Rusu David-Gabriel -314CB

Proiect PCLP3

Cerinta 1:
Pentru acest script am utilizat biblioteca panda pentru a putea identifica anomaliile dintr-un set de date predifinit, in cazul nostru, “train.csv”.

Rezolvare:

- Am inclus bibilotecile necesare
- Am citit setul de date initial si l-am memorat in variabila date
- Am definit o functie elimina. Aceasta are ca parametrii coloana pentru care vrem sa identificam outlierii si setul de date initial. Calculeaza Q1 si Q3, luand coloana specificata si Q1 reprezinta percentline de 25%, iar Q3 reprezinta percentlineul de 75%. Ulterior IQR este dat de diferenta lor si am calculat threshold-ul inferior si superior pentru a putea elimina outlierii din setul de date
- Aplicam functia pe date si obtinem date_noi
- Afisam diferenta de size si setul de date nou obtinut.

```
def elimina(data, col):
    Q1 = data[col].quantile(0.25)
    Q3 = data[col].quantile(0.75)
    IQR = Q3 - Q1
    inferior = Q1 - 1.5 * IQR
    superior = Q3 + 1.5 * IQR
    return data[(data[col] >= inferior) & (data[col] <= superior)]
```

Diferenta de marime intre setul de date original si cel actualizat:

```
Dimensiunea initiala: (891, 12)
Dimensiunea modificata: (703, 12)
```

(891, 12) -> (703,12) (coloanele raman neschimbate, deoarece stergem o intreaga linie.

Cerinta 2:
Acest script elimina outlierii bazati pe Z_score-ul fiecarui pasager. Z-score-ul este calculat pentru fiecare pasager, prin scaderea varstei a tuturor pasagerilor din varsta pasagerului curent si apoi impartind la deviatie medie de varsta.

Rezolvare:

- Am introdus valoarea lui z pentru care vom determina outlierii
- Am definit functia care va primi ca parametru setul de date, coloana dupa care vrem sa aflam z_score si z dupa care trebuie sa eliminam pasagerii
- Am aplicat functia setului de date si am obtinut un nou set de date

- Am afisat dimensiune initiala si diminesiunea actualizata a setului de date.

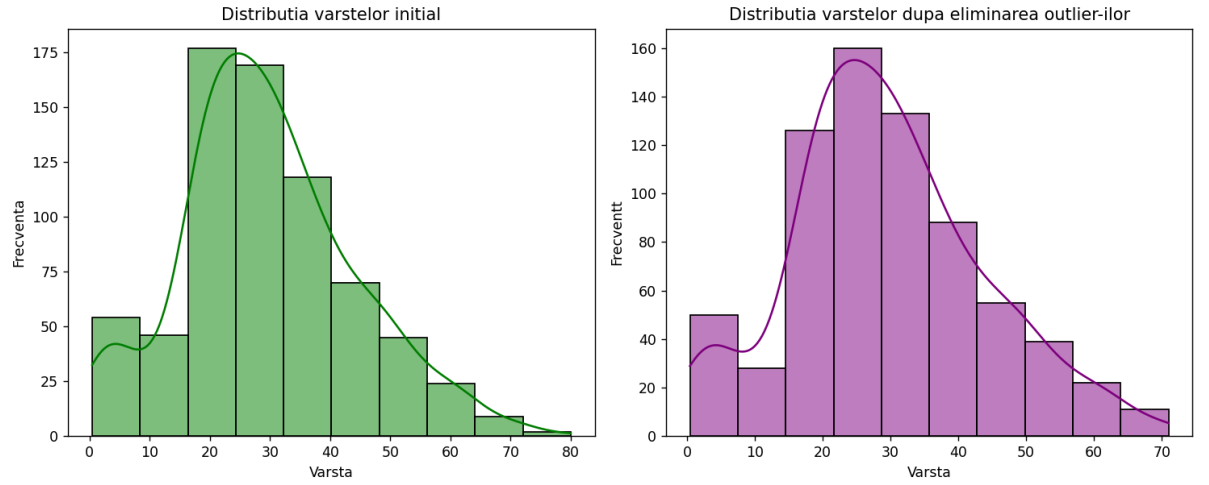
Diferenta de marime intre setul de date original si cel actualizat:

```
Introduceti valoarea pentru z:3
Dimensiunea initiala: (891, 12)
Dimensiunea noua: (712, 12)
```

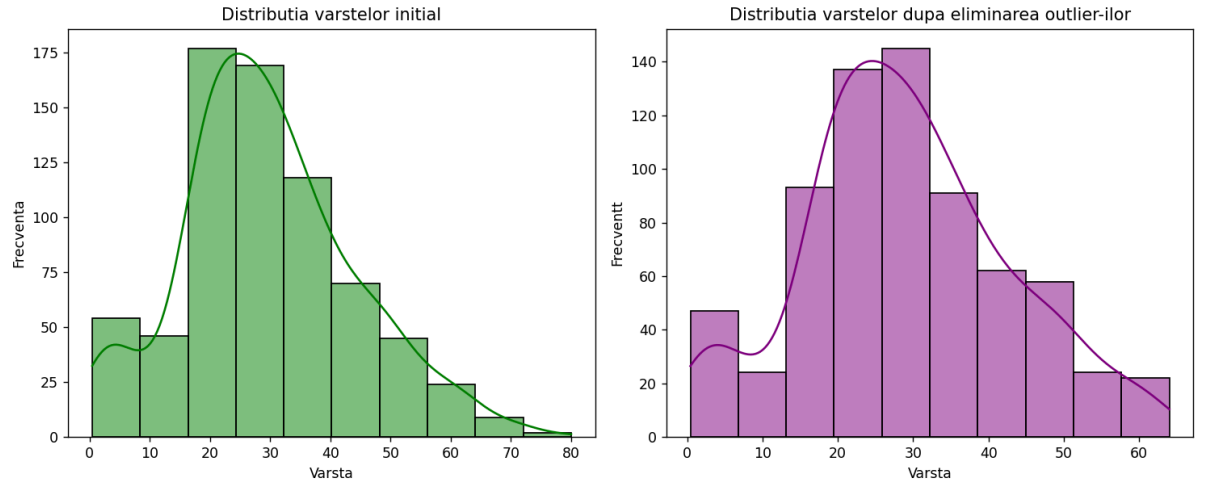
Cerinta 3:

