Technical report

Turtle Games: Customer segmentation and spending behaviour

Prepared by: Yuliya Pauzunova
16 December 2024

I. gaBackground/context of the business scenario

Client: Turtle Games

Business problem: Improving overall sales performance by analysing and considering customer trends.

Analytical problem: Identify statistically significant features and segment customers to predict loyalty point balances.

The project aims to address the following analytical objectives/questions:

- Identify relevant features that best explain the variance in customer spending behaviour and evaluate possible relationships between features and loyalty point balances to develop predictive models.
- Group customers based on models' predictions to increase marketing strategy efficiency by implementing tailored marketing campaigns and promotions based on the identified segments.
- Identify and summarise common themes and opinions in product reviews and evaluate sentiment to improve overall sales performance by addressing customers' needs related to customer service or products

II. Analytical approach

2.1. Data cleaning and exploration

The analysis was completed in Python and R (Appendix 1).

To avoid repetition of code we used user-defined functions (Appendix 2).

Data source: 'turtle_reviews.csv' (2,000 observations)

Pre-cleaning

```
reviews.info()

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

Post-cleaning

```
reviews.info()

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

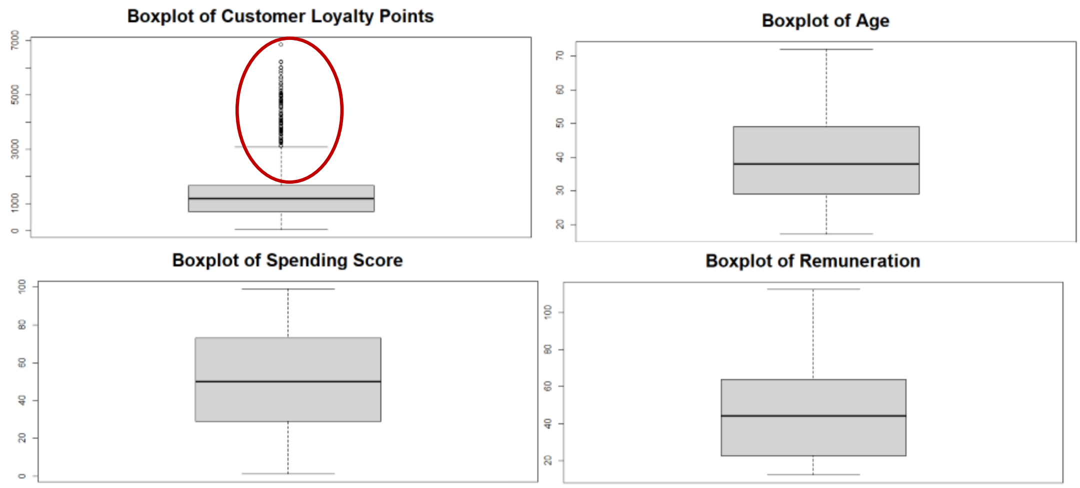
Duplicates analysis: 1,218 duplicates based on the attributes describing a customer (as presented in the exhibit below). We assume that each group of duplicate records is associated with a distinct customer. We removed duplicates and created a refined dataframe.

```
customers.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 782 entries, 0 to 781
Data columns (total 6 columns):
 #   Column          Non-Null Count  Dtype  
---  ------          --------------  -----  
 0   gender          782 non-null    string 
 1   age             782 non-null    int64  
 2   remuneration    782 non-null    float64
 3   spending_score  782 non-null    int64  
 4   loyalty_points  782 non-null    int64  
 5   education       782 non-null    string 
dtypes: float64(1), int64(3), string(2)
memory usage: 36.8 KB
```
.

