Algorithms for Data Science - ADSA H6013
Practical Report (Part 3, Regression / Association)

Mariem Nsiri
MSc in Applied Data Science and Analytics
Technological University Dublin

Declaration: I herby certify that this material, which I now submit for assessment on the programme of study leading to the award of MSc in Computing in Applied Data Science and Analytics, TU Dublin, is entirely my own work except where otherwise stated, and has not been submitted for assessment for an academic purpose at this or any other academic institution other than in partial fullfilment of the requirements of that stated above.

Mariem Nsiri

January 7, 2025

1 Regression

This section aims to build and evaluate a regression model to predict an individual's height using skeletal measurements in the dataset. The dataset contains 507 observations, each with different numeric values representing bone dimensions, such as shoulder width (`biacromial`), pelvic breadth (`pelvicBreadth`), and wrist diameter (`wristDiam`).

The dataset has a total of 12 columns, with each column representing a different feature or attribute. The dataset contains no missing values, as all 507 entries are complete. The attributes are of various types, with 11 columns being numerical (of type `float64`) and one column, `age`, being an integer (type `int64`). The dataset's total memory usage is 47.7 KB (see the outputs below).

Height was chosen as the dependent variable for the following reasons:

- **Direct link to bone structure:** Height is closely related to bone structure, making it a logical choice for a model based on skeletal measurements.
- **Predictability and clarity:** Height is less affected by external factors like age or weight, making it more predictable and easy to understand in the model.
- **Real-world applications:** Predicting height from skeletal measurements is useful in fields such as anthropology, ergonomics, and forensic science.

The main goal is to create a model that provides accurate predictions while being simple and fitting well with the dataset. This section will outline the methodology, the results from the regression analysis, and how well the model predicts height based on the available features.

After explaining the choice of the dependent variable, the next section will go over the methodology for building and evaluating the regression model.

```
import pandas as pd
smd = pd.read_csv('data/assessment_regression_dataset.csv')
print(smd.shape)
smd.info()
```

```
(507, 12)
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 507 entries, 0 to 506
Data columns (total 12 columns):
 #   Column          Non-Null Count  Dtype
---  ------          --------------  -----
 0   biacromial      507 non-null    float64
 1   pelvicBreadth   507 non-null    float64
 2   bitrochanteric  507 non-null    float64
 3   chestDepth      507 non-null    float64
 4   chestDiam       507 non-null    float64
 5   elbowDiam       507 non-null    float64
 6   wristDiam       507 non-null    float64
 7   kneeDiam        507 non-null    float64
 8   ankleDiam       507 non-null    float64
 9   age             507 non-null    int64
 10  weight          507 non-null    float64
 11  height          507 non-null    float64
dtypes: float64(11), int64(1)
memory usage: 47.7 KB
```

The output of `print(smd.describe())` appears to be normally distributed, with some variation across key features. The following observations can be made on some independent attributes:

- **Height and weight:** There is significant variation in both height and weight, with mean values reflecting typical human dimensions. This suggests that these features will be useful for predictive modeling, although other factors like gender or body type could also play a role.
- **Age:** Age varies widely in the dataset, but it is not expected to have a strong direct correlation with height after adulthood. The variation in height remains relatively constant across age groups, supporting this observation.
- **Pelvic breadth:** Pelvic breadth shows moderate variability, indicating that skeletal features can differ from person to person, though not drastically.

