Inteligență Artificială
Tema 2

Mihai Nan, Ștefania Ghiță, Mihai Trăscău

Deadline: 15 Ianuarie 2023

v1.0.1

Scopul temei este înțelegerea și aprofundarea regresiei prin implementarea mai multor probleme ce au la bază regresia liniară, precum și familiarizarea cu metodele de evaluare ale unor astfel de algoritmi.

1 Problema 1

Atunci când discutăm despre modelul de Regresie Liniară, avem mai multe variante prin care putem realiza estimarea parametrilor acestui model. În cadrul acestei probleme vom explora 4 variante de estimare și vom realiza o comparație între performanțele acestora.

Astfel, pentru această analiză vom porni de la următoarea problemă:

Enunț problemă

Se dă setul de date $(\mathbf{X}, \mathbf{t})$ constând dintr-o mulțime $\mathbf{X}$ de N puncte de dimensiune D (i.e. fiecare $x^{(i)} \in \mathbf{X}$ este de forma $\mathbf{x} = (x_1, x_2, \ldots, x_D)$).
Pentru fiecare $\mathbf{x}^{(n)} \in \mathbf{X}$ există un $t^{(n)}$ reprezentând valoarea unei funcții f (necunoscută) în punctul $\mathbf{x}^{(n)}$, i.e. $t^{(n)} = f(\mathbf{x}^n)$.
Dorim să rezolvăm această problemă folosind un model de regresie liniară simplă care dorește să aproximeze funcția f și definit ca:

$$\mathbf{y} = \mathbf{X}\mathbf{w} + b \tag{1}$$

unde $\mathbf{w}$ este un vector D-dimensional de ponderi, iar b este termenul de bias.

În unele cazuri, funcția f din care provine setul de date $(\mathbf{X}, \mathbf{t})$ nu este una care să poată fi aproximată printr-o simplă combinație liniară a **spațiului de intrare** (i.e. a domeniului din care provine $\mathbf{X}$).

Într-un astfel de caz, spațiul de intrare poate fi *transformat* într-unul mai complex, având o capabilitate de modelare mai bogată.

Acest lucru se face prin intermediul unor *funcții de transformare* ϕ care duc $\mathbf{x^{(n)}} \rightarrow \phi(\mathbf{x^{(n)}})$.

Regresie liniară cu extragere de atribute

Prin introducerea acestei *funcții de transformare*, obținem următorul model de regresie liniară:

$$\mathbf{y} = \phi(\mathbf{X})\mathbf{w} + b \tag{2}$$

Evaluarea modelului

Pentru a verifica predicția modelului obținut, folosim funcția de eroare:

$$\mathcal{L}(y^{(n)}, t^{(n)}) = (y^{(n)} - t^{(n)})^2 \tag{3}$$

În cadrul procesului de antrenare, această funcție de eroare o să fie *mediată* peste toate exemplele din setul de antrenare.

$$\mathcal{E}(w, b) = \frac{1}{N}\sum_{i=1}^{N}(y^{(i)} - t^{(i)})^2 = \frac{1}{N}\sum_{i=1}^{N}(w \cdot \phi\left(x^{(i)}\right) + b - t^{(i)})^2 \tag{4}$$

1.1 Soluția în formă închisă

Determinați formula matematică pentru soluția în formă închisă și implementați un model liniar ce utilizează această formulă pentru estimarea parametrilor.

Această metodă este recomandată atunci când setul de date conține un număr redus de exemple.

Evaluați calitatea soluției obținută pentru această metodă de estimare: realizați un grafic în care să evidențiați care este predicția modelului de regresie liniară, un grafic în care să evidențiați calitatea predicției modelului și determinați valoarea funcției de eroare $\mathcal{E}(w, b)$ (4) pentru seturile de date de antrenare și testare.

