Name:Kaushik Kotian Roll No.:30 Div:D15B Batch:B

Experiment no.:3

AIM: Perform Data Modeling.

PROBLEM STATEMENT:

a. Partition the data set, for example 75% of the records are included in the training data set and 25% are included in the test data set.
b. Use a bar graph and other relevant graphs to confirm your proportions.
c. Identify the total number of records in the training data set.
d. Validate partition by performing a two-sample Z-test.

Theory:

Data Partitioning

Data partitioning is a critical step in the data preparation phase, especially in machine learning and statistical modeling. It involves dividing the dataset into separate sets to train models, validate their performance, and test them before deployment. The primary partitions are:

1. Training Set: This is the largest portion of the data, used to train the model. The model learns to make predictions or classifications based on this data.
2. Validation Set: This subset is used to fine-tune model parameters and select the best model version after training. It helps in avoiding overfitting to the training set by providing a separate dataset for model evaluation.
3. Test Set: This final subset is used to assess the model's performance after it has been trained and validated. The test set should only be used once, at the end of the modeling process, to provide an unbiased evaluation of the final model's performance.

Purpose of Data Partitioning

- Prevent Overfitting: By using separate data for training and validation, you ensure that the model can generalize well to new, unseen data.
- Model Selection: The validation set allows you to compare different models or configurations and select the best performer.

- Performance Estimation: The test set gives an unbiased estimate of how well the model is expected to perform on new data.

Hypothesis Testing

Hypothesis testing is a statistical method used to make decisions or infer conclusions about a population based on sample data. It involves:

- Null Hypothesis (H0): A statement indicating no effect or no difference. It's the hypothesis that the test aims to provide evidence against.
- Alternative Hypothesis (H1 or Ha): Contrasts the null hypothesis and represents what the researcher aims to support. It indicates the presence of an effect or a difference.

Steps in Hypothesis Testing

1. Formulate Hypotheses: Define the null and alternative hypotheses based on the research question.
2. Choose Significance Level (α): The probability of rejecting the null hypothesis when it is true. Common choices are 0.05 or 0.01.
3. Select Test Statistic: Depending on the data and the hypothesis, choose an appropriate statistical test (e.g., t-test, chi-square test).
4. Calculate Test Statistic and P-value: Perform the test to calculate the test statistic and the p-value, which indicates the probability of observing the test results under the null hypothesis.
5. Make a Decision: If the p-value is less than the chosen significance level, reject the null hypothesis in favor of the alternative. Otherwise, do not reject the null hypothesis.

Purpose of Hypothesis Testing

- Inferential Analysis: Helps in making inferences about populations based on sample data.
- Decision Making: Provides a systematic way to decide whether to support or refute a particular theory or assumption about a population parameter.

Implementation:

1.First of all the dataset is loaded and read using the pandas library of python. Then we can have some basic information and statistics about the dataset using df.describe(), df.info(), df.isnull().sum() where the dataset is stored in df.

```
[3]  import pandas as pd
     import matplotlib.pyplot as plt

     df=pd.read_csv('CVD_cleaned.csv')
```

```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 308854 entries, 0 to 308853
Data columns (total 19 columns):
 #   Column                        Non-Null Count   Dtype
---  ------                        --------------   -----
 0   General_Health                308854 non-null  object
 1   Checkup                       308854 non-null  object
 2   Exercise                      308854 non-null  object
 3   Heart_Disease                 308854 non-null  object
 4   Skin_Cancer                   308854 non-null  object
 5   Other_Cancer                  308854 non-null  object
 6   Depression                    308854 non-null  object
 7   Diabetes                      308854 non-null  object
 8   Arthritis                     308854 non-null  object
 9   Sex                           308854 non-null  object
 10  Age_Category                  308854 non-null  object
 11  Height_(cm)                   308854 non-null  float64
 12  Weight_(kg)                   308854 non-null  float64
 13  BMI                           308854 non-null  float64
 14  Smoking_History               308854 non-null  object
 15  Alcohol_Consumption           308854 non-null  float64
 16  Fruit_Consumption             308854 non-null  float64
 17  Green_Vegetables_Consumption  308854 non-null  float64
 18  FriedPotato_Consumption       308854 non-null  float64
dtypes: float64(7), object(12)
memory usage: 44.8+ MB
```

df=df.drop(columns=['Smoking_History','Alcohol_Consumption','Fruit_Consumption','Green_Vegetables_Consumption','FriedPotato_Consumption','Skin_Cancer','Other_Cancer','Arthritis'])
df.head()

