Advanced Analytics to Improve Sales at Turtle Games

Technical Report

By Chris Buck

Business Context

Turtle Games manufactures and sells games globally. It wants to use advanced analytics to answer the following questions:

1. Which customer details best predict the number of loyalty points they accumulate?
2. How can customers be grouped for targeted marketing to improve sales?
3. How can customer reviews inform marketing campaigns to improve sales?
4. Is the loyalty points data suitable for predictive modeling?

This project answers these questions by using the Mutually Exclusive Collectively Exhaustive (MECE) framework to reveal opportunities for increasing Turtle Games' sales with targeted marketing and loyalty program optimization.

Analytical Approach

<u>Data Cleaning (Python):</u>

I dropped columns (language, platform) with only one unique value, cleaned the column names that had special characters and changed the data type of the product column from integer to string.

```
# Drop unnecessary columns.
reviews.drop(['language', 'platform'], axis=1, inplace=True)

# View column names.
reviews.columns
```

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

```
# Rename the column headers.
reviews.rename(columns={'remuneration (k£)': 'income', 'spending_score (1-100)': 'spending_score'}, inplace=True)

# View column names.
reviews.columns
```

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

```
# Change 'product' column from integer to string data type
reviews['product'] = reviews['product'].astype(str)

# View column names.
reviews.dtypes
```

```
gender             object
age                 int64
income            float64
spending_score      int64
loyalty_points      int64
education          object
product            object
review             object
summary            object
dtype: object
```

Data Cleaning (R)

Only columns with more than one unique value were added to the new data frame. I cleaned the column names that had special characters and changed the data type of the product column from integer to character to make it easy to exclude it from the correlation matrix.

```r
# Create a new dataframe with fewer, renamed, and changed-type columns
clean_df <- data.frame(
  gender = df$gender,
  age = df$age,
  income = df$remuneration..k..,
  spending_score = df$spending_score..1.100.,
  loyalty_points = df$loyalty_points,
  education = df$education,
  product = as.character(df$product),
  review = df$review,
  summary = df$summary
)
```

Data Validation (Python)

I used a function to perform missing, duplicate, and unique value checks as well as compute summary statistics. I used the latter to sense check numerical columns like age and income. The large range of loyalty points was noteworthy and relevant to the fourth business question.

```python
# Data validation function
def validate_data(df):
    """
    This function compiles several functions for validating a dataframe.

    Arg:
        df: dataframe name

    Returns:
        prints missing counts
        prints unique counts per column
        prints duplicate counts
        prints summary statistics for the dataframe
    """
    #check for missing values
    missing_counts = df.isnull().sum()
    print("Missing values per column:")
    print(missing_counts)

    #check for unique values
    unique_counts = df.nunique()
    print("\nUnique values per column:")
    print(unique_counts)

    #checking for duplicates
    duplicate_count = df.duplicated().sum()
    print("\nNumber of duplicate rows:")
    print(duplicate_count)

    #summary stats
    summary_stats = df.describe()
    print("\nSummary statistics:")
    print(summary_stats)
```

```python
# Validate data
validate_data(reviews)
```

```
Missing values per column:
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

Unique values per column:
gender                       2
age                         45
remuneration (k£)           64
spending_score (1-100)      84
loyalty_points             627
education                    5
language                     1
platform                     1
product                    200
review                    1980
summary                   1432
dtype: int64

Number of duplicate rows:
0

Summary statistics:
               age  remuneration (k£)  spending_score (1-100)  loyalty_points
count  2000.000000        2000.000000             2000.000000     2000.000000
mean     39.495000          48.079060               50.000000     1578.032000
std      13.573212          23.123984               26.094702     1283.239705
min      17.000000          12.300000                1.000000       25.000000
25%      29.000000          30.340000               32.000000      772.000000
50%      38.000000          47.150000               50.000000     1276.000000
75%      49.000000          63.960000               73.000000     1751.250000
max      72.000000         112.340000               99.000000     6847.000000

           product
count  2000.000000
mean   4320.521500
std    3148.938839
min     107.000000
25%    1589.250000
50%    3624.000000
75%    6654.000000
max   11086.000000
```

Data Validation (R)

I checked the reviews for a single product number to verify they are the same product but this doesn't seem to be the case so I did not use this data in the analysis.

```r
# Filter clean_df by product number
filter(clean_df, product == "9080")
```

```
review
## 1
Awesome gift
## 2
Great tool #SchoolPsychologist
## 3
If I could give this egg zero stars I would. It is poorly made and rudiculously hard to open. What should be a te
nder moment spent with your children is a huge headache. I had to use a knife to open it and the knife literally
broke off. That is how difficult it is. Horrible product. Dont buy.
## 4
Perfect!
## 5
Tough game to learn just out of the box. Went online and watched some videos which helped A LOT. Played it first
time though and ran into all sorts of things that weren't covered in the video so I was lost. After just playing
a couple times though and having the manuals with you to reference, it became a lot of fun with no need to refere
nce the manuals. All in all, it's a fun board game. Not just for "nerds" either. All can have fun with this game.
## 6
well worth the money! I was initially irked that I had to pay so much for some bits of printed paper, but these a
re sturdy, like a baby's board book, and nice quality! I'm looking forward to getting my other sets to mix and ma
tch.
## 7
Came as exactly as described. This expansion makes Lords of Waterdeep more fun and more complex.  Love it!
## 8  Just got back from an evening at the Green Dragon.  I heard some marvelous tales of daring do and adventure
some hobbits.  Lots of fun.  I lured my wife and daughter in to a game, just like Gandalf did with his tale to Be
orn.  Before they knew it we were spinning tales with the best of them. The games is very easy to play.  We were
all able to quickly use the cards as prompts the store along with hardly a pause or break in the action.\n\nThe c
omponents are nice with thick stock, nice art, and a sturdy box to carry on adventures.\n\nThis game is lots of f
un for families.  Can't wait until the next time we visit the Green Dragon.\n\nLots of fun!
## 9
We bought this for our son for Christmas when he was 12.  He is into Legos, WII and math.  He played with this lo
nger than everything else he got, and all the friends that are into math and building love it.  I wouldn't give i
t to younger kids who will lose the pieces, but it is a great toy for traveling.  We kept it on the coffee table
and you wouldn't believe how many adults picked it up and played with it.  It feels really cool in your hand, ver
y solid.  We now look for more toys like this.
## 10
This is a great tool to have at hand when playing Quiddler.
```

