ML Dataset Evaluation

Abstract

Proiectul a vizat clasificarea calității vinului utilizând algoritmul Ridge Classifier aplicat asupra caracteristicilor fizico-chimice. Seturile de date au inclus 4.899 de probe de vin alb și 1.600 de vin roșu, fiecare cu variabile precum aciditate fixă și volatilă, acid citric, zahăr rezidual, cloruri, dioxid de sulf, densitate, pH, sulfați și alcool. Calitatea a fost etichetată în trei categorii: **low** (3–4), **medium** (5–6) și **high** (7–9). Datele au fost împărțite 80% pentru antrenare și 20% pentru testare. Ridge Classifier a fost ales pentru capacitatea de regularizare și gestionare a corelațiilor dintre variabile. Modelul final a demonstrat că proprietățile fizico-chimice pot prezice eficient nivelul calității vinului.

RidgeClassifier – White Wine Dataset

Codul încarcă datele despre calitatea vinului din fișiere CSV de antrenare și testare, apoi grupează scorurile calității (quality) în trei categorii: **low** (3–4), **medium** (5–6) și **high** (7–8-9). După afișarea distribuției acestor grupuri în seturile de date, se separă variabilele independente (X) de etichetele de clasificare (y). Datele sunt standardizate cu StandardScaler pentru a avea medie 0 și deviație standard 1, astfel încât toate caracteristicile să fie pe aceeași scară numerică. Apoi, se antrenează un model de clasificare **RidgeClassifier** cu **regularizare L2** (alpha=1.0), care se potrivește pe datele de antrenament scalate și apoi prezice categoriile de calitate pentru datele de test. La final, codul calculează și afișează acuratețea globală a modelului pe setul de test și un raport de clasificare detaliat care include precizia, recall-ul și scorul F1 pentru fiecare categorie ("low", "medium", "high").

Mai sus a fost prezentat modelul de bază. În alte fișiere ipynb, au fost încercate alte metode a îmbunătăți modelul, precum: optimizarea hiperparametrilor, de a mări complexitatea modelului, deoarece modelul nu putea prezice clasa low(quality 3-4), dar și implementarea **ADASYN**, o metodă de a genera date în clasele „rare”.

RidgeClassifier: Acest clasificator convertește mai întâi valorile țintă în {-1, 1}, apoi tratează problema ca o sarcină de regresie (regresie multi-output în cazul multi-clasă).

alpha *(float, implicit=1.0)* puterea regularizării; trebuie să fie un număr pozitiv. Regularizarea îmbunătățește condiționarea problemei și reduce varianța estimărilor. Valorile mai mari indică o regularizare mai puternică.

class_weight*(dict sau 'balanced',implicit=None)*
Ponderile asociate claselor, sub forma {eticheta_clasei: pondere}. Dacă nu se specifică, toate clasele sunt considerate cu pondere egală (1).

Modul **„balanced"** folosește valorile lui *y* pentru a ajusta automat ponderile invers proporțional cu frecvența claselor în datele de intrare, conform: *n_samples / (n_classes * np.bincount(y))*

ADASYN (Suprasampling utilizând algoritmul Adaptive Synthetic): Această metodă este similară cu SMOTE, dar generează un număr diferit de exemple sintetice în funcție de o estimare a distribuției locale a clasei care trebuie suprasamplată.

n_neighbors *(int sau obiect estimator, implicit=5)*

Numărul de vecini apropiați folosiți pentru a defini vecinătatea eșantioanelor utilizate în generarea noilor exemple sintetice.

sampling_strategy *(float, str, dict sau callable, implicit='auto')*
Informații despre modul de resampling al setului de date.

- 'minority': resamplează doar clasa minoritară;
- 'not minority': resamplează toate clasele cu excepția clasei minoritare;
- 'not majority': resamplează toate clasele cu excepția clasei majoritare;
- 'all': resamplează toate clasele;

- 'auto': echivalent cu 'not majority'.

```
Distribuție train:
quality_group
medium    2932
high       833
low        153
Name: count, dtype: int64

Distribuție test:
quality_group
medium    723
high      227
low        30
Name: count, dtype: int64
```

Figura 1. Distribuția datelor în seturile de train și test

```
Acuratețea modelului RidgeClassifier pe grupuri: 76.22%

Classification Report:
              precision    recall  f1-score   support

        high       0.67      0.20      0.31       227
         low       0.00      0.00      0.00        30
      medium       0.77      0.97      0.86       723

    accuracy                           0.76       980
   macro avg       0.48      0.39      0.39       980
weighted avg       0.72      0.76      0.70       980
```

