TURTLE GAMES:
WHAT DO OUR CUSTOMERS NEED?

SEPTEMBER 2024

PREPARED BY

Henry Pescod

PREPARED FOR

Turtle Games - LSE

Predicting future outcomes for Turtle Games

Course 3 Assignment submission Henry Pescod - 30th September 2024

Table of Contents

1. The business scenario

Turtle Games is a global games manufacturer and retailer seeking to improve their overall sales performance by improving their understanding of their customers. To achieve this Turtle Games believes that a better understanding of their loyalty program, customer behaviour and product reviews will enhance their loyalty program and overall customer engagement.

To approach this analysis I used a fishbone diagram to disaggregate the problem of "what do Turtle Games customers need"? into component parts.

I decided on three questions to tackle the analysis:

1. What drives loyalty points of Turtle Games customers?
2. How does this help to understand Turtle Games customers?
3. How does this help Turtle Games?

The following table maps these questions to the problem component, and the analysis approach this report takes to answer each question.

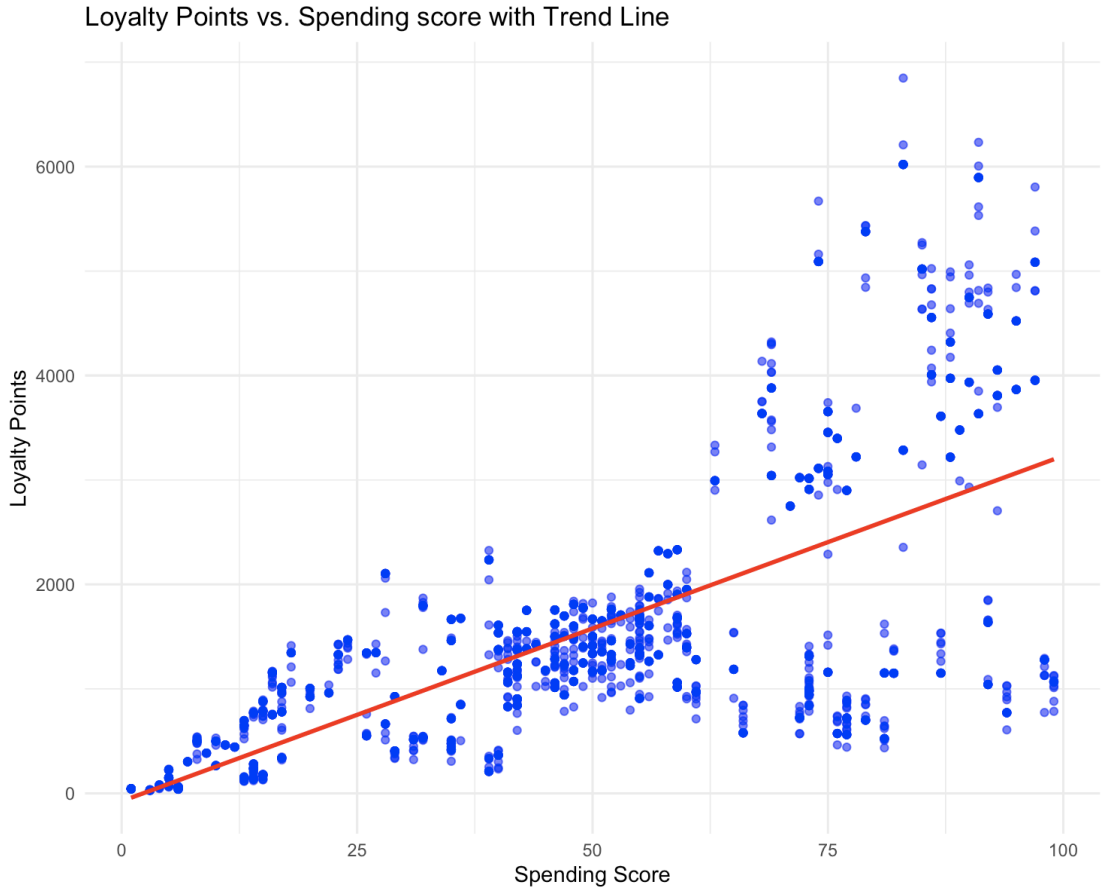
Question	Problem Component	Analysis approach	Why is this a problem?
1. What drives Loyalty points of Turtle Games customers?	Loyalty Points	Linear Regression Correlation Plot Decision tree Regression Multiple linear Regression	Customers may not be accumulating points in a way that drives engagement and repeat sales.
2. How does this help to understand Turtle Games customers?	Customer Segmentation	K means clustering EDA by grouping of clusters	Lack of clear understanding of how different customer segments behave.
	Customer Feedback	NLP and sentiment analysis by clusters	Customer reviews may not be adequately informing team practices.
	Sales per Product	See limitations of this report.	Inefficient strategies may be leading to lower sales for certain products.
3. How does this help Turtle Games?	Resource Allocation & Team Improvement	See limitations of this report.	Uncertainty about where to allocate resources for maximum impact on loyalty and sales.

2. What drives Loyalty points of Turtle Games customers?

To establish the key drivers of loyalty point accumulation my analysis was broken down into 4 parts

a) Linear Regression & Correlation Plot

Once I was satisfied the data set was clean, I conducted linear regression analysis to evaluate possible linear relationships between loyalty points, age, remuneration and spending scores. I also created a correlation plot in R using the corplot function.



29]:

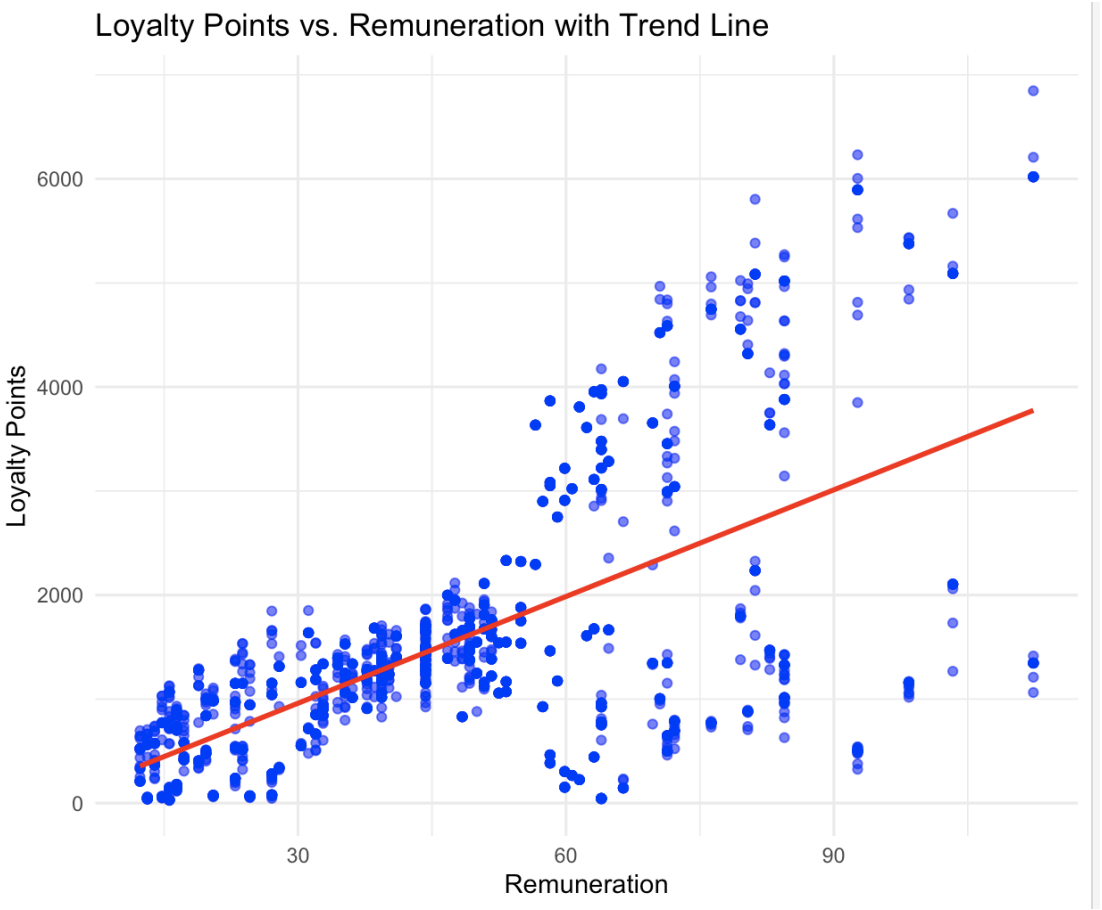
OLS Regression Results

Dep. Variable:	y	**R-squared:**	0.452
Model:	OLS	**Adj. R-squared:**	0.452
Method:	Least Squares	**F-statistic:**	1648.
Date:	Sun, 29 Sep 2024	**Prob (F-statistic):**	2.92e-263
Time:	18:18:47	**Log-Likelihood:**	-16550.
No. Observations:	2000	**AIC:**	3.310e+04
Df Residuals:	1998	**BIC:**	3.312e+04
Df Model:	1		
Covariance Type:	nonrobust		

	coef	std err	t	P>|t|	[0.025	0.975]
Intercept	-75.0527	45.931	-1.634	0.102	-165.129	15.024
x	33.0617	0.814	40.595	0.000	31.464	34.659

Omnibus:	126.554	**Durbin-Watson:**	1.191
Prob(Omnibus):	0.000	**Jarque-Bera (JB):**	260.528
Skew:	0.422	**Prob(JB):**	2.67e-57
Kurtosis:	4.554	**Cond. No.**	122.

Pescod_Henry_DA301_Assignment_Report.pdf



Pescod_Henry_DA301_Assignment_Report.pdf

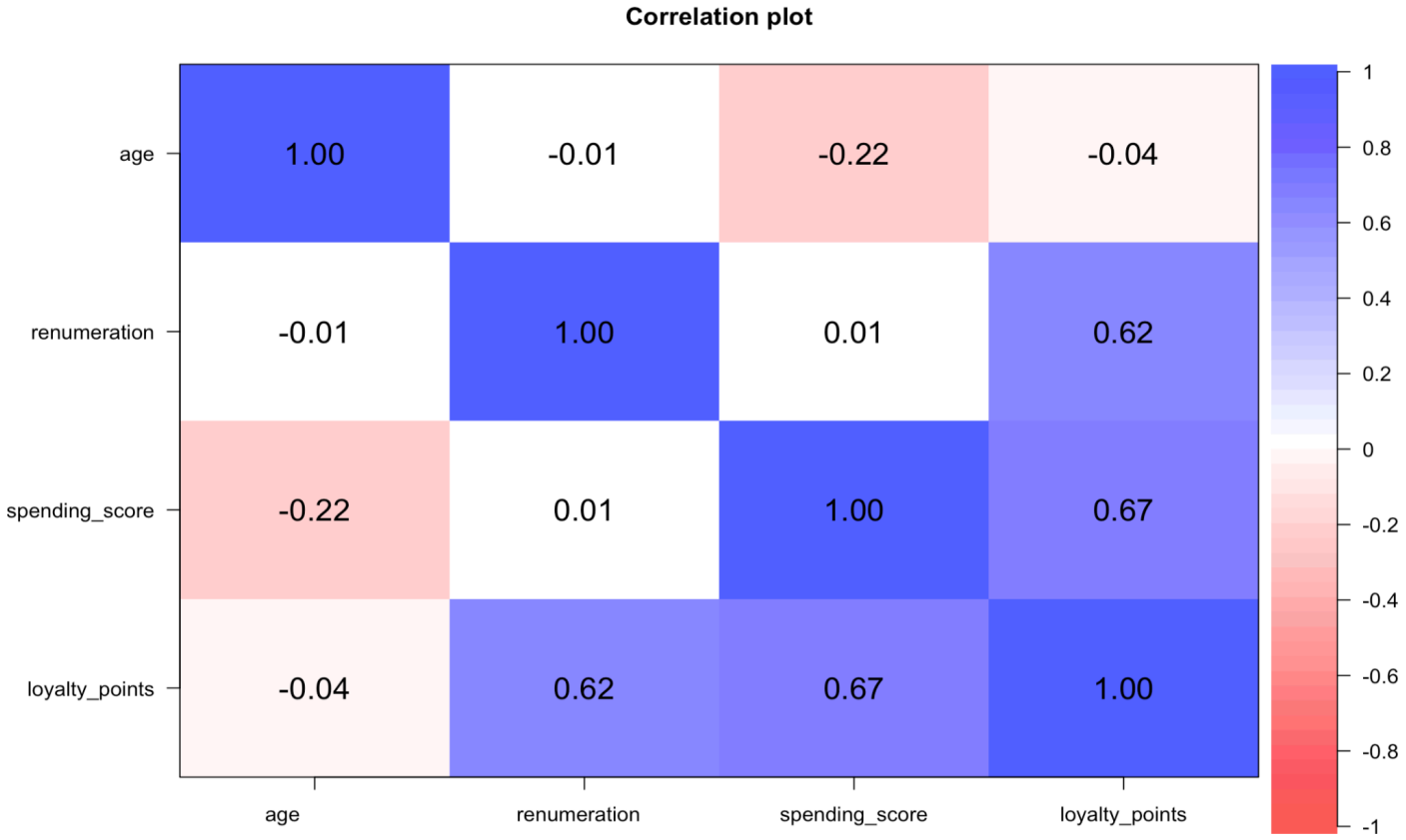
:

OLS Regression Results

Dep. Variable:	y	**R-squared:**	0.380
Model:	OLS	**Adj. R-squared:**	0.379
Method:	Least Squares	**F-statistic:**	1222.
Date:	Sun, 29 Sep 2024	**Prob (F-statistic):**	2.43e-209
Time:	13:18:19	**Log-Likelihood:**	-16674.
No. Observations:	2000	**AIC:**	3.335e+04
Df Residuals:	1998	**BIC:**	3.336e+04
Df Model:	1		
Covariance Type:	nonrobust		

	coef	std err	t	P>|t|	[0.025	0.975]
Intercept	-65.6865	52.171	-1.259	0.208	-168.001	36.628
x	34.1878	0.978	34.960	0.000	32.270	36.106

Omnibus:	21.285	**Durbin-Watson:**	3.622
Prob(Omnibus):	0.000	**Jarque-Bera (JB):**	31.715
Skew:	0.089	**Prob(JB):**	1.30e-07
Kurtosis:	3.590	**Cond. No.**	123.



The linear regression analysis highlighted a positive correlation between loyalty points and spending score (.452 r2) and remuneration (0.380 r2). Both showed a heavy tailed positive skew with significant kurtosis figures.

The correlation plot highlighted a strong correlation of spending_score (0.67) and loyalty points (0.62). Age has a broadly neutral effect.

b) Decision tree Regression

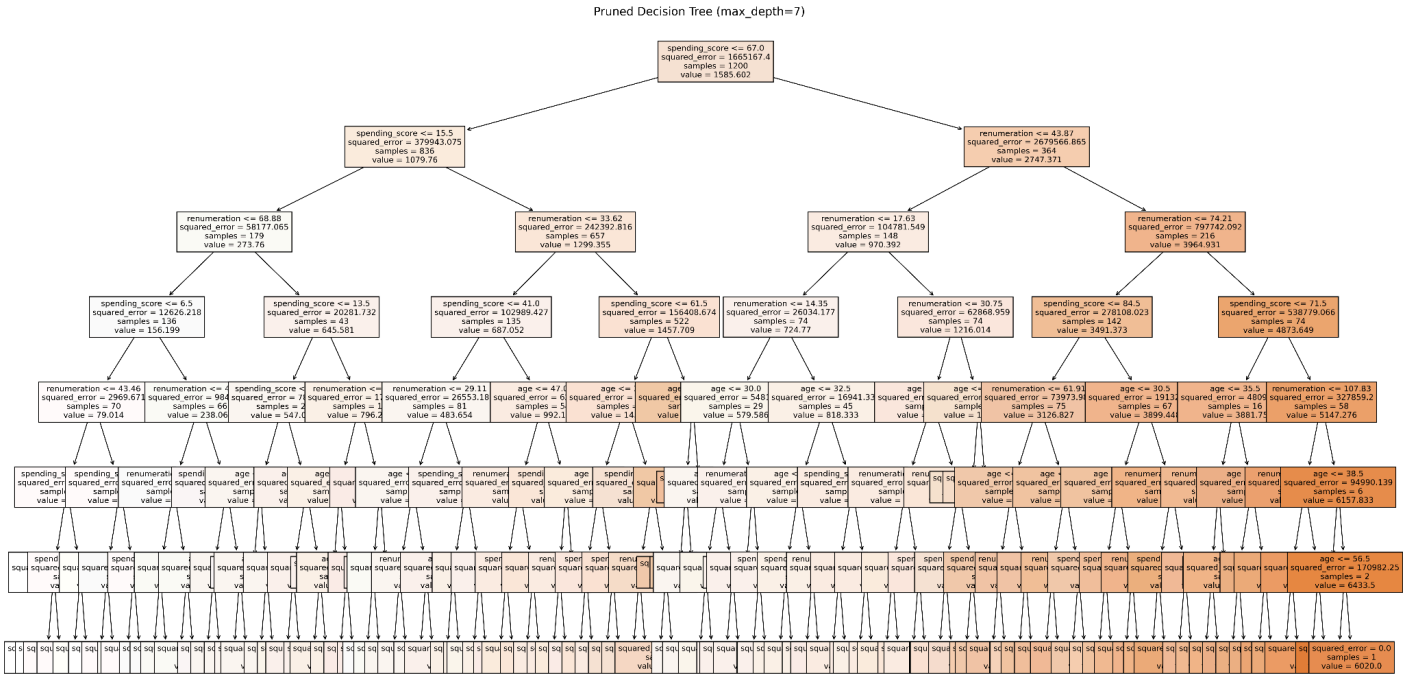
I used decision tree regression analysis to understand how the different variables of the data contribute to the accumulation of loyalty points.

I defined the independent variables as age, spending score and remuneration. I trained, tested, and validated several models using remuneration, spending score, and age as the independent variables, with loyalty points as the dependent variable.

Model	Mean Squared Error	Root Mean Squared Error	Mean Absolute Error	R-squared

Unpruned Decision Tree	5,442.09	73.77	23.96	0.9969
Pruned Decision Tree (Depth 3)	146,819.57	383.17	279.90	0.9170
Pruned Decision Tree (Depth 5)	75,027.12	273.91	182.27	0.9576
Pruned Decision Tree (Depth 7)	33,921.68	184.18	117.08	0.9808
Pruned Decision Tree (Depth 9)	20,983.43	144.86	86.64	0.9881

I decided to recommend a tree depth of 7 for balance of reduced complexity and reduced risk of overfitting.



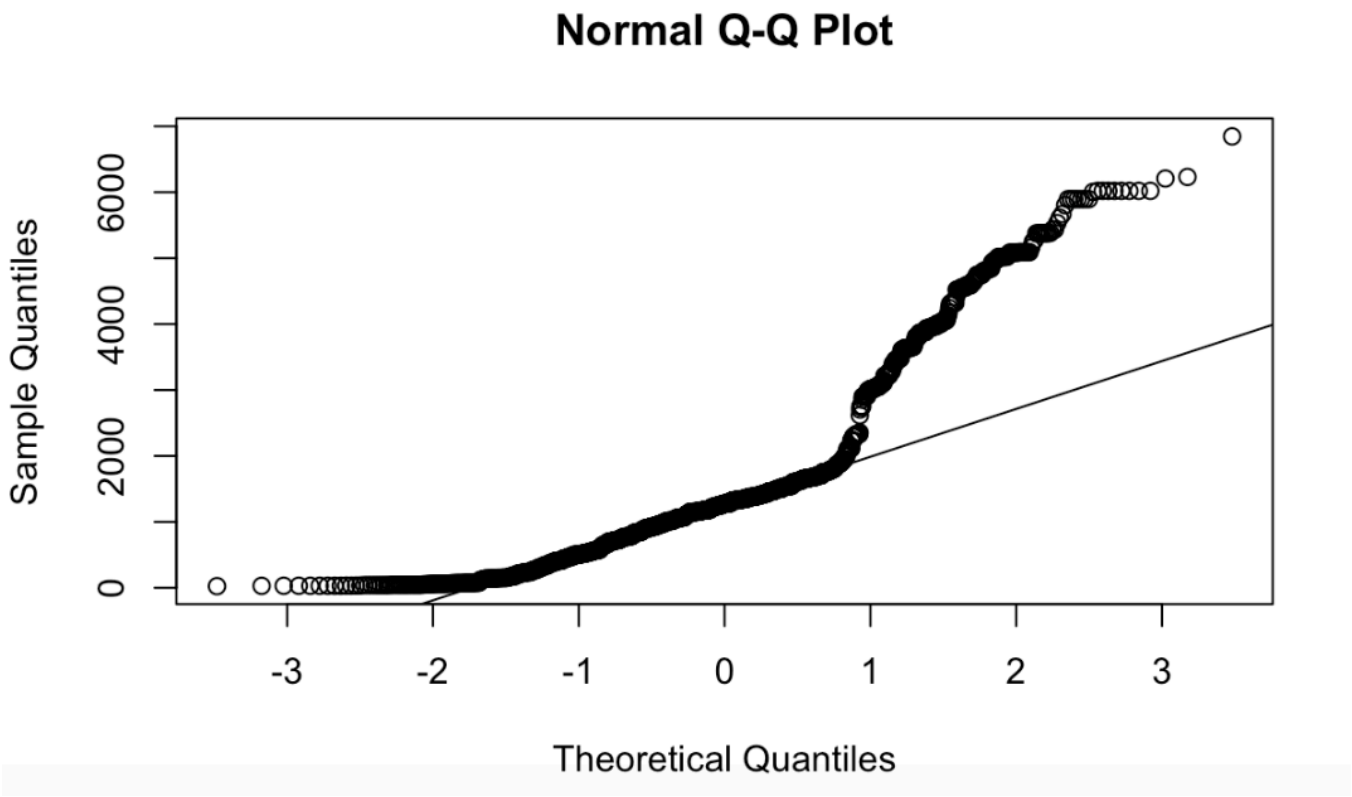
Spending score and remuneration are the most critical factors affecting accumulation of loyalty points

The unpruned model showed a very high R^2 value of 99.69%, indicating that 99.69% of the variability in loyalty points can be explained by the independent variables. Such a high R^2 may suggest overfitting in the unpruned model. To reduce complexity and minimise the risk of overfitting, I settled on a tree depth of 7.

c) Multiple linear Regression

My final step in answering this question was to use multiple linear regression to build a predictive model for loyalty points.

