Experiment 2 : Visualization of data

1. Correlation Heatmap

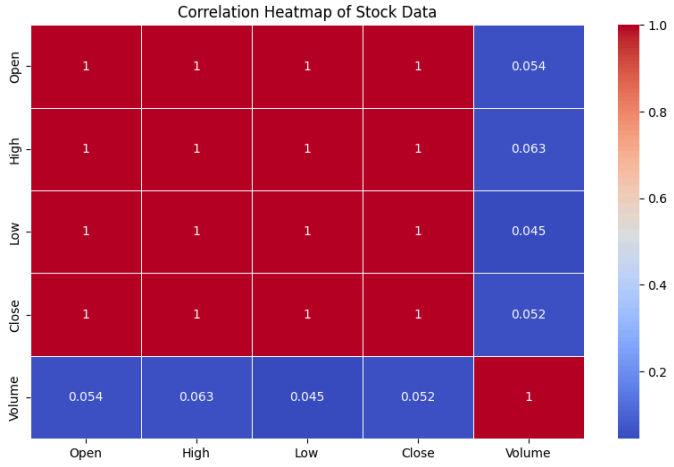
A heatmap is used to visualize the correlation between numerical features in the dataset. It helps in identifying relationships between different variables.

Code:

```
Unset
import seaborn as sns
import matplotlib.pyplot as plt

df_numeric = df.select_dtypes(include=[np.number])  # Select only
numeric columns
plt.figure(figsize=(10, 6))
sns.heatmap(df_numeric.corr(), annot=True, cmap='coolwarm',
linewidths=0.5)
plt.title('Correlation Heatmap of Stock Data')
plt.show()
```

Explanation:

- The function `df.corr()` calculates the correlation matrix.
- `sns.heatmap()` is used to create the heatmap.
- The `annot=True` argument ensures that correlation values are displayed.
- The colormap `coolwarm` visually distinguishes positive and negative correlations.
-



---

2. Stock Price Trend Graph

A line plot is used to visualize the stock price trend over time. This helps in understanding how stock prices fluctuate.

Code:

```
Unset
import matplotlib.dates as mdates

plt.figure(figsize=(12, 6))
plt.plot(df['Date'], df['Close'], color='blue')
plt.title('Stock Price Trend Over Time')
plt.xlabel('Date')
plt.ylabel('Closing Price ($)')

plt.xticks(rotation=45)  # Rotate x-axis labels
plt.gca().xaxis.set_major_locator(mdates.AutoDateLocator())  #
Auto adjust date intervals

plt.grid(True)
plt.show()
```

Explanation:

- `plt.plot()` is used to create a line graph of closing prices over time.
- `plt.xticks(rotation=45)` improves readability by rotating the dates.
- `mdates.AutoDateLocator()` automatically adjusts date intervals for better display.
- Grid lines are added for clarity.
- 

---

3. Network Graph

A network graph is used to visualize relationships between numerical features based on their correlation.

