Nume: Enache Stefan
Gherghe Andrei-Daniel
Grupa: 331AC

Proiect-Tehnici de Învățare Automată

Competiția se concentrează pe clasificarea datelor provenite de la pacienți, cu șase caracteristici distincte legate de incidență, înclinare, unghi, pantă, rază și grad. Scopul nostru este să dezvoltăm un model care să poată identifica cu precizie aceste anomalii, contribuind astfel la îmbunătățirea diagnosticului medical.

În rezolvarea proiectului am avut ca sprijin câteva date de care ne-am putut folosi, precum:

- id (Integer): Un identificator unic pentru fiecare punct de date.
- feature_0 până la feature_5 (Float): Șase caracteristici numerice diferite asociate fiecărui punct de date, ele fiind de tip 'Float'.
    - feature_0: Valoare indicând incidența.
    - feature_1: Valoare indicând înclinarea.
    - feature_2: Valoare indicând unghiul.
    - feature_3: Valoare indicând panta.
    - feature_4: Valoare indicând raza.
    - feature_5: Valoare indicând gradul.
- is_anomaly (Binar: 0 sau 1): Aceasta este variabila țintă, valoarea 0 indicând faptul că punctul de date este considerat normal, în timp ce o valoare de 1 indică că punctul de date este o anomalie.

Totodată, ne-au fost puse la dispoziție două fișiere, "testfile" și "trainfile", ce reprezintă obiectele de fișiere deschise pentru fișierele de test și antrenament, noi urmând să returnăm un fișier de tip submission.

Pentru o bună desfășurare a proiectului și pentru a scoate un F1 score cât mai bun în urma submission-urilor realizate, am făcut un research si am descoperit că pentru detectarea anomaliilor punctelor multidimesionale, Support Vector Machine(SVM) e foarte bine optimizat. SVM poate identifica aceste anomalii prin plasarea hiperplanului astfel încât să maximizeze marginea față de majoritatea datelor, punând mai puțină greutate pe punctele de antrenament care sunt departe de hiperplan, adică posibile anomalii. Acesta este un algoritm supervizat în care supervizarea restrânge spectrul la o zonă de anomalii mai mici.

Algoritmul debutează cu inițializarea instanței SVM cu parametrii specificați, precum rata de învățare (learning_rate), numărul de epoci (epochs) și parametrul de regularizare (C), biass si ponderile modelului fiind egalate cu "none". Ulterior, comanda "fit" implementează antrenarea modelului SVM folosind gradient descent. Calculează gradientul funcției de cost și actualizează ponderile și termenul de deplasare în direcția care minimizează funcția de cost. Functia de predicție a modelului SVM este implementată pe baza ponderilor și termenului de deplasare, ea returnând etichetele prezise.

Implementarea algoritmului ales: Am deschis fișierele test.csv și train.csv, le-am citit și am început procesarea datelor de antrenament. Pentru a le parcurge, am folosit "float()" pentru a converti valorile din șiruri la numere în virgulă mobilă și "round()" pentru a rotunji la 2 zecimale. Am construit o listă "train_ids" pentru a stoca identificatorii, o listă "train" pentru a stoca datele (fără identificatori) și o listă "train_labels" pentru a stoca etichetele de clasă. Listele train și test sunt convertite în matrice "Numpy" pentru a putea fi utilizate în