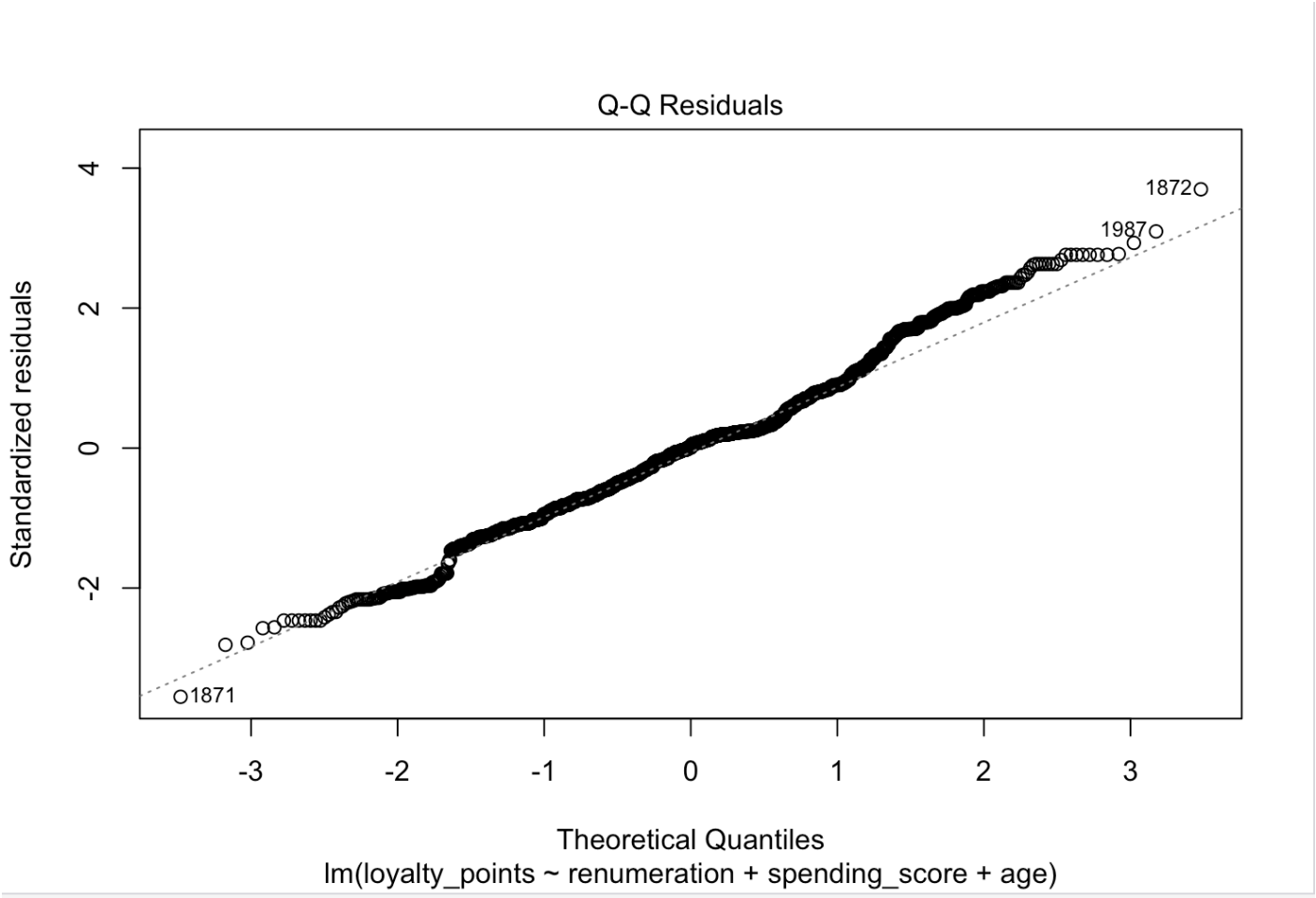
First I analysed the distribution of the data and used a Q-Q plot to visualise the distribution of the data against a best fit line. The plot and summary statistics highlighted a non normal distribution.



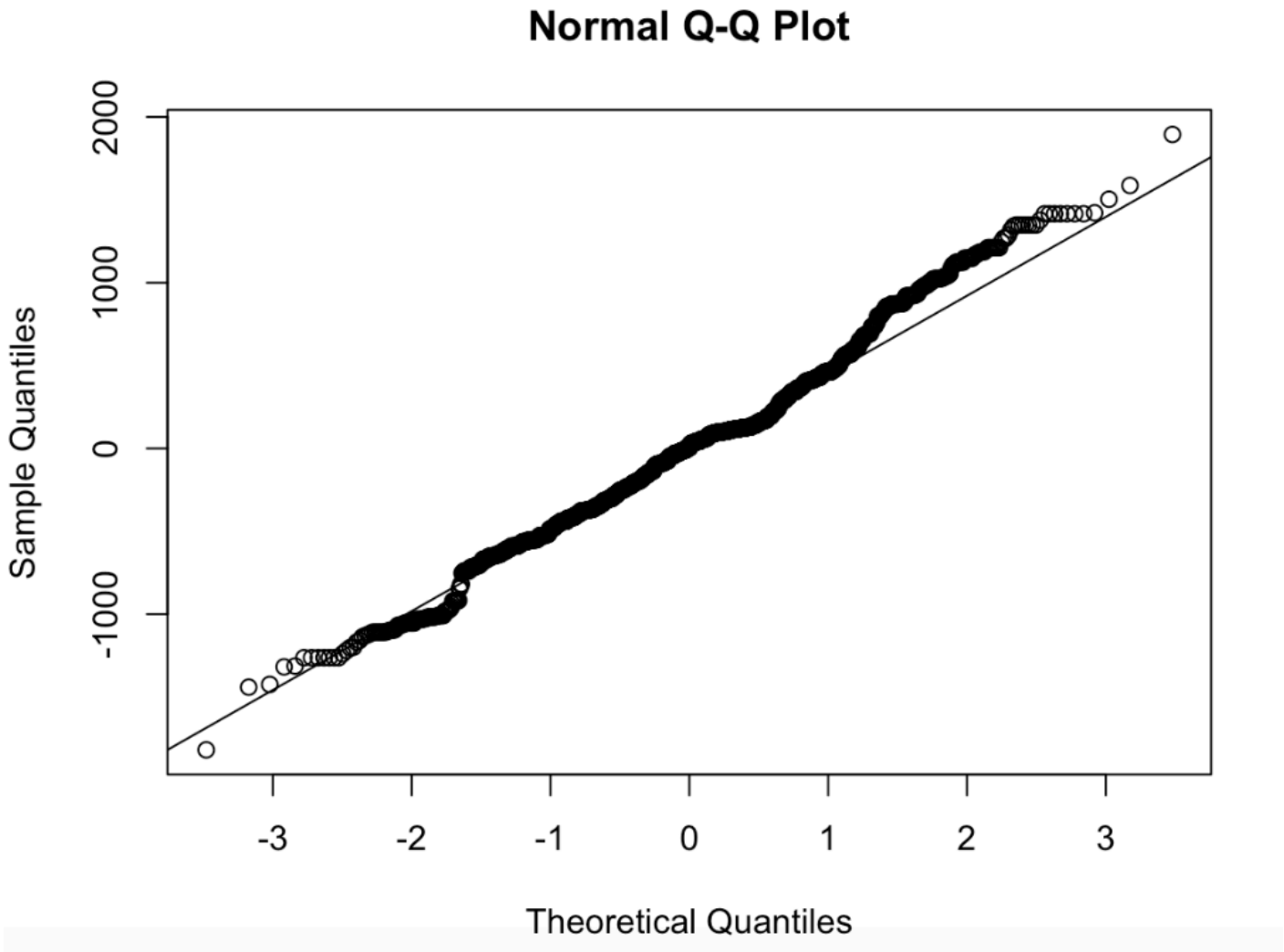
Through testing a series of models, I was able to confirm that remuneration, spending score, and age are still strong predictors of loyalty points despite the non-normal distribution of the data.

```
> summary(modelc)

Call:
lm(formula = loyalty_points ~ renumeration + spending_score +
    age, data = reviews_new)

Residuals:
     Min       1Q   Median       3Q      Max
-1819.11  -350.84     4.61   291.00  1894.62

Coefficients:
                 Estimate Std. Error t value Pr(>|t|)
(Intercept)    -2203.0598    52.3609  -42.08   <2e-16 ***
renumeration      34.0084     0.4970   68.43   <2e-16 ***
spending_score    34.1832     0.4519   75.64   <2e-16 ***
age               11.0607     0.8688   12.73   <2e-16 ***
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

Residual standard error: 513.8 on 1996 degrees of freedom
Multiple R-squared:  0.8399,    Adjusted R-squared:  0.8397
F-statistic:  3491 on 3 and 1996 DF,  p-value: < 2.2e-16
```

Below is a residual plot for the multiple linear regression model (Model C), assessing model fit and error distribution.



Below is a Q-Q plot of residuals from the multiple linear regression model (Model C), used to check for normality in the residuals.



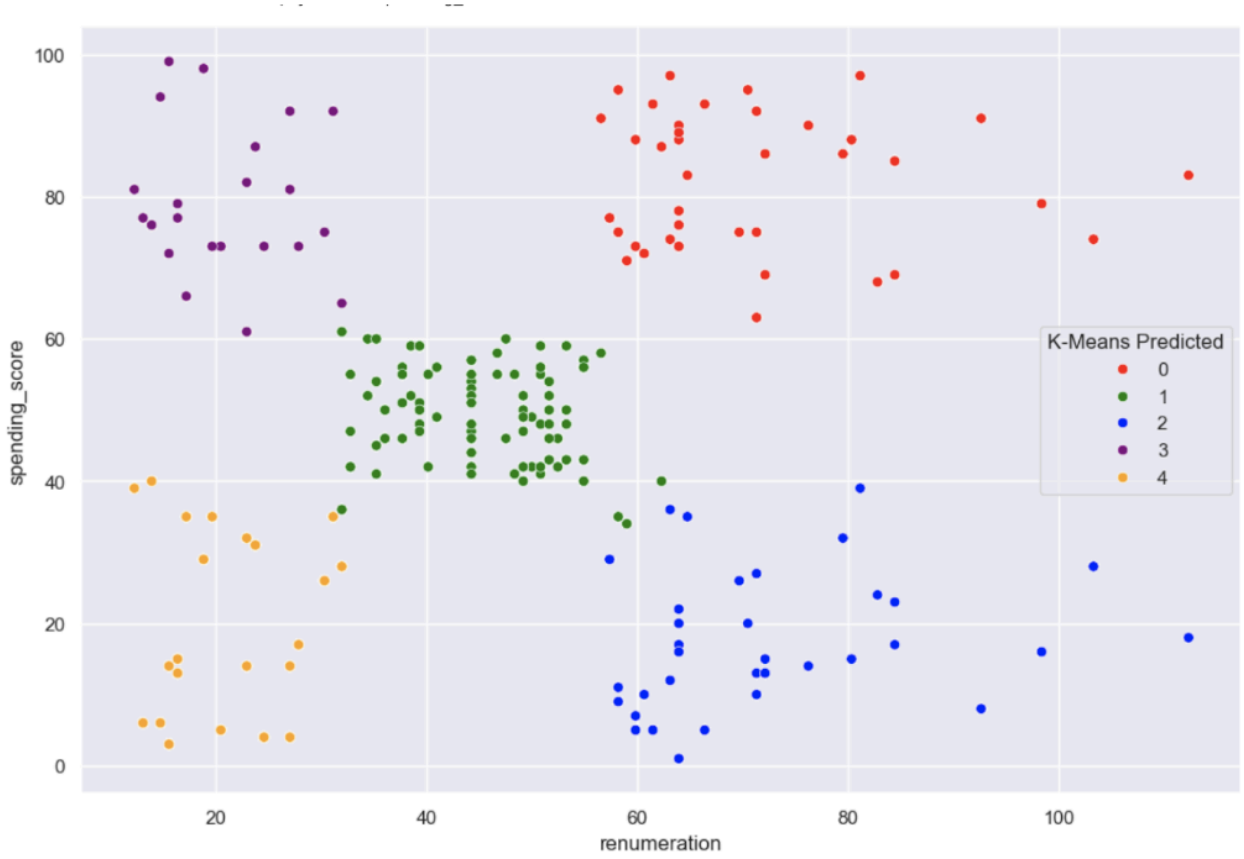
Model C (3 variables - age, spending score and remuneration) achieved an adjusted R^2 value of 83.97%, indicating that the variables age, remuneration, and loyalty explain 83.97% of the variability in the number of loyalty points a new customer will accumulate. This suggests the model has a strong fit for the data.

3. How does this help to understand Turtle Games customers?

a) K means clustering

I used the K means clustering method from the sk.learns package to segment customers based on remuneration and spending score.

There are 5 distinct segments of customer. I established names from these 5 clusters based on each cluster's attributes that could help with customer segmentation.



I identified the optimal cluster number of 5.

b) Exploratory data analysis by grouping of clusters in R

I assigned clusters and cluster names to the data set for further analysis. I produced a matrix of each cluster's attributes. I used Facet grids to combine bar plots visualising loyalty points, against mean remuneration, mean spending score and mean loyalty points.

Cluster	cluster name	avg age	Avg spending score	Avg Remuneration	Number of customers	total loyalty points	Avg loyalty points
0	High Spenders	36	82.01	73.24	356	1419813	3988
1	Moderate Earners	42	49.53	44.42	774	1099376	1420
2	Low Spenders, High Income	41	17.42	74.83	330	300881	912
3	Value Seekers	32	79.42	20.35	269	261453	972

4	Budget-Conscious	44	19.76	20.42	271	74541	275

What does this tell us about customer behaviour?

Customer cluster	Insight
"High Spenders	Highly valuable customers
"Moderate earners"	largest portion of the sample data set. Consider certain products that appeal specifically to their profile
"Low Spenders, High Earners"	Suggests potential for growth - encourage spending among this segment.
"Value Seekers"	seem to seek value
"Budget Conscious"	might be more price sensitive?

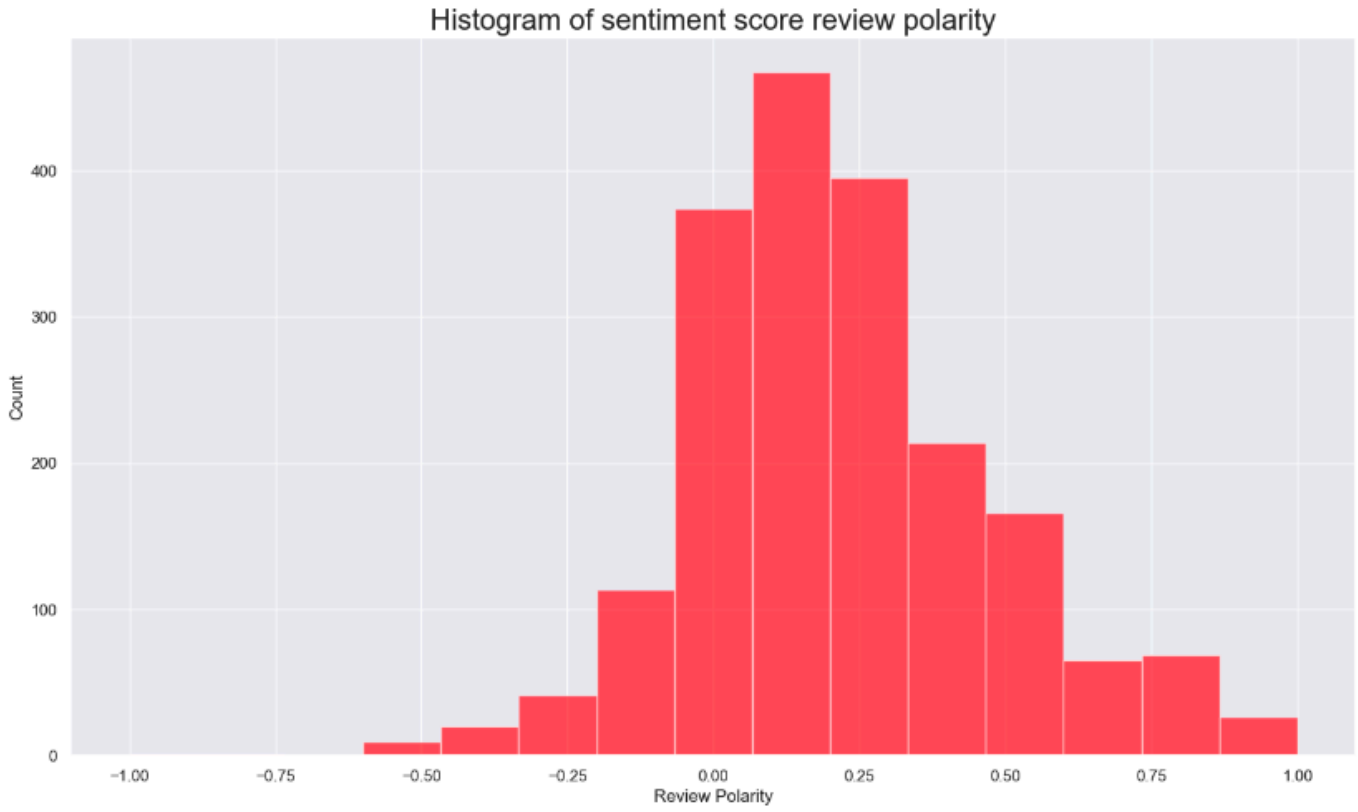
These segments can help Turtle Games predict customer behaviour, enabling more accurate forecasting and analysis of spending patterns.

c) NLP and sentiment analysis by clusters

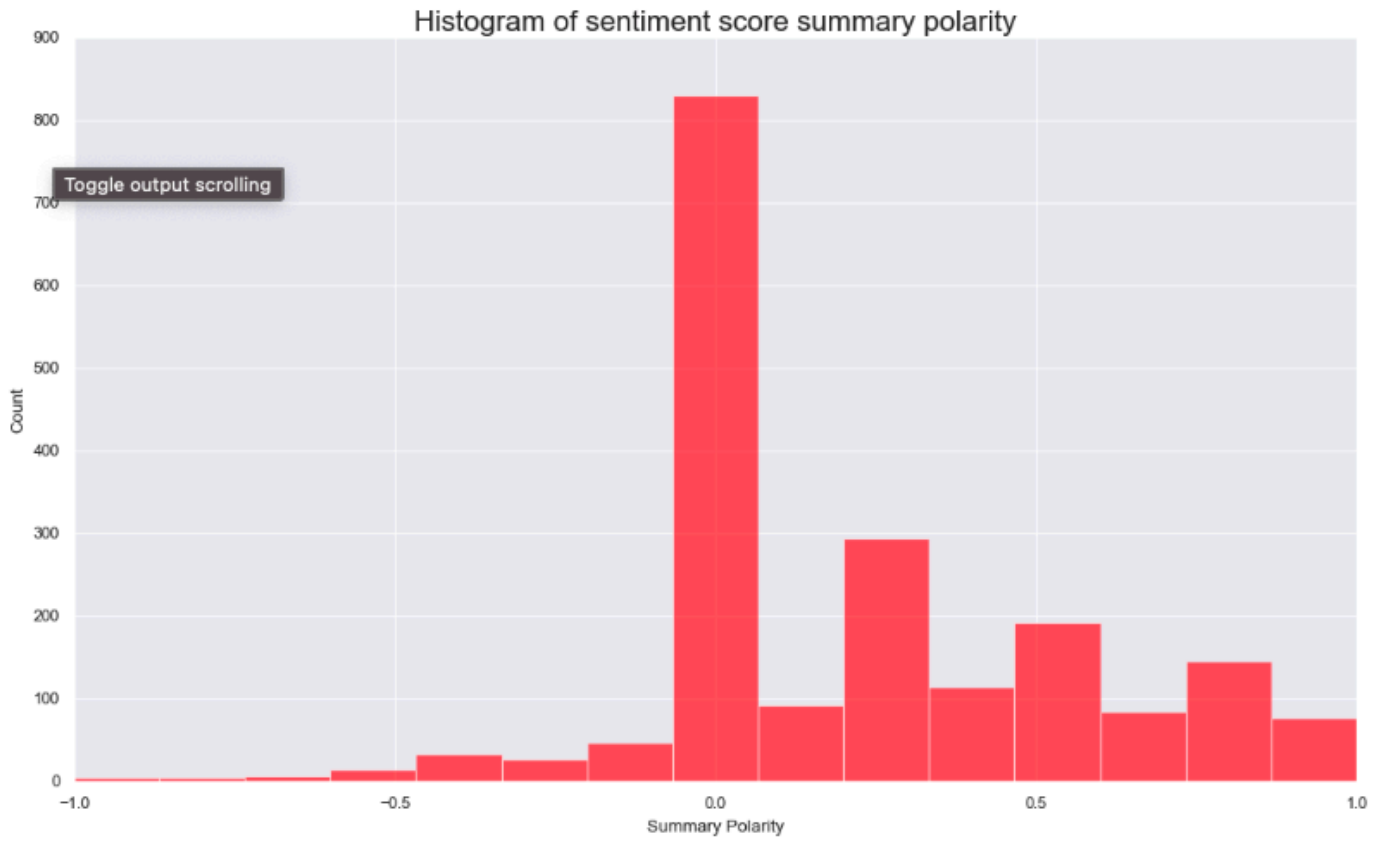
I conducted natural language processing in python based on the customer reviews within each of these clusters.

I established frequency distribution or top words, word clouds and sentiment polarity of all reviews and summaries, and then analysed reviews by cluster. I analysed review sentiment polarity overall and by cluster.

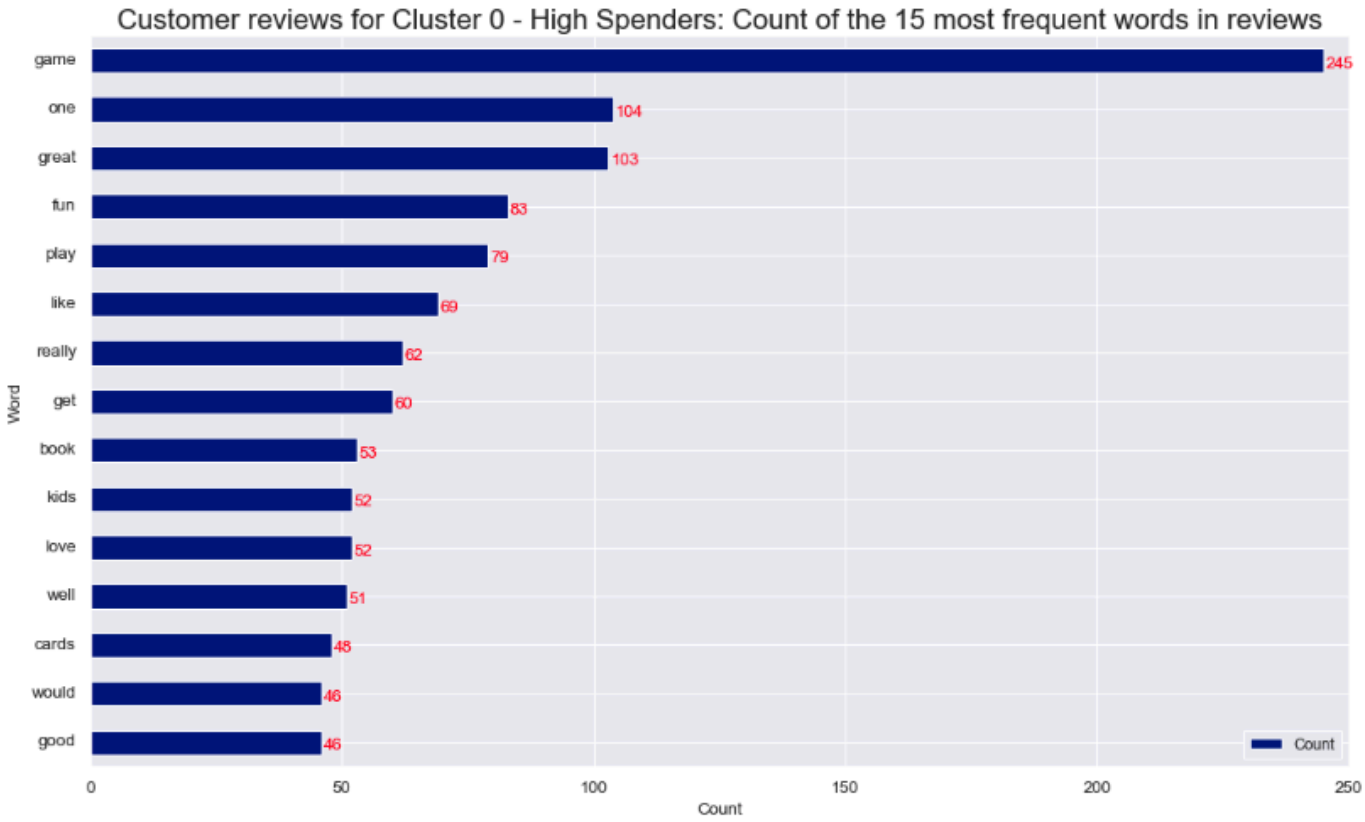
Sentiment polarity on dataset (no customer segmentation)