Acestea sunt graficele pentru fiecare dintre cele doua metode de eliminare a outlierilor in functie de varsta.

Cu metoda folosind Z_score (z = 3)



Cu metoda folosind IQR



Codul folosit pentru afisarea graficelor este urmatorul:

```
figura, axe = plt.subplots(1, 2, figsize=(12, 5))
```

Aceasta linie de cod creeaza o figura care contine doua subgrafice alaturate. Prin plt.subplots(1,2) se specifica ca avem un rand si doua coloane de subgrafice,

figsize(12,5) specifica dimensiunile figurii (lungimea și lățimea). Aceste doua variabile vor reprezenta axele si figura unde vor fi afisate graficele.

```
sns.histplot(date['Age'], bins=10, kde=True, color='green', ax=axe[0])
```

Aceasta linie de cod creeaza primul grafic si va avea aceeasi explicatie ca si pentru graficul al doilea:

- sns.histplot creeaza histograma
- date[Age], respectiva date_noi[Age] sau fara_outlieri[Age] specifica ce vrem sa vizualizam
- bins = 10 specifica numarul de bare intervale in histograma
- kde = True adauga linia de densitate
- color = ce culoare sa fie graficele
- ax = axe[0] subgraficul in care vrem sa plasam graficul

Cerinta 4:

Acest script creeaza un model de prezicere a supravietuirii unuia dintre pasageri in functie de diferite caracteristici prezente in train.csv. Scriptul se foloseste de urmatoarele biblioteci pentru a antrena modelul.

```
import seaborn as sns
from sklearn.model_selection import train_test_split
from sklearn.ensemble import RandomForestClassifier
from sklearn.metrics import accuracy_score
```

1)Protocolul de testare:

- Aici impartim setul de date modificat (am aplicat cele doua functii de eliminare a outlierilor pentru varsta asupra setului de date). Si impartim setul de date in 2 parti (pasageri_antrenament), (pasageri_valizi) si (starea pasagerilor, adica mort sau viu).

Functia folosita este urmatoare:

```
train_test_split(final_data.drop(columns=['Survived']), final_data['Survived'], test_size=0.2, random_state=20)
```

2) Preprocesare datelor:

Scopul acestei parti din script este de a completa datele pasagerilor cat mai mult, astfel incat modelul nostru sa aiba cat mai multe informatii cu care sa lucreze, de exemplu transforma datele de Barbat sau Femeie in 0 si 1 pentru a lucra cu acestea.

```
def umple(data):
    medie_varsta = data['Age'].mean()
    medie_fare = data['Fare'].mean()

    data['Age'].fillna(medie_varsta, inplace=True)
    data['Fare'].fillna(medie_fare, inplace=True)
```

Aceasta functie umple campul unui pasager, in cazul in care este gol, cu media generala a oamenilor de pe vas.

```
def encode(data):
    data['Sex'] = data['Sex'].map({'male': 0, 'female': 1})
```

Transforma sexul pasagerului in 0 sau 1.

```
def standardizeaza(data):
    medie_varsta = data['Age'].mean()
    varsta_std = data['Age'].std()
    data['Age'] = (data['Age'] - medie_varsta) / varsta_std

    medie_fare = data['Fare'].mean()
    fare_std = data['Fare'].std()
    data['Fare'] = (data['Fare'] - medie_fare) / fare_std
```

Normalizeaza valorile numerice.

```
pasager_antrenare= pasager_antrenare.drop(columns=['Name', 'Cabin', 'Embarked', 'Ticket'])
pasager_valid = pasager_valid.drop(columns=['Name', 'Cabin', 'Embarked', 'Ticket'])
```

Eliminam coloanele care nu pot fi transformate in valori numerice.

3) Antrenarea modelului

Aici este creat un clasificator Random Forest cu un random_state = 20 specificat pentru a putea reproduce acelasi rezultat daca vrem sa-l rulam de mai multe ori. Acesta este antrenat pe datele de antrenare (pasager_antrenare si pasager_valid) folosind metoda fit.

```
# 3. Antrenarea modelului
clf = RandomForestClassifier(random_state=20)
clf.fit(pasager_antrenare, stare_antrenare)
```

4) Evaluarea modelului

```
predictie = clf.predict(pasager_valid)
acuratete = accuracy_score(stare_valid, predictie)
print("Acuratetea implementarii este de: {:.2f}%".format(acuratete * 100))
```

Afisam predictia modelului.