A DATA-DRIVEN ANALYSIS OF CUSTOMER BEHAVIOUR TO ENHANCE TURTLE GAMES' SALES STRATEGY

Kelvin Dhondee

Table of Contents

Background information

Turtle Games is a global game manufacturer and retailer selling proprietary and third-party products (books, board games, video games, toys). The company has extensive sales data and customer reviews but needs deeper insights to improve overall sales performance.

***Business Problem**: How can Turtle Games leverage existing customer data to identify trends, optimise marketing strategies, and ultimately increase sales?*

This analysis aims to systematically:

a) explore loyalty points accumulation/engagement patterns,
b) create meaningful customer segments for targeted marketing,
c) extract actionable insights from customer reviews,
d) and validate data suitability for predictive modelling through statistical analysis.

Analytical approach

Data wrangling and validation

In Python, having imported the necessary libraries, a user-defined function was used to validate the data (see screenshot of docstring). Two columns ('language' and 'platform' were dropped, another two columns were renamed (as 'remuneration' and 'spending_score' respectively) and the validated dataframe was saved as 'reviews.csv' for further analysis.

```
# Function to validate data.
def validate_data(df):
    """
    Performs comprehensive data validation and exploration on a pandas DataFrame.

    This function provides multiple views of the data to help identify potential issues
    and understand the dataset structure. It displays the first and last rows, shape,
    metadata, missing values, unique values, duplicates, and summary statistics.

    Parameters:
    -----------
    df : pandas.DataFrame
        The DataFrame to be validated.

    Returns:
    --------
    None
        Results are printed to the console.

    Notes:
    ------
    The function performs the following checks:
    - Displays first and last 5 rows of the DataFrame
    - Shows the shape (rows, columns)
    - Provides DataFrame metadata via info()
    - Counts missing values in each column
    - Counts unique values in each column
    - Counts duplicate rows
    - Displays summary statistics using describe()
    """
```

Data wrangling and validation of 'reviews.csv' in R followed a similar approach, checking for missing values and duplicates. The 'gender' and 'education' columns were converted to factors.

For a detailed breakdown of the dataset used for this analysis, including interpretations and limitations of each variable, please refer to Detailed description of turtle_reviews.csv.

For an overview of the Python and R libraries used, please refer to Libraries utilised in Python and R.

Choice of Analytical Methods

Machine Learning models (supervised):

- Linear Regression (Python & R)
- Decision Tree Regressor (Python)
- Random Forest Regressor (Python & R)
- Support Vector Machine classifier (Python)

Machine Learning models (unsupervised):

- K-Means Clustering (Python)
- Hierarchical Clustering (Python)

Natural Language Processing techniques:

- TextBlob (Python)
- Valence Aware Dictionary and sEntiment Reasoner (Python)

Statistical Hypothesis Tests:

- Student's t-test (R)
- Wilcoxon Rank-Sum test (Mann-Whitney U Test) (R)

For an explanation of the rationale behind the choice of each of these analytical methods, please refer to Rationale behind choice of analytical methods.

Assumptions and Limitations

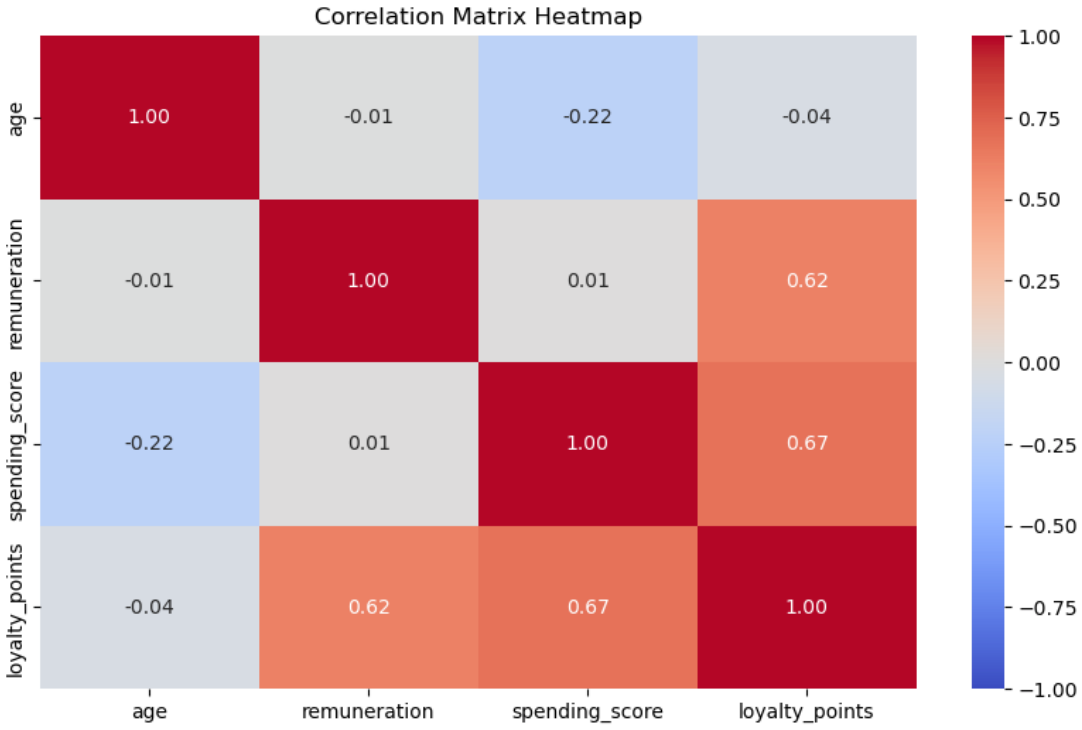
See Assumptions , limitations and concerns.

Visualisations and insights

EDA: How do customers engage with and accumulate loyalty points? Can descriptive statistics justify predictive models?

Investigating the relationship between numerical variables

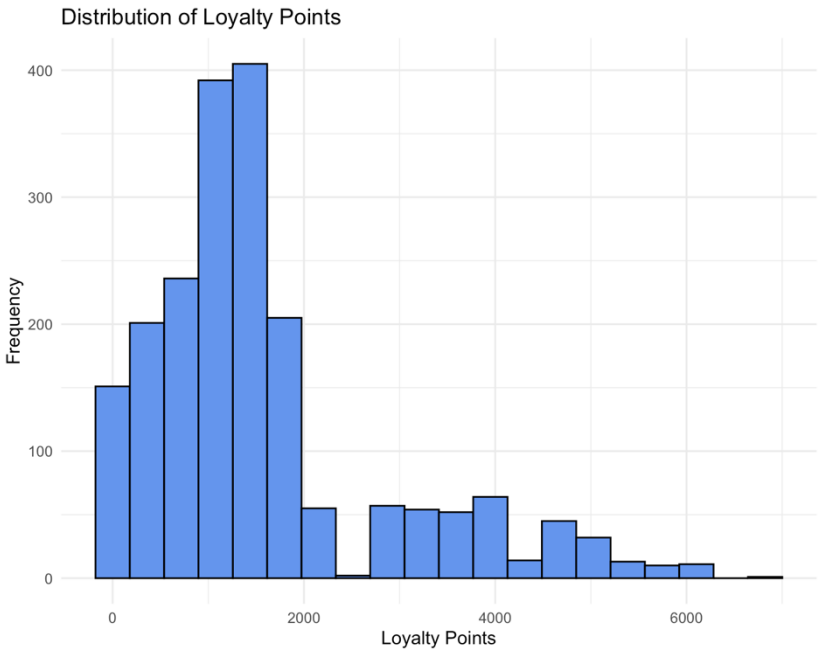
In Python, exploratory data analysis (EDA) assessed the distribution and relationships among numerical variables. In line with Turtle Games' goal of understanding loyalty points accumulation, 'loyalty_points' showed a moderately strong positive correlation with 'spending_score' (0.67) and 'remuneration' (0.61), while 'age' did not appear to have a linear relationship with 'loyalty_points'.

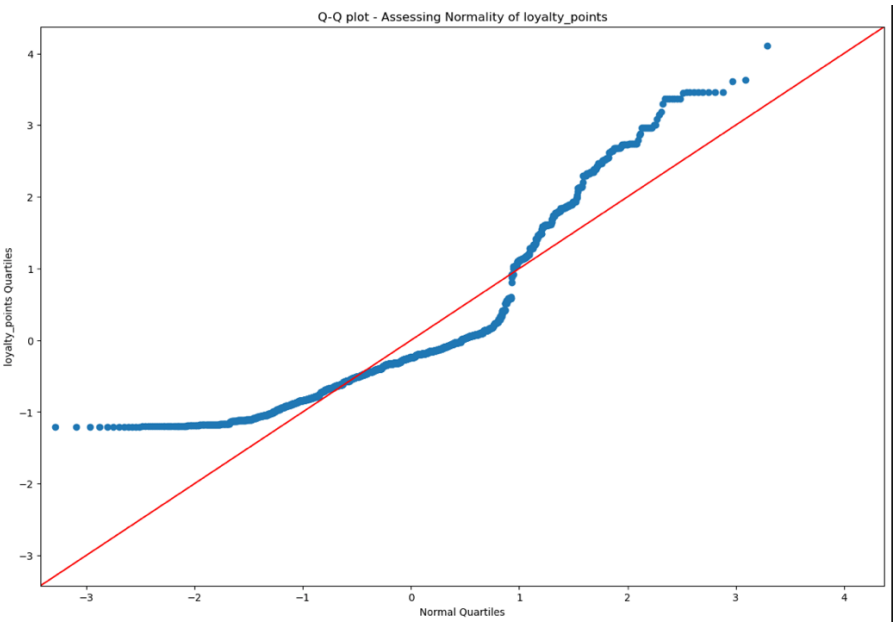


This correlation matrix heatmap was preferred to scatter plots, pair plots or joint plots ad it is space efficient and helps to easily spot multicollinearity and find strong predictors for 'loyalty_points'.

See Further EDA visualisations for more analysis.

Assessing the suitability of 'loyalty_points' for creating predictive models





A histogram helped to quickly visualise the distribution of 'loyalty_points', revealing a clear right skew. While linear regression does not require the dependent variable to be normally distributed, normality of residuals is crucial for valid inference. Although Turtle Games may prefer more of a left-skewed distribution, a right skew is common, as top customers often drive most spending in loyalty programs (Bond Brand Loyalty, 2020).

Also important to note that tree-based models (e.g. Random Forests) do not require normality of the data

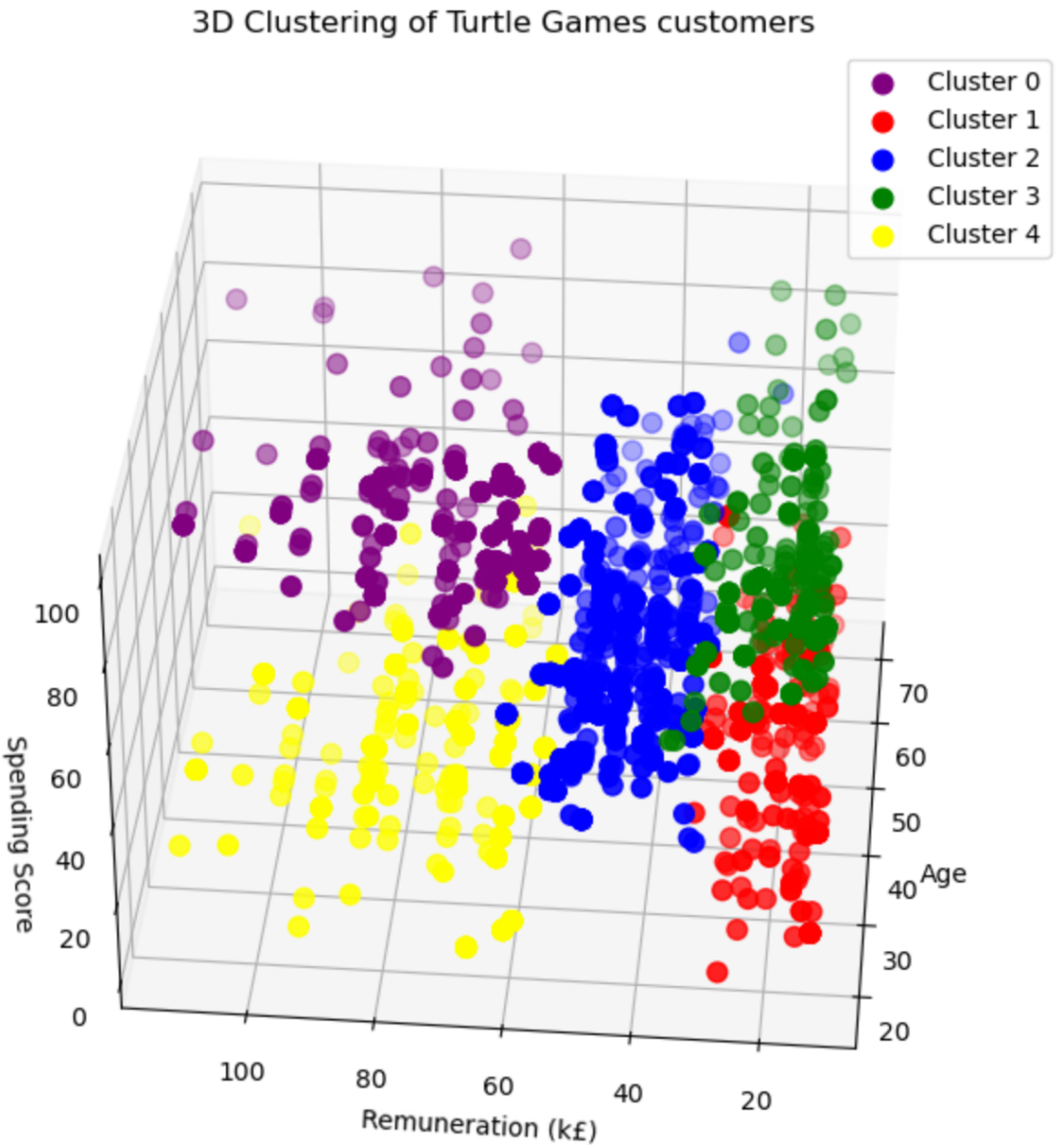
Considering these factors, 'loyalty_points' was deemed appropriate for further statistical testing and machine learning analysis.

Statistical Hypothesis Testing: Focus on specific customer demographic

A t-test in R was used to assess whether mean 'loyalty_points' differed significantly between genders. Due to the results of the t-test, a non-parametric alternative, the Wilcoxon rank-sum test (Mann-Whitney U test, was employed.