Linear Regression

With the assistance of AI chatbots I generated a comprehensive function that handles both simple and multiple linear regressions by providing the relevant visualisations and statistical summaries to evaluate the models.

```python
# Linear regression function
def lin_reg(df, X, y, **kwargs):
    """
    This function was created in conversation with ChatGPT and the Claude AI Chatbot.
    It can perform a simple or multiple linear regression using the selected dataframe, x variable(s), and y variable.
    It uses an 80/20 train and test split.
    It provides summary stastics, residual plots, residual histograms, and q-q plots.
    It plots the regression line for simple linear regressions.
    It provides VIF statistics for multiple linear regressions.

    Args:
        df: dataframe name
        X: column name(s) for independent variable(s)
        y: column name for dependent variable

    Returns:
        prints head and tail of predicted y values
        prints error statistics
        prints R-squared statistics
        prints OLS summary statistics for the test set
        prints residuals plot, histogram and q-q plot
        prints regression line (SLR)
        prints VIF values (MLR)
    """
```

I used the sklearn library because it supports splitting the data into train and test sets, which is important when using the models for predictive purposes.

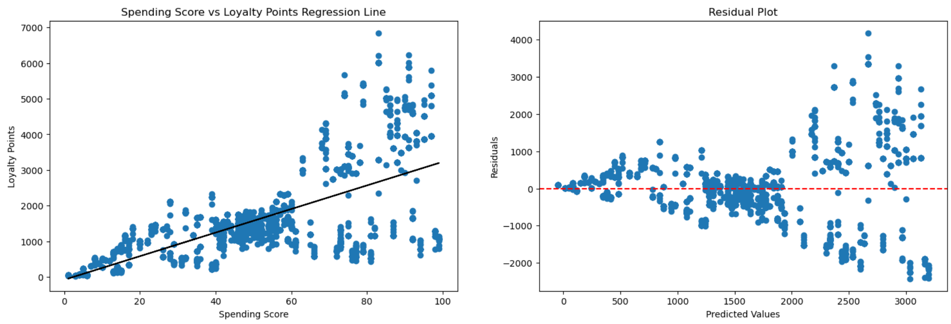
```
# Define the dependent variable
y = df[y]

# Define the independent variables
X = df[X]

# Split the data into training (80%) and testing (20%) sets
x_train, x_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)

# Run regression on the train subset
mlr = LinearRegression()
mlr.fit(x_train, y_train)

# Predict on training data
y_pred_train = mlr.predict(x_train)

# Calculate residuals
residuals = y_train - y_pred_train

# Predictions on the test set
y_pred_test = mlr.predict(x_test)

# View the output head and tail
print("Predicted y values (head and tail):")
print(y_pred_test[:5])
print("...")
print(y_pred_test[-5:],"\n")
```

I included the OLS summary statistics table because it provides the coefficient and F-statistic p-value and helps validate the R-squared value for the training set.

```
X = ['spending_score']
lin_reg(reviews, X, 'loyalty_points')
```

```
Predicted y values (head and tail):
[  50.83207662 2435.00839173 2269.44059207 1309.14735404 1441.60159377]
...
[2401.8948318  1805.85075302 2799.25755099 1739.62363316 2832.37111092]

Mean Absolute Error: 651.6958975724068
Mean Squared Error: 865341.5814675748
Root Mean Squared Error: 930.2373790960966
R-squared for training set:  44.83889403237179
R-squared for test set:  46.638946430618866

                            OLS Regression Results
==============================================================================
Dep. Variable:         loyalty_points   R-squared:                       0.448
Model:                            OLS   Adj. R-squared:                  0.448
Method:                 Least Squares   F-statistic:                     1299.
Date:                Sun, 13 Apr 2025   Prob (F-statistic):          1.09e-208
Time:                        20:24:46   Log-Likelihood:                -13248.
No. Observations:                1600   AIC:                         2.650e+04
Df Residuals:                    1598   BIC:                         2.651e+04
Df Model:                           1
Covariance Type:            nonrobust
==================================================================================
                     coef    std err          t      P>|t|      [0.025      0.975]
----------------------------------------------------------------------------------
const            -81.6222     52.028     -1.569      0.117    -183.672      20.428
spending_score    33.1136      0.919     36.041      0.000      31.311      34.916
==============================================================================
Omnibus:                      100.304   Durbin-Watson:                   1.995
Prob(Omnibus):                  0.000   Jarque-Bera (JB):              205.865
Skew:                           0.414   Prob(JB):                     1.98e-45
Kurtosis:                       4.549   Cond. No.                         123.
==============================================================================
```

These plots show that the relationship between spending score and loyalty points doesn't appear to be linear and that the residuals are not homoscedastic.



The output for multiple linear regressions includes the adjusted R-squared value for the test set as well as the VIF values to check for multicollinearity. This model demonstrates the effectiveness of income and spending score in predicting the loyal points customers accumulate.

```
# Select x variables
X = ['income', 'spending_score']

# Run multiple linear regression function
lin_reg(reviews, X, 'loyalty_points')
```

```
Predicted y values (head and tail):
[-725.11711735 2976.65149188 2644.50590347 1416.28990278 1434.2479047 ]
...
[3141.08822013 1680.71301019 1956.16988192 1390.19029468 3227.60609785]

Mean Absolute Error: 429.66362016909125
Mean Squared Error: 300944.0917834269
Root Mean Squared Error: 548.5837144715716
R-squared for training set:  82.98594267896443
R-squared for test set:  81.44236432529975

                            OLS Regression Results
==============================================================================
Dep. Variable:         loyalty_points   R-squared:                       0.830
Model:                            OLS   Adj. R-squared:                  0.830
Method:                 Least Squares   F-statistic:                     3895.
Date:                Sun, 13 Apr 2025   Prob (F-statistic):               0.00
Time:                        20:24:47   Log-Likelihood:                -12307.
No. Observations:                1600   AIC:                         2.462e+04
Df Residuals:                    1597   BIC:                         2.464e+04
Df Model:                           2
Covariance Type:            nonrobust
==================================================================================
                     coef    std err          t      P>|t|      [0.025      0.975]
----------------------------------------------------------------------------------
const          -1700.3237     39.588    -42.950      0.000   -1777.974   -1622.674
income            34.3346      0.574     59.838      0.000      33.209      35.460
spending_score    32.6439      0.510     63.947      0.000      31.643      33.645
==============================================================================
Omnibus:                        2.977   Durbin-Watson:                   2.034
Prob(Omnibus):                  0.226   Jarque-Bera (JB):                2.923
Skew:                           0.075   Prob(JB):                        0.232
Kurtosis:                       3.147   Cond. No.                         220.
==============================================================================

Notes:
[1] Standard Errors assume that the covariance matrix of the errors is correctly specified.
Adjusted R-squared for training set:  82.96463515570703
Adjusted R-squared for test set:  81.3488749768126

Variance Inflation Factors:
   VIF Factor        features
0         8.9           const
1         1.0          income
2         1.0  spending_score
```

Decision Tree

I performed a decision tree regression because it is non-linear and can incorporate categorical variables that might help predict customer loyalty point accumulation.

