

EDA and Preprocessing Machine Learning Assignment II

Objective:

The main objective of this project is to design and implement a robust data preprocessing system that addresses common challenges such as missing values, outliers, inconsistent formatting, and noise. By performing effective data preprocessing, the project aims to enhance the quality, reliability, and usefulness of the data for machine learning.

```
In [40]: import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
import warnings
warnings.filterwarnings("ignore")
```

```
In [41]: df=pd.read_csv('Employee.csv')
df.head()
```

```
Out[41]:
```

	Company	Age	Salary	Place	Country	Gender
0	TCS	20.0	NaN	Chennai	India	0
1	Infosys	30.0	NaN	Mumbai	India	0
2	TCS	35.0	2300.0	Calcutta	India	0
3	Infosys	40.0	3000.0	Delhi	India	0
4	TCS	23.0	4000.0	Mumbai	India	0

```
In [42]: print(f"\nFeature properties of the dataset:\n")
df.info()
```

Feature properties of the dataset:

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 148 entries, 0 to 147
Data columns (total 6 columns):
 #   Column   Non-Null Count   Dtype  
 ---  --       --           --      
 0   Company  140 non-null    object  
 1   Age      130 non-null    float64 
 2   Salary   124 non-null    float64 
 3   Place    134 non-null    object  
 4   Country  148 non-null    object  
 5   Gender   148 non-null    int64   
dtypes: float64(2), int64(1), object(3)
memory usage: 7.1+ KB
```

In [43]: `df.shape`

Out[43]: (148, 6)

In [44]: `dfc=df.copy()`

In [45]: `# Encode Gender: Male = 1, Female = 0`
`dfc['Gender'] = dfc['Gender'].apply(lambda x: 'Male' if x == 1 else 'Female')`

In [46]: `dfc.columns`

Out[46]: Index(['Company', 'Age', 'Salary', 'Place', 'Country', 'Gender'], dtype='object')

In [47]: `dfc.head()`

	Company	Age	Salary	Place	Country	Gender
0	TCS	20.0	NaN	Chennai	India	Female
1	Infosys	30.0	NaN	Mumbai	India	Female
2	TCS	35.0	2300.0	Calcutta	India	Female
3	Infosys	40.0	3000.0	Delhi	India	Female
4	TCS	23.0	4000.0	Mumbai	India	Female

In [48]: `dfc.describe()`

Out[48]:

	Age	Salary
count	130.000000	124.000000
mean	30.484615	5312.467742
std	11.096640	2573.764683
min	0.000000	1089.000000
25%	22.000000	3030.000000
50%	32.500000	5000.000000
75%	37.750000	8000.000000
max	54.000000	9876.000000

Data Exploration:

Explore the data, list down the unique values in each feature and find its length. Perform the statistical analysis and renaming of the columns.

In [50]:

```
# 1. Unique values and their counts
unique_values = [
    col: {"unique": df[col].unique().tolist(), "count": len(df[col].unique())} for
]
```

In [51]:

```
print("Unique Values and Counts:\n", unique_values)
```

Unique Values and Counts:

```
{'Company': {'unique': ['TCS', 'Infosys', 'CTS', nan, 'Tata Consultancy Services',
'Cognizant', 'Infosys Pvt Lmt'], 'count': 7}, 'Age': {'unique': [20.0, 30.0, 35.0,
40.0, 23.0, nan, 34.0, 45.0, 18.0, 22.0, 32.0, 37.0, 50.0, 21.0, 46.0, 36.0, 26.0, 4
1.0, 24.0, 25.0, 43.0, 19.0, 38.0, 51.0, 31.0, 44.0, 33.0, 17.0, 0.0, 54.0], 'coun
t': 30}, 'Salary': {'unique': [nan, 2300.0, 3000.0, 4000.0, 5000.0, 6000.0, 7000.0,
8000.0, 9000.0, 1089.0, 1234.0, 3030.0, 3045.0, 3184.0, 4824.0, 5835.0, 7084.0, 894
3.0, 8345.0, 9284.0, 9876.0, 2034.0, 7654.0, 2934.0, 4034.0, 5034.0, 8202.0, 9024.0,
4345.0, 6544.0, 6543.0, 3234.0, 4324.0, 5435.0, 5555.0, 8787.0, 3454.0, 5654.0, 500
9.0, 5098.0, 3033.0], 'count': 41}, 'Place': {'unique': ['Chennai', 'Mumbai', 'Calcutta',
'Delhi', 'Podicherry', 'Cochin', nan, 'Noida', 'Hyderabad', 'Bhopal', 'Nagpur', 'Pune'],
'count': 12}, 'Country': {'unique': ['India'], 'count': 1}, 'Gender': {'unique': [0, 1],
'count': 2}}
```

In [52]:

```
# Display unique values and their lengths for each feature
for col in df.columns:
    unique_vals = df[col].unique()
    print(f"Feature: {col}\nUnique Values: {unique_vals}\nCount: {len(unique_vals)}
```

Feature: Company

Unique Values: ['TCS' 'Infosys' 'CTS' nan 'Tata Consultancy Services' 'Congnizant' 'Infosys Pvt Lmt']

Count: 7

Feature: Age

Unique Values: [20. 30. 35. 40. 23. nan 34. 45. 18. 22. 32. 37. 50. 21. 46. 36. 26. 41.

24. 25. 43. 19. 38. 51. 31. 44. 33. 17. 0. 54.]

Count: 30

Feature: Salary

Unique Values: [nan 2300. 3000. 4000. 5000. 6000. 7000. 8000. 9000. 1089. 1234. 30 30.

3045. 3184. 4824. 5835. 7084. 8943. 8345. 9284. 9876. 2034. 7654. 2934.

4034. 5034. 8202. 9024. 4345. 6544. 6543. 3234. 4324. 5435. 5555. 8787.

3454. 5654. 5009. 5098. 3033.]

Count: 41

Feature: Place

Unique Values: ['Chennai' 'Mumbai' 'Calcutta' 'Delhi' 'Podicherry' 'Cochin' nan 'Noi da'

'Hyderabad' 'Bhopal' 'Nagpur' 'Pune']

Count: 12

Feature: Country

Unique Values: ['India']

Count: 1

Feature: Gender

Unique Values: [0 1]

Count: 2

statistical analysis

In [54]: `print("\nStatistical Analysis (Numeric Columns):\n", df.describe())`

Statistical Analysis (Numeric Columns):

| | Age | Salary | Gender |
|-------|------------|-------------|------------|
| count | 130.000000 | 124.000000 | 148.000000 |
| mean | 30.484615 | 5312.467742 | 0.222973 |
| std | 11.096640 | 2573.764683 | 0.417654 |
| min | 0.000000 | 1089.000000 | 0.000000 |
| 25% | 22.000000 | 3030.000000 | 0.000000 |
| 50% | 32.500000 | 5000.000000 | 0.000000 |
| 75% | 37.750000 | 8000.000000 | 0.000000 |
| max | 54.000000 | 9876.000000 | 1.000000 |

In [55]: `dfc.head(2)`

| | Company | Age | Salary | Place | Country | Gender |
|---|---------|------|--------|---------|---------|--------|
| 0 | TCS | 20.0 | NaN | Chennai | India | Female |
| 1 | Infosys | 30.0 | NaN | Mumbai | India | Female |

Rename the columns : Here all the names are capitalising and changing 'Place' to 'City'

```
In [57]: dfc.columns = dfc.columns.str.upper()
```

```
In [58]: dfc.head()
```

| | COMPANY | AGE | SALARY | PLACE | COUNTRY | GENDER |
|---|---------|------|--------|----------|---------|--------|
| 0 | TCS | 20.0 | NaN | Chennai | India | Female |
| 1 | Infosys | 30.0 | NaN | Mumbai | India | Female |
| 2 | TCS | 35.0 | 2300.0 | Calcutta | India | Female |
| 3 | Infosys | 40.0 | 3000.0 | Delhi | India | Female |
| 4 | TCS | 23.0 | 4000.0 | Mumbai | India | Female |

```
In [59]: #Changing the "Place" to "City"
dfc.rename(columns={'PLACE':'CITY'},inplace=True)
dfc.head()
```

| | COMPANY | AGE | SALARY | CITY | COUNTRY | GENDER |
|---|---------|------|--------|----------|---------|--------|
| 0 | TCS | 20.0 | NaN | Chennai | India | Female |
| 1 | Infosys | 30.0 | NaN | Mumbai | India | Female |
| 2 | TCS | 35.0 | 2300.0 | Calcutta | India | Female |
| 3 | Infosys | 40.0 | 3000.0 | Delhi | India | Female |
| 4 | TCS | 23.0 | 4000.0 | Mumbai | India | Female |

Data Cleaning:

Find the missing and inappropriate values, treat them appropriately

```
In [62]: dfc.isnull().sum()
```

```
Out[62]: COMPANY      8
          AGE        18
          SALARY     24
          CITY       14
          COUNTRY     0
          GENDER      0
          dtype: int64
```

Treating the missing values in Company with mode value

```
In [64]: #Imputation method
dfc['COMPANY'].fillna(dfc['COMPANY'].mode()[0], inplace=True)
dfc['COMPANY'].isnull().sum()
```

```
Out[64]: 0
```

Treating the missing values in City with mode value

```
In [66]: #Imputation method
dfc['CITY'].fillna(dfc['CITY'].mode()[0], inplace=True)
dfc['CITY'].isnull().sum()
```

```
Out[66]: 0
```

Treating the missing values in AGE with mode value

```
In [68]: #Imputation method
dfc['AGE'].fillna(dfc['AGE'].mode()[0], inplace=True)
dfc['AGE'].isnull().sum()
```

```
Out[68]: 0
```

Treating the missing values in SALARY with median value .