Figura 2. Rezultatele modelului de bază

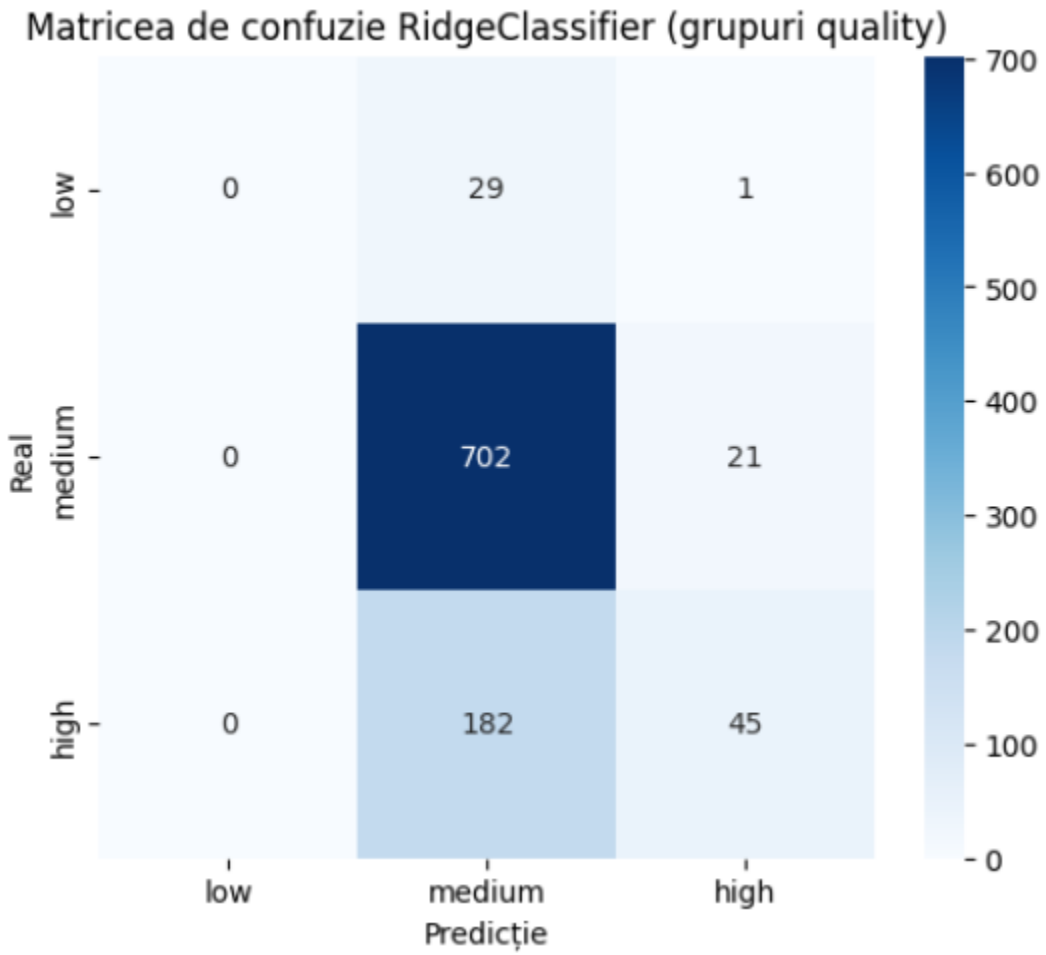

Figura 3. Matricea de confuzie

În matricea de confuzie de mai sus, pe linii sunt reprezentate clasele reale, iar pe coloane sunt predicțiile modelului. Pentru clasa **low**, 29 de exemple au fost prezise ca medium și 1 ca high, rezultând că modelul nu a reușit mai deloc să identifice clasa low. Pentru clasa **medium**, 702 de exemple au fost prezise corect și 21 de exemple greșit, prezise că aparțin clasei high, ceea ce arată că această clasă este cel mai bine identificată. Pentru clasa **high**, 182 de exemple au fost prezise greșit în clasa medium, dar 45 de exemple au fost prezise corect. Cel mai probabil, distibuția dezechilibrată a claselor influențează performanța.

