HyperionDev

Data Visualisation – Python Libraries

Model Answer Approach

Visit our website

Auto-graded task 1

Task 1 involved creating four graphs using the matplotlib Python package from the **Cars93.csv** (93CARS) dataset and answering the accompanying questions in the markdown cells of the Jupyter notebook.

The first graph was a box plot for the revs per mile for the Audi, Hyundai, Suzuki, and Toyota car manufacturers. The approach taken was to filter the data for the selected manufacturers and then create a box plot for each of them. One common pitfall in this approach could be not properly filtering the data, which would lead to incorrect results. Another could be not correctly setting up the box plot data, which would lead to errors during plotting. This solution is fitting as it allows for a clear comparison of the revs per mile distributions for the selected manufacturers.

The second one was a histogram of MPG in the city. On the same axis, it also showed a histogram of MPG on the motorway. The approach taken was to create two histograms on the same axis, one for MPG in the city and one for MPG on the motorway. This allowed for a direct comparison of the two distributions. One common pitfall in this approach could be not setting the “alpha” parameter for the histograms, which would result in one histogram completely obscuring the other. Another could be not correctly setting up the “bins” parameter, which would result in poorly defined or uneven bins. This solution is fitting as it allows for a clear comparison of the MPG distributions for city and motorway driving. The use of transparency in the histograms allows both distributions to be visible, and the use of a legend allows the viewer to easily understand which histogram corresponds to which type of driving.

The third one was a line plot showing the relationship between the wheelbase and turning circle. The approach taken was to create a scatter plot of the data, perform a linear regression to find the line of best fit, and then plot this trend line on the same axes as the original data. One common pitfall in this approach could be not correctly performing the linear regression, which would result in an incorrect trend line. Another could be not correctly plotting the trend line, which would result in it not appearing on the plot. This solution is fitting as it allows for a clear visualisation of the relationship between the wheelbase and turning circle of a car. The trend line provides a summary of this relationship, making it easier to understand. The code uses the `linregress`

function from the `scipy.stats` module, which requires the scipy library to be installed in your Python environment. If it's not installed, you can add it using pip: `pip install scipy`.

Lastly, the fourth graph was a bar plot showing the mean horsepower for each car type (small, midsize, etc.). The approach taken was to calculate the mean horsepower for each car type and then create a bar plot to visualise these. One common pitfall in this approach could be not correctly calculating the mean horsepower, which would lead to incorrect results. Another could be not correctly setting up the bar plot, which would result in errors during plotting. This solution is fitting as it allows for a clear comparison of the mean horsepower for each car type. The use of different colours for the bars makes it easy to distinguish between the different car types.

From these graphs, we were able to gain valuable insights into the performance and characteristics of different cars, and the visualisations made it easier to understand the data and draw conclusions. This task demonstrated the power of matplotlib in data analysis and visualisation.

Auto-graded task 2

Task 2 explores the wine quality dataset to understand the rating distributions of three wine varieties: "Cabernet Sauvignon", "Pinot Noir", and "Chardonnay".

This approach begins by loading the **wine** CSV file into a pandas DataFrame. By setting `index_col=[0]`, we use the first column of the CSV file as the index of the DataFrame, avoiding the creation of the "Unnamed: 0" column.

After that, we check the basic information of the data by using the `.info()` method and `.unique()` in the variety column to check the unique values there.

Then, we filtered the data that contains the "Cabernet Sauvignon", "Pinot Noir", and "Chardonnay" wine varieties by using the `.isin()` method, which checks membership in a pandas series against a list. By passing it inside `wine[]`, it keeps only the rows where the condition is True.

Next, we can get a sense of how many reviews are associated with each wine variety by using `.value_counts()`, which counts each type of element present in the variety column of the filtered data.

Lastly, we create a multi-plot grid by using seaborn `FacetGrid` where each grid will represent one of the three wine varieties, and within each grid we display a histogram

for wine ratings (represented by the “points” column). This analysis aids in understanding rating distributions of wine ratings for selected varieties, potentially revealing trends.