AI Usage Declaration Form

# Referencing the Output of Artificial Intelligence Language Models

*Output text of an artificial intelligence language model (such as a ChatBot) for written assessment items of this unit will not be treated as your own work.*

* *If any output text is used, it must be quoted in italics, and the app or website must be referenced appropriately (AI language model Name/Chatbot Name, Developer or Organisation, date accessed).*

*• If any output ideas are used and paraphrased, the app or website must be referenced appropriately (AI language model Name/Chatbot Name, Developer or Organisation, date accessed).*

* *If the output of an artificial intelligence language model is prescribed by your Lecturer as an integral part of an assessment item, the app or website must be referenced appropriately (AI language model Name/Chatbot Name, Developer or Organisation, date accessed).*

(excerpt from Unit Information)

**Step 1:** Select the declaration statement that best describes the level of AI tool usage in this assignment.

**Table 1**. Declaration Statement

| Tick (✔) only 1 statement that applies | Declaration statements |
| --- | --- |
|  | **Declaration 1:**  We do not use any content generated by AI tools to be presented as our work. |
|  | **Declaration 2:**  We acknowledge the use of AI tool(s) to generate materials that aid with the process of completing this assessment (e.g. for background research or self-study). None of these materials are presented as our work. |
| ✔ | **Declaration 3:**  We acknowledge the use of AI tool(s) to generate materials that are included in our work in modified forms. |

**Step 2:** Complete the following form if you selected either Declaration 2 or Declaration 3. You do not need to complete this section if you select Declaration 1.