```
In [70]: #Imputation method
dfc['SALARY'].fillna(dfc['SALARY'].mean(), inplace=True)
dfc['SALARY'].isnull().sum()
```

```
Out[70]: 0
```

Remove all duplicate rows.

```
In [72]: dfc.duplicated().sum()
```

```
Out[72]: 4
```

```
In [73]: dfc.drop_duplicates(inplace=True)
```

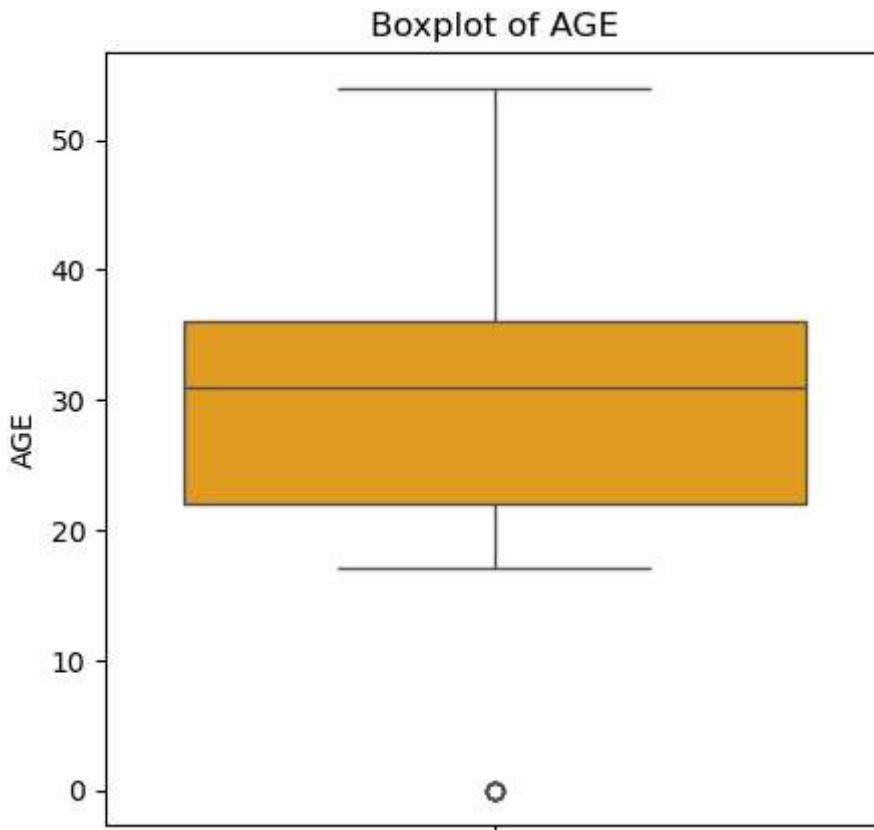
```
In [74]: dfc.duplicated().sum()
```

Out[74]: 0

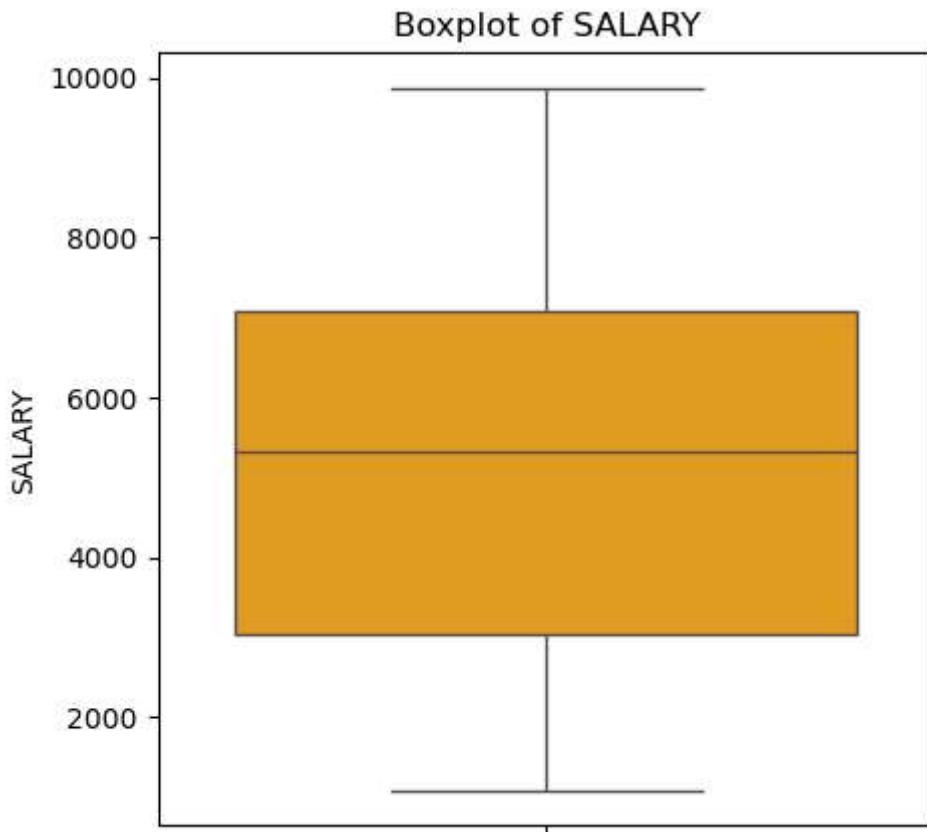
Find the outliers.

Finding outliers using plotting a box plot