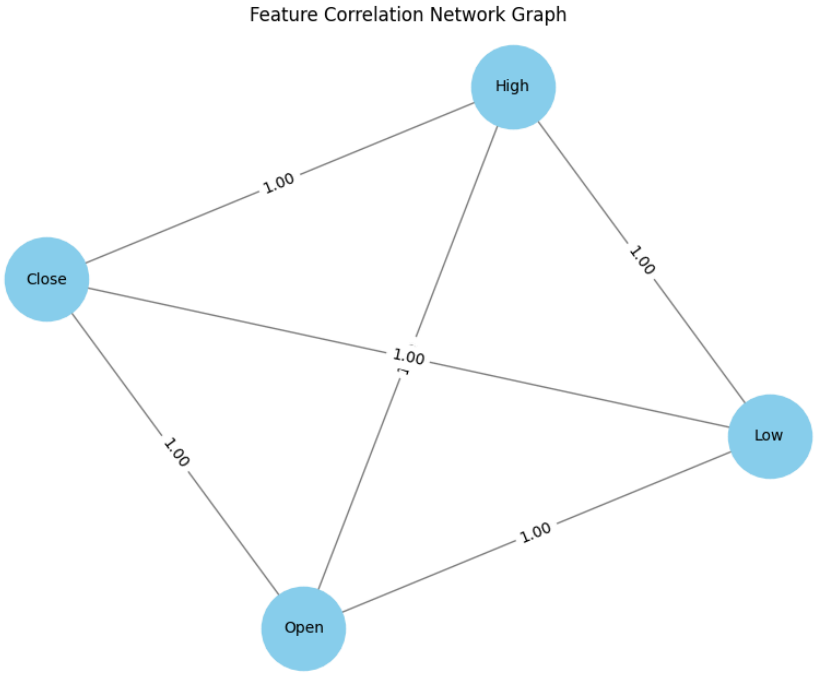
Code:

```
Unset
import networkx as nx

df_numeric = df.select_dtypes(include=[np.number])
corr_matrix = df_numeric.corr()

G = nx.Graph()

threshold = 0.5  # Only show strong correlations
for i in corr_matrix.columns:
    for j in corr_matrix.columns:
        if i != j and abs(corr_matrix.loc[i, j]) > threshold:
            G.add_edge(i, j, weight=corr_matrix.loc[i, j])

plt.figure(figsize=(8, 6))
pos = nx.spring_layout(G)
nx.draw(G, pos, with_labels=True, node_color='skyblue',
edge_color='gray', node_size=3000, font_size=10)

# Draw edge labels (correlation values)
edge_labels = {(i, j): f"{corr_matrix.loc[i, j]:.2f}" for i, j in
G.edges()}
nx.draw_networkx_edge_labels(G, pos, edge_labels=edge_labels)

plt.title("Feature Correlation Network Graph")
plt.show()
```

Explanation:

- The correlation matrix is computed using `df.corr()`.
- A graph is created where nodes represent numerical features.

- Edges (connections) are added if the correlation between two features exceeds a threshold (`0.5`).
- `nx.spring_layout(G)` is used to position nodes in a visually appealing way.
- Labels and edge weights (correlation values) are displayed.



-

4 Distribution of Microsoft Stock Closing Prices

A histogram is used to visualize the distribution of Microsoft stock's closing prices, helping to understand how often certain price ranges occur.

Code:

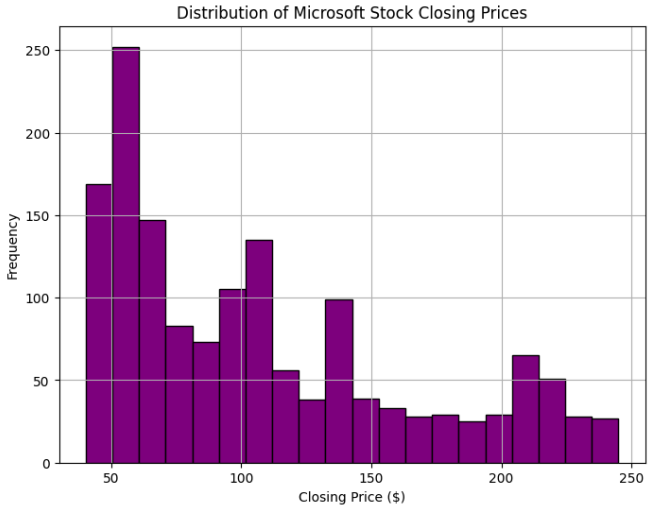
python

CopyEdit

- `plt.figure(figsize=(8, 6))`
- `plt.hist(df['Close'], bins=20, color='purple', edgecolor='black')`
- `plt.title('Distribution of Microsoft Stock Closing Prices')`
- `plt.xlabel('Closing Price ($)')`
- `plt.ylabel('Frequency')`
- `plt.grid(True)`
- `plt.show()`