**Table 2.** Details of AI Usage

| Reference to one or more AI tools. Example: <https://shorturl.at/8UT25> | Response:  AI tools used  ChatGpt 5  llama 4 Scout  GitHub Co-pilot |
| --- | --- |
| List all prompts you have used. | Response:  **Duck.ai**  Prompt 1: what does update layout function do line by line    Prompt 2: and describe add trace function    Prompt 3:  *import numpy as np*  *from scipy.stats import binomtest*  *# Initialize an array to store p-values*  *p\_values = []*  *# Loop through threshold values from 1 to 10*  *for threshold in range(1, 5):*  *# Create a new column based on the current threshold*  *df5['above\_seconds'] = (df5['bat\_landing\_to\_food'] > threshold).astype(int)*  *# Count how many 1s*  *successes = df5['above\_seconds'].sum()*  *# Total number of trials*  *n = len(df5)*  *# Perform one-sided binomial test*  *result = binomtest(successes, n, p=0.5, alternative='greater')*  *# Store the p-value in the array*  *p\_values.append(result.pvalue.round(10))*  *# Print the results for the current threshold*  *print(f"Threshold: {threshold}, Number of ones: {successes}, Number of trials: {n}, p-value: {result.pvalue.round(10)}")*  *# Convert p\_values to a NumPy array for further analysis if needed*  *p\_values\_array = np.array(p\_values)*  can you make it increment it by 0.2    **ChatGPT**  Prompt 1:  df\_cleaned\_2 = df\_cleaned.iloc[:, 0:5]  # plot the correlation matrix  numeric\_df = df\_cleaned\_2.select\_dtypes(include=[np.number])  correlation\_matrix = numeric\_df.corr()  plt.figure(figsize=(10, 8))  im = plt.imshow(correlation\_matrix, cmap='coolwarm', interpolation='nearest')  cbar = plt.colorbar(im, fraction=0.046, pad=0.04)  cbar.ax.tick\_params(labelsize=10)  ticks = np.arange(len(correlation\_matrix.columns))  plt.xticks(ticks, correlation\_matrix.columns, rotation=90)  plt.yticks(ticks, correlation\_matrix.columns)  for i in range(len(correlation\_matrix.columns)):  for j in range(len(correlation\_matrix.columns)):  plt.text(j, i, f"{correlation\_matrix.iloc[i, j]:.2f}",  ha='center', va='center', color='black', fontsize=8)  plt.title("Correlation Matrix of Cleaned Dataset", fontsize=14)  plt.tight\_layout()  plt.show()  convert this to plotly library    Prompt 2: change no variables or comments, add \_cleaned to the title of each plot    Prompt 3: dont chnage any of my comments or documanteation or any lines of code that donty need to be changed. just make it so that all the prints are labelled with print statements telling what the following output is for for each cell of code    Prompt 4:  # this shows that the rat is always present before the bat as the 'rat\_period\_start' is always before 'start\_time'  df1['rat\_period\_start'] = pd.to\_datetime(df1['rat\_period\_start'], format='%H:%M')  df1['start\_time'] = pd.to\_datetime(df1['start\_time'], format='%H:%M')  df1['time\_difference'] = (df1['start\_time'] - df1['rat\_period\_start']).dt.total\_seconds()  # Example: Suppose your DataFrame has a column "risk"  # with values 0 and 1  # df['risk']  # Count how many 1s  successes = (df1['risk'] == 0).sum()  # Total number of trials  n = len(df1)  # Perform one-sided binomial test  result = binomtest(successes, n, p=0.5, alternative='greater')  print("Number of ones:", successes)  print("Number of trials:", n)  print("p-value:", result.pvalue.round(4))  p\_val = result.pvalue  if p\_val < 0.05:  print("\t We reject the null hypothesis.")  else:  print("\t We accept the null hypothesis.")  # the p value was 0.471 which is above 0.05, so we accept the null hypothesis that the risk of bat presence is not significantly different from 0.  give an appropriate documentation line explaining this batch of code    Prompt 5:  print("\n")  print(f'\nfor {y} when the dataset is split using {x}:\n')  print("\tRats -")  print("\tMean:\n", mean\_1)  print("\tStandard Deviation:\n", std\_1)  print("\tSamples:\n", n\_1)  print("\nNo Rats:")  print("Mean:\n", mean\_2)  print("Standard Deviation:\n", std\_2)  print("Samples:\n", n\_2)  """  Perform a two-sided Z-test to compare the means of y between the two groups:  df\_above\_zero and df\_zero. Returns the Z-statistic (z\_stats) and the p-value (p\_val) to test  the null hypothesis that the two group means are equal.  """  z\_stats, p\_val = ztest(df\_above\_zero[y], df\_zero[y], alternative='two-sided')  print("\n Computing z\* ...")  print("\t z-statistic (t\*): %.2f" % z\_stats)  print("\n Computing p-value ...")  