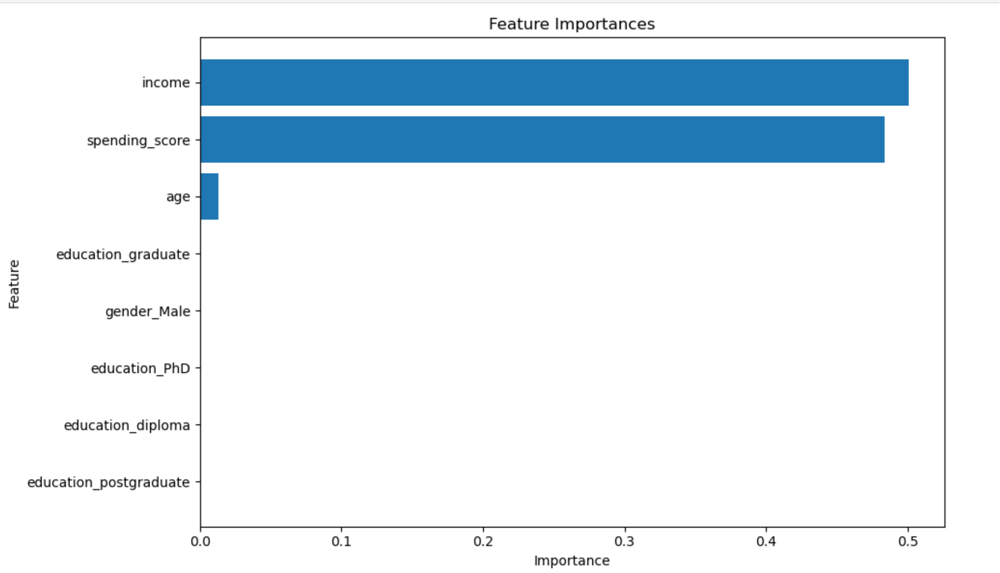
```
# Make dummy variables for categorical features
categorical_cols = ['gender', 'education']
decision = pd.get_dummies(decision, columns=categorical_cols, drop_first=True)
```

gender_Male	education_PhD	education_diploma	education_graduate	education_postgraduate
True	False	False	True	False
True	False	False	True	False
False	False	False	True	False
False	False	False	True	False
False	False	False	True	False
...	...	...	...	...
False	True	False	False	False
False	True	False	False	False
True	False	False	True	False
True	True	False	False	False
True	True	False	False	False

I separated the data into train and test sets to avoid overfitting the model using the same 80/20 split as before.

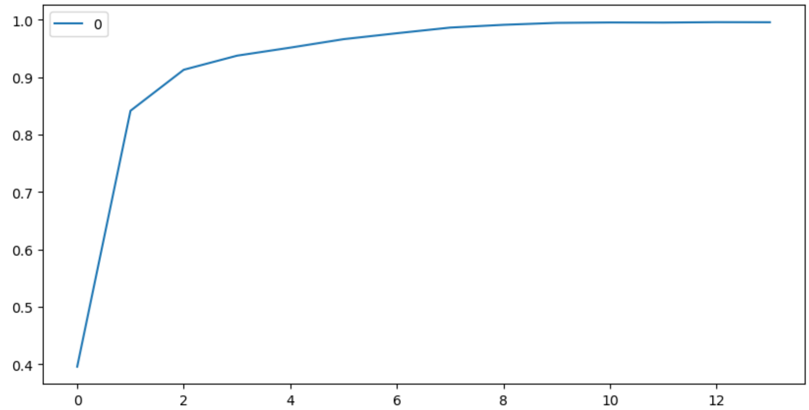
```
# Isolate the independent variables from loyalty_points
cols = decision.columns[decision.columns != 'loyalty_points']

# Specify X as independent variables and y as dependent variable.
X = decision[cols]
y = decision['loyalty_points']

# Split the data training and testing 20/80.
X_train, X_test, y_train, y_test = train_test_split(X, y,
                                                    test_size=0.2,
                                                    random_state=42)
```

Despite the inclusion of gender and education, they were not important features for the model.



I plotted the accuracy values to determine an optimal pruning range.

```
# Make list of values to try for max_depth
max_depth_range = list(range(1, 15))

# have a loop that fits decision trees for different 'max_depth', and calculate/store the accuracy of each tree
accuracy = []
for depth in max_depth_range:
    regressor2 = DecisionTreeRegressor(max_depth = depth, random_state = 42)
    regressor2.fit(X_train, y_train)
    score = regressor2.score(X_test, y_test)
    accuracy.append(score)

# Plot accuracy values across the range of depth values evaluated
plt.rcParams['figure.figsize'] = [10, 5]
accuracy = pd.DataFrame(accuracy)
accuracy.plot()
plt.show()
```



I pruned at a max depth of 3 branches because additional depth provides diminishing returns. At this depth income and spending score are still sufficient predictors and the nonlinear nature of the decision tree regression model makes it a better fit than the MLR as indicated by the higher R-squared score (91.29% vs 81.44%).

```
# Prune the model to a max depth of 3
dtr = DecisionTreeRegressor(max_depth=3, random_state=42)
```