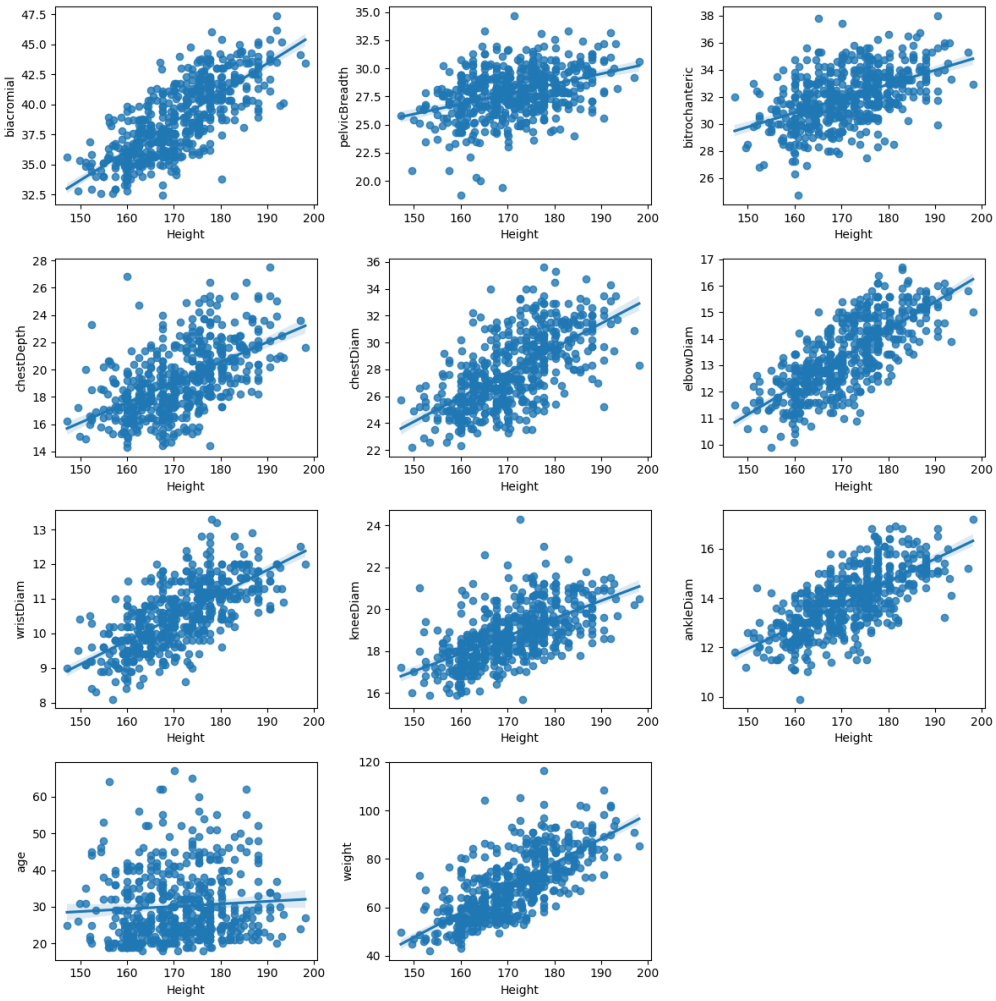

Figure 1: Regression plots of skeletal measurements against height

1.1 Features-`height` relationships visualization

To explore the relationships between height and various skeletal measurements, regression plots were created. The following code generates regression plots for each feature in relation to height. A total of 11 regression plots were produced, each comparing height with one of the skeletal measurements. A regression line was fitted to each plot, showing the relationship between height and the corresponding feature (see Figure 1).

The slope of the lines for most features indicates a monotonic relationship, with height increasing consistently as the skeletal measurements increase. However, the nearly flat line for the `age` attribute shows that there is no significant relationship between age and height, which is expected as height growth typically stops after adulthood. This suggests that age is unlikely to be an important factor in predicting height, which may affect the selection of variables for future regression models.

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

smd_atts_to_plot = ['biacromial', 'pelvicBreadth', 'bitrochanteric', 'chestDepth', 'chestDiam', 'elbowDiam',
                    'wristDiam', 'kneeDiam', 'ankleDiam', 'age', 'weight']

plt.figure(figsize=(12, 12))

for i, col in enumerate(smd_atts_to_plot, 1):
  plt.subplot(4, 3, i)
  sns.regplot(data=smd, x='height', y=col)
  plt.title(f'Regression plot: height vs {col}')
  plt.xlabel('Height')
  plt.ylabel(col)

plt.tight_layout()
plt.show()
```

1.2 Model implementations

This section outlines the implementation of the regression model used to predict height based on various skeletal measurements. Different techniques and approaches are discussed, including the selection of features, the training process, and the evaluation of model performance. The goal is to develop a reliable model that can accurately predict height using the available data.