Please refer to Statistical Hypothesis Testing for further information.

Customer Segmentation: How can customers be segmented into groups for targeted marketing?

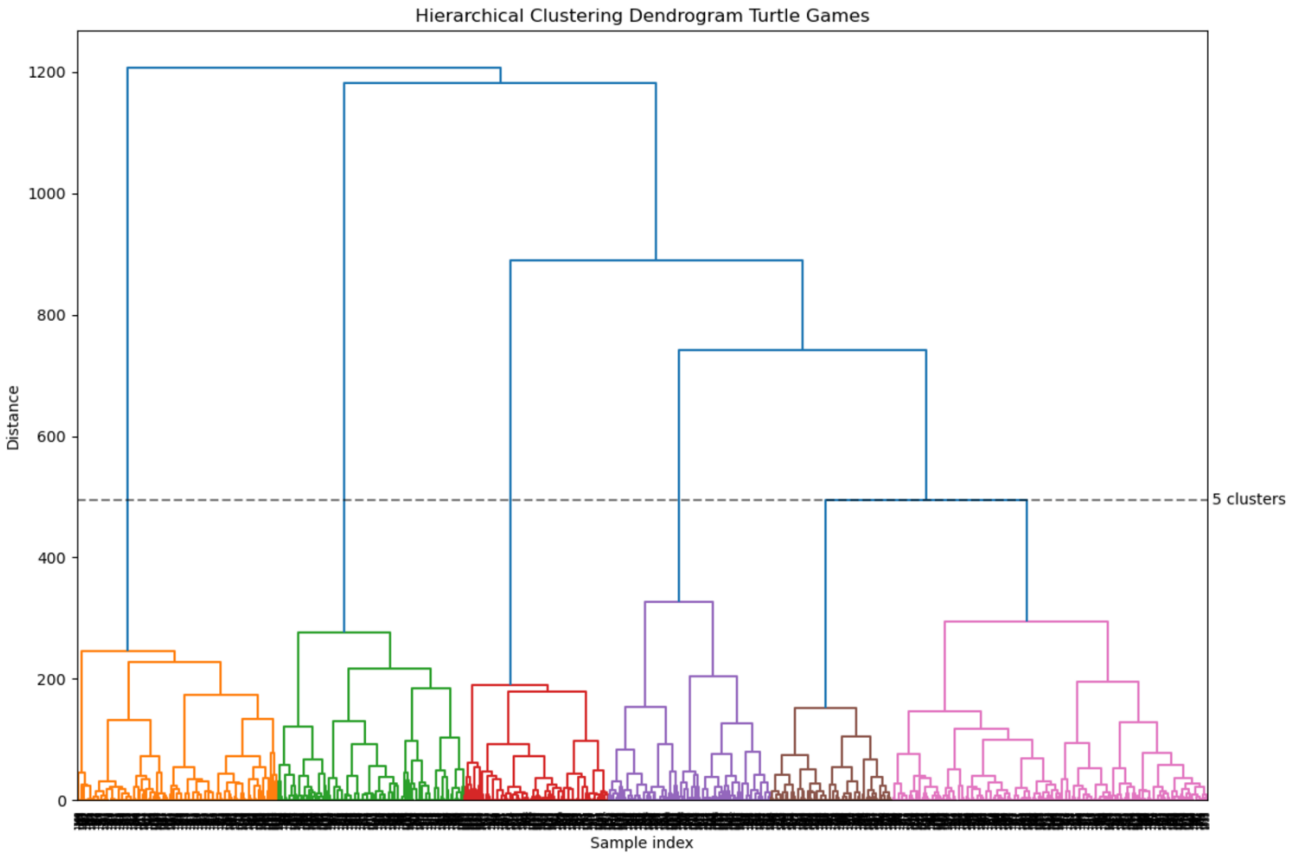


The number of k-means clusters was informed using the Elbow method. K-Means Clustering results were further substantiated through agglomerative hierarchical clustering which came up with similar numbers in five clusters.

```
      age  remuneration  spending_score  label  Hierarchical Predicted
1860   72         24.60               4      1                       1
353    38         63.96              76      0                       4
1333   33         59.04              71      0                       4
905    23         50.84              42      2                       0
1289   49         47.56              46      2                       0
1273   62         41.00              56      2                       0
938    18         60.68              10      4                       2
1731   39         19.68              73      3                       3
65     19         39.36              59      2                       0
1323   39         56.58              91      0                       4
Hierarchical Predicted
0    780
4    356
2    330
1    286
3    248
Name: count, dtype: int64
```

```
df3de_copy['label'].value_counts()
```

```
label
2    776
0    356
4    330
1    270
3    268
Name: count, dtype: int64
```



As a final check, Support Vector Machine Classifier model was used to validate the k-means clustering results and strengthen the case for 5 clusters and the inclusion of third variable ‘age’.

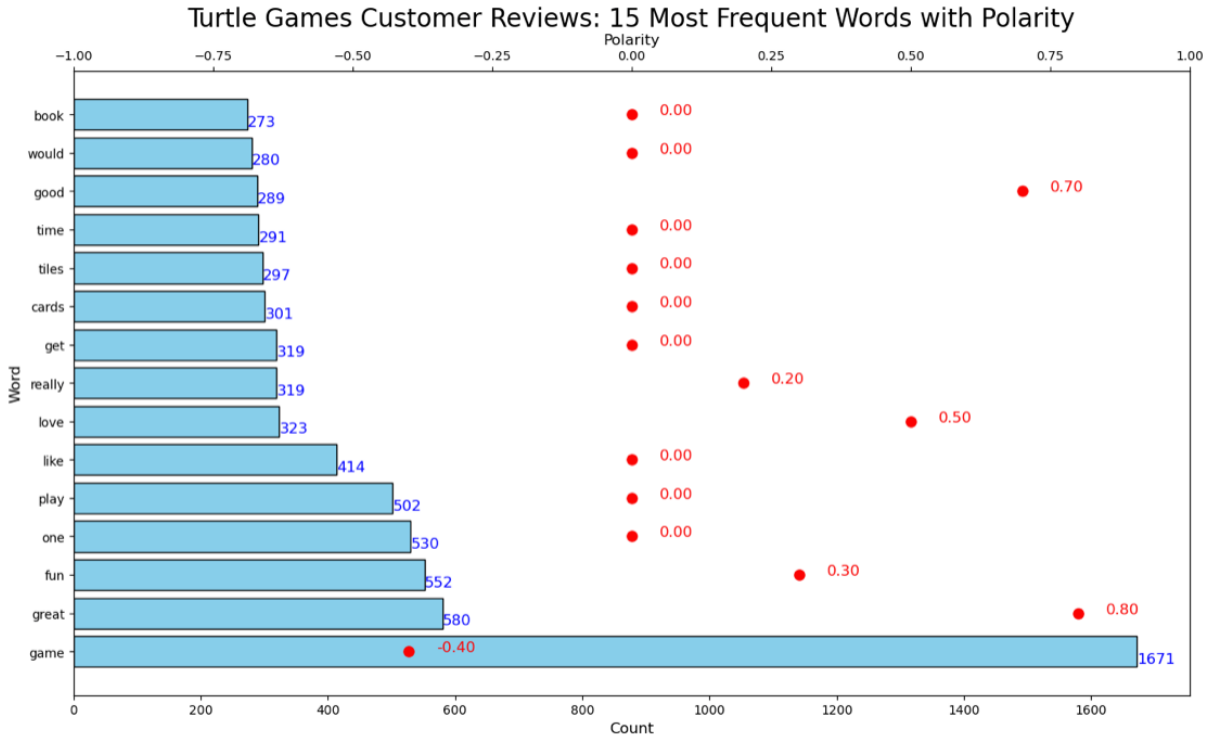
```
Accuracy Score:  0.9825

Classification Report:
              precision    recall  f1-score   support

           0       1.00      1.00      1.00        63
           1       1.00      0.98      0.99        53
           2       0.96      1.00      0.98       153
           3       1.00      0.94      0.97        52
           4       1.00      0.96      0.98        79

    accuracy                           0.98       400
   macro avg       0.99      0.98      0.98       400
weighted avg       0.98      0.98      0.98       400
```

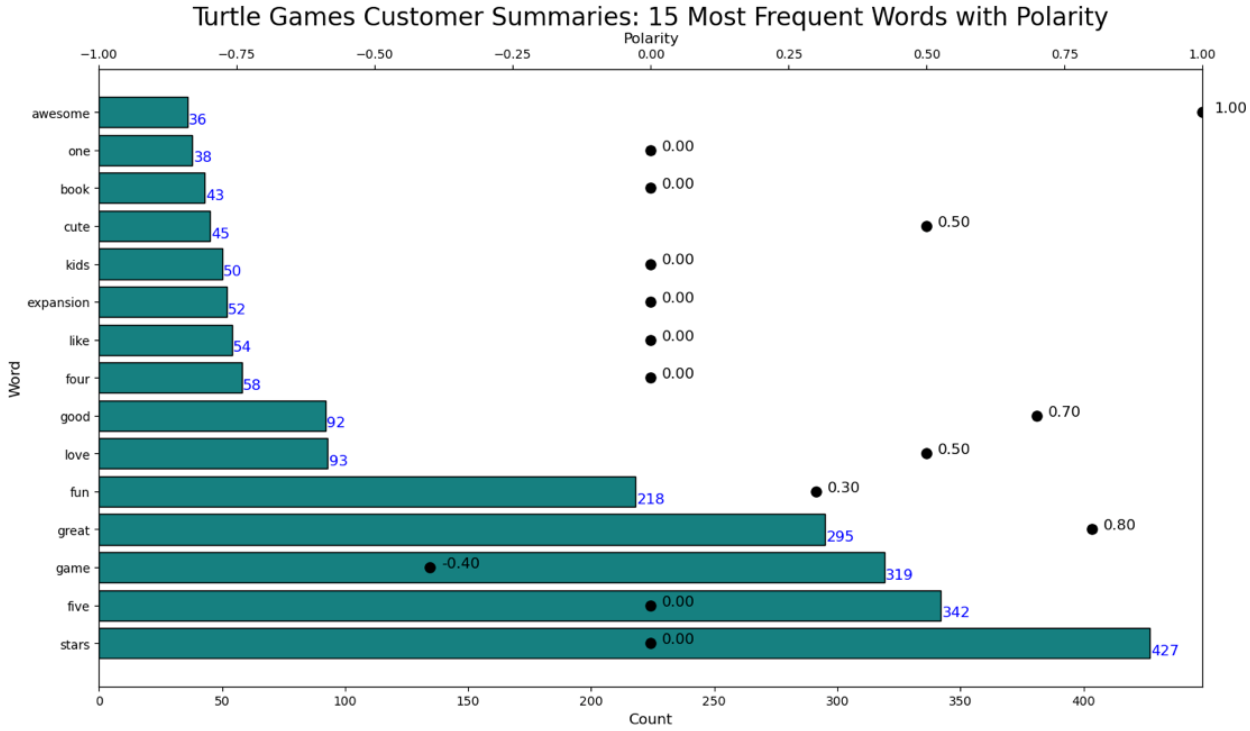
Recommendations and actionable insights:

Cluster 0: premium and exclusive offers, cluster 1: focus on value, cluster 2: mid-range with occasional luxury, cluster 3: community and engagement, cluster 4:convenience.

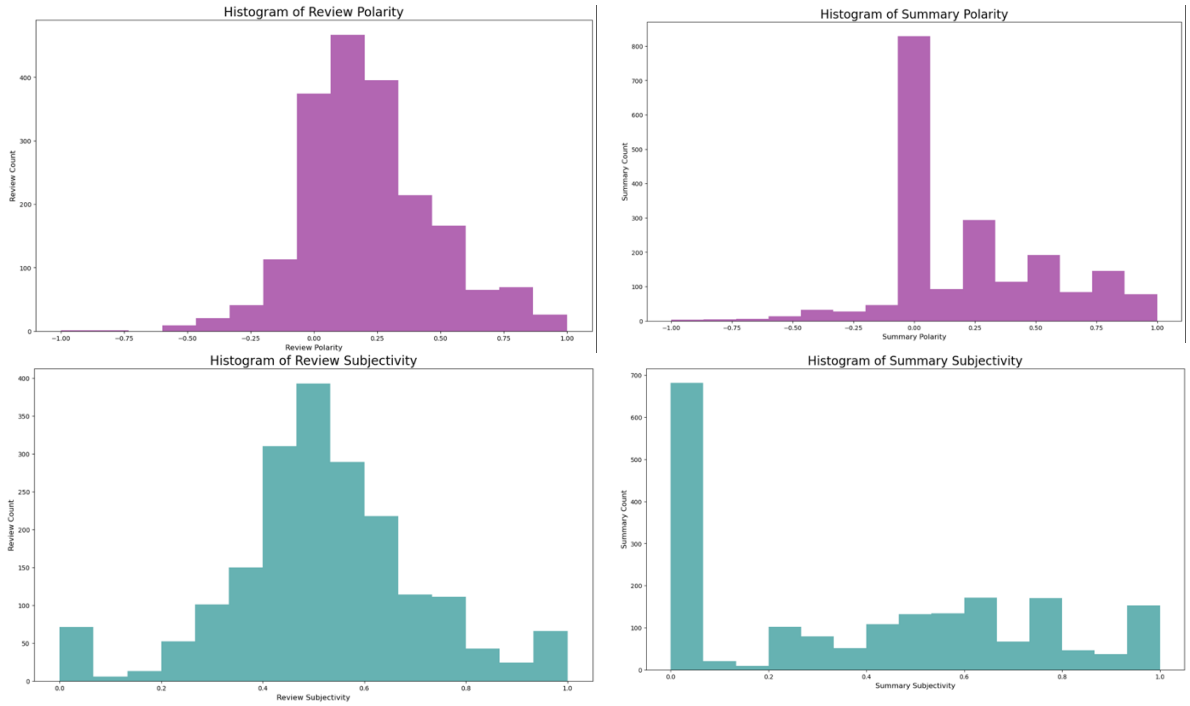
Sentiment Analysis: How can text data inform campaigns and improvements ?

TextBlob - Word Frequency with Polarity Score.





