213208MuteebLabTask3

February 28, 2025

[1]:
```
import pandas as pd
import numpy as np

# Load the dataset
file_path = "employee_data.xlsx"
df = pd.read_excel(file_path)
```

[2]:
```
# Problem 1
# I. Identify missing values
```

[3]:
```
missing_values = df.isnull().sum()
print("Missing Values:\n", missing_values)
```

```
Missing Values:
 Employee ID           0
Age                   0
Salary ($)            1
Experience (Years)    0
Performance Score     1
Department            0
dtype: int64
```

[4]:
```
# II. Mean and Median Imputation
```

[5]:
```
mean_salary = df['Salary ($)'].mean()
median_salary = df['Salary ($)'].median()
mean_perf = df['Performance Score'].mean()
median_perf = df['Performance Score'].median()
```

[7]:
```
# Creating deep copies for mean and median imputation
df_mean_imputed = df.copy(deep=True)
df_median_imputed = df.copy(deep=True)

df_mean_imputed['Salary ($)'] = df_mean_imputed['Salary ($)'].
  ↪fillna(mean_salary)
df_mean_imputed['Performance Score'] = df_mean_imputed['Performance Score'].
  ↪fillna(mean_perf)
```

```
df_median_imputed['Salary ($)'] = df_median_imputed['Salary ($)'].
    ↪fillna(median_salary)
df_median_imputed['Performance Score'] = df_median_imputed['Performance Score'].
    ↪fillna(median_perf)

print("\nMean Imputation:\n", df_mean_imputed)
print("\nMedian Imputation:\n", df_median_imputed)
```

```
Mean Imputation:
    Employee ID  Age      Salary ($)  Experience (Years)  Performance Score  \
0          101   22   25000.000000                   1           3.200000
1          102   28   28000.000000                   3           3.800000
2          103   30   59444.444444                   5           4.000000
3          104   35   40000.000000                   7           4.500000
4          105   40   42000.000000                  10           4.700000
5          106   45   45000.000000                  12           5.000000
6          107   50   60000.000000                  15           3.900000
7          108   60   85000.000000                  20           3.966667
8          109  100   90000.000000                  30           4.100000
9          110  150  120000.000000                  35           2.500000

  Department
0         HR
1         IT
2    Finance
3         IT
4         HR
5    Finance
6         IT
7         HR
8    Finance
9         IT

Median Imputation:
    Employee ID  Age  Salary ($)  Experience (Years)  Performance Score  \
0          101   22     25000.0                   1                3.2
1          102   28     28000.0                   3                3.8
2          103   30     45000.0                   5                4.0
3          104   35     40000.0                   7                4.5
4          105   40     42000.0                  10                4.7
5          106   45     45000.0                  12                5.0
6          107   50     60000.0                  15                3.9
7          108   60     85000.0                  20                4.0
8          109  100     90000.0                  30                4.1
9          110  150    120000.0                  35                2.5
```

```
  Department
0         HR
1         IT
2    Finance
3         IT
4         HR
5    Finance
6         IT
7         HR
8    Finance
9         IT
```

[8]:
```
# III. Comparison: Mean vs Median Imputation
salary_diff = df_mean_imputed['Salary ($)'] - df_median_imputed['Salary ($)']
perf_diff = df_mean_imputed['Performance Score'] -␣
 ↪df_median_imputed['Performance Score']
print("\nSalary Difference (Mean - Median):\n", salary_diff)
print("\nPerformance Score Difference (Mean - Median):\n", perf_diff)
```

```
Salary Difference (Mean - Median):
 0        0.000000
1        0.000000
2    14444.444444
3        0.000000
4        0.000000
5        0.000000
6        0.000000
7        0.000000
8        0.000000
9        0.000000
Name: Salary ($), dtype: float64

Performance Score Difference (Mean - Median):
 0    0.000000
1    0.000000
2    0.000000
3    0.000000
4    0.000000
5    0.000000
6    0.000000
7   -0.033333
8    0.000000
9    0.000000
Name: Performance Score, dtype: float64
```

[9]:
```
# Problem 2: Variance Calculation
```

[10]:
```
var_age = df['Age'].var()
var_salary = df['Salary ($)'].var()
var_experience = df['Experience (Years)'].var()
print("\nVariance:\nAge:", var_age, "\nSalary:", var_salary, "\nExperience:",␣
 ↪var_experience)
```

```
Variance:
Age: 1584.2222222222222
Salary: 1040027777.7777779
Experience: 130.4
```

[11]:
```
# Identifying the column with the highest variance
variance_dict = {"Age": var_age, "Salary ($)": var_salary, "Experience (Years)":
 ↪ var_experience}
highest_var_column = max(variance_dict, key=variance_dict.get)
print(f"\nThe column with the highest variance is {highest_var_column},␣
 ↪indicating it has the most spread out data.")
```

```
The column with the highest variance is Salary ($), indicating it has the most
spread out data.
```

[12]:
```
# Problem 3: Pearson Correlation
```

[14]:
```
corr_age_salary = df[['Age', 'Salary ($)']].corr(method='pearson').iloc[0, 1]
corr_exp_salary = df[['Experience (Years)', 'Salary ($)']].
 ↪corr(method='pearson').iloc[0, 1]
corr_perf_salary = df[['Performance Score', 'Salary ($)']].
 ↪corr(method='pearson').iloc[0, 1]

print("\nPearson Correlation:\nAge & Salary:", corr_age_salary, "\nExperience &␣
 ↪Salary:", corr_exp_salary, "\nPerformance Score & Salary:", corr_perf_salary)
```

```
Pearson Correlation:
Age & Salary: 0.9464657959418248
Experience & Salary: 0.9783194249518953
Performance Score & Salary: -0.4847665607139448
```

[15]:
```
# Problem 4: Interpretations
```

[16]:
```
strongest_corr = max(abs(corr_age_salary), abs(corr_exp_salary),␣
 ↪abs(corr_perf_salary))
print("\nThe strongest correlation is:", strongest_corr)
```

```
negative_correlations = [
    ("Age & Salary", corr_age_salary),
    ("Experience & Salary", corr_exp_salary),
    ("Performance Score & Salary", corr_perf_salary)
]
negative_corrs = [pair for pair in negative_correlations if pair[1] < 0]
print("\nNegative Correlations:", negative_corrs)
```

```
The strongest correlation is: 0.9783194249518953

Negative Correlations: [('Performance Score & Salary', -0.4847665607139448)]
```

[17]:
```
# Explanation of variance
print("\nA high variance means the data is widely spread out, indicating more␣
↪diversity in the values.")
```

```
A high variance means the data is widely spread out, indicating more diversity
in the values.
```

[18]:
```
# Explanation of high correlation for decision-making
print("\nIf two variables have a high positive correlation, one can predict the␣
↪other, useful for salary predictions, hiring decisions, etc.")
```

```
If two variables have a high positive correlation, one can predict the other,
useful for salary predictions, hiring decisions, etc.
```

[19]:
```
# Impact of missing values on correlation
print("\nMissing values reduce sample size and distort correlation calculations.
↪")
```

```
Missing values reduce sample size and distort correlation calculations.
```

[20]:
```
# Recommended preprocessing technique
print("\nNormalization and outlier handling improve correlation analysis by␣
↪reducing distortions in data distribution.")
```

```
Normalization and outlier handling improve correlation analysis by reducing
distortions in data distribution.
```

[ ]:
```

```