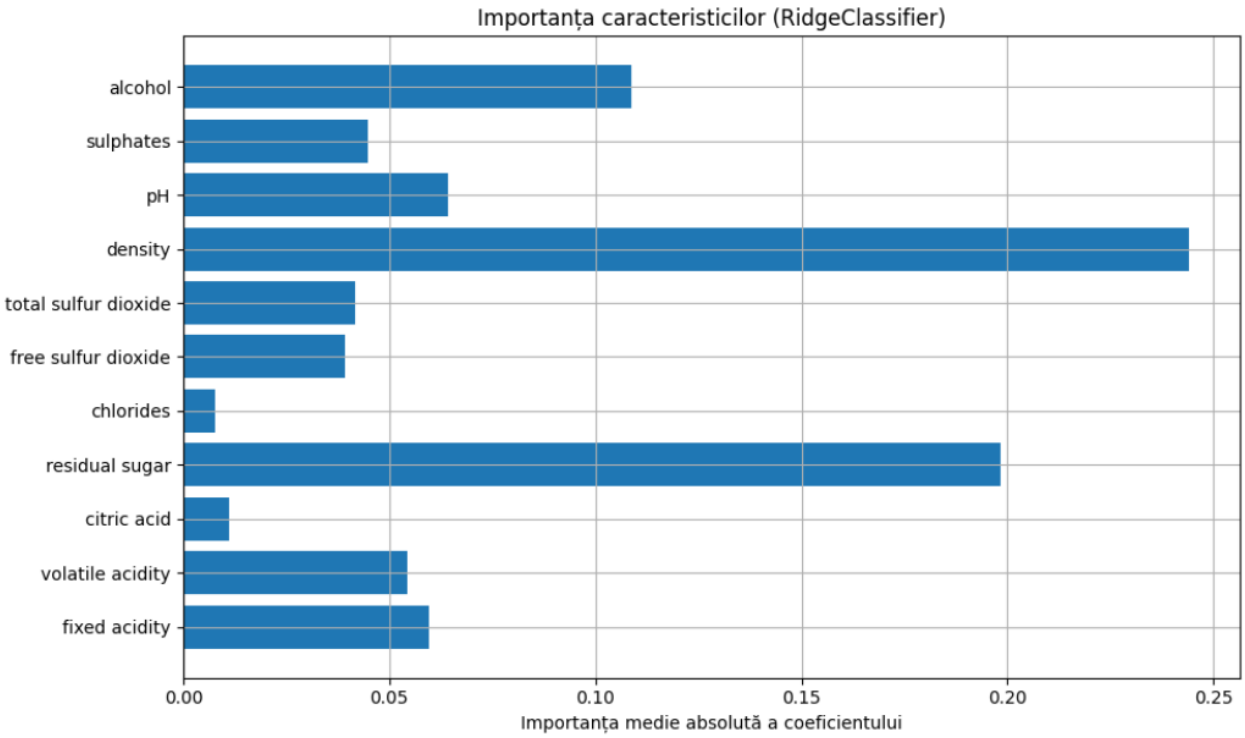

Figura 4. Importanța caracteristicilor

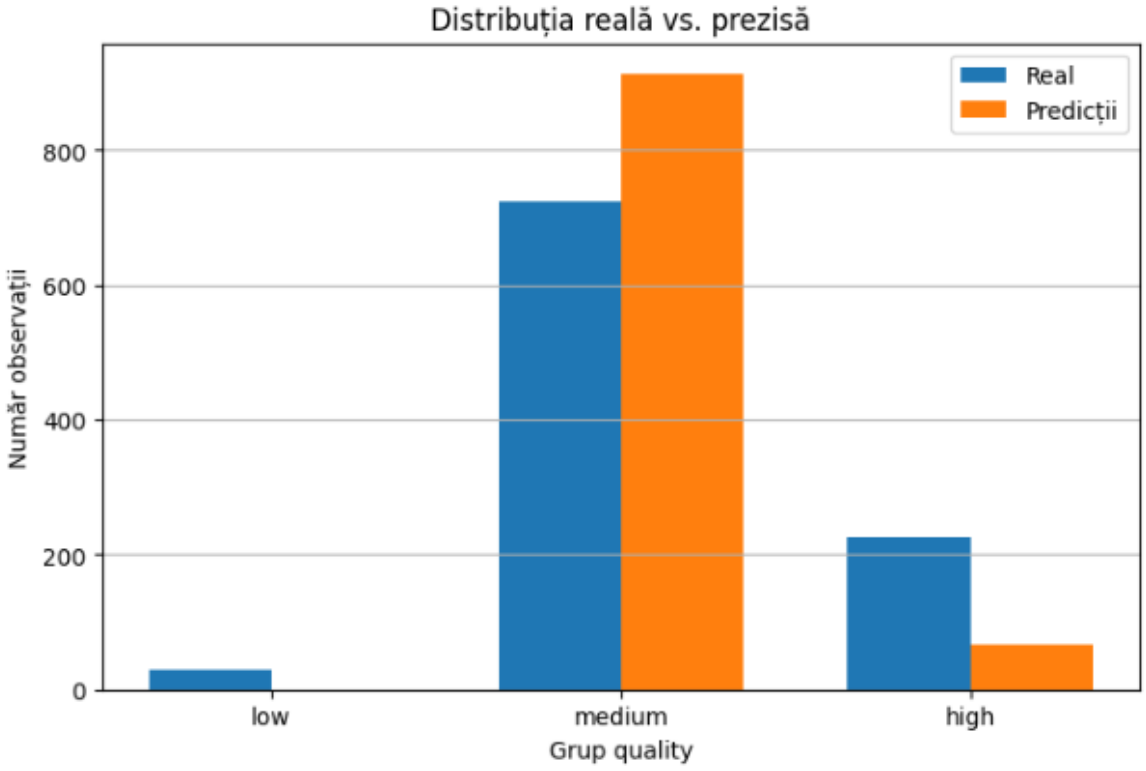