TextBlob - Distribution of Polarity and Subjectivity Scores



TextBlob – Analysis of Top 20 Negative Reviews and Summaries

Top 20 Negative Reviews:

	review	review_polarity
1	booo unles you are patient know how to measure i didnt have the patience neither did my daughter boring unless you are a craft person which i am not	-1.00
2	incomplete kit very disappointing	-0.78
3	im sorry i just find this product to be boring and to be frank juvenile	-0.58
4	one of my staff will be using this game soon so i dont know how well it works as yet but after looking at the cards i believe it will be helpful in getting a conversation started regarding anger and what to do to control it	-0.55
5	expensive for what you get	-0.50
6	i bought this as a christmas gift for my grandson its a sticker book so how can i go wrong with this gift	-0.50
7	i found the directions difficult	-0.50
8	instructions are complicated to follow	-0.50
9	difficult	-0.50
10	this was a gift for my daughter i found it difficult to use	-0.50
11	i sent this product to my granddaughter the pompom maker comes in two parts and is supposed to snap together to create the pompoms however both parts were the same making it unusable if you cant make the pompoms the kit is useless since this was sent as a gift i do not have it to return very disappointed	-0.49
12	my 8 yearold granddaughter and i were very frustrated and discouraged attempting this craft it is definitely not for a young child i too had difficulty understanding the directions we were very disappointed	-0.45
13	i purchased this on the recommendation of two therapists working with my adopted children the children found it boring and put it down half way through	-0.44
14	very hard complicated to make these	-0.44
15	was a gift for my son he loves the game	-0.40
16	this game is a blast	-0.40
17	if you like me used to play dd but now you and your friends growed up and cant be together because all the responsibilities and bla bla bla this game is for you come to the dungeon	-0.40
18	jun game	-0.40
19	you can play the expansions one at a time or add then both in for a longer game if your into lords of waterdeep this is a must have	-0.40
20	i bought this for my son he loves this game	-0.40

Top 20 Negative Summaries:

	summary	summary_polarity
1	boring unless you are a craft person which i am	-1.00
2	the worst value ive ever seen	-1.00
3	boring	-1.00
4	before this i hated running any rpg campaign dealing with towns because it	-0.90
5	another worthless dungeon masters screen from galeforce9	-0.80
6	disappointed	-0.75
7	disappointed	-0.75
8	disappointed	-0.75
9	disappointed	-0.75
10	promotes anger instead of teaching calming methods	-0.70
11	too bad this is not what i was expecting	-0.70
12	bad qualityall made of paper	-0.70
13	at age 31 i found these very difficult to make	-0.65
14	mad dragon	-0.62
15	small and boring	-0.62
16	disappointing	-0.60
17	disappointing	-0.60
18	disappointing	-0.60
19	disappointing	-0.60
20	then you will find this board game to be dumb and boring	-0.59

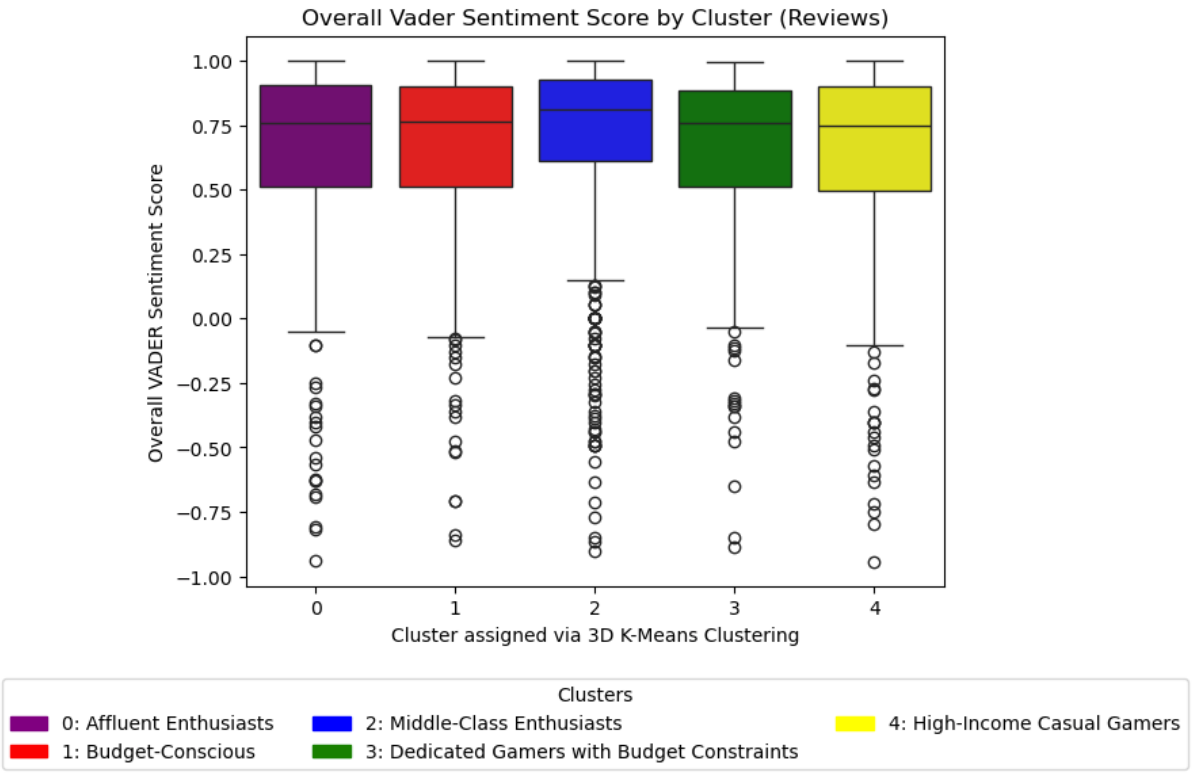
Potential Mislabelling Issues:
Difficulty level is a recurring complaint. Many customers described certain products as "boring," suggesting they may not match the advertised challenge level. Age appropriateness, particularly in terms of complexity, was also frequently questioned.

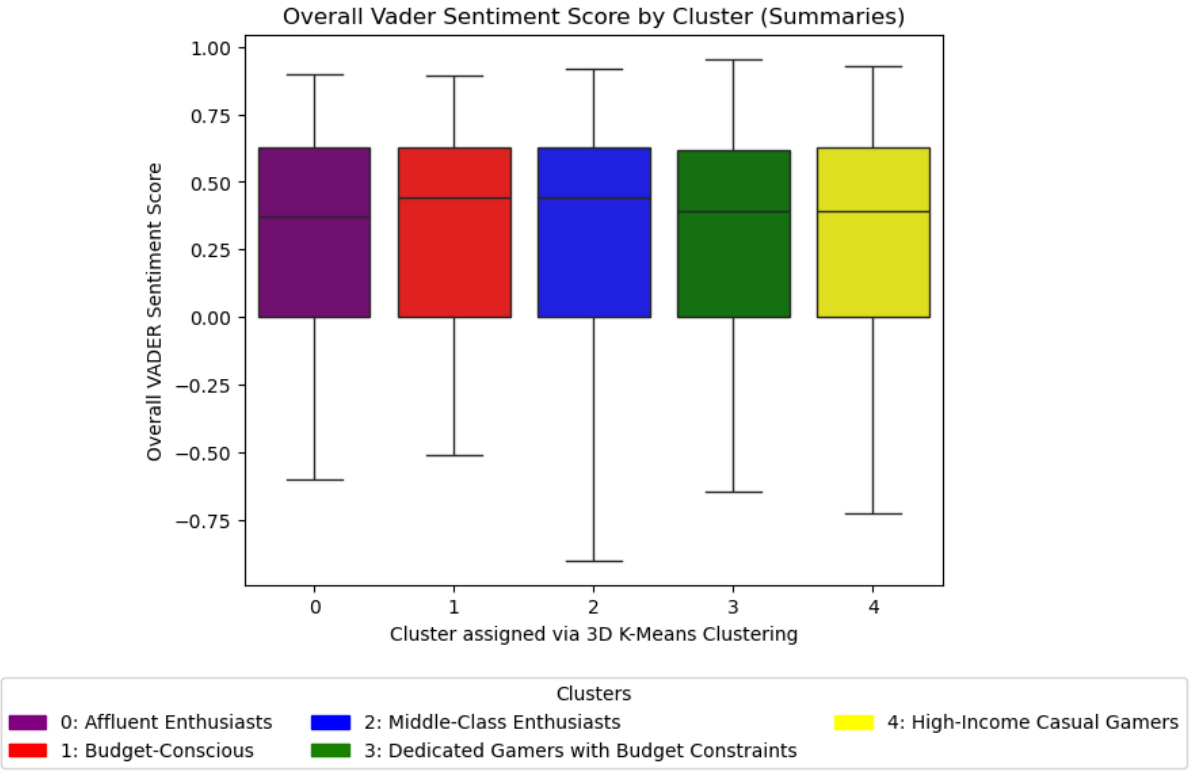
Potential Quality Control Issues:
Several reviews mentioned problems such as incomplete kits and poor product quality.

Mispricing Concerns:
Some users felt the products were overpriced relative to the value offered, with comments like "expensive for what you get."

TextBlob Sentiment Analysis Limitations:
A few reviews (e.g., items 15, 16, 19, 20) received negative polarity scores despite containing positive language, such as "loves the game" and "this game is a blast," indicating possible misclassification by the sentiment analysis tool.

Overall VADER Sentiment Score by Cluster (Reviews and Summaries)





Please see Analysis Overall VADER Sentiment Score by Cluster.

Predictive Modelling

Multiple Linear Regression modelfe for predicting 'loyalty_points'

In R, modelfe incorporated a feature-engineered variable, 'potential_value', calculated as the product of 'spending_score' and 'remuneration'. After extensive experimentation, the combination of 'potential_value' and 'spending_score' proved to be the most effective for predicting 'loyalty_points'. This approach captures both the direct effect of 'spending_score' and its interaction with 'remuneration'. 'Potential_value' can thus be interpreted as a proxy variable for the combined influence of these two predictors on customer loyalty.

Importantly, the model generated only non-negative predictions (on test scenarios), aligning with business expectations and providing more interpretable results than the log-transformed model. On the training data, modelfe achieved an adjusted R-squared of 0.9755, with all coefficients and the overall model statistically significant. Test set performance metrics were also excellent:

Recommendations:

- **Target High Spending Score Customers:**
  'spending_score' emerged as the strongest predictor of loyalty_points. Marketing strategies should prioritize customers with high spending scores, as they consistently generate higher loyalty rewards and are likely more engaged.
- **Integrate modelfe in loyalty prediction dashboard.**
- **Leverage the 'potential_value' Metric:**
  The feature-engineered variable 'potential_value' (calculated as *spending_score* × *remuneration*) proved to be a valuable predictor. Customers with high potential_value scores should be prioritized in targeted campaigns, as they represent a high-return segment.
- **Segment Marketing by Income Levels:**
  Although 'remuneration' is less influential than 'spending_score', it still plays a significant role in predicting loyalty. Tailored marketing strategies should be developed for different income segments, in alignment with the K-Means clustering analysis that identified five distinct customer groups.
- **Engage High-Value Outliers Strategically:**
  Retaining outliers slightly worsened the overall regression model fit, but closer inspection suggests that the 266 high-value customers are meaningful outliers, likely representing premium segments. These customers should not be

- ignored—consider designing exclusive loyalty tiers or personalized benefits to encourage retention and further engagement. Also see Outlier detection.

```
> # View the model summary
> summary(modelfe)

Call:
lm(formula = loyalty_points ~ spending_score + potential_value,
    data = df2)

Residuals:
     Min       1Q   Median       3Q      Max
-1766.50   -62.24    19.73   119.91   615.50

Coefficients:
                 Estimate Std. Error t value Pr(>|t|)
(Intercept)     23.638451   9.394713   2.516   0.0119 *
spending_score  -1.763447   0.232637  -7.580 5.25e-14 ***
potential_value  0.682316   0.003186 214.184  < 2e-16 ***
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

Residual standard error: 194.1 on 1997 degrees of freedom
Multiple R-squared:  0.9771,    Adjusted R-squared:  0.9771
F-statistic: 4.268e+04 on 2 and 1997 DF,  p-value: < 2.2e-16
```

For a comprehensive overview of the regression modelling performed in this analysis, please refer to Comprehensive overview of regression modelling.

Random Forest rfg2 as the overall superior model

Below is a comparison of rfg2 with the best-performing Random Forest model implemented in R.

Based on the metrics provided, Random Forest rfg2 is the better model:

Random Forest rfg2 (Python):

Test RMSE: 68.13

Test MSE: 4641.22

Test R^2: 0.997

Test MAE: 22.00

Number of trees: 100

rf_model1(R):

Test RMSE: 120.95

MSR (Mean of squared residuals): 12980.71

% Var explained: 99.21 (equivalent to R^2 of 0.9921)

MAPE: 10.23%

Number of trees: 500

Despite rf_model1 using 5 times more trees (500 vs 100), Random Forest rfg2 has a significantly lower RMSE on the test data (68.13 vs 120.95), which indicates better predictive performance. The lower test MSE of rfg2 (4641.22) compared to rf_model1's mean squared residuals (12980.71) further confirms this conclusion.

Additionally, rfg2 has a slightly higher R^2 value (0.997 vs 0.992), indicating it explains more variance in the target variable.

Recommendation:

- Deploy the Random Forest model (rfg2) for forecasting. It is preferred over modelfe due to ongoing issues with non-normality of residuals and heteroskedasticity in the latter. Tree-based models like Random Forest do not rely on the assumption of normally distributed residuals, making rfg2 a more robust and reliable choice for prediction in this context.

- Possibly incorporate ‘potential_value’ in the future iterations of rfg2.
- Evaluate and optimize rfg2 to enhance its performance on new data.

```
]: # Create and train the Random Forest regressor
rfg2 = RandomForestRegressor(random_state=42)
rfg2.fit(X_train_reduced, y_train)

# Predictions
y_train_pred_rfg2 = rfg2.predict(X_train_reduced)
y_test_pred_rfg2 = rfg2.predict(X_test_reduced)

## Calculate Training and Test Errors
train_mse = mean_squared_error(y_train, y_train_pred_rfg2)
test_mse = mean_squared_error(y_test, y_test_pred_rfg2)
train_rmse = np.sqrt(train_mse)
test_rmse = np.sqrt(test_mse)
train_r2 = r2_score(y_train, y_train_pred_rfg2)
test_r2 = r2_score(y_test, y_test_pred_rfg2)
train_mae = mean_absolute_error(y_train, y_train_pred_rfg2)
test_mae = mean_absolute_error(y_test, y_test_pred_rfg2)

print(f"Random Forest rfg2 - Train MSE: {train_mse}, Train RMSE: {train_rmse}, Train R²: {train_r2}, Train MAE: {train_mae}")
print(f"Unpruned Random Forest rfg2 - Test MSE: {test_mse}, Test RMSE: {test_rmse}, Test R²: {test_r2}, Test MAE: {test_mae}")

n_trees = rfg.n_estimators
print(f"Number of trees in Random Forest rfg2: {n_trees}")

Random Forest rfg2 - Train MSE: 788.6900840714285, Train RMSE: 28.083626618929195, Train R²: 0.9995239954883456, Train MAE: 7.824892857142854
Unpruned Random Forest rfg2 - Test MSE: 4641.222773833331, Test RMSE: 68.12652034144509, Test R²: 0.997134688529027, Test MAE: 22.00078333333
333
Number of trees in Random Forest rfg2: 100

]: # Make predictions on new data
new_customer = pd.DataFrame({'remuneration': [19.68],
                             'spending_score': [73],
                             'age': [27]})

predicted_points = rfg2.predict(new_customer)
print(f"Predicted Loyalty Points for new customer: {predicted_points[0]:.2f}")

Predicted Loyalty Points for new customer: 840.00
```

For a detailed overview of the tree-based models explored in this analysis, please refer to Detailed overview of tree-based models.