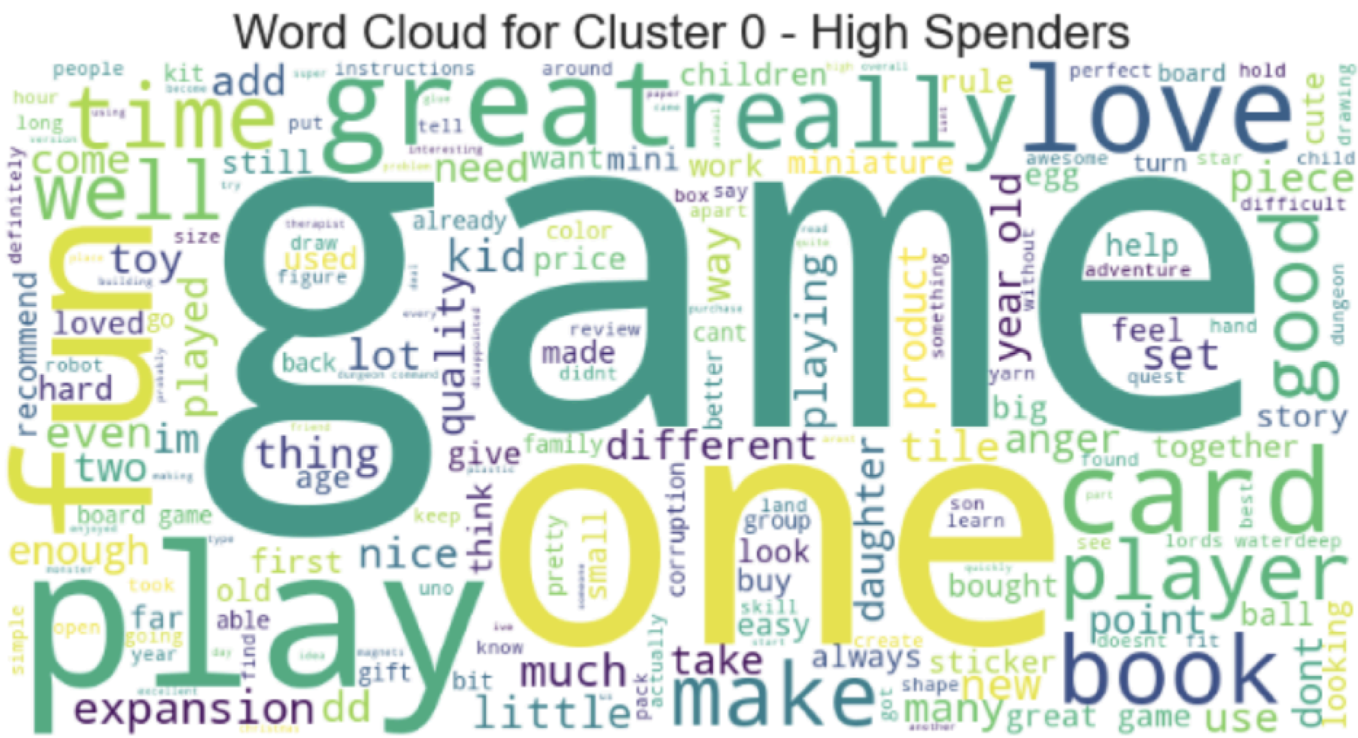


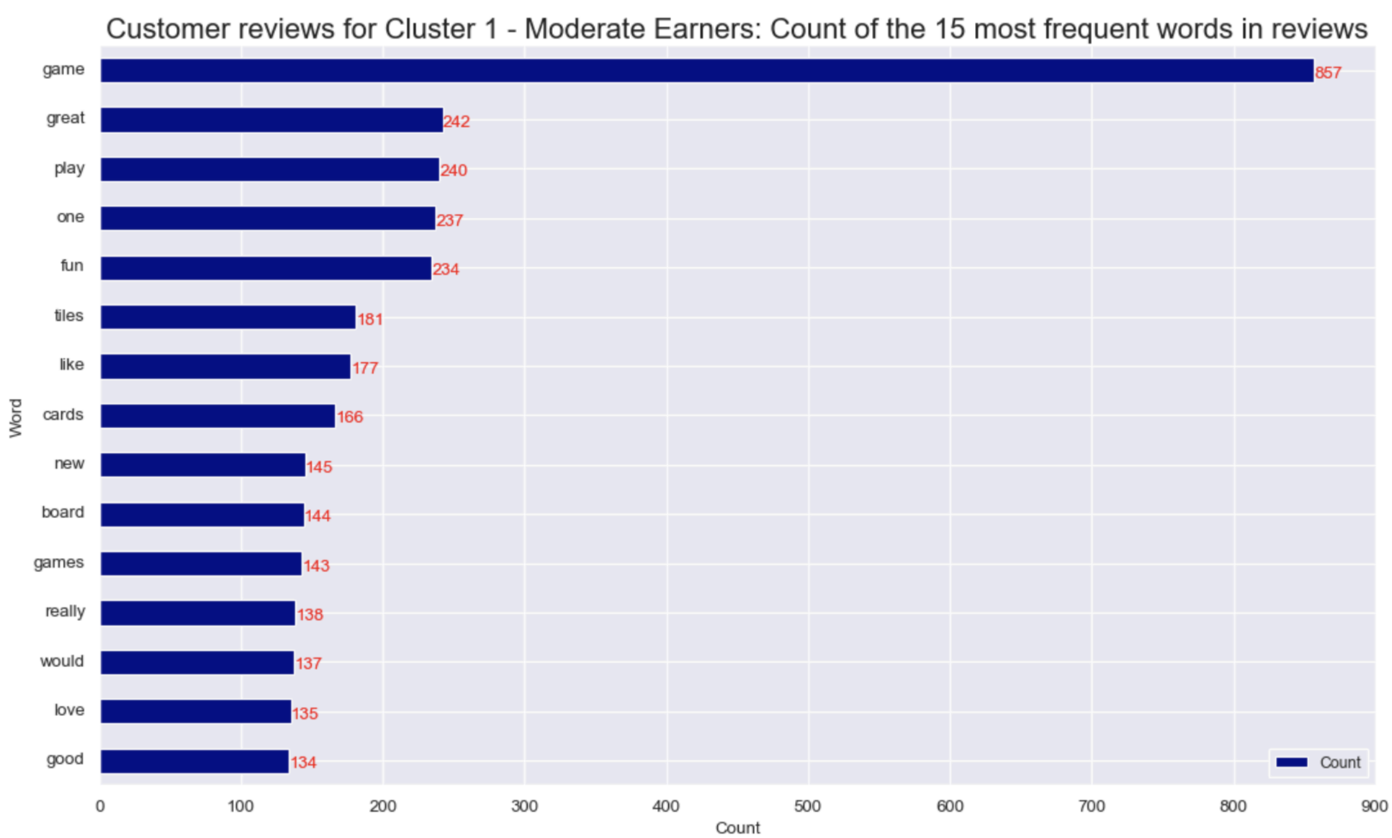
Pescod_Henry_DA301_Assignment_Report.pdf



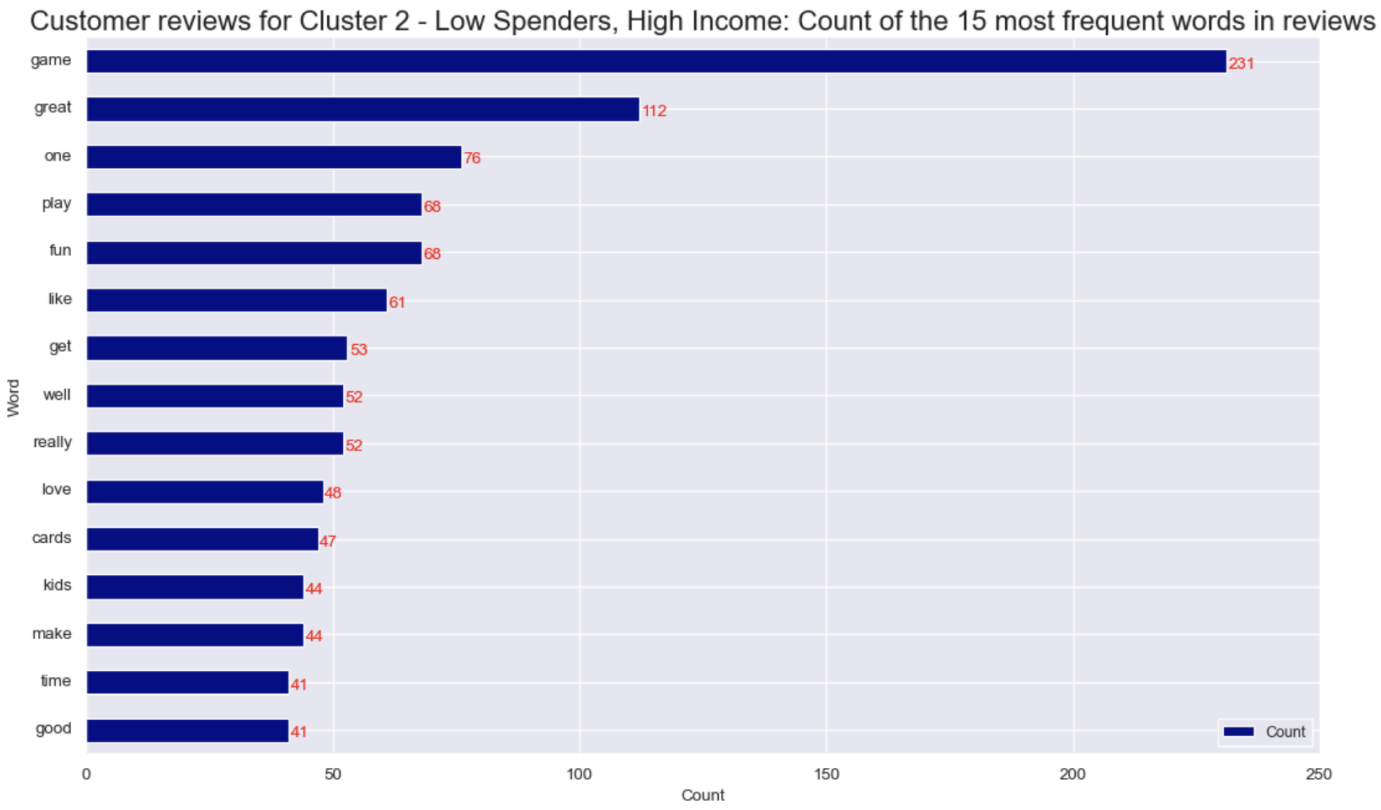
Word frequency distribution by Cluster



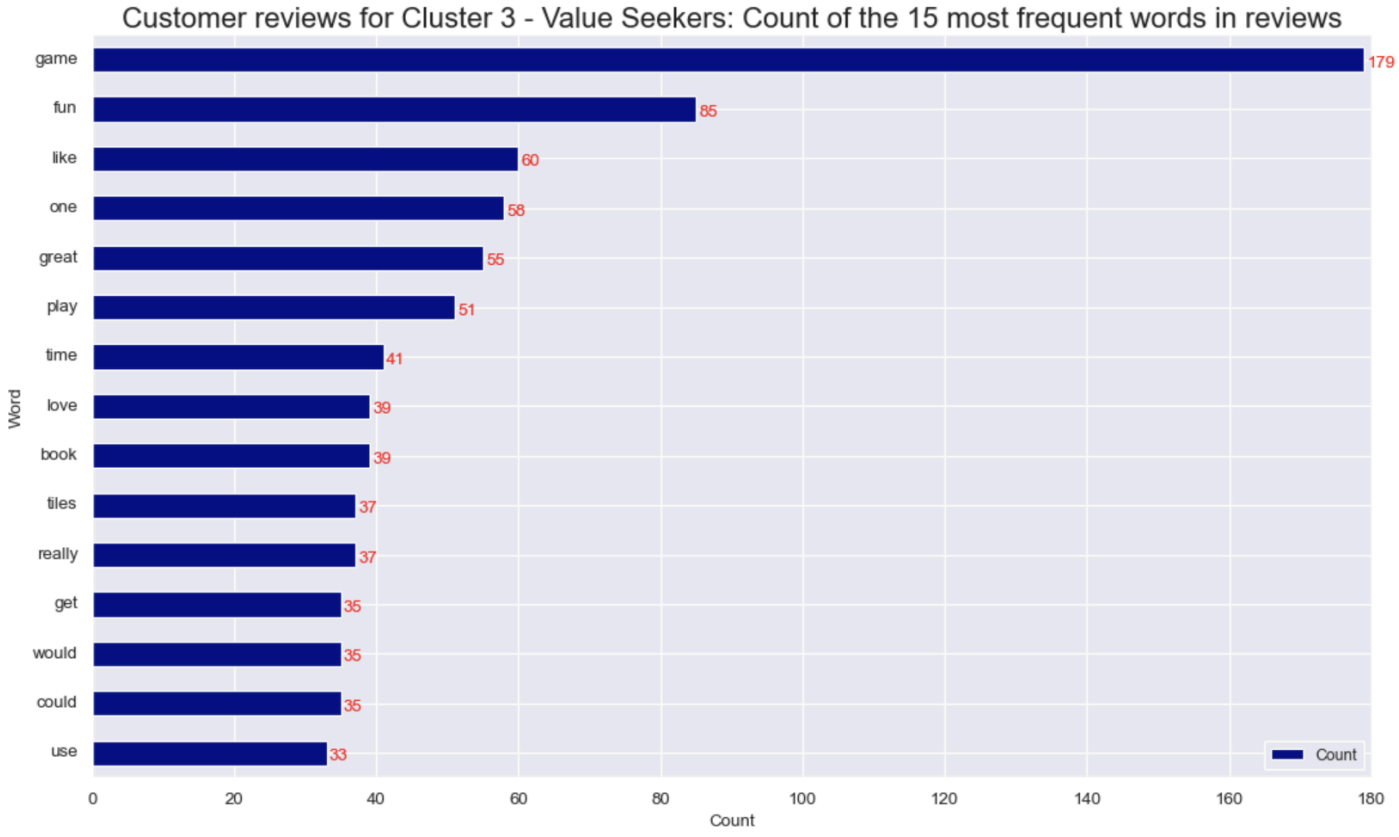

Word Cloud for Cluster 0 - High Spenders
game
one
play
fun
great
really
love
time
well
card
player
book
make
good
expansion
little
much
take
different
thing
lot
kid
price
quality
playing
product
year old
daughter
tile
anger
together
story
point
set
feel
help
piece
cute
miniature
children
rule
add
people
instructions
around
perfect
board
hold
need
want
mini
work
still
come
toy
loved
hard
played
recommend
even
im
two
enough
board game
first
nice
old
far
able
bought
always
sticker
new
many
great game
use
dont
looking
easy
buy
look
group
land
son
learn
lords waterdeep
ball
bit
gift
dd
know
uno
small
pretty
think
give
family
age
back
made
review
already
color
used
draw
figure
way
cant
better
egg
turn
star
child
difficult
adventure
dungeon
quest
big
hand
yarn

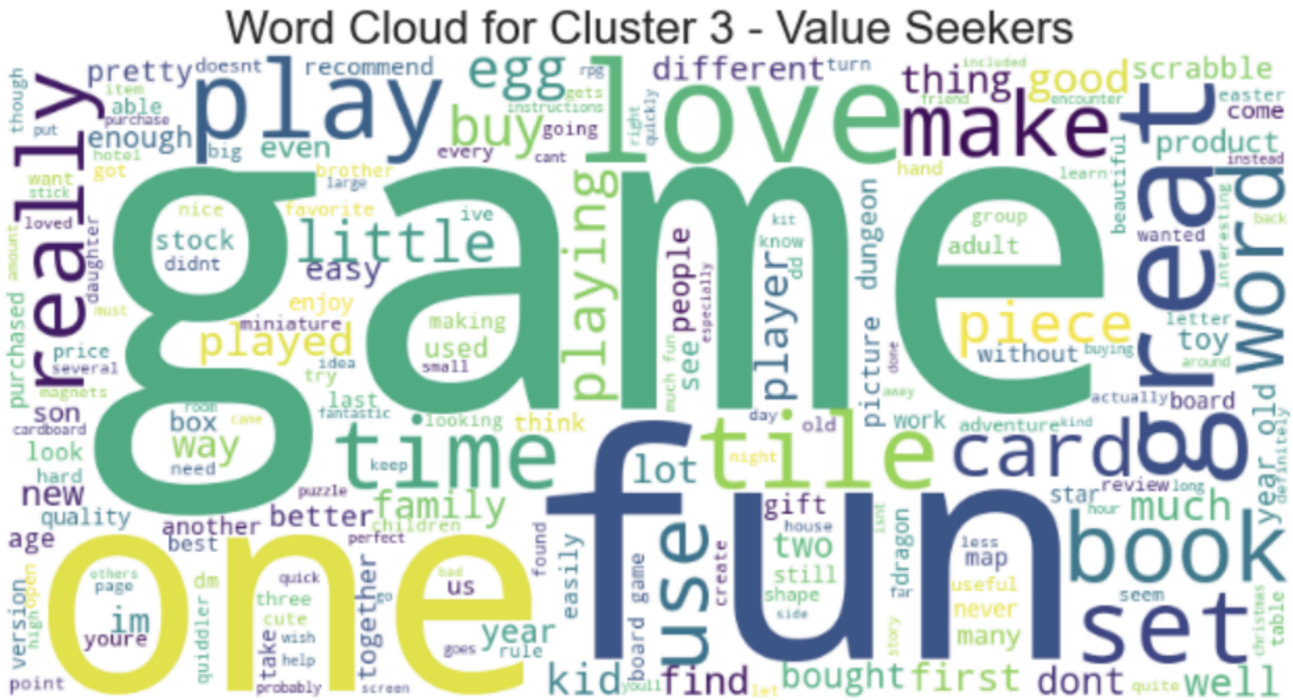

Customer reviews for Cluster 1 - Moderate Earners: Count of the 15 most frequent words in reviews
Word
Count
game 857
great 242
play 240
one 237
fun 234
tiles 181
like 177
cards 166
new 145
board 144
games 143
really 138
would 137
love 135
good 134
0
100
200
300
400
500
600
700
800
900

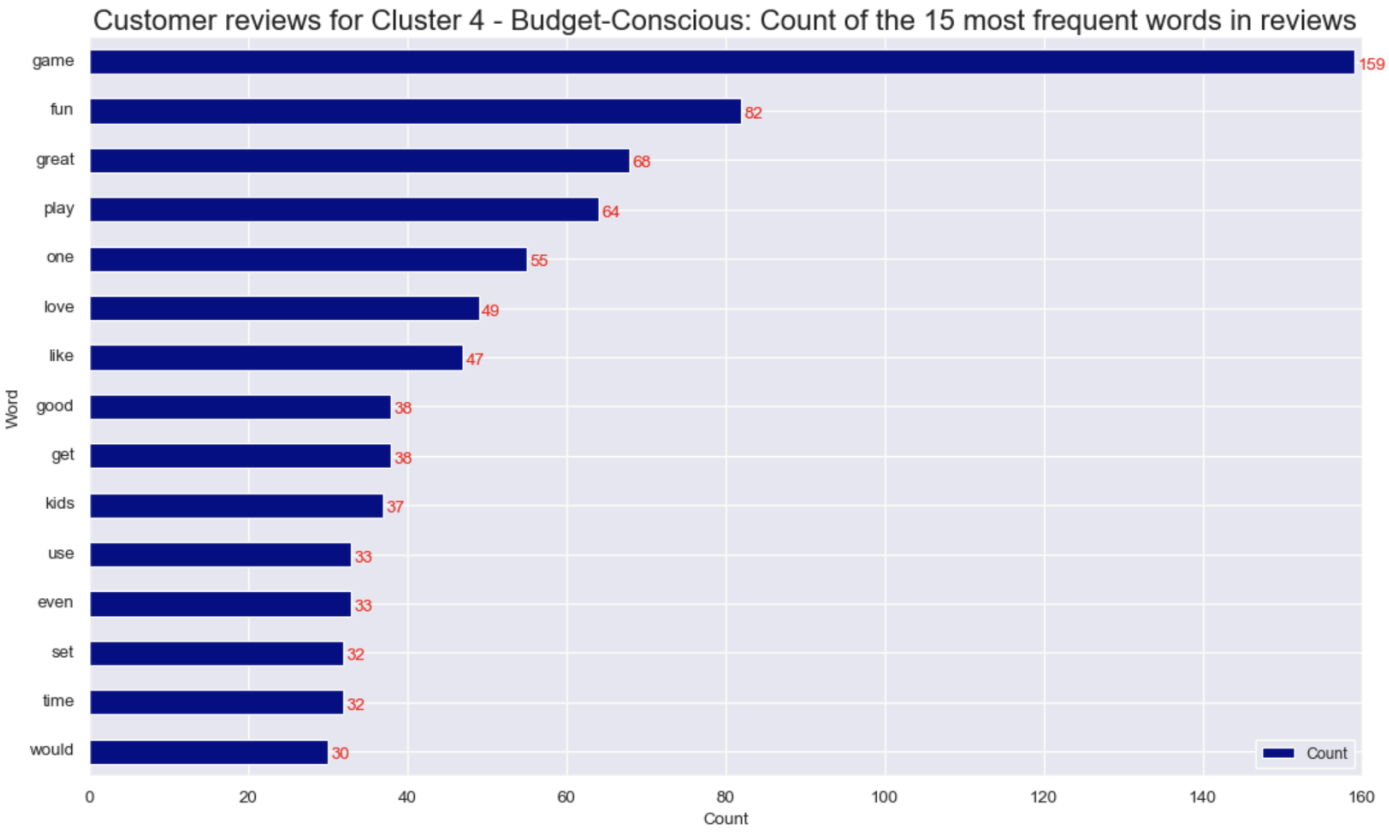
Pescod_Henry_DA301_Assignment_Report.pdf



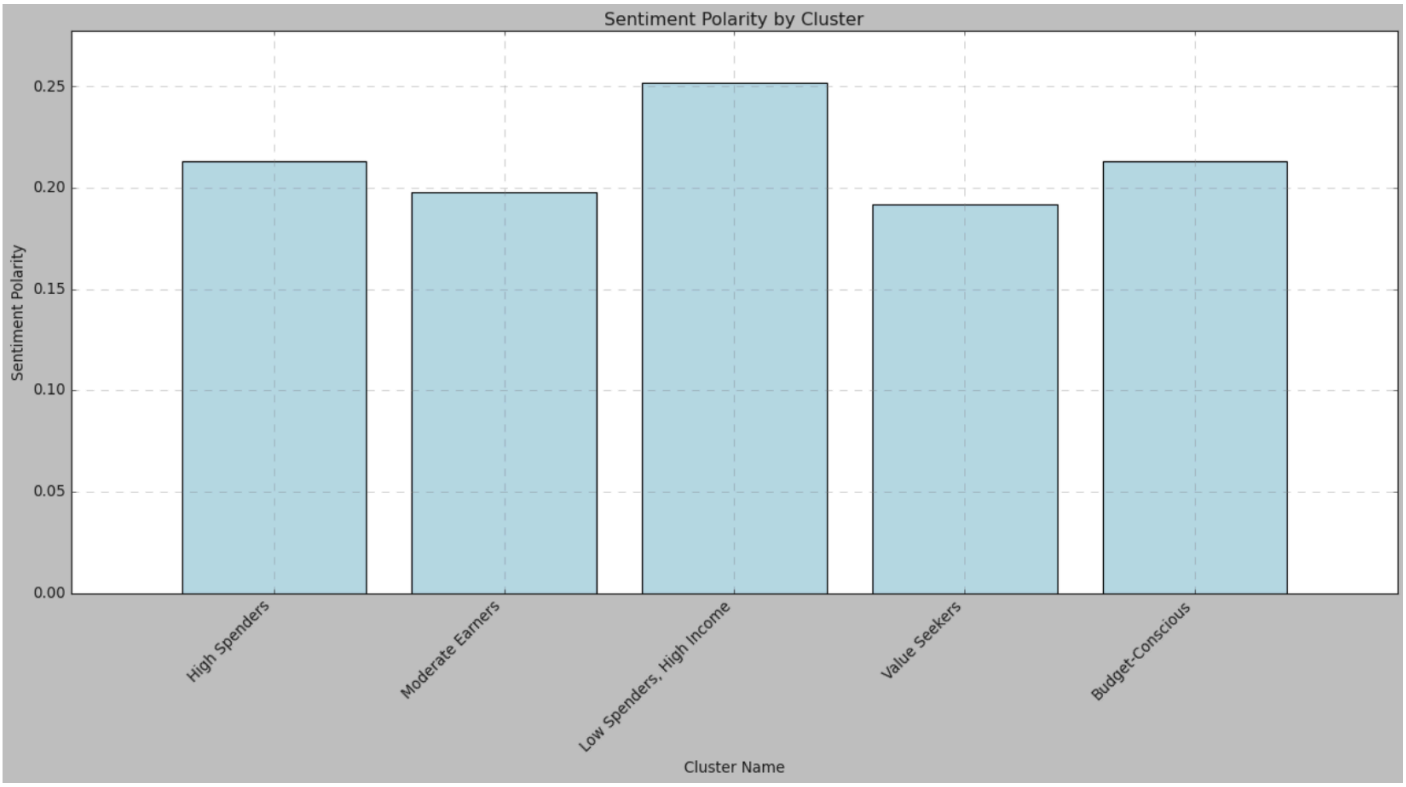
Pescod_Henry_DA301_Assignment_Report.pdf




Word Cloud for Cluster 3 - Value Seekers
game
fun
one
love
play
great
make
really
time
tile
card
book
set
use
word
egg
buy
little
playing
piece
good
player
kid
find
bought
first
dont
well
year
family
lot
gift
two
pretty
different
thing
scrabble
product
recommend
enough
even
stock
easy
played
used
making
people
dungeon
picture
adult
toy
without
much
year old
new
quality
another
better
age
best
im
youre
version
together
dragon
map
still
shape
useful
never
many
look
way
box
son
think
adventure
work
board
review
star
old
price
purchased
three
cute
us
dm
doesnt
able
big
got
want
loved
didnt
enjoy
favorite
nice
brother
going
every
letter
wanted
come
easter
instead
hand
group
know
kit
day
house
less
seem
table
point
page
take
quiddler
board game
create
found
easily



Sentiment polarity on dataset (using customer segmentation)



The sentiment polarity by cluster identified all have a positive view towards turtle game. The “low spend high income” group showed the highest positive sentiment which is possibly a good sign for attempting to target this segment for a growth in spending.