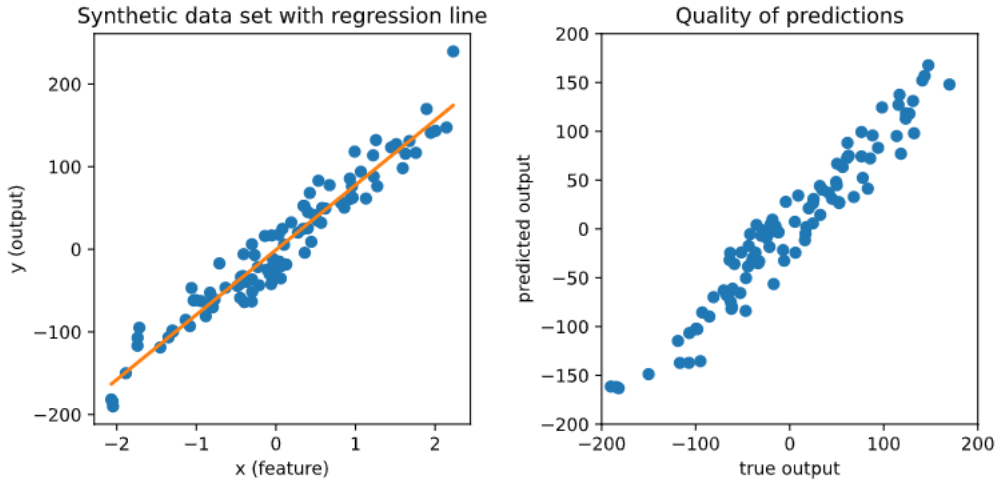


Figure 1: Exemple de grafice

Putem evalua modelul propus și din perspectiva unei alte funcții de eroare. Astfel, propunem utilizarea următoarei funcții pentru evaluarea modelului (Root Mean Squared Error):

$$\mathcal{E}(w,b) = \sqrt{\sum_{i=1}^{N} \frac{1}{N} \left[y_i - \mathbf{w} \cdot \phi\left(x^{(i)}\right) - b\right]^2} \tag{5}$$

Determinați dacă există sau nu vreo legătură între valoarea zgomotului utilizat atunci când sunt generate datele din setul de antrenare și valoarea funcției $\mathcal{E}(w,b)$ (5).

1.2 Gradient Descent

Atunci când utilizăm tehnica Gradient Descent pentru estimarea parametrilor modelului, ponderile sunt actualizate incremental după fiecare epocă.

Vom considera următoarea funcție de cost:

$$\mathcal{J}(w,b) = \frac{1}{2} \sum_{i=1}^{N} (y^{(i)} - t^{(i)})^2$$

Gradientul acestei funcții de eroare ne va indica modalitatea în care trebuie actualizate ponderile modelului:

$$\Delta w_j = \eta \frac{\partial \mathcal{J}}{\partial w_j}$$

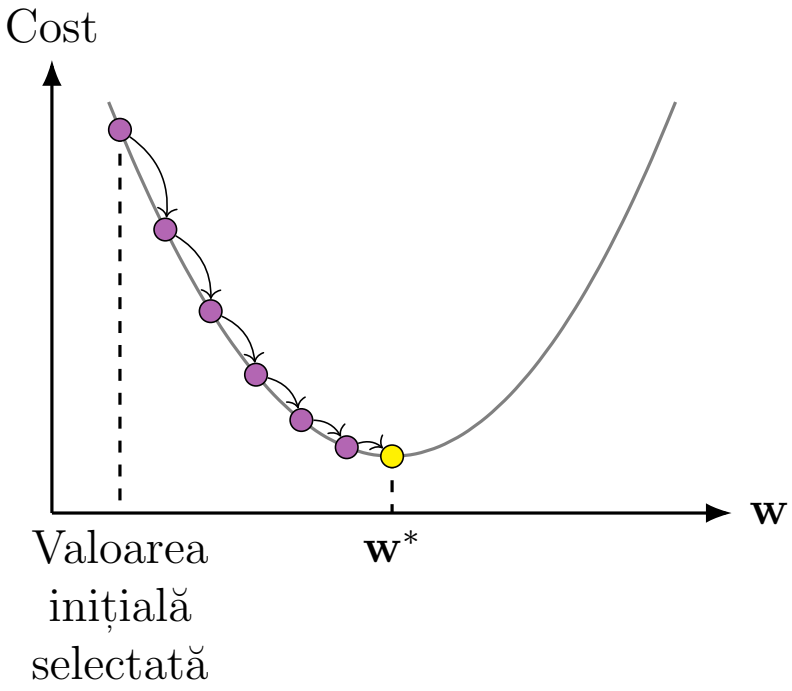
unde η reprezintă rata de învățare.

