KNN VS Most Frequent imputer

Wednesday, January 29, 2025 2:23 PM

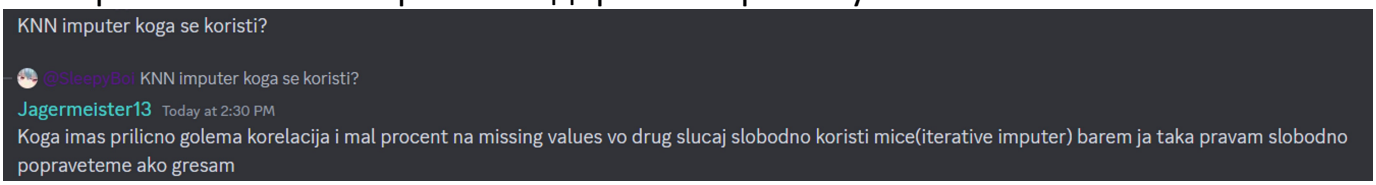
```
df.describe()
```

	BI-RADS assessment	Age	Shape	Margin	Density	Severity
count	959.000000	956.000000	930.000000	913.000000	885.000000	961.000000
mean	4.348279	55.487448	2.721505	2.796276	2.910734	0.463059
std	1.783031	14.480131	1.242792	1.566546	0.380444	0.498893
min	0.000000	18.000000	1.000000	1.000000	1.000000	0.000000
25%	4.000000	45.000000	2.000000	1.000000	3.000000	0.000000
50%	4.000000	57.000000	3.000000	3.000000	3.000000	0.000000
75%	5.000000	66.000000	4.000000	4.000000	3.000000	1.000000
max	55.000000	96.000000	4.000000	5.000000	4.000000	1.000000

INSIGHTS:

1) Според count можеме да согледаме дека има missing вредности
2) Ги гледаме просекот и медијаната, ако постои голема разлика меѓу нив тоа е индикатор дека постои **искривување** и податоците треба да бидат **нормализирани**
3) Можеме да согледаме дека Shape, Margin и Density се категорични податоци (иако се претставени нумерички) затоа што имаат фиксни вредности од 1-4 или 1-5

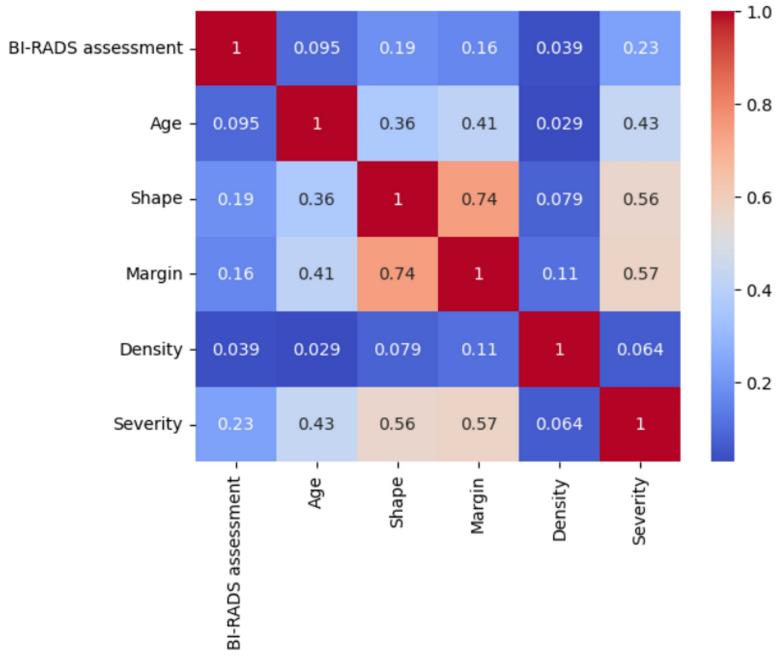
- Проверка: print(data['Shape'].unique()) … така за сите

- Пополнување на missing values:
  - Категориски: не може да користиме просек или медијана бидејќи немаат нумеричко значење, затоа ја користиме најчестата вредност
    Упатство: да се избегнува most frequent imputer
  - Правилниот распоред за категориски податоци е: label encoding -> KNN но во случајов веќе ни се нумерички претставени категориите па директно преминуваме на KNN
- 

- KNN Imputer: Се користи кога има значителна корелација и мал процент на missing values

1) Проверка за корелација

```
corr_matrix = df.corr()
sns.heatmap(corr_matrix, annot=True, cmap='coolwarm')
plt.show()
```



- Анализа: Shape и Margin имаат значителна корелација (0,74) + мал % missing values што значи дека се соодветни за KNN imputation
- Density иако е категорична нема висока зависност од другите 2 па може едноставно да се пополни со најчестата вредност (мода) или most frequent imputer

Most Frequent imputer

- Simple Imputer бара 2D низа, затоа во првиот пример се применува reshape

```
frequent_imputer = SimpleImputer(strategy='most_frequent')
data['HomePlanet'] = frequent_imputer.fit_transform(data['HomePlanet'].values.reshape(1, -1)
```

- Во вториот пример само се стават колоните во две загради што го дава истиот резултат

```
frequent_imputer = SimpleImputer(strategy='most_frequent')
data[['HomePlanet']] = frequent_imputer.fit_transform(data[['HomePlanet']])
```