	General_Health	Checkup	Exercise	Heart_Disease	Depression	Diabetes	Sex	Age_Category	Height_(cm)	Weight_(kg)	BMI
0	Poor	Within the past 2 years	No	No	No	No	Female	70-74	150.0	32.66	14.54
1	Very Good	Within the past year	No	Yes	No	Yes	Female	70-74	165.0	77.11	28.29
2	Very Good	Within the past year	Yes	No	No	Yes	Female	60-64	163.0	88.45	33.47
3	Poor	Within the past year	Yes	Yes	No	Yes	Male	75-79	180.0	93.44	28.73
4	Good	Within the past year	No	No	No	No	Male	80+	191.0	88.45	24.37

```
df.describe()
```

	Height_(cm)	Weight_(kg)	BMI
count	308854.000000	308854.000000	308854.000000
mean	170.615249	83.588655	28.626211
std	10.658026	21.343210	6.522323
min	91.000000	24.950000	12.020000
25%	163.000000	68.040000	24.210000
50%	170.000000	81.650000	27.440000
75%	178.000000	95.250000	31.850000
max	241.000000	293.020000	99.330000

```
[8] df['General_Health'].value_counts()
```

```
Very Good    110395
Good          95364
Excellent     55954
Fair          35810
Poor          11331
Name: General_Health, dtype: int64
```

```
df.isnull().sum()
```

```
General_Health    0
Checkup           0
Exercise          0
Heart_Disease     0
Depression        0
Diabetes          0
Sex               0
Age_Category      0
Height_(cm)       0
Weight_(kg)       0
BMI               0
dtype: int64
```

```
[10] from sklearn.preprocessing import LabelEncoder
     le =  LabelEncoder()
```

```
[11] df['General_Health'] = le.fit_transform(df['General_Health'])
     df.head()
```

	General_Health	Checkup	Exercise	Heart_Disease	Depression	Diabetes	Sex	Age_Category	Height_(cm)	Weight_(kg)	BMI
0	3	Within the past 2 years	No	No	No	No	Female	70-74	150.0	32.66	14.54
1	4	Within the past year	No	Yes	No	Yes	Female	70-74	165.0	77.11	28.29
2	4	Within the past year	Yes	No	No	Yes	Female	60-64	163.0	88.45	33.47
3	3	Within the past year	Yes	Yes	No	Yes	Male	75-79	180.0	93.44	28.73
4	2	Within the past year	No	No	No	No	Male	80+	191.0	88.45	24.37

2.Here, we've split the data in train and test sets where Y holds the column 'Outcome' while X holds the rest of the data columns. The data is randomly splitted in sizes of 75% and 25% for train and test respectively.

```
from sklearn.model_selection import train_test_split
X =df.drop(columns=['General_Health'])
Y=df['General_Health']
x_train, x_test, y_train, y_test = train_test_split(X,Y,test_size=0.25)
```

3.Looking at the train and test sets counts.

```
[13] print(x_train.shape)
     print(x_test.shape)

     (231640, 10)
     (77214, 10)
```

```
y_test.value_counts()

4    27534
2    23925
0    14166
1     8798
3     2791
Name: General_Health, dtype: int64
```

```
[15] y_train.value_counts()

     4    82861
     2    71439
     0    41788
     1    27012
     3     8540
     Name: General_Health, dtype: int64
```

4.To perform the z-test, first we import the z-test function from the statsmodel package.
Now we'll pass the train and test sets of Y to the z-test function to get two values. First one is the z-score and other o is the p value

```
[16] from statsmodels.stats.weightstats import ztest as ztest
     ztest(y_test, y_train, value=0)

     (-1.0343087123740289, 0.3009918651698058)
```

5. Next, we've declared a function where we pass the 'p' value to determine the result of our hypothesis testing. The function basically takes the 'p' value and checks if it is greater than equal to 0.05 or less than 0.05 (0.05 is the level of significance in z-test when not specified). If it is greater or equal to 0.05, then our null hypothesis is accepted, else it is rejected.

```
[26] def results(p):
       if (p['p_value']<0.05):p['hypothesis_accepted']='alternative'
       if (p['p_value']>=0.05):p['hypothesis_accepted']='null'

       df = pd.DataFrame(p, index=[''])
       cols = ['value1', 'value2', 'score', 'p_value', 'hypothesis_accepted']
       return df[cols]
```

6.Finally, we've declared 'p' as a dictionary where we've stored multiple values used to pass it to the 'results' function.

```
[27] p={}
     p['value1'], p['value2'] = y_train.mean(),y_test.mean()
     p['score'], p['p_value'] = ztest(y_train, y_test, alternative='two-sided')
     results(p)
```

	value1	value2	score	p_value	hypothesis_accepted
	2.274883	2.268462	1.034309	0.300992	null

Conclusion:

Data partitioning and hypothesis testing are foundational concepts in data science and statistics, serving different but complementary purposes. Data partitioning ensures that models are trained, validated, and tested effectively for reliable predictions. Hypothesis testing offers a rigorous framework for testing assumptions and making inferences about populations from sample data. Both are essential for robust data analysis, model building, and scientific research.