```
In [77]: plt.figure(figsize=(5,5))
sns.boxplot(y=dfc['AGE'],color='orange')
plt.title('Boxplot of AGE')
plt.ylabel('AGE')
plt.show()
```



```
In [78]: plt.figure(figsize=(5,5))
sns.boxplot(y=dfc['SALARY'],color='orange')
plt.title('Boxplot of SALARY')
plt.ylabel('SALARY')
plt.show()
```



Using IQR method, finding the outlier

```
In [80]: # Calculate Q1 (25th percentile) and Q3 (75th percentile)
q1=dfc['AGE'].quantile(0.25)
q3=dfc['AGE'].quantile(0.75)
iqr=q3-q1
lower_lim=q1-iqr*1.5
upper_lim=q3+iqr*1.5

print(f"Lower Limit: {lower_lim}, Upper Limit: {upper_lim}")
```

Lower Limit: 1.0, Upper Limit: 57.0

```
In [81]: #Detecting outliers
outliers = dfc[(dfc['AGE'] < lower_lim) | (dfc['AGE'] > upper_lim)]
print(f"Number of outliers detected: {len(outliers)}")
```

Number of outliers detected: 6

```
In [82]: #Trimming method for removing outliers
dfc_cleaned=dfc[(dfc['AGE']>lower_lim) & (dfc['AGE']<upper_lim)]
print(f"Dataset after trimming: {dfc_cleaned.shape}")
```

Dataset after trimming: (138, 6)

```
In [83]: #Detecting outliers
outliers = dfc_cleaned[(dfc_cleaned['AGE'] < lower_lim) | (dfc_cleaned['AGE'] > upper_lim)]
print(f"Number of outliers detected: {len(outliers)}")
```

Number of outliers detected: 0

Outliers in AGE is cleaned and saved as dfc_cleaned

```
In [85]: # Calculate Q1 (25th percentile) and Q3 (75th percentile)
q1=dfc['SALARY'].quantile(0.25)
q3=dfc['SALARY'].quantile(0.75)
iqr=q3-q1
low_lim=q1-iqr*1.5
up_lim=q3+iqr*1.5

print(f"Lower Limit: {low_lim}, Upper Limit: {up_lim}")
```

Lower Limit: -3013.5, Upper Limit: 13142.5

```
In [86]: #Detecting outliers
outliers = dfc[(dfc['SALARY'] < low_lim) | (dfc['SALARY'] > up_lim)]
print(f"Number of outliers detected: {len(outliers)}")
```

Number of outliers detected: 0

There is no outlier for SALARY Column

Replace the value 0 in age as NaN

```
In [89]: # Replace 0 with NaN in the 'Age' column
dfc['AGE'] = dfc['AGE'].replace(0, np.nan)
```

Data Analysis:

Filter the data with age >40 and salary<5000