The most frequently used word across all clusters is "game," which isn't surprising given the context, also words such as "fun," "great," "play," appear a lot. The table below highlights some insights about customer behaviour based on text analysis of these reviews.

Customer cluster	Insight based on reviews
"High Spenders	Possibly family-oriented customers, given the words "kids" and "love."
"Moderate earners"	Possibly board game enthusiasts who enjoy traditional elements based on frequent words like "tiles," "cards," and "boards"
"Low Spenders, High Earners"	The words "make," "kids," and "well" could suggest a focus on games that are well-designed and engaging for children.
"Value Seekers"	Mention of "book" and "tiles" might indicate that this cluster prefers more strategic or intellectually engaging games.
"Budget Conscious"	Presence of words like "kids," "time," and "set" might suggest family-oriented customers who value easy to set up games that are fun for children

Further product analysis may be helpful to add further insight here.

4. How does this help Turtle Games?

a) Conclusions

Firstly Turtle Games should be targeting their marketing strategy to match the customer profiling. For example making sure the high spenders are well looked after as they have high future value for turtle games.

Secondly Turtle Games should be targeting the product strategy to match the customer behaviour/ For example, the target growth cluster low spend income income, could be encouraged to spend with the right product.

Lastly, the Turtle Games marketing team should use the linear regression model to predict a customer's loyalty points and segment them, with the model explaining 84% of the variability in loyalty points. To give an example, a new customer with age 36, annual income of 74,000 and spending score of 82, should generate approximately 3500 loyalty points and therefore should be treated as a “High Spender” for the purposes of marketing and product strategy.

b) Recommendations for further exploration

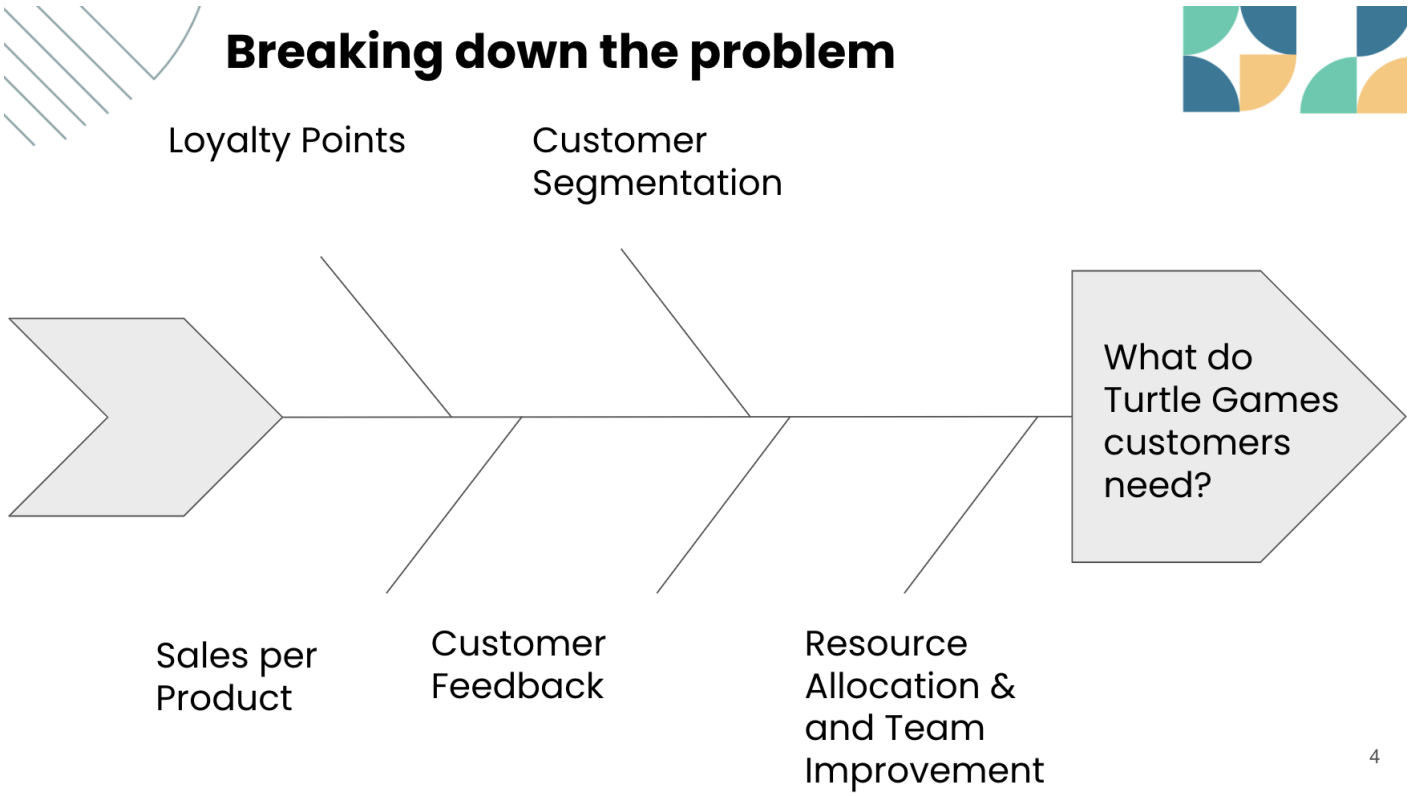
1. Incorporating detailed product data, such as descriptions and categories, would enable deeper insights into product performance and customer preferences.
2. Further model training on larger datasets, along with validation using new customer data, will improve predictive accuracy and robustness over time.

-

Word Count = 1,062

5. Appendix:

a) The business problem



b) Risks, assumptions and limitations

Review of supplied data

Column	Sample value	Interpretation of columns
gender	male or female	The gender of the customer.
age	number	Age of the customer in years.
remuneration (k£)	number	Total income per customer per year in pounds, where k=1000.
spending_score (1-100)	number	A score is assigned to the customer by Turtle Games based on the customer's spending nature and behaviour. The value ranges between 1 and 100.
loyalty_points	number	A score based on the point value of the purchase, converting the monetary value to point value, and the point value of an action (purchase).
education	graduate	Level of qualification that the customer obtained. For example: Diploma: completed school Graduate: undergraduate degree Postgraduate: postgraduate degree PhD degree.
language	EN	All the reviews were in English.
platform	Web	All the reviews were obtained from the website of Turtle Games.
product	number	Unique code allocated to product based on the item description.
review	When it comes to a DM's screen, ...	Online reviews submitted by customers who purchased and used the products.
summary	The fact that 50% of this ...	Summary of the customer's review

TG Resources information?

- Data set of 2,000. All assumptions made in this analysis would need to be tested on a larger data set.
- Sales per product analysis
    - - product-specific sales data to identify underperforming products.
    - - Sales data provided did not provide, any indication of product type. This was due to only having product code data, rather than product descriptions and product categories. therefore I was unable to properly analyse with the data provided
- Resource Allocation
- Based on the findings more recommendations unable to properly analyse with the data provided
    - Analysis of the problem component 'resource allocation' would also be limited as there is no data provided on turtle games employees or work information

c) Importing the necessary libraries and datasets provided and sensecheck

My first steps were to import all required python packages and libraries, and import and clean the data.

Full list of libraries imported
Import all required packages and libraries to be used in this notebook

# Import all the necessary packages
import numpy as np

```
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
import statsmodels.api as sm
from statsmodels.formula.api import ols
import math
import warnings
import nltk
import os

# Import all the necessary libraries
from sklearn.tree import DecisionTreeRegressor, plot_tree
from sklearn.metrics import accuracy_score, confusion_matrix, ConfusionMatrixDisplay
from sklearn.metrics import classification_report
from sklearn.model_selection import train_test_split
from sklearn import metrics
from sklearn.metrics import mean_absolute_error, mean_squared_error, r2_score
from sklearn.cluster import KMeans
from sklearn.preprocessing import StandardScaler
from sklearn.cluster import KMeans
from sklearn.metrics import silhouette_score
from scipy.spatial.distance import cdist
from wordcloud import WordCloud
from nltk.tokenize import word_tokenize
from nltk.probability import FreqDist
from nltk.corpus import stopwords
from textblob import TextBlob
from scipy.stats import norm
from collections import Counter
!pip install Wordcloud
!pip install textblob
```

d) Cleaning the data - method

Cleaning of the data involved the following:

- I remove redundant columns (language and platform)

- I changed column headings to names that are easier to reference (e.g. remuneration and spending_score.

1b. Drop columns

[12]:
```
# Drop unnecessary columns.

reviews_new = reviews.drop(['language', 'platform'],
                           axis=1)
# View column names.
reviews_new.columns
```

[12]:
```
Index(['gender', 'age', 'remuneration (k£)', 'spending_score (1-100)',
       'loyalty_points', 'education', 'product', 'review', 'summary'],
      dtype='object')
```

1c. Rename columns

[21]:
```
# Rename the column headers.
reviews_new = reviews_new.rename(columns={'remuneration (k£)':'remuneration', \
                                          'spending_score (1-100)':'spending_score'})

# View column names.
reviews_new.columns
```

[21]:
```
Index(['gender', 'age', 'remuneration', 'spending_score', 'loyalty_points',
       'education', 'product', 'review', 'summary'],
      dtype='object')
```

Pescod_Henry_DA301_Assignment_Report.pdf

- I saved a copy of the clean DataFrame as a CSV file.

1d. Save the DataFrame as a CSV file

23]:
```
# Create a CSV file as output.

reviews_new.to_csv('reviews_new.csv', index=False)
```

25]:
```
# Import new CSV file with Pandas.

turtle_reviews = pd.read_csv('reviews_new.csv')

# View DataFrame.
turtle_reviews.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 2000 entries, 0 to 1999
Data columns (total 9 columns):
 #   Column          Non-Null Count  Dtype
---  ------          --------------  -----
 0   gender          2000 non-null   object
 1   age             2000 non-null   int64
 2   remuneration    2000 non-null   float64
 3   spending_score  2000 non-null   int64
 4   loyalty_points  2000 non-null   int64
 5   education       2000 non-null   object
 6   product         2000 non-null   int64
 7   review          2000 non-null   object
 8   summary         2000 non-null   object
dtypes: float64(1), int64(4), object(4)
memory usage: 140.8+ KB
```

- I checked for missing values and checked the data types of the columns

[8]:
```
# Any missing values?

# Determine the number of missing values.
reviews.isna().sum()
```

[8]:
```
gender                   0
age                      0
remuneration (k£)        0
spending_score (1-100)   0
loyalty_points           0
education                0
language                 0
platform                 0
product                  0
review                   0
summary                  0
dtype: int64
```

[9]:
```
# Explore the data.

reviews.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 2000 entries, 0 to 1999
Data columns (total 11 columns):
 #   Column                  Non-Null Count  Dtype
---  ------                  --------------  -----
 0   gender                  2000 non-null   object
 1   age                     2000 non-null   int64
 2   remuneration (k£)       2000 non-null   float64
 3   spending_score (1-100)  2000 non-null   int64
 4   loyalty_points          2000 non-null   int64
 5   education               2000 non-null   object
 6   language                2000 non-null   object
 7   platform                2000 non-null   object
 8   product                 2000 non-null   int64
 9   review                  2000 non-null   object
 10  summary                 2000 non-null   object
dtypes: float64(1), int64(4), object(6)
memory usage: 172.0+ KB
```

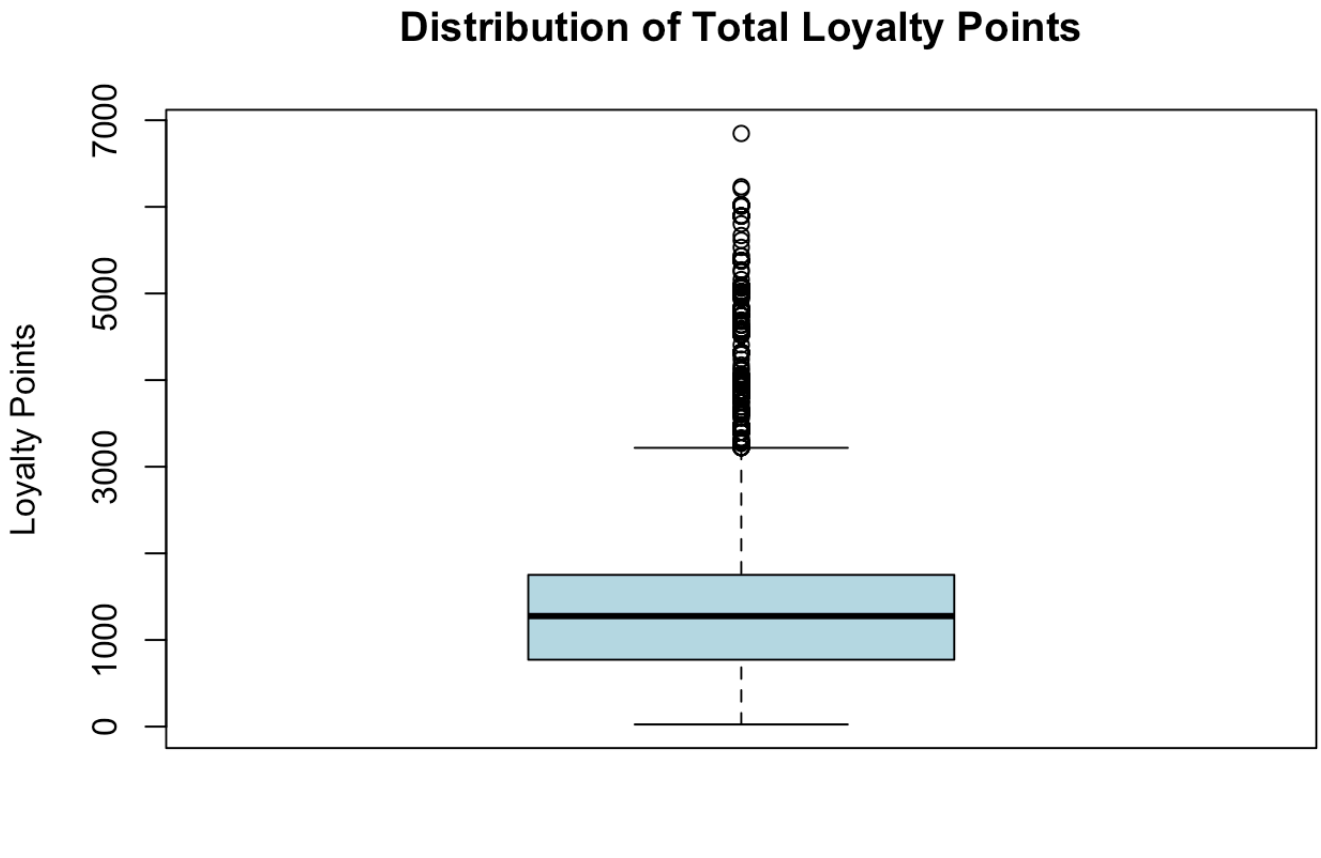
- Conducted the same steps in R

```
################################################################################

# 2. Load and explore the data

# Read the CSV file (reviews_new.csv).
reviews_new <- read.csv('reviews_new_5_clusters.csv', header=TRUE)

# View the first six lines of the data frame.
head(reviews_new)
str(reviews_new)

# Review/sense-check the data set.
as_tibble (reviews_new)

# These functions provide summary statistics of the data set.
summary(reviews_new)

View(reviews_new)

# Count number of unique values in product column
reviews_new %>%
  count(product)

# To search for missing values in a data set.
reviews_new[is.na(reviews_new)]

################################################################################
```

e) Initial Review of data - exploratory data analysis

Key questions to answer:
What is the distribution of the data?
Once I was satisfied the data set was clean, I conducted exploratory data analysis in Python and R using a combination of scatter plots, box plots, histograms, bar plots and correlation plots to review the distribution of the data and gained an understanding of the customer demographics of the customers.
I used a box plot to show the summary statistics

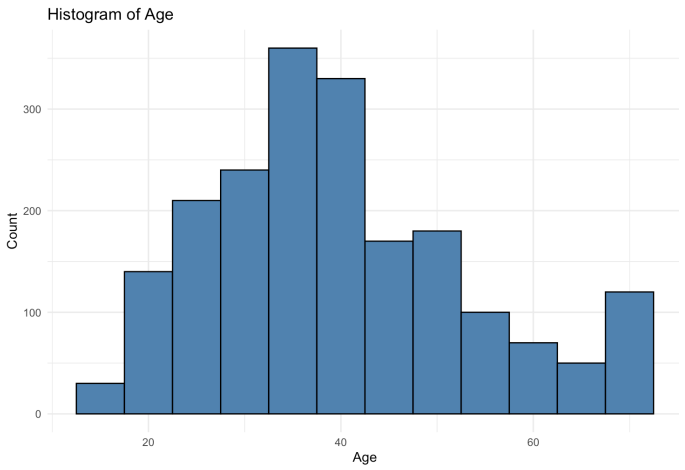
```r
# visualise data with boxplot to determine normal

# Boxplot using base R
boxplot(reviews_new$loyalty_points,
        col = "lightblue",
        border = "black",
        main = "Distribution of Total Loyalty Points",
        ylab = "Loyalty Points",
        xlab = "")
```

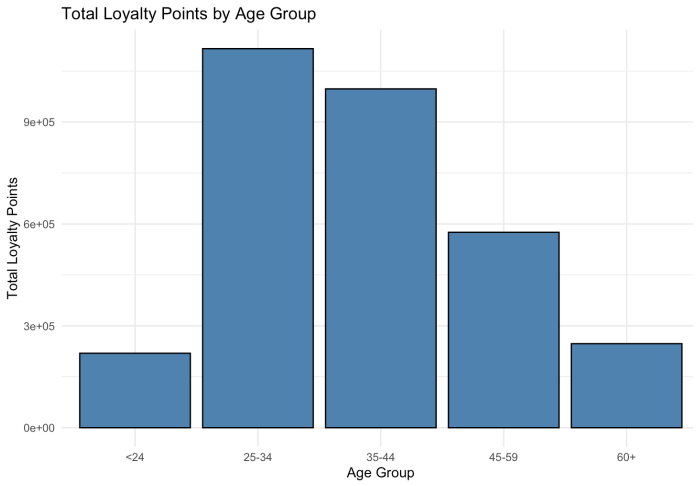


Histograms and bar plot with age group (bins) to highlight the count of age, and age grouped by loyalty points

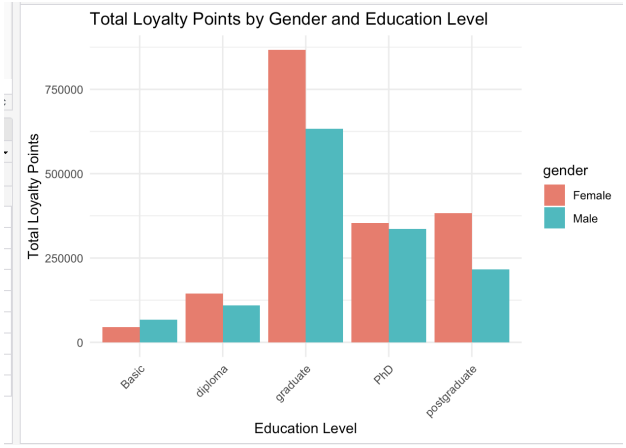
Pescod_Henry_DA301_Assignment_Report.pdf

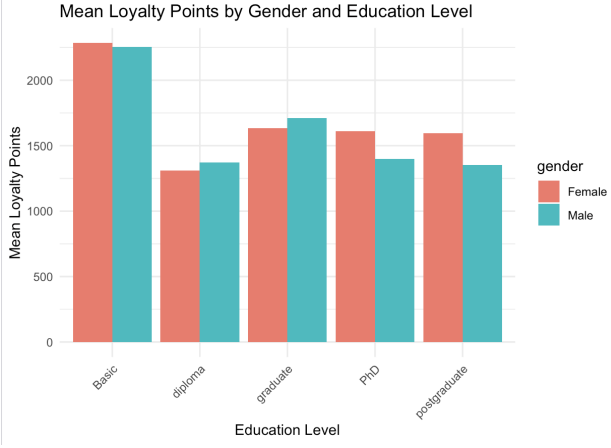
```r
# visualise data with histogram

# Histogram of age
ggplot(reviews_new,aes(x = age)) +
  # Add a geom layer to specify the plot type
  geom_histogram(stat = 'count')

# Histogram of age with assigned bin widths
ggplot(reviews_new, aes(x = age)) +
  # Add a geom layer to specify the plot type with bin width
  geom_histogram(binwidth = 5, fill = "steelblue", color = "black") +
  labs(title = "Histogram of Age",
       x = "Age",
       y = "Count") +
  theme_minimal()

##########################################################################################
```



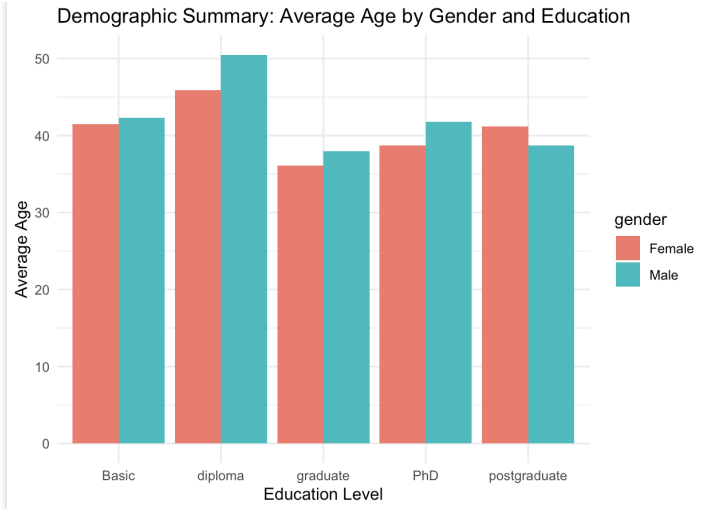


- Demographic summary from exploratory data analysis in R
- Education vs loyalty points & gender (total vs mean)

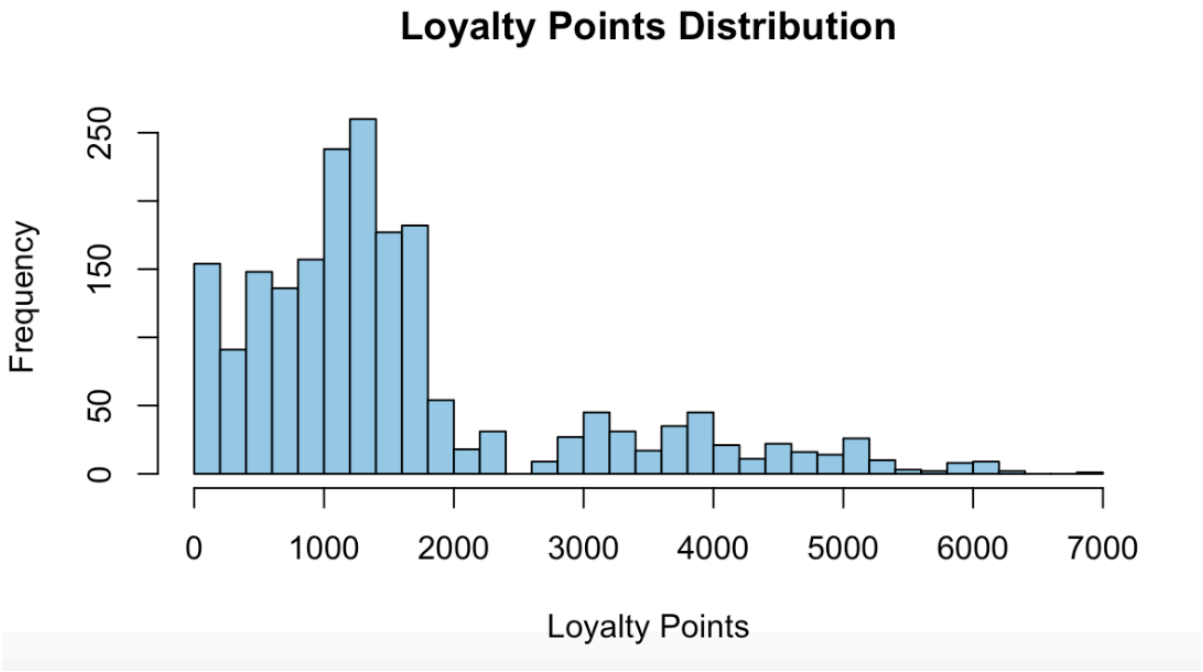




Pescod_Henry_DA301_Assignment_Report.pdf



Loyalty points distribution (using R) to highlight the right / positive skewed of the data.