După fiecare epocă, actualizăm ponderile utilizând următoarea formulă:

$$\mathbf{w} := \mathbf{w} + \Delta \mathbf{w}$$

Algorithm 1: Metoda de estimare Gradient Descent

Data: $(\mathbf{X}, \mathbf{t})$, epochs, η
Result: $\mathbf{w}^*$
$\mathbf{w} \leftarrow$ random_values;
for $k \leftarrow 1 \ldots$ *epochs* **do**
  for $j \leftarrow 1 \ldots M$ **do**
    $w_j \leftarrow w_j - \eta \cdot \sum_{i=1}^{N} \left[h_w\left(x^{(i)}\right) - t^{(i)}\right] \cdot x_j^{(i)}$;
  end
end



Implementați această modalitate de estimare a ponderilor parametrilor modelului de regresie liniară indicat.

Evaluați această metodă pe seturile de date propuse pentru antrenare și testare, evidențiind valoarea funcției $\mathcal{E}(w, b)$ (4) la fiecare epocă.

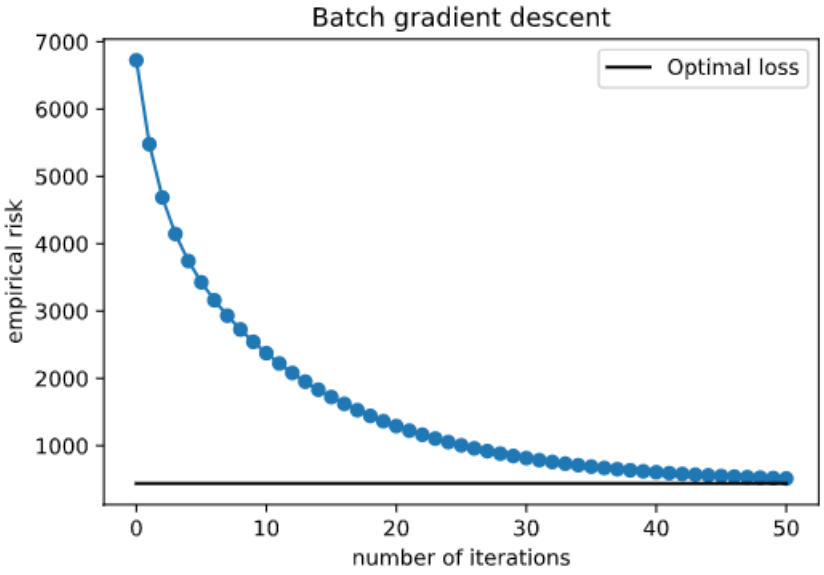


Figure 2: Exemplu orientativ de grafic

De asemenea, trebuie să realizați un grafic în care să evidențiați diferențele care există între calitatea modelului obținut prin folosirea soluției în forma închisă și calitatea modelului estimat de Gradient descent.

