Instructions_19.09_Mathplotlib

Matplotlib Visualization Instructions

Task 1: Bar Plot - Average Salary by Department

1. Import `matplotlib.pyplot` and `pandas`, and load the dataset into a DataFrame.
2. Compute the average salary for each department using the `groupby` method.
3. Create a bar plot showing the average salary by department with `plt.bar()`.
4. Rotate x-axis labels using `plt.xticks(rotation=45)` for better readability.
5. Add a title with `plt.title()`, x-axis label with `plt.xlabel()`, and y-axis label with `plt.ylabel()`.
6. Display the plot with `plt.show()`.

Task 2: Scatter Plot - Salary vs Performance Rating

1. Import `matplotlib.pyplot` and `pandas`, and load the dataset into a DataFrame.
2. Create a scatter plot with salary on the x-axis and performance rating on the y-axis using `plt.scatter()`.
3. Use different colors or markers for different departments with the `c` or `marker` parameters.
4. Add a title with `plt.title()`, x-axis label with `plt.xlabel()`, and y-axis label with `plt.ylabel()`.
5. Display the plot with `plt.show()`.

Task 3: Box Plot - Salary Distribution by Department

1. Import `matplotlib.pyplot` and `pandas`, and load the dataset into a DataFrame.
2. Prepare the data by extracting unique department names and salary data for each department.
3. Create a box plot with `plt.boxplot()`.
4. Rotate x-axis labels using `plt.xticks(rotation=45)` for better readability.
5. Add a title with `plt.title()`, x-axis label with `plt.xlabel()`, and y-axis label with `plt.ylabel()`.
6. Display the plot with `plt.show()`.

Task 4: Count Plot - Number of Employees by Department

1. Import `matplotlib.pyplot` and `pandas`, and load the dataset into a DataFrame.
2. Compute the number of employees per department using `groupby` and `size()`.
3. Create a bar plot showing the count of employees in each department with `plt.bar()`.
4. Rotate x-axis labels using `plt.xticks(rotation=45)` for better readability.
5. Add a title with `plt.title()`, x-axis label with `plt.xlabel()`, and y-axis label with `plt.ylabel()`.
6. Display the plot with `plt.show()`.

Task 5: Line Plot - Salary Over Time (Start Dates)

1. Import `matplotlib.pyplot` and `pandas`, and load the dataset into a DataFrame.
2. Convert start dates to datetime format using `pd.to_datetime()`.
3. Create a line plot with start dates on the x-axis and salaries on the y-axis using `plt.plot()`.
4. Add markers to the line plot using the `marker` parameter.
5. Add a title with `plt.title()`, x-axis label with `plt.xlabel()`, and y-axis label with `plt.ylabel()`.
6. Display the plot with `plt.show()`.

Task 6: Heatmap - Correlation Matrix

1. Import `matplotlib.pyplot` and `pandas`, and load the dataset into a DataFrame.
2. Compute the correlation matrix for numerical features using `DataFrame.corr()`.
3. Create a heatmap with `plt.imshow()` and `plt.colorbar()` for the correlation matrix.
4. Add annotations using `plt.text()` to display correlation values.
5. Add a title with `plt.title()`.
6. Display the plot with `plt.show()`.

Task 7: Violin Plot - Salary Distribution by Department

1. Import `matplotlib.pyplot` and `pandas`, and load the dataset into a DataFrame.
2. Create a violin plot with `plt.violinplot()`.
3. Rotate x-axis labels using `plt.xticks(rotation=45)` for better readability.
4. Add a title with `plt.title()`, x-axis label with `plt.xlabel()`, and y-axis label with `plt.ylabel()`.

5. Display the plot with `plt.show()`.

Task 8: Pair Plot - Relationships Between Numerical Features

1. Import `matplotlib.pyplot` and `pandas`, and load the dataset into a DataFrame.
2. Select the numerical columns for the pair plot.
3. Create a pair plot using `plt.scatter()` in a nested loop to show relationships between these numerical features.
4. Add a title with `plt.suptitle()`.
5. Display the plot with `plt.show()`.

Task 9: Histogram - Salary Distribution

1. Import `matplotlib.pyplot` and `pandas`, and load the dataset into a DataFrame.
2. Create a histogram with `plt.hist()`.
3. Optionally, include a kernel density estimate (KDE) curve using `scipy.stats.gaussian_kde()`.
4. Add a title with `plt.title()`, x-axis label with `plt.xlabel()`, and y-axis label with `plt.ylabel()`.
5. Display the plot with `plt.show()`.

Task 10: FacetGrid - Performance Rating by Project

1. Import `matplotlib.pyplot` and `pandas`, and load the dataset into a DataFrame.
2. Use `matplotlib` subplots to create scatter plots for each project.
3. Use `plt.scatter()` for each subplot to show salary versus performance rating.
4. Display the plots using `plt.show()`.

Task 11: Strip Plot - Performance Rating by Role

1. Import `matplotlib.pyplot` and `pandas`, and load the dataset into a DataFrame.
2. Create a strip plot using `plt.scatter()` with roles on the x-axis and performance ratings on the y-axis.
3. Add jitter to the plot to prevent overlapping of data points using `plt.scatter()` with randomized x-values.
4. Rotate x-axis labels using `plt.xticks(rotation=45)` for better readability.

5. Add a title with `plt.title()`, x-axis label with `plt.xlabel()`, and y-axis label with `plt.ylabel()`.
6. Display the plot with `plt.show()`.