Appendices

Appendix A

Detailed description of turtle_reviews.csv

The dataset, **turtle_reviews.csv**, contains the following columns with interpretations and associated limitations:

1. **Gender**
    - **Interpretation**: Indicates the gender of the customer providing the review.
    - **Limitations**: Only includes "male" and "female" categories, which excludes non-binary or other gender identities.
2. **Age**
    - **Interpretation**: Represents the customer's age in years.
3. **Remuneration**
    - **Interpretation**: The customer's annual income, expressed in thousands of pounds.
    - **Limitations**: Since remuneration may be self-reported or estimated, inaccuracies could arise.
4. **Spending Score**
    - **Interpretation**: A score, ranging from 1 to 100, that reflects the customer's purchasing behaviour.
    - **Limitations**: The methodology behind the score calculation is not specified, so understanding the underlying algorithm or rules is necessary for accurate interpretation.
5. **Loyalty Points**
    - **Interpretation**: A score reflecting the customer's purchase points, which correlates with the value of their purchases and interactions.
    - **Limitations**: The conversion from monetary value to loyalty points is unclear, making it difficult to fully interpret the significance of the points.
6. **Education**
    - **Interpretation**: Represents the customer's highest level of education, with categories such as diploma, graduate, postgraduate, and PhD.
    - **Limitations**: Education is grouped into broad categories, which may miss nuances such as specific fields of study or individual educational experiences. Self-reporting may also lead to inaccuracies due to exaggeration or understatement.
7. **Language**
    - **Interpretation**: All reviews are submitted in English (EN).
    - **Limitations**: The dataset is limited to English-language reviews, which may not capture the preferences or behaviours of non-English-speaking customers. The data may also be geographically biased towards English-speaking regions.

8. **Platform**
    - **Interpretation**: Identifies the platform used by the customer to submit their review (in this case, the Turtle Games website).
    - **Limitations**: The dataset only includes reviews from the website, excluding customers who may use mobile apps or in-store systems to interact with Turtle Games.
9. **Product**
    - **Interpretation**: A unique code assigned to each product.
    - **Limitations**: Product codes are provided without descriptions, making it difficult to associate customer feedback with specific products and to ensure the uniqueness of product codes.
10. **Review**
    - **Interpretation**: The full text of the review submitted by the customer.
    - **Limitations**: Reviews vary greatly in length and detail. Some may be brief or vague, while others are more detailed, making consistent analysis challenging. As customer reviews are inherently subjective, sentiment analysis may be influenced by individual interpretation.
11. **Summary**
    - **Interpretation**: A brief summary highlighting the key points of the customer's review.
    - **Limitations**: The quality and accuracy of the summaries depend on the extraction method used. If these summaries are AI-generated, they may not fully capture all relevant aspects of the review, and their reliability is uncertain.

Libraries utilised in Python and R

In addition to the standard core data and visualization libraries—such as Pandas, NumPy, Matplotlib, and Seaborn—several other libraries were employed to support this analysis. These included SciPy and Statsmodels for statistical analysis, with Statsmodels also used for machine learning tasks and SciPy for hierarchical clustering. Scikit-learn was utilized for broader machine learning applications, and TextBlob and NLTK were used for natural language processing.

```
# Statistical Analysis
import scipy.stats as stats
from scipy.stats import shapiro, norm
from scipy.spatial.distance import cdist
import statsmodels.api as sm
import statsmodels.formula.api as smf
import statsmodels.stats.api as sms
from statsmodels.formula.api import ols
from statsmodels.stats.outliers_influence import variance_inflation_factor
from statsmodels.graphics.gofplots import qqplot

# Machine Learning
from sklearn.linear_model import LinearRegression
from sklearn.model_selection import train_test_split
from sklearn import metrics
from sklearn.metrics import (
    mean_squared_error, mean_absolute_error, r2_score,
    accuracy_score, precision_score, recall_score, f1_score,
    confusion_matrix, ConfusionMatrixDisplay, classification_report,
    silhouette_score
)
from sklearn.tree import DecisionTreeRegressor, plot_tree
from sklearn.preprocessing import LabelEncoder, StandardScaler
from sklearn.utils.validation import column_or_1d
from sklearn.ensemble import RandomForestRegressor
from sklearn.cluster import KMeans, AgglomerativeClustering
from sklearn.svm import SVC

# Natural Language Processing
import nltk
import contractions
from nltk import word_tokenize, pos_tag
from nltk.corpus import stopwords, wordnet as wn
from nltk.tokenize import word_tokenize
from nltk.stem import WordNetLemmatizer, PorterStemmer
from nltk.probability import FreqDist
from vaderSentiment.vaderSentiment import SentimentIntensityAnalyzer
from wordcloud import WordCloud
from textblob import TextBlob

# Hierarchical Clustering
from scipy.cluster.hierarchy import linkage, dendrogram, fcluster

# NLTK Downloads (uncomment if needed)
# nltk.download('punkt')
# nltk.download('stopwords')
nltk.download('averaged_perceptron_tagger_eng')
```

Whilst the analysis conducted in R was less extensive than on Python, the R libraries represent a comprehensive toolkit for statistical analysis, from data preparation to model building and evaluation.

```
# libraries
library(dplyr) # Data manipulation
library(ggplot2) # Data visualisation
library(moments) # Skewness and kurtosis
library(corrplot) # Correlation visualisation
library(car) # VIF calculation
library(lmtest) # Statistical tests
library(caret) # Classification and REgression Training
library(randomForest) # Random Forest modeling
```

- **dplyr**: A popular data manipulation package that provides functions for filtering, selecting, mutating, and summarizing data.
- **ggplot2**: A powerful data visualisation package based on the grammar of graphics.
- **moments**: Used for calculating statistical moments such as skewness and kurtosis.
- **corrplot**: Specialized in visualization of correlation matrices.
- **car**: Provides functions for Variance Inflation Factor (VIF) calculation, useful for multicollinearity detection.
- **lmtest**: Contains statistical tests for linear models.
- **caret**: A comprehensive framework for Classification and Regression Training.
- **randomForest**: Implements the Random Forest algorithm for classification and regression modelling

Rationale behind choice of analytical methods

Machine Learning models (supervised):

- Linear Regression (Python & R)
- Decision Tree Regressor (Python)
- Random Forest Regressor (Python & R)
- Support Vector Machine classifier (Python)

Machine Learning models (unsupervised):

- K-Means Clustering (Python)
- Hierarchical Clustering (Python)

Natural Language Processing techniques:

- TextBlob (Python)
- Valence Aware Dictionary and sEntiment Reasoner (Python)

Statistical Hypothesis Tests:

- Student's t-test (R)
- Wilcoxon Rank-Sum test (Mann-Whitney U Test) (R)

Linear regression models were chosen for their relative simplicity and interpretability. They are fast and efficient to conduct in both Python and R. As 'loyalty_points' showed a moderately strong linear relationship with 'remuneration' and 'spending_score', linear regression was a natural choice. The fitted linear regression models act as baseline models for comparing more complex models.

The tree-based models were implemented to potentially capture non-linear relationships between 'loyalty_points' and all independent variables. Tree split data is based on thresholds and as such, tree-based models are less sensitive to outliers and do not assume normality and homoscedasticity. This is particular important given points raised here Assumptions and Limitations. In fact, random forest rfg2 was the superior model of all the machine learning models utilised for the purpose of regression. The use of linear regression and tree-based models are in line with Turtle Games' request to use data for predictive modelling through statistical analysis and to investigate loyalty points accumulation.

In terms of unsupervised machine learning models, Hierarchical Clustering (with dendrograms) and Support Vector Machine classifier were used to substantiate the results of the 2D K-Means clusters (using 'remuneration' and 'spending_score'). However, a decision was made to introduce 'age' as a third variable. 'age' is potentially of importance given that Turtle Games manufacturers and sells games. The unsupervised machine learning models create meaningful customer segments for targeted marketing, as was requested by Turtle Games.

The steps undertaken when conducting sentiment analysis using VADER, represented a more sophisticated approach to when using TextBlob (e.g. lemmatisation was

implemented with VADER), the overall VADER Sentiment Score was used to investigate potential gender differences and differences in the aforementioned 3D K-Means clusters. Turtle Games wished to extract actionable insights from customer reviews and this work addresses this brief.

On R, statistical hypothesis tests were conducted to establish whether the gender difference in mean 'loyalty_points' was statistically significant. Because the data violated the assumption of normality, the non-parametric test alternative to the t-test, Mann-Whitney U test was implemented.

Assumptions , limitations and concerns

- The right-skewed distribution of ‘loyalty_points’, raised concerns about heteroskedasticity, as it may lead to inefficient coefficient estimates, incorrect standard errors, invalid confidence intervals, poor model fit and reduced predictive accuracy.

- Unfortunately, all linear regression models, except for model4, still displayed signs of heteroskedasticity, and the residuals were not normally distributed. This remained the case even after applying the natural logarithm transformation to 'loyalty_points' (in Python) and incorporating a new variable, 'potential_value,' through feature engineering (in R).

- After reviewing the range of all numerical variables, standardisation and normalisation were deemed unnecessary for the purpose of this analysis.

- During the initial Python analysis, after fitting model1 (loyalty_points ~ spending_score), a decision was made to apply a natural log transformation to ‘loyalty_points’ for the following reasons:

  (i) ‘loyalty_points’ was not normally distributed (right-skewed distribution),
  (ii) the residuals of model1 were also not normally distributed, and
  (iii) the Breusch-Pagan test indicated clear evidence of heteroscedasticity.

  The log transformation improved the normality of the distribution (Shapiro-Wilk statistic improved from 0.84 to ~0.90, but still a significant deviation from normality p-value (~1.23e-33)) and helped reduce heteroscedasticity.

- The R analysis did not include the log-transformed version of ‘loyalty_points’.

- In R, a new variable, ‘potential_value’, was feature-engineered by multiplying ‘spending_score’ with ‘remuneration’. This metric was intended to represent the potential value of a given customer.

- The methodology behind the calculation of spending_score is not specified.

- The conversion from monetary value to loyalty points is unclear.

- In preparation for the TextBlob sentiment analysis, 39 duplicates were removed from the ‘review’ and ‘summary’ columns, while all duplicates were retained for

the VADER sentiment analysis. This preprocessing difference should be considered when comparing TextBlob Polarity and VADER Compound scores.

- Some customer reviews associated with the same ‘product’ code appeared to reference different products. Due to the lack of product names and descriptions to confirm the uniqueness of this variable, it was excluded from the analysis.

- Prior to fitting the Decision Tree Regressor and Random Forest Regressor models, the 'Basic' and 'Diploma' categories in the ‘education’ column of df2 were combined into a single category labelled 'Low', and the column was then encoded using *LabelEncoder*. Since ‘education’ was excluded when fitting MLR models, the comparability between the two approaches (MLR vs. tree-based models) is compromised where the tree-based models included ‘education’.

- ‘education’ was not included in any subsequent analyses. Given the nature of Turtle Games' business, ‘remuneration’ was considered a sufficient proxy for educational level. Additionally, when fitting tree-based models, ‘education’ ranked poorly in terms of feature importance.

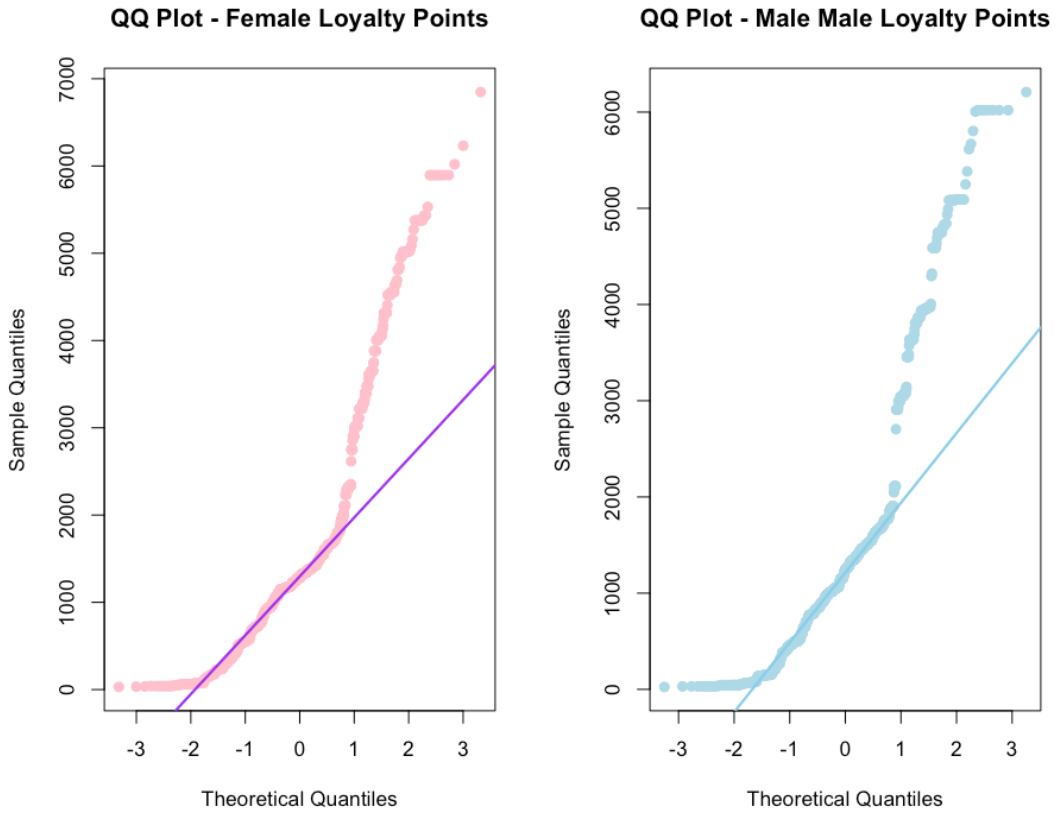
Appendix B

Statistical Hypothesis Testing

```
> cat("Loyalty points by gender:\n")
Loyalty points by gender:

> print(group_by_gender)
# A tibble: 2 × 4
  gender mean_loyalty median_loyalty count
  <fct>         <dbl>          <dbl> <int>
1 Female        1601.           1281  1120
2 Male          1549.           1248   880
```

Whilst the raw mean difference of 52.58 between the two genders was not found to be statistically significant ($p > 0.05$), the data violated the normality assumptions.

```
> # Perform the t-test
> t_test_result <- t.test(female_loyalty, male_loyalty)

> print(t_test_result)

        Welch Two Sample t-test

data:  female_loyalty and male_loyalty
t = 0.90348, df = 1836.2, p-value = 0.3664
alternative hypothesis: true difference in means is not equal to 0
95 percent confidence interval:
 -61.55895 166.71788
sample estimates:
mean of x mean of y
 1601.167  1548.588
```



Therefore, a non-parametric alternative (the Mann-Whitney U test) was conducted to confirm the results.

```
        Wilcoxon rank sum test with continuity correction

data:  female_loyalty and male_loyalty
W = 521198, p-value = 0.02675
alternative hypothesis: true location shift is not equal to 0


> # Unlike the t-test, the Wilcoxon test indicates there is a statistically
> # significant difference in the distribution of loyalty points between  .... [TRUNCATED]
```

Unlike the t-test, it showed a significant difference in the distribution of loyalty points between female and male customers (p = 0.02675). The result of Wilcoxon test is favoured as:

- loyalty_points data significantly deviated from normal distribution as per Shapiro-Wilk test.
- Wilcoxon test compares median distributions rather than mean (less sensitive to outliers).
- Female customers tend to have higher loyalty point values than male customers, and this difference is statistically significant (p = 0.02675).

