται το περιβάλλον παραγωγής για τις τελικές δοκιμές.

Δοκιμή αποδοχής χρηστών (UAT): Επιτρέψτε σε μια ομάδα χρηστών να δοκιμάσουν την εφαρμογή στο περιβάλλον σταδιοποίησης για τη συλλογή ανατροφοδότησης και τον εντοπισμό τυχόν προβλημάτων.

5. Ανάπτυξη

Έκδοση παραγωγής: Μόλις η εφαρμογή περάσει την UAT, αναπτύσσεται στο περιβάλλον παραγωγής. Αυτό μπορεί να γίνει με τη χρήση containerization (π.χ. Docker) για ευκολότερη διαχείριση και επεκτασιμότητα.

Παρακολούθηση: Εφαρμογή εργαλείων παρακολούθησης (π.χ. Prometheus, Grafana) για την παρακολούθηση της απόδοσης της εφαρμογής και των αλληλεπιδράσεων των χρηστών μετά την ανάπτυξη.

6. Ανατροφοδότηση και επανάληψη

Ανατροφοδότηση χρηστών: Συλλογή feedback από τους χρήστες για τον εντοπισμό περιοχών προς βελτίωση και αιτημάτων για νέα χαρακτηριστικά.

Συνεχής βελτίωση: Χρησιμοποίηση της ανατροφοδότησης για τον σχεδιασμό της μελλοντική επανάληψης της εφαρμογής, διασφαλίζοντας ότι αυτή εξελίσσεται με βάση τις ανάγκες των χρηστών.

7. Συντήρηση

Τακτικές ενημερώσεις: Προγραμματισμένες τακτικές ενημερώσεις για τη διόρθωση σφαλμάτων, τη βελτίωση των επιδόσεων και την προσθήκη νέων χαρακτηριστικών.

Υποστήριξη: Παροχή καναλιών υποστήριξης χρηστών (π.χ. φόρουμ, συνομιλία) για την βοήθεια των χρηστών σε τυχόν προβλήματα που αντιμετωπίζουν.

8. Τεκμηρίωση

Τεκμηρίωση χρήστη: Δημιουργία ολοκληρωμένων εγχειριδίων χρήσης και ηλεκτρονικών πηγών βοήθειας για να βοηθηθούν οι χρήστες στην πλοήγηση στην εφαρμογή.

Τεχνική τεκμηρίωση: Διατήρηση της τεχνικής τεκμηρίωσης για τους προγραμματιστές, ώστε να διευκολύνεται η μελλοντική ανάπτυξη και η εισαγωγή νέων μελών της ομάδας.

Συμπέρασμα

Με την υιοθέτηση του μοντέλου DevOps, η εφαρμογή μπορεί να αναπτυχθεί αποτελεσματικά, να δοκιμαστεί και να κυκλοφορήσει στο ευρύ κοινό. Αυτή η προσέγγιση όχι μόνο ενισχύει τη συνεργασία και την αποτελεσματικότητα, αλλά διασφαλίζει επίσης ότι η εφαρμογή ανταποκρίνεται στις προσδοκίες των χρηστών και προσαρμόζεται στις μεταβαλλόμενες ανάγκες με την πάροδο του χρόνου.

UML diagram