



## **Mathematics for Data Science**

**Tutor:** Mr.Muneer Ahmad

**Student:** Mohd Iftequar Ahmed Farooqui

**Student ID:** MOH23622649

**Topic:** Fundamentals of Descriptive Statistics and Probability Distributions

**Mathematics for Data Science**

**MSc in Data Science**

**15-02-2024**

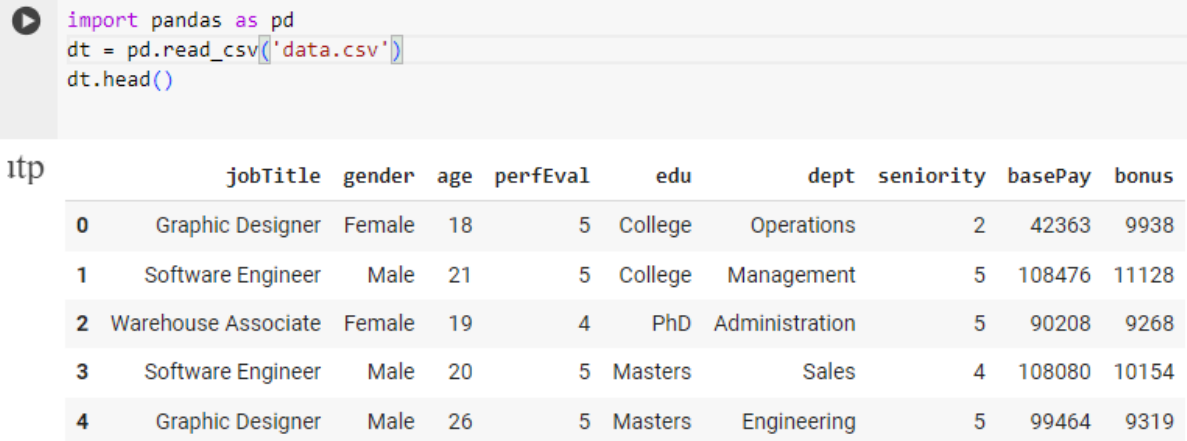
Table of Contents:

<b>1. Measure of Central Tendency</b>	
• 1.1 Mean	3 - 4
• 1.2 Median	4 - 5
• 1.3 Mode	5
<b>2. Measure of Spread/Dispersion</b>	
• 2.1 Range	6
• 2.2 Variance	6 - 7
• 2.3 Standard Deviation	7 - 8
<b>3. Probability Mass Function (PMF)</b>	8 - 9
<b>4. Probability Density Function (PDF)</b>	10 - 11

## Measure of Central Tendency

### Mean Theory:

Importing Pandas Library and Visualising Data in Fig: 1.



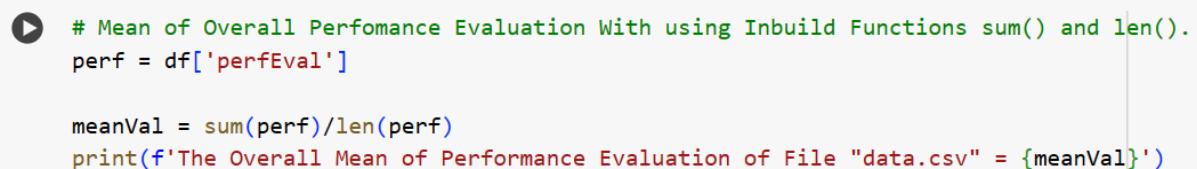
```
import pandas as pd
dt = pd.read_csv('data.csv')
dt.head()
```

	jobTitle	gender	age	perfEval	edu	dept	seniority	basePay	bonus
0	Graphic Designer	Female	18	5	College	Operations	2	42363	9938
1	Software Engineer	Male	21	5	College	Management	5	108476	11128
2	Warehouse Associate	Female	19	4	PhD	Administration	5	90208	9268
3	Software Engineer	Male	20	5	Masters	Sales	4	108080	10154
4	Graphic Designer	Male	26	5	Masters	Engineering	5	99464	9319

Fig:1

Here I calculate Overall Mean for Performance Evaluation as well as for Particular Job sectors from file "data.csv".