Although the effect size of the difference is small, Cohen's d effect size < 0.2

```
> # Calculate effect size (Cohen's d)
> pooled_sd <- sqrt(((female_count-1)*female_sd^2 + (male_count-1)*male_sd^2) /
+                     (female_c .... [TRUNCATED]

> cohen_d <- abs(female_mean - male_mean) / pooled_sd

> cat("Cohen's d effect size:", cohen_d, "\n")
Cohen's d effect size: 0.04097223

> # The difference in loyalty points between women and men is extremely small (<0.2).
>
```

Recommendations:

- female-oriented campaigns to acknowledge their consistent participation
- male-oriented campaigns to boost their participation.
- A/B test different messaging styles & visuals based on gender (even if core offers similar).
- Avoid overly different approaches given extremely small effect size.

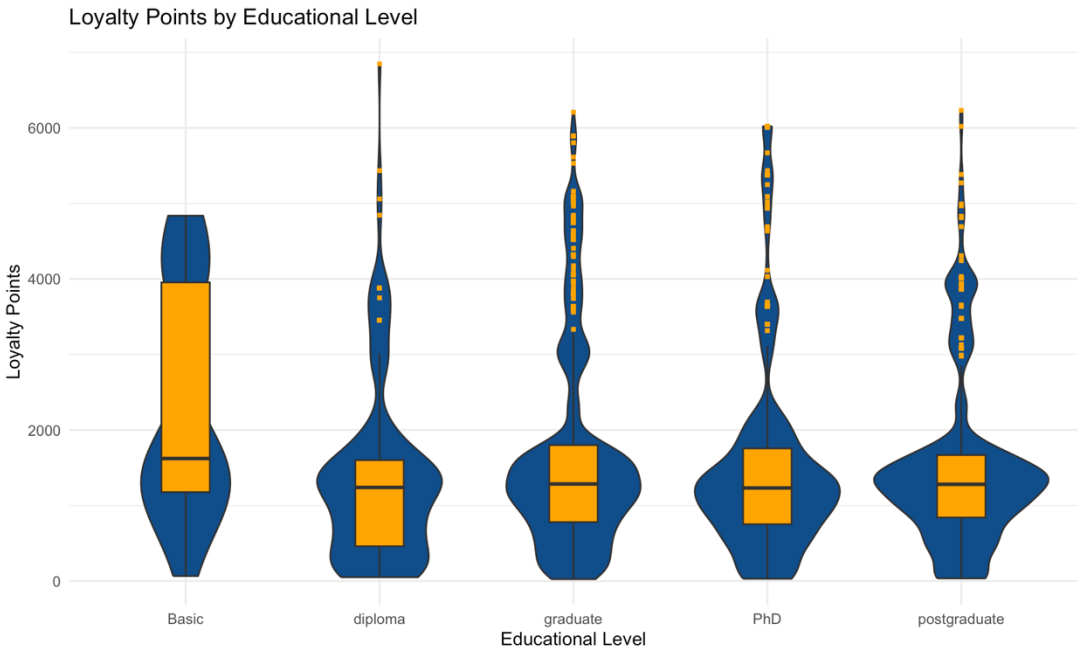
Limitations:

This gender-based analysis could introduce bias, as it does not capture other demographic or psychographic factors that influence purchasing behaviour.

Further EDA visualisations

Using R, the EDA also examined the frequency distribution of categorical variables, group-level 'loyalty_points' statistics (by educational level), detailed measures of shape and outlier detection for 'loyalty_points'.

```
> ####################################################
>
> # 5.2a Group statistics
>
> group_by_gender <- df %>%
+   gro .... [TRUNCATED]

> group_by_education <- df %>%
+   group_by(education) %>%
+   summarize(
+     mean_loyalty = mean(loyalty_points),
+     median_loyalty = median(loy .... [TRUNCATED]

> cat("Loyalty points by gender:\n")
Loyalty points by gender:

> print(group_by_gender)
# A tibble: 2 × 4
  gender mean_loyalty median_loyalty count
  <fct>         <dbl>          <dbl> <int>
1 Female        1601.           1281  1120
2 Male          1549.           1248   880

> cat("\nLoyalty points by education:\n")

Loyalty points by education:

> print(group_by_education)
# A tibble: 5 × 4
  education    mean_loyalty median_loyalty count
  <fct>               <dbl>          <dbl> <int>
1 Basic               2265.           1622    50
2 diploma             1336.           1239   190
3 graduate            1666.           1285   900
4 PhD                 1500.           1232   460
5 postgraduate        1499.           1281   400

> ####################################################
```



The violin plot was selected for its ability to illustrate both the distribution and density of 'loyalty_points' across education levels. Customers with a 'Basic' education show the highest mean 'loyalty_points', possibly due to having more time to engage with the platform and accumulate points.

Interestingly, customers with a 'Diploma' have the lowest mean 'loyalty_points', despite expectations they might behave similarly to the 'Basic' group. In contrast, those with higher education levels show more compact distributions with a greater number of outliers—perhaps indicating less frequent purchases of higher-value items.

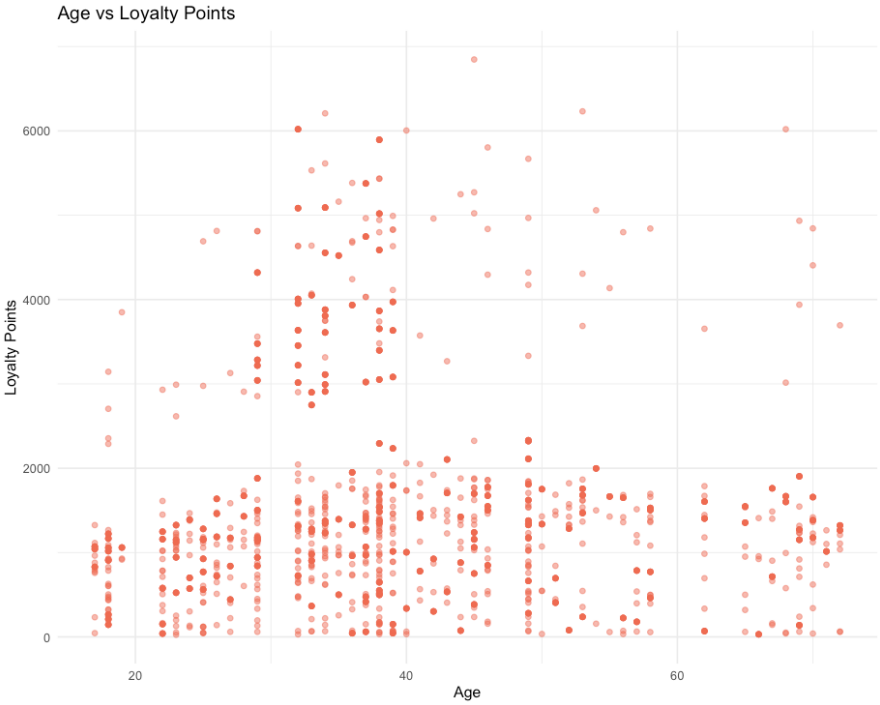
Recommendation: Turtle Games could explore strategies to boost engagement and 'loyalty_points' accumulation among Diploma-holders. Turtle Games could also devise initiatives to target the outliers with the higher educational level e.g. by offering premium and exclusive product offerings.

```
- # 5.2c Measures of Shape
-
- # Shapiro-Wilk test for normalit .... [TRUNCATED]

        Shapiro-Wilk normality test

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


- # Skewness and Kurtosis
- skewness(df$loyalty_points)
[1] 1.463694

- kurtosis(df$loyalty_points)
[1] 4.70883



> # Display results
> list(
+   Range = range_loyalty_points,
+   Difference = difference_high_low,
+   IQR = iqr_loyalty_points,
+   Variance = varia .... [TRUNCATED]
$Range
[1]   25 6847

$Difference
[1] 6822

$IQR
[1] 979.25

$Variance
[1] 1646704

$Standard_Deviation
[1] 1283.24



> # 5.2d More Measures of Shape
>
> scores <- df$loyalty_points

> # Calculate mean, median, and mode
> mean_score <- mean(scores)

> median_score <- median(scores)

> mode_score <- as.numeric(names(sort(table(scores), decreasing = TRUE)[1]))

> # Print the results
> cat("Mean:", mean_score, "\n")
Mean: 1578.032

> cat("Median:", median_score, "\n")
Median: 1276

> cat("Mode:", mode_score, "\n")
Mode: 1014
```

Interpretation of Measures of Shape

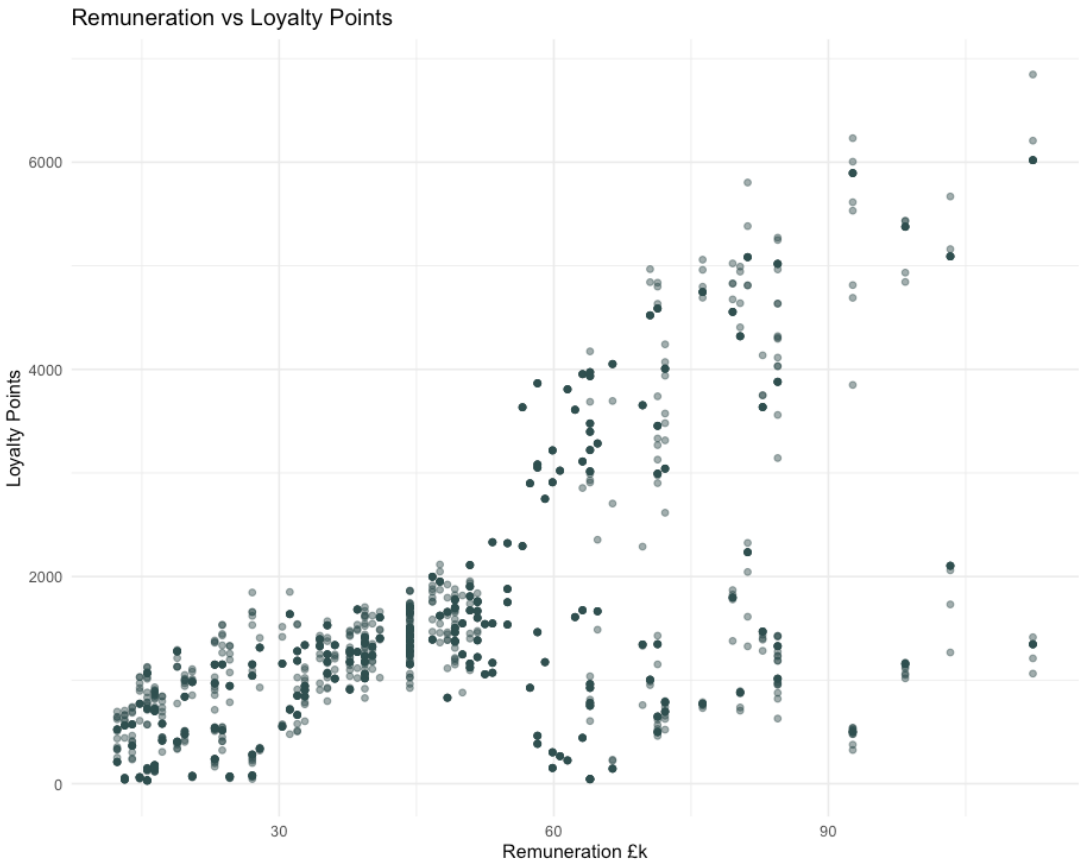
Shapiro-Wilk test for normality, $p < 0.05$, therefore loyalty_points is not normally distributed
Skewness = 1.463694.moderate to high right skew (>0)
Kurtosis = 4.70883. Sharper peak and fatter tails than a normal distribution (>3). Therefore, more prone to outliers
Range: min (25) max (6847). Difference: 6822. IQR: 979.25, Variance: 1646704, Standard Deviation: 1283.24

