**Tricks/points—**

**General**

1. Aggregation

columns\_of\_interest=['Genetic\_Risk','Air\_Pollution','Alcohol\_Use','Smoking','Obesity\_Level']

summary= data[columns\_of\_interest] . agg(['mean','std','min','max'])

2. In Python is a built-in function that makes it easy to loop over an iterable (like a list, tuple, string, etc.) and keep track of the index at the same time.

fruits = ["apple", "banana", "cherry"]

for fruit in fruits:

print(fruit)

---------------------------------------------------------------------------------------

fruits = ["apple", "banana", "cherry"]

for i, fruit in enumerate(fruits):

print(i, fruit)

3. Data leakage

It happens when your features contain information that would only be known after the prediction is made (i.e., information from the future).

Example (Bank loan default prediction

Target: default (Yes/No)

Feature: outstanding\_balance\_after\_3\_months

This feature directly reveals what happens after the default decision → leakage.

Train-Test Contamination\*\*

This happens when information from the test set leaks into training.\*\*

Example:\*\*

You normalize the entire dataset (train + test) together before splitting.

The mean and std from the test set leak into training.

Model looks great in evaluation but won’t generalize.

4. Counts=data[data['Cancer\_Type']=='Breast']['Cancer\_Stage'].value\_counts()

early\_stage\_sum=Counts.get('Stage 0',0)+Counts.get('Stage I',0)

5. data['Customer\_Status'] = data['Customer\_Status'].map({'Stayed': 0, 'Churned': 1})

6. import pandas as pd --LAMBDA FUNCTION & APPLY METHOD

# Example DataFrame

df = pd.DataFrame({

"price": [200000, 750000, 2000000, 500000, 1500000, 1600000]

})

# Create categorical column using lambda

df["category"] = df["price"].apply(

lambda x: "basic" if x <= 500000

else ("medium" if x <= 1500000 else "premium")

)

print(df)

**OR**

df = pd.DataFrame({ **--USING FUNCTION**

"price": [200000, 750000, 2000000, 500000, 1500000, 1600000]

})

# Define function for categorization

def categorize\_price(x):

if x <= 500000:

return "basic"

elif x <= 1500000:

return "medium"

else:

return "premium"

# Apply function to column

df["category"] = df["price"].apply(categorize\_price)

print(df)

***EDA***

**Numerical columns**

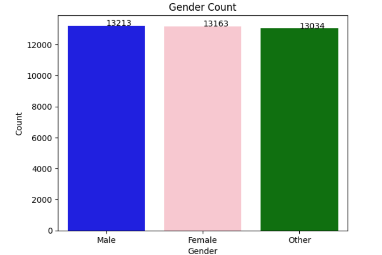
sns.kdeplot(data['Age'],fill=True,color="lightgreen")

sns.histplot(data['Age'],bins=30,kde=True,color="cyan")

**Categorical columns**

***Barplot/count plot***

sns.barplot(x=data['Gender'].value\_counts().index,y=data['Gender'].value\_counts().values,palette=['blue','pink','green'])

for i,v in enumerate(data['Gender'].value\_counts()):

plt.text(i,v,str(v))

plt.title('Gender Count')

plt.xlabel('Gender')

plt.ylabel('Count')

plt.show()

Pie Chart:

country\_counts=data['Country\_Region'].value\_counts()

plt.figure(figsize=(5,5))

plt.pie(x=country\_counts.values,labels=country\_counts.index,autopct='%1.1f%%')

plt.title('country\_region dist')

plt.show()

**Statistical tests:**

1. **To check linearity :**

from scipy.stats import linregress

risk\_factors=['Genetic\_Risk','Air\_Pollution','Alcohol\_Use','Smoking','Obesity\_Level']

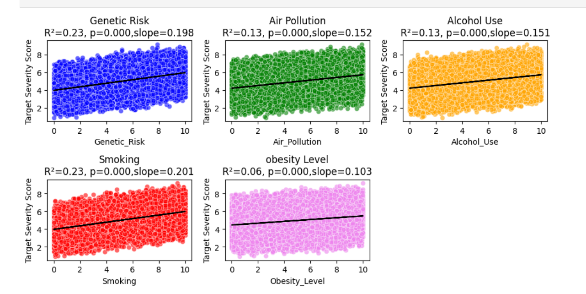
titles=['Genetic Risk','Air Pollution','Alcohol Use','Smoking','obesity Level']

colors = ['blue','green','orange','red','violet']

plt.figure(figsize=(15, 10))

for i, (factor, title,color) in enumerate(zip(risk\_factors,titles,colors),1):

plt.subplot(2, 3, i)

 X=data[factor]

y=data['Target\_Severity\_Score']

slope,intercept,r\_value,p\_value,std\_err =linregress(X,y)

sns.scatterplot(x=X, y=y, color=color, alpha=0.6)

# regression line

plt.plot(X, intercept + slope\*X, color="black")

# titles & labels

plt.title(f"{title}\nR²={r\_value\*\*2:.2f}, p={p\_value:.3f}, slope={slope:.3f}")

plt.xlabel(factor)

plt.ylabel("Target Severity Score")

plt.tight\_layout()

plt.show()

1. **To check the correlation among input & target variable.**

features=['Age', 'Genetic\_Risk','Air\_Pollution','Alcohol\_Use','Smoking','Obesity\_Level','Treatment\_Cost']

target=['Survival\_Years','Target\_Severity\_Score']

pearson\_res=pearson\_corr[target]

spearman\_res=spearman\_corr[target]

df\_corr= pd.concat([pearson\_res,spearman\_res],axis=1,keys=['pearson','spearman'])

df\_corr

1. **Feature Importance plot:**

It gives idea as after implementing a particular model which feature is contributing more.

importances = rf\_model.feature\_importances\_

1. Subplots example –

import matplotlib.pyplot as plt

import seaborn as sns

num\_col = ['salary','savings','loan amt']

plt.figure(figsize=(12,6)) # bigger canvas

for i, col in enumerate(num\_col):

plt.subplot(2, 3, i+1)

sns.histplot(df[col], kde=True)

plt.title(col)

plt.tight\_layout()

plt.show()