print("\t p-value: %.10f" % p\_val.round(30))  print("\n Conclusion:")  make it so this is all simplified into fstring prints where every print is tabbed and and every value is rounded to 2 decimal places    Prompt 6:  z\_stats, p\_val = ztest(df\_above\_zero['bat\_landing\_number'], df\_zero['bat\_landing\_number'], alternative='two-sided')  write an appropriate comment explaining this line   You said: for col in df1.select\_dtypes(include='number').columns: Q1 = df1[col].quantile(0.25) Q3 = df1[col].quantile(0.75) IQR = Q3 - Q1 lower\_fence = Q1 - 1.5 \* IQR upper\_fence = Q3 + 1.5 \* IQR for row in col: if df1[col][row] > upper\_fence or df1[col][row] < lower\_fence: df1.drop(col, axis=0, inplace=True) # Fences for outliers above\_upper = (df1[col] > upper\_fence).sum() below\_lower = (df1[col] < lower\_fence).sum() print(f"Column: {col}") print(f" Percentage of values above upper range: {above\_upper/len(df1) \* 100:.2f}%") print(f" Percentage of values below lower range: {below\_lower/len(df1) \* 100:.2f}%") print("-" \* 40) Im running this and geting this error: for col in df1.select\_dtypes(include='number').columns: Q1 = df1[col].quantile(0.25) Q3 = df1[col].quantile(0.75) IQR = Q3 - Q1 lower\_fence = Q1 - 1.5 \* IQR upper\_fence = Q3 + 1.5 \* IQR for row in col: if df1[col][row] > upper\_fence or df1[col][row] < lower\_fence: df1.drop(col, axis=0, inplace=True) # Fences for outliers above\_upper = (df1[col] > upper\_fence).sum() below\_lower = (df1[col] < lower\_fence).sum() print(f"Column: {col}") print(f" Percentage of values above upper range: {above\_upper/len(df1) \* 100:.2f}%") print(f" Percentage of values below lower range: {below\_lower/len(df1) \* 100:.2f}%") print("-" \* 40) ChatGPT said: The error is happening because of this line:  for row in col:  col here is a **string** (the column name), not your data. Iterating over it gives you letters, not rows.  Also, df1.drop(col, axis=0, inplace=True) is wrong — axis=0 drops **rows**, but you’re giving it a column name, so Pandas gets confused.  If your goal is to **drop rows that have outliers** for that column, you can do this cleanly:  import pandas as pd  # Loop through numeric columns  for col in df1.select\_dtypes(include='number').columns:  Q1 = df1[col].quantile(0.25)  Q3 = df1[col].quantile(0.75)  IQR = Q3 - Q1  # Fences  lower\_fence = Q1 - 1.5 \* IQR  upper\_fence = Q3 + 1.5 \* IQR  # Drop rows with outliers in this column  df1 = df1[(df1[col] >= lower\_fence) & (df1[col] <= upper\_fence)]  # Count percentages (based on original length before filtering)  above\_upper = (df1[col] > upper\_fence).sum()  below\_lower = (df1[col] < lower\_fence).sum()  print(f"Column: {col}")  print(f" Percentage of values above upper range: {above\_upper / len(df1) \* 100:.2f}%")  print(f" Percentage of values below lower range: {below\_lower / len(df1) \* 100:.2f}%")  print("-" \* 40)  **Key fixes:**   1. Iterate through **rows** with a boolean mask instead of for row in col. 2. Use **(df[col] >= lower) & (df[col] <= upper)** to keep only non-outliers. 3. Remove axis=0/axis=1 confusion — rows are filtered via masking.   If you want, I can rewrite this so it **does not drop any rows** but still reports the **percentages of outliers** for each column in your original dataset. That’s usually better so you don’t mess up later analysis. You said: col\_x = df.select\_dtypes(include=[np.number]).columns for col in df1.select\_dtypes(include='number').columns: Q1 = df1[col].quantile(0.25) Q3 = df1[col].quantile(0.75) IQR = Q3 - Q1 lower\_fence = Q1 - 1.5 \* IQR upper\_fence = Q3 + 1.5 \* IQR for row in col\_x: if df1[col\_x][row] >= upper\_fence or df1[col\_x][row] <= lower\_fence: df1.drop(col\_x, axis=0, inplace=True) # Fences for outliers above\_upper = (df1[col] > upper\_fence).sum() below\_lower = (df1[col] < lower\_fence).sum() print(f"Column: {col\_x}") print(f" Percentage of values above upper range: {above\_upper/len(df1) \* 100:.2f}%") print(f" Percentage of values below lower range: {below\_lower/len(df1) \* 100:.2f}%") print("-" \* 40) c:\Users\user\anaconda3\envs\hit140env\Lib\site-packages\pandas\core\generic.py in ?(self) 1575 @final 1576 def \_\_nonzero\_\_(self) -> NoReturn: -> 1577 raise ValueError( 1578 f"The truth value of a {type(self).\_\_name\_\_} is ambiguous. " 1579 "Use a.empty, a.bool(), a.item(), a.any() or a.all()." 1580 ) ValueError: The truth value of a Series is ambiguous. Use a.empty, a.bool(), a.item(), a.any() or a.all().  