Here the column Performance Evaluation is a set 1000 values with zero null values within a range from 1 to 5.



```
# Mean of Overall Perfomance Evaluation With using Inbuild Functions sum() and len().
perf = df['perfEval']

meanVal = sum(perf)/len(perf)
print(f'The Overall Mean of Performance Evaluation of File "data.csv" = {meanVal}')
```

Fig: 1.1.1

The Overall mean for Performance Evaluation is = 3.037 which is almost 3, as we can see in Fig: 1.1.1. To Check whether which sectors Performing Good and Bad. I Calculate the Mean for Particular Job Sectors. So, that those Sectors have less Performance we can have more focus on Job Sectors who have Less Average. To Improve the Overall Performance of the Company.

```

▶ # Calculate Mean of Each Field Separately
job_titles = list(set(df['jobTitle']))
for job_title in job_titles:
    vals = df[df['jobTitle'] == job_title]['perfEval']
    mean_val = sum(vals)/len(vals)
    print(f'Mean of {job_title} = {mean_val}')

```

```

itp Mean of IT = 3.125
Mean of Warehouse Associate = 3.2444444444444445
Mean of Sales Associate = 2.797872340425532
Mean of Data Scientist = 2.97196261682243
Mean of Software Engineer = 3.128440366972477
Mean of Driver = 3.065934065934066
Mean of Marketing Associate = 2.9237288135593222
Mean of Graphic Designer = 3.122448979591837
Mean of Financial Analyst = 3.2710280373831777
Mean of Manager = 2.7

```

Fig: 1.1.2

In Fig: 1.1.2, we can see Mean's for both Sectors is near to Overall Mean. So, Organization must work on Each and Every Sector. But more over on Manager.

### Median Theory:

```

▶ # median for particular column
perf = df['perfEval']
def median(column):
    sorted_column = sorted(column)
    mid_val_index = len(column)//2
    if((len(column))%2 == 0):
        median_val = (sorted_column[mid_val_index - 1] + sorted_column[mid_val_index])/2
    return median_val
    return sorted_column[mid_val_index]

median(perf)

```

```
itp 3.0
```

Fig: 1.2.1

Here I calculate Median of Overall Performance Evaluation from File "data.csv" in Fig: 1.2.1., The Overall Median Performance Evaluation = 3.0, It is Similar to Overall Mean = 3.037 which means there is not many Outliers in Data set. To Overcome with the problem of Outliers, it's better to use Median rather than Mean.

```

▶ # Calculate Median of Each Field Separately
job_titles = list(set(df['jobTitle']))
for job_title in job_titles:
    vals = df[df['jobTitle'] == job_title]['perfEval']
    print(f'Median of {job_title} = {median(vals)}')

```

```

itp Median of IT = 3.5
     Median of Warehouse Associate = 2.5
     Median of Sales Associate = 3.0
     Median of Data Scientist = 3
     Median of Software Engineer = 3
     Median of Driver = 4
     Median of Marketing Associate = 3.0
     Median of Graphic Designer = 4.0
     Median of Financial Analyst = 4
     Median of Manager = 3.0

```

Fig: 1.2.3

Now I Calculate the Median for Particular Sector in Fig: 1.2.3., we can see that the median value for particular Sector is Similar to Mean of Particular Sector Except Financial Analyst. Which means there is outliers in Financial Analyst.

### Mode Theory:

```

▶ # Mode in a Column
perf = df['perfEval']
def mode(values):
    counts = {}
    for s in values:
        if(s in counts):
            counts[s] = counts[s] + 1
        else:
            counts[s] = 1
    max_count = max(counts.values())
    modes = [v for v in set(values) if counts[v] == max_count]
    return modes
print(mode(perf))

```

```
itp [5]
```

Fig: 1.3.1

Here I Calculate Mode of Overall Performance Evaluation = 5 in Fig: 1.3.1, which means the Repetition of Value 5 is High when compared to each data point. When Compare Overall Mode = 5 to Overall, Mean = 3.037 and Median = 3.0 Mode is Vary or Different.

