



## KALINGA INSTITUTE OF INDUSTRIAL TECHNOLOGY (KIIT)

(Deemed to be University)

### SCHOOL OF COMPUTER APPLICATIONS

SPRING SEMESTER 2025-26

Date: 29.01.2026

- |   |                  |
|---|------------------|
| <b>1. Course Code:</b> BCA 3004                               |                  |
| <b>2. Course title:</b> Information to Data Science (IDS) Lab | <b>Marks = 4</b> |

#### Assignment 4 (Descriptive Statistics)

**Title of the Assignment: Descriptive Statistics - Measures of Central Tendency and variability**

Perform the following operations on any open source dataset (e.g., data.csv)

1. Provide summary statistics (mean, median, minimum, maximum, standard deviation) for a dataset (age, income etc.) with numeric variables grouped by one of the qualitative (categorical) variables. For example, if your categorical variable is age groups and quantitative variable is income, then provide summary statistics of income grouped by the age groups. Create a list that contains a numeric value for each response to the categorical variable.
2. Write a Python program to display some basic statistical details like percentile, mean, standard deviation etc. of the species of 'Iris-setosa', 'Iris-versicolor' and 'Iris- versicolor' of iris.csv dataset.
3. Perform LinearRegression on your dataset. Transform a feature using the Box-Cox transformation and then apply LinearRegression again on your dataset. Finally compare both results.

Provide the codes with outputs and explain everything that you do in this step.

**Objective of the Assignment:** Students should be able to perform the Statistical operations and data transformation using Python on any open source dataset.

These are the sample codes which helps do perform different activities on your downloaded dataset. Kindly, learn from the sample python code and their outputs and, then apply then on your assignment.

#### Python Code:

##### 1. Mean

###### To find mean of all columns

Syntax:

```
df.mean()
```

Output:

CustomerID	100.50
Age	38.85
Annual Income (k\$)	60.56
Spending Score (1-100)	50.20
dtype: float64	

###### To find mean of specific column

Syntax:

```
df.loc[:, 'Age'].mean()
```

Output:

38.85

###### To find mean row wise

Syntax:

```
df.mean(axis=1) [0:4]
```

Output:

0	18.50
1	29.75
2	11.25
3	30.00
dtype: float64	

## 2. Median

### To find median of all columns

Syntax:

```
df.median()
```

Output:

```
CustomerID      100.5
Age             36.0
Annual Income (k$) 61.5
Spending Score (1-100) 50.0
dtype: float64
```

### To find median of specific column

Syntax:

```
df.loc[:, 'Age'].median()
```

Output:

```
36.0
```

### To find median row wise

Syntax:

```
df.median(axis=1)[0:4]
```

Output:

```
0    17.0
1    18.0
2    11.0
3    19.5
dtype: float64
```

## 3. Mode

### To find mode of all columns

Syntax:

```
df.mode()
```

Output:

	CustomerID	Genre	Age	Annual Income (k\$)	Spending Score (1-100)
0	1	Female	32.0	54.0	42.0
1	2	NaN	NaN	78.0	NaN
2	3	NaN	NaN	NaN	NaN
3	4	NaN	NaN	NaN	NaN
4	5	NaN	NaN	NaN	NaN
...	...	...	...	...	...
195	196	NaN	NaN	NaN	NaN
196	197	NaN	NaN	NaN	NaN
197	198	NaN	NaN	NaN	NaN
198	199	NaN	NaN	NaN	NaN
199	200	NaN	NaN	NaN	NaN

200 rows × 5 columns

In the Genre Column mode is Female, for column Age mode is 32 etc. If a particular column does not have mode all the values will be displayed in the column.

#### To find the mode of a specific column.

Syntax:

```
df.loc[:, 'Age'].mode()
```

Output:

32

#### 4. Minimum

##### To find median of all columns

Syntax:

```
df.min()
```

Output:

CustomerID	1
Genre	Female
Age	18
Annual Income (k\$)	15
Spending Score (1-100)	1
dtype: object	

##### To find median of Specific column

Syntax:

```
df.loc[:, 'Age'].min(skipna = False)
```

Output:

## 5. Maximum

### To find median of all columns

Syntax:

```
df.max()
```

Output:

```
CustomerID      200
Genre           Male
Age            70
Annual Income (k$)    137
Spending Score (1-100) 99
dtype: object
```

### To find median of Specific column

Syntax:

```
df.loc[:, 'Age'].min(skipna = False)
```

Output:

18

## 6. Standard Deviation

### To find Standard Deviation of all columns

Syntax:

```
df.std()
```

Output:

```
CustomerID      57.879185
Age            13.969007
Annual Income (k$)    26.264721
Spending Score (1-100) 25.823522
dtype: float64
```

### To find Standard Deviation of specific column

Syntax:

```
df.loc[:, 'Age'].std()
```

Output:

13.969007331558883

#### To find Standard Deviation row wise

Syntax:

```
df.std(axis=1) [0:4]
```

Output:

```
0    15.695010
1    35.074920
2     8.057088
3    32.300671
dtype: float64
```

### 3. Summary statistics of income grouped by the age groups

**Problem Statement:** For example, if your categorical variable is age groups and quantitative variable is income, then provide summary statistics of income grouped by the age groups. Create a list that contains a numeric value for each response to the categorical variable.

Categorical Variable: Genre

Quantitative Variable : Age

Syntax:

```
df.groupby(['Genre']) ['Age'].mean()
```

Output:

```
Genre
Female    38.098214
Male      39.806818
Name: Age, dtype: float64
```

Categorical Variable: Genre

Quantitative Variable : Income

Syntax:

```
df_u=df.rename(columns= {'Annual Income  
k$':'Income'},inplace=False)
```

```
(df_u.groupby(['Genre']).Income.mean())
```

Output:

Genre
Female 59.250000
Male 62.227273
Name: Income, dtype: float64

To create a list that contains a numeric value for each response to the categorical variable.

```
from sklearn import preprocessing  
enc = preprocessing.OneHotEncoder()  
enc_df = pd.DataFrame(enc.fit_transform(df[['Genre']]).toarray())  
enc_df
```

	0	1
0	0.0	1.0
1	0.0	1.0
2	1.0	0.0
3	1.0	0.0
4	1.0	0.0

To concat numerical list to dataframe

```
df_encode =df_u.join(enc_df)  
df_encode
```

	CustomerID	Genre	Age	Income	Spending Score (1-100)	0	1
0	1	Male	19	15	39	0.0	1.0
1	2	Male	21	15	81	0.0	1.0
2	3	Female	20	16	6	1.0	0.0
3	4	Female	23	16	77	1.0	0.0
4	5	Female	31	17	40	1.0	0.0
...	...	...	...	...	...	...	...
195	196	Female	35	120	79	1.0	0.0
196	197	Female	45	126	28	1.0	0.0
197	198	Male	32	126	74	0.0	1.0
198	199	Male	32	137	18	0.0	1.0
199	200	Male	30	137	83	0.0	1.0
200 rows × 7 columns							

#### 4. Display basic statistical details on the iris dataset.

##### Algorithm:

1. Import Pandas Library

2. The dataset is downloaded from UCI repository.

```
csv_url = 'https://archive.ics.uci.edu/ml/machine-learning-databases/iris/iris.data'
```

3. Assign Column names

```
col_names =
```

```
['Sepal_Length', 'Sepal_Width', 'Petal_Length', 'Petal_Width', 'Species']
```

4. Load Iris.csv into a Pandas data frame

```
iris = pd.read_csv(csv_url, names = col_names)
```

5. Load all rows with Iris-setosa species in variable irisSet

```
irisSet = (iris['Species'] == 'Iris-setosa')
```

6. To display basic statistical details like percentile, mean, standard deviation etc. for Iris-setosa use describe

```
print('Iris-setosa')
```

```
print(iris[irisSet].describe())
```

7. Load all rows with Iris-versicolor species in variable irisVer

```
irisVer = (iris['Species']== 'Iris-versicolor')
```

8. To display basic statistical details like percentile, mean,standard deviation etc. for Iris-versicolor use describe

```
print('Iris-versicolor')  
print(iris[irisVer].describe())
```

9. Load all rows with Iris-virginica species in variable irisVir

```
irisVir = (iris['Species']== 'Iris-virginica')
```

10. To display basic statistical details like percentile, mean,standard deviation etc. for Iris-virginica use describe

```
print('Iris-virginica')  
print(iris[irisVir].describe())
```