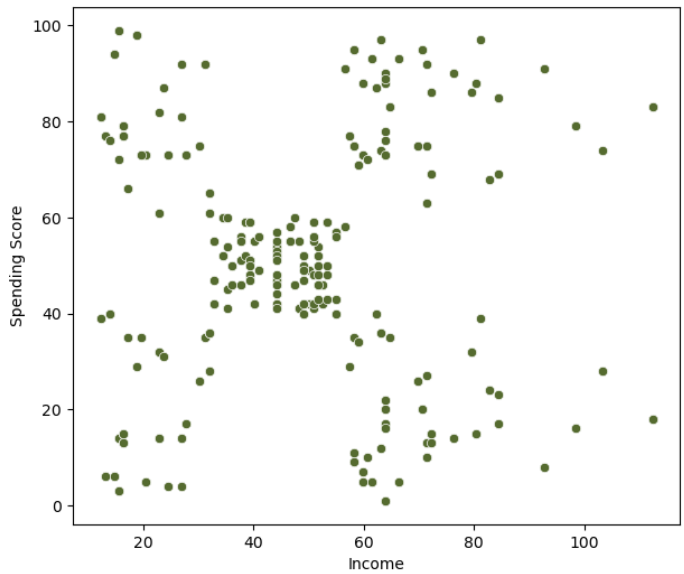
```
#Train Decision Tree Regressor
dtr.fit(X_train, y_train)

#Predict TotalCharges on test data
y_pred_pruned = dtr.predict(X_test)

#Evaluate the pruned model
mse_pruned = mean_squared_error(y_test, y_pred_pruned)
mae_pruned = mean_absolute_error(y_test, y_pred_pruned)
rmse_pruned = mean_squared_error(y_test, y_pred_pruned, squared=False)
r2_pruned = r2_score(y_test, y_pred_pruned)

print(f"Pruned Model Mean Squared Error: {mse_pruned}")
print(f"Pruned Model Mean Absolute Error: {mae_pruned}")
print(f"Pruned Model R-squared: {r2_pruned}")
print(f"Root Mean Squared Error: {rmse_pruned}")
```

```
Pruned Model Mean Squared Error: 141276.38891623393
Pruned Model Mean Absolute Error: 271.8726068995849
Pruned Model R-squared: 0.9128822985223626
Root Mean Squared Error: 375.86751511168654
```

Clustering

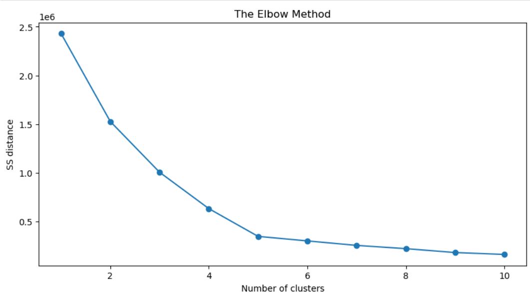
Unsupervised machine learning is a good way to generate MECE clusters to answer the second business question regarding customer groups for targeted marketing. I chose income and spending score because their importance has already been demonstrated by two different regression models.

```
#Clean labels
clean_income = clean_label('income')
clean_spending = clean_label('spending_score')

# Create a scatterplot with Seaborn.
plt.figure(figsize=(7, 6))
ax = sns.scatterplot(x='income',
                y='spending_score',
                data=cluster, color='darkolivegreen')
ax.set(xlabel=clean_income, ylabel=clean_spending)

# Save plot
fig = ax.get_figure()
fig.savefig("income_spending.png", bbox_inches = "tight")
```

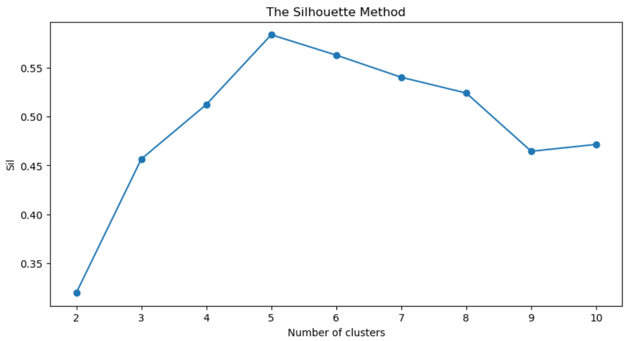


The scatterplot suggests there are 5 clusters and I confirmed this with the elbow and silhouette methods.

```
# Determine the number of clusters: Elbow method.
ss = []
for i in range(1, 11):
    kmeans = KMeans(n_clusters=i,
                    init='k-means++',
                    max_iter=300,
                    n_init=10,
                    random_state=0)
    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()
```



```python
# Determine the number of clusters: 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()
```



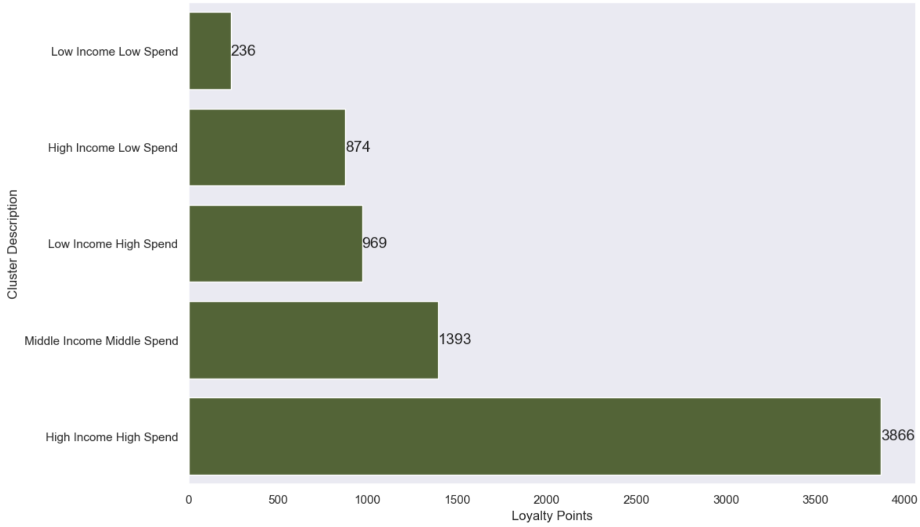
To explore the relationship between these clusters of customers and loyalty point accumulation I added the K-means predicted values to the main data frame and then mapped more descriptive labels onto them.

```python
# Add K-means predicted values to the main dataframe
reviews_cluster = reviews_cluster.set_index(['income', 'spending_score'])
x = x.set_index(['income', 'spending_score'])

# Then join the K-Means column
reviews_cluster['kmeans_cluster'] = x['K-Means Predicted']

# Reset index if needed
reviews_cluster = reviews_cluster.reset_index()
```

```python
# Create your mapping dictionary
cluster_mapping = {
    0: 'High Income High Spend',
    1: 'Middle Income Middle Spend',
    2: 'High Income Low Spend',
    3: 'Low Income High Spend',
    4: 'Low Income Low Spend'
}

# Apply the mapping
reviews_cluster['cluster_description'] = reviews_cluster['kmeans_cluster'].map(cluster_mapping)
```