Figura 5. Distribuția modelului

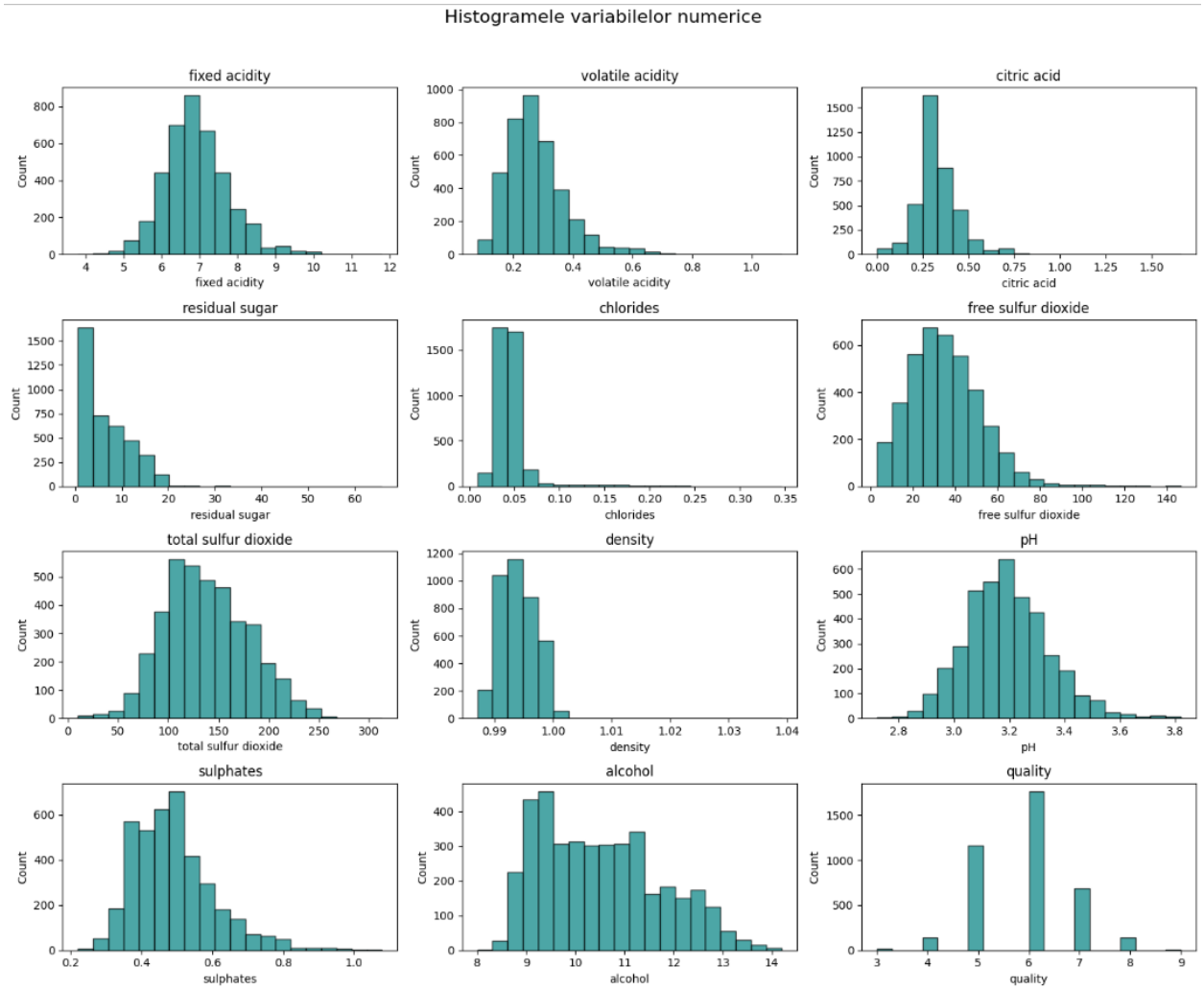

Figura 6. Histogramele caracteristicilor numerice și etichetei "quality"