Iris-setosa				
	Sepal_Length	Sepal_Width	Petal_Length	Petal_Width
count	50.00000	50.000000	50.000000	50.00000
mean	5.00600	3.418000	1.464000	0.24400
std	0.35249	0.381024	0.173511	0.10721
min	4.30000	2.300000	1.000000	0.10000
25%	4.80000	3.125000	1.400000	0.20000
50%	5.00000	3.400000	1.500000	0.20000
75%	5.20000	3.675000	1.575000	0.30000
max	5.80000	4.400000	1.900000	0.60000
Iris-versicolor				
	Sepal_Length	Sepal_Width	Petal_Length	Petal_Width
count	50.00000	50.000000	50.000000	50.00000
mean	5.936000	2.770000	4.260000	1.326000
std	0.516171	0.313798	0.469911	0.197753
min	4.900000	2.000000	3.000000	1.000000
25%	5.600000	2.525000	4.000000	1.200000
50%	5.900000	2.800000	4.350000	1.300000
75%	6.300000	3.000000	4.600000	1.500000
max	7.000000	3.400000	5.100000	1.800000
Iris-virginica				
	Sepal_Length	Sepal_Width	Petal_Length	Petal_Width
count	50.00000	50.000000	50.000000	50.00000
mean	6.58800	2.974000	5.552000	2.02600
std	0.63588	0.322497	0.551895	0.27465
min	4.90000	2.200000	4.500000	1.40000
25%	6.22500	2.800000	5.100000	1.80000
50%	6.50000	3.000000	5.550000	2.00000
75%	6.90000	3.175000	5.875000	2.30000
max	7.90000	3.800000	6.900000	2.50000

### **Step-1: Import necessary Dependencies**

```
import numpy as np
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt
import scipy.stats as stats
```

### **Step-2: Import useful packages**

```
from sklearn.model_selection import train_test_split
from sklearn.model_selection import cross_val_score
from sklearn.linear_model import LinearRegression
from sklearn.metrics import r2_score
from sklearn.preprocessing import PowerTransformer
```

### **Step-3: Read and Load your dataset**

```
df = pd.read_csv('your dataset.csv')
df.head()
```

### **Step-4: Find the number of missing values per column**

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

### **Step-5: Finding Statistical measures for columns**

```
df.describe()
```

### **Step-6: For example: Separate independent and dependent variables for your dataset**

```
X = df.iloc[:, :8]
y = df.iloc[:, -1]
```

### **Step-7: Split our dataset into train and test subsets**

```
X_train, X_test, y_train, y_test =
train_test_split(X,y,test_size=0.33,random_state=105)
```

### **Step-8: Train our Linear Regression model and check the metric**

```
lr = LinearRegression()
lr.fit(X_train,y_train)
y_pred = lr.predict(X_test)
print(r2_score(y_test,y_pred))
```

### **Step-9: Plotting the distplots without any transformation**

```
import warnings
warnings.filterwarnings('ignore')
for col in X_train.columns:
    plt.figure(figsize=(14,4))
    plt.subplot(121)
    sns.distplot(X_train[col])
```

```

plt.title(col)
plt.subplot(122)
stats.probplot(X_train[col], dist="norm", plot=plt)
plt.title(col)
plt.show()

```

#### **Step-10: Apply the Box-Cox transformation**

```

pt = PowerTransformer(method='box-cox')
X_train_transformed = pt.fit_transform(X_train+0.0000001)
X_test_transformed = pt.transform(X_test+0.0000001)
pd.DataFrame({'cols':X_train.columns,'box_cox_lambdas':pt.lambdas_})

```

#### **Step-11: Train our model on transformed data and check the metric**

```

lr = LinearRegression()
lr.fit(X_train_transformed,y_train)
y_pred2 = lr.predict(X_test_transformed)

print(r2_score(y_test,y_pred2))

```

#### **Step-12: Plotting the distplots after transformation**

```

X_train_transformed = pd.DataFrame(X_train_transformed,columns=X_train.columns)
for col in X_train_transformed.columns:
    plt.figure(figsize=(14,4))
    plt.subplot(121)
    sns.distplot(X_train[col])
    plt.title(col)
    plt.subplot(122)
    sns.distplot(X_train_transformed[col])
    plt.title(col)
    plt.show()

```

**Conclusion:** Here we implement the **Box-Cox** transformation but by changing the parameters inside the function you can implement **Yeo-Johnson** Transformation also.

- The idea behind running the **describe()** function is to check the values present in the columns and verify the assumptions of Power Transformation i.e, Box-Cox transformation only accepts strictly positive numbers.
- We also observe that there is an increment in the accuracy of the model, since our problem statement is a “**Regression**” Problem statement and we apply the linear regression, and by transformations, we make the columns closer to a normal distribution, which satisfies the assumptions of the linear regression algorithm.
- We add a very small value to all the points of the dataset so that no point value remains exactly zero and our assumption still holds for Box-Cox transformation.

### **Viva Questions:**

1. Explain Measures of Central Tendency with examples.
2. What are the different types of variables. Explain with examples.
3. Which method is used to statistic the dataframe? write the code.