Unsurprisingly the high income, high spending score customers accumulate the most loyalty points on average.

```python
# Clean labels
clean_cluster = clean_label('cluster_description')
clean_loyalty = clean_label('loyalty_points')

# Horizontal bar plot with value labels and no grid lines
ax = sns.barplot(cluster_points, y="cluster_description", x="loyalty_points", orient="h", color='darkolivegreen')
ax.bar_label(ax.containers[0], fontsize=14)
ax.set(xlabel=clean_loyalty, ylabel=clean_cluster)
ax.grid(False)

fig = ax.get_figure()
fig.savefig("cluster_loyalty.png", bbox_inches = "tight")
```



Sentiment Analysis

I used a preprocessing function due to the informal nature of online reviews.

```python
# Preprocessing function
def preprocess_text(text):
    '''
    This function removes URLs, hashtags, special characters, and stop words.
    It also converts uppercase letters to lowercase.

    Arg:
        text: text to be cleaned
    Return:
        Cleaned text
    '''
    text = contractions.fix(text)  # Expand contractions i.e I'm not good goes to I am not good
    text = re.sub(r'http\S+', '', text)  # Remove URLs
    text = re.sub('#', '', text)         # Remove hashtags
    text = re.sub(r'\W', ' ', text)      # Remove special characters
    text = text.lower()                  # Convert to lowercase
    #Below is to create a set of stop words from the NLTK library's predefined list but not is excluded.
    stop_words = set(stopwords.words('english')) - {'not'}
    text = ' '.join([word for word in text.split() if word not in stop_words])
    return text
```

cleaned_review	cleaned_summary
comes dm screen space screen absolute premium ...	fact 50 space wasted art not terribly informat...
open letter galeforce9 unpainted miniatures no...	another worthless dungeon master screen galefo...
nice art nice printing two panels filled gener...	pretty also pretty useless
amazing buy bought gift new dm perfect	five stars
review gf9 previous screens completely unneces...	money trap
...	...
perfect word game mixed ages mom perhaps givin...	perfect word game mixed ages mom
great game not think would like first received...	super fun
great game keeps mind nimble	great game
fun game	four stars
game fun lot like scrabble without little tile...	love game

I applied tokenization and lemmatization in preparation for the sentiment analysis.

```python
# Create new dataframe
reviews_token = reviews_sentiment

# Apply tokenization to cleaned review and summary columns
reviews_token['tokenized_summary'] = reviews_token['cleaned_summary'].apply(word_tokenize)
reviews_token['tokenized_review'] = reviews_token['cleaned_review'].apply(word_tokenize)

# View dataframe
reviews_token
```

```python
# Create new dataframe
reviews_lemmatized = reviews_token

# Define the tag map for POS tagging
tag_map = defaultdict(lambda: wn.NOUN)
tag_map['J'] = wn.ADJ
tag_map['V'] = wn.VERB
tag_map['R'] = wn.ADV

# Lemmatize the tokens with correct POS tags
lemma_function = WordNetLemmatizer()

def lemmatize_tokens(tokens):
    #For each word in the token list, it lemmatizes the word with the correct part-of-speech
    lemmatized_tokens = [lemma_function.lemmatize(token, tag_map[tag[0]]) for token, tag in pos_tag(tokens)]
    return lemmatized_tokens

reviews_lemmatized['lemmatized_summary'] = reviews_lemmatized['tokenized_summary'].apply(lemmatize_tokens)
reviews_lemmatized['lemmatized_review'] = reviews_lemmatized['tokenized_review'].apply(lemmatize_tokens)
```

tokenized_summary	tokenized_review	lemmatized_summary	lemmatized_review
[fact, 50, space, wasted, art, not, terribly, ...	[comes, dm, screen, space, screen, absolute, p...	[fact, 50, space, waste, art, not, terribly, i...	[come, dm, screen, space, screen, absolute, pr...
[another, worthless, dungeon, master, screen, ...	[open, letter, galeforce9, unpainted, miniatur...	[another, worthless, dungeon, master, screen, ...	[open, letter, galeforce9, unpainted, miniatur...
[pretty, also, pretty, useless]	[nice, art, nice, printing, two, panels, fille...	[pretty, also, pretty, useless]	[nice, art, nice, printing, two, panel, fill, ...
[five, stars]	[amazing, buy, bought, gift, new, dm, perfect]	[five, star]	[amaze, buy, buy, gift, new, dm, perfect]
[money, trap]	[review, gf9, previous, screens, completely, u...	[money, trap]	[review, gf9, previous, screen, completely, un...

I chose the Vader library for sentiment analysis because it is specifically designed to handle informal online text like you find in product reviews.

```
# Create a variable sia to store the SentimentIntensityAnalyser() method.
sia = SentimentIntensityAnalyzer()
```

```
# Run through a dictionary comprehension to take every cleaned comment
# Next run the polarity score function on the string.
# This will return four values in a dictionary

df_polarity_review = {" ".join(_) : sia.polarity_scores(" ".join(_)) for _ in reviews_lemmatized['lemmatized_review']}
```

```
# Convert the list of dictionary results to a Pandas DataFrame.
# The index is the cleaned review
polarity_review = pd.DataFrame(df_polarity_review).T

# View the DataFrame.
polarity_review
```

I used a function to convert compound sentiment scores into positive, neutral, and negative categories in accordance with MECE framework. This helps simplify the data relevant to the third business question for stakeholders.