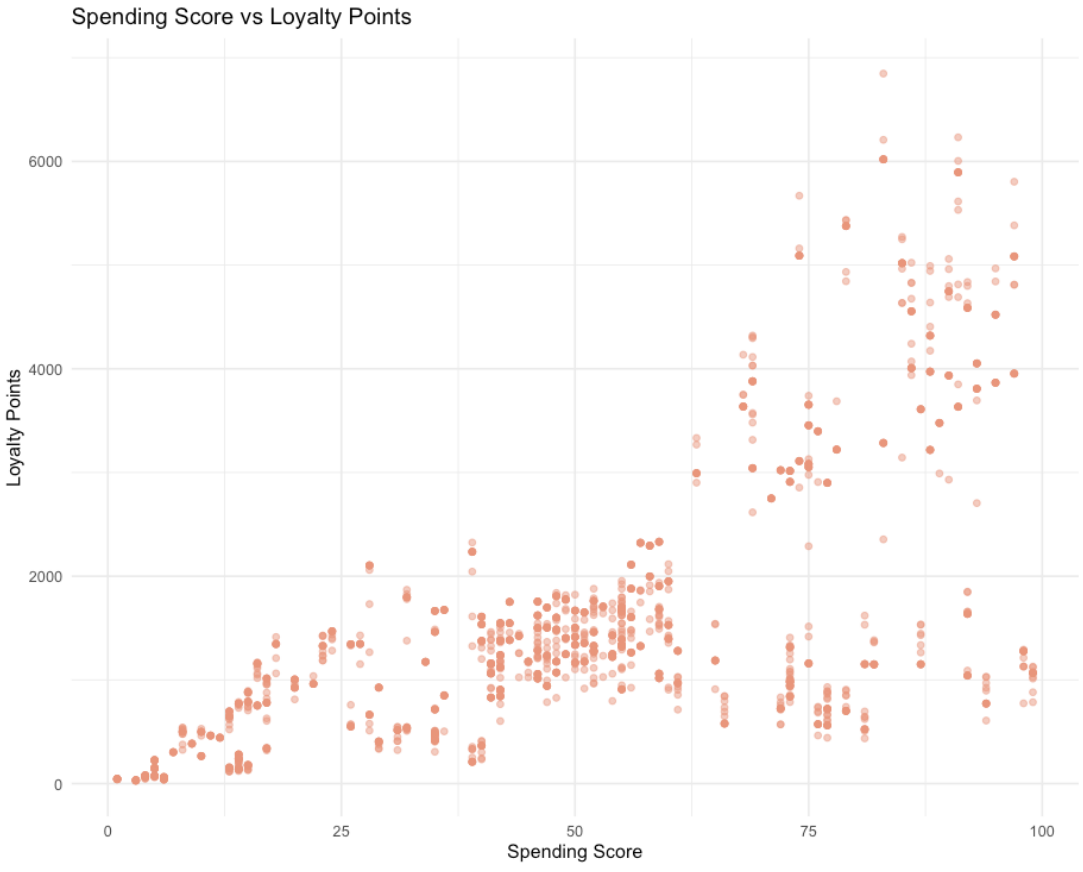
Correlation and Feature Importance



Scatter plot shows no linear relationship between 'age' and 'loyalty_points'

On the other hand, 'remuneration' and 'spending_score' did display a clear linear relationship with 'loaylty_points'.


Remuneration vs Loyalty Points
Loyalty Points
6000
4000
2000
0
30
60
90
Remuneration £k


Spending Score vs Loyalty Points
Loyalty Points
6000
4000
2000
0
0
25
50
75
100
Spending Score

Outlier detection

```
> # Determine lower and upper bounds
> lower_bound <- Q1 - 1.5 * IQR_value

> upper_bound <- Q3 + 1.5 * IQR_value

> # Identify outliers (loyalty_points outside of the bounds)
> outliers <- df$loyalty_points < lower_bound | df$loyalty_points > upper_bound

> # Count the outliers
> sum(outliers)
[1] 266

> # Identify number of customers with loyalty_points above the upper bound
> upper_outliers <- df$loyalty_points > upper_bound

> # Count how many customers exceed the upper bound
> num_upper_outliers <- sum(upper_outliers)

> cat("Number of customers with loyalty points above the upper bound:", num_upper_outliers, "\n")
Number of customers with loyalty points above the upper bound: 266

> # Identify number of customers with loyalty_points below the lower bound
> lower_outliers <- df$loyalty_points < lower_bound

> # Count how many customers are below the lower bound
> num_lower_outliers <- sum(lower_outliers)

> cat("Number of customers with loyalty points below the lower bound:", num_lower_outliers, "\n")
Number of customers with loyalty points below the lower bound: 0

> # Remove rows with outliers in loyalty_points
> df_no_outliers <- df[!outliers, ]
```

Whilst outlier detection was carried out in respect of all numerical variables, the total number of outliers were 266 and all above the upper bound *(Q3 + 1.5 * IQR_value)* and all in relation to loyalty_points.

Additional Sentiment Analysis Visualisations

TextBlob – Top 20 Positive Reviews and Summaries

Top 20 Positive Reviews:

	review	review_polarity
1	perfect	1.00
2	excellent toy to simulate thought	1.00
3	awesome gift	1.00
4	perfect for tutoring my grandson in spelling	1.00
5	awesome addition to my rpg gm system	1.00
6	best set buy 2 if you have the means	1.00
7	perfect just what i ordered	1.00
8	awesome toy	1.00
9	it is the best thing to play with and also mind blowing in some ways	1.00
10	wonderful for my grandson to learn the resurrection story	1.00
11	awesome set	1.00
12	this was perfect to go with the 7 bean bags i just wish they were not separate orders	1.00
13	one of the best board games i played in along time	1.00
14	delightful product	1.00
15	its awesome	1.00
16	came in perfect condition	1.00
17	awesome book	1.00
18	wonderful product	1.00
19	awesome	1.00
20	excellent activity for teaching selfmanagement skills	1.00

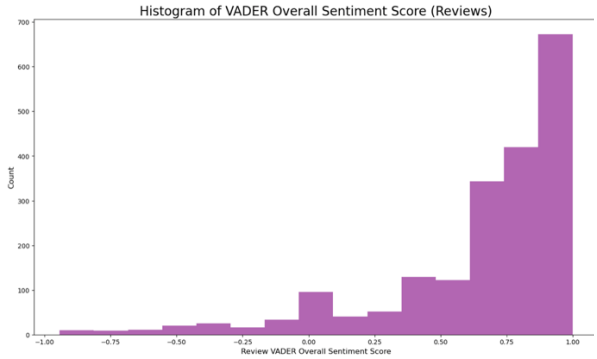
Top 20 Positive Summaries:

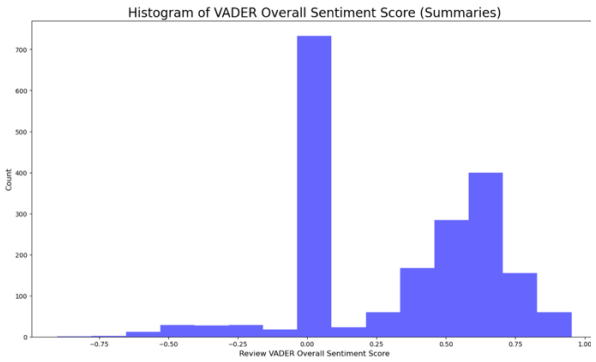
	summary	summary_polarity
1	one of the best	1.00
2	excellent	1.00
3	one of the best games ever	1.00
4	awesome learning tool	1.00
5	he was very happy with his gift	1.00
6	best orcs from wotc	1.00
7	awesome expansion	1.00
8	perfect gift	1.00
9	awesome sticker activity for the price	1.00
10	awesome	1.00
11	wonderful	1.00
12	all f the mudpuppy toys are wonderful	1.00
13	the perfect gift for preschool construction fans	1.00
14	best dungeon crawler	1.00
15	awesome puzzle	1.00
16	the best among the dd boardgames	1.00
17	awesome addition to our dd antics	1.00
18	awesome book	1.00
19	wonderful and	1.00
20	awesome expansion	1.00

VADER – WordCloud (Reviews)

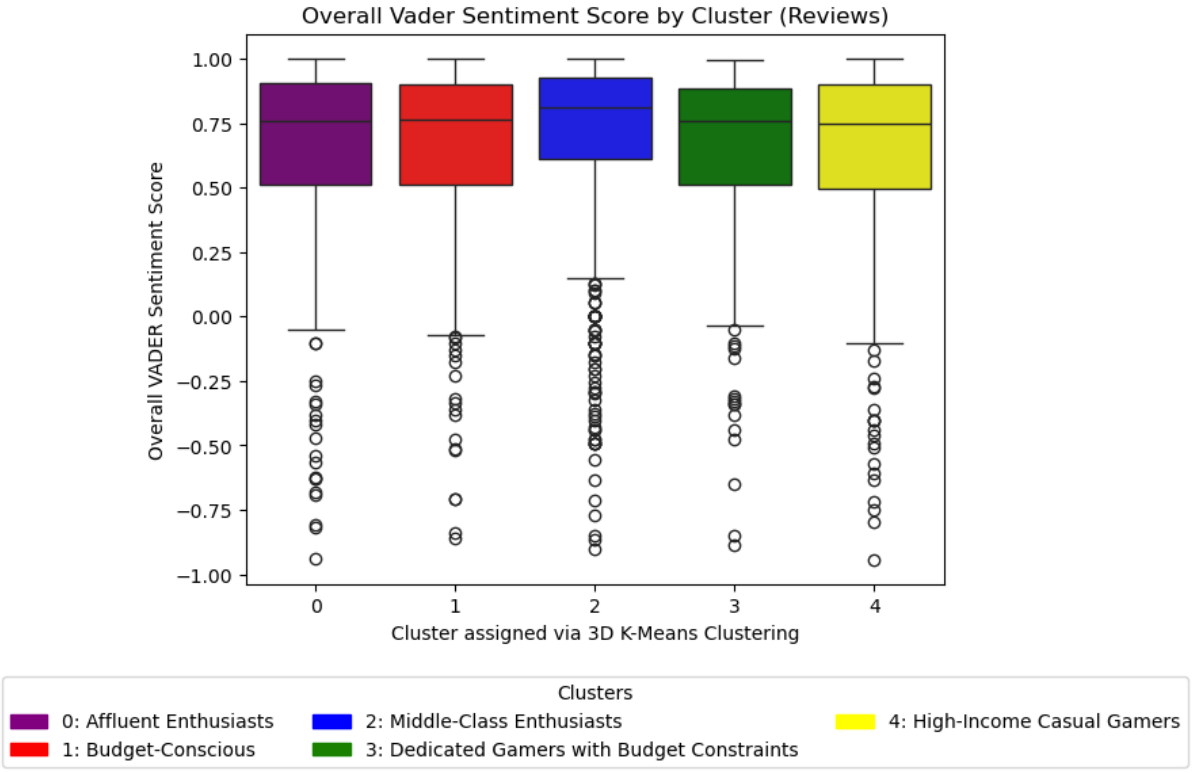
VADER – WordCloud (Summaries)

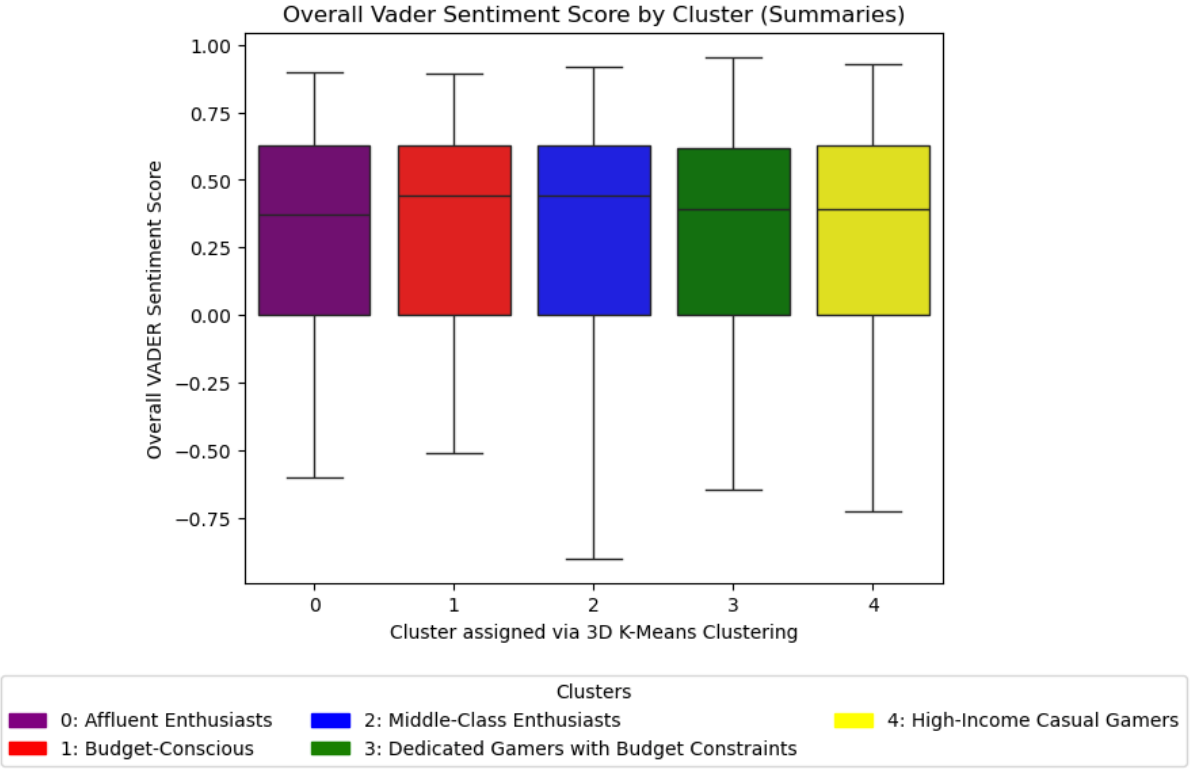
VADER – Distribution of Overall Sentiment (Compound) Score





Analysis of Overall VADER Sentiment Score by Cluster





Comprehensive overview of regression modelling

Several multiple linear regression (MLR) models were fitted and evaluated using a train-test split to assess predictive performance. Diagnostic checks for normality of residuals, multicollinearity, and heteroskedasticity were performed.

In Python, non-normality of residuals and heteroskedasticity were detected, along with an issue where the model was producing negative predictions for 'loyalty_points'—a non-sensical outcome given the context. To address these issues, a natural log transformation was applied to the dependent variable ('loyalty_points'). Potential reasons for the negative predictions included:

- The model extrapolating beyond the observed data range.
- Poor model fit.
- An inability to capture the underlying distribution accurately.

After applying the log transformation, the model no longer produced negative predicted values for 'log_loyalty_points'. While performance metrics such as R-squared, RMSE, MAE, and MAPE cannot be directly compared between models predicting the original and log-transformed targets (without exponentiating the log predictions back to the original scale), model7 achieved an R-squared of 79.5% on the training data, with all coefficients statistically significant.