```
from sklearn.linear_model import LinearRegression
# Loading data and creating the model
X_smd = smd[smd.columns.difference(['height'])]
y_smd = smd['height']
lr = LinearRegression()
# Calling the cross-validation function
ad.xval_regress(lr, X_smd[['biacromial']], y_smd, lin=True, return_=True)
# Fit the model to the entire dataset
lr.fit(X_smd[['biacromial']], y_smd)
# Print model coefficient and intercept
print("Model coefficient: ", lr.coef_)
print("Model intercept: ", lr.intercept_)
# Predict on the same dataset
y_smd_pred = lr.predict(X_smd[['biacromial']])
# Plotting True vs Predicted
plt.figure(figsize=(8,6))
sns.regplot(x=y_smd, y=y_smd_pred, scatter_kws={'s': 10})
plt.xlabel("True Height")
plt.ylabel("Predicted Height")
plt.show()
```

```
Average cross-validated RMSE: 6.36 +/- 2.31
Average cross-validated R-squared: 0.24 +/- 0.21
```

The model was tested using cross-validation, and the results show the average root mean square error (RMSE) of 6.36, with a standard deviation of 2.31. This indicates that, on average, the model's predictions are off by about 6.36 units in height. The R-squared value is 0.24, meaning the model explains only 24% of the variance in the data. This suggests that the model does not capture most of the variation in height based on the given feature (biacromial width).

Although the model shows a weak correlation, it offers insight into how skeletal measurements like biacromial width can be used to estimate height. However, the low R-squared value indicates that additional features or a more complex model could improve prediction accuracy.

```
Coefficients across all folds....
Coeficient:  [2.4644931] , Intercept:  75.54736906007341
Coeficient:  [2.17836973] , Intercept:  86.12627793443927
Coeficient:  [2.26872404] , Intercept:  83.04547499850237
Coeficient:  [2.22108238] , Intercept:  85.48561870760186
Coeficient:  [2.3445875] , Intercept:  80.13523816961542
Model coefficient:  [2.30302635]
Model intercept:  81.7600184502303
```

The new model, which uses all the available features, performs better than the previous one. The average RMSE has decreased from 6.36 to 5.53, and the R-squared value has increased from 0.24 to 0.38, showing that the model now predicts height more accurately. The coefficients for each feature also indicate different impacts on height, with some features increasing height and others decreasing it (see Figure 2, left). Although the model's performance has improved, there is still room for better accuracy. Future improvements could include exploring new features, using more advanced modeling techniques, or adding more data to the model. This could help further increase the model's ability to predict height.

```
lr = LinearRegression()
# Calling the cross-validation function
ad.xval_regress(lr, X_smd, y_smd, lin=True, return_=True)
# Fit the model to the entire dataset
lr.fit(X_smd, y_smd)
# Print model coefficient and intercept
print("Model coefficient: ", lr.coef_)
print("Model intercept: ", lr.intercept_)
# Predict on the same dataset
y_smd_pred = lr.predict(X_smd)
# Plotting True vs Predicted
plt.figure(figsize=(8,6))
sns.regplot(x=y_smd, y=y_smd_pred, scatter_kws={'s': 10})
#plt.title("Regression Plot: True vs Predicted Height")
plt.xlabel("True Height")
plt.ylabel("Predicted Height")
plt.show()
```