I have a csv file that I have imported into python. I want you write a code that will plot a histogram for each of the columns excluding the first one ChatGPT said: Perfect 👍 Thanks for clarifying!  If you want to plot a histogram for **every column in your CSV except the first one**, here’s a clean solution with **pandas** and **matplotlib**:  import pandas as pd  import matplotlib.pyplot as plt  # Load your CSV file  df = pd.read\_csv("your\_file.csv")  # Loop through all columns except the first one  for col in df.columns[1:]:  plt.figure(figsize=(6,4))  plt.hist(df[col].dropna(), bins=30, edgecolor='black')  plt.title(f"Histogram of {col}")  plt.xlabel(col)  plt.ylabel("Frequency")  plt.grid(True, linestyle='--', alpha=0.6)  plt.show() 🔎 Explanation:  * df.columns[1:] skips the first column. * df[col].dropna() ensures missing values don’t break the histogram. * bins=30 gives a reasonably detailed histogram (you can adjust). * Each histogram is shown separately in a new figure.   👉 Do you want me to **save all histograms into one PDF file** (so you can scroll through them easily) or just display them one by one? You said: col = df1.columns[2] plt.figure(figsize=(6,4)) plt.hist(df[col], bins=5, edgecolor='black') plt.title(f"Histogram of {col}") plt.xlabel(col) plt.ylabel("Frequency") plt.grid(True, linestyle='-', alpha=1) plt.show() for this code make it so that the axis steps very 1 unit ChatGPT said: Got it ✅ — you can set the axis tick steps using plt.xticks() (for x-axis) and plt.yticks() (for y-axis).  Since you want the ticks to step every **1 unit**, we can use range() with the min and max values.  Here’s your updated code:  import matplotlib.pyplot as plt  col = df1.columns[2]  plt.figure(figsize=(6,4))  plt.hist(df1[col], bins=5, edgecolor='black')  plt.title(f"Histogram of {col}")  plt.xlabel(col)  plt.ylabel("Frequency")  plt.grid(True, linestyle='-', alpha=1)  # Set axis ticks every 1 unit  plt.xticks(range(int(df1[col].min()), int(df1[col].max()) + 1, 1))  plt.yticks(range(0, df1[col].value\_counts().max() + 2, 1))  plt.show() 🔎 What changed:  * plt.xticks(...): ensures x-axis ticks are spaced by 1. * plt.yticks(...): ensures y-axis ticks are spaced by 1, up to the max frequency.   👉 Do you want me to generalize this so it **automatically applies the 1-unit step for every column’s histogram** in a loop (like you asked earlier for all columns except the first one)? You said: plt.xticks(range(int(df1[col].min()), int(df1[col].max()) + 1, 1)) can you edit this so its structured better between 0 and 102 ChatGPT said: Ah I see 👍 You’d like the x-axis ticks to be **structured from 0 to 102 in steps of 1** (instead of starting at the column’s min and max). That way the histogram is aligned consistently.  You can simply replace the plt.xticks(...) line with:  plt.xticks(range(0, 103, 1)) # ticks from 0 to 102, step 1  So your full code becomes:  import matplotlib.pyplot as plt  col = df1.columns[2]  plt.figure(figsize=(6,4))  plt.hist(df1[col], bins=5, edgecolor='black')  plt.title(f"Histogram of {col}")  plt.xlabel(col)  plt.ylabel("Frequency")  plt.grid(True, linestyle='-', alpha=1)  # Set x-axis ticks between 0 and 102 (step = 1)  plt.xticks(range(0, 103, 1))  # Optional: y-axis ticks every 1 unit as well  plt.yticks(range(0, df1[col].value\_counts().max() + 2, 1))  plt.show()  ⚡ Question: do you want the histogram to **always** use the 0–102 range (like a fixed scale for every column), or just for this specific column? You said: col = df1.columns[2] plt.figure(figsize=(6,4)) plt.hist(df1[col], bins=int(df1[col].max()) + 1, edgecolor='black') plt.title(f"Histogram of {col}") plt.xlabel(col) plt.ylabel("Frequency") plt.xticks(range(int(df1[col].min()), int(df1[col].max()) + 1, 1)) plt.show() please align the bins to the xaxis values ChatGPT said: Got it ✅ — right now your bins are *floating* between x-axis values, which makes the bars not line up with the tick marks.  To **align histogram bins exactly with integer x-axis values**, you should explicitly define the bin edges as integers.  Here’s your corrected code:  import matplotlib.pyplot as plt  import numpy as np  col = df1.columns[2]  plt.figure(figsize=(6,4))  # Define bins so that each integer value has its own bin  bins = np.arange(df1[col].min(), df1[col].max() + 2) # +2 to include the last edge  plt.hist(df1[col], bins=bins, edgecolor='black', align='left')  plt.title(f"Histogram of {col}")  plt.xlabel(col)  plt.ylabel("Frequency")  # Make x-axis ticks line up with integer values  plt.xticks(range(int(df1[col].min()), int(df1[col].max()) + 1, 1))  plt.show() 🔎 Key Fixes  * bins = np.arange(min, max+2) → makes sure each integer has its own bin. * align='left' → aligns bars with tick marks. * Now, if you have values like 0, 1, 2, … 102, each gets a **centered bar** on its tick.   