Inteligență Artificială - Tema 2

- When does Santa's shopping bring revenue? -

1. Descriere generala

În practica de zi cu zi a unui inginer sau cercetător în domeniul inteligenței artificiale intra frecvent următoarele aspecte:

- Vizualizarea și "explorarea" datelor unei probleme (Exploratory Data Analysis) pentru a înțelege valorile unui atribut (eng. feature)
- Investigarea de modele de învățare automată (machine learning) pentru predicția unui obiectiv dorit

Obiectivul temei este cel de a pune în aplicare aspectele de mai sus pe un set de date ce analizează potențialul de a prezice dacă tiparul de navigare online pe site-uri de e-commerce conduce într-un final la cumpărarea unui produs (și, implicit, la venit pentru magazinul online).

2. Descrierea Setului de Date

Setul de date oferit este în format .csv și conține următoarele coloane, fiecare linie fiind asociată unei interacțiuni cu platforma online de e-commerce:

- **Administrative**, **Informational**, **ProductRelated** (int)
  Numărul de pagini de acest tip (administrative, informative, legate de produs) pe care utilizatorul le-a vizitat.
- **Administrative_Duration**, **Informational_Duration**, **ProductRelated_Duration** (float)
  Durata petrecută în fiecare dintre categoriile de pagini descrise mai sus.
- **BounceRates** (float)
  Procentul de vizitatori care intră pe site prin acea pagină și ies fără a declanșa sarcini suplimentare.
- **ExitRates** (float)
  Procentul de vizualizări ale site-ului care se termină la această pagină.
- **PageValues** (float)
  Valoarea medie a paginii, calculată în raport cu valoarea paginii țintă și/sau finalizarea unei tranzacții eCommerce.

- **SpecialDay** (float)
  Această valoare reprezintă proximitatea datei de navigare față de zilele speciale sau sărbători (de exemplu: Crăciun, Ziua Mamei, etc.).
- **Month** (string)
  Luna calendaristică în care a avut loc vizita.
- **OperatingSystems** (int)
  Sistemul de operare folosit.
- **Browser** (int)
  Browserul folosit.
- **Region** (int)
  Regiunea utilizatorului.
- **TrafficType** (int)
  Tipul de trafic: referral, direct, etc.
- **VisitorType** (str)
- Tipul utilizatorului: new_visitor, returning_visitor, etc.
- **Weekend** (bool)
  Arată dacă vizita a avut loc în weekend sau nu.
- **Revenue** (bool) -> **ținta predicției voastre**
  Arată dacă magazinul a încheiat sau nu o tranzacție în urma acestei vizite (dacă utilizatorul a cumpărat sau nu produsul)

Citirea seturilor de date

Pentru citirea setului de date vă recomandăm să folosiți biblioteca Pandas:

```
import pandas as pd

df = pd.read_csv('training_dataset.csv')

df.head()
```

Astfel puteți beneficia de facilitățile formatului DataFrame:
- Suport pentru citire și salvare folosind formate populare precum: csv, xls, parquet, etc.
- Generare automată a statisticilor folosind metode precum: **.head(), .count(), .mean(), .sum()**, etc.
- Filtrare facilă: **data[data['column'] > value]**, **df.isnull()**, etc.
- Grupare și agregare: **df.groupby('column').mean()**
- Sortare: **df.sort_values('column')**
- Și multe altele

3. Cerințe

3.1. Explorarea Datelor (Exploratory Data Analysis) [2p]

Primul pas cerut în rezolvarea unei probleme de clasificare este câștigarea unor cunoștințe asupra caracteristicilor principale ale datelor problemei.
De regulă, foarte folositoare în această etapă este aplicarea unor metode de ***vizualizare a datelor*** și de ***raportare a distribuțiilor de valori*** ale variabilelor folosite în predicție.

Analize sugerate

1. Analiza echilibrului de clase

Realizați un grafic al frecvenței de apariție a fiecărei etichete (clase) în setul de date de antrenare / test, folosind **barplot-uri** / **countplot**.
Pentru realizarea unor astfel de barplot-uri puteți folosi mai multe biblioteci:

- Folosind biblioteca seaborn pentru barplot sau countplot
- Direct dintr-un DataFrame Pandas folosind pandas.DataFrame.plot.bar

2. Vizualizarea atributelor

A. Realizați o analiză a distribuției valorilor pentru fiecare atribut. Țineți cont de faptul că atributele din setul de date furnizat sunt atât **numerice,** cât și **categorice.**
   - Identificați atributele numerice. Pentru fiecare dintre acestea, **realizați un grafic** care să prezinte **distribuția valorilor în percentile cu granularitate de 10%** (e.g. câte sample-uri din dataset au valori cuprinse în intervalul [min, min+1/10 (max-min)], câte sample-uri din dataset au valori cuprinse în intervalul [min+1/10 (max-min), min+2/10 (max-min)], sasmd).
     - Folosiți numpy.percentile ca indicații în acest sens.
   - Pentru atributele categorice **faceți un grafic al histogramei** pentru fiecare valoare posibilă a acestora.
     - Folosiți pandas.plot.hist pentru indicații.