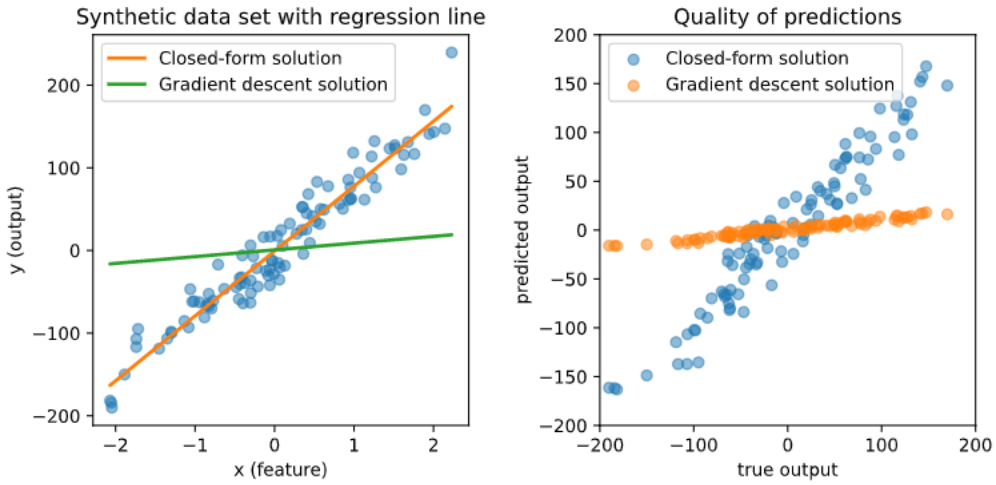


Figure 3: Exemplu orientativ de grafic

Atunci când funcția de eroare aleasă este o funcție de clasă C^2, putem determina rata de învățare utilizând metoda Newton-Raphson:

$$\mathbf{w}^{(t+1)} := \mathbf{w}^{(t)} - \|\nabla^2 J(\boldsymbol{w}^{(t)})\| \tag{6}$$

În cazul nostru, $\|\nabla^2 J(\boldsymbol{w})\| = \|X^\top X\| = \sigma_{\max}$, unde $\sigma_{\max}$ este cea mai mare valoare proprie a matricei $X^\top X$. Astfel, rata de învățare sugerată de această metodă este constantă și egală cu $\frac{1}{\sigma_{\max}}$.

Evaluați metoda **Gradient Descent** utilizând mai multe variante pentru rata de învățare printre care și $\frac{1}{\sigma_{\max}}$. Realizați un grafic pentru a evidenția rezultatele pentru fiecare epocă.

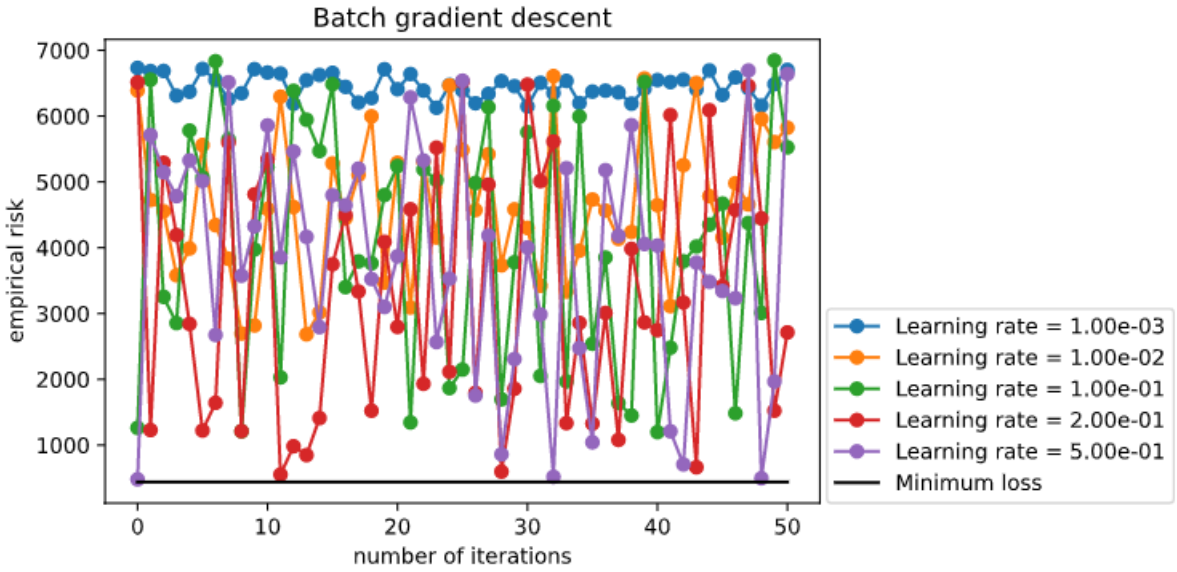


Figure 4: Exemplu de grafic orientativ

1.3 Mini-Batch Gradient Descent

Atunci când lucrăm cu seturi de date cu multe exemple, este foarte greu să calculăm gradientul luând în calcul toate exemplele. Astfel, alegem să folosim doar un minibatch cu M exemple (o submulțime cu M exemple), unde $M \ll N$.