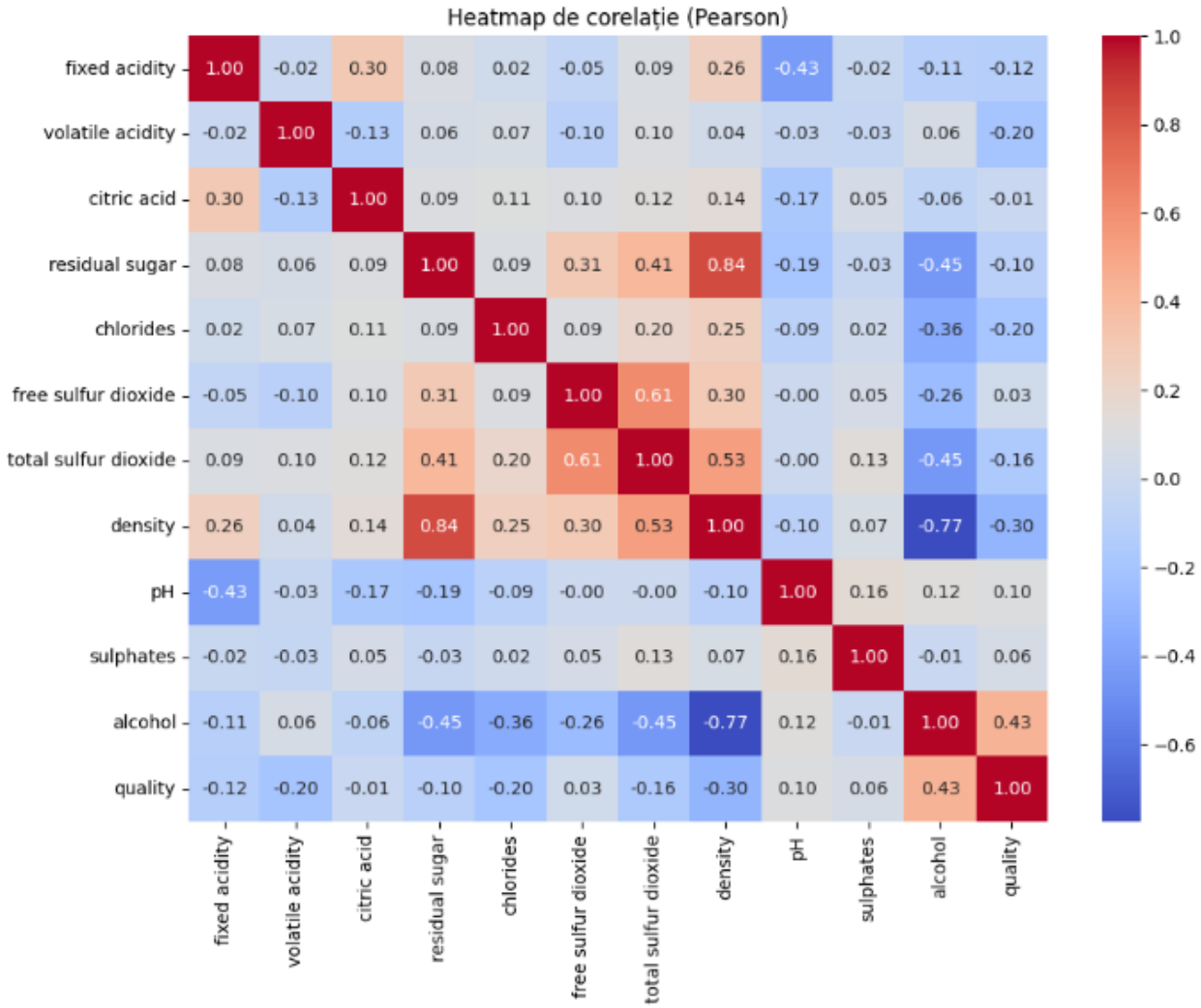

Figura 7. Corelațiile Pearson

Corelațiile Pearson între fiecare variabilă numerică și quality:

alcohol 0.431681
pH 0.103161
sulphates 0.062008
free sulfur dioxide 0.028810
citric acid -0.014022
residual sugar -0.096854
fixed acidity -0.117994
total sulfur dioxide -0.162517
chlorides -0.199168
volatile acidity -0.204256
density -0.300557

Alcohol (+0.43): Cea mai puternică corelație pozitivă. Cu cât alcoolul e mai mare, cu atât scorul de calitate e mai mare și nu e surprinzător: vinurile cu tărie mai mare tind să fie considerate mai bune.

pH (+0.10): O corelație pozitivă slabă, dar care sugerează că un pH mai mare poate fi asociat cu o calitate mai bună.

Sulphates (+0.06): Corelație foarte slab pozitivă. Sulfații contribuie la stabilitate și conservare, dar efectul e mic.

Density (–0.30): Corelație negativă. Vinurile mai dense tind să fie de calitate mai slabă, probabil pentru că densitatea mare sugerează zahăr rezidual excesiv sau fermentare incompletă.

Volatile acidity (–0.20): Corelație negativă. Aciditatea volatilă ridicată scade calitatea vinului, reflectând un gust acru, neplăcut.

Chlorides (–0.19): Corelație negativă. Sarea(cloruri) ridicată în vin afectează gustul.

Fixed acidity (–0.11): Corelație negativă slabă. Aciditatea fixă mai mare poate indica un gust mai acru, ceea ce nu e ideal pentru calitate.

Restul caracteristicilor (volatile acidity, citric acid, free sulfur dioxide, total sulfur dioxide, residual sugar) au corelații slabe sau foarte slabe cu quality, nu influentează semnificativ calitatea vinului.

```
model = RidgeClassifier(alpha=5.0, class_weight='balanced')
Acuratețea modelului RidgeClassifier pe grupuri: 50.82%

Classification Report:
              precision    recall  f1-score   support

        high       0.42      0.76      0.54       227
         low       0.08      0.57      0.14        30
      medium       0.87      0.43      0.57       723

    accuracy                           0.51       980
   macro avg       0.46      0.58      0.42       980
weighted avg       0.74      0.51      0.55       980



model = RidgeClassifier(alpha=10.0)
Acuratețea modelului RidgeClassifier pe grupuri: 75.92%

Classification Report:
              precision    recall  f1-score   support

        high       0.65      0.19      0.29       227
         low       0.00      0.00      0.00        30
      medium       0.77      0.97      0.86       723

    accuracy                           0.76       980
   macro avg       0.47      0.39      0.38       980
weighted avg       0.72      0.76      0.70       980


model = RidgeClassifier(alpha=5.0, fit_intercept=True, solver='sag', random_state=42)
Acuratețea modelului RidgeClassifier pe grupuri: 76.22%

Classification Report:
              precision    recall  f1-score   support

        high       0.67      0.20      0.31       227
         low       0.00      0.00      0.00        30
      medium       0.77      0.97      0.86       723

    accuracy                           0.76       980
   macro avg       0.48      0.39      0.39       980
weighted avg       0.72      0.76      0.70       980
```