B. Realizați o analiză a gradului de corelare al fiecărui atribut cu variabila țintă (Revenue). **Notă:** tipul de analiză a corelației depinde de tipul atributului (numeric sau categorial).
   - Pentru toate atributele pe care le-ați identificat a fi **numerice**, calculați coeficientul de **Point-Biserial Correlation** (explicatie pe Wikipedia aici, exemplu de calculare in Python aici). Treceți într-un tabel indexat pe

linii după numele atributului, valorile pentru coeficientul obținut și valoarea **p-value** aferentă.
        - Pentru atributele care au p-value <= 0.05, afișați **pe un același grafic,** folosind un bar plot, coeficientul de Point-Biserial Correlation**.**
    - Pentru toate atributele de tip **categorial**, efectuați testul Pearson Chi-squared (explicație Wikipedia aici, exemplu de calcul in Python aici). Treceți într-un tabel indexat pe linii după numele atributului, valorile pentru coeficientul obținut și valoarea **p-value** aferentă.
        - Pentru atributele care au p-value <= 0.05, afișați **pe un același grafic,** folosind un bar plot, coeficientul statisticii chi-squared**.**

    - **Analizați rezultatele pe baza tabelelor și graficelor obținute. Ce indică aceste grafice despre puterea de predicție a fiecărui atribut în parte? Cum vă puteți folosi de informația aflată în cadrul unui task de predicție?**

3.2. Antrenarea și Evaluarea Algoritmilor de Predicție [8p]

3.2.1. Regresie Logistică [3p]

Dezvoltați un model de regresie logistică pentru predicția intenției de a cumpăra sau nu.

Înaintea rulării propriu-zise a algoritmului țineți cont de următoarele:

- Pentru a putea aplica regresia logistică, variabilele **categoriale** (incluzând-o pe cea țintă) trebuie convertite într-o formă numerică. Utilizați în acest scop clasa utilitară LabelEncoder din biblioteca scikit-learn.
- Este posibil ca atributele din setul de date să conțină valori extreme (outliers). Explorați utilizarea metodelor de **normalizare a datelor** folosind în mod comparativ MinMaxScaler, StandardScaler, RobustScaler.
    - **Când efectuați analiza rezultatelor, raportați în tabelul de rezultate care metodă de scalare a dat rezultate mai bune. Justificați!**

Se cer următoarele două abordări:

1. **Implementare manuală [1.5p]**
    - Pornind de la rezolvarea din laborator, implementați modelul de regresie logistică pentru problema dată.
2. **Implementare folosind biblioteca scikit-learn [1.5p]**
    - Folosiți documentația din biblioteca scikit-learn pentru a antrena și evalua un model de regresie logistică.

3.2.2. Arbore de Decizie [3p]

Pașii necesari pentru normalizarea datelor sunt similari cu cei din cazul regresiei logistice. **Nu mai este obligatorie** conversia variabilelor categorice la valori numerice. Tipul lor trebuie însă avut în vedere atunci când se crează split-ul într-un nod al unui arbore de decizie.

Se cer următoarele două abordări.

1. Antrenați și evaluați un model de arbore de decizie folosind biblioteca scikit-learn. Documentația pentru DecisionTreeClassifier este disponibilă aici. [2p]
2. Implementați un model de arbore de decizie plecând de la codul din laborator [1p]
    - **Atenție:** pentru că atributele din setul de date pot fi și numerice, este nevoie să considerați crearea split-ului de atribute numerice, folosind metoda împărțirii binare pe intervale. Găsiți exemple de implementare în acest blog și în acest video explicativ.

3.2.3. Evaluare Comparativă a Rezultatelor [2p]

Procedura generală de antrenare și evaluare este următoarea:

- Realizați 10 împărțiri aleatoare ale setului de date în 80% date pentru antrenare și 20% pentru test. Pentru împărțire folosiți metoda train_test_split din scikit-learn.
- Pentru fiecare din cele 10 împărțiri:
    - Antrenați algoritmul de Regresie logistică (în varianta scikit, cât și de implementare manuală) cu fiecare opțiune de scalare a datelor (MinMax, Standard, Robust).
        - Evaluați pe datele de test din split-ul curent, calculând precizie, recall și F1
    - Antrenați algoritmul de Decision Tree (în varianta scikit, cât și de implementare manuală) variind adâncimea maximă a arborelui (între 3 și 6) și pentru fiecare opțiune de scalare a datelor (MinMax, Standard, Robust, fără scalare)
        - Evaluați pe datele de test din split-ul curent calculând precizie, recall și F1

- Pentru fiecare configurație a algoritmilor (e.g. Regresie Logistică folosind Robust Scaler; Arbore de Decizie de adâncime 6, folosind MinMaxScaler) **raportați media și varianța metricilor de precizie, recall și F1 într-un tabel comun.**
    - **Evidențiați prin bolduire** care este configurația pentru fiecare tip de algoritm, care obține cea mai bună metrică F1**.**

- **Comentați asupra rezultatelor, explicând de ce credeți că setup-ul cu rezultat mai bun pe care îl observați în tabel obține această performanță.**