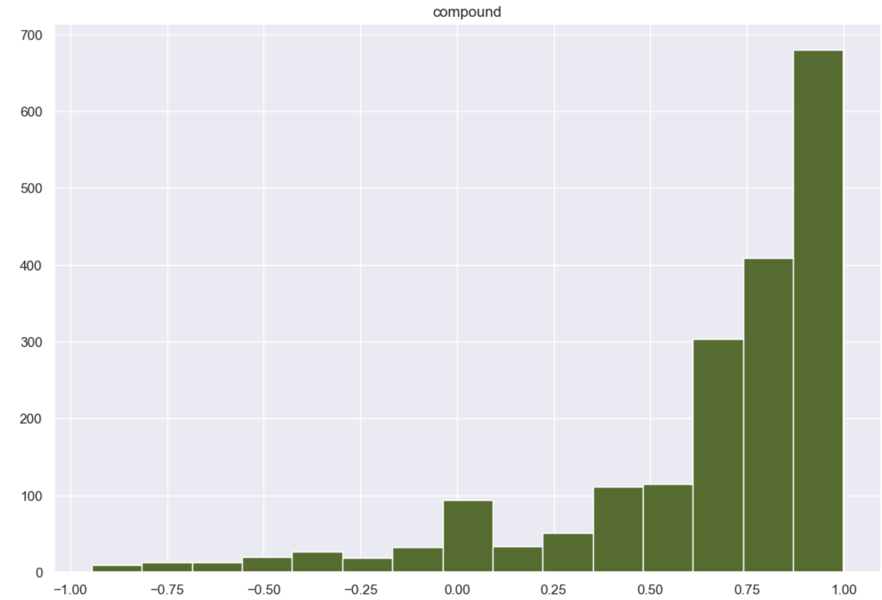
```
# Sentiment categories
def sentiment_cat(value):
    '''
    This function can be used when creating a new categorical column for sentiment based on the compound polarity score

    Arg:
        value: the compound polarity score

    Return:
        category (positive, negative, or neutral)
    '''
    if value > 0.05:
        return "positive"
    elif value < -0.05:
        return "negative"
    else:
        return "neutral"
```

```
# Add a sentiment category column based on summary compound score
polarity_summary['sentiment'] = polarity_summary.apply(lambda row: sentiment_cat(row['compound']), axis = 1)
polarity_summary
```

	neg	neu	pos	compound	sentiment
fact 50 space waste art not terribly informative need art	0.204	0.583	0.213	0.0320	neutral
another worthless dungeon master screen galeforce9	0.367	0.633	0.000	-0.4404	negative
pretty also pretty useless	0.275	0.098	0.627	0.5574	positive
five star	0.000	1.000	0.000	0.0000	neutral
money trap	0.697	0.303	0.000	-0.3182	negative
...	...	...	...	...	...
fun card game people like word	0.000	0.408	0.592	0.7003	positive
sort card game equivalent scrabble lot easy	0.000	0.674	0.326	0.4404	positive
great game keep mind active	0.000	0.306	0.694	0.7783	positive
great mind game	0.000	0.328	0.672	0.6249	positive
perfect word game mixed age mom	0.000	0.575	0.425	0.5719	positive

A histogram shows that the overall sentiment expressed in the reviews is positive.

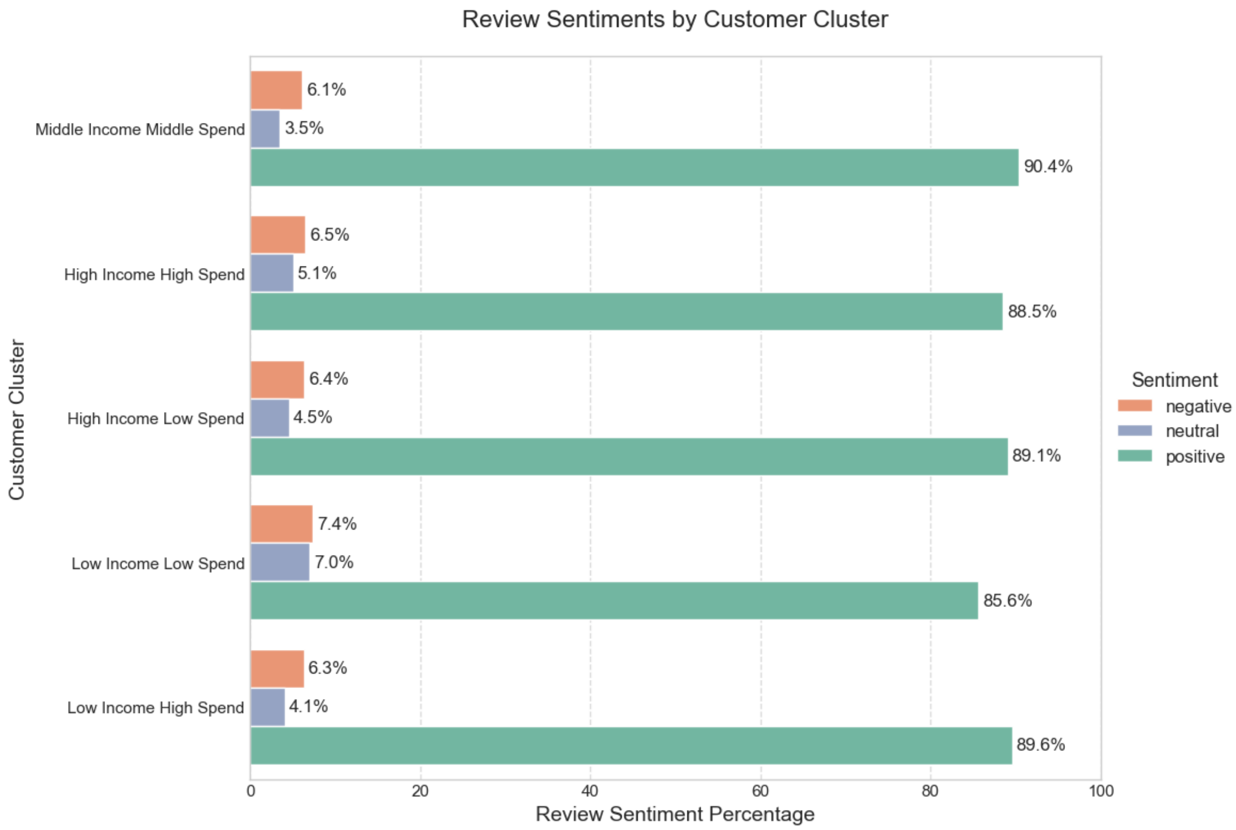
```
# Histogram of review compound scores
polarity_review['compound'].sort_values().reset_index().hist(bins=15, color='darkolivegreen')
plt.savefig('review_sentiment_hist.png')
```



The sentiment analysis of reviews is more accurate than that of the summaries because presumably positive summaries like “five stars” are categorized as neutral.

