**Decision Trees**

* Terminologii ale algoritmului: splitting, noduri, ramura, pruning
* Esența Gini Impurity ***g=p\_clasa1(1-p\_clasa1)+ p\_clasa2(1-p\_clasa2)+...***
* Necesitatea minimizării Gini Impurity pentru nodurile frunza
* Etapele de elaborarea a unui Decision Tree cu Gini Impurity:
* Selectarea unei caracteristicii ce inițial se va considera nodului rădăcina
* Selectarea condiției de divizare (spliting) în functie de valorile caracteristicii
* Determinarea valorilor fiecărei clase din coloana eticheta in fiecare ramura
* Determinarea Gini Impurity pentru fiecare ramura ***g\_stanga*** și ***g\_dreapta***
* Determinarea medii ponderate a Gini Impurity ***g=(n\_stanga/n\_total)\*g\_stanga+ (n\_dreapta/n\_total)\*g\_dreapta***
* Repetarea procedurii cu altă caracteristica ca nod rădăcina
* Selectarea finala drept nod rădăcina a caracteristicii cu cea mai mica Gini Impurity
* Selectarea condiției de divizare pentru nodul cu 2 clase: conform valorilor claselor
* Selectarea condiției de divizare pentru nodul cu valori continue:
* sortare în ordine crescătoare a valorilor caracteristicii nodului
* determinarea mediei dintre valorile vecine ale caracteristicii nodului
* selectarea primei valori medii ca condiție de divizare (cu operatorul <=) si determinarea Ginii Impurity
* repetarea procedurii si cu celelalte valori medii
* selectarea drept condiție de divizare valoarea medie cu cea mai mica Gini Impurity
* Selectarea condiției de divizare pentru nodul cu multi-clase:
* selectarea primei clase ca condiție de divizare (cu operatorul ==) si determinarea Ginii Impurity
* repetarea procedurii si cu celelalte clase
* repetarea procedurii si cu combinații de 2 clase, 3 clase etc
* selectarea drept condiție de divizare clasa/combinația de clase cu cea mai mica Gini Impurity
* Necesitatea hiper-parametrului de prag minim de descreștere a gini impurity
* Necesitatea hiper-parametrului de adâncime maxima
* Algoritmul de elaborare a modelului Decision Trees în Scikit-Learn:
* importul datelor
* vizualizarea informației despre date ***df.info()***
* verificarea datelor lipsa ***df.isnull().sum()***
* excluderea datelor lipsa ***df.dropna()***
* determinarea valorilor unice in coloanele categoriale ***df[col].unique()***
* vizualizarea grupării claselor in funcție de 2 caracteristici numerice ***sns.scatterplot(data=df, x=col1, y=col2, hue=col3)***
* vizualizarea grupării claselor in funcție de toate caracteristici numerice ***sns.pairplot(data=df, hue=col\_label)***
* vizualizarea distributie valorilor unei caracteristici numerice in functie de 2 caracteristici categoriale ***sns.catplot(data=df, x=col\_cat1, y=col\_num, col=col\_cat2, kind=’box’)***
* transformarea caracteristicilor categoriale string in categoriale numerice ***pd.get\_dummies(df.drop(col\_label,axis=1),drop\_first=True)***
* crearea setului X si y ***X=df.drop(col\_label, axis=1), y=df[col\_label]***
* crearea seturilor de train si de test ***X\_train, X\_test, y\_train, y\_test = train\_test\_split(X, y, test\_size=0.3, random\_state=9)***
* importul si crearea modelului DecisionTreeClassifier de bază **model = DecisionTreeClassifier()**
* ajustarea modelului pe datele de training ***model.fit(X\_train, y\_train)***
* realizarea predicției ***model.predict(X\_test)***
* determinarea acurateti modelului ***accuracy\_score(y\_test,y\_pred)***
* determinarea confusion matrix a modelului ***confusion\_matrix(y\_test,y\_pred)***
* vizualizarea grafica a confusion matrix a modelului ***plot\_confusion\_matrix(model,X\_test,y\_test)***
* elaborarea classificasion report a modelului ***print(classification\_report(y\_test,y\_pred))***
* determinarea importantei fiecarei caracteristici ***model.feature\_importances\_***
* Importul functiei de vizualizare a arborelui ***from sklearn.tree import plot\_tree***
* vizualizarea arborelui ***plot\_tree(model);***
* parametrizarea arborelui ***plot\_tree(model, filled=True, feature\_names=X.columns);***
* Influența hiper-parametrului Max Depth:
* crearea modelului DecisionTreeClassifier cu parametru Max Depth de diferite valori **model = DecisionTreeClassifier(max\_depth=2)**
* ajustarea modelului pe date, realizara predictei, elaborarea classificasion report, vizualizarea confusion matrix și a arborelui
* Influența hiper-parametrului Max Leaf Nodes:
* crearea modelului DecisionTreeClassifier cu parametru Max Leaf Nodes de diferite valori **model = DecisionTreeClassifier(max\_leaf\_nodes=3)**
* ajustarea modelului pe date, realizara predictei, elaborarea classificasion report, vizualizarea confusion matrix și a arborelui
* Influența hiper-parametrului Criterion:
* crearea modelului DecisionTreeClassifier cu parametru Criterion entropy **model = DecisionTreeClassifier(criterion='entropy')**
* ajustarea modelului pe date, realizara predictei, elaborarea classificasion report, vizualizarea confusion matrix și a arborelui