Cea mai simplă variantă de a construi aceste minibatch-uri este să iterăm prin setul de date și să considerăm de fiecare dată slice-uri de dimensiune M. Astfel, formula pentru gradient o să devină:

$$\frac{\partial \mathcal{J}(w,b)}{\partial \mathbf{w}} = \frac{1}{M}\mathbf{\Phi}_{k:k+M}^{\top}\left(\mathbf{\Phi}_{k:k+m}\mathbf{w} - y_{k:k+m}\right) \tag{7}$$

Evaluați această metodă de estimare prin utilizarea unor valori diferite pentru dimensiunea unui minibatch. Realizați un grafic în care să evidențiați ce impact are dimensiunea minibatch-ului asupra funcției de eroare $\mathcal{E}(w,b)$ (4).

1.4 Stochastic Gradient Descent

Metoda de estimare **Stochastic Gradient Descent** este un caz particular al metodei **Mini-Batch Gradient Descent**:

- utilizăm un minibatch de dimensiune 1;
- după fiecare epocă avem grijă să *amestecăm* exemplele din setul de antrenare.

Evaluați această metodă de estimare și realizați un grafic în care să evidențiați valoarea funcției de eroare $\mathcal{E}(w,b)$ (4) după fiecare epocă.

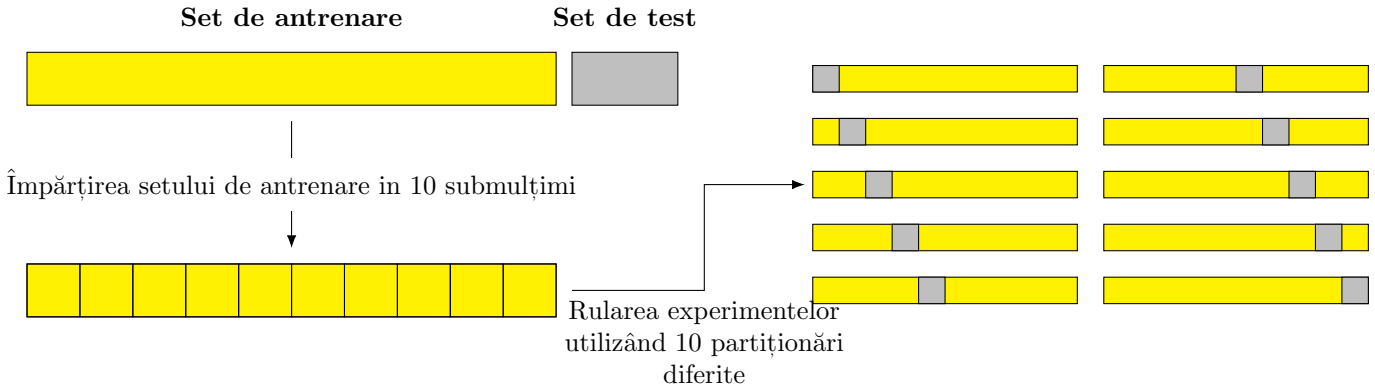
1.5 Analiză comparativă

Realizați o analiză comparativă între aceste metode de optimizare. Alegeți ce funcție doriți pentru extrage caracteristici suplimentare pornind de la exemplele din setul de antrenare (veți avea doar exemple pentru care $x^{(i)} \in \mathbb{R}$). Demonstrați prin grafice cum influențează numărul de caracteristici suplimentare performanțele celor 4 metode de optimizare propuse.

2 Problema 2

Evaluați performanța unui algoritm de regresie liniară cu regularizare L2 de tipul $\alpha w^T w$ aplicat pe setul de date dat. Găsiți cea mai bună valoare a lui α folosind metoda *k-fold validation* astfel încât eroarea să fie minimă.

(a) Pentru a evalua performanța algoritmului veți calcula eroarea medie obținută după 100 de teste. Pentru fiecare test veți aplica metoda validării cu 10 submulțimi (*10-fold validation*): împărțiți setul de date în 10 submulțimi, și alegeți pe rând una dintre ele pe care să o folosiți pentru testare, în timp ce celelalte 9 vor fi folosite pentru antrenare. Repetați procesul până când fiecare submulțime a fost folosită ca set de testare pentru algoritm.