summary	kmeans_cluster	...	compound_review	neg_review	neu_review	pos_review	compound_summary	neg_summary	neu_summary	pos_summary	sentiment_review	sentiment_summary
Five Stars	3	...	0.8779	0.0	0.284	0.716	0.0	0.0	1.0	0.0	positive	neutral
Five Stars	3	...	0.6369	0.0	0.192	0.808	0.0	0.0	1.0	0.0	positive	neutral
Five Stars	3	...	0.5719	0.0	0.351	0.649	0.0	0.0	1.0	0.0	positive	neutral
Five Stars	4	...	0.2732	0.0	0.323	0.677	0.0	0.0	1.0	0.0	positive	neutral
Five Stars	3	...	0.5106	0.0	0.377	0.623	0.0	0.0	1.0	0.0	positive	neutral
...	...	...	...	...	...	...	...	...	...	...	...	...
Five Stars	4	...	0.2960	0.0	0.000	1.000	0.0	0.0	1.0	0.0	positive	neutral
Five Stars	3	...	0.9042	0.0	0.152	0.848	0.0	0.0	1.0	0.0	positive	neutral
Five Stars	4	...	0.6249	0.0	0.000	1.000	0.0	0.0	1.0	0.0	positive	neutral
Five Stars	2	...	0.7430	0.0	0.241	0.759	0.0	0.0	1.0	0.0	positive	neutral
Five Stars	4	...	0.0000	0.0	1.000	0.000	0.0	0.0	1.0	0.0	neutral	neutral

With the help of the Claude AI chatbot I was able generate a plot of sentiment by customer cluster to reveal patterns that can inform Turtle Games' targeted marketing strategy. For example, the low income, low spending score customers might be vulnerable to churn because it has the highest percentage of negative reviews.



Descriptive Statistics

To answer the fourth business question, I wrote a function in R that returns statistical measures of central tendency, variability, shape, and normality and applied it to the loyalty points data. The loyalty points data fails normality tests but using a log transformation might help in future analyses.

```r
descriptive_stats <- function(x){
  # Calculate mean, median, and mode
  mean_score <- mean(x)
  median_score <- median(x)
  mode_score <- as.numeric(names(sort(table(x), decreasing = TRUE)[1]))

  # Calculate Range
  range_satisfaction <- range(x)

  # Calculate Difference between highest and lowest values
  difference_high_low <- diff(range_satisfaction)

  # Calculate Interquartile Range (IQR)
  iqr_satisfaction <- IQR(x)

  # Calculate Variance
  variance_satisfaction <- var(x)

  # Calculate Standard Deviation
  std_deviation_satisfaction <- sd(x)

  # Skewness and Kurtosis
  skewness_score = skewness(x)
  kurtosis_score = kurtosis(x)

  # Shapiro-Wilk test for normality
  shapiro_score = shapiro.test(x)
  shapiro_p <- shapiro_score$p.value

  # Print the results
  cat("MEASURES OF CENTRAL TENDENCY", "\n")
  cat("Mean:", mean_score, "\n")
  cat("Median:", median_score, "\n")
  cat("Mode:", mode_score, "\n")
  cat("\n")
  cat("MEASURES OF VARIABILITY", "\n")
  cat("Range:", range_satisfaction, "\n")
  cat("Difference:", difference_high_low, "\n")
  cat("IQR:", iqr_satisfaction, "\n")
  cat("Variance:", variance_satisfaction, "\n")
  cat("Standard Deviation:", std_deviation_satisfaction, "\n")
  cat("\n")
  cat("MEASURES OF SHAPE", "\n")
  cat("Skewness:", skewness_score, "\n")
  cat("Kurtosis:", kurtosis_score, "\n")
  cat("\n")
  cat("MEASURES OF NORMALITY", "\n")
  cat("Shapiro-Wilk p-value:", formatC(shapiro_p, format = "e", digits = 2))
}
```

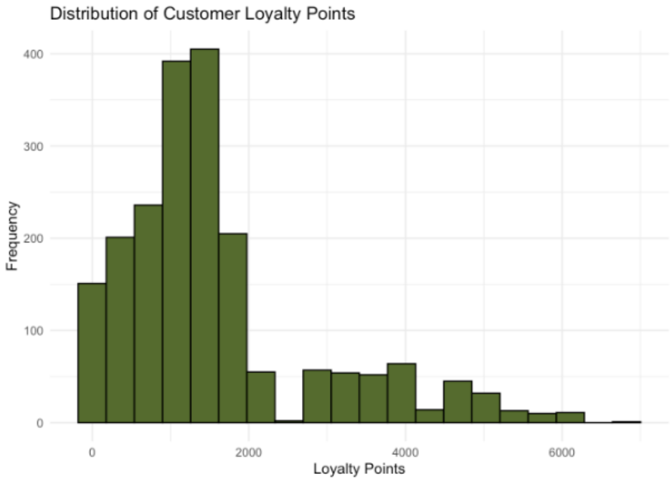
```r
descriptive_stats(clean_df$loyalty_points)
```

```
## MEASURES OF CENTRAL TENDENCY
## Mean: 1578.032
## Median: 1276
## Mode: 1014
##
## MEASURES OF VARIABILITY
## Range: 25 6847
## Difference: 6822
## IQR: 979.25
## Variance: 1646704
## Standard Deviation: 1283.24
##
## MEASURES OF SHAPE
## Skewness: 1.463694
## Kurtosis: 4.70883
##
## MEASURES OF NORMALITY
## Shapiro-Wilk p-value: 1.24e-40
```

Visualisation and Insights through Exploratory Data Analysis

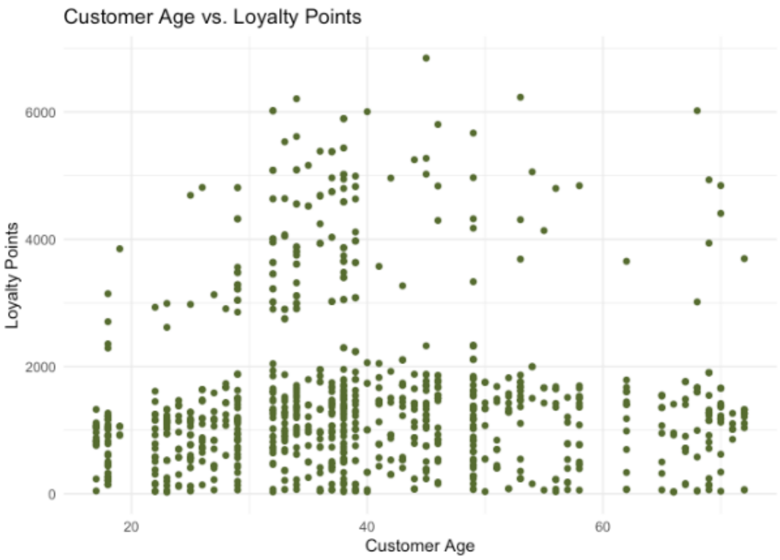
I plotted a histogram for loyalty points to give a visual illustration of the measures of shape (skewness and kurtosis).

```
# Create a histogram for 'loyalty_points'
plot <- ggplot(data = clean_df, aes(x = loyalty_points)) +
  geom_histogram(bins = 20, fill = "darkolivegreen", color = "black") +
  labs(title = "Distribution of Customer Loyalty Points", x = "Loyalty Points", y = "Frequency") +
  theme_minimal()

plot
```