Further investigation of the outliers in R.

```
#########################################################################
# Investigate the outliers further
# Step 1 - visualise the distribution

# Histogram and density plot for loyalty_points
hist(reviews_new$loyalty_points, main= "Loyalty Points Distribution", xlab = "Loyalty Points", col = "skyblue", breaks
plot(density(reviews_new$loyalty_points), main = "Loyalty Points Density", col = "red")

# Repeat for other variables like renumeration, spending_score, and age
hist(reviews_new$renumeration, main = "Renumeration Distribution", xlab = "Renumeration", col = "lightgreen", breaks =
plot(density(reviews_new$renumeration), main = "Renumeration Density", col = "green")

# Boxplot for loyalty_points
boxplot(reviews_new$loyalty_points, main = "Boxplot of Loyalty Points", col = "lightblue")

# Boxplot for other variables
boxplot(reviews_new$renumeration, main = "Boxplot of Renumeration", col = "lightgreen")
boxplot(reviews_new$spending_score, main = "Boxplot of Spending score", col = "blue")
boxplot(reviews_new$age, main = "Boxplot of Age", col = "pink")
```

```
# Step 2 - Identify outliers
# Calculate the IQR for loyalty_points
iqr_loyalty_points <- IQR(reviews_new$loyalty_points)
upper_bound <- quantile(reviews_new$loyalty_points, 0.75) + 1.5 * iqr_loyalty_points
lower_bound <- quantile(reviews_new$loyalty_points, 0.25) - 1.5 * iqr_loyalty_points

# Identify outliers
outliers <- reviews_new$loyalty_points[reviews_new$loyalty_points > upper_bound | reviews_new$loyalty_points < lower_bo

# Print outliers
print(outliers)

# Step 3: investigate the extreme values
# Subset the data to focus on outliers
outlier_data <- reviews_new[reviews_new$loyalty_points > upper_bound | reviews_new$loyalty_points < lower_bound, ]

# Check if outliers are driven by renumeration, spending score, or age
summary(outlier_data)
```

f) Linear Regression - method

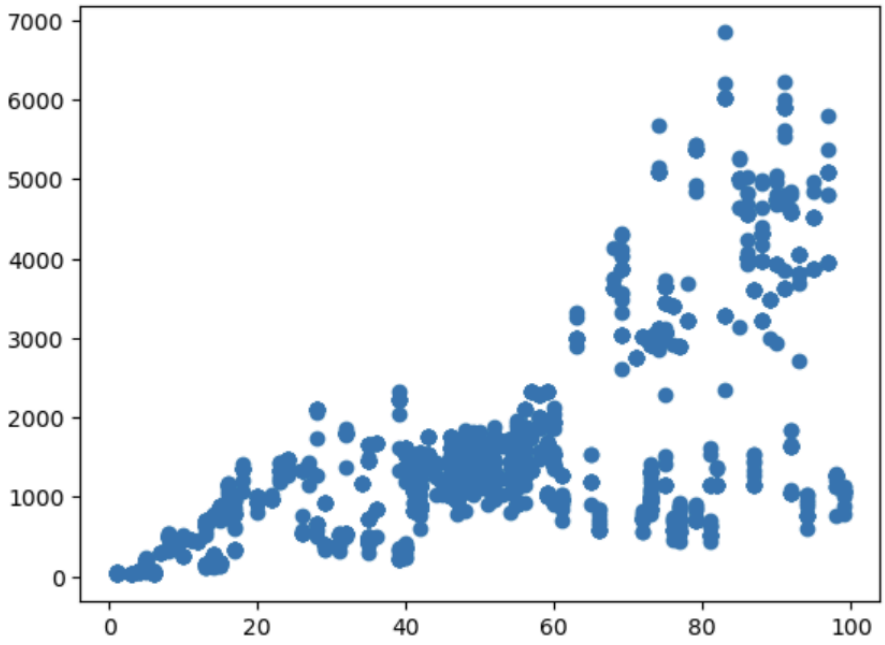
I conducted linear regression analysis in Python and R to evaluate possible linear relationships between loyalty points, age, remuneration and spending scores. This would help determine whether these could be used to predict the accumulation of loyalty points.

- Specify the independent and dependent variables.

i) Whats the relationship between spending score vs loyalty points?

[27]:
```
# Define independent variable.
x = turtle_reviews['spending_score']

# Define dependent variable.
y = turtle_reviews['loyalty_points']

# Check for linearity with Matplotlib.
plt.scatter(x, y)
```

[27]:
```
<matplotlib.collections.PathCollection at 0x13aea6c30>
```



- Create the OLS model.

[1] Standard Errors assume that the covariance matrix of the errors is correctly specified.

[31]:
```
# Extract the estimated parameters.
print("Parameters: ", test1.params)

# Extract the standard errors.
print("Standard errors: ", test1.bse)

# Extract the predicted values.
print("Predicted values: ", test1.predict())
```

```
Parameters:  Intercept   -75.052663
x            33.061693
dtype: float64
Standard errors:  Intercept    45.930554
x             0.814419
dtype: float64
Predicted values:  [1214.35337415 2602.94449102  123.31749662 ... 2933.56142361  453.93442921
  189.44088314]
```

[33]:
```
# Create the linear regression model.
```

- Extract the estimated parameters, standard errors, and predicted values.
- Generate the regression table based on the X coefficient and constant values.

Notes:

[1] Standard Errors assume that the covariance matrix of the errors is correctly specified.

[31]:
```
# Extract the estimated parameters.
print("Parameters: ", test1.params)

# Extract the standard errors.
print("Standard errors: ", test1.bse)

# Extract the predicted values.
print("Predicted values: ", test1.predict())
```

```
Parameters:  Intercept   -75.052663
x            33.061693
dtype: float64
Standard errors:  Intercept    45.930554
x             0.814419
dtype: float64
Predicted values:  [1214.35337415 2602.94449102  123.31749662 ... 2933.56142361  453.93442921
  189.44088314]
```

[33]:
```
# Create the linear regression model.
# Set the X coefficient to 33.061 and the constant to -75.052 to generate the regression table..
y_pred_1 = (-75.052) + 33.061 * turtle_reviews['spending_score']

# View the output.
y_pred_1
```

[33]:
```
0       1214.327
1       2602.889
2        123.314
3       2470.645
4       1247.388
          ...
1995    2206.157
1996     189.436
1997    2933.499
1998     453.924
1999     189.436
Name: spending_score, Length: 2000, dtype: float64
```

-

[29]:
```
# Create model and print summary of metrics.
f1 = 'y ~ x'
test1 = ols(f1, data = turtle_reviews).fit()

# Print the regression table.
test1.summary()
```

[29]:

OLS Regression Results

Dep. Variable:	y	**R-squared:**	0.452
Model:	OLS	**Adj. R-squared:**	0.452
Method:	Least Squares	**F-statistic:**	1648.
Date:	Sun, 29 Sep 2024	**Prob (F-statistic):**	2.92e-263
Time:	18:18:47	**Log-Likelihood:**	-16550.
No. Observations:	2000	**AIC:**	3.310e+04
Df Residuals:	1998	**BIC:**	3.312e+04
Df Model:	1		
Covariance Type:	nonrobust		

	coef	std err	t	P>|t|	[0.025	0.975]
Intercept	-75.0527	45.931	-1.634	0.102	-165.129	15.024
x	33.0617	0.814	40.595	0.000	31.464	34.659

Omnibus:	126.554	**Durbin-Watson:**	1.191
Prob(Omnibus):	0.000	**Jarque-Bera (JB):**	260.528
Skew:	0.422	**Prob(JB):**	2.67e-57
Kurtosis:	4.554	**Cond. No.**	122.

- A positive correlation evident between spending score and loyalty points (.452 r2).
- A heavy tailed positive skew highlighted by Kurtosis figure of 4.554
- A positive correlation evident between remuneration and loyalty points (0.380 r2).
- A heavy tailed positive skew highlighted by Kurtosis figure of 3.59

- Plot the linear regression and add a regression line.

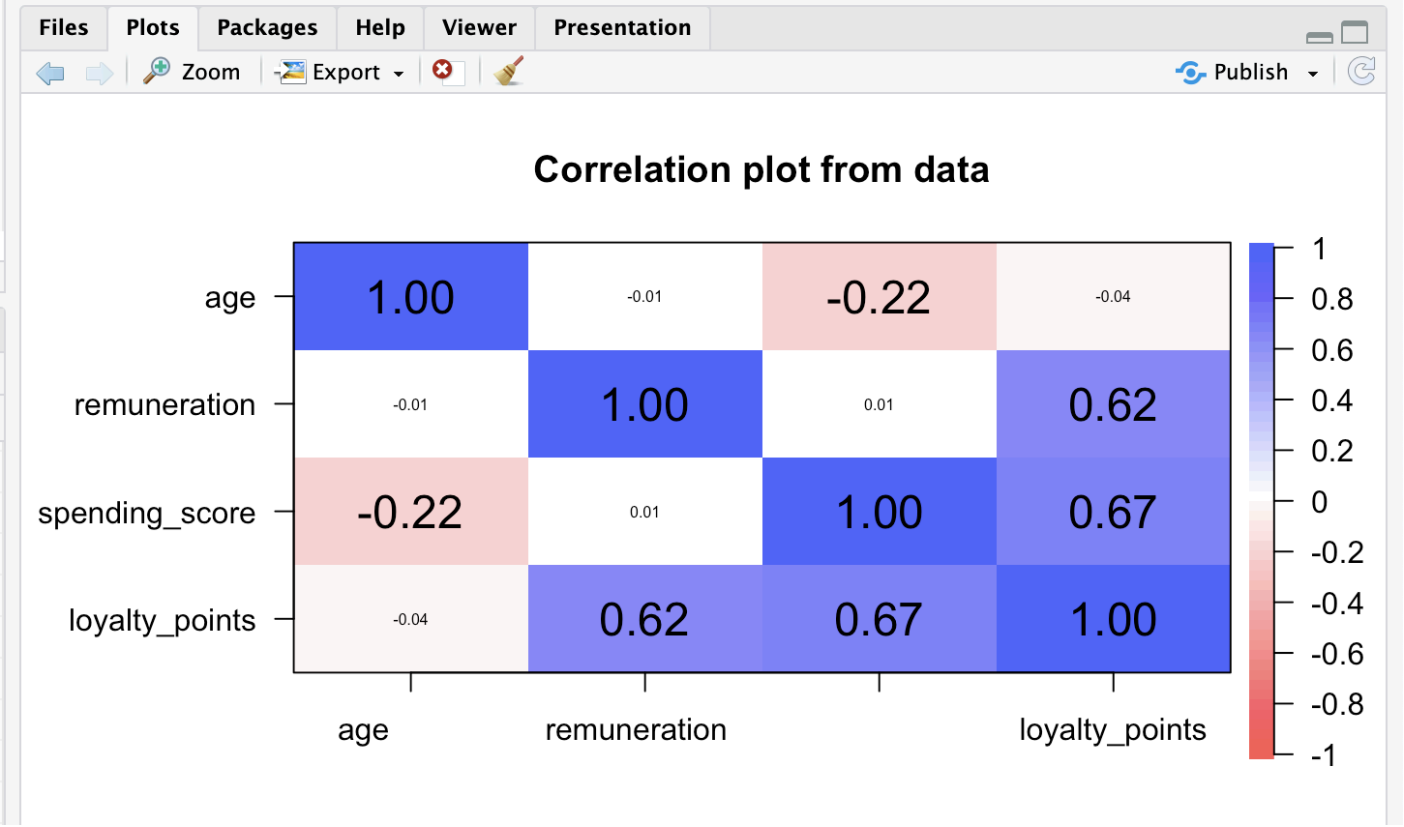
```
# 3. Exploratory data analysis and visualisation

# Scatter plot of spend vs loyalty points with a linear regression line
ggplot(data = reviews_new,
       mapping = aes(x = spending_score, y = loyalty_points)) +
  geom_point(color = 'blue', alpha = 0.6) +
  geom_smooth(method = "lm", color = "red", se = FALSE) +
  labs(title = "Loyalty Points vs. Spending score with Trend Line", x = "Spending Score", y = "Loyalty
  theme_minimal()

##############################################################################

# Scatter plot of remuneration vs loyalty points with a linear regression line
ggplot(data = reviews_new,
       mapping = aes(x = remuneration, y = loyalty_points)) +
  geom_point(color = "blue", alpha = 0.6) +
  geom_smooth(method = "lm", color = "red", se = FALSE) +
  labs(title = "Loyalty Points vs. Remuneration with Trend Line", x = "Remuneration", y = "Loyalty Poin
  theme_minimal()

##############################################################################
```

g) Correlation Plot - method

I used a correlation plot which mapped out the effect of the numeric variables on the dependent variable (loyalty points).

```
# Correlation Plot
install.packages('psych')

# Import the psych package
library(psych)

# Remove the 'product' and 'cluster' variables from the data frame
reviews_new_subset <- reviews_new[, !(names(reviews_new) %in% c("product", "cluster'

# Check the column names in the data frame
print(names(reviews_new_subset))

# Use the corPlot() function
# Specify the numeric variables from the data frame (reviews_new)
# Character size (cex=1)
corPlot(reviews_new_subset[, sapply(reviews_new_subset, is.numeric)], cex=1)
```

```
> # Use the corPlot() function
> # Specify the numeric variables from the data frame (reviews_new)
> # Character size (cex=1)
> corPlot(reviews_new_subset[, sapply(reviews_new_subset, is.numeric)], cex=1)
>
```



h) Decision tree Regression - method

Using the decision tree regressor library in sklearn.tree, I trained and tested 5 different decision tree regressor models each with different tree depths. I checked each model test metrics against the validation metrics to check for overfitting.

I compiled the output metrics into a table to compare the accuracy of the decision tree regression model in predicting loyalty points. This includes error metrics such as the Mean Squared Error (MSE), Mean Absolute Error (MAE), and the R-squared (R^2) value, which quantifies how well the model fits the data. These metrics helped to evaluate the model's performance and provide insight into its predictive accuracy.

I split the data into a 3 parts:

- A training set (60%) to train the model
- A test set (20%) to test the model

- A validation set (20%) to validate the results and check for overfitting

Python code used to split the data

```
#### iii) Define variables

[55]: # Specify Y.
y = df_fin['loyalty_points']

# Specify X.
X = df_fin[['age', 'remuneration', 'spending_score']]

#### iv). Create training, testing and validation data sets (60% train / 20% test / 20% validate)

[55]: # Split the data into 60% training, 20% validation, and 20% test sets
X_train, X_temp, y_train, y_temp = train_test_split(X, y, test_size=0.4, random_state=42)
X_val, X_test, y_val, y_test = train_test_split(X_temp, y_temp, test_size=0.5, random_state=42)

#### v). Create Decision tree regressor

[57]: # Create decision tree regressor.

# Create the 'DecisionTreeRegressor' class
regressor = DecisionTreeRegressor(random_state=42)

# Fit the regressor object to the data set.
regressor.fit(X_train,y_train)

[57]: DecisionTreeRegressor
DecisionTreeRegressor(random_state=42)
```

Using the decision tree regressor library in sklearn.tree, I trained and tested 5 different decision tree regressor models each with different tree depths. I checked each model test metrics against the validation metrics to check for overfitting.

I compiled the output metrics into a table to compare the accuracy of the decision tree regression model in predicting loyalty points. This includes error metrics such as the Mean Squared Error (MSE), Mean Absolute Error (MAE), and the R-squared (R^2) value, which quantifies how well the model fits the data. These metrics helped to evaluate the model's performance and provide insight into its predictive accuracy.

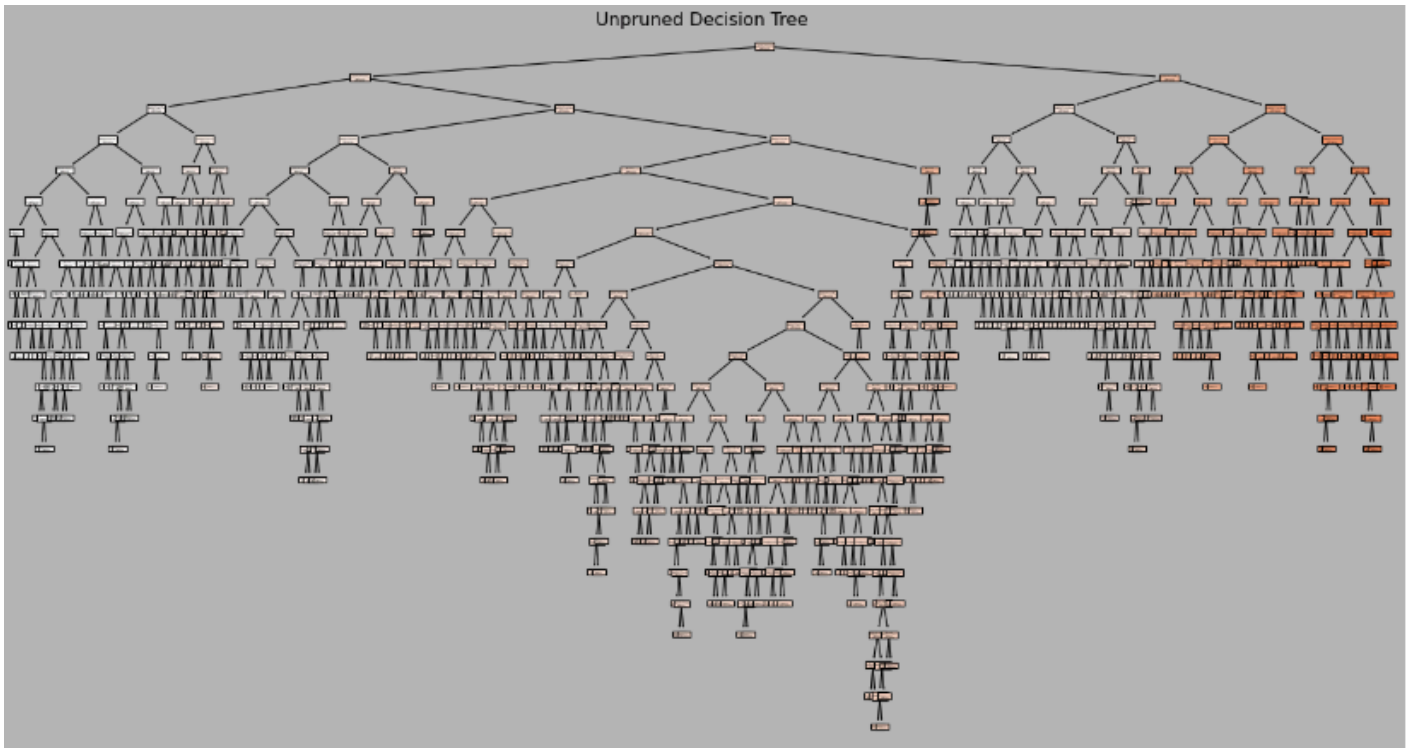
[58]:
```
# Evaluate the model on the test set.
y_test_predict_unpruned = regressor.predict(X_test)

# Calculate metrics for the test set
mae_test_unpruned = metrics.mean_absolute_error(y_test, y_test_predict_unpruned)
mse_test_unpruned = metrics.mean_squared_error(y_test, y_test_predict_unpruned)
rmse_test_unpruned = math.sqrt(metrics.mean_squared_error(y_test, y_test_predict_unpruned))
r2_test_unpruned = r2_score(y_test, y_test_predict_unpruned)

# Print the metrics for the test set
print(f"Test Metrics:\nMean Absolute Error: ", mae_test_unpruned)
print("Mean Squared Error: ", mse_test_unpruned)
print("Root Mean Squared Error: ", rmse_test_unpruned)
print(f"R-squared: {r2_test_unpruned}")
```

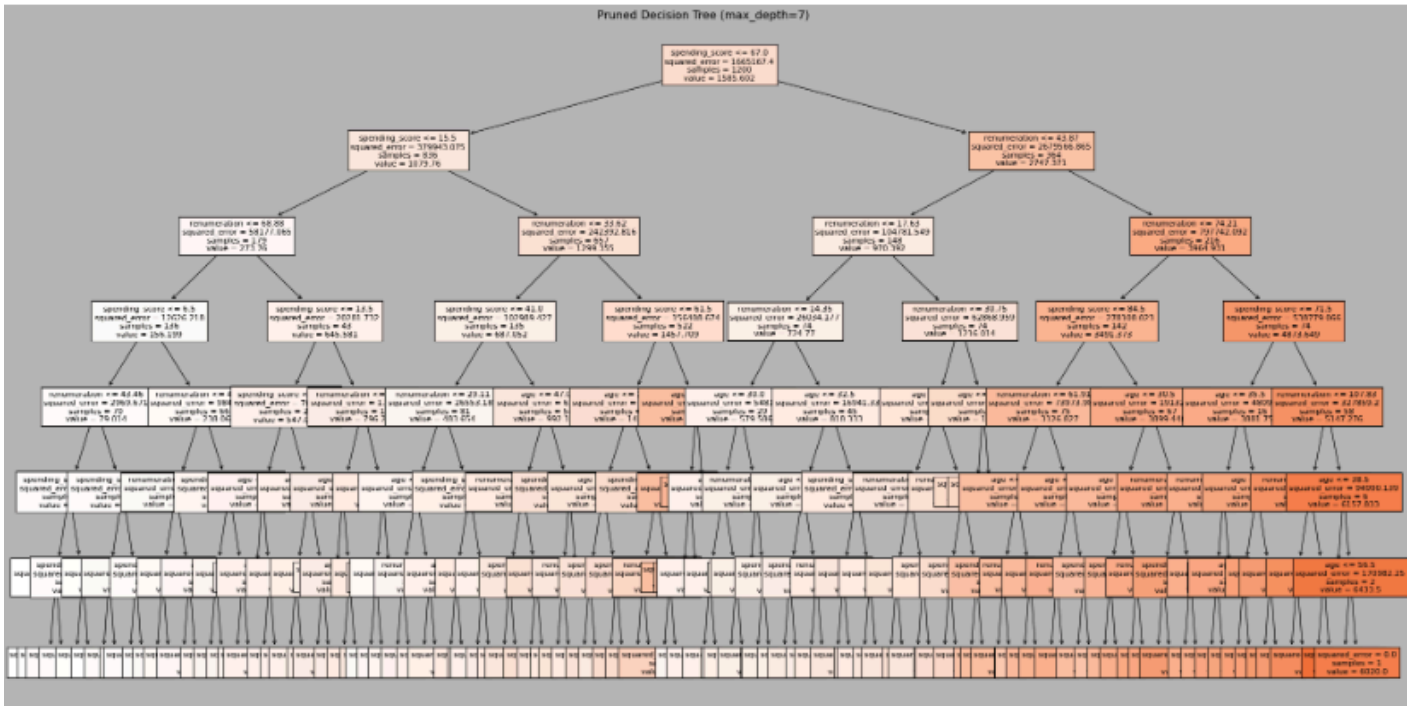
```
Test Metrics:
Mean Absolute Error:  23.96
Mean Squared Error:  5442.09
Root Mean Squared Error:  73.7705225682996
R-squared: 0.9969223368264497
```