The Measures of Central Tendency for the Data "data.csv" is similar but Mode is Different. This Measure of Central Tendency will be helps to Find Variance, Standard Deviation, and in Distribution Process.

## Measure of Spread/Dispersion

### Range Theory:

Here I visualise the Data to Understand in Fig: 2.

```
import pandas as pd
dt = pd.read_csv('HRDataset_v14.csv')
dt.head()
```

✓ 0.0s

	Employee_Name	EmpID	MarriedID	MaritalStatusID	GenderID	EmpStatusID	DeptID	PerfScoreID	FromDiversityJobFairID	Salary	...
0	Adinolfi, Wilson K	10026	0	0	1	1	5	4	0	62506	...
1	Ait Sidi, Karthikeyan	10084	1	1	1	5	3	3	0	104437	...
2	Akinkuolie, Sarah	10196	1	1	0	5	5	3	0	64955	...
3	Alagbe, Trina	10088	1	1	0	1	5	3	0	64991	...
4	Anderson, Carol	10069	0	2	0	5	5	3	0	50825	...

5 rows × 36 columns

Fig: 2

I calculate the Range of Salaries of each Department from the file "HRDataset\_v14.csv". To get the Rough Idea About, How the salaries had distributed among the Employees in Terms of Range from minimum to maximum in each Individual Sector.

```
# Range to get Rough Idea about Spread of Data
depts = list(set(dt['Department']))
def range_dt(column):
    return max(column) - min(column)

for dept in depts:
    salaries = dt[dt['Department'] == dept]['Salary']
    print(f'{dept} = {range_dt(salaries)}')
```

✓ 0.0s

```
Sales = 124125
IT/IS = 170272
Production = 125454
Admin Offices = 56447
Software Engineering = 31295
Executive Office = 0
```

Fig: 2.1.1

In Fig: 2.1.1, we can see the range Values of Individual Department. This measure provides a quick overview of the spread of salary values of each Employee within the dataset, with a larger range suggesting greater variability in compensation levels.

### Variance Theory:

To know the Variability between Salaries and Dispersion of Salaries by using the Measure of Central Tendency Mean.

Here, I calculate the Variance of Salaries of each Department from the same file.

```
def variance(column):
    mean = sum(column)/len(column)
    variance_val = sum((val-mean)**2 for val in column)/len(column)
    return variance_val

for dept in depts:
    salaries = dt[dt['Department'] == dept]['Salary']
    print(f'{dept} = {variance(salaries)}')
```

✓ 0.0s

```
Software Engineering = 83198210.24793388
Sales = 437876293.73985434
Production          = 129852796.58286211
IT/IS = 1080820774.9903998
Admin Offices = 418815713.65432096
Executive Office = 0.0
```

Fig: 2.2

In Fig: 2.2, The Variance of Salaries of each Department has high value, Except Executive Office Department, which means there is no difference between salaries of Employees in Executive Office Department.

The Higher Variance of each Department Shows that there is huge difference in Salaries of Employees of Same Department.

### Standard Deviation:

Method to Know the most near Salaries of Employees to Mean or Average is Standard Deviation. It gives Spread of Data which is Tightly near to Average.

Here, I calculate the Standard deviation of Salaries of each Department from the same file.

```
# Spread of Salaries tightly near to Mean.
def std(column):
    return variance(column)**0.5

for dept in depts:
    salaries = dt[dt['Department'] == dept]['Salary']
    print(f'{dept} = {std(salaries)}')
```