```
Average cross-validated RMSE: 5.53 +/- 1.89
Average cross-validated R-squared: 0.38 +/- 0.19
Coefficients across all folds....
Coeficient:  [-0.05788615  1.81558372  1.49961002 -0.32644281 -0.54601203 -0.91963787
  1.61991078 -0.98825048  0.40109629  0.31720243 -0.04772787] , Intercept:  100.3061332637363
Coeficient:  [-0.08741576  1.22472675  1.13144938 -0.07747291 -0.22911329 -0.49018651
  1.72088039 -1.26237001  0.4471125   0.27770367  0.48406314] , Intercept:  97.4759191426929
Coeficient:  [-0.09939468  1.49883345  1.34516396 -0.06292214 -0.12483952 -0.76081592
  0.99052895 -1.15808844  0.48159033  0.24391376  1.11645578] , Intercept:  93.33635619678614
Coeficient:  [-0.06163883  1.59461772  1.39370103 -0.01528579 -0.26458542 -0.8090716
  1.76734593 -0.71317514  0.4368369   0.250669   -0.13297236] , Intercept:  86.62830454650994
Coeficient:  [-0.06954862  1.51472292  1.17635656  0.05132931 -0.18227139 -0.66140949
  1.24195274 -1.04695412  0.321673    0.28601357  0.53948141] , Intercept:  95.82200167332626
Model coefficient:  [-0.07564375  1.50470175  1.3165914  -0.09472071 -0.26982115 -0.72184791
  1.4637309  -1.04046371  0.41063737  0.27467773  0.41956151]
Model intercept:  95.0123452907645
```

Figure 2 (right) presents a 3D visualization of the relationship between two independent variables, `biacromial` and `pelvicBreadth`, and the dependent variable `height`. The plot displays the data points as blue markers, while a red regression plane represents the predicted values of height based on these variables. The axes are labeled for clarity, showing `biacromial`, `pelvicBreadth`, and `height`. The plot's layout is designed to ensure a clear view with an aspect ratio that highlights the data's structure, offering an interactive representation of the regression model's behavior.

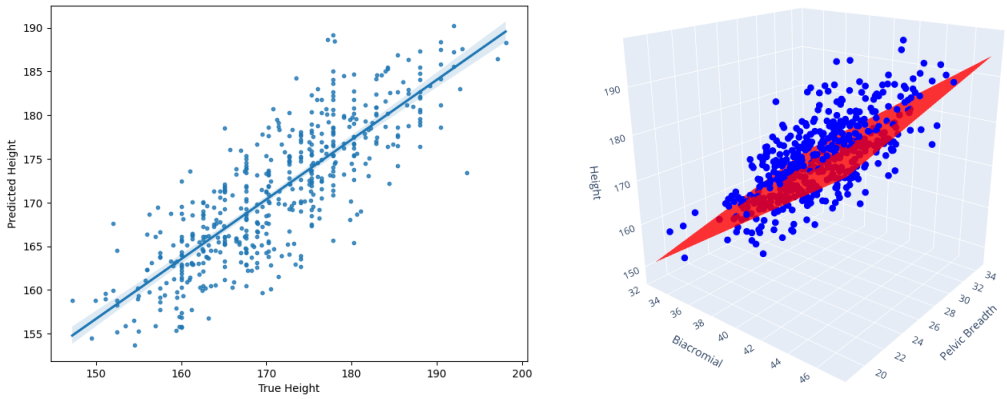

Figure 2: Regression plot: true vs predicted height (left); 3D regression plane: `biacromial`, `pelvicBreadth` vs `height` (right)

```
from sklearn.linear_model import ElasticNet
# Define ElasticNet model
elastic_net = ElasticNet()
# Hyperparameter grid to search
param_grid = {
  'alpha': [0.1, 0.5, 1, 10], # Regularization strength
  'l1_ratio': [0.1, 0.5, 0.9] # The ratio between Lasso (L1) and Ridge (L2)
}
# Grid search with cross-validation
grid_search =
  GridSearchCV(estimator=elastic_net, param_grid=param_grid,
               cv=5,scoring='neg_mean_squared_error')

grid_search.fit(X_smd, y_smd) # Make sure to use the proper training data (X_smd, y_smd)
# Best parameters from GridSearch
best_params = grid_search.best_params_
print(f"Best ElasticNet parameters: {best_params}")
# Model performance
best_model = grid_search.best_estimator_
# Use the best model to perform cross-validation using xval_regress
print("\nPerforming cross-validation using the best ElasticNet model:")
ad.xval_regress(best_model, X_smd, y_smd, lin=True)
```