[89]:
```
# Store the metrics in a DataFrame for comparison
comparison_metrics = pd.DataFrame({
    'Model': ['Unpruned Decision Tree', 'Pruned_3 Decision Tree', 'Pruned_5 Decision Tree','Pruned_7 Decision Tree','Pruned_9 Decision Tree
    'Mean Squared Error': [mse_test_unpruned, mse_test_pruned_3, mse_test_pruned_5, mse_test_pruned_7, mse_test_pruned_9],
    'Root Mean Squared Error': [rmse_test_unpruned, rmse_test_pruned_3, rmse_test_pruned_5, rmse_test_pruned_7, rmse_test_pruned_9],
    'Mean Absolute Error': [mae_test_unpruned, mae_test_pruned_3, mae_test_pruned_5, mae_test_pruned_7, mae_test_pruned_9],
    'R-squared': [r2_test_unpruned, r2_test_pruned_3, r2_test_pruned_5, r2_test_pruned_7, r2_test_pruned_9]
})

# Display the comparison
print("\nComparison of Decision Tree Evaluation Metrics:\n")
print(comparison_metrics)
```

```
Comparison of Decision Tree Evaluation Metrics:

                  Model  Mean Squared Error  Root Mean Squared Error  \
0  Unpruned Decision Tree         5442.090000                73.770523
1  Pruned_3 Decision Tree       146819.571383               383.170421
2  Pruned_5 Decision Tree        75027.118780               273.910786
3  Pruned_7 Decision Tree        33921.684329               184.178404
4  Pruned_9 Decision Tree        20983.431899               144.856591

   Mean Absolute Error  R-squared
0            23.960000   0.996922
1           279.895332   0.916969
2           182.266856   0.957570
3           117.079906   0.980816
4            86.640614   0.988133
```

- Visualisation





- Insight

While the unpruned model performs better statistically, depth of 7 appears to be a good option for the following reasons:

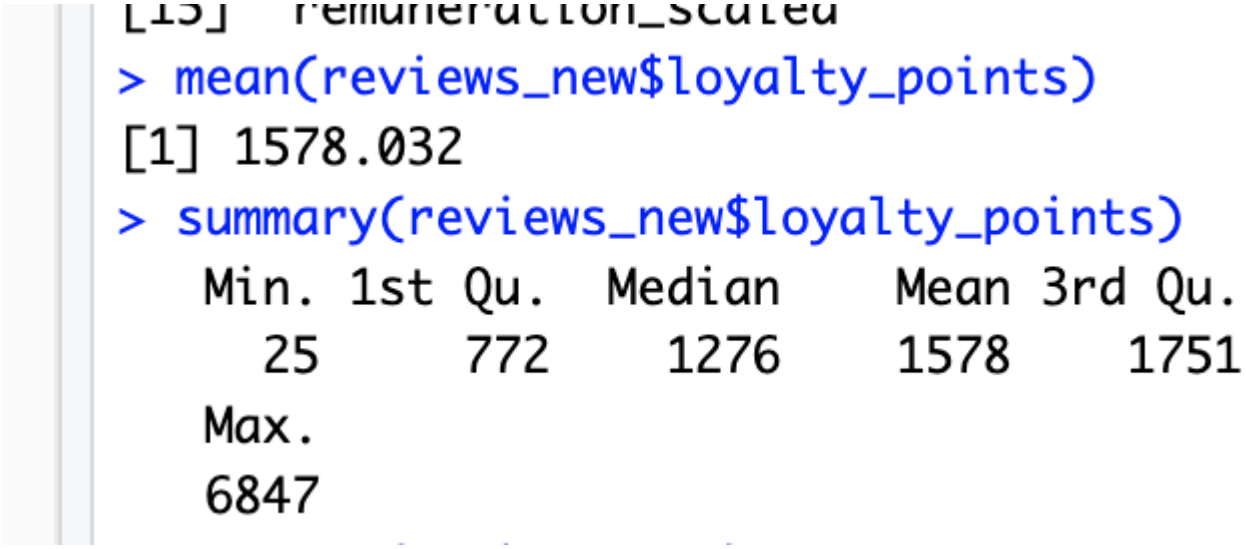
- the r2 of 0.9808 and mean squared error (MSE) of 33,921 are relatively good compared to the tree depth models of 3 and 5.
- while the unpruned model and the decision tree depth 9 result in higher r2 and lower mean square error (MSE) metrics they add more complexity, therefore harder to interpret, and also have a higher risk of overfitting on new data.

Interpreting the chosen decision tree

The tree splits first on 67 spending score <= 67.0, then on remuneration level <= 43.87, then on remuneration <= 74.21. This suggests the following:

- the tree first splits on spending score suggesting that this is the most critical factor in predicting loyalty point accumulation.
- A threshold of 67 indicates a key dividing line between different customer behaviours.
- Customers with a spending score above 67 and remuneration above 74.21 are the most valuable, likely accumulating the most loyalty points.
- The decision tree reveals that both spending score and remuneration are crucial in predicting loyalty point accumulation
- tree's structure emphasises the importance of segmentation (focusing on spending score and remuneration) in understanding customer behaviour
- Doesn't give the complete picture about customer behaviour needed for a robust marketing strategy.

i) Multiple linear Regression - method

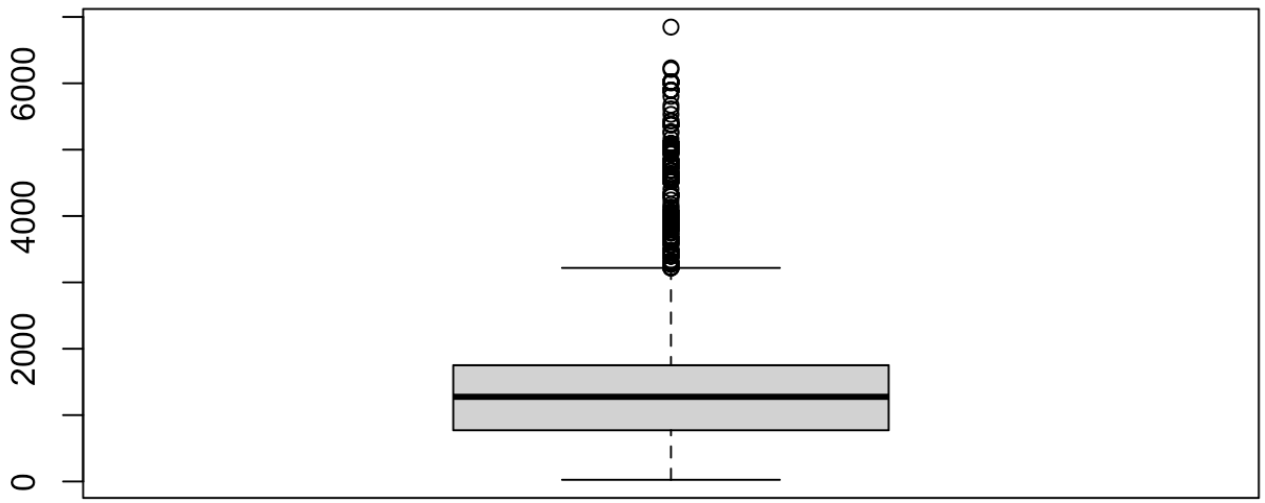


```
> # range = Max - min
> max(reviews_new$loyalty_points) - min(reviews_new$loyalty_points)
[1] 6822
> # calculate IQR
> IQR(reviews_new$loyalty_points)
[1] 979.25
> # Measures of dispersion
> var(reviews_new$loyalty_points)
[1] 1646704
> # return the standard deviation
> sd(reviews_new$loyalty_points)
[1] 1283.24
>
```

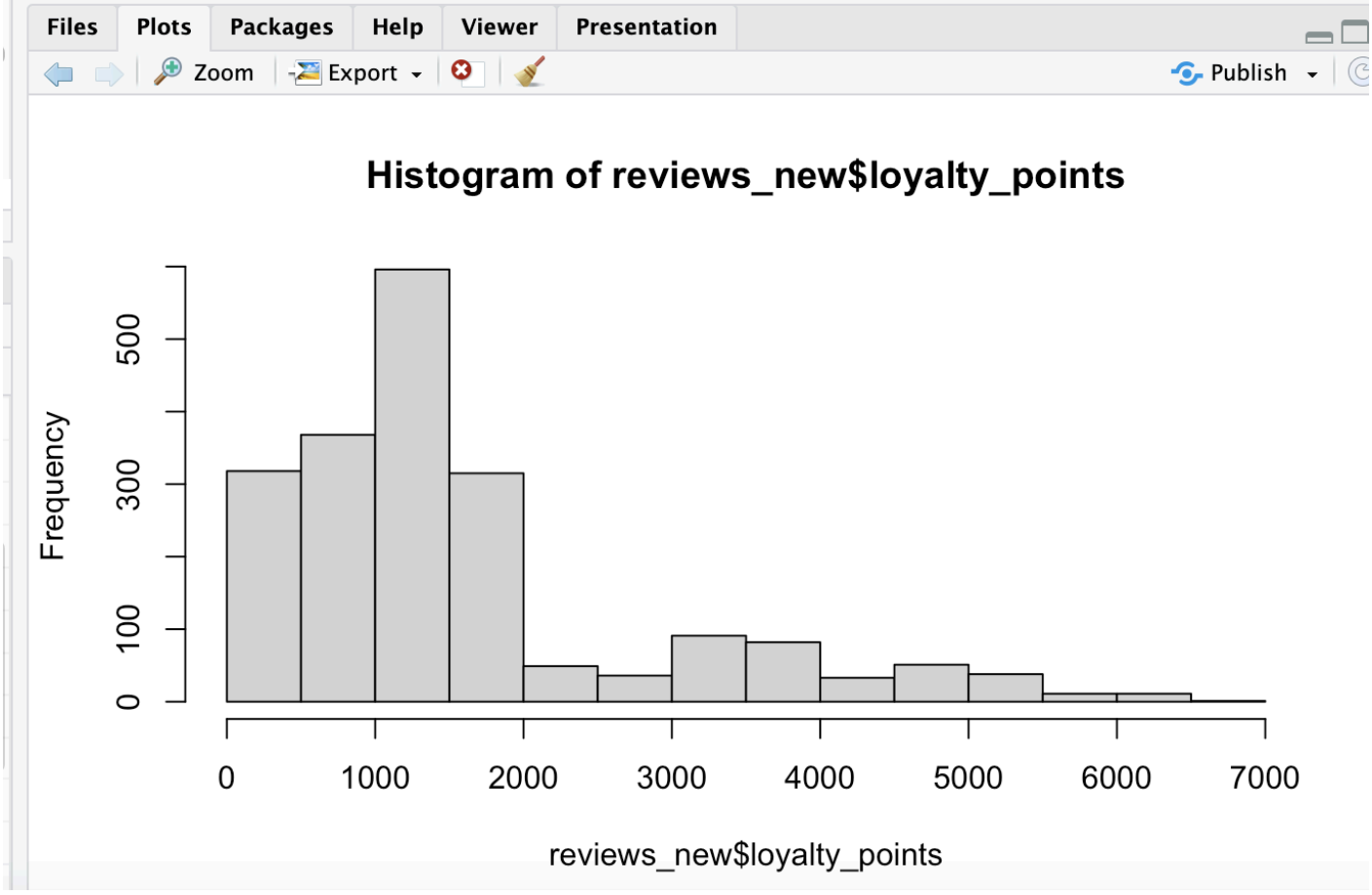
```
> # Distribution of the data
> # specify box plot function
> boxplot(reviews_new$loyalty_points)
>
```

Files | Plots | Packages | Help | Viewer | Presentation

Zoom | Export | Publish

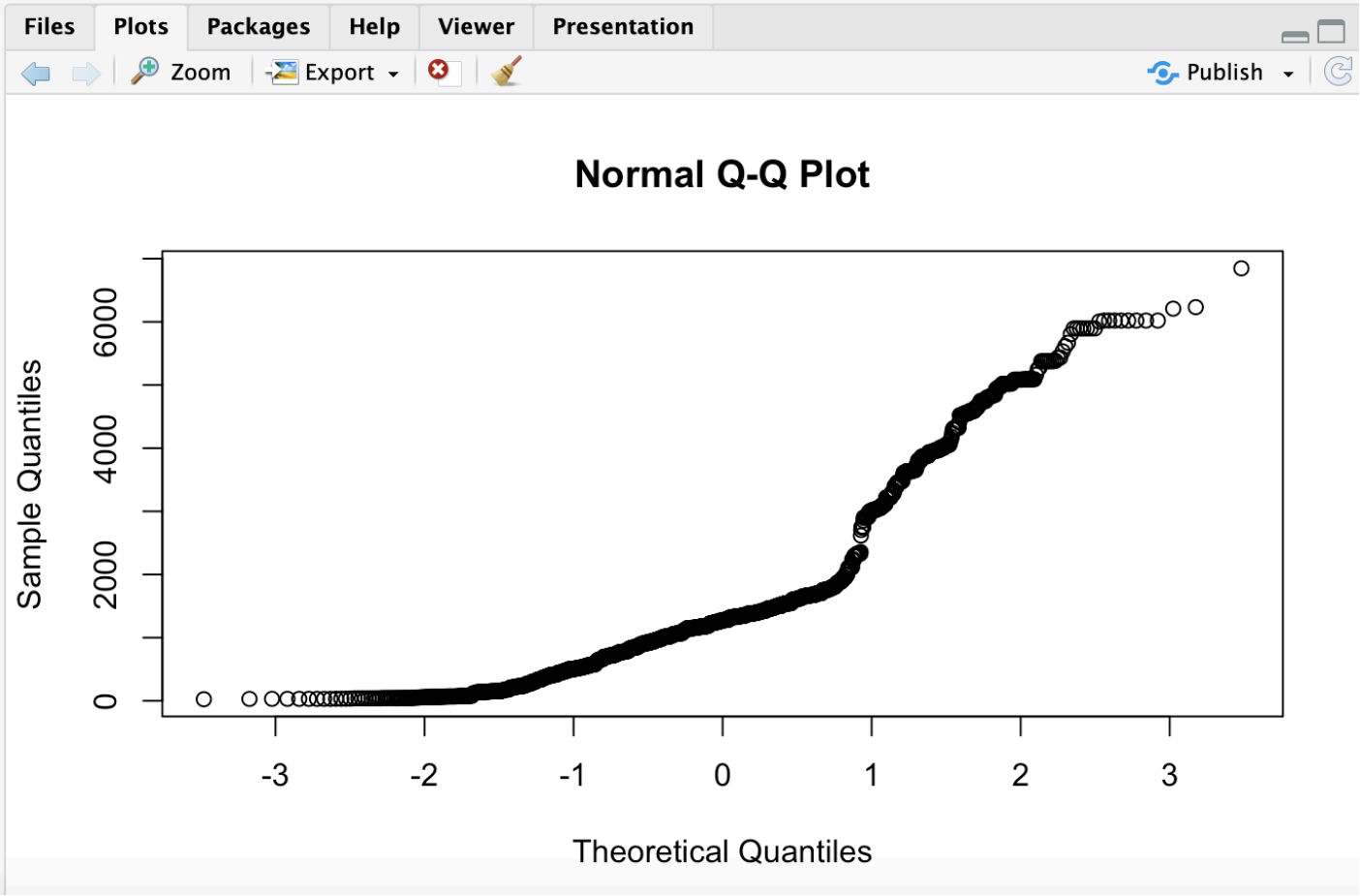


```
> boxplot(reviews_new$loyalty_points)
> hist(reviews_new$loyalty_points)
>
```



Conducted a shapiro-wilk test

```
> # specify qqnorm function(draw a qqplot)
> qqnorm(reviews_new$loyalty_points)
> # Conduct a shapiro-wilk test
> shapiro.test(reviews_new$loyalty_points)

        Shapiro-Wilk normality test

data:  reviews_new$loyalty_points
W = 0.84307, p-value < 2.2e-16

>
```



Checked for skewness and kurtosis

```
> # specify the skewness and kurtosis functions
> skewness(reviews_new$loyalty_points)
[1] 1.463694
> kurtosis(reviews_new$loyalty_points)
[1] 4.70883
>
```

The skewness value of 1.46 indicates that the distribution of customer loyalty points is moderately right-skewed, meaning there are more low values with a tail of higher values. The kurtosis value of 4.71 suggests that the distribution has heavier tails and a sharper peak compared to a normal distribution, indicating potential outliers or extreme values.

Created different models.

```
# 4. Create a multiple linear regression model
# using selected numeric features

# Model a - loyalty points vs remuneration and spending score
modela = lm(loyalty_points~remuneration+spending_score, data=reviews_new)

# print the summary statistics
summary(modela)

# Add new variables.
modelb = lm(loyalty_points~remuneration+spending_score+age+product, data=reviews_new

# Change the model name.
summary(modelb)

# New model.
modelc = lm(loyalty_points~remuneration+spending_score+age, data=reviews_new)

summary(modelc)
```

Testing of the model

```
> # Model a - loyalty points vs remuneration and spending score
> modela = lm(loyalty_points~remuneration+spending_score, data=re
views_new)
> # print the summary statistics
> summary(modela)

Call:
lm(formula = loyalty_points ~ remuneration + spending_score,
    data = reviews_new)

Residuals:
     Min       1Q   Median       3Q      Max
-1646.02  -363.66    40.34   280.59  1999.95

Coefficients:
                Estimate Std. Error t value Pr(>|t|)
(Intercept)    -1700.3051    35.7396  -47.58   <2e-16 ***
remuneration      33.9795     0.5166   65.77   <2e-16 ***
spending_score    32.8927     0.4578   71.84   <2e-16 ***
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

Residual standard error: 534.1 on 1997 degrees of freedom
Multiple R-squared:  0.8269,    Adjusted R-squared:  0.8267
F-statistic:  4770 on 2 and 1997 DF,  p-value: < 2.2e-16
```

```
> summary(modelb)

Call:
lm(formula = loyalty_points ~ remuneration + spending_score +
    age + product, data = reviews_new)

Residuals:
    Min      1Q  Median      3Q     Max
-1823.5  -352.9     4.3   290.6  1890.3

Coefficients:
                Estimate Std. Error t value Pr(>|t|)
(Intercept)    -2.200e+03  5.311e+01 -41.430   <2e-16 ***
remuneration    3.406e+01  5.220e-01  65.242   <2e-16 ***
spending_score  3.418e+01  4.520e-01  75.620   <2e-16 ***
age             1.106e+01  8.690e-01  12.729   <2e-16 ***
product        -1.221e-03  3.834e-03  -0.319     0.75
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

Residual standard error: 513.9 on 1995 degrees of freedom
Multiple R-squared:  0.8399,    Adjusted R-squared:  0.8396
F-statistic:  2617 on 4 and 1995 DF,  p-value: < 2.2e-16
```

```
################################################################################
# Test the model

# Load the new data file (reviews_new.csv), and view its structure.
loyaltytest <- read.csv('reviews_new.csv', header=TRUE)

# View the data.
View(loyaltytest)

# Create a new object and specify the predict function.
predictTest = predict(modelc, newdata=loyaltytest,
                      interval='confidence')
# Print the object.
predictTest


# Plot residuals
plot(modelc)
```

```

# QQ-plot for normality check
qqnorm(residuals(modelc))
qqline(residuals(modelc))

```

Revised model only using spending, remuneration and age

```
> # New model:
> modelc = lm(loyalty_points~remuneration+spending_score+age, dat
a=reviews_new)
> summary(modelc)

Call:
lm(formula = loyalty_points ~ remuneration + spending_score +
    age, data = reviews_new)

Residuals:
     Min       1Q   Median       3Q      Max
-1819.11  -350.84     4.61   291.00  1894.62

Coefficients:
                Estimate Std. Error t value Pr(>|t|)
(Intercept)    -2203.0598    52.3609  -42.08   <2e-16 ***
remuneration      34.0084     0.4970   68.43   <2e-16 ***
spending_score    34.1832     0.4519   75.64   <2e-16 ***
age               11.0607     0.8688   12.73   <2e-16 ***
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

Residual standard error: 513.8 on 1996 degrees of freedom
Multiple R-squared:  0.8399,    Adjusted R-squared:  0.8397
F-statistic:  3491 on 3 and 1996 DF,  p-value: < 2.2e-16
```

The skewness and kurtosis results indicate that the distribution of loyalty points is not normally distributed, with a right skew and heavier tails, meaning extreme values are present. This non-normality can affect the residuals of the regression model, although the model results show a strong fit ($R^2 = 0.8399$), suggesting that remuneration, spending score, and age are still strong predictors of loyalty points despite the non-normal distribution of the data.

j) K means clustering - method

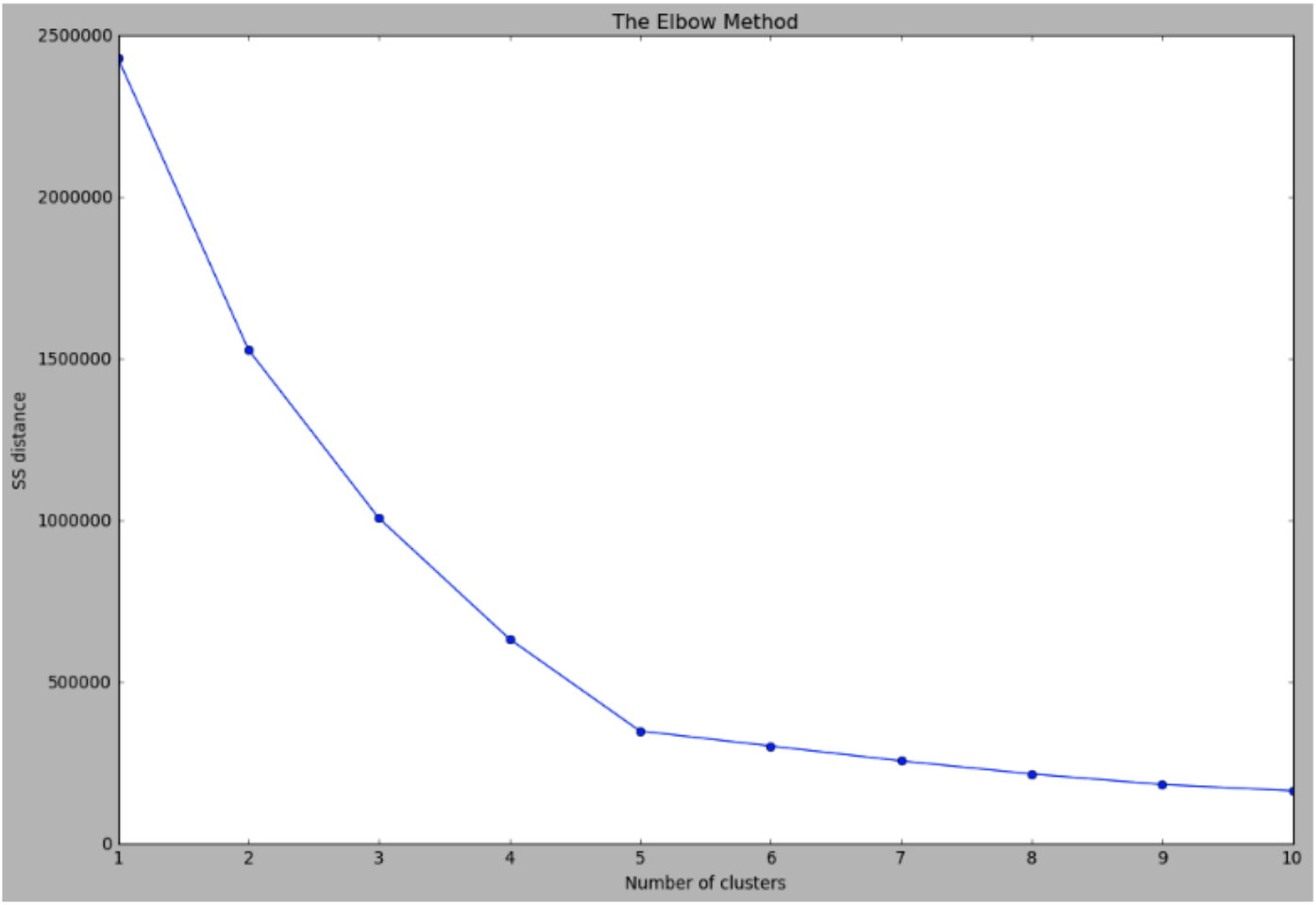
Using the elbow and silhouette methods from sklearn.metrics to identify the optimal cluster number. I evaluated the k means model at different values of k (3, 4, 5 & 6) and compared the k means plots against the value.counts() for each model.