✓ 0.0s

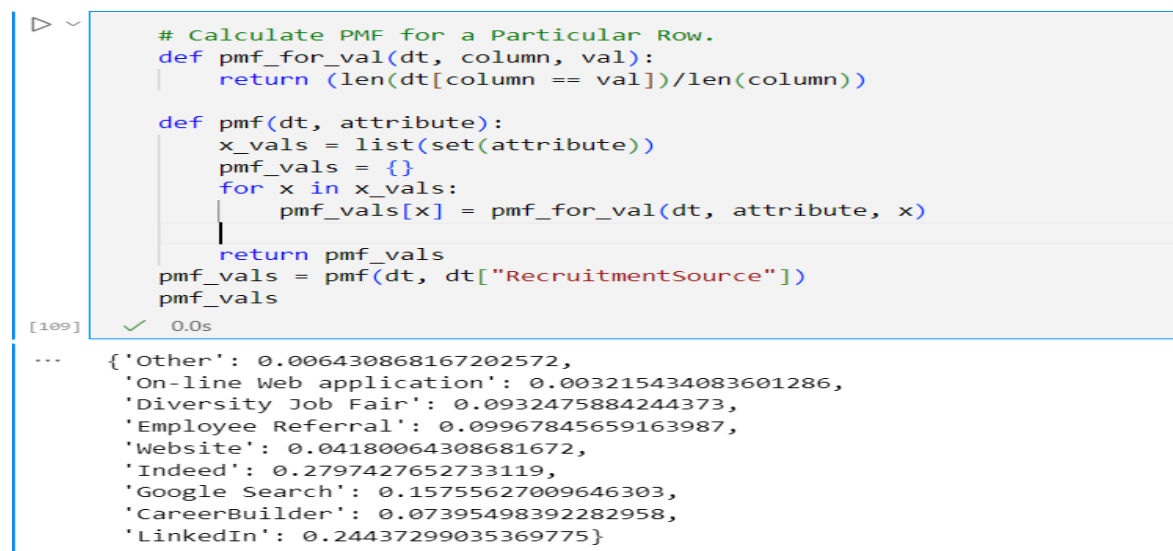
```
Software Engineering = 9121.305292990355
Sales = 20925.49387087087
Production          = 11395.297125694535
IT/IS = 32875.83877242373
Admin Offices = 20464.987506820544
Executive Office = 0.0
```

Fig: 2.3

In Fig: 2.3, The Standard Deviation of Salaries of each Department has Calculated, which states that the spread of Salaries most near to Mean or Average.

## Probability Mass Function

In Fig: 3.1, I Calculate Probability Mass Function (PMF) of Recruitment Sources of same Data which I used for Measure of Dispersion HRDataset\_v14.csv.



```
# Calculate PMF for a Particular Row.
def pmf_for_val(dt, column, val):
    return (len(dt[column == val])/len(column))

def pmf(dt, attribute):
    x_vals = list(set(attribute))
    pmf_vals = {}
    for x in x_vals:
        pmf_vals[x] = pmf_for_val(dt, attribute, x)
    return pmf_vals
pmf_vals = pmf(dt, dt["RecruitmentSource"])
pmf_vals
```

```
[109] ✓ 0.0s
```

```
... {'Other': 0.006430868167202572,
      'On-line Web application': 0.003215434083601286,
      'Diversity Job Fair': 0.0932475884244373,
      'Employee Referral': 0.09967845659163987,
      'Website': 0.04180064308681672,
      'Indeed': 0.2797427652733119,
      'Google Search': 0.15755627009646303,
      'CareerBuilder': 0.07395498392282958,
      'LinkedIn': 0.24437299035369775}
```

Fig: 3.1

In the above figure, I display the PMF values of Particular Recruitment Source in the form of Dictionary.

HR can perform Multiple tasks and analyses based on this data to get Optimized Recruitment Strategy.

Such as this PMF values helps HR to Identify which Recruitment Sources are most Effective in Attracting Candidates.

So, that they can Invest more on such Sources with higher Probabilities and can reduce investment for Less Effective Sources.

HR can select Cost-effective Source with the Help of PMF. They can increase their Recruitment Strategies of higher Probability Sources.

Using historical PMF data, HR can forecast future applicant distributions for different Recruitment Sources. This can help in making long-term strategic planning.

In Summary, HR can make decisions to Improve Recruitment outcomes, Optimize Resource Allocation, and can enhance the Overall Effectiveness of the Hiring process with the help of PMF values.