```
Average cross-validated RMSE: 5.53 +/- 1.89
Average cross-validated R-squared: 0.38 +/- 0.19
Coefficients across all folds.... Model intercept:  95.0123452907645
Coeficient:  [-0.05788615  1.81558372  1.49961002 -0.32644281 -0.54601203 -0.91963787
  1.61991078 -0.98825048  0.40109629  0.31720243 -0.04772787] , Intercept:  100.3061332637363
Coeficient:  [-0.08741576  1.22472675  1.13144938 -0.07747291 -0.22911329 -0.49018651
  1.72088039 -1.26237001  0.4471125   0.27770367  0.48406314] , Intercept:  97.4759191426929
Coeficient:  [-0.09939468  1.49883345  1.34516396 -0.06292214 -0.12483952 -0.76081592
  0.99052895 -1.15808844  0.48159033  0.24391376  1.11645578] , Intercept:  93.33635619678614
Coeficient:  [-0.06163883  1.59461772  1.39370103 -0.01528579 -0.26458542 -0.8090716
  1.76734593 -0.71317514  0.4368369   0.250669   -0.13297236] , Intercept:  86.62830454650994
Coeficient:  [-0.06954862  1.51472292  1.17635656  0.05132931 -0.18227139 -0.66140949
  1.24195274 -1.04695412  0.321673    0.28601357  0.53948141] , Intercept:  95.82200167332626
Model coefficient:  [-0.07564375  1.50470175  1.3165914  -0.09472071 -0.26982115 -0.72184791
  1.4637309  -1.04046371  0.41063737  0.27467773  0.41956151]
```

The three models tested—linear regression, Ridge, and ElasticNet—each offer different performances when predicting height. The linear regression model shows an RMSE of 5.53 and an R-squared value of 0.38, indicating a moderate fit but leaving room for improvement. The coefficients across the folds are stable, suggesting that the relationships between the independent variables and height are consistent, although the model only explains 38% of the variance, pointing to factors not captured in the current model.

The Ridge regression model, with various alpha values tested, provides a consistent RMSE of 5.53 and an R-squared value of 0.38, similar to the linear regression model. Despite testing different alphas, the model's performance does not show significant improvements, as the RMSE and R-squared values remain stable. Ridge regression doesn't significantly outperform linear regression, suggesting that the regularization technique does not add substantial benefit in this case, particularly for height prediction.

The ElasticNet model, using the parameters $\alpha = 0.1$ and $\texttt{l1\_ratio} = 0.1$, performs slightly better than both linear regression and Ridge. It achieves an RMSE of 5.51 and an R-squared value of 0.39, which suggests that it captures the data's underlying patterns more effectively than the other two models. ElasticNet benefits from combining both L1 and L2 regularization, improving the model's performance by managing feature relationships more efficiently. Although it still only explains about 39% of the variance, ElasticNet proves to be the most optimal model for height prediction among the three tested models, though further improvements can be made with additional features or tuning.

2 Association

This section explores the relationships between books purchased on an online platform, using a dataset of 700 transactions. Seven books, each representing a different topic, are encoded as binary variables, with 1 indicating a purchase and 0 indicating no purchase. The goal is to uncover association rules that reveal relationships among the books using three evaluation metrics. The analysis generates frequent itemsets based on a minimum support threshold, extracts association rules, and identifies the most significant rules using percentile-based criteria.