Amestecați setul de date (shuffle) înainte de fiecare test astfel încât submulțimile să fie diferite între teste. Nu amestecați datele în timpul evaluării cu metoda *10-fold validation*, pentru ca submulțimile de testare să nu se suprapună.

La finalul metodei de evaluare veți avea 1000 de valori obținute pe baza cărora veți calcula eroarea medie a algoritmului.

Nu folosiți funcțiile deja definite în diferitele biblioteci pentru metoda *k-fold validation*. Unul dintre obiectivele temei este de a înțelege cum funcționează această metodă de evaluare, așa că va trebui să o implementați de la zero.

Generați un grafic care să reprezinte valorile erorii obținute cu metoda *10-fold validation* pe parcursul celor 100 de teste.

(b) Pentru a găsi cea mai bună valoare a hiperparametrului α veți analiza erorile obținute de algoritm folosind metoda *10-fold validation*. Definiți un interval de căutare pentru valorile lui α (valoarea minimă, valoarea maximă și pasul de incrementare) și aplicați metoda *10-fold validation* pentru fiecare valoare aleasă. Alegeți valoarea din interval care a obținut eroarea minimă în urma evaluării cu *10-fold validation*. Rulați de cel puțin 10 ori această metodă și alegeți valoarea pentru hiperparametrul α care a fost aleasă de cele mai multe ori.

Justificați alegerea parametrilor pentru intervalul hiperparametrului α. Parametrii aleși vor trebui să genereze cel puțin 20 de valori diferite pe care să le testați.

Folosiți metoda *k-fold validation* implementată la punctul a.

Generați un grafic care să reprezinte eroarea medie obținută pentru fiecare valoare din interval a hiperparametrului α cu metoda *10-fold validation* pe parcursul celor (cel puțin) 10 rulări.

Generați un grafic care să reprezinte valoarea aleasă din interval la fiecare rulare pentru hiperparametrului α.

Punctajul obținut pentru această problemă va fi împărțit între: implementarea corectă a regresiei liniare cu regularizare L2 pentru setul de date dat, implementarea corectă a metodei de evaluare *k-fold validation*, evaluarea regresiei prin aplicarea repetată a metodei de evaluare *k-fold validation*, alegerea valorii optime pentru hiperparametrul α pe baza rezultatelor

obținute cu metoda de evaluare *k-fold validation*, justificarea intervalului de căutare ales pentru hiperparametrul α, și cele 3 grafice de reprezentare a evoluției metodelor de evaluare.

3 Problema 3

Arborii de regresie reprezintă un tip special de arbori de decizie care realizează predicții pentru exemplele date la intrare unde valoarea țintă generată este continuă. Astfel, arborii de regresie sunt utilizați pentru a partiționa spațiul de decizie (la fel ca în cazul arborilor de decizie) cu nodurile intermediare reprezentând decizii făcute la nivel de variabilă (fie ea continuă sau discretă) iar în frunze, pentru variabila țintă se va stoca o valoare continuă (față de o clasă în cazul arborilor de decizie).

3.1 Funcția de eroare pentru regresie

Dacă la arborii de decizie construcția se baza pe optimizarea câștigului informațional (eng. Information Gain) în arborii de regresie ne bazăm pe suma reziduală a pătratelor (RSS - eng. Residual Sum of Squares). Astfel, pentru o variabilă continuă dată, X_j, se va căuta valoarea de tăietură c pe baza valorii RSS. În funcție de valoarea de tăietură obținem, pentru nodul respectiv, două subseturi disjuncte de observații: $X_{X_j \leq c}$ pentru exemplele cu valori ale atributului X_j mai mici decât c și $X_{X_j > c}$ pentru exemplele cu valori ale atributului X_j mai mari decât c. Pentru fiecare dintre cele două subseturi putem calcula valoarea medie, $\bar{y}_{X_j \leq c}$ respectiv $\bar{y}_{X_j > c}$. Prin urmare, putem calcula valoarea RSS pentru o variabilă X_j și o valoare de tăietură pentru aceasta:

$$RSS = \sum_{k \in X_{X_j \leq c}} (y_k - \bar{y}_{X_j \leq c})^2 + \sum_{k \in X_{X_j \leq c}} (y_k - \bar{y}_{X_j > c})^2 \tag{8}$$