```
In [92]: print('Filtered data : \n')
filtered=dfc[(dfc['AGE']> 40) & (dfc['SALARY']<5000)]
filtered
```

Filtered data :

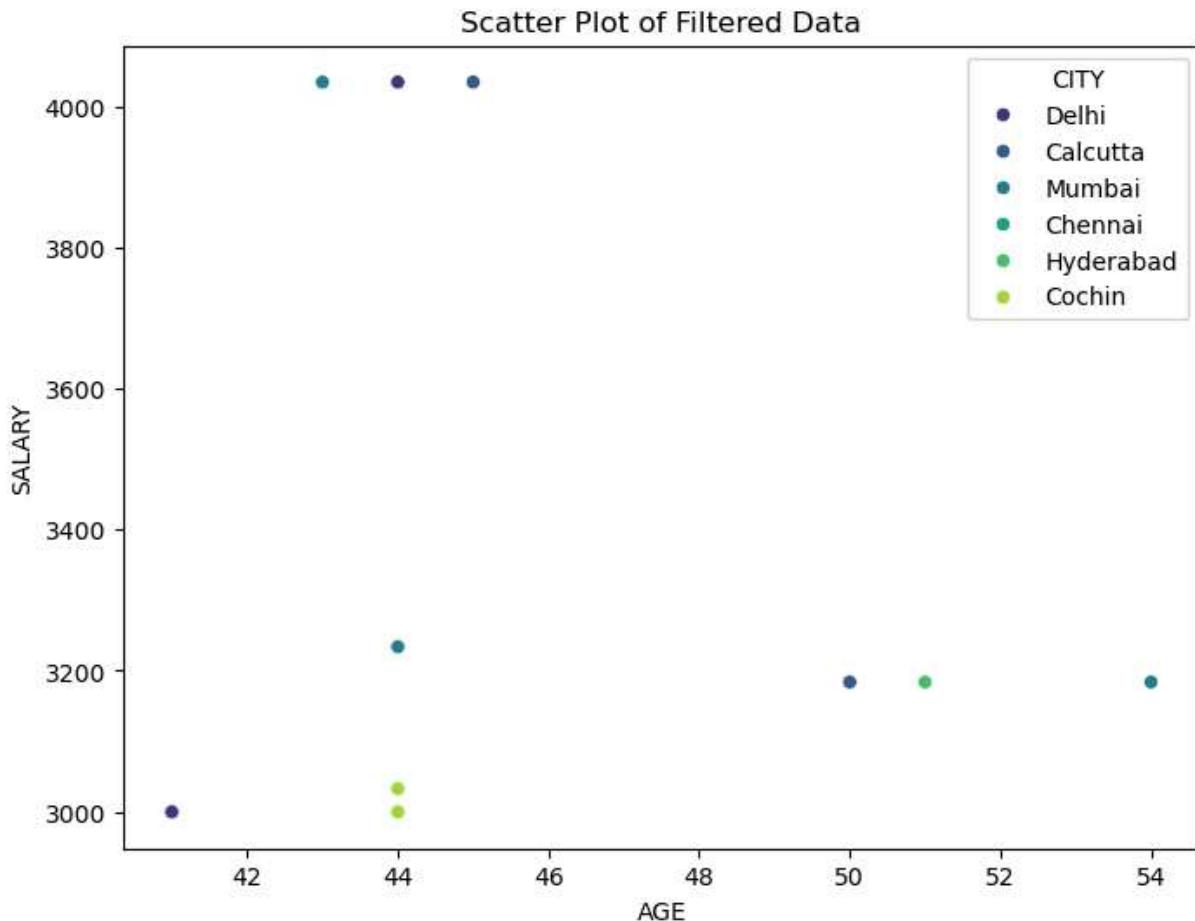
Out[92]:

| | COMPANY | AGE | SALARY | CITY | COUNTRY | GENDER |
|-----|---------|------|--------|-----------|---------|--------|
| 21 | Infosys | 50.0 | 3184.0 | Delhi | India | Female |
| 32 | Infosys | 45.0 | 4034.0 | Calcutta | India | Female |
| 39 | Infosys | 41.0 | 3000.0 | Mumbai | India | Female |
| 50 | Infosys | 41.0 | 3000.0 | Chennai | India | Female |
| 57 | Infosys | 51.0 | 3184.0 | Hyderabad | India | Female |
| 68 | Infosys | 43.0 | 4034.0 | Mumbai | India | Female |
| 75 | Infosys | 44.0 | 3000.0 | Cochin | India | Female |
| 86 | Infosys | 41.0 | 3000.0 | Delhi | India | Female |
| 93 | Infosys | 54.0 | 3184.0 | Mumbai | India | Female |
| 104 | Infosys | 44.0 | 4034.0 | Delhi | India | Female |
| 122 | Infosys | 44.0 | 3234.0 | Mumbai | India | Female |
| 129 | Infosys | 50.0 | 3184.0 | Calcutta | India | Female |
| 138 | CTS | 44.0 | 3033.0 | Cochin | India | Female |
| 140 | Infosys | 44.0 | 4034.0 | Hyderabad | India | Female |
| 145 | Infosys | 44.0 | 4034.0 | Delhi | India | Male |

Plot the chart with age and salary

In [94]:

```
plt.figure(figsize=(8, 6))
sns.scatterplot(data=filtered, x='AGE', y='SALARY', hue='CITY', palette='viridis')
plt.title('Scatter Plot of Filtered Data')
plt.xlabel('AGE')
plt.ylabel('SALARY')
plt.legend(title='CITY')
plt.show()
```



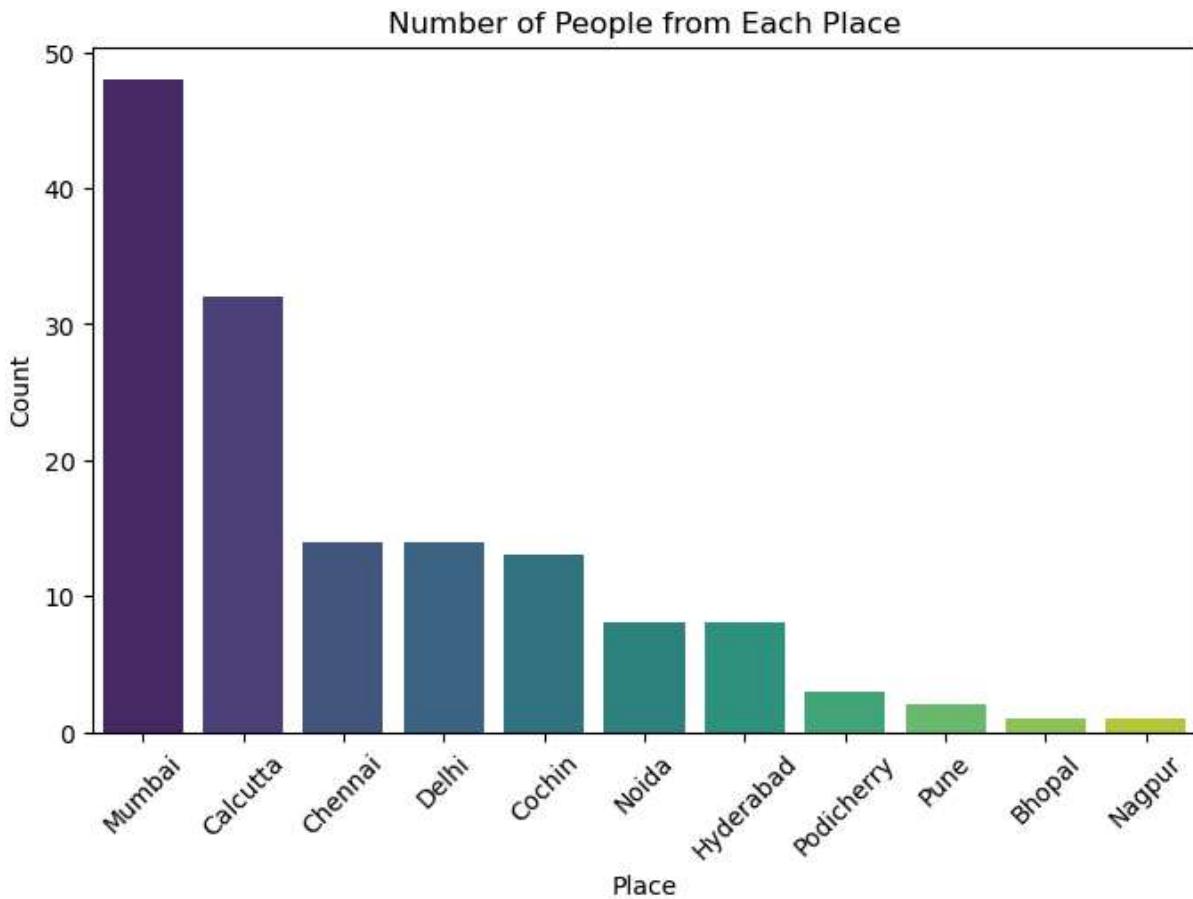
Count the number of people from each place and represent it visually

```
In [96]: # Count the number of people from each place
place_count = dfc['CITY'].value_counts()
place_count
```

```
Out[96]: CITY
Mumbai      48
Calcutta    32
Chennai     14
Delhi       14
Cochin      13
Noida        8
Hyderabad   8
Podicherry  3
Pune         2
Bhopal       1
Nagpur       1
Name: count, dtype: int64
```

```
In [97]: # Plot the counts
plt.figure(figsize=(8, 5))
sns.barplot(x=place_count.index, y=place_count.values, palette='viridis')
plt.title('Number of People from Each Place')
plt.xlabel('Place')
plt.xticks(rotation=45)
```

```
plt.ylabel('Count')
plt.show()
```



Data Encoding:

Convert categorical variables into numerical representations using techniques such as one-hot encoding, label encoding, making them suitable for analysis by machine learning algorithms.