```python
import pandas as pd
import matplotlib.pyplot as plt
# Load the dataset
data = pd.read_csv('data/assessment_association_dataset.csv' )
# Display the first few rows of the dataset
print("Dataset Preview:")
print(data.head())
# Check for missing values
print("\nMissing Values:")
print(data.isnull().sum())
# Get basic statistics of the dataset
print("\nDataset Summary:")
print(data.describe())
# Check the dataset's structure
print("\nDataset Structure:")
print(data.info())
# Analyze the distribution of purchases for each book
print("\nItem Purchase Counts:")
print(data.sum())
```

```
Dataset Structure:
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 700 entries, 0 to 699
Data columns (total 7 columns):
 #   Column                                  Non-Null Count  Dtype
---  ------                                  --------------  -----
 0   Student Social Life Attractions in Dublin  700 non-null    int64
 1   Python Programming for Beginners        700 non-null    int64
 2   Introduction to Data Mining             700 non-null    int64
 3   Quick Meals for Busy Students           700 non-null    int64
 4   Introduction to Student Psychology      700 non-null    int64
 5   How to Deal with Procrastination        700 non-null    int64
 6   Ethics and Morality                     700 non-null    int64
dtypes: int64(7)
memory usage: 38.4 KB
```

```
Item Purchase Counts:
Student Social Life Attractions in Dublin    358
Python Programming for Beginners             259
Introduction to Data Mining                  253
Quick Meals for Busy Students                 61
Introduction to Student Psychology           140
How to Deal with Procrastination             311
Ethics and Morality                           23
dtype: int64
```

The book titled 'Student Social Life Attractions in Dublin" was the most frequently purchased, appearing in 51.1% of all transactions, making it the clear favorite among users. In contrast, the book 'Ethics and Morality" had the lowest purchase count, appearing in only 3.3% of transactions, indicating a significantly lower level of interest. This disparity suggests potential opportunities for strategic actions. For instance, businesses could consider bundling the popular books together to boost sales further or implementing targeted marketing campaigns to increase the visibility of the less frequently purchased titles. By identifying such patterns, businesses can optimize their promotional strategies and allocate resources more effectively, either by capitalizing on the popularity of certain books or by fostering greater interest in those that are currently underperforming.

2.1 Frequent itemset mining

```python
from mlxtend.frequent_patterns import apriori, association_rules
# Convert the dataset to a boolean DataFrame (binary encoding where 1
# indicates purchase and 0 indicates no purchase)
data = data.astype(bool)
# Specify the minimum support threshold (e.g., 0.1)
min_support = 0.1
# Use the Apriori algorithm to find frequent itemsets
frequent_itemsets = apriori(data, min_support=min_support, use_colnames=True)
# Extract association rules with 'lift' as the evaluation metric
rules = association_rules(frequent_itemsets, metric="lift", min_threshold=1, num_itemsets=None)
# Display the frequent itemsets
print("\nFrequent Itemsets:")
print(frequent_itemsets)
# Display the association rules
print("\nAssociation Rules:")
print(rules)
```

```
    support                                        itemsets
0   0.511429        (Student Social Life Attractions in Dublin)
1   0.370000                  (Python Programming for Beginners)
2   0.361429                       (Introduction to Data Mining)
3   0.200000                (Introduction to Student Psychology)
4   0.444286                  (How to Deal with Procrastination)
5   0.204286  (Python Programming for Beginners, Student Soc...
6   0.212857  (Introduction to Data Mining, Student Social L...
7   0.142857  (Introduction to Student Psychology, Student S...
8   0.265714  (How to Deal with Procrastination, Student Soc...
9   0.218571  (Python Programming for Beginners, Introductio...
10  0.112857  (Python Programming for Beginners, Introductio...
11  0.217143  (Python Programming for Beginners, How to Deal...
12  0.122857  (Introduction to Student Psychology, Introduct...
13  0.224286  (How to Deal with Procrastination, Introductio...
14  0.150000  (How to Deal with Procrastination, Introductio...
15  0.144286  (Python Programming for Beginners, Introductio...
16  0.135714  (Python Programming for Beginners, How to Deal...
17  0.152857  (How to Deal with Procrastination, Introductio...
18  0.114286  (How to Deal with Procrastination, Introductio...
19  0.148571  (Python Programming for Beginners, How to Deal...
20  0.107143  (Python Programming for Beginners, How to Deal...
```

```
Association Rules:
                                      antecedents
0                 (Python Programming for Beginners)
1         (Student Social Life Attractions in Dublin)
2                      (Introduction to Data Mining)
3         (Student Social Life Attractions in Dublin)
4               (Introduction to Student Psychology)
..                                              ...
59  (Introduction to Data Mining, Student Social L...
60                (Python Programming for Beginners)
61                (How to Deal with Procrastination)
62                     (Introduction to Data Mining)
63        (Student Social Life Attractions in Dublin)

                                      consequents  antecedent support
0         (Student Social Life Attractions in Dublin)            0.370000
1                 (Python Programming for Beginners)            0.511429
2         (Student Social Life Attractions in Dublin)            0.361429
3                      (Introduction to Data Mining)            0.511429
4         (Student Social Life Attractions in Dublin)            0.200000
..                                              ...                 ...
59  (Python Programming for Beginners, How to Deal...            0.212857
...
```

The generated association rules provide insightful relationships between the various books purchased on the platform. For instance, the rule showing a lift of 1.396648 between *Introduction to Student Psychology* and *Student Social Life Attractions in Dublin* suggests that customers who purchase the latter are more likely to purchase the former as well. The confidence values, such as 0.714286 for the rule *Introduction to Student Psychology* $\rightarrow$ *Student Social Life Attractions in Dublin*, further confirm this relationship. However, while some rules indicate strong associations, such as those with high lift values, many rules also exhibit relatively low lift and confidence, which might not be as actionable for marketing or promotional strategies.

2.2 Systematic rule generation and threshold-based filtering

In this section, we apply a systematic approach to generate association rules from the dataset and filter them using thresholds determined by percentile values. Initially, all association rules are generated from the frequent itemsets using the Apriori algorithm. Then, a reasonable threshold for three different metrics—confidence, lift, and leverage—is established based on the 75th percentile of their distributions. Finally, the dataset is filtered by these thresholds to produce three new sets of rules, each focusing on one of the chosen metrics. This approach helps refine the analysis and highlights the most relevant and significant relationships within the data.

```python
import numpy as np
# Specify the minimum support threshold (e.g., 0.1)
min_support = 0.1
# Use the Apriori algorithm to find frequent itemsets
frequent_itemsets = apriori(data_bools, min_support=min_support, use_colnames=True)
# Extract all association rules
rules = association_rules(frequent_itemsets, metric="lift", min_threshold=0, num_itemsets=None)
# Display the frequent itemsets
print("\nFrequent Itemsets:")
print(frequent_itemsets)
# Display all association rules
print("\nAll Association Rules:")
print(rules)
# Define a systematic approach to determine thresholds using percentiles
metrics = ['confidence', 'lift', 'leverage'] # Specify the metrics to use
thresholds = {}
# Calculate percentile-based thresholds for each metric
for metric in metrics:
  thresholds[metric] = np.percentile(rules[metric], 75) # Using the 75th percentile as an example threshold
```

```
# Generate new sets of rules for each metric based on the determined thresholds
rules_by_metric = {}
for metric, threshold in thresholds.items():
  rules_by_metric[metric] = rules[rules[metric] >= threshold]
# Display the thresholds and the new sets of rules
print("\nThresholds:")
print(thresholds)
for metric, filtered_rules in rules_by_metric.items():
print(f"\nRules filtered by {metric} (threshold: {thresholds[metric]:.2f}):")
print(filtered_rules)
```