[93]:
```
# Determine the number of clusters: Silhouette method.
from sklearn.metrics import silhouette_score

# Find the range of clusters to be used using silhouette method.
sil = []
kmax = 10

for k in range(2, kmax+1):
    kmeans_s = KMeans(n_clusters=k).fit(x)
    labels = kmeans_s.labels_
    sil.append(silhouette_score(x,
                                labels,
                                metric='euclidean'))

# Plot the silhouette method.
plt.plot(range(2, kmax+1),
         sil,
         marker='o')

# Insert labels and title.
plt.title("The Silhouette Method")
plt.xlabel("Number of clusters")
plt.ylabel("Sil")

plt.show()
```

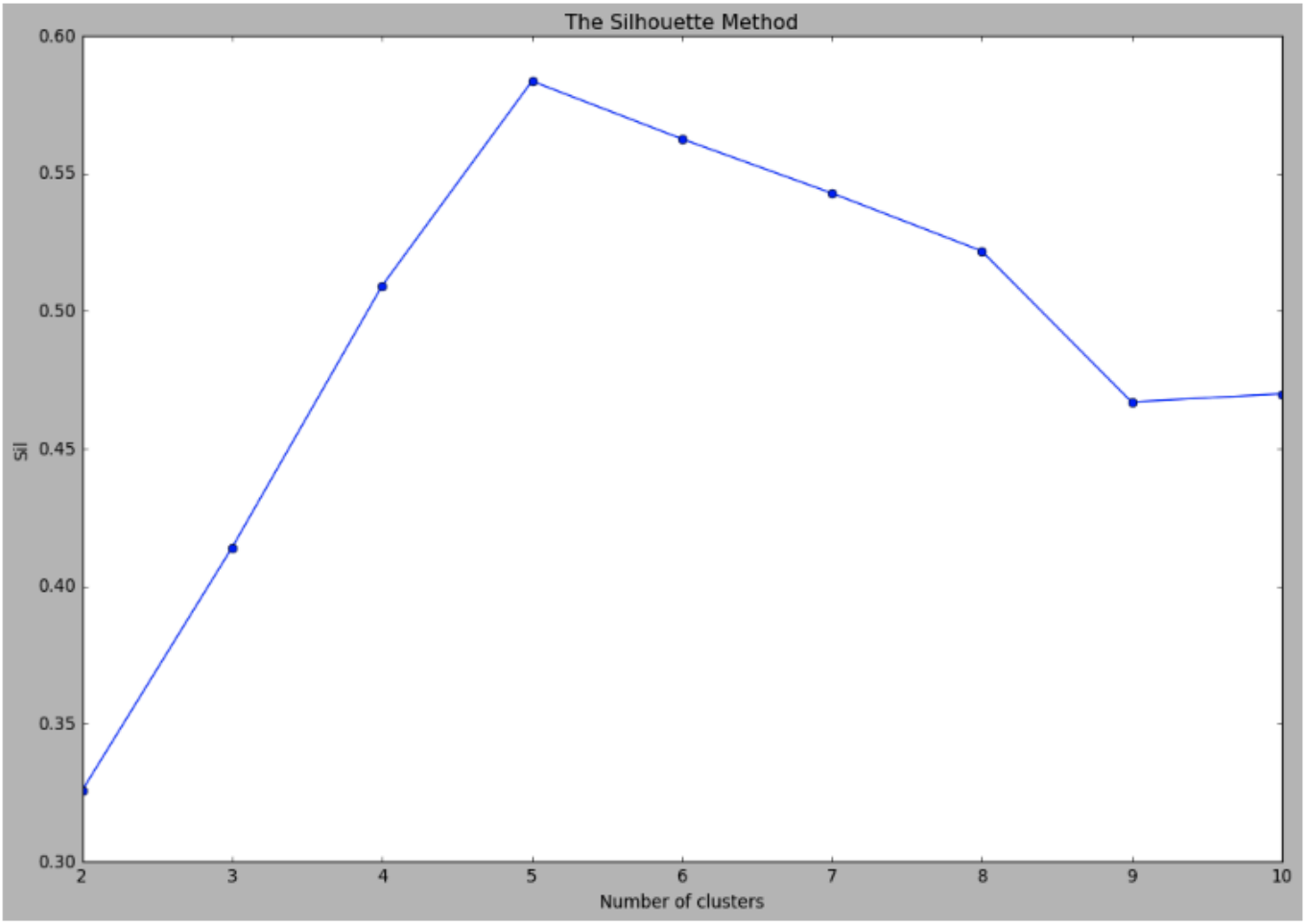

iii). Elbow and silhoutte methods

[91]:
```
# Determine the number of clusters: Elbow method.

# Elbow chart for us to decide on the number of optimal clusters.
ss = []
for i in range(1, 11):
    kmeans = KMeans(n_clusters=i,
                    init='k-means++',
                    max_iter=500,
                    n_init=10,
                    random_state=42)
    kmeans.fit(x)
    ss.append(kmeans.inertia_)

# Plot the elbow method.
plt.plot(range(1, 11),
         ss,
         marker='o')

# Insert labels and title.
plt.title("The Elbow Method")
plt.xlabel("Number of clusters")
plt.ylabel("SS distance")

plt.show()
```

Pescod_Henry_DA301_Assignment_Report.pdf

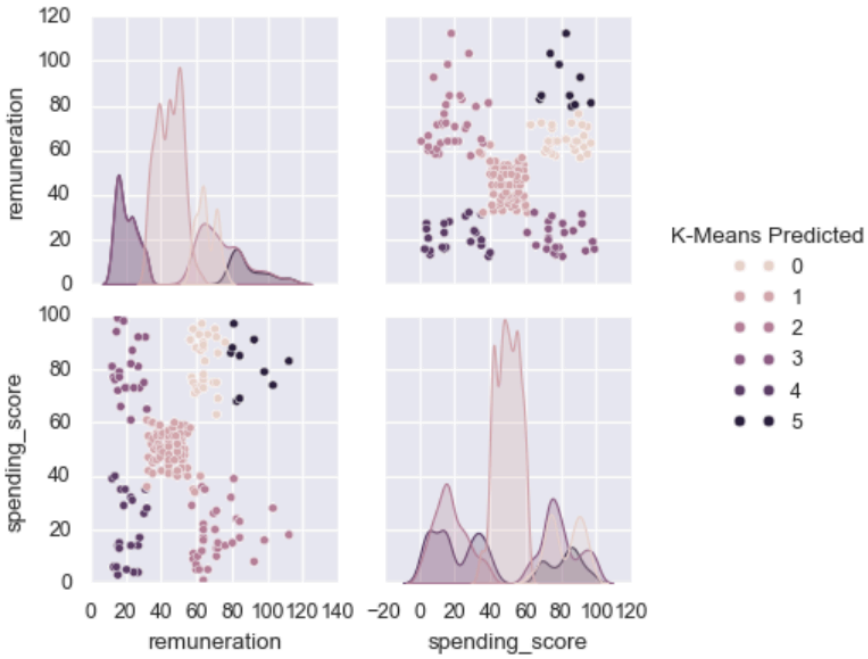


Pescod_Henry_DA301_Assignment_Report.pdf



[99]:
```
# Use Six clusters.
kmeans = KMeans(n_clusters = 6,
                max_iter = 15000,
                init='k-means++',
                random_state=42).fit(x)

clusters = kmeans.labels_

x['K-Means Predicted'] = clusters

# Plot the predicted.
sns.pairplot(x,
             hue='K-Means Predicted',
             diag_kind= 'kde')
```

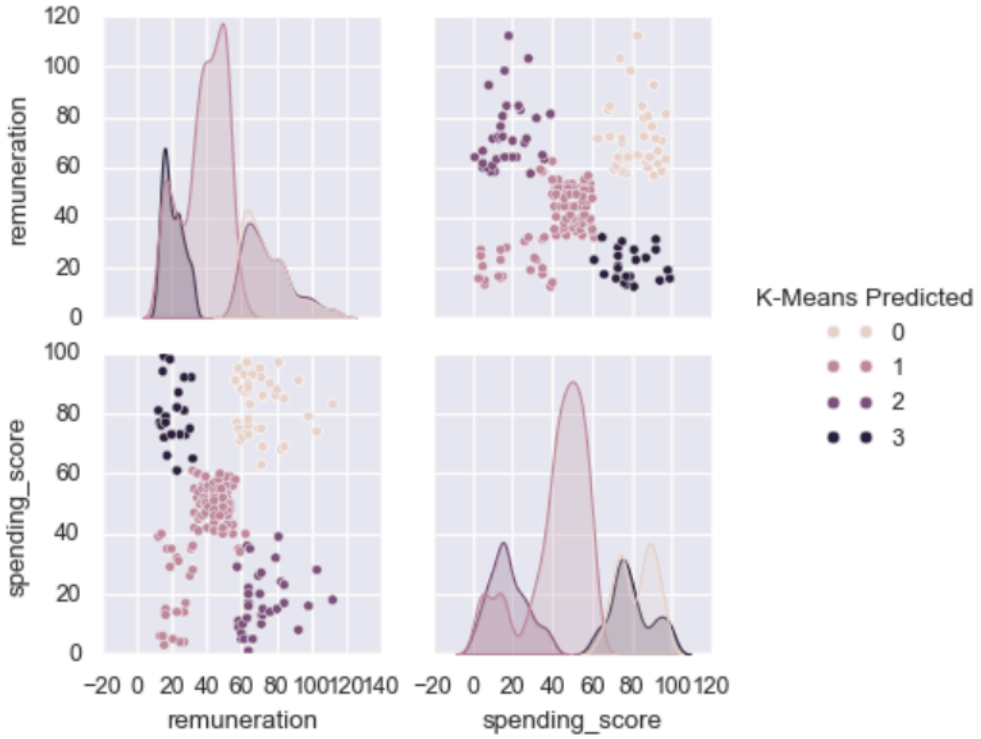
[99]:
```
<seaborn.axisgrid.PairGrid at 0x13ee17320>
```



-

[103]:
```
# Use four clusters.
kmeans = KMeans(n_clusters = 4,
                max_iter = 15000,
                init='k-means++',
                random_state=42).fit(x)

clusters = kmeans.labels_

x['K-Means Predicted'] = clusters

# Plot the predicted.
sns.pairplot(x,
             hue='K-Means Predicted',
             diag_kind= 'kde')
```

[103]:
```
<seaborn.axisgrid.PairGrid at 0x157b0d4c0>
```



vi). Plot and interpret the clusters

17]:
```
# Visualising the clusters.

# View the DataFrame.
print(x.head())
```

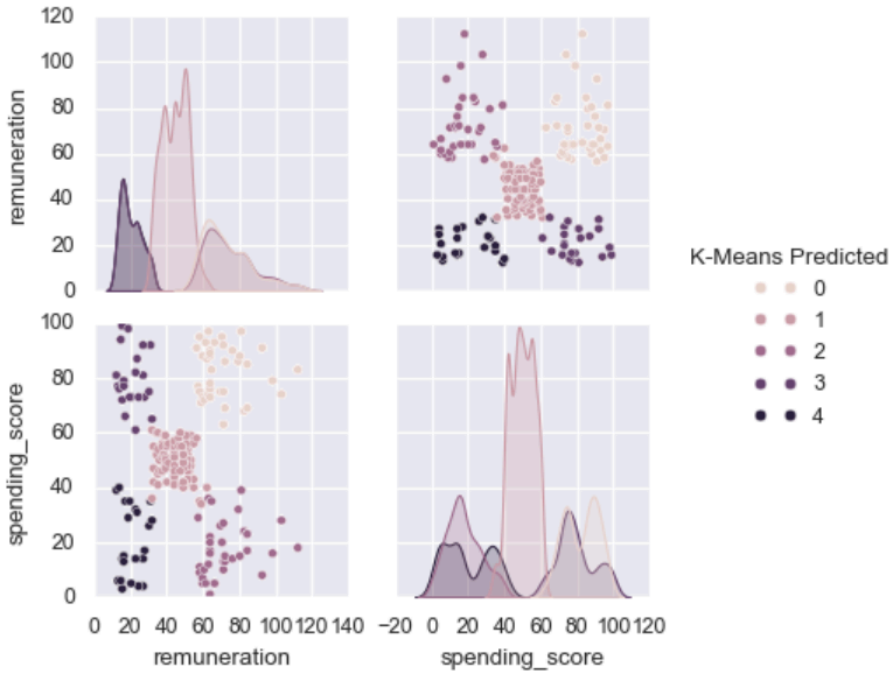
```
   remuneration  spending_score  K-Means Predicted
0         12.30              39                  4
1         12.30              81                  3
2         13.12               6                  4
3         13.12              77                  3
4         13.94              40                  4
```

19]:
```
# Visualising the clusters.
# Set plot size.
sns.set(rc = {'figure.figsize':(12, 8)})

# Create a scatterplot.
sns.scatterplot(x='remuneration' ,
                y ='spending_score',
                data=x,
                hue='K-Means Predicted',
                palette=['red', 'green', 'blue', 'purple', 'orange'])
```

iv). Evaluate k-means model at different values of *k*

[95]:
```
# Use five clusters.
kmeans = KMeans(n_clusters = 5,
                max_iter = 15000,
                init='k-means++',
                random_state=42).fit(x)

clusters = kmeans.labels_

x['K-Means Predicted'] = clusters

# Plot the predicted.
sns.pairplot(x,
             hue='K-Means Predicted',
             diag_kind= 'kde')
```

[95]:
```
<seaborn.axisgrid.PairGrid at 0x13df83920>
```



k) EDA by grouping of clusters - method

I add these cluster groupings to the dataset and exported to CSV to be able to analyse further in R.

263]:
```
# Check the number of observations per predicted class.
x['K-Means Predicted'].value_counts()
```

263]:
```
K-Means Predicted
1    774
0    356
2    330
4    271
3    269
Name: count, dtype: int64
```

264]:
```
# Assign Cluster Labels to the data set:

reviews_new['cluster'] = clusters
```

265]:
```
# Save the Updated DataFrame to a CSV File

# Save the updated DataFrame with the new 'cluster' column to a CSV file
reviews_new.to_csv('reviews_new_5_clusters.csv', index=False)
```

I established names from these 5 clusters based on each clusters attributes that could help with customer segmentation

- Cluster 0 - high remuneration and high spending score, “high spenders”
- Cluster 1 - mid remuneration, mid spending score - “moderate earners”
- Cluster 2 - high remuneration and low spending score - “low spenders, high income”
- Cluster 3 - high spending score, low remuneration - “value seekers”
- Cluster 4 - low spending score, low remuneration - “budget conscious”

I generated combined bar plots for loyalty points, spending scores, and remuneration, faceting by Education, gender and cluster name.

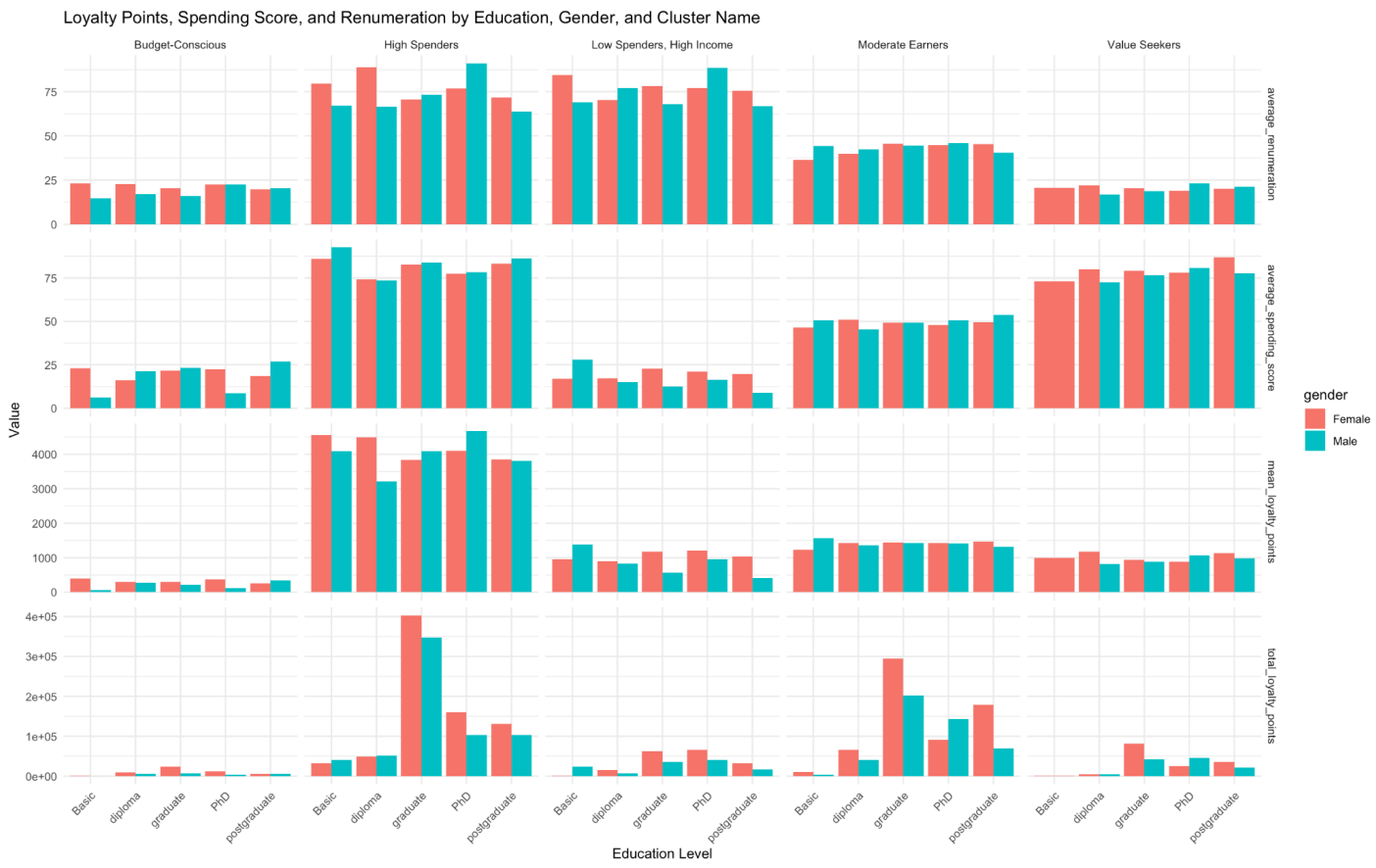
```
# Step 5: Create age bins and summarize by age group and cluster
# Create age bins using the `cut()` function
reviews_new <- reviews_new %>%
  mutate(age_group = cut(age, breaks = c(0, 25, 35, 45, 60, Inf), right = FALSE,
                         labels = c("<=24", "25-34", "35-44", "45-59", "60+")))

# Summarize mean loyalty points, average spending score, average remuneration by age, gender, and cluster
demographic_summary_age <- reviews_new %>%
  group_by(age_group, gender, cluster) %>%
  summarize(
    mean_loyalty_points = mean(loyalty_points, na.rm = TRUE),
    average_spending_score = mean(spending_score, na.rm = TRUE),
    average_remuneration = mean(remuneration, na.rm = TRUE),
    .groups = "drop"
  )
```

```r
# Summarize mean loyalty points, average spending score, average remuneration by age, gender, and cluster
demographic_summary_age <- reviews_new %>%
  group_by(age_group, gender, cluster) %>%
  summarize(
    mean_loyalty_points = mean(loyalty_points, na.rm = TRUE),
    average_spending_score = mean(spending_score, na.rm = TRUE),
    average_remuneration = mean(remuneration, na.rm = TRUE),
    .groups = "drop"
  )

# Add cluster names to the summary
demographic_summary_age <- demographic_summary_age %>%
  mutate(cluster_name = case_when(
    cluster == 0 ~ "High Spenders",
    cluster == 1 ~ "Moderate Earners",
    cluster == 2 ~ "Low Spenders, High Income",
    cluster == 3 ~ "Value Seekers",
    cluster == 4 ~ "Budget-Conscious",
    TRUE ~ NA_character_
  ))
```

```r
# Now pivot the data to long format including cluster_name
demographic_summary_age_long <- demographic_summary_age %>%
  pivot_longer(cols = c(mean_loyalty_points, average_spending_score, average_remuneration),
               names_to = "metric",
               values_to = "value")

# Create the combined plot for age group and cluster name
ggplot(demographic_summary_age_long, aes(x = age_group, y = value, fill = gender)) +
  geom_bar(stat = "identity", position = "dodge") +
  labs(title = "Mean Loyalty Points, Spending Score, and Remuneration by Age Group, Gender and Cluster Name",
       x = "Age Group",
       y = "Value") +
  theme_minimal() +
  theme(axis.text.x = element_text(angle = 45, hjust = 1)) +
  facet_grid(metric ~ cluster_name, scales = "free_y")

View(demographic_summary_age_long)
```

```r
# Add cluster names to the summary
cluster_summary <- cluster_summary %>%
  mutate(cluster_name = case_when(
    cluster == 0 ~ "High Spenders",
    cluster == 1 ~ "Moderate Earners",
    cluster == 2 ~ "Low Spenders, High Income",
    cluster == 3 ~ "Value Seekers",
    cluster == 4 ~ "Budget-Conscious",
    TRUE ~ NA_character_
  ))

# View the summary data
print(cluster_summary)
View(cluster_summary)
```

```r
# Step 6: Create a summary matrix table showing the difference metrics of each cluster
cluster_summary <- reviews_new %>%
  group_by(cluster) %>%
  summarize(
    avg_spending_score = round(mean(spending_score, na.rm = TRUE), 2),
    avg_remuneration = round(mean(remuneration, na.rm = TRUE), 2),
    number_of_customers = n(),
    total_loyalty_points = sum(loyalty_points, na.rm = TRUE),
    avg_loyalty_points = round(mean(loyalty_points, na.rm = TRUE), 0),
    avg_age = round(mean(age, na.rm = TRUE), 0),
    .groups = "drop"
  )
```



Initial observations of the segments

- Cluster 0 - "High Spenders" combination of high spending, high remuneration and high loyalty points indicates that these customers are valuable and engaged
- Cluster 1 - "Moderate earners" make up the largest portion of the sample data set (774 customers). Should this pattern repeat on a larger data set, Turtle Games may want to consider positioning certain products that appeal specifically to their profile when conducting product sales
- Cluster 2 - "Low Spenders, High Earners" - low spending score but high average remuneration suggests potential for growth if Turtle games marketing strategies can encourage spending among this segment.
- Cluster 3 - "Value Seekers" high spenders but lower remuneration, could suggest these customers seem to seek value, possibly making purchases during promotions or discounts.
- Cluster 4 - "Budget Conscious" - Low spending score and remuneration indicating they might be more price-sensitive.

Following exploratory data analysis in R based on these clusters, I was able to spot some interesting patterns in the data.

Firstly the “high spenders” scored well on mean loyalty points suggesting they are very engaged and valuable customers

Secondly the “low spend, high income” group have a low amount of loyalty points, but do present a growth opportunity in terms of spending

Thirdly - the “value seekers” spend highly but accumulate few loyalty points, possibly making purchases during promotions or discounts which maybe don’t generate loyalty points.