Selecția la nivel de nod intermediar se face prin minimizarea valorii RSS:

$$\min_{X_j, c} RSS(X_j, c) \tag{9}$$

3.2 Construcția arborilor de regresie

Astfel, setul de date cu care lucrăm constă în N intrări (exemple) exprimate prin p variabile, adică $x^{(i)} = (x_1^{(i)}, x_2^{(i)}, ..., x_p^{(i)})$, pentru care avem definite valori țintă $y^{(i)}$ formând perechile $(x^{(i)}, y^{(i)})$ cu $i = 1, 2, ..., N$. Pentru construcția arborilor de regresie folosim procedeul de *partiționare binară recursivă*, prin care selectăm într-un nod (intermediar) o variabilă și o valoare de tăietură pentru aceasta prin care împărțim setul de date în 2 subseturi. Criteriul de selecția este dat de minimizarea valorii RSS pentru perechea variabilă - valoare de tăietură. Cu toate acestea, găsirea celei mai bune partiționări binare folosind RSS nu este nu este tractabil computațional. Prin urmare, vom utiliza tehnici *greedy* de generare. Plecând de la setul de date complet, considerăm pe rând toate variabilele x_j după care putem împărți setul de date. Variabilele pot fi de tip continuu sau de discret (categorice). Pentru variabilele continue, valorile punctelor de tăietură se aleg în funcție de valorile unice regăsite în setul de

date pentru variabila respectivă ca media a două valori consecutive. De exemplu, dacă pentru o variabilă continuă x_1 găsim în setul de date că avem valorile unice (sortate ascendent) $\{10, 12, 13, 44, 50\}$ vom avea ca posibile puncte de tăietură mulțimea $\{11, 12.5, 28.5, 47\}$. Pentru variabilele discrete nu există puncte de tăietură, generându-se un descendent pentru fiecare valoare posibilă din domeniul variabilei. Astfel, se va construi arborele de regresie complet, T prin generarea de noduri intermediare care minimizează RSS. Frunzele sunt create pe baza strategiei de oprire, dată de regulile:

- Eroarea pătratică calculată pentru nodul curent pentru este 0
- Numărul de exemple pentru nodul curent ≤ 5

În final, pentru o frunză veți returna valoarea medie a variabilei țintă pentru exemplele din frunza respectivă.

3.3 Reducerea arborilor prin tăiere (pruning)

Utilizând o astfel de procedură pentru selecția nodurilor intermediare, chiar și cu o strategie prin care forțăm generarea unui nod frunză care să conțină minim k exemple (e.g. $k = 5$), poate duce la construcția unui arbore de regresie de dimensiuni foarte mari. Oprirea timpurie a construcției, pe baza adâncimii, a numărului total de noduri sau alte strategii similare (pe care le utilizăm frecvent în cazul arborilor de decizie) nu sunt foarte utile în arborii de regresie. În acest caz vom utiliza tehnica tăierii de subarbori pentru a reduce dimensiunea arborelui de regresie complet la una acceptabilă atât din punct de vedere cost (RSS) cât și complexitate (număr de noduri). Definim astfel o funcție de *cost-complexitate*, C_α, pentru un arbore de regresie prin Ecuația 10.

$$C_\alpha(T) = RSS(T) + \alpha\,|T| \tag{10}$$

unde $RSS(T)$ este valoarea RSS calculată la nivelul întregului arbore (media pentru toate frunzele), $|T|$ este *numărul de noduri frunză* din arborele T iar α este coeficientul de penalizare pentru complexitate. Plecând de la arborele de regresie complet, T^1, aplicăm Algoritmul 2 pentru a obține o serie de tăieturi optime în funcție de parametrul α.

Figura 5 prezintă modalitatea prin care eliminăm un subarbore. Practic, eliminarea lui T_t din T înseamnă reducerea acestuia la nodul t și înlocuirea acestuia în arborele T.

Din cele două liste rezultate în urma rulării Algoritmului 2 va trebui să alegeți valoarea α prin procedura de *cross-validation* pe setul de date de antrenare. Adică, pentru fiecare arbore T_i generat de algoritm, veți calcula $RSS(T_i)$ prin *cross-validation* și veți selecta acel arbore pentru care valoarea este minimă.