Figura 8. Optimizarea hiperparametrilor

```
model = RidgeClassifier(alpha=10.0, fit_intercept=True, solver='auto', random_state=42)
adasyn = ADASYN(random_state=42)
Distribuție după ADASYN:
Counter({'low': 2950, 'medium': 2932, 'high': 2845})

Acuratețea modelului RidgeClassifier cu ADASYN: 50.41%

Classification Report:
              precision    recall  f1-score   support

        high       0.42      0.78      0.55       227
         low       0.09      0.63      0.15        30
      medium       0.87      0.41      0.56       723
...
    accuracy                           0.50       980
   macro avg       0.46      0.61      0.42       980
weighted avg       0.74      0.50      0.54       980

model = RidgeClassifier(alpha=10.0, fit_intercept=True, solver='auto', random_state=42)
adasyn = ADASYN(random_state=42, sampling_strategy="minority")
Distribuție după ADASYN:
Counter({'low': 2950, 'medium': 2932, 'high': 833})

Acuratețea modelului RidgeClassifier cu ADASYN: 57.24%

Classification Report:
              precision    recall  f1-score   support

        high       0.77      0.09      0.16       227
         low       0.08      0.67      0.15        30
      medium       0.73      0.72      0.72       723
...
    accuracy                           0.57       980
   macro avg       0.53      0.49      0.34       980
weighted avg       0.72      0.57      0.58       980

model = RidgeClassifier(alpha=10.0, fit_intercept=True, solver='auto', random_state=42)
adasyn = ADASYN(random_state=42, sampling_strategy="all")
Distribuție după ADASYN:
Counter({'low': 2950, 'medium': 2932, 'high': 2845})

Acuratețea modelului RidgeClassifier cu ADASYN: 50.41%

Classification Report:
              precision    recall  f1-score   support

        high       0.42      0.78      0.55       227
         low       0.09      0.63      0.15        30
      medium       0.87      0.41      0.56       723
...
    accuracy                           0.50       980
   macro avg       0.46      0.61      0.42       980
weighted avg       0.74      0.50      0.54       980
```

Figura 9. Model RidgeClassifier – ADASYN

RidgeClassifier – Red Wine Dataset

Codul încarcă datele despre calitatea vinului roșu din fișierul CSV, verifică și elimină rândurile duplicate, apoi grupează scorurile de calitate în trei categorii („low”, „mediu” și „high”) folosind o funcție de mapare. Variabilele independente (caracteristicile chimice ale vinului) sunt standardizate cu StandardScaler, iar datele sunt împărțite în seturi de antrenament și testare (80/20) cu stratificare pentru a păstra distribuția claselor. Codul afișează distribuția scorurilor originale și a categoriilor grupate prin contorizare și un grafic cu bare. Se antrenează un model de clasificare RidgeClassifier, se fac predicții asupra setului de test, iar performanța modelului este evaluată prin acuratețe și un raport de clasificare detaliat.

```
quality
5    577
6    535
7    167
4     53
8     17
3     10
Name: count, dtype: int64
Distribuție scoruri grupate:
 quality_grouped
high      184
low        63
mediu    1112
```

Figura 10. Distribuția datelor

```
Acuratețe: 0.8161764705882353
Raport de clasificare:
              precision    recall  f1-score   support

        high       0.50      0.05      0.10        37
         low       0.00      0.00      0.00        13
       mediu       0.82      0.99      0.90       222

    accuracy                           0.82       272
   macro avg       0.44      0.35      0.33       272
weighted avg       0.74      0.82      0.75       272
```

