**AdaBoost**

* Boosting în calitate de algoritm meta-learning
* Stump (model slab) – un model foarte simplu a oricărui algoritm ce nu poate asigura performanțe bune
* Suma ponderata a tuturor stump-urilor rezultă într-un model foarte puternic
* Algoritmul AdaBoost – crearea serie a stump-urilor prin adaptarea datelor greșite în stump-ul anterior
* Algoritmul de implementare teoretica a AdaBoost:
* Determinarea ponderii eșantioanelor ca ***pond\_esant=1/n\_esantioane***
* Elaborarea arborilor stump conform tuturor caracteristicilor
* Selectarea primului stump pe baza proprietarii informaționale (entropie sau gini impurity)
* Determinarea erorii totale a acestui stump ***err\_total=n\_err\*pond\_esant***
* Determinarea performantei acestui stump ***perf=(1/2)\*ln[(1-err\_total)/err\_total]***
* Actualizarea ponderii eșantioanelor eronate ***pond\_err=pond\_esant\*exp(perf)***
* Actualizarea ponderii eșantioanelor corecte ***pond\_cor=pond\_esant\*exp(-perf)***
* Normalizarea ponderilor actualizate ***pond\_norm=pond\_err|pond\_cor/sum(pond\_err&pond\_corr)***
* Crearea intervalelor de probabilitate pentru ponderile normalizate in gama 0...1
* Crearea unui nou set de date din același număr de eșantioane din datele vechi prin selectarea aliatoare a acestora conform intervalelor de probabilitate
* Repetarea procedurii pentru noul set de date
* Rezultatul predicției: suma dintre produsul rezultatele fiecărui stump și performanta sa
* Algoritmul de elaborare a modelului AdaBoostClassifier în Scikit-Learn:
* importul datelor
* vizualizarea informației despre date ***df.info()***
* verificarea prezentei lipsurilor ***df.isna().sum()***
* Vizualizarea informatiei statistice despre date ***df.describe().transpose()***
* determinarea numărului de valori pentru clasele label ***sns.countplot(data=df,x=col\_label)***
* afișarea grafica a numărului de valori unice pe categorie ***sns.barplot(data=df.describe().transpose().reset\_index().sort\_values('unique'),x='index',y='unique')***
* crearea setului X si y ***X=df.drop(col\_label, axis=1), y=df[col\_label]***
* transformarea datelor categoriale in numerice in setul X ***X = pd.get\_dummies(X,drop\_first=True)***
* crearea seturilor de train si de test ***X\_train, X\_test, y\_train, y\_test = train\_test\_split(X, y, test\_size=0.15, random\_state=9)***
* importul algoritmului AdaBoostClassifier ***from sklearn.ensemble import AdaBoostClassifier***
* crearea modelului AdaBoostClassifier cu un stump ***model = AdaBoostClassifier(n\_estimators=1)***
* 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 fiecărei caracteristici ***model.feature\_importances\_***
* determinarea denumirii caracteristicii celei mai importante ***X.columns[model.feature\_importances\_.argmax()]***
* vizualizarea numărului de valori ale clasei label in funcție de caracteristică ce mai importanta ***sns.countplot(data=df,x=col\_importanta,hue=col\_label)***
* Influența numărului de stump-uri a modelului AdaBoostClassifier în Scikit-Learn:
* determinarea numărului de coloane ***len(X.columns)***
* elaborarea modelelor AdaBoostClassifier cu numarul de stump-uri din gama range(1,len(X.columns)+1) și determinarea erorilor de predictie
* afișarea grafică a dependenței valorii erorii de numărul de stamp-uri ***plt.plot(range(1, len(X.columns)+1),lista\_erori)***
* crearea modelului AdaBoostClassifier cu numarul optim de stump-uri ***model = AdaBoostClassifier(n\_estimators=n\_optim)*** si determinarea caractersiticii cele mai importante
* afișarea grafica a caracteristicilor în functie de importanta
* crearea modelului AdaBoostClassifier cu numarul maxim de stump-uri ***model = AdaBoostClassifier(n\_estimators=n\_max)*** si determinarea caractersiticii cele mai importante
* afișarea grafica a caracteristicilor în functie de importanta