3.4 Seturi de date de evaluare

Veți analiza performanțele arborelui de regresie din punct de vedere al erorii pătratice medii (eng.- Root Mean Squared Error) pe seturile de date indicate, ca în Ecuația 11, unde y_i este valoarea țintă a unui exemplu din setul de test (i.e. ground truth) iar $\hat{y}_i$ este predicția arborelui de regresie.

$$RMSE = \sqrt{\frac{\sum_{i=1}^{n}(y_i - \hat{y}_i)}{n}} \tag{11}$$

Algorithm 2: Tăierea cost-complexitate pentru arbori de regresie (Sursă)

Data: $\mathbf{T^1}$, arborele complet obținut cu $\alpha^{\mathbf{1}} = 0$

Result: O secvență de arbori tăiați $T^1 \supseteq T^2 \supseteq \dots T^k \supseteq \dots \supseteq \{root\}$ și o secvență de parametri $\alpha^{(1)} \leq \alpha^{(2)} \leq \dots \leq \alpha^{(k)} \leq \dots$

Pasul 1:

Selectează nodul $t \in T^1$ care minimizează $g_1(t) = \frac{RSS(t) - RSS(T_t^1)}{|T_t^1| - 1}$

Fie t_1 nodul selectat

Fie $\alpha^{(2)} = g_1(t_1)$

Fie $T^2 = T^1 - T_{t_1}^1$

Pasul *i*:

Selectează nodul $t \in T^i$ care minimizează $g_i(t) = \frac{RSS(t) - RSS(T_t^i)}{|T_t^i| - 1}$

Fie t_i nodul selectat

Fie $\alpha^{(i+1)} = g_i(t_i)$

Fie $T^{i+1} = T^i - T_{t_i}^i$

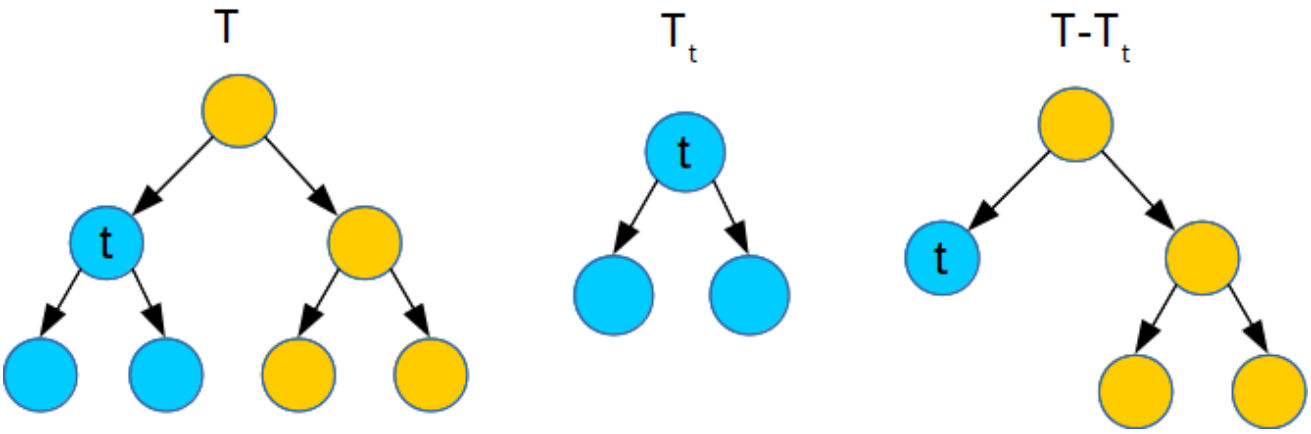


Figure 5: Tăierea arborelui T prin eliminarea subarborelui T_t (Sursă)

Seturile de date, selectate din colecția de seturi de date UCI, sunt Auto MPG Data Set care conține doar atât variabile discrete cât și continue, și Wine Quality care conține doar variabile continue. Pentru primul set de date veți folosi acest fișier iar pentru al doilea set de date veți utiliza acest fișier. Generați grafice care să descrie valoarea $RMSE$ pentru fiecare dintre arborii returnați de Algoritmul 2.