Figura 11. Rezultatele modelului

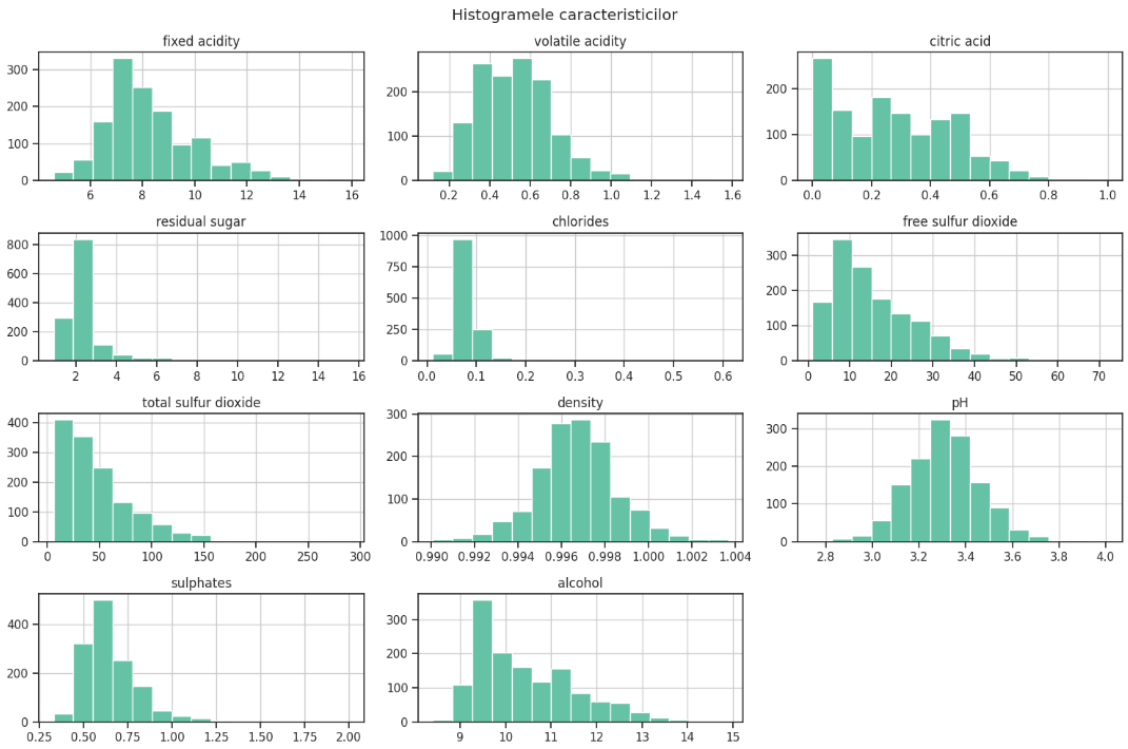

Figura 12. Histogramele caracteristicilor

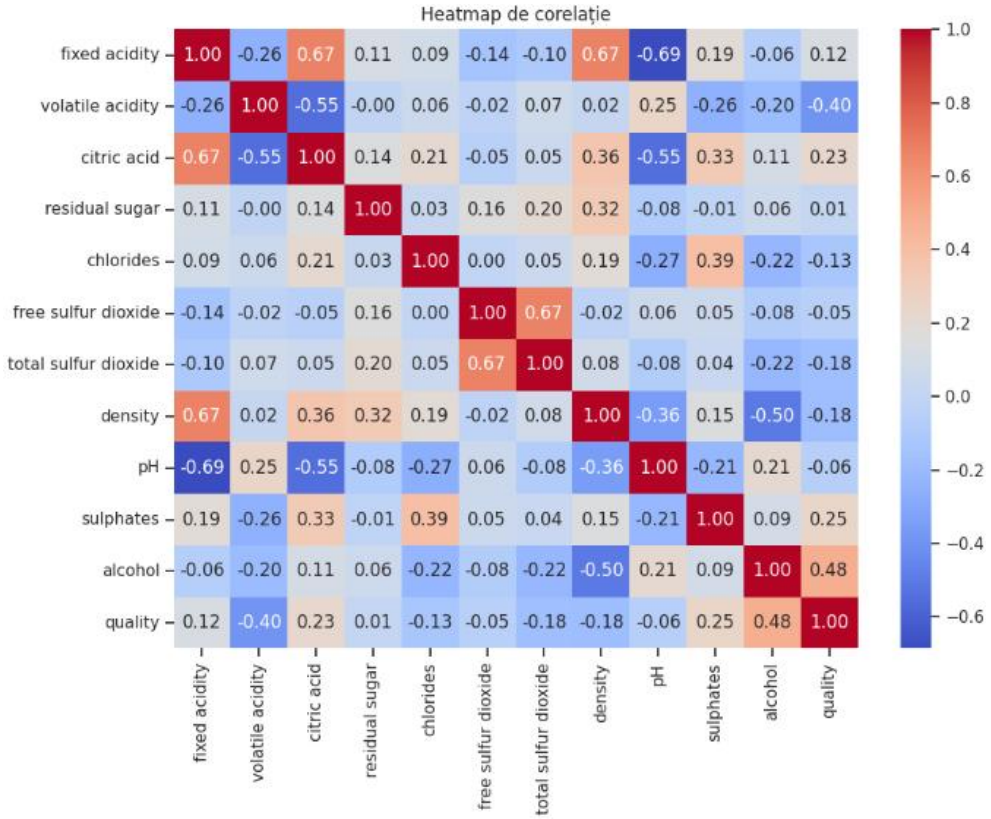

Figura 13. Corelațiile Pearson

Corelațiile Pearson între fiecare variabilă numerică și quality:

alcohol	0.480343
sulphates	0.248835
citric acid	0.228057
fixed acidity	0.119024
residual sugar	0.013640
free sulfur dioxide	-0.050463
pH	-0.055245
chlorides	-0.130988
total sulfur dioxide	-0.177855
density	-0.184252
volatile acidity	-0.395214

Corelații pozitive (când crește variabila, crește și calitatea):
alcohol (0.48) – Vinurile cu mai mult alcool sunt, în general, considerate de calitate mai bună.

sulphates (0.25) – Sulfații contribuie la stabilitatea vinului și sunt asociați cu scoruri mai bune.

citric acid (0.23) – O ușoară aciditate adaugă prospețime și e apreciată în vinurile bune.

fixed acidity (0.12) – Are o influență slabă, dar pozitivă asupra gustului.

residual sugar (0.01) – Nu pare sa aibă o legatură clară cu scorul calității.

Corelații negative (când crește variabila, scade calitatea):
free sulfur dioxide (-0.05) – Efect foarte mic, dar în cantitate prea mare poate afecta gustul.

pH (-0.06) – Un pH mai scăzut poate fi mai bun, dar influența generală este foarte redusă.

chlorides (-0.13) – Clorurile pot da un gust sărat și apar mai des in vinurile mai slabe.

total sulfur dioxide (-0.18) – Nivelurile ridicate pot strica aroma și sunt asociate cu scoruri mai mici.

density (-0.18) – Vinurile mai dense (de obicei mai dulci) tind sa fie cotate mai slab.

volatile acidity (-0.40) – Are cea mai puternică legatură negativă, aciditatea volatilă mare e percepută ca defect (gust înțepător, miros de oțet).

GridSearchCV: Căutare asupra valorilor specificate ale parametrilor pentru un estimator.

Membrii importanți sunt metodele **fit** și **predict**.

GridSearchCV implementează metodele **fit** și **score**. De asemenea, implementează metodele **score_samples**, **predict**, **predict_proba**, **decision_function**, **transform** și **inverse_transform**, dacă acestea sunt implementate în estimatorul utilizat.

Parametrii estimatorului, folosiți pentru aplicarea acestor metode, sunt optimizați printr-o căutare în grilă cu validare încrucișată (cross-validated grid-search) asupra unei grile de parametri.