```
In [100...]: from sklearn.preprocessing import LabelEncoder
```

```
In [101...]: dfc.head(2)
```

```
Out[101...]:
```

| | COMPANY | AGE | SALARY | CITY | COUNTRY | GENDER |
|---|---------|------|-------------|---------|---------|--------|
| 0 | TCS | 20.0 | 5312.467742 | Chennai | India | Female |
| 1 | Infosys | 30.0 | 5312.467742 | Mumbai | India | Female |

```
In [102...]: # Apply Label Encoding
label_encoder = LabelEncoder()
dfc['AGE']=label_encoder.fit_transform(dfc['AGE'])
dfc['SALARY']=label_encoder.fit_transform(dfc['SALARY'])
```

```
In [103...]: print('Dataset after Label encoding for Nominal columns :\n')  
dfc.head(3)
```

Dataset after Label encoding for Nominal columns :

```
Out[103...]:
```

| | COMPANY | AGE | SALARY | CITY | COUNTRY | GENDER |
|---|---------|-----|--------|----------|---------|--------|
| 0 | TCS | 3 | 21 | Chennai | India | Female |
| 1 | Infosys | 10 | 21 | Mumbai | India | Female |
| 2 | TCS | 15 | 3 | Calcutta | India | Female |

```
In [ ]:
```

```
In [104...]: # Apply One-Hot Encoding  
one_hot_encoded = pd.get_dummies(dfc, columns=['CITY', 'COMPANY', 'COUNTRY', 'GENDER'])  
print("One-Hot Encoded Data:")  
print(one_hot_encoded)
```

One-Hot Encoded Data:

| | AGE | SALARY | CITY_Bhopal | CITY_Calcutta | CITY_Chennai | CITY_Cochin | \ |
|------|-----------------|--------------------|-----------------|-----------------------------------|--------------|-------------|-----|
| 0 | 3 | 21 | False | False | True | False | |
| 1 | 10 | 21 | False | False | False | False | |
| 2 | 15 | 3 | False | True | False | False | |
| 3 | 19 | 5 | False | False | False | False | |
| 4 | 6 | 12 | False | False | False | False | |
| .. | ... | ... | ... | ... | ... | ... | ... |
| 142 | 5 | 33 | False | False | False | False | |
| 143 | 13 | 38 | False | True | False | False | |
| 145 | 22 | 13 | False | False | False | False | |
| 146 | 13 | 19 | False | False | False | False | |
| 147 | 5 | 33 | False | False | False | True | |
|
 | | | | | | | |
| | CITY_Delhi | CITY_Hyderabad | CITY_Mumbai | CITY_Nagpur | ... | CITY_Pune | \ |
| 0 | False | False | False | False | ... | False | |
| 1 | False | False | True | False | ... | False | |
| 2 | False | False | False | False | ... | False | |
| 3 | True | False | False | False | ... | False | |
| 4 | False | False | True | False | ... | False | |
| .. | ... | ... | ... | ... | ... | ... | ... |
| 142 | False | False | True | False | ... | False | |
| 143 | False | False | False | False | ... | False | |
| 145 | True | False | False | False | ... | False | |
| 146 | False | False | True | False | ... | False | |
| 147 | False | False | False | False | ... | False | |
|
 | | | | | | | |
| | COMPANY_CTS | COMPANY_Congnizant | COMPANY_Infosys | \ | | | |
| 0 | False | False | False | False | | | |
| 1 | False | False | True | | | | |
| 2 | False | False | False | | | | |
| 3 | False | False | True | | | | |
| 4 | False | False | False | | | | |
| .. | ... | ... | ... | ... | | | |
| 142 | False | False | False | False | | | |
| 143 | False | False | False | False | | | |
| 145 | False | False | True | | | | |
| 146 | False | False | False | False | | | |
| 147 | False | False | True | | | | |
|
 | | | | | | | |
| | COMPANY_Infosys | Pvt Lmt | COMPANY_TCS | COMPANY_Tata Consultancy Services | \ | | |
| 0 | | False | True | | False | | |
| 1 | | False | False | | False | | |
| 2 | | False | True | | False | | |
| 3 | | False | False | | False | | |
| 4 | | False | True | | False | | |
| .. | ... | ... | ... | | ... | | |
| 142 | | True | False | | False | | |
| 143 | | False | True | | False | | |
| 145 | | False | False | | False | | |
| 146 | | False | True | | False | | |
| 147 | | False | False | | False | | |
|
 | | | | | | | |
| | COUNTRY_India | GENDER_Female | GENDER_Male | | | | |
| 0 | True | True | False | | | | |
| 1 | True | True | False | | | | |

```

2          True      True    False
3          True      True    False
4          True      True    False
..
142         ...       ...     ...
143         True      False   True
145         True      False   True
146         True      False   True
147         True      True    False

```

[144 rows x 22 columns]

Feature Scaling:

After the process of encoding, perform the scaling of the features using standardscaler and minmaxscaler.

In [106...]