Do you want me to extend this so it automatically applies to **all numeric columns (excluding the first one)** in your CSV, with aligned integer bins for each? You said: bins = np.arange(df1[col].min(), df1[col].max() + 2) # +2 to include the last edge please explain this line ChatGPT said: Great question 🙌 let’s break that line down carefully:  bins = np.arange(df1[col].min(), df1[col].max() + 2)  This is creating the **bin edges** for your histogram. 1. np.arange(start, stop)  * Creates an array of numbers from start up to (but **not including**) stop. * Example: np.arange(0, 5) → [0, 1, 2, 3, 4]  2. df1[col].min()  * The **smallest value** in your column. * This becomes the **first bin edge**.  3. df1[col].max() + 2  * Why +2?    + Normally, to include the maximum value you’d do +1 since np.arange excludes the endpoint.   + But histograms need **one extra edge beyond the maximum value**, otherwise the last bar won’t cover the max.   + So:      - max + 1 → last edge sits **on the maximum**, leaving no room for a bar.     - max + 2 → last edge goes **one step past the maximum**, so the final bin fully includes it.  🔎 Example Suppose df1[col] has values between **0 and 5**.  bins = np.arange(0, 5 + 2) # np.arange(0, 7)  This gives:  [0, 1, 2, 3, 4, 5, 6]   * Bin 1: [0,1) → covers values at 0 * Bin 2: [1,2) → covers values at 1 * … * Bin 6: [5,6) → covers values at 5   So every integer value (0 through 5) has its **own bar**, aligned with the x-axis tick.  ⚡ Would you like me to also show you a quick **visual comparison** (with a toy dataset) of what happens if you use +1 vs +2 so you can *see* the difference in bin alignment? You said: for col in df.columns[1:]: plt.figure(figsize=(6,4)) plt.hist(df[col].dropna(), bins=30, edgecolor='black') plt.title(f"Histogram of {col}") plt.xlabel(col) plt.ylabel("Frequency") plt.grid(True, linestyle='--', alpha=0.6) plt.show() edit this so its line graph ChatGPT said: Got it ✅ — to turn your **histogram plots** into **line graphs**, you just need to plot the frequency counts as a line instead of bars.  Here’s the edited version using plt.plot instead of plt.hist:  import matplotlib.pyplot as plt  import numpy as np  for col in df.columns[1:]:  plt.figure(figsize=(6,4))  # Get histogram counts and bin edges  counts, bin\_edges = np.histogram(df[col].dropna(), bins=30)  # Use midpoints of bins for the x-axis  bin\_centers = (bin\_edges[:-1] + bin\_edges[1:]) / 2  # Plot as line graph  plt.plot(bin\_centers, counts, marker='o', linestyle='-', color='b')  plt.title(f"Line Graph of {col}")  plt.xlabel(col)  plt.ylabel("Frequency")  plt.grid(True, linestyle='--', alpha=0.6)  plt.show() 🔎 Explanation:  * np.histogram() → calculates bin **counts** (y-values) and **bin edges** (x-values). * bin\_centers = midpoints of each bin, so the line sits where the histogram bars would have been. * plt.plot() with marker='o' makes it a line graph with dots at each bin center.   👉 Do you want this line graph to be **smoothed (like a density curve)** instead of jagged lines between bins? You said: I want to plot two line on one line graph. Can you please do it please ChatGPT said: Perfect 👍 You want to overlay **two columns** from your dataframe on the same line graph for comparison.  Here’s an example where we plot the line distributions of **two columns** side by side:  import matplotlib.pyplot as plt  import numpy as np  # Choose the two columns you want to compare  col1 = df.columns[1]  col2 = df.columns[2]  # Histogram for column 1  counts1, bin\_edges1 = np.histogram(df[col1].dropna(), bins=30)  bin\_centers1 = (bin\_edges1[:-1] + bin\_edges1[1:]) / 2  # Histogram for column 2  counts2, bin\_edges2 = np.histogram(df[col2].dropna(), bins=30)  bin\_centers2 = (bin\_edges2[:-1] + bin\_edges2[1:]) / 2  # Plot both on one line graph  plt.figure(figsize=(8,5))  plt.plot(bin\_centers1, counts1, marker='o', linestyle='-', label=col1)  plt.plot(bin\_centers2, counts2, marker='s', linestyle='--', label=col2)  plt.title(f"Line Graph of {col1} vs {col2}")  plt.xlabel("Value")  plt.ylabel("Frequency")  plt.legend()  plt.grid(True, linestyle='--', alpha=0.6)  plt.show() 🔎 What this does:  * Builds histograms for **both columns** using the same binning style. * Plots them together on one figure, with different markers/styles. * plt.legend() makes it clear which line belongs to which column.   ⚡ Do you want me to extend this so it **automatically plots all numeric columns on the same graph** (instead of just picking 2), or just stick with comparing **two at a time**? You said: stop puting the histogram code in there please ChatGPT said: Got it 👍 — no histogram code, just **line graphs directly from the column values**.  If you want to plot **two columns on the same line graph**, here’s the clean version:  import matplotlib.pyplot as plt  # Pick two numeric columns (example: 2nd and 3rd column)  col1 = df.columns[1]  col2 = df.columns[2]  plt.figure(figsize=(8,5))  plt.plot(df[col1], label=col1, marker='o')  plt.plot(df[col2], label=col2, marker='s')  plt.title(f"Line Graph of {col1} and {col2}")  plt.xlabel("Index")  plt.ylabel("Value")  plt.legend()  plt.grid(True, linestyle='--', alpha=0.6)  plt.show() 🔎 Key points:  * This just plots the **raw column values** against the row index. * marker='o' and marker='s' help distinguish the lines. * Works for time series or any sequential data in your CSV.   👉 Do you want the **x-axis** to stay as the default index, or should it use the **first column of your CSV** (maybe a timestamp or ID) as the x-axis? You said: Can you plot the first 500 values of this column please ChatGPT said: Sure thing 👍 If you want to plot the **first 500 values of a column**, here’s the code:  import matplotlib.pyplot as plt  # Select the column you want (example: 2nd column)  col = df.columns[1]  plt.figure(figsize=(8,5))  plt.plot(df[col].head(500), label=col, marker='o')  plt.title(f"First 500 values of {col}")  plt.xlabel("Index")  plt.ylabel("Value")  plt.legend()  plt.grid(True, linestyle='--', alpha=0.6)  plt.show()  🔎 Notes:   * .head(500) takes the first 500 rows. * The x-axis will be the row **index** (0–499).   👉 Do you want the x-axis to instead use the **first column of your CSV** (e.g., if that’s a timestamp), instead of the default index? You said: actually lets plot all of the values but sample every 5th value ChatGPT said: Got it ✅ — if you want to plot **all values but only every 5th sample** (downsampling), you can use Python slicing with iloc[::5]. Here’s the code:  import matplotlib.pyplot as plt  # Select the column (example: 2nd column)  col = df.columns[1]  plt.figure(figsize=(8,5))  plt.plot(df[col].iloc[::5], label=f"{col} (every 5th value)", marker='o')  plt.title(f"{col} (sampled every 5th value)")  plt.xlabel("Index (every 5th)")  plt.ylabel("Value")  plt.legend()  plt.grid(True, linestyle='--', alpha=0.6)  plt.show()  👉 Do you want me to also plot **two columns together with this 5th-sampling**, like you asked before, but on the same line graph? |
| Reflect on how you have used these AI tools, how they have helped with your assignment and the limitations you encountered. | Response:  AI tools have been a tremendous help to us with respect to boosting our efficiency in this project. These tools have allowed us to bring ideas that are subsisting in our brain to fruition quickly. Many of the queries that we would ask a generative AI(Gen AI) tool were linked with accelerating what we already planned to do. As an example, we already know the concept of removing outliers using the interquartile range (IQR) method. However, we would employ an AI to generate the code that would filter through the columns and use the IQR method to remove outliers.  Furthermore, AI has a lot of utility in helping us with libraries that we may not be aware of. As an example, we wanted to plot using the Plotly library in python because of its aesthetic qualities, but we had no experience with it. So, we would ask a Gen AI to produce a specific plot for us using that library. This then allowed us to also learn how to use the library as well.  We also used generative AI to debug our codes as the final program files amounted to being very large. We used the aid of AI to find errors in the code.  So just to summarize we did use Gen AI tools, but it was just to speed the process of generating code but it wasn’t used to give us an approach. Moreover, we also used gen aI to expand our knowledge into using new libraries that helped enlarge our horizons and our skills. |
| Do you maintain an accessible AI prompt history that is verifiable by your lecturer if requested? | Yes / No  If No, provide explanations: |