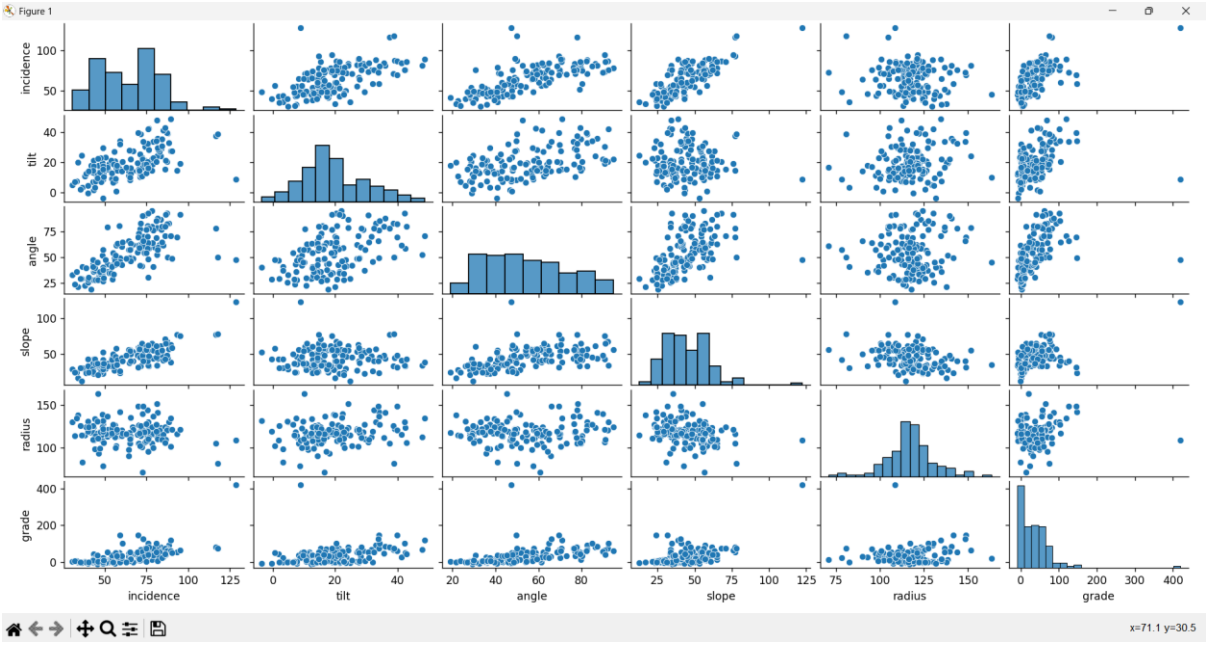
Nume: Enache Stefan
Gherghe Andrei-Daniel
Grupa: 331AC

implementarea modelului SVM, se construiește o listă output_formatat pentru a stoca rezultatele formatate pentru a fi salvate în fișierul CSV final, ce va conține un identificator (id) și valoarea prezisă pentru anomalie (is_anomaly).

La prima încărcare a fișierului submission.csv am obținut un F1 score destul de mare, de 0.79375. Pentru o îmbunătățire a algoritmului, am schimbat numărul de epoci din 100 în 200, modelul având mai mult timp să învețe din datele de antrenare și să ajusteze parametrii săi, progresând până la scorul de 0.80625. De asemenea, am observat că dacă aș scrie fiecare feature cu doar 2 zecimale, scorul meu ar deveni unul mai bun. În timp ce încercam să optimizez algoritmul, am observat si decăderi datorate de creșterea learning_rate-ului, proces numit și overshooting (depășirea minimului global).

Am obținut totodată mai multe grafice și aglomerări de puncte (Figure 1) care se randează în urma compilării fișierului nostru din python ce conține algoritmul SVM. Figura obținută pe baza acestui cod (Figure 1) reprezintă o matrice de diagrame de dispersie care arată relațiile dintre mai multe variabile precum: incidență, înclinare, unghi, pantă, rază și grad. Fiecare diagramă mică reprezintă o diagramă de dispersie a două variabile diferite, cu histograme de-a lungul liniei diagonale. Celelalte diagrame reprezintă relațiile dintre perechile de atribute.

Conform algoritmului KNN (K-Nearest Neighbors) explicat în laboratorul 5, dacă avem o aglomerare de puncte, este influențat în mod semnificativ de etichetele punctelor vecine din acea zonă. Astfel, probabilitatea de a fi clasificat în aceeași clasă cu majoritatea punctelor adunate este crescută.



Totodată, am implementat și algoritmul arborelui de decizie (Decision Trees) făcut în laboratorul 4. Acesta se folosește de o clasă DecisionTree care generează un arbore de decizie pentru probleme de clasificare. Clasa DecisionNode reprezintă un nod în arbore și conține atribute precum indexul caracteristicii, valoarea de decizie și nodurile copil stâng și drept. Gini impurity este calculată pentru nodurile copil stâng și drept pentru a ghida procesul de divizare.

Nume: Enache Stefan
Gherghe Andrei-Daniel
Grupa: 331AC

Pe lângă avantajele pe care le oferă acești arbori de decizie, sunt predispuși la suprapunere și pot fi instabili în fața variațiilor mici ale datelor, care pot duce la arbori cu structuri complet diferite. De aceea, în situația noastră, folosind acest algoritm, am obținut un scor mult prea mic, aproximativ 0.5. Astfel, am căzut de acord să păstrăm alegerea de a folosi algoritmul SVM care a demonstrat cea mai bună eficiență.