In R, an alternative MLR model was fitted after removing outliers in an attempt to improve prediction of 'loyalty_points'. However, this resulted in a decline in model performance, suggesting that the outliers were in fact informative—possibly reflecting meaningful subgroups or patterns in the data. Given the small sample size, excluding these outliers negatively impacted the regression fit and model robustness.

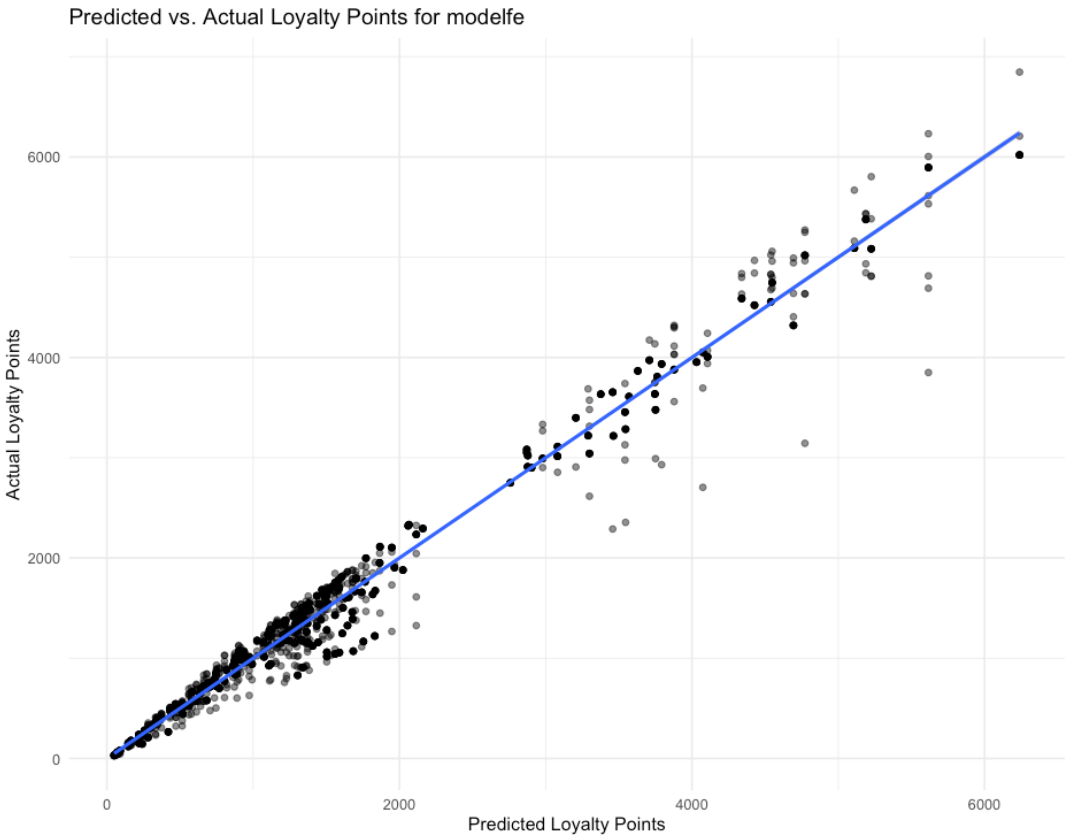
A further model in R, referred to as modelfe, incorporated a feature-engineered variable, 'potential_value', calculated as the product of 'spending_score' and 'remuneration'. After extensive experimentation, the combination of 'potential_value' and 'spending_score' proved to be the most effective for predicting 'loyalty_points'. This approach captures both the direct effect of 'spending_score' and its interaction with 'remuneration'. 'Potential_value' can thus be interpreted as a proxy variable for the combined influence of these two predictors on customer loyalty.

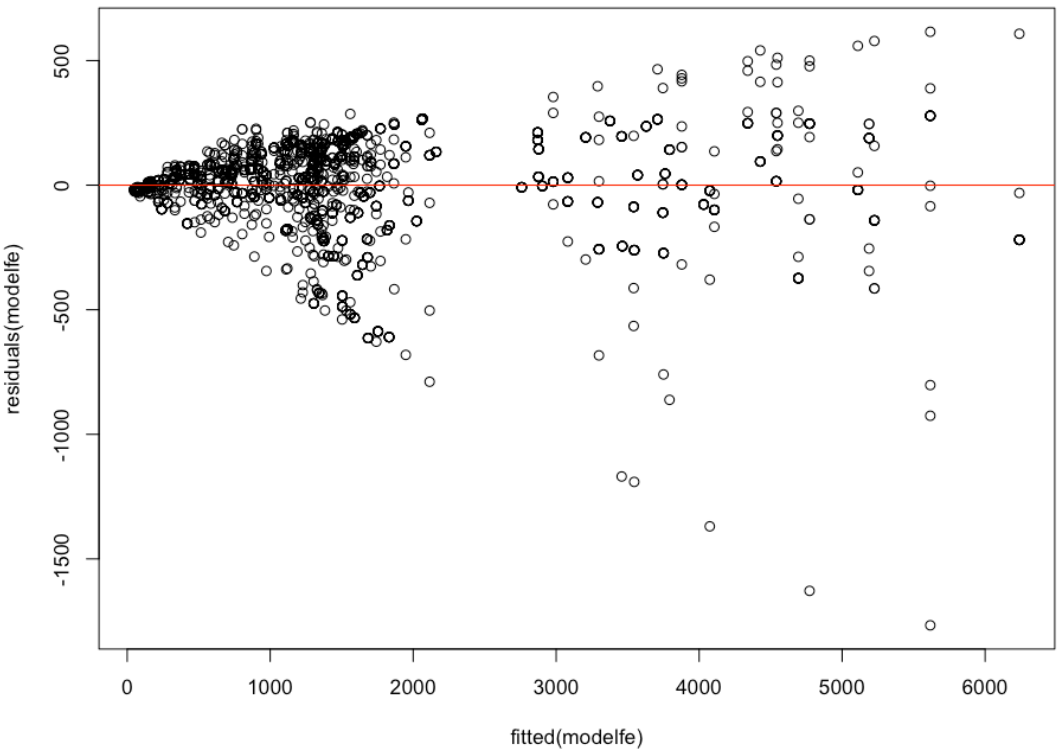
Importantly, the model generated only non-negative predictions (on test scenarios), aligning with business expectations and providing more interpretable results than the log-transformed model. On the training data, modelfe achieved an adjusted R-squared of 0.9755, with all coefficients and the overall model statistically significant. Test set performance metrics were excellent: **RMSE**: 184.544, **MAE**: 122.101. **R-squared**: 0.9824.

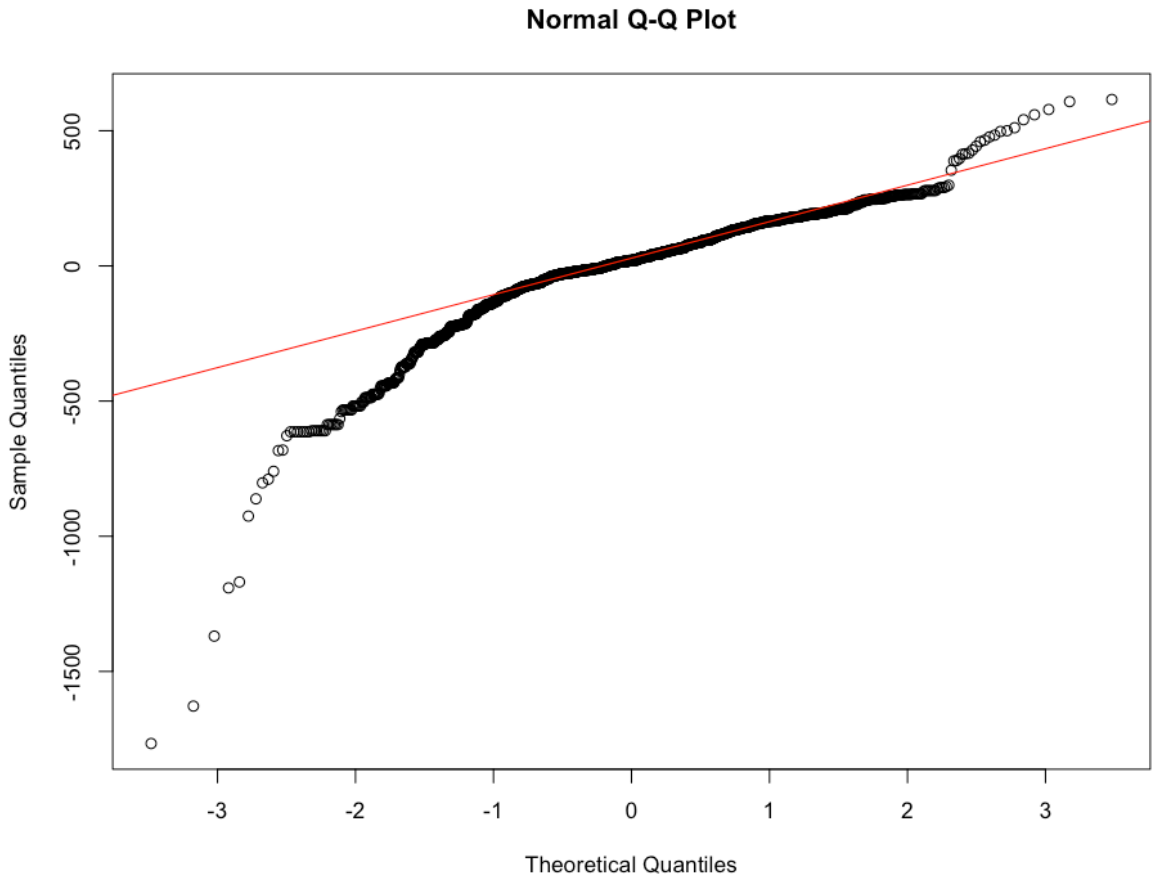
Recommendations:

- **Target High Spending Score Customers:**
  'spending_score' emerged as the strongest predictor of loyalty_points. Marketing strategies should prioritize customers with high spending scores, as they consistently generate higher loyalty rewards and are likely more engaged.
- **Integrate modelfe in loyalty prediction dashboard.**
- **Leverage the 'potential_value' Metric:**
  The feature-engineered variable 'potential_value' (calculated as *spending_score* × *remuneration*) proved to be a valuable predictor. Customers with high potential_value scores should be prioritized in targeted campaigns, as they represent a high-return segment.
- **Segment Marketing by Income Levels:**
  Although 'remuneration' is less influential than 'spending_score', it still plays a significant role in predicting loyalty. Tailored marketing strategies should be developed for different income segments, in alignment with the K-Means clustering analysis that identified five distinct customer groups.
- **Engage High-Value Outliers Strategically:**
  Retaining outliers slightly worsened the overall regression model fit, but closer inspection suggests that the 266 high-value customers are meaningful outliers, likely representing premium segments. These customers should not be ignored—consider designing exclusive loyalty tiers or personalized benefits to encourage retention and further engagement.

```
> # View the model summary
> summary(modelfe)

Call:
lm(formula = loyalty_points ~ spending_score + potential_value,
    data = df2)

Residuals:
     Min       1Q   Median       3Q      Max
-1766.50   -62.24    19.73   119.91   615.50

Coefficients:
                 Estimate Std. Error t value Pr(>|t|)
(Intercept)     23.638451   9.394713   2.516   0.0119 *
spending_score  -1.763447   0.232637  -7.580 5.25e-14 ***
potential_value  0.682316   0.003186 214.184  < 2e-16 ***
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

Residual standard error: 194.1 on 1997 degrees of freedom
Multiple R-squared:  0.9771,    Adjusted R-squared:  0.9771
F-statistic: 4.268e+04 on 2 and 1997 DF,  p-value: < 2.2e-16
```


Predicted vs. Actual Loyalty Points for modelfe
Actual Loyalty Points
Predicted Loyalty Points
0
2000
4000
6000


residuals(modelfe)
fitted(modelfe)
500
0
-500
-1000
-1500
0
1000
2000
3000
4000
5000
6000