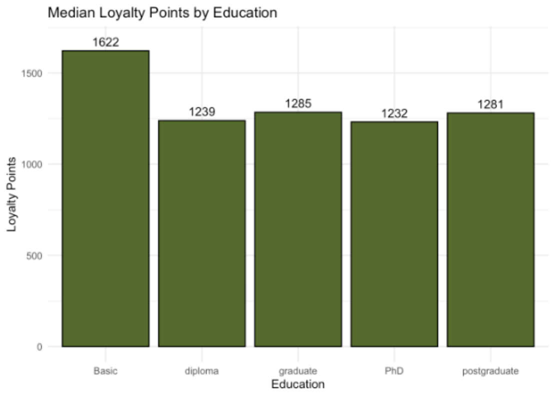
I used scatter plots in R for numerical variables and loyalty points to check for linearity before running linear regressions. This example shows how age and loyalty points do not appear to be correlated to each other, which is relevant to the first business question.

```
# Create a scatter plot for 'age' vs 'loyalty_points'
ggplot(data = clean_df, aes(x = age, y = loyalty_points)) +
  geom_point(color = "darkolivegreen") +
  labs(title = "Customer Age vs. Loyalty Points", x = "Customer Age", y = "Loyalty Points") +
  theme_minimal()
```



I used bar plots in R for categorical variables and median loyalty points. This example shows that customers with only a basic education level tend to accumulate more loyalty points on average than customers with higher levels of education.

```r
# Create a bar plot for average 'loyalty_points' by 'education'
clean_df %>%
  group_by(education) %>%
  summarize(m = median(loyalty_points)) %>%
  ggplot(aes(x = education, y = m)) +
    geom_bar(stat = "identity", fill = "darkolivegreen", color = "black") +
    labs(title = "Median Loyalty Points by Education", x = "Education", y = "Loyalty Points") +
    geom_text(aes(label = m), nudge_y = 50) +
    theme_minimal()
```

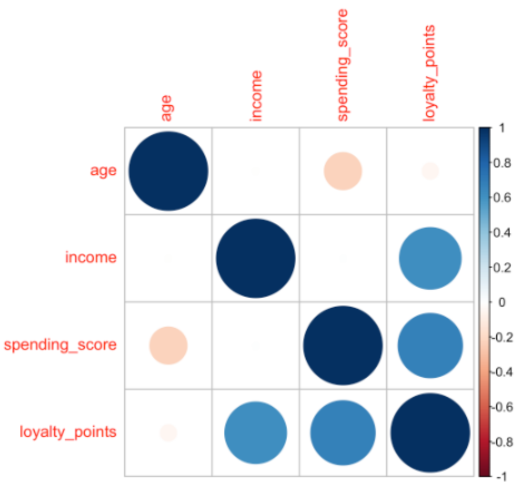


I plotted the correlation matrix because it is a good visual to share with stakeholders; it provides a concise answer to the first business question by making it clear that income and spending score are most correlated with loyalty points out of all the relevant numerical variables.

```r
# Calculate the correlation matrix
correlation_matrix <- cor(numeric_df, use = "complete.obs")

# View the correlation matrix
correlation_matrix
```

```
##                       age       income spending_score loyalty_points
## age             1.000000000 -0.005708284   -0.224334309    -0.04244465
## income         -0.005708284  1.000000000    0.005612492     0.61606475
## spending_score -0.224334309  0.005612492    1.000000000     0.67231011
## loyalty_points -0.042444647  0.616064748    0.672310112     1.00000000
```

```r
# Create the correlation plot
corrplot(correlation_matrix, method = "circle")
```



Patterns and Predictions:

1. Income and spending score used in a decision tree regression best predicts the number of loyalty points customers will accumulate.

2. Customers can be grouped into five categories for targeted marketing to improve sales:

    - Low income, low spending score
    - Low income, high spending score
    - Middle income, middle spending score
    - High income, low spending score
    - High income, high spending score.

    I predict that customers with mid-to-high incomes and low-to-mid spending scores can be encouraged to buy more products from Turtle Games through a promotion that enables them to earn bonus loyalty points on new purchases.

3. Customer reviews can inform targeted marketing campaigns by helping to identify clusters of customers who have a higher percentage of negative reviews and might be vulnerable to churn. Offering these customers additional loyalty points could be a way to keep their business and improve sales (Hollenbeck and Taylor, 2021).

4. The loyalty points data does not pass several normality tests but a log transformation might be able to fix this for future analyses. That said, the nature of the loyalty points system should be determined by what improves sales as opposed to what is most convenient for data analysts.

Technical Recommendations

- Each unique product should have a unique product number so it is possible to analyze the sentiments of reviews of specific products.
- The dataset should also include product names and categories because these are not always obvious from the reviews.
- Including a customer ID number in the dataset would make it possible to conduct more a more granular sentiment analysis.
- Including the date on which reviews are published could enable time series sentiment analyses.

Reference List

Competition and Markets Authority (2024) Review of loyalty pricing in the groceries sector. Available at: https://www.gov.uk/government/publications/review-of-loyalty-pricing-in-the-groceries-sector (Accessed: 10 April 2025).

Crouch, E. et al (2024) Loyalty programs are growing—so are customer expectations. Available at: https://www.bcg.com/publications/2024/loyalty-programs-customer-expectations-growing (Accessed: 8 April 2025).

Harvard Business Review (2021) Why customer loyalty programs can backfire. Available at: https://hbr.org/2021/05/why-customer-loyalty-programs-can-backfire (Accessed 8 April 2025).

Hollenbeck, B. and Taylor, W. (2021) How to make your loyalty program pay off. Available at: https://hbr.org/2021/10/how-to-make-your-loyalty-program-pay-off (Accessed 8 April 2025).