The Visualization of the PMF values is shown in Fig: 3.2.



```
import matplotlib.pyplot as plt

plt.figure(figsize=(12,5))
plt.bar(pmf_vals.keys(), pmf_vals.values(), alpha = 1)
plt.title('PMF of Recruitment Source')
plt.xlabel('Recruitment Sources')
plt.ylabel('PMF Values')
plt.xticks(rotation=45)
```

✓ 0.6s

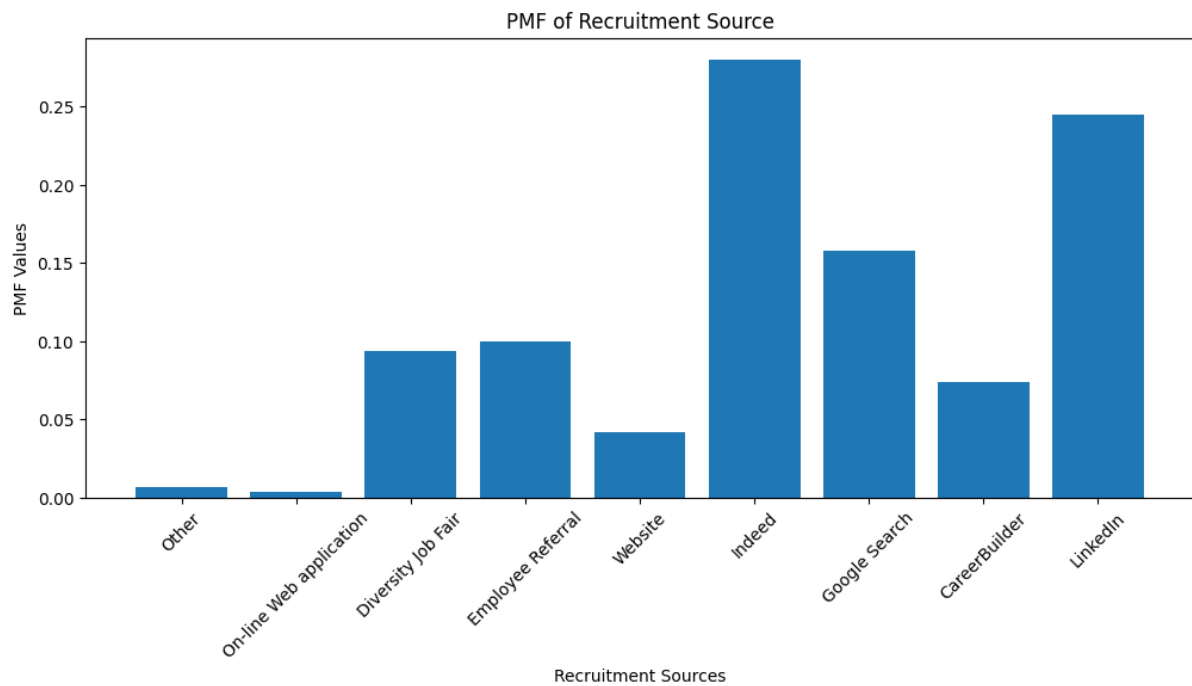


Fig:3.2

## Probability Density Function

In Fig: 4.1, I Calculate Probability Density Function (PDF) of salaries from and HRDataset\_v14.csv can helpful for various Tasks and Decision-making.

```
# Calculate PDF for Column Salary from Same Dataset by Using Gaussian KDE.
import numpy as np
import matplotlib.pyplot as plt
from scipy.stats import gaussian_kde

salary = data['Salary']

min_value = salary.min()
max_value = salary.max()

range_values = np.linspace(min_value, max_value, 1000)

kde = gaussian_kde(salary)
kde_values = kde(range_values)

plt.figure(figsize=(8,3))
plt.plot(range_values, kde_values)
plt.xlabel('Salary')
plt.ylabel('Density')
plt.title('Kernel Density Estimate (KDE) for Salary')
plt.show()
```