Explanation:

1. `plt.figure(figsize=(8, 6))`
    - **Purpose:** This initializes the figure with a width of 8 inches and a height of 6 inches.
    - **Explanation:** The `figsize` argument defines the size of the plot. A smaller size of `(8, 6)` is chosen to fit a histogram without excessive space around it.
2. `plt.hist(df['Close'], bins=20, color='purple', edgecolor='black')`
    - **Purpose:** This creates the histogram of closing prices.
    - **Explanation:**
        - `df['Close']`: Refers to the 'Close' column in the DataFrame `df`, which contains the stock's closing prices.
        - `bins=20`: Specifies the number of bins (intervals) in the histogram. Here, there are 20 bins, which helps in displaying the distribution of prices across different price ranges.
        - `color='purple'`: Sets the color of the bars in the histogram to purple.
        - `edgecolor='black'`: Adds a black border around each bin for better visibility and separation between bars.



5) Box Plot of Microsoft Stock Closing Price

A box plot is used to visualize the distribution of Microsoft stock's closing prices, highlighting key statistics like the median, quartiles, and outliers.

Code:

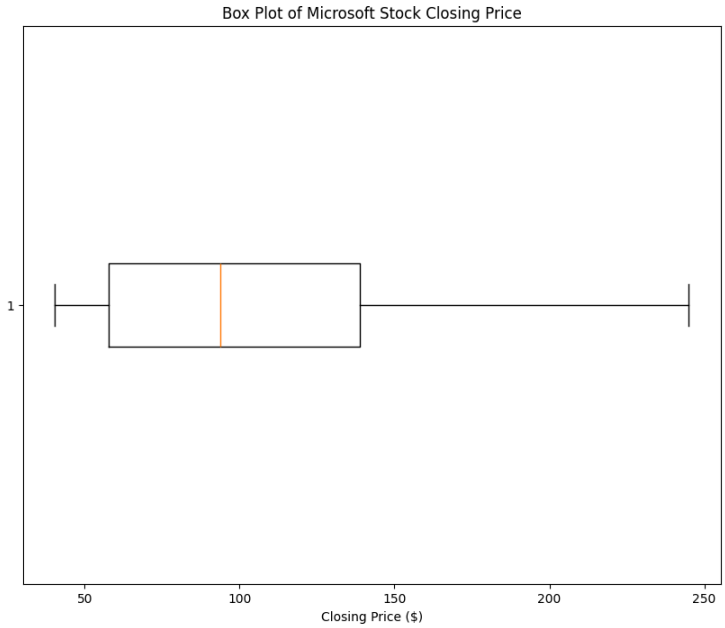
python

CopyEdit

```
plt.figure(figsize=(10, 8))

plt.boxplot(df['Close'], vert=False)

plt.title('Box Plot of Microsoft Stock Closing Price')

plt.xlabel('Closing Price ($)')

plt.show()
```

Explanation:

1. **`plt.figure(figsize=(10, 8))`**
   - **Purpose:** Initializes the figure with a specified size.
   - **Explanation:** The `figsize` argument sets the dimensions of the figure. In this case, the figure has a width of 10 inches and a height of 8 inches, which is suitable for displaying a clear box plot.
2. **`plt.boxplot(df['Close'], vert=False)`**
   - **Purpose:** Creates the box plot for the 'Close' data.
   - **Explanation:**
     - `df['Close']`: Refers to the 'Close' column in the DataFrame `df`, containing the stock's closing prices.
     - `plt.boxplot()`: This function generates the box plot, which shows the spread and skewness of the closing prices.
     - `vert=False`: Specifies that the box plot should be horizontal (vert=False). By default, box plots are vertical, but setting it to `False` changes the orientation for easier reading when dealing with numeric data.



Result:

Thus the visualization techniques in Time Series Analysis and Forecasting has been studied successfully.