1. **Metode de citire**
   1. **Din API** (c1 ex2, t1 ex1)
      1. *response = requests.get(url)* – efectueaza o cerere HTTP GET catre url-ul specificat
      2. *if response.status\_code == 200* – verifica daca raspunsul a fost OK
      3. *data = response.json()* – parseaza continutul raspunsului ca JSON intr-un dictionar
   2. **Web scraping** (c1 ex3, t1 ex2)
      1. *response = requests.get(url)* – efectueaza o cerere HTTP GET catre pagina specificata
      2. *soup = BeautifulSoup(response.text, “html.parser”)* – parseaza HTML-ul paginii si creeaza un obiect BeautifulSoup pentru navigare DOM
   3. **Baza de date** (c1 ex4)
      1. *conn = sqlite3.connect(‘bd.db’)* – deschide/creeaza fisierul .db si returneaza conexiunea
      2. *cursor = conn.cursor()* – creeaza un cursor pentru executarea interogarilor
      3. *cursor.execute(“CREATE TABLE IF NOT EXISTS table\_name (…)”* – defineste structura tabelului daca nu exista
      4. *cursor.executemany(“INSERT INTO table\_name(col1, col2, …) VALUES (?, ?, …)”* – insereaza multiple randuri, folosind parametrizare
      5. *conn.commit()* – salveaza modificarile
      6. *cursor.execute(“SELECT FROM table\_name WHERE col = ?”, x))* – extrage randuri filtrate dupa x
      7. *rows = cursor.fetchall()* – returneaza toate randurile rezultate ca lipsa de tuple
      8. *cursor.close()* – inchide cursorul
      9. *conn.close()* – inchide conexiunea
   4. **CSV** (t1 ex3, t2 ex4)
      1. *pd.read\_csv(url, sep=”\t”, names=column\_names, usecols=[“col1”, “col2”])* – incarca un fisier csv de pe un url sau disc local, cu separator tab
      2. *pd.read\_csv(“fisier.csv”, nrows=100)* – citeste primele 100 de randuri dintr-un fisier
2. **Operatii pe date**
   1. **Eliminarea duplicatelor** (c1 ex 5, t1 ex2)
      1. *df.drop\_duplicates()* – elimina duplicatele exacte pe toate coloanele, pastrand prima aparitie
      2. *df.drop\_duplicates(subset=[“col1”, “col2”], keep=”first”)* – elimina duplicatele bazate pe un subset de coloane, pastrand prima aparitie
   2. **Tratarea valorilor lipsa** (c1 ex6, t1 ex4, t4 ex1)
      1. *df[“col”].fillna(df[“col”].mean()* – imputare cu media coloanei
      2. *df[“col”].fillna(df[“col”].median()* – imputare cu mediana coloanei
      3. *df[“col”].fillna(df[“col”].mode()* – imputare cu modul coloanei
      4. *df.dropna()* – elimina randurile care au cel putin o valoare lipsa
      5. *df.isna().sum()* – afiseaza numarul de valori lipsa per coloanal
      6. *SimpleImputer(strategy='mean')* – imputare cu media coloanei
   3. **Normalizarea** (c1 ex7, t1 ex4, c3 ex9, t3 ex6, c4 ex1a)
      1. *minmax\_scaler.fit\_transform(df[[“col1”, “col2”, …])* – aplicarea scalarii min-max pe coloanele numerice
      2. *standard\_scaler.fit\_transform(df[[“col1”, “col2”, …])* – aplicarea scalarii standard, cu valorile centrate
   4. **Codificarea valorilor categorice** (c1 ex8, t1 ex4, c2 ex3, c3 ex1, c3 ex4, c3 ex9)
      1. *map({valoare1: 0, valoare2: 1})* – rezulta o coloana numerica cu 0 si 1
      2. *pd.get\_dummies(df, columns=[“col1”, “col2”], drop\_first=True)* – creeaza coloane dummy pentru fiecare valoare unica, iar drop\_first pe true arata ca prima coloana adaugata e eliminata
   5. **Extragerea cuvintelor cheie** (c1 ex9, c2 ex2, t2 ex5)
      1. *TfidfVeztorizer(stop\_words=”english”, max\_features=k)* – elimina cuvintele commune (the, and, etc.) si pastreaza doar primele K termini dupa scor TF-IDF
      2. *vectorizer.fit\_transform(corpus)* – potriveste vocabularul si transforma fiecare document intr-o vectorizare TF-IDF
      3. *vectorizer.get\_feature\_names\_out()* – returneaza vectorul celor mai representative k cuvinte-cheie
3. **Antrenare**
   1. **Naïve-Bayes** (c1 ex1, c2 ex2, t2 ex1, t2 ex2, t2 ex4) – presupune ca fiecare caracteristica e independenta de celelalte (clasificare de text, analiza de sentiment)
      1. *MultinomialNB()* – creeaza un clasificator Naïve Bayes pentru date discrete
      2. *fit(X\_train, y\_train)* – antreneaza modelul pe datele de intrare (X\_train) si etichete (y\_train)
      3. *predict (X\_test)* – returneaza etichetele prezise pentru datele noi (X\_test)
      4. *score(X\_test, y\_test)* – calculeaza acuratetea modelului
   2. **Regresie logistica** (t1 ex5, c2 ex1, c2 ex6, c2 ex7, c2 ex8, t2 ex2, t2 ex5, t2 ex6) – transforma scorul generat de o combinatie liniara intr-o probabilitate (clasificare binara)
      1. *X\_train, X\_test, y\_train, y\_test = train\_test\_split(X, y, test\_size=0.2, random\_state=42)* – imparte datele in set de antrenament (80%) si test (20%)
      2. *LogisticRegression()* – creeaza un clasificator de regresie logistica
   3. **Regresie logistica cu regularizare** (c2 ex5, c3 ex6, t3 ex4)
      1. *LogisticRegression(penalty=’Lx’, solver=’saga’, max\_iter=1000, random\_state=42) –* creeaza un clasificator cu regularizare Lx (L1 sau L2), unde L1 forteaza coeficientii sa devina 0
      2. *Ridge(alpha=1.0)* – creeaza un model de regresie liniara cu regularizare L2, penalizand patratul coeficientilor pentru a-i mentine mici
      3. *Lasso(alpha=1.0)* – creeaza un model de regresie liniara cu regularizare L1, penalizand suma absoluta a coeficientilor si fortand unii coeficienti sa devina exact 0
   4. **SVM** (c2 ex3) – incearca sa gaseasca intr-un spatiu de caracteristici o margine care separa cat mai bine clasele (performanta ridicata, set de date mic)
      1. *SVC (kernel=’linear’, random\_state=42)* – creeaza un clasificator SVM cu kernel liniar (alte optiuni: rbf, poly)
   5. **Decision tree** (c2 ex4, t2 ex3, c3 ex4, t3 ex3, t3 ex5) – imparte spatial caracteristicilor in regiuni pe baza unor criterii de impartire (pentru date care au relatii non-liniare)
      1. *DecisionTreeClassifier(max\_depth=3, random\_state=42)* – creeaza un arbore de decizie, limitand adancimea la 3
      2. *DecisionTreeRegressor(random\_state=42)* – creeaza un arbore de decizie pentru regresie, folosind criteriul de minimizare a erorii patratice medii pentru a alege split-urile si prezice o valoare continua
   6. **Regresie liniara** (c3 ex2, c3 ex3, c3 ex5, c3 ex8, t3 ex1, t3 ex2) – potriveste o linie dreapta prin date pentru a modela relatii liniare intre caracteristici si tinta
      1. *PolynomialFeatures(degree=x)* – genereaza noi caracteristici polinomiale pana la gradul x (pentru x=2 adauga coloanele 1, X, X2)
      2. *LinearRegression()* – creeaza un model de regresie liniara simpla
   7. **GridSearchCV** (c2 ex7, t2 ex6, t3 ex5, c4 ex3b) – mecanism de a gasi automat cei mai buni hiperparametri pentru orice model din cei de mai sus
      1. *GridSearchCV(estimator=DecisionTreeClassifier(random\_state=42), param\_grid=param\_grid, cv=5, scoring='accuracy', n\_jobs=-1)* – initializeaza o cautare prin grila a hiperparametrilor pentru DecisionTreeClasifier, folosind validare incrucisata e 5 fold-uri, masurand acuratetea si paralelizand cu toate nucleele disponibile
      2. *best\_params\_* – dictionarul care contine combinatia de valori din “param\_grid” care a obtinut scorul cel mai bun in timpul validarii incrucisate
      3. *best\_estimator\_* – instanta lui DecisionTreeClasifier antrenata pe intregul set de antrenament, cu hiperparametrii optimi gasiti pe GridSearchCV
   8. **KNN Regressors** (c3 ex9, t3 ex6, c4 ex1) – prezice valoarea tintei pe baza mediei tintelor celor mai apropiati vecini in spatial caracteristicilor
      1. *KNeighborsRegressor(n\_neighbors=x)* – creeaza un model KNN pentru regresie, unde x este numarul de vecini luati in considerare
   9. **Comune**
      1. *fit(X\_train, y\_train)* – antreneaza modelul pe caracteristicile X\_train si etichetele y\_train
      2. *predict(X\_test)* – genereaza predictiile (0/1) pentru datele noi X\_test
      3. *accuracy\_score(y\_test, y\_pred)* – calculeaza acuratetea, adica proportia predictiilor corecte fata de total
      4. *classification\_report(t\_test, y\_pred)* – afiseaza precision, f1-score si numarul de exemple pentru fiecare clasa
      5. *(y\_pred == y\_test).mean() –* calculeaza acuratetea simpla, adica verifica daca predictia e corecta
      6. *Feature\_importances\_* – returneaza importanta fiecarei caracteristici, adica cat de mult contribuie la separarea claselor
      7. *coef\_[0]* – returneaza vectorul de coeficienti pentru fiecare caracteristica care arata cat de importanta e caracterstica pentru decizia modelului, semnul indicand directia
      8. *intercept\_[0]* – returneaza vectorul bias-ul liniei de decizie
      9. *coef\_comparison["L1\_Coefficient"] != 0]["Feature"].tolist()* – returneaza lista caracteristicilor pentru care coeficientul L1 nu e zero
      10. *precision\_score(y\_test, y\_pred)* – calculeaza precizia
      11. *recall\_score(y\_test, y\_pred)* – calculeaza sensibilitatea, adica dintre exemplele pozitive cate reale au fost detectate
      12. *confusion\_matrix(y\_test, y\_pred)* – returneaza matricea de confuzie
      13. *plot\_tree(model, feature\_names=wine.feature\_names, class\_names=wine.target\_names, filled=True)* – deseneaza grafic structura arborelui de decizie, folosind feature\_names pentru etichetele caracteristicilor si class\_names pentru etichetele claselor
      14. *r2\_score(actual, predicted)* – returneaza coeficientul de determinare R2, adica proportia variantei tintei explicata de model
      15. *mean\_squared\_error(actual, predicted) –* calculeaza eroarea patratica medie (MSE)
      16. *mean\_absolute\_error(actual, predicted)* – calculeaza eroarea absoluta medie (MAE)
      17. *describe()* – statistici descriptive
4. **Clustering**
   1. **KMeans** (c4 ex1, c4 ex1a, c4 ex3a, c4 ex5) – imparte datele neetichetate in k clustere prin atribuirea fiecarui ounct celui mai apropiat centroid
      1. *KMeans(n\_clusters=3, random\_state=42)* – creeaza un model de clustering KMeans care imparte datele in 3 clustere
      2. *kmeans.inertia\_* – returneaza suma erorilor patratice (SSE) dintre fiecare punct si centrul clusterului sau, masurand cat de compacte sunt clusterele
   2. **DBSCAN** (c4 ex2, c4 ex3a, c4 ex4) – detecteaza regiuni dense de puncta si marcheaza punctele isolate ca outlieri
      1. *DBSCAN(eps=0.8, min\_samples=5)* – creeaza un model de clustering care detecteaza regiuni dense de puncta si marcheaza punctele isolate ca outlieri
      2. *labels\_* – returneaza eticheta clusterului atribuita fiecarui punct
   3. **Linkage** (c4 ex3) – realizeaza clustering ierarhic construind o matrice de legaturi intre puncte/clustere
      1. *linkage(X, method=’ward’)* – creeaza matricea de linkuri Z pentru clustering ierarhic
      2. *dendrogram(Z, labels=[…])* – afiseaza dendrograma bazata pe matricea Z aratand cum se unesc treptat clusterele pe masura ce creste distanta de legatura
   4. **Comune**
      1. *silhouette\_score(X, y\_pred)* – masoara coeziunea si separarea clusterelor
      2. *davies\_bouldin\_score(X, y\_pred)* – cuantifica similaritatea dintre fiecare cluster si cel mai apropiat alt cluster
      3. *calinski\_harabasz\_score(X, y\_pred)* – raportul dintre variatia totala explicate de clustere si varianta neexplicata
      4. *rand\_score(y\_true, y\_pred)* – proportia perechilor de puncte care sunt etichetate coerent
      5. *adjusted\_rand\_score(y\_true, y\_pred)* – masoara similaritatea intre doua partitii fara a fi influentat etichete permutate
      6. *f1\_score(y\_true, y\_pred, average='macro')* – media aritmetica a F1-score-urilor calculate separat pentru fiecare clasa
5. **Breviar de teorie**
   1. **Matricea de confuzie (Confusion Matrix)**
      1. Un tabel simplu cu patru numere care îți arată: câte exemple pozitive ai prezis corect (TP), câte negative ai prezis corect (TN), câte false alarme ai dat (FP) și câte cazuri pozitive ai ratat (FN).
   2. **Acuratețe (Accuracy)**
      1. Proporția totală de predicții corecte (este bună doar când clasele sunt aproximativ egale ca număr).
   3. **Precizie (Precision)**
      1. Dintre toate exemplele etichetate ca „pozitive”, câte erau cu adevărat pozitive (te interesează când vrei să eviți alarmele false).
   4. **Sensibilitate / Recall**
      1. Dintre toate exemplele care erau cu adevărat pozitive, câte le-ai prins (te interesează când nu vrei să ratezi cazuri importante).
   5. **F1-score**
      1. Un „echilibru” între precizie și recall, util când vrei să nu sacrifici niciuna dintre ele.
   6. **R² (Coeficientul de determinare)**
      1. Raportul care îți spune cât de bine „potrivește” modelul datele (1 = ideal, 0 = servește la fel de bine ca o constantă, < 0 = mai prost decât să ghicești media).
   7. **MSE (Mean Squared Error)**
      1. În medie, cât de „departe” sunt predicțiile față de valorile reale, cu penalizare mai mare pentru erorile mari.
   8. **MAE (Mean Absolute Error)**
      1. În medie, cu cât s-a înșelat modelul (valoarea absolută a diferențelor), fără penalizare exagerată pentru erori mari.