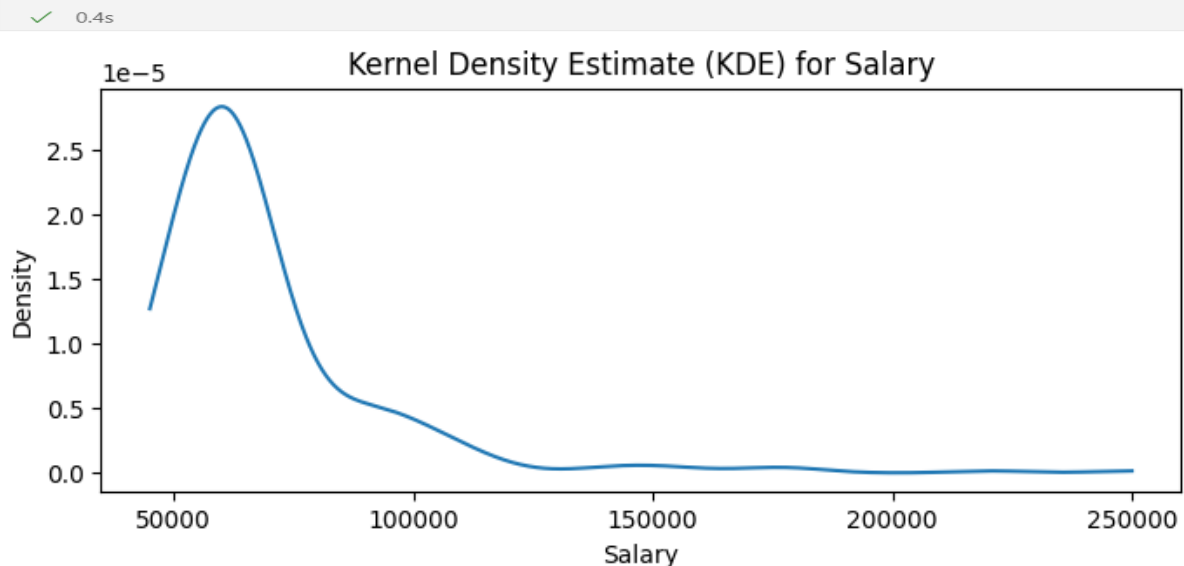


Fig: 4.1

For Example, HR can analyse the Distribution of salaries Among the Employees. By visualizing the PDF, HR can identify the Central Tendency, Spread, and Shape of the Salary Distribution. This can help in setting competitive salary, identifying Outliers, and Compensation practices.

HR can analyse the PDF to identify Salary levels at which Employee Salary is High or Low in Terms of Investment. This information helps to develop Strategies, Such as Adjusting Compensation, Offering Additional benefits, or Providing Opportunities to check Employee Talent.

HR can use Historical salary Distribution data in the PDF to forecast Future Salary Budgets. This helps to Design Incentive Programs, Reward Employees Effectively, Bonus Structures, and Salary Adjustment based on Salary Distribution PDF.

The PDF Distribution also can help to make decisions in Layoffs Duration, HR can compare the outliers Employees (High Salaries Employees) Performance with Low Salaries Employees Performance. If they are same, can Remove Outliers.

In Fig: 4.2, I Code a function to Calculate Probability between Range of Two Values.

```
# Calculate Probability in between Range of Salaries.

from scipy.stats import norm

def z_val(num, mean, std):
    return (num - mean)/std

def t_val(num1, num2, column):
    z_val1 = z_val(num1, column.mean(), column.std())
    z_val2 = z_val(num2, column.mean(), column.std())
    t_val1 = norm.cdf(z_val1, 0, 1)
    t_val2 = norm.cdf(z_val2, 0, 1)
    return t_val2 - t_val1

x1 = 100000
x2 = 250000

print(f"CDF between {x2} and {x1} = {t_val(x1, x2, salary)}")
```

✓ 0.0s

CDF between 250000 and 100000 = 0.10907600619111424

This Function converts the salaries into Standard Normal Values called Z values, then calculates T-table values of the Z-values.

# Thank-you.