**Step 3:** Answer the following question only if you selected Declaration 3. You do not need to complete this section if you select Declaration 1 or Declaration 2.

How have you modified the generative AI’s output? You may include as many details as possible, including screenshots.

When we took the output of the Gen AI tool we would usually have to modify it to match our assignment. This was the case when the Gen AI would be producing a generic response that wouldn’t exactly meet our requirements. As an example, if we had asked the Gen AI to produce a plot of us using a create library it would be aware of the necessary column names, axes titles, plot titles and so on. So we would have to accordingly change or even add further code into the response to ensure that the code matched our judgement.

Moreover, we also have enquired using Gen AI to help with the binomtest for the one sample section as well. This was done because we hadn’t learned this concept in class however we had to apply it in the context of the assignment. So we took the output of the Gen AI but had to modify it to match how our own data was structured and what kind of one sample binomtest we wanted to run specifically.

examples of us modifying AI output:

**1.**

**AI -**

import matplotlib.pyplot as plt

# Select the column (example: 2nd column)

col = df.columns[1]

plt.figure(figsize=(8,5))

plt.plot(df[col].iloc[::5], label=f"{col} (every 5th value)", marker='o')

plt.title(f"{col} (sampled every 5th value)")

plt.xlabel("Index (every 5th)")

plt.ylabel("Value")

plt.legend()

plt.grid(True, linestyle='--', alpha=0.6)

plt.show()

**Ours -**

col1 = df\_cleaned.columns[3]

col2 = df\_cleaned.columns[4]

plt.figure(figsize=(8,5))

plt.plot(df\_cleaned[col1][::40], label=col1, marker='o')

plt.plot(df\_cleaned[col2][::40], label=col2, marker='s')

plt.title(f"Line Graph of {col1} and {col2}")

plt.xlabel("Index")

plt.ylabel("Value")

plt.legend()

plt.grid(True, linestyle='--', alpha=0.6)

plt.show()

**2.**

**AI -**

import matplotlib.pyplot as plt

import numpy as np

col = df1.columns[2]

plt.figure(figsize=(6,4))

# Define bins so that each integer value has its own bin

bins = np.arange(df1[col].min(), df1[col].max() + 2) # +2 to include the last edge

plt.hist(df1[col], bins=bins, edgecolor='black', align='left')

plt.title(f"Histogram of {col}")

plt.xlabel(col)

plt.ylabel("Frequency")

# Make x-axis ticks line up with integer values

plt.xticks(range(int(df1[col].min()), int(df1[col].max()) + 1, 1))

plt.show()

**Ours -**

col = df1.columns[2]

bins = np.arange(df1[col].min(), df1[col].max() + 2) # +2 to include the last edge

plt.figure(figsize=(6,4))

plt.hist(df1[col], bins=bins, edgecolor='black')

plt.title(f"Histogram of {col}")

plt.xlabel(col)

plt.ylabel("Frequency")

plt.xticks(range(int(df1[col].min()), int(df1[col].max()) + 2, 1))

plt.show()

**Student Signatures**

| Student ID | Full Name | Signature |
| --- | --- | --- |
| S393516 | **Syed Haroon Ahmad** |  |
| s393148 | **Mehraab Ferdouse** |  |
| s385833 | **Simbarashe Mutyambizi** | **Simbarashe Mutyambizi** |
| s395696 | **Saugat Poudel** |  |
|  |  |  |

Date:\_\_\_\_\_\_7/09/2025\_\_\_\_\_\_\_\_

The contents of this AI Usage Declaration form have been adapted from:

* University of Birmingham (2025), *Acknowledging and Citing the Use of Generative AI by Student*, accessed on 4 Mar 2025. Link: <https://www.birmingham.ac.uk/libraries/education-excellence/gai/acknowledging-gai-by-students>
* University of Cambridge (2025), *Template Declaration of the Use of Generative Artificial Intelligence*, accessed on 4 Mar 2025. Link: <https://www.cshss.cam.ac.uk/education/generative-artificial-intelligence-ai-and-scholarship/template-declaration-use-generative>