Normal Q-Q Plot
Sample Quantiles
Theoretical Quantiles
500
0
-500
-1000
-1500
-3
-2
-1
0
1
2
3

```
> # Check for multicollinearity
> vif_valuesfe <- vif(modelfe)

> cat("Variance Inflation Factors:\n")
Variance Inflation Factors:

> print(vif_valuesfe)
 spending_score potential_value
       1.954994        1.954994

> # VIF between 1-5, therefore, low multicollinearity.
>
> # Test for heteroscedasticity
> bp_testfe <- bptest(modelfe)

> cat("Breusch-Pagan Test for Heteroscedasticity:\n")
Breusch-Pagan Test for Heteroscedasticity:

> print(bp_testfe)

        studentized Breusch-Pagan test

data:  modelfe
BP = 121.53, df = 2, p-value < 2.2e-16


> # p < 0.05, therefore, evidence of heteroskedasticity in the regression model.
```

```
> # Test for autocorrelation of residuals
> dw_testfe <- durbinWatsonTest(modelfe)

> cat("Durbin-Watson Test for Autocorrelation:\n")
Durbin-Watson Test for Autocorrelation:

> print(dw_testfe)
 lag Autocorrelation D-W Statistic p-value
   1      0.01236174       1.97519    0.58
 Alternative hypothesis: rho != 0
```

```
> # Shapiro-Wilk normality test
> shapiro_testfe <- shapiro.test(residuals(modelfe))

> cat("Shapiro-Wilk Test for Normality:\n")
Shapiro-Wilk Test for Normality:

> print(shapiro_test)

        Shapiro-Wilk normality test

data:  residuals(model1)
W = 0.99158, p-value = 2.391e-09


> # Create scenarios for prediction
> scenariosfe <- data.frame(
+   spending_score = c(76, 6, 94, 3),
+   potential_value = c(1059.44, 88.56, 1387.44 .... [TRUNCATED]

> # Make predictions
> scenariosfe$predicted_loyalty <- predict(modelfe, newdata = scenariosfe)

> print(scenariosfe)
  spending_score potential_value predicted_loyalty
1             76         1059.44         612.48966
2              6           88.56          73.48370
3             94         1387.44         804.54736
4              3           46.74          50.23957
```

```
> # 6.5a Evaluate predictive performance of modelfe
>
> # Set seed for reproducibility
> set.seed(42)

> # Create an index for splitting (80% training, 20% testing)
> sample_index <- sample(1:nrow(df2), size = round(0.8 * nrow(df2)))

> # Split the data
> train_data <- df2[sample_index, ]

> test_data <- df2[-sample_index, ]

> # Fit the model on training data
> model_train <- lm(loyalty_points ~ spending_score + potential_value, data = train_data)

> # Summarize the model
> summary(model_train)

Call:
lm(formula = loyalty_points ~ spending_score + potential_value,
    data = train_data)

Residuals:
     Min       1Q   Median       3Q      Max
-1755.47   -61.60    21.97   123.31   626.53

Coefficients:
               Estimate Std. Error t value Pr(>|t|)
(Intercept)     18.68396   10.68407   1.749   0.0805 .
spending_score  -1.59011    0.26414  -6.020 2.16e-09 ***
potential_value  0.67973    0.00368 184.711  < 2e-16 ***
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

Residual standard error: 196.5 on 1597 degrees of freedom
Multiple R-squared:  0.9755,    Adjusted R-squared:  0.9755
```

```
Residual standard error: 196.5 on 1597 degrees of freedom
Multiple R-squared:  0.9755,    Adjusted R-squared:  0.9755
F-statistic: 3.177e+04 on 2 and 1597 DF,  p-value: < 2.2e-16


> # Make predictions on test data
> predictions_train_fe <- predict(model_train, newdata = test_data)

> # Evaluate model performance
> # Calculate Root Mean Squared Error (RMSE)
> rmse <- sqrt(mean((test_data$loyalty_points - predictions_train_fe)^2))

> print(paste("RMSE:", rmse))
[1] "RMSE: 184.544104971727"

> # Calculate Mean Absolute Error (MAE)
> mae <- mean(abs(test_data$loyalty_points - predictions_train_fe))

> print(paste("MAE:", mae))
[1] "MAE: 122.101862402661"

> # Calculate R-squared on test data
> ss_total <- sum((test_data$loyalty_points - mean(test_data$loyalty_points))^2)

> ss_residual <- sum((test_data$loyalty_points - predictions_train_fe)^2)

> r_squared <- 1 - (ss_residual / ss_total)

> print(paste("Test R-squared:", r_squared))
[1] "Test R-squared: 0.982390275396215"
```

```
# Compare R-squared values
print("R-squared (OLS model): ", model7aols.rsquared)
print("R-squared (Sklearn LM model): ", model7alm.score(X_test, y_test))
```

```
R-squared (OLS model):  0.7952860387636151
R-squared (Sklearn LM model):  0.8142573833960147
```

```
# Evaluate performance of OLS model
print("Mean Absolute Error (OLS): ", mean_absolute_error(y_test, predictions7a_OLS))
print("Mean Squared Error (OLS): ", mean_squared_error(y_test, predictions7a_OLS))
print("RMSE (OLS): ", math.sqrt(mean_squared_error(y_test, predictions7a_OLS)))
```

```
Mean Absolute Error (OLS):  0.33996151779891304
Mean Squared Error (OLS):  0.17845375504696245
RMSE (OLS):  0.42243787122719295
```

```
# Evaluate performance of Linear Regression model
print("Mean Absolute Error (LM): ", mean_absolute_error(y_test, predictions7a_LM))
print("Mean Squared Error (LM): ", mean_squared_error(y_test, predictions7a_LM))
print("RMSE (LM): ", math.sqrt(mean_squared_error(y_test, predictions7a_LM)))
```

```
Mean Absolute Error (LM):  0.33996151779891276
Mean Squared Error (LM):  0.17845375504696243
RMSE (LM):  0.4224378712271929
```

Check for normality of residuals

```
# To properly check for normality of residuals, need to compute residuals based on training data.
predictions_train7a = model7alm.predict(X_train)
residuals_train7a = y_train - predictions_train7a
sns.displot(residuals_train7a, bins=20, kde=True)
```

Detailed overview of tree-based models

In Python, two Decision Tree Regressor models were developed, with pruning techniques applied to enhance model performance and prevent overfitting. The feature_importances_ attribute indicated that 'remuneration', 'age' and 'spending_score' were the most influential features. Consequently, the final decision tree model (dtr2) was trained using only these selected features, with a max_depth parameter set to 14. This refinement improved the model's generalization capabilities and reduced overfitting.

Building on this analysis, a Random Forest model (rfg2) was trained using the same reduced feature set. Below is a comparison of rfg2 with the best-performing Random Forest model implemented in R. Random Forest models typically perform better than the single Decision Trees because they reduce overfitting by averaging multiple trees.

Based on the metrics provided, Random Forest rfg2 is the better model:

Random Forest rfg2 (Python):

Test RMSE: 68.13

Test MSE: 4641.22

Test R^2: 0.997

Test MAE: 22.00

Number of trees: 100

rf_model1(R):

Test RMSE: 120.95

MSR (Mean of squared residuals): 12980.71

% Var explained: 99.21 (equivalent to R^2 of 0.9921)

MAPE: 10.23%

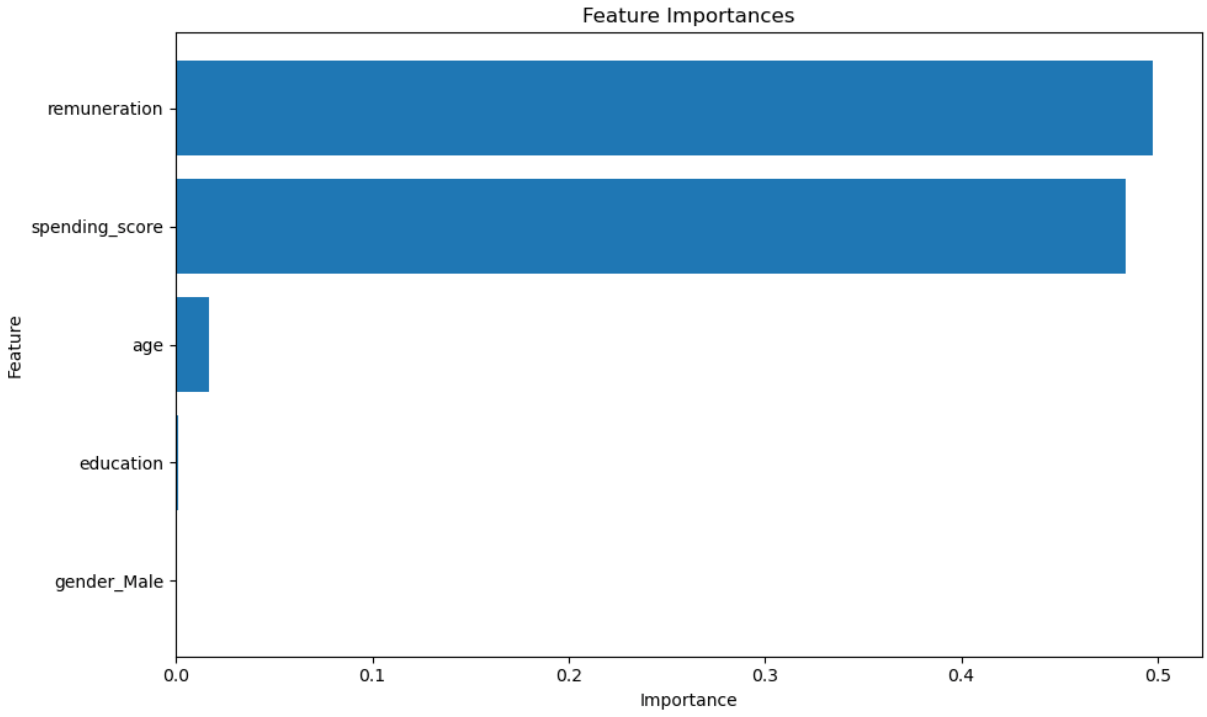
Number of trees: 500

Despite rf_model1 using 5 times more trees (500 vs 100), Random Forest rfg2 has a significantly lower RMSE on the test data (68.13 vs 120.95), which indicates better predictive performance. The lower test MSE of rfg2 (4641.22) compared to rf_model1's mean squared residuals (12980.71) further confirms this conclusion.

Additionally, rfg2 has a slightly higher R^2 value (0.997 vs 0.992), indicating it explains more variance in the target variable.

Recommendation:

- Deploy the Random Forest model (rfg2) for forecasting. It is preferred over modelfe due to ongoing issues with non-normality of residuals and heteroskedasticity in the latter. Tree-based models like Random Forest do not rely on the assumption of normally distributed residuals, making rfg2 a more robust and reliable choice for prediction in this context.
- Possibly incorporate 'potential_value' in the future iterations of rfg2.
- Evaluate and optimize rfg2 to enhance its performance on new data.



```python
# Calculate Training and Test Errors for pruned Tree (dtr_pruned)

mse_pruned_train = mean_squared_error(y_train, y_pred_pruned_train) # Training error (MSE)
mse_pruned_test = mean_squared_error(y_test, y_pred_pruned_test) # Test error (MSE)

mae_pruned_train = mean_absolute_error(y_train, y_pred_pruned_train) # Training error (MAE)
mae_pruned_test = mean_absolute_error(y_test, y_pred_pruned_test) # Test error (MAE)

rmse_pruned_train = mean_squared_error(y_train, y_pred_pruned_train, squared=False) # Training error (RMSE)
rmse_pruned_test = mean_squared_error(y_test, y_pred_pruned_test, squared=False) # Test error (RMSE)

r2_pruned_train = r2_score(y_train, y_pred_pruned_train) # Training R²
r2_pruned_test = r2_score(y_test, y_pred_pruned_test) # Test R²

# Output the results for pruned tree
print("Pruned Tree - Training Error (MSE):", mse_pruned_train)
print("Pruned Tree - Test Error (MSE):", mse_pruned_test)
print("Pruned Tree - Training Error (MAE):", mae_pruned_train)
print("Pruned Tree - Test Error (MAE):", mae_pruned_test)
print("Pruned Tree - Training Error (RMSE):", rmse_pruned_train)
print("Pruned Tree - Test Error (RMSE):", rmse_pruned_test)
print("Pruned Tree - Training R²:", r2_pruned_train)
print("Pruned Tree - Test R²:", r2_pruned_test)
```

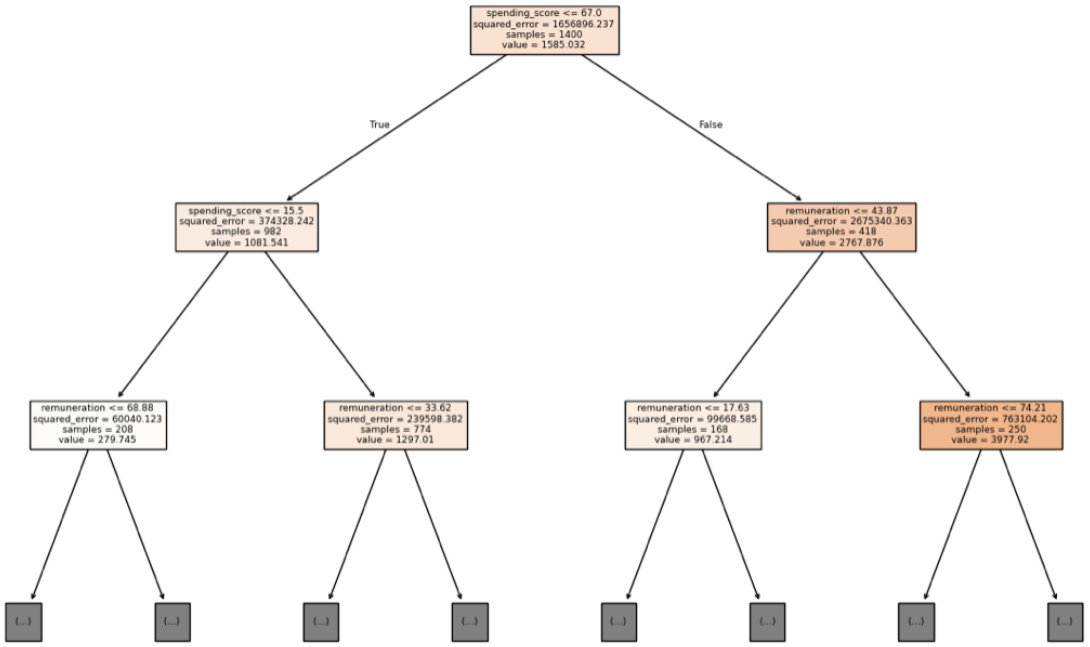
```
Pruned Tree - Training Error (MSE): 1108.7726715115036
Pruned Tree - Test Error (MSE): 8296.772192337798
Pruned Tree - Training Error (MAE): 9.28421893282184
Pruned Tree - Test Error (MAE): 37.474222885586485
Pruned Tree - Training Error (RMSE): 33.29823826438125
Pruned Tree - Test Error (RMSE): 91.08661917283898
Pruned Tree - Training R²: 0.9993308134529675
Pruned Tree - Test R²: 0.9948778936730243
```

```python
# Make predictions on new data
new_customer = pd.DataFrame({'remuneration': [19.68],
                             'spending_score': [73],
                             'age': [27]})

predicted_points = dtr_pruned.predict(new_customer)
print(f"Predicted Loyalty Points for new customer: {predicted_points[0]:.2f}")
```

```
Predicted Loyalty Points for new customer: 840.00
```

```python
# Get the max depth of the tree
max_depth = dtr2.get_depth()

# Display the max depth
print("Max depth of the decision tree:", max_depth)
```

```
Max depth of the decision tree: 14
```



Works Cited

Bond Brand Loyalty. (2020). *The Loyalty Report 2020: Executive Summary*. Retrieved from https://f.hubspotusercontent10.net/hubfs/352767/TLR%20Library/Bond%20Loyalty%20Report%202020_US%20ExecSummary.pdf

Bibliography

- Kumar, V., & Reinartz, W. (2018). "Customer Loyalty Programs: Design and Effectiveness." Journal of the Academy of Marketing Science, 46(6), 793-817.

- Wedel, M., & Kamakura, W. A. (2022). "Market Segmentation: Conceptual and Methodological Foundations." Springer Science. (

- Pang, B., & Lee, L. (2023). "Opinion Mining and Sentiment Analysis." Foundations and Trends in Information Retrieval, 2(1-2), 1-135.

- Breiman, L. (2001). "Random Forests." Machine Learning, 45(1), 5-32.

- Williams, D., Consalvo, M., et al. (2023). "Looking for Gender: Gender Roles and Behaviors Among Online Gamers." Journal of Communication, 59(4), 700-725.

- Wei, P. S., & Lu, H. P. (2022). "An examination of the celebrity endorsements and online customer reviews influence female users' shopping behavior." Computers in Human Behavior, 28(5), 1527-1535.

- Venkatesan, R., & Kumar, V. (2022). "A Customer Lifetime Value Framework for Customer Selection and Resource Allocation Strategy." Journal of Marketing, 68(4), 106-125.