```

from sklearn.preprocessing import StandardScaler, MinMaxScaler

# Initialize StandardScaler and MinMaxScaler
standard_scaler = StandardScaler()
minmax_scaler = MinMaxScaler()

```

In [107...]

```

# Select only the numerical columns i.e., 'AGE', 'SALARY'
numerical_columns = ['AGE', 'SALARY']
# Apply StandardScaler to numerical columns
dfc_standard_scaled = dfc.copy()
dfc_standard_scaled[numerical_columns] = standard_scaler.fit_transform(dfc_standard)

# Applying MinMaxScaler to numerical columns
dfc_minmax_scaled = dfc.copy()
dfc_minmax_scaled[numerical_columns] = minmax_scaler.fit_transform(dfc_minmax_scaled)

# Display the first few rows of the scaled data
print("Data after Standard Scaling:")
print(dfc_standard_scaled.head(10))

print("\nData after Min-Max Scaling:")
print(dfc_minmax_scaled.head(10))

```

Data after Standard Scaling:

| | COMPANY | AGE | SALARY | CITY | COUNTRY | GENDER |
|---|---------|-----------|-----------|----------|---------|--------|
| 0 | TCS | -1.246731 | 0.139604 | Chennai | India | Female |
| 1 | Infosys | -0.312378 | 0.139604 | Mumbai | India | Female |
| 2 | TCS | 0.355017 | -1.368123 | Calcutta | India | Female |
| 3 | Infosys | 0.888933 | -1.200598 | Delhi | India | Female |
| 4 | TCS | -0.846294 | -0.614259 | Mumbai | India | Female |
| 5 | Infosys | -0.979773 | -0.195446 | Calcutta | India | Female |
| 6 | TCS | -0.979773 | 0.558417 | Chennai | India | Male |
| 7 | Infosys | -0.846294 | 0.809705 | Mumbai | India | Male |
| 8 | TCS | 0.221538 | 1.060993 | Calcutta | India | Male |
| 9 | CTS | 1.422849 | 1.479806 | Delhi | India | Female |

Data after Min-Max Scaling:

| | COMPANY | AGE | SALARY | CITY | COUNTRY | GENDER |
|---|---------|----------|--------|----------|---------|--------|
| 0 | TCS | 0.107143 | 0.525 | Chennai | India | Female |
| 1 | Infosys | 0.357143 | 0.525 | Mumbai | India | Female |
| 2 | TCS | 0.535714 | 0.075 | Calcutta | India | Female |
| 3 | Infosys | 0.678571 | 0.125 | Delhi | India | Female |
| 4 | TCS | 0.214286 | 0.300 | Mumbai | India | Female |
| 5 | Infosys | 0.178571 | 0.425 | Calcutta | India | Female |
| 6 | TCS | 0.178571 | 0.650 | Chennai | India | Male |
| 7 | Infosys | 0.214286 | 0.725 | Mumbai | India | Male |
| 8 | TCS | 0.500000 | 0.800 | Calcutta | India | Male |
| 9 | CTS | 0.821429 | 0.925 | Delhi | India | Female |

Feature Scaling using MIN-MAX Scaling method and STANDARD Scaling method is done

```
In [109...]: # pip install notebook ipython jupyter-dash
```

```
In [110...]: import plotly
print(plotly.__version__)
```

5.24.1

Data Visualization with Plotly

Show the distribution of employees based on their location.

```
In [113...]: dfc.head()
```

| | COMPANY | AGE | SALARY | CITY | COUNTRY | GENDER |
|---|---------|-----|--------|----------|---------|--------|
| 0 | TCS | 3 | 21 | Chennai | India | Female |
| 1 | Infosys | 10 | 21 | Mumbai | India | Female |
| 2 | TCS | 15 | 3 | Calcutta | India | Female |
| 3 | Infosys | 19 | 5 | Delhi | India | Female |
| 4 | TCS | 6 | 12 | Mumbai | India | Female |

```
In [115...]  
import plotly.express as px  
fig = px.pie(  
    dfc,  
    names="CITY",  
    title="Distribution of Employees by Place",  
    hole=0.4, # For a donut chart  
    color_discrete_sequence=px.colors.sequential.RdBu,  
)  
  
fig.show()
```

```
In [119...]  
# Calculate average salary by company  
fig = px.bar(  
    dfc,  
    x="COMPANY",  
    y="SALARY",  
    color="COMPANY",  
    title="Average Salary by Company",  
    labels={"SALARY": "Average Salary"},  
    barmode="group",  
)  
  
fig.show()
```

```
In [131]: fig = px.scatter(  
    df,  
    x="Age",  
    y="Salary",  
    color="Company",  
    title="Age vs. Salary by Company",  
    labels={"AGE": "Age", "SALARY": "Salary"},  
    hover_data=["Place"],  
)  
  
fig.show()
```

In []: