Table of Contents

Introduction
Association rule mining, particularly the extraction of patterns from vast datasets, is pivotal in data analytics. The technique reveals hidden relationships among variables, offering insights that might be obscured in large volumes of data (Telikani et al., 2020). This research employs the Apriori algorithm to discern patterns associated with individuals earning over $50K annually.

Salary often serves as a key metric in socio-economic research, reflecting an individual's financial health and providing a lens into broader socio-economic trends and disparities (Sasaki, 2022). Understanding the determinants of earning above $50K, especially without higher education, is essential. Such insights can guide policymakers, employers, and educators to develop strategies that promote non-traditional pathways to financial success.

The primary objective of the study extends beyond its academic scope. The discovered patterns and linkages have practical significance in the real world, informing decision-making and strategic planning in diverse areas of society. By employing both R and Python, this research guarantees the reliability and validity of its conclusions, capitalising on the unique advantages offered by each programming language to offer a thorough and all-encompassing examination of the available data.

Motivation
This study is motivated by multiple factors. In contemporary times characterised by escalating tuition rates and student loan burdens, it is increasingly vital to comprehend alternative pathways to achieve financial prosperity beyond acquiring university degrees. If specific patterns or traits frequently result in a wage over $50,000 without the possession of a degree, these areas could be deemed as focal points for adults, educators, and legislators alike (Li et al., 2013).

From a business perspective, recognising specific skills or experiences correlating with higher earnings can guide recruitment and training strategies. Additionally, pinpointing demographics with higher earning potential may reveal areas of socio-economic progress or biases in employment. Such insights are invaluable for social scientists and policymakers.

Implementation of the Apriori Algorithm
This study utilised the Apriori algorithm, a widely recognised association rule mining approach, to identify patterns associated with salaries over $50K. Apriori operates by first recognising the frequent individual items inside the database. It then proceeds to expand these recurring things into more extensive itemsets as long as these item sets occur with a significant frequency within the database (Hegland, 2007). The concept of support determines the measurement of item set frequency, whereas the concept of

confidence is employed to eliminate rules. The two frameworks are utilised here to build the solution, which is Python and R.

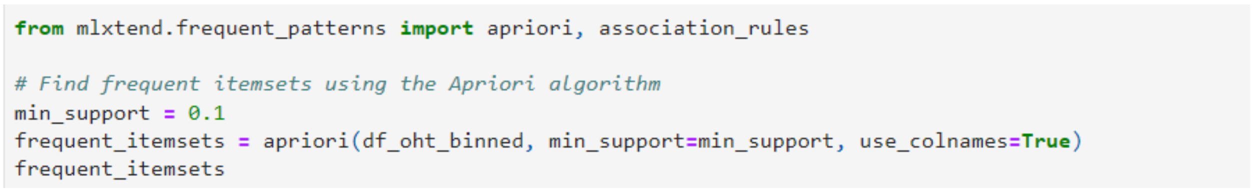
The mlxtend package is widely utilised for mining association rule databases in the Python programming language (Room). The tool offers an intuitive user experience and a diverse range of features that facilitate the application of the Apriori algorithm.

Before the deployment of the Apriori algorithm, data preprocessing was undertaken, as emphasised by García et al. (2015). In Python, the pandas library was instrumental in streamlining this process. Of particular note is the meticulous handling of missing values. Duplicate records were identified and eliminated, ensuring data integrity. Subsequently, binning techniques were applied to transform numerical columns into categorical ones. Upon completing the preprocessing steps, the Apriori algorithm was executed using the mlxtend package.

Conversely, the Arules package stands out for the R environment as a potent tool tailored for association rule mining, encompassing the Apriori algorithm, as highlighted by Hahsler et al. (2023). Data preparation in R is equally pivotal, mirroring the rigour observed in Python. While the preparatory steps align with those in Python, R adopts distinct methodologies to optimise the dataset for association rule mining. The Apriori algorithm, recognised for its widespread application in data mining within the realm of association rules, can be seamlessly integrated into the R framework through the Arules package.

Core Functionalities of the Code

In examining the Apriori algorithm, we employed two leading programming languages, Python and R, to navigate its intricacies. Within the Python environment, the mlxtend library emerges as a widely acknowledged tool tailored for this endeavour. The pivotal apriori() function facilitates the generation of frequent itemsets from one-hot encoded data frames, guided by specified minimum support thresholds. The association_rules() function, housed within the same library, also crafts association rules derived from these frequent itemsets. This functionality equips us with the capability to evaluate associations through metrics like "support", "confidence", and "lift", thereby refining our insights to meet specific criteria. The subsequent section presents the Python code exemplifying the use of the Apriori method.



```python
from mlxtend.frequent_patterns import apriori, association_rules

# Find frequent itemsets using the Apriori algorithm
min_support = 0.1
frequent_itemsets = apriori(df_oht_binned, min_support=min_support, use_colnames=True)
frequent_itemsets
```

```
# Extract rules with a minimum confidence (for demonstration, I'm using 0.3, but you can adjust this as needed)
rules = association_rules(frequent_itemsets, metric="confidence", min_threshold=0.2)
```

```
rules_below_or_equal_50K = rules[rules['consequents'] == frozenset({'salary_ >50K'})].sort_values(by="confidence", ascending=False)
print("\nRules for salary >50K:\n")
for index, rule in rules_below_or_equal_50K.head(10).iterrows():
    print(f"If {set(rule['antecedents'])} then {set(rule['consequents'])}")
    print(f"Support: {rule['support']:.3f}, Confidence: {rule['confidence']:.3f}, Lift: {rule['lift']:.3f}")
    print("-" * 50)
```

The R programming language boasts robust capabilities for association rule mining, predominantly powered by the arules package. Central to this package is the apriori() function, adept at discerning association rules and frequent itemsets. To ensure compatibility, input data should be formatted as transactions. Fine-tuning of results can be achieved by adjusting parameters like support and confidence. For a clear visualisation of the mined rules, R introduces the inspect() function, streamlining the presentation of results. The subsequent section delineates the R code, exemplifying this process.

```
# Convert the dataset to transaction format
transactions <- as(df_oht_binned, "transactions")

# Apply the Apriori algorithm
rules <- apriori(transactions, parameter = list(supp = 0.1, conf = 0.2))
```

```
# Where salary is greater than 50K

# 3. Filter Rules
# Filter rules for consequences where the salary is greater than 50K.
salary_rules <- subset(rules, rhs %in% "salary...50K=1")

top_rules10 <- head(sort(salary_rules, by="confidence"), 10)
inspect(top_rules10)
```

Both Python and R offer comprehensive frameworks for conducting association rule mining. While the mlxtend library in Python gains traction for data manipulation due to its modern interface, R's arules package garners acclaim for its efficiency and purpose-built features tailored for association rules. The choice between these languages and their respective libraries often hinges on the specific requirements of the analysis and the researcher's familiarity with the tools. Our study harnessed the combined strengths of both platforms, ensuring a multifaceted analysis that capitalised on the unique offerings of each environment.

Results

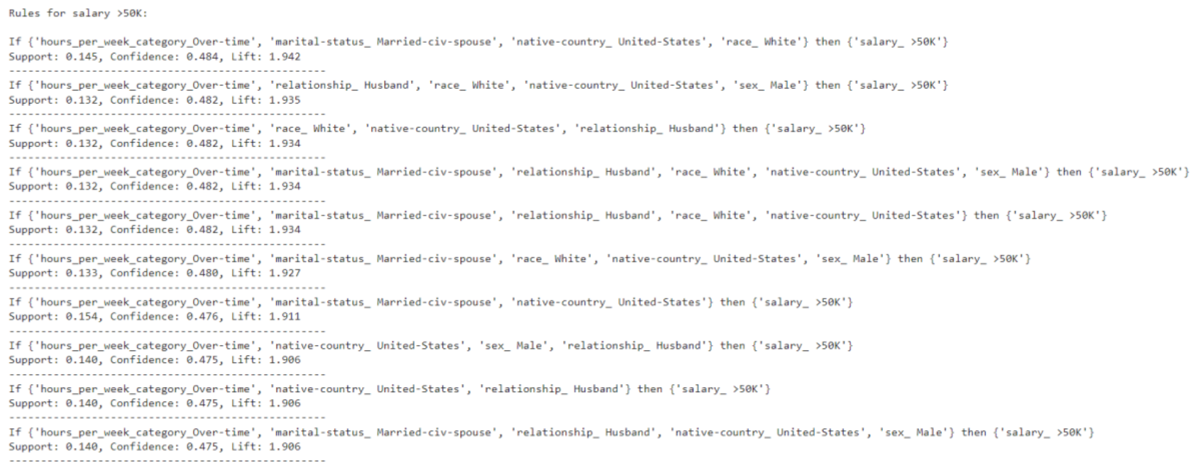
In our analytical pursuit to uncover patterns linked to incomes exceeding $50,000, we adopted the Apriori algorithm, concentrating on three core metrics: support, confidence, and lift. These metrics provided a comprehensive view of the relationships, evaluating their frequency, reliability, and relevance. To ensure a broad and thorough analysis, we leveraged Python and R, setting a minimum support threshold of 0.1 and a confidence level of 0.2. These values aimed to capture many impactful associations while preserving statistical validity.

The top 10 association rules using Python for a salary more significant than 50K are shown below. The interpretation for each rule is also given here.

1. Individuals who engage in overtime work, are married to civilian partners, reside within the United States, and self-identify as white have a 48.4% likelihood of earning an annual income above $50,000. The prevalence of this link is 1.942 times higher than what would be anticipated based on random chance.
2. Individuals who self-identify as male, belong to the white racial category, engage in overtime work, fulfil the position of a husband, and dwell within the United States have a 48.2% probability of earning a wage above $50,000. The strength of this association is 1.935 times greater than the average.
3. In the United States, a correlation exists between being of White ethnicity, working overtime, and assuming the position of a husband, with a 48.2% probability of earning an annual income above $50,000.
4. It has been observed that individuals who identify as white men, are married to civilian spouses, engage in overtime work, fulfil the responsibilities associated with being a husband, and reside within the United States, demonstrate a statistically significant likelihood of 48.2% to earn an annual income over $50,000.
5. The association between marital roles and demographic factors reveals that White individuals residing in the United States who are married to civilian spouses and working overtime as a husband have a 48.2% probability of having an annual income above $50,000.
6. In the United States, male individuals married to civilian wives who work overtime demonstrate a 48.0% probability of earning a wage that surpasses $50,000.
7. A positive association exists between being married to civilian spouses and working overtime in the United States, with a 47.6% probability of earning an annual income above $50,000.
8. There is a correlation between gender and relationship status, where males who work overtime, fulfil the function of a husband, and live in the United States, have a 47.5% probability of earning a salary above $50,000.
9. Individuals who work overtime, assume the position of a spouse, and reside within the United States demonstrate a 47.5% probability of earning a salary above $50,000.

10. The likelihood of males married to civilian spouses, working overtime, fulfilling the function of a husband, and residing in the United States, having an annual income over $50,000 is 47.5%.

In summary, the rules emphasise the importance of working additional hours beyond the standard work schedule and being legally wedded to a non-military spouse to surpass the threshold of earning $50,000 or more. Moreover, the regulations often incorporate demographic variables, such as race (specifically white) and nationality (specifically U.S. residency), suggesting their possible impact on the income category. Moreover, there appears to be a notable correlation between male persons, particularly those fulfilling the function of a husband, and membership in the higher income bracket. The constant lift values, consistently close to or exceeding 2, indicate that these relationships exhibit a notable level of strength beyond what would be anticipated by random chance.



```
Rules for salary >50K:

If {'hours_per_week_category_Over-time', 'marital-status_ Married-civ-spouse', 'native-country_ United-States', 'race_ White'} then {'salary_ >50K'}
Support: 0.145, Confidence: 0.484, Lift: 1.942
--------------------------------------------------
If {'hours_per_week_category_Over-time', 'relationship_ Husband', 'race_ White', 'native-country_ United-States', 'sex_ Male'} then {'salary_ >50K'}
Support: 0.132, Confidence: 0.482, Lift: 1.935
--------------------------------------------------
If {'hours_per_week_category_Over-time', 'race_ White', 'native-country_ United-States', 'relationship_ Husband'} then {'salary_ >50K'}
Support: 0.132, Confidence: 0.482, Lift: 1.934
--------------------------------------------------
If {'hours_per_week_category_Over-time', 'marital-status_ Married-civ-spouse', 'relationship_ Husband', 'race_ White', 'native-country_ United-States', 'sex_ Male'} then {'salary_ >50K'}
Support: 0.132, Confidence: 0.482, Lift: 1.934
--------------------------------------------------
If {'hours_per_week_category_Over-time', 'marital-status_ Married-civ-spouse', 'relationship_ Husband', 'race_ White', 'native-country_ United-States'} then {'salary_ >50K'}
Support: 0.132, Confidence: 0.482, Lift: 1.934
--------------------------------------------------
If {'hours_per_week_category_Over-time', 'marital-status_ Married-civ-spouse', 'race_ White', 'native-country_ United-States', 'sex_ Male'} then {'salary_ >50K'}
Support: 0.133, Confidence: 0.480, Lift: 1.927
--------------------------------------------------
If {'hours_per_week_category_Over-time', 'marital-status_ Married-civ-spouse', 'native-country_ United-States'} then {'salary_ >50K'}
Support: 0.154, Confidence: 0.476, Lift: 1.911
--------------------------------------------------
If {'hours_per_week_category_Over-time', 'native-country_ United-States', 'sex_ Male', 'relationship_ Husband'} then {'salary_ >50K'}
Support: 0.140, Confidence: 0.475, Lift: 1.906
--------------------------------------------------
If {'hours_per_week_category_Over-time', 'native-country_ United-States', 'relationship_ Husband'} then {'salary_ >50K'}
Support: 0.140, Confidence: 0.475, Lift: 1.906
--------------------------------------------------
If {'hours_per_week_category_Over-time', 'marital-status_ Married-civ-spouse', 'relationship_ Husband', 'native-country_ United-States', 'sex_ Male'} then {'salary_ >50K'}
Support: 0.140, Confidence: 0.475, Lift: 1.906
--------------------------------------------------
```

The top 10 rules using the R language are shown below with the interpretation.

1. Individuals who are classified as "Married-civ-spouse" who are not employed in jobs related to "Craft repair" exhibit a 48.36% probability (confidence level) of generating an income exceeding $50,000. The likelihood of this rule is 1.94 times greater than that of a random relationship, as measured by the lift.
2. Individuals married to civilian spouses and do not belong to jobs classified as "Other service" have a 47.12% probability of earning above $50K. Moreover, the strength of this link is 1.89 times greater than what would be expected by chance.
3. According to the data, married individuals not employed as "Machine op inspect" (Machine Operators/Inspectors) have a 47.09% probability of earning an annual income above $50,000.

4. The probability of married civilians not engaged in full-time employment having a salary exceeding $50,000 is 46.87%.
5. The Self-Employed Exclusion refers to individuals who are married to civilian spouses and do not engage in self-employment activities, as shown by the "Self.emp.not.inc" category. This group demonstrates a 46.75% probability of surpassing the income level of $50,000.
6. Individuals who are married to civilian spouses and are not engaged in occupations related to transportation and moving have a probability of 46.70% earning a salary exceeding $50,000.
7. Civilians who are married and not employed in occupations related to farming and fishing exhibit a 46.69% probability of generating an annual income above $50,000.
8. Native residents of the United States who are married have a 46.57% probability of earning an income above $50,000.
9. Workers who are married and work overtime, exceeding the standard working hours, have a probability of 46.55% of generating an income that exceeds $50,000.
10. Individuals who are married to civilian spouses who are not employed as "Handler’s cleaners" have a 46.46% probability of earning a salary over $50,000.

The variable "Married-civ-spouse" regularly exhibits a notable influence in these regulations, indicating a robust correlation with an income over $50,000. The influence of occupation categories, particularly the removal of specific roles, impacts earning potential. In addition, the number of hours worked, particularly those exceeding regular working hours, is a significant factor associated with increased wages. The observed lift values, which are all approximately equal to or larger than 2, highlight the robustness of these connections in contrast to a random setting.

```
##                                                                              lhs
## 98           {marital.status..Married.civ.spouse=1,occupation..Craft.repair=0}
## 102         {marital.status..Married.civ.spouse=1,occupation..Other.service=0}
## 108     {marital.status..Married.civ.spouse=1,occupation..Machine.op.inspct=0}
## 94  {marital.status..Married.civ.spouse=1,hours_per_week_category.Full.time=0}
## 105       {workclass..Self.emp.not.inc=0,marital.status..Married.civ.spouse=1}
## 109      {marital.status..Married.civ.spouse=1,occupation..Transport.moving=0}
## 116       {marital.status..Married.civ.spouse=1,occupation..Farming.fishing=0}
## 104     {marital.status..Married.civ.spouse=1,native.country..United.States=1}
## 88  {marital.status..Married.civ.spouse=1,hours_per_week_category.Over.time=1}
## 113     {marital.status..Married.civ.spouse=1,occupation..Handlers.cleaners=0}
##                  rhs   support confidence     lift
## 98  {salary...50K=1} 0.1851754  0.4836641 1.942067
## 102 {salary...50K=1} 0.2088656  0.4711826 1.891950
## 108 {salary...50K=1} 0.2045191  0.4708578 1.890645
## 94  {salary...50K=1} 0.1961910  0.4686534 1.881794
## 105 {salary...50K=1} 0.1924085  0.4674728 1.877053
## 109 {salary...50K=1} 0.2026278  0.4670031 1.875168
## 116 {salary...50K=1} 0.2088988  0.4669238 1.874849
## 104 {salary...50K=1} 0.1976509  0.4656817 1.869861
## 88  {salary...50K=1} 0.1654335  0.4655028 1.869143
## 113 {salary...50K=1} 0.2096951  0.4646376 1.865669
```

Comparison

1. Rule Composition:
    a. Python: The rules derived from Python predominantly emphasise demographic and personal attributes like marital status, working hours, race, and country of residence.
    b. R: The R-based rules are more diverse in their composition, focusing on marital status combined with various occupational exclusions.
2. Marital Status:
    a. Both toolkits identified the "Married-civ-spouse" status as a strong predictor for the >50K salary bracket. This status consistently appears in the top rules for both languages, emphasising its importance.
3. Working Hours:
    a. Python: The concept of overtime (working hours exceeding a standard threshold) is a recurrent theme in the Python rules, suggesting that those who work longer hours are more likely to earn above $50K.
    b. R: Only one of the top 10 R rules specifically mentions overtime, indicating that while it is a factor, it is not as dominant as in the Python rules.
4. Occupation:

   a. Python: Occupation is not a primary focus in the Python rules.
   b. R: Several R rules highlight the absence of specific occupations as predictors for earning >50K, such as not being in "Craft repair" or "Other service" jobs.
5. Demographics:
   a. Python: Demographics like race (mainly being White) and being a resident of the United States frequently appear in the Python rules.
   b. R: Only one of the top R rules mentions being a native resident of the United States.
6. Metrics:
   a. Support: R rules generally have slightly higher support than the Python ones, indicating that the combinations in the R rules appear more frequently in the dataset.
   b. Confidence: Both toolkits produce rules with similar confidence levels, generally hovering around the mid-40% range.
   c. Lift: The lift values for both Python and R rules are close to 2 or slightly below, indicating that the likelihood of the antecedent and consequent occurring together is roughly twice as high as expected if they were statistically independent.

Number	Python Rule	Support	Confidence	Lift	R Rule	Support2	Confidence2	Lift2
1	{Overtime, Married-civ-spouse, United-States, White}	0.145	0.484	1.942	{Married-civ-spouse, not Craft-repair}	0.185	0.484	1.942
2	{Overtime, Husband, White, United-States, Male}	0.132	0.482	1.935	{Married-civ-spouse, not Other-service}	0.209	0.471	1.892
3	{Overtime, White, United-States, Husband}	0.132	0.482	1.934	{Married-civ-spouse, not Machine-op-inspct}	0.205	0.471	1.891
4	{Overtime, Married-civ-spouse, Husband, White, United-States, Male}	0.132	0.482	1.934	{Married-civ-spouse, not Full-time}	0.196	0.469	1.882
5	{Overtime, Married-civ-spouse, Husband, White, United-States}	0.132	0.482	1.934	{Married-civ-spouse, not Self-employed}	0.192	0.467	1.877
6	{Overtime, Married-civ-spouse, White, United-States, Male}	0.133	0.48	1.927	{Married-civ-spouse, not Transport-moving}	0.203	0.467	1.875
7	{Overtime, Married-civ-spouse, United-States}	0.154	0.476	1.911	{Married-civ-spouse, not Farming-fishing}	0.209	0.467	1.875
8	{Overtime, United-States, Male, Husband}	0.14	0.475	1.906	{Married-civ-spouse, US-native}	0.198	0.466	1.87
9	{Overtime, United-States, Husband}	0.14	0.475	1.906	{Married-civ-spouse, Overtime}	0.165	0.466	1.869
10	{Overtime, Married-civ-spouse, Husband, United-States, Male}	0.14	0.475	1.906	{Married-civ-spouse, not Handlers-cleaners}	0.21	0.465	1.866
Average		**0.138**	**0.4793**	**1.9235**		**0.1972**	**0.4693**	**1.8839**

While there are similarities in the rules generated by Python and R, especially concerning marital status, there are also evident differences. Python's rules tend to be more demographic-focused, whereas R's rules incorporate more information about occupation. The choice between toolkits might depend on the specific objectives of the analysis and the desired insights. Both toolkits, however, have shown the capability to generate meaningful and insightful association rules from

Summary of the Analysis

Our primary aim was to leverage the Apriori algorithm to discern association rules associated with an annual income exceeding $50,000. To accomplish this, we employed both Python and R, tapping into the unique strengths and libraries each offers. While our dataset was a mix of numerical and categorical data, the Apriori algorithm is tailored for categorical data. Thus, a significant part of our effort was devoted to transforming numerical columns into categorical bins, which proved pivotal in unveiling meaningful patterns.

Python and R were harnessed to apply the Apriori algorithm, using a support threshold of 0.1 and a confidence level of 0.2. These thresholds were chosen to strike a balance between generating rules that are statistically significant and insightful. The results illuminated intriguing insights about the demographic and occupational profiles of individuals earning more than $50,000 annually. Notably, factors like working overtime, being married to a civilian partner, and identifying as a white male in the United States consistently emerged as significant in our top rules. The lift values, often exceeding 1.8, further bolstered the strength of these associations.

The repeated prevalence of some traits in Python and R outputs was notably striking. The persistent presence of factors such as extended work hours and marital status highlights their importance in shaping compensation classifications. An unexpected finding pertained to the robust correlation between demographic characteristics, particularly race and country of residency, and the categorisation of pay levels. Although it could be anticipated that work-related characteristics would be the primary focus, the regularity with which these demographic attributes arise suggests the presence of underlying socio-economic trends.

The study also highlighted the complementarity of Python and R. Although both tools were used in similar capacities, they each provided slightly different nuances, showcasing the adaptability and collaborative potential of these two powerful analytical platforms. Overall, this endeavour not only furnished critical insights into the determinants of income but also emphasised the importance of meticulous data preprocessing and the rich toolkit available to data scientists—the dataset.

References

García, S., Luengo, J., & Herrera, F. (2015). *Data preprocessing in data mining* (Vol. 72). Springer.

Hegland, M. (2007). The apriori algorithm–a tutorial. In *Mathematics and Computation in imaging science and information processing* (pp. 209-262).

Li, J., Le, T. D., Liu, L., Liu, J., Jin, Z., & Sun, B. (2013). Mining causal association rules. In *2013 IEEE 13th International Conference on Data Mining Workshops*. IEEE.

Room, C. *Market Basket Analysis*.

Sasaki, Y. (2022). Path association rule mining. *arXiv preprint arXiv:2210.13136*.

Telikani, A., Gandomi, A. H., & Shahbahrami, A. (2020). A survey of evolutionary computation for association rule mining. *Information Sciences, 524*, 318-352.

Appendix

Python code

```
import numpy as np # linear algebra
import pandas as pd # data processing, CSV file I/O (e.g. pd.read_csv)
import os
import matplotlib.pyplot as plt
import seaborn as sns
import warnings
warnings.filterwarnings('ignore')
```

```
#Load the file into pandas
df = pd.read_csv("salary.csv")
#Check the first 5 rows of the dataframe
df.head()
```

```
#Check the structure of the dataframe
df.info()
```

```
#Check the occurrence of null values
df.isna().sum()
```

```
# Check for duplicate rows in the dataset
duplicate_rows = df[df.duplicated()]

# Display all the duplicate rows
all_duplicate_rows = duplicate_rows.copy()

# Remove the duplicate rows from the original dataset
df = df.drop_duplicates()

len(all_duplicate_rows), len(df)
```

```
df['workclass'] = df['workclass'].replace(' ?', np.nan)
df['occupation'] = df['occupation'].replace(" ?", np.nan)
df['native-country'] = df['native-country'].replace(" ?", np.nan)
```

```
#Check the occurrence of null values
df.isna().sum()
```

```
# replace the work class msinng vlaues with other
df['workclass'] = df['workclass'].fillna('other')
```

```
# remove all other misisng values
df=df.dropna(how='any')
```

```
#Check the occurrence of null values
df.isna().sum()
```

```
df['salary'].value_counts()
```

```
df.info()
```

```
df_oht_binned.to_csv('abb.csv')
```

Data conversion

```
# Bin 'fnlwgt' into quartiles
df['fnlwgt_category'] = pd.qcut(df['fnlwgt'], 4, labels=["Q1", "Q2", "Q3", "Q4"])
```

```python
# Bin 'fnlwgt' into quartiles
df['fnlwgt_category'] = pd.qcut(df['fnlwgt'], 4, labels=["Q1", "Q2", "Q3", "Q4"])

# Bin 'capital-gain' and 'capital-loss'
capital_bins = [-1, 1, 5000, 100000]
capital_labels = ["No Gain/Loss", "Low Gain/Loss", "High Gain/Loss"]
df['capital_gain_category'] = pd.cut(df['capital-gain'], bins=capital_bins, labels=capital_labels)
df['capital_loss_category'] = pd.cut(df['capital-loss'], bins=capital_bins, labels=capital_labels)

# Bin 'hours-per-week'
hours_bins = [0, 20, 40, 60, 100]
hours_labels = ["Part-time", "Full-time", "Over-time", "Excessive-time"]
df['hours_per_week_category'] = pd.cut(df['hours-per-week'], bins=hours_bins, labels=hours_labels, right=False)

# Drop the original columns but retain the 'salary' column
df_binned = df.drop(['fnlwgt', 'education-num', 'capital-gain', 'capital-loss', 'hours-per-week', 'age','education'], axis=1)

# Convert the dataset into a transactional format
transactional_data_binned = df_binned.values.tolist()
df_oht_binned = pd.get_dummies(pd.DataFrame(transactional_data_binned))

# Display the first few rows of the transformed dataset
print(df_oht_binned.head())
```

```python
# Check if 'salary' is in df_binned
print(df_binned['salary'].head())
```

```python
df_oht_binned = pd.get_dummies(df_binned)
```

```python
df_oht_binned.info()
```

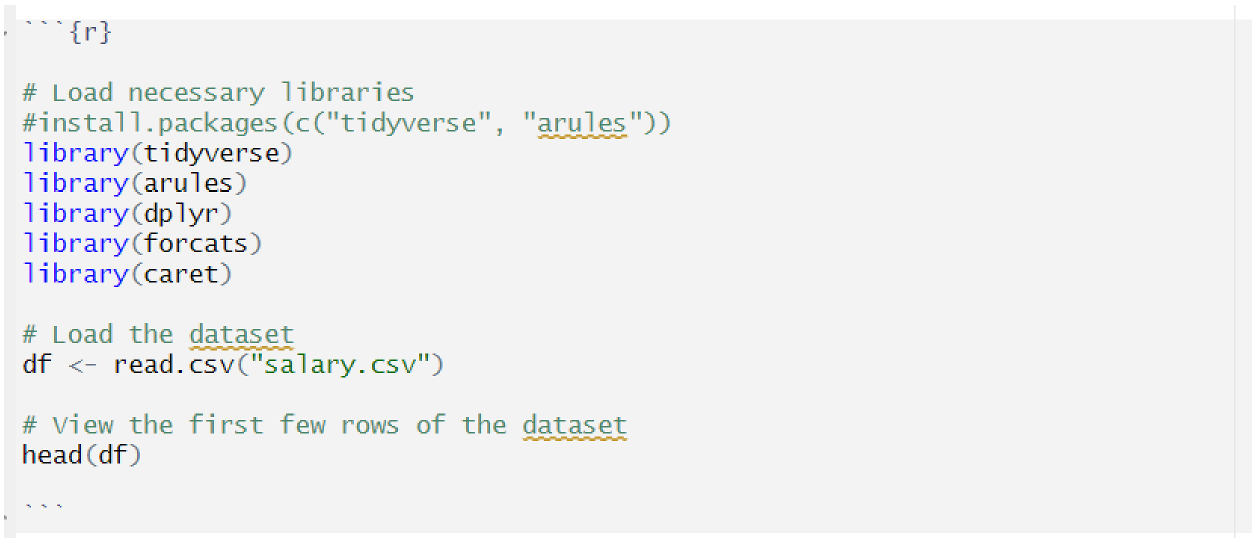
```python
from mlxtend.frequent_patterns import apriori, association_rules

# Find frequent itemsets using the Apriori algorithm
min_support = 0.1
frequent_itemsets = apriori(df_oht_binned, min_support=min_support, use_colnames=True)
frequent_itemsets
```

```python
# Extract rules with a minimum confidence (for demonstration, I'm using 0.3, but you can adjust this as needed)
rules = association_rules(frequent_itemsets, metric="confidence", min_threshold=0.2)
```

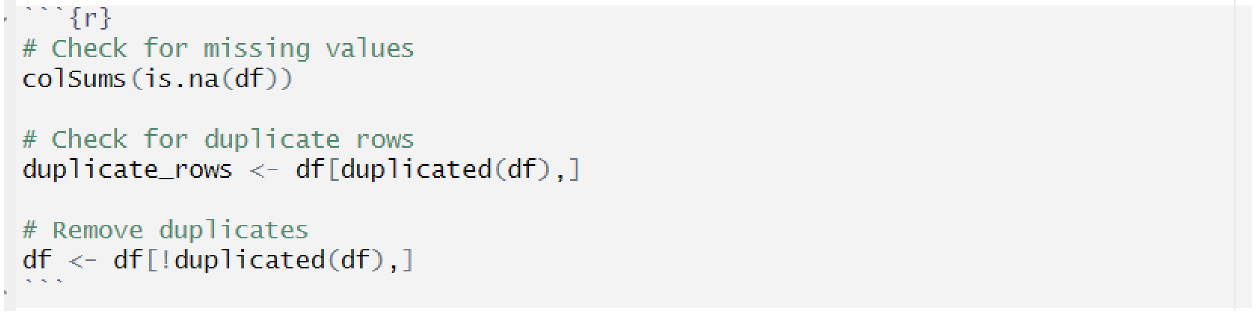
```python
rules_below_or_equal_50K = rules[rules['consequents'] == frozenset({'salary_ >50K'})].sort_values(by="confidence",
                                                                                                  ascending=False)
print("\nRules for salary >50K:\n")
for index, rule in rules_below_or_equal_50K.head(10).iterrows():
    print(f"If {set(rule['antecedents'])} then {set(rule['consequents'])}")
    print(f"Support: {rule['support']:.3f}, Confidence: {rule['confidence']:.3f}, Lift: {rule['lift']:.3f}")
    print("-" * 50)
```

```python
rules_below_or_equal_50K = rules[rules['consequents'] == frozenset({'salary_ <=50K'})].sort_values(by="confidence",
                                                                                                   ascending=False)
print("\nRules for salary <=50K:\n")
for index, rule in rules_below_or_equal_50K.head(10).iterrows():
    print(f"If {set(rule['antecedents'])} then {set(rule['consequents'])}")
    print(f"Support: {rule['support']:.3f}, Confidence: {rule['confidence']:.3f}, Lift: {rule['lift']:.3f}")
    print("-" * 50)
```

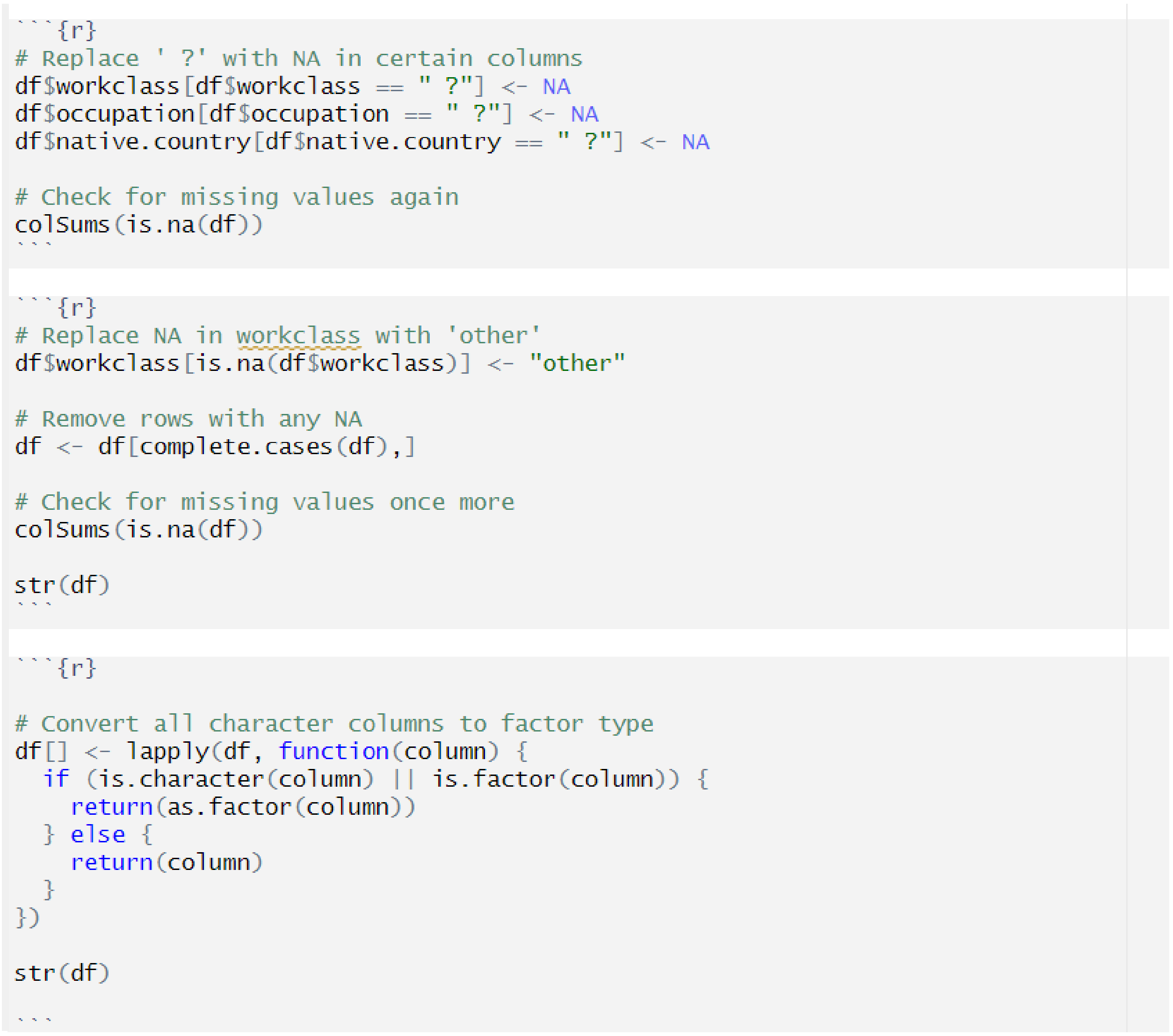
R code



~~~
```{r}

# Load necessary libraries
#install.packages(c("tidyverse", "arules"))
library(tidyverse)
library(arules)
library(dplyr)
library(forcats)
library(caret)

# Load the dataset
df <- read.csv("salary.csv")

# View the first few rows of the dataset
head(df)

```
~~~



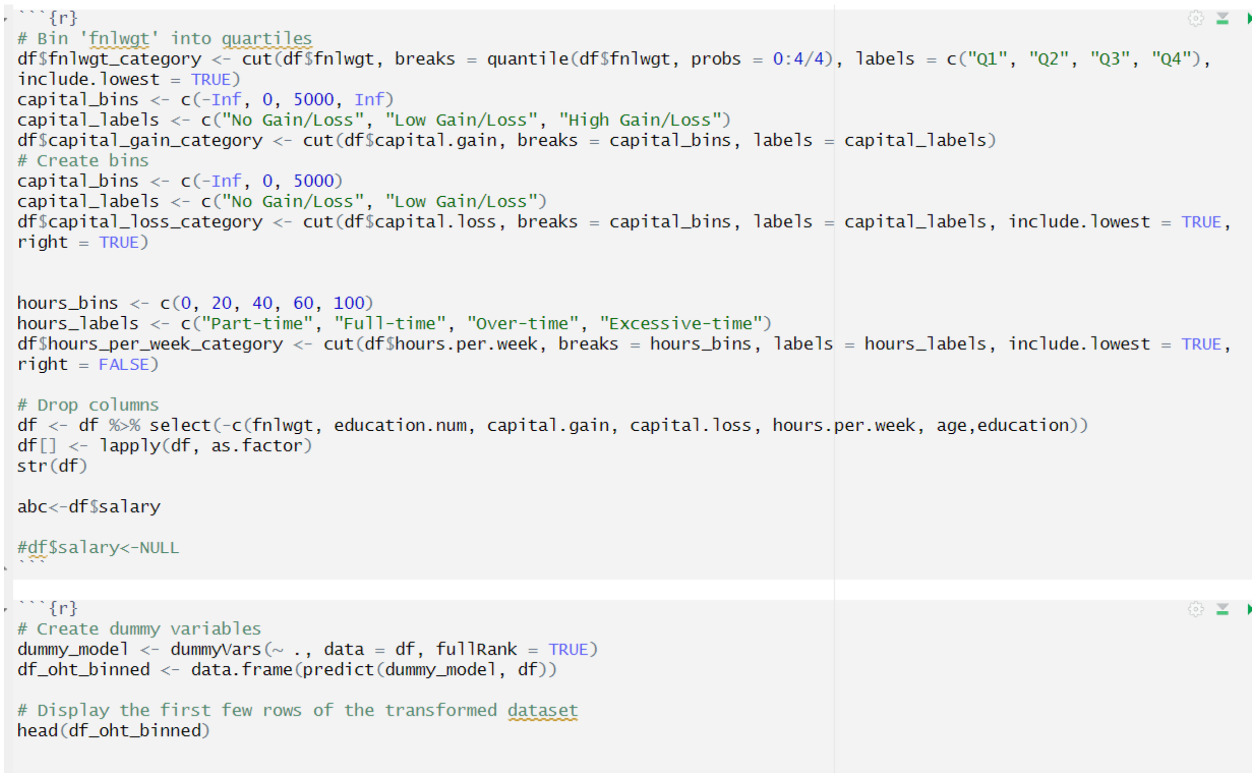
~~~
```{r}
# Check for missing values
colSums(is.na(df))

# Check for duplicate rows
duplicate_rows <- df[duplicated(df),]

# Remove duplicates
df <- df[!duplicated(df),]
```
~~~



~~~
```{r}
# Replace ' ?' with NA in certain columns
df$workclass[df$workclass == " ?"] <- NA
df$occupation[df$occupation == " ?"] <- NA
df$native.country[df$native.country == " ?"] <- NA

# Check for missing values again
colSums(is.na(df))
```
~~~

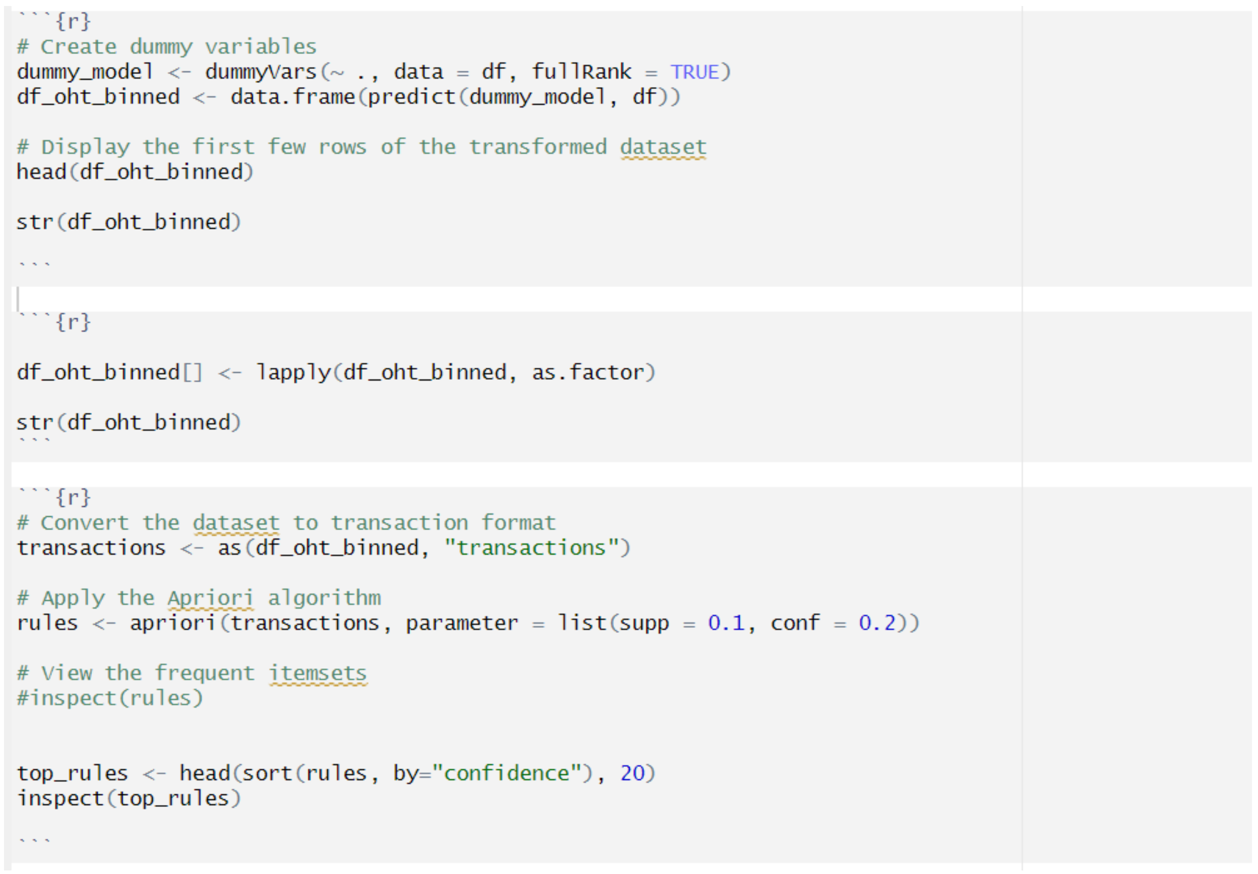
~~~
```{r}
# Replace NA in workclass with 'other'
df$workclass[is.na(df$workclass)] <- "other"

# Remove rows with any NA
df <- df[complete.cases(df),]

# Check for missing values once more
colSums(is.na(df))

str(df)
```
~~~

~~~
```{r}

# Convert all character columns to factor type
df[] <- lapply(df, function(column) {
  if (is.character(column) || is.factor(column)) {
    return(as.factor(column))
  } else {
    return(column)
  }
})

str(df)

```
~~~



~~~r
```{r}
# Bin 'fnlwgt' into quartiles
df$fnlwgt_category <- cut(df$fnlwgt, breaks = quantile(df$fnlwgt, probs = 0:4/4), labels = c("Q1", "Q2", "Q3", "Q4"),
include.lowest = TRUE)
capital_bins <- c(-Inf, 0, 5000, Inf)
capital_labels <- c("No Gain/Loss", "Low Gain/Loss", "High Gain/Loss")
df$capital_gain_category <- cut(df$capital.gain, breaks = capital_bins, labels = capital_labels)
# Create bins
capital_bins <- c(-Inf, 0, 5000)
capital_labels <- c("No Gain/Loss", "Low Gain/Loss")
df$capital_loss_category <- cut(df$capital.loss, breaks = capital_bins, labels = capital_labels, include.lowest = TRUE,
right = TRUE)


hours_bins <- c(0, 20, 40, 60, 100)
hours_labels <- c("Part-time", "Full-time", "Over-time", "Excessive-time")
df$hours_per_week_category <- cut(df$hours.per.week, breaks = hours_bins, labels = hours_labels, include.lowest = TRUE,
right = FALSE)

# Drop columns
df <- df %>% select(-c(fnlwgt, education.num, capital.gain, capital.loss, hours.per.week, age,education))
df[] <- lapply(df, as.factor)
str(df)

abc<-df$salary

#df$salary<-NULL
~~~

```{r}
# Create dummy variables
dummy_model <- dummyVars(~ ., data = df, fullRank = TRUE)
df_oht_binned <- data.frame(predict(dummy_model, df))

# Display the first few rows of the transformed dataset
head(df_oht_binned)
```



~~~r
```{r}
# Create dummy variables
dummy_model <- dummyVars(~ ., data = df, fullRank = TRUE)
df_oht_binned <- data.frame(predict(dummy_model, df))

# Display the first few rows of the transformed dataset
head(df_oht_binned)

str(df_oht_binned)

~~~

```{r}

df_oht_binned[] <- lapply(df_oht_binned, as.factor)

str(df_oht_binned)
```

```{r}
# Convert the dataset to transaction format
transactions <- as(df_oht_binned, "transactions")

# Apply the Apriori algorithm
rules <- apriori(transactions, parameter = list(supp = 0.1, conf = 0.2))

# View the frequent itemsets
#inspect(rules)


top_rules <- head(sort(rules, by="confidence"), 20)
inspect(top_rules)

```
```
```

~~~
```{r}

# Where salary is greater than 50K

# 3. Filter Rules
# Filter rules for consequences where the salary is greater than 50K.
salary_rules <- subset(rules, rhs %in% "salary...50K=1")

top_rules10 <- head(sort(salary_rules, by="confidence"), 10)
inspect(top_rules10)


```
~~~

~~~
```{r}

# Extracting the rules into a data frame
rules_df <- data.frame(
  lhs = labels(lhs(salary_rules)),
  rhs = labels(rhs(salary_rules)),
  support = quality(salary_rules)$support,
  confidence = quality(salary_rules)$confidence,
  lift = quality(salary_rules)$lift
)

# Sorting the rules by confidence
top_rules_df <- head(rules_df[order(-rules_df$confidence), ], 10)

# Displaying the top 10 rules
print(top_rules_df)


```
~~~

~~~
```{r}

# Where salary is less or equal to 50K

# 3. Filter Rules
# Filter rules for consequences where the salary is greater than 50K.
salary_rules <- subset(rules, rhs %in% "salary...50K=0")

top_rules10 <- head(sort(salary_rules, by="confidence"), 10)
inspect(top_rules10)

```
~~~

~~~
```{r}

# Extracting the rules into a data frame
rules_df <- data.frame(
  lhs = labels(lhs(salary_rules)),
  rhs = labels(rhs(salary_rules)),
  support = quality(salary_rules)$support,
  confidence = quality(salary_rules)$confidence,
  lift = quality(salary_rules)$lift
)

# Sorting the rules by confidence
top_rules_df <- head(rules_df[order(-rules_df$confidence), ], 10)

# Displaying the top 10 rules
print(top_rules_df)


```
~~~