I) NLP and sentiment analysis by clusters - method

I used the nltk library to process text reviews and summaries.

ii. Prepare the data for NLP

i) Change to lower case and join the elements in each of the columns respectively (review and summary)

```
.32]: # Review: Change all to lower case and join with a space.
      df5['review'] = df5['review'].apply(lambda x: " ".join(x.lower() for x in x.split()))

      df5['review'].head()
```

```
.32]: 0    when it comes to a dm's screen, the space on t...
      1    an open letter to galeforce9*: your unpainted ...
      2    nice art, nice printing. why two panels are fi...
      3    amazing buy! bought it as a gift for our new d...
      4    as my review of gf9's previous screens these w...
      Name: review, dtype: object
```

```
.32]: # Summary: Change all to lower case and join with a space.
      df5['summary'] = df5['summary'].apply(lambda x: " ".join(x.lower() for x in x.split()))

      df5['summary'].head()
```

```
.32]: 0    the fact that 50% of this space is wasted on a...
      1    another worthless dungeon master's screen from...
      2                      pretty, but also pretty useless
      3                                           five stars
      4                                           money trap
      Name: summary, dtype: object
```

ii) Replace punctuation in each of the columns respectively (review and summary)

134]:
```
# Replace all the punctuations in review column.
df5['review'] = df5['review'].str.replace('[^\w\s]','', regex=True)

# Preview the result.
df5['review'].head()
```

134]:
```
0    when it comes to a dms screen the space on the...
1    an open letter to galeforce9 your unpainted mi...
2    nice art nice printing why two panels are fill...
3    amazing buy bought it as a gift for our new dm...
4    as my review of gf9s previous screens these we...
Name: review, dtype: object
```

136]:
```
# Replace all the puncuations in summary column.
df5['summary'] = df5['summary'].str.replace('[^\w\s]','', regex=True)

# Preview the result.
df5['summary'].head()
```

136]:
```
0    The fact that 50 of this space is wasted on ar...
1    Another worthless Dungeon Masters screen from ...
2                       pretty but also pretty useless
3                                           Five Stars
4                                           Money trap
Name: summary, dtype: object
```

iii) Drop duplicates in both columns

138]:
```
# Check the number of duplicate values in the review & summary columns.
print(df5[['review','summary']].duplicated().sum())

# Drop duplicates in both columns.
df6 = df5.drop_duplicates(subset=['review','summary'])

# View DataFrame.
df6.reset_index(inplace=True)
print(df6.head())
```

```
39
   index                                             review  \
0      0  when it comes to a dms screen the space on the...
1      1  an open letter to galeforce9 your unpainted mi...
2      2  nice art nice printing why two panels are fill...
3      3  amazing buy bought it as a gift for our new dm...
4      4  as my review of gf9s previous screens these we...

                                        summary
0  The fact that 50 of this space is wasted on ar...
1  Another worthless Dungeon Masters screen from ...
2                     pretty but also pretty useless
3                                       Five Stars
4                                       Money trap
```

140]:
```
# Check the number of duplicate values in the review & summary columns.
print(df6[['review','summary']].duplicated().sum())
```

```
0
```

iii). Tokenise and create wordclouds

[12]:
```
# Create new DataFrame (copy DataFrame).
df7=df6.copy()

# View DataFrame.
df7.head()
```

[12]:

	index	review	summary
0	0	when it comes to a dms screen the space on the...	The fact that 50 of this space is wasted on ar...
1	1	an open letter to galeforce9 your unpainted mi...	Another worthless Dungeon Masters screen from ...
2	2	nice art nice printing why two panels are fill...	pretty but also pretty useless
3	3	amazing buy bought it as a gift for our new dm...	Five Stars
4	4	as my review of gf9s previous screens these we...	Money trap

[14]:
```
# Apply tokenisation to both columns.
df7['review_tokens'] = df7['review'].apply(word_tokenize)
df7['summary_tokens'] = df7['summary'].apply(word_tokenize)

# View DataFrame.
print(df7['review_tokens'].head())
print(df7['summary_tokens'].head())
```

```
0    [when, it, comes, to, a, dms, screen, the, spa...
1    [an, open, letter, to, galeforce9, your, unpai...
2    [nice, art, nice, printing, why, two, panels, ...
3    [amazing, buy, bought, it, as, a, gift, for, o...
4    [as, my, review, of, gf9s, previous, screens, ...
Name: review_tokens, dtype: object
0    [The, fact, that, 50, of, this, space, is, was...
1    [Another, worthless, Dungeon, Masters, screen,...
2                 [pretty, but, also, pretty, useless]
3                                       [Five, Stars]
4                                       [Money, trap]
Name: summary_tokens, dtype: object
```

146]:
```
# Review: Create a word cloud.
# Define an empty list of tokens
all_review_tokens = []
for i in range(df7.shape[0]):
     # Add each comment.
       all_review_tokens = all_review_tokens + df7['review_tokens'][i]

# Convert the list of tokens back into a single string (required by WordCloud)
all_review_tokens_string = ' '.join(all_review_tokens)
```

148]:
```
# Import along with matplotlib and seaborn for visualisation.

sns.set(color_codes=True)

# Create a WordCloud object.
word_cloud = WordCloud(width = 1600, height = 900,
                background_color ='white',
                colormap = 'plasma',
                stopwords = 'none',
                min_font_size = 10).generate(all_review_tokens_string)
```

149]:
```
# Review: Plot the WordCloud image.
plt.figure(figsize = (16, 9), facecolor = None)
plt.imshow(word_cloud)
plt.axis('off')
plt.tight_layout(pad = 0)
plt.show()
```

Cluster	Sentiment Polarity	Cluster Name
0	0.213211	High Spenders
1	0.197935	Moderate Earners
2	0.252067	Low Spenders, High Income
3	0.192042	Value Seekers
4	0.212877	Budget-Conscious

- Visualisation

Word cloud of highest frequency words for reviews and summaries (without stop words)

ii) Remove alphanumeric characters and stopwords

[163]:
```
# Delete all the alpanum.
# Filter out tokens that are neither alphabets nor numbers (to eliminate punctuation marks, etc.).
tokens_review = [word for word in all_review_tokens if word.isalnum()]
tokens_summary = [word for word in all_summary_tokens if word.isalnum()]
```

[167]:
```
# Review: Remove all the stopwords

# Download the stopword list.
nltk.download ('stopwords')
from nltk.corpus import stopwords

# Create a set of English stopwords.
english_stopwords = set(stopwords.words('english'))

# Create a filtered list of tokens without stopwords.
tokens2_review = [x for x in tokens_review if x.lower() not in english_stopwords]

# Define an empty string variable.
tokens2_review_string = ''

for value in tokens2_review:
    # Add each filtered token word to the string.
    tokens2_review_string = tokens2_review_string + value + ' '
```

i) Create frequency distribution

[59]:
```
# Determine the frequency distribution.

# Calculate the frequency distribution.
fdist_review = FreqDist(all_review_tokens)
fdist_summary = FreqDist(all_summary_tokens)

# Preview data.
fdist_review
```

[59]:
```
FreqDist({'the': 5451, 'and': 3233, 'to': 3162, 'a': 3160, 'of': 2488, 'i': 2090, 'it': 2083, 'is': 1782, 'this': 1776, 'g
e': 1671, ...})
```

[61]:
```
# Preview data.
fdist_summary
```

[61]:
```
FreqDist({'Stars': 423, 'Five': 340, 'game': 262, 'for': 224, 'the': 213, 'Great': 202, 'to': 190, 'a': 176, 'and': 168, '
137, ...})
```

```
[nltk_data]   Package stopwords is already up-to-date!
```

[169]:
```
# Summary: Remove all the stopwords

# Create a filtered list of tokens without stopwords.
tokens2_summary = [x for x in tokens_summary if x.lower() not in english_stopwords]

# Define an empty string variable.
tokens2_summary_string = ''

for value in tokens2_summary:
    # Add each filtered token word to the string.
    tokens2_summary_string = tokens2_summary_string + value + ' '
```

iii) Create wordcloud without stopwords

[171]:
```
# Review: Create a wordcloud without stop words.

wordcloud = WordCloud(width = 1600, height = 900,
                background_color ='white',
                colormap='plasma',
                min_font_size = 10).generate(tokens2_review_string)
```

[172]:
```
# Review: Plot the wordcloud image.

plt.figure(figsize = (16, 9), facecolor = None)
plt.imshow(wordcloud)
plt.axis('off')
plt.tight_layout(pad = 0)
plt.show()
```

Word cloud of highest frequency words for summaries (without stop words)

iv) Identify 15 most common words and polarity (without stopwords)

179]:
```
# Calculate the frequency distribution.
fdist_review2 = FreqDist(tokens2_review)
fdist_summary2 = FreqDist(tokens2_summary)

# Preview data.
fdist_review2
```

179]:
```
FreqDist({'game': 1671, 'great': 580, 'fun': 552, 'one': 530, 'play': 502, 'like': 414, 'love': 323, 'really': 319, 'g
9, 'cards': 301, ...})
```

181]:
```
# Preview data.
fdist_summary2
```

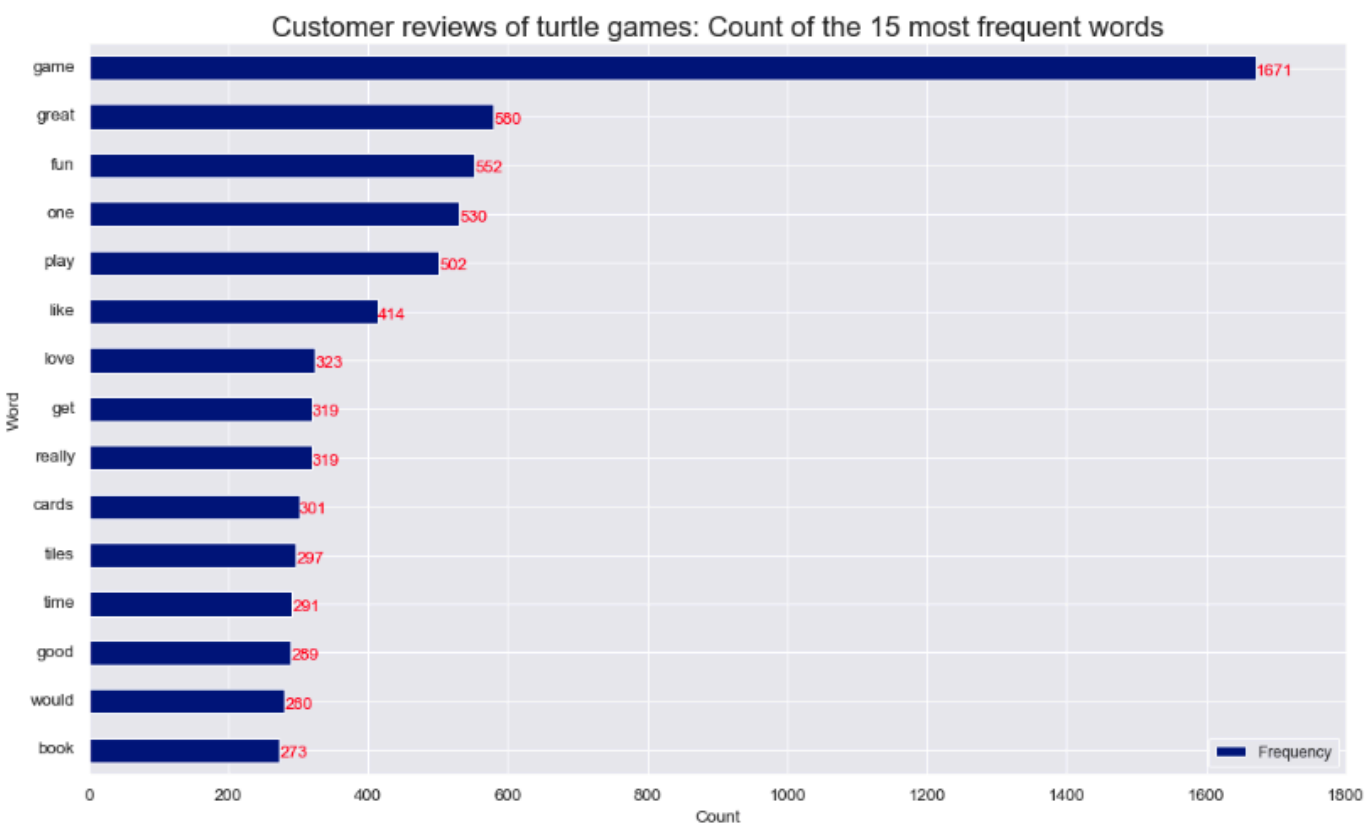
181]:
```
FreqDist({'Stars': 423, 'Five': 340, 'game': 262, 'Great': 202, 'fun': 126, 'great': 90, 'Fun': 85, 'Four': 57, 'Game'
ood': 49, ...})
```

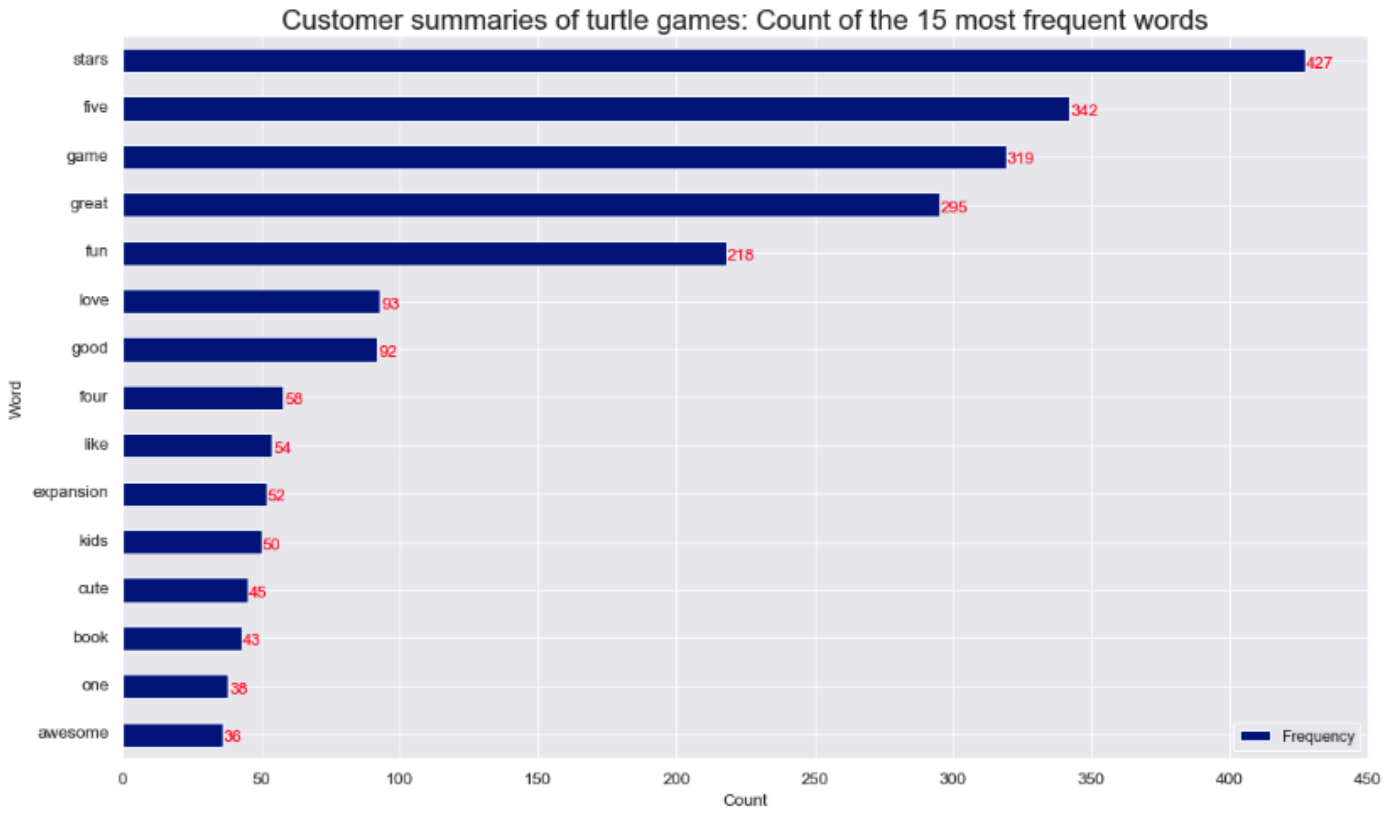
184]:
```
# Review: Determine the 15 most common words.

# Generate a DataFrame from Counter.
counts_review = pd.DataFrame(Counter(tokens2_review).most_common(15),
                             columns=['Word', 'Frequency']).set_index('Word')

# Preview data.
counts_review
```

ii) Remove alphanumeric characters and stopwords

3]:
```
# Delete all the alpanum.
# Filter out tokens that are neither alphabets nor numbers (to eliminate punctuation marks, etc.).
tokens_review = [word for word in all_review_tokens if word.isalnum()]
tokens_summary = [word for word in all_summary_tokens if word.isalnum()]
```

7]:
```
# Review: Remove all the stopwords

# Download the stopword list.
nltk.download ('stopwords')
from nltk.corpus import stopwords

# Create a set of English stopwords.
english_stopwords = set(stopwords.words('english'))

# Create a filtered list of tokens without stopwords.
tokens2_review = [x for x in tokens_review if x.lower() not in english_stopwords]

# Define an empty string variable.
tokens2_review_string = ''

for value in tokens2_review:
    # Add each filtered token word to the string.
    tokens2_review_string = tokens2_review_string + value + ' '
```





- Cluster 0: High Spenders. This group might include family-oriented customers, given the words "kids" and "love."
- Cluster 1: Moderate Earners. Frequent words like "tiles," "cards," and "boards" suggest this cluster might be composed of board game enthusiasts who enjoy traditional elements.
- Cluster 2: Low spenders, high income. The words "make," "kids," and "well" could suggest a focus on games that are well-designed and engaging for children.
- Cluster 3: Value seekers. Mention of "book" and "tiles" might indicate that this cluster prefers more strategic or intellectually engaging games.
- Cluster 4: Budget conscious. Presence of words like "kids," "time," and "set" might suggest this cluster includes family-oriented customers who value games that are easy to set up and provide fun for children.

Sentiment polarity

The sentiment polarity across the clusters is consistently positive, ranging from 0.19 to 0.25.

Cluster 0: High Spenders. Positive polarity indicates a favourable reception among these users

Cluster 1: Moderate Earners. Moderate polarity shows general satisfaction with these products.

Cluster 2: Low spenders, high income. This cluster has the highest sentiment polarity, indicating the most positive reviews.

Cluster 3: Value seekers. slightly lower sentiment polarity suggests that while they enjoy these games, they might also be more critical or have higher expectations.

Cluster 4: Budget conscious. Positive polarity indicates a favourable reception among these users

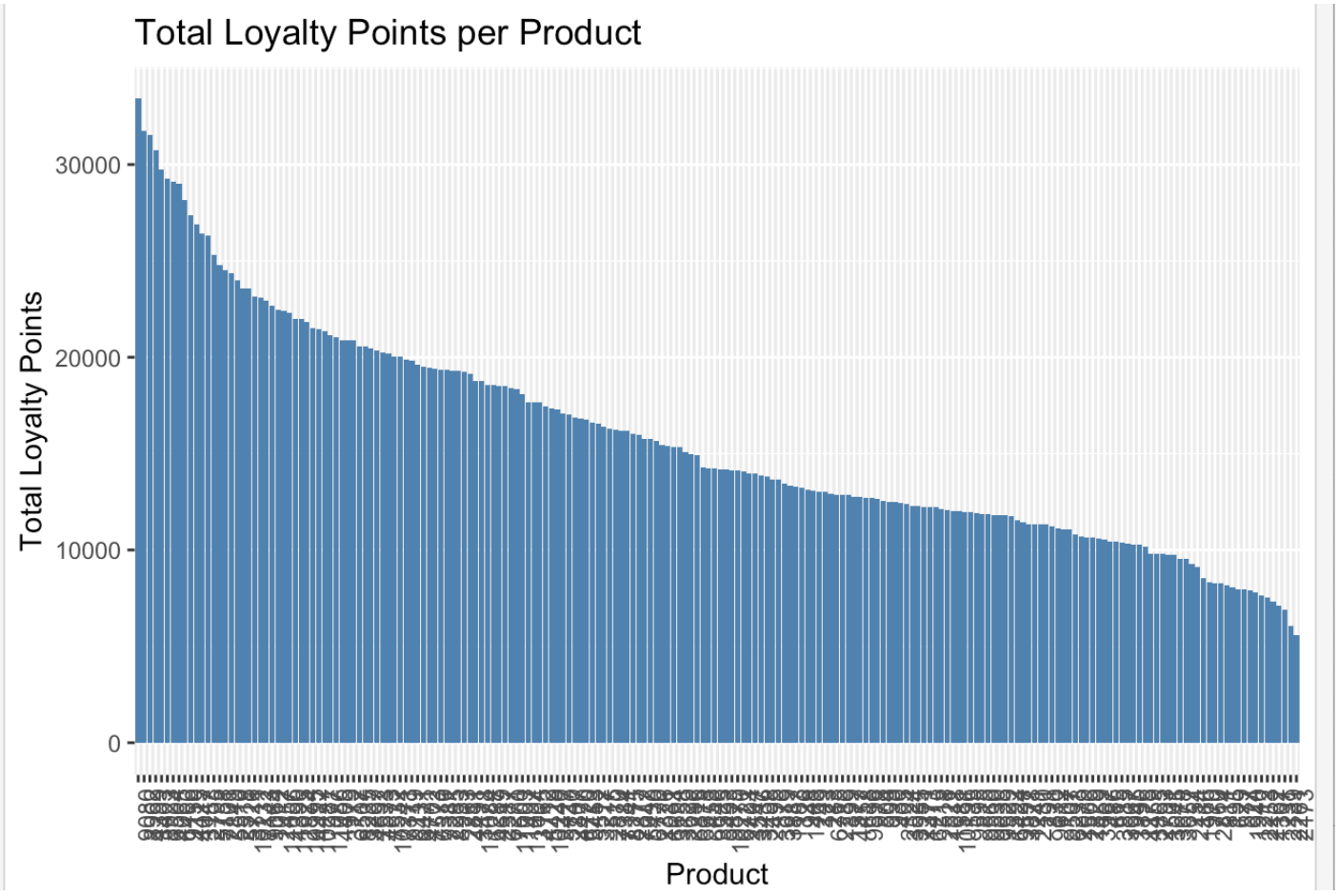
m) Analysis of product sales - method

Analysis of product sales in R

- # Looking at sales per product analysis by creating a proxy for sales using loyalty points
- # Analyse Reviews and Summaries for Top 5 Products based on loyalty points by product

```
> # Create a sales summary dataset
> sales_per_product <- reviews_new %>%
+   group_by(product) %>%
+   summarise(
+     total_loyalty_points = sum(loyalty_points),
+     average_age = mean(age),
+     average_spending_score = mean(spending_score),
+     count_reviews = n()
+   )
> # Inspect the aggregated sales data
> head(sales_per_product)
# A tibble: 6 × 5
  product total_loyalty_points average_age
  <fct>                  <int>       <dbl>
1 107                    11117        30.9
2 123                    13035        35.7
3 195                     8156        42.8
4 231                     7124        33.2
5 249                    13079        48.9
6 254                     9538        34
# ℹ 2 more variables: average_spending_score <dbl>,
#   count_reviews <int>
>
```

```
> # View the unique product counts [Not telling us much]
> head(unique_products)
  product  n
1     107 10
2     123 10
3     195 10
4     231 10
5     249 10
6     254  8
>
```

The exploratory analysis showed the maximum sales per product at 10 sales for the majority of products which would limit the amount of product sale analysis that could be done .

The explore I created a list of top product reviews that might help to help to analyse which products the reviews of the top selling products.

```
+   theme(axis.text.x = element_text(angle = 90, hjust = 1))
> # Step 1: Calculate total loyalty points per product
> top_products <- reviews_new %>%
+   group_by(product) %>%
+   summarise(total_loyalty_points = sum(loyalty_points, na.rm = TRUE)) %>%
+   arrange(desc(total_loyalty_points)) %>%
+   slice(1:5) # Get top 5 products
> # View the top 5 products
> print(top_products)
# A tibble: 5 × 2
  product total_loyalty_points
  <fct>                  <int>
1 9080                   33408
2 8962                   31726
3 4399                   31556
4 3403                   30725
5 8923                   29722
> # Step 2: Join the reviews and summaries to the top products
> top_product_reviews <- top_products %>%
+   left_join(reviews_new %>% select(product, review, summary), by = "product")
> View(top_product_reviews)
>
```

n) Predicting loyalty points - method

```
> # Predict loyalty points for new data for new customer Lina (see presentation)
> new_data <- data.frame(age = c(36), renumeration = c(74),
+                        spending_score = c(82))
> predictions <- predict(modelc, new_data)
> print(predictions)
       1
3514.767
```

- Known data for new customer Lina
  - Age = 36, Annual income = 74,000, Spending score = 82
- Based on model c prediction, Lina will generate approximate 3500 loyalty points
- This would place her in ‘cluster 0: High Spender’ segment segment