```
...
Thresholds:
'confidence': 0.6607031347570875, 'lift': 1.8352310195146053, 'leverage': 0.06114285714285713
...
```

The analysis revealed key patterns and relationships among the frequent itemsets and association rules. Frequent itemsets with high support values, particularly above 0.35, show significant importance. For instance, *Student Social Life Attractions in Dublin* achieved the highest support of 0.511, followed by *How to Deal with Procrastination* (0.444), and *Python Programming for Beginners* (0.370). Similarly, combined itemsets such as *Student Social Life Attractions in Dublin* with *How to Deal with Procrastination* also demonstrated meaningful support (0.265). Filtered association rules based on a confidence threshold of 0.66 identify strong connections. For example, the rule *Introduction to Student Psychology* $\rightarrow$ *How to Deal with Procrastination* has a confidence of 0.75 and a lift of 1.688, indicating its relevance. Additionally, the rule *Python Programming for Beginners and How to Deal with Procrastination* $\rightarrow$ *Introduction to Data Mining* showed a lift of 1.893 and a confidence of 0.684, making it another notable finding. Finally, the rules filtered by a lift threshold of 1.84 highlight the most impactful relationships. For instance, the rule *Student Social Life Attractions in Dublin and Python Programming for Beginners* $\rightarrow$ *Introduction to Data Mining* achieved a lift of 1.954 and a confidence of 0.706. These high-lift rules underscore valuable insights into the interconnections between itemsets and suggest potential areas for deeper exploration.

```
# For each metric-based set, filter rules and select one interesting rule using support-confidence framework
for metric, filtered_rules in rules_by_metric.items():
  print(f"\nRules filtered by {metric} (threshold: {thresholds[metric]:.2f}):")
  # Select one rule based on a combination of support and confidence
  selected_rule = filtered_rules[(filtered_rules['support'] >= 0.1) & (filtered_rules['confidence'] >= 0.7)].iloc[0]
  # Extract details of the selected rule
  antecedents = selected_rule['antecedents']
  consequents = selected_rule['consequents']
  support = selected_rule['support']
  confidence = selected_rule['confidence']
  lift = selected_rule['lift']
  leverage = selected_rule['leverage']
  conviction = selected_rule['conviction']
  # Display and interpret the rule
  print(f"Selected Rule: {antecedents} -> {consequents}")
  print(f"Support: {support:.2f}, Confidence: {confidence:.2f}, Lift:
        {lift:.2f}, Leverage: {leverage:.2f}, Conviction: {conviction:.2f}")
  # Interpretation of the rule
  print("Interpretation:")
  print(f"If {antecedents} is purchased, {consequents} is likely to be bought as well.")
  print(f"Confidence: {confidence:.2f} ({confidence*100}% of transactions).")
  print(f"Lift: {lift:.2f} suggests a stronger association than random chance.")
  print(f"Leverage: {leverage:.2f} shows significant relationship.")
  print(f"Conviction: {conviction:.2f} reflects the likelihood of {consequents} being purchased with {antecedents}.")
  print("\n")
```

2.3 Rule selection and interpretation

This section filters association rules using confidence, lift, and leverage with specified thresholds. One interesting rule is selected for each metric based on support and confidence. The selected rules are then displayed and interpreted to understand their significance and relationships between the books (see the code above). To save space the output is not shown and explained below.

Association rules based on the confidence metric are analyzed. The output shows that purchasing "Introduction to Student Psychology" has a 71% chance of leading to the purchase of "Student Social Life Attractions in Dublin". With a confidence of 0.71, a lift of 1.40, and a leverage of 0.04, this relationship is stronger than random chance, and a conviction of 1.71 indicates a higher likelihood of both books being bought together.

Next, rules filtered by the lift metric are examined. It is revealed that buying both "Student Social Life Attractions in Dublin" and "Python Programming for Beginners" increases the likelihood of purchasing "Introduction to Data Mining". The high confidence of 0.71 and lift of 1.95 suggest a strong connection, with a leverage of 0.07 and a conviction of 2.17 indicating a high probability of all three books being bought together.

Finally, the rules filtered by leverage show that purchasing "Introduction to Student Psychology" greatly increases the likelihood of buying "How to Deal with Procrastination". With 75% confidence, a lift of 1.69, a leverage of 0.06, and a conviction of 2.22, this rule indicates a strong association between the two books and a high likelihood of both being purchased together.