Pentru fiecare model (RidgeClassifier, RandomForestClassifier și GradientBoostingClassifier) se definește o grilă de hiperparametri și se utilizează GridSearchCV cu validare încrucișată pe 5 fold-uri pentru a găsi combinația de parametri care maximizează acuratețea. Modelul optim găsit este evaluat pe setul de testare, iar rezultatele includ cea mai bună configurare a hiperparametrilor, acuratețea finală și raportul de clasificare (precizie, recall și F1-score pentru fiecare clasă). Astfel, scriptul compară direct cele trei algoritmi și selectează configurațiile lor cele mai performante.

```
=== RidgeClassifier ===
Cea mai buna valoare alpha: 10
Acuratete pe setul de test: 0.8313
              precision    recall  f1-score   support

         bun       0.60      0.14      0.23        43
       mediu       0.84      0.98      0.91       264
        slab       0.00      0.00      0.00        13

    accuracy                           0.83       320
   macro avg       0.48      0.37      0.38       320
weighted avg       0.77      0.83      0.78       320
```

Figura 14. Rezultate RidgeClassifier – GridSearchCV

```
=== RandomForestClassifier ===
Cei mai buni parametri: {'max_depth': 20, 'min_samples_split': 2, 'n_estimators': 200}
Acuratete pe setul de test: 0.8688
              precision    recall  f1-score   support

         bun       0.73      0.56      0.63        43
       mediu       0.89      0.96      0.92       264
        slab       0.00      0.00      0.00        13

    accuracy                           0.87       320
   macro avg       0.54      0.51      0.52       320
weighted avg       0.83      0.87      0.85       320
```

Figura 15. Rezultate RandomForestClassifier – GridSearchCV

Un **Random Forest** este un meta-estimator care antrenează un număr de clasificatori de tip **arbore de decizie** pe diferite sub-eșantioane ale setului de date și utilizează medierea (averaging) pentru a îmbunătăți acuratețea predicțiilor și pentru a controla supraînvățarea (*over-fitting*).

Arborii din pădure folosesc strategia de împărțire optimă, adică echivalentă cu transmiterea opțiunii splitter="best" către clasificatorul de tip arbore de decizie subordonat (*DecisionTreeClassifier*). Dimensiunea sub-eșantionului este controlată prin parametrul max_samples dacă bootstrap=True (implicit), altfel fiecare arbore este construit utilizând întregul set de date.

max_depth*(int,implicit=None)*
Adâncimea maximă a arborelui. Dacă este None, nodurile sunt extinse până când toate frunzele devin pure sau până când toate frunzele conțin mai puține eșantioane decât min_samples_split.

min_samples_split *(int sau float, implicit=2)*
Numărul minim de eșantioane necesar pentru a împărți un nod intern.

n_estimators *(int, implicit=100)*
Numărul de arbori din pădure.

```
=== GradientBoostingClassifier ===
Cei mai buni parametri: {'learning_rate': 0.05, 'max_depth': 5, 'n_estimators': 200}
Acuratete pe setul de test: 0.8562
              precision    recall  f1-score   support

         bun       0.66      0.63      0.64        43
       mediu       0.89      0.94      0.91       264
        slab       0.00      0.00      0.00        13

    accuracy                           0.86       320
   macro avg       0.52      0.52      0.52       320
weighted avg       0.83      0.86      0.84       320
```

Figura 16. Rezultate GradientBoostingClassifier – GridSearchCV

GradientBoostingClassifier: Acest algoritm construiește un model aditiv într-o manieră secvențială (*forward stage-wise*); permite optimizarea oricărei funcții de pierdere derivabile. La fiecare etapă, se antrenează câte un arbore de regresie pentru fiecare clasă (*n_classes_* arbori), folosind gradientul negativ al funcției de pierdere (de exemplu, *log loss* binar sau multiclasă).

learning_rate *(float, implicit=0.1)*
Rata de învățare reduce contribuția fiecărui arbore prin înmulțirea cu *learning_rate*. Există un compromis între *learning_rate* și *n_estimators*: o rată mai mică de învățare necesită un număr mai mare de estimatori pentru performanțe bune.

n_estimators *(int, implicit=100)*
Numărul de etape de *boosting* care vor fi efectuate. *Gradient boosting* este destul de robust la supraînvățare, astfel că un număr mare de estimatori duce de obicei la performanțe mai bune.

max_depth *(int sau None, implicit=3)*
Adâncimea maximă a fiecărui estimator de regresie individual. Aceasta limitează numărul de noduri din arbore. Ajustați acest parametru pentru performanță optimă; valoarea ideală depinde de interacțiunea variabilelor de intrare. Dacă este None, nodurile sunt extinse până când toate frunzele devin pure sau până când toate frunzele conțin mai puține eșantioane decât *min_samples_split*.

Bibliografie:

- RidgeClassifier — scikit-learn 1.7.0 documentation
- ADASYN — Version 0.13.0
- GridSearchCV — scikit-learn 1.7.0 documentation
- RandomForestClassifier — scikit-learn 1.7.0 documentation
- GradientBoostingClassifier — scikit-learn 1.7.0 documentation
- Hands-On Machine Learning with Scikit-Learn, Keras & TensorFlow, Aurélien Geron