**Naive Bayes Classification**

* Esența teoremei lui Bayes **p(a|b)=p(b|a)\*p(a)/p(b)**
* Utilizarea teoremei Bayes în probleme de clasificare **p(ck|x)=p(x|ck)\*p(ck)/p(x)** unde x={x1,x2,…xn} caracteristici, iar ck – clasa k in coloana label
* Daca se presupune că caracteristicile sunt independente atunci **p(ck|x) proportional p(ck)\*p(x1|ck) \*p(x2|ck)\*…**
* Algoritmul de implementare teoretica a Multinomial Naive Bayes în Natural Language Processing (NLP):
* Se divizează textele pe clase
* Se determina probabilitatea fiecărei clase conform numărului de texte **P(cl)**
* Se creează vocabularul din toate textele
* Se vectorizează cuvintele pe clase: numărul de apariție a cuvintelor vocabularului în fiecare clasă
* Se sumează 1 la număr de apariție a fiecărui cuvinte pentru a se evita existența valorii 0
* Pentru fiecare clasă se determina probabilitatea de apariție a fiecărui cuvânt: numărul de apariție a cuvântului divizat la numărul toata de cuvinte în clasa dată **P(cuvn|cl)**
* Predicția clasei pentru un text nou va fi clasa care va avea valoarea cea mai marea pentru **P(cl)\* P(cuv1|cl)\* P(cuv2|cl)...**
* Feature extraction pentru text – transformare cuvintelor în numere în funcție de importanța lor conform claselor de clasificare
* **Count Vectorization** - creează vocabularul din cuvintele tuturor textelor și pentru fiecare text formează o listă de valori numerice ce corespunde numărului de apariție a cuvintelor vocabularului în textul respectiv
* Count Vectorization formează Document Term Matrix (DTM): listele de valori numerice pentru toate textele
* Neajunsul Count Vectorization: 1) valoare numerică mare pentru cuvintele scurte, fără semnificație și des utilizate; 2) prezența valorilor numerice și pentru cuvintele caracteristice tuturor claselor ce nu ajută la clasificare
* Stop Words include cuvintele scurte larg utilizate fără semnificație care se vor exclude din vocabular
* **TF-IDF – Term Frequency** **– Inverse Document Frequency** – transformă cuvintele în numere cu valoare proporțională importanței lor pentru clasificare
* TF-IDF se determină ca numărul **nC,t** de apariții a cuvântului **C** în textul **t** înmulțit logaritmul dintre numărul total de texte **N** divizat la numărul de texte NC în care apare cuvântul **C TF-IDF=nC,t\*log(N/NC)**
* Algoritmul de extragere a caracteristicilor pentru text în Scikit-Learn:
* Includerea textelor într-o listă ***text***
* Importul modului CountVectorizer ***from sklearn.feature\_extraction.text import CountVectorizer***
* Crearea convertorului ***cv = CountVectorizer()***
* Ajustarea convertorului pe lista de texte și crearea matricii DTM sparse ***sparse\_mat = cv.fit\_transform(text)***
* Afișarea matricei DTM dense ***sparse\_mat.todense()***
* Vizualizarea vocabularului cv.vocabulary\_
* Repetarea procedurii cu includerea stop\_words ***cv2 = CountVectorizer(stop\_words='english')***
* Importul modului TfidfTransformer ***from sklearn.feature\_extraction.text import TfidfTransformer***
* Crearea convertorului ***tfidf\_transformer = TfidfTransformer()***
* Ajustarea convertorului pe matricea sparse și crearea matricii tfidf ***tfidf = tfidf\_transformer.fit\_transform(sparse\_mat)***
* Afișarea matricei tfidf ***tfidf.todense()***
* Importul modulului Pipeline ***from sklearn.pipeline import Pipeline***
* Crearea unui pipe pentru vectorizare și transformare ***pipe = Pipeline([('cv',CountVectorizer()),('tfidf',TfidfTransformer())])***
* Ajustarea pipe pe lista de texte și crearea matricei tfidf ***rezultat = pipe.fit\_transform(text)***
* Afișarea matricei tfidf ***rezultat.todense()***
* Importul modului TfidfVectorizer ***from sklearn.feature\_extraction.text import TfidfVectorizer***
* Crearea convertorului ***tfidf = TfidfVectorizer()***
* Ajustarea convertorului pe lista de texte și crearea matricei tfidf ***rezultat\_nou*** ***= tfidf.fit\_transform(text)***
* Afișarea matricei tfidf ***rezultat\_nou.todense()***
* Algoritmul de utilizarea a modelului Naive Bayes pentru NLP în Scikit-Lean:
* Importul datelor
* Analiza datelor prin vizualizarea grafică a valorilor mai multor coloane sns.countplot()
* Crearea seturilor X și y
* Crearea seturilor de train si de test
* Crearea convertorului cu cuvinte stop ***tfidf = TfidfVectorizer(stop\_words='english')***
* Ajustarea convertorului pe setul X\_train și crearea matricei tfidf ***X\_train\_tfidf = tfidf.fit\_transform(X\_train)***
* aplicarea convertorului și asupra setului X\_test ***X\_test\_tfidf = tfidf.transform(X\_test)***
* importul nodulului algoritmului Naive Bayes Multinomial ***from sklearn.naive\_bayes import MultinomialNB***
* Crearea modelului MultinomialNB ***nb = MultinomialNB()***
* ajustarea modelului pe datele de training ***nb.fit(X\_train\_tfidf, y\_train)***
* realizarea predicției **y\_*pred = nb.predict(X\_test\_tfidf)***
* 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(nb,X\_test\_tfidf,y\_test)***
* elaborarea classificasion report a modelului ***print(classification\_report(y\_test,y\_pred))***