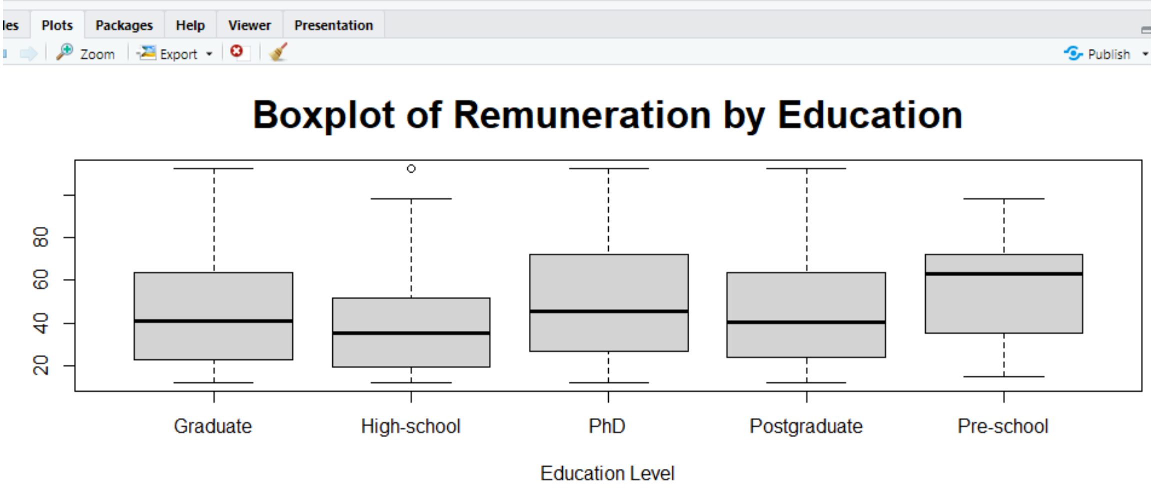
We employed EDA to identify patterns and interpret customer clusters.

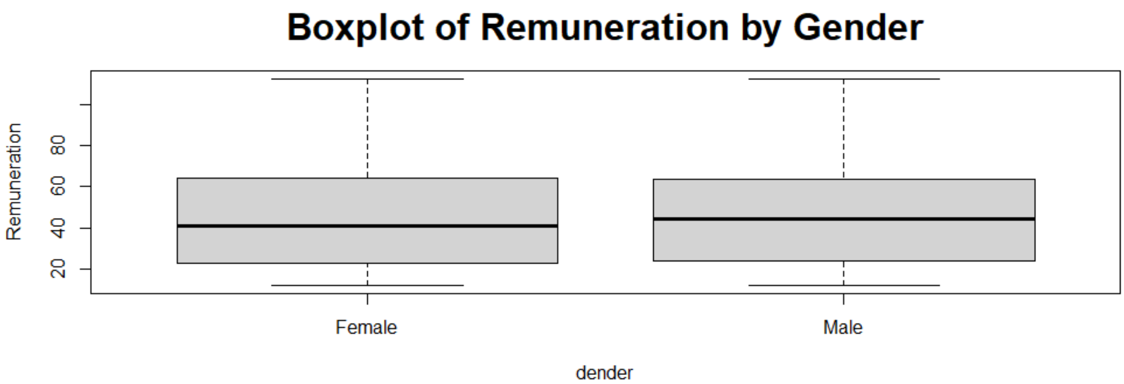
```
# view distribution: loyalty points, age, remun, count review
boxplot(customers$loyalty_points,
        main = "Boxplot of Customer Loyalty Points",
        cex.main = 2)
boxplot(customers$age,
        main = "Boxplot of Age",
        cex.main = 2)
boxplot(customers$remuneration,
        main = "Boxplot of Remuneration",
        cex.main = 2)
boxplot(customers$spending_score,
        main = "Boxplot of Spending Score",
        cex.main = 2)
boxplot(customers$count_review)
```



Loyalty points follow an asymmetric leptokurtic distribution, exhibiting a skewness of 1.66, a kurtosis of 5.3, and a Shapiro-Wilk statistic of 0.80 (LR section). Given the non-normality of the distribution, Turkey's method and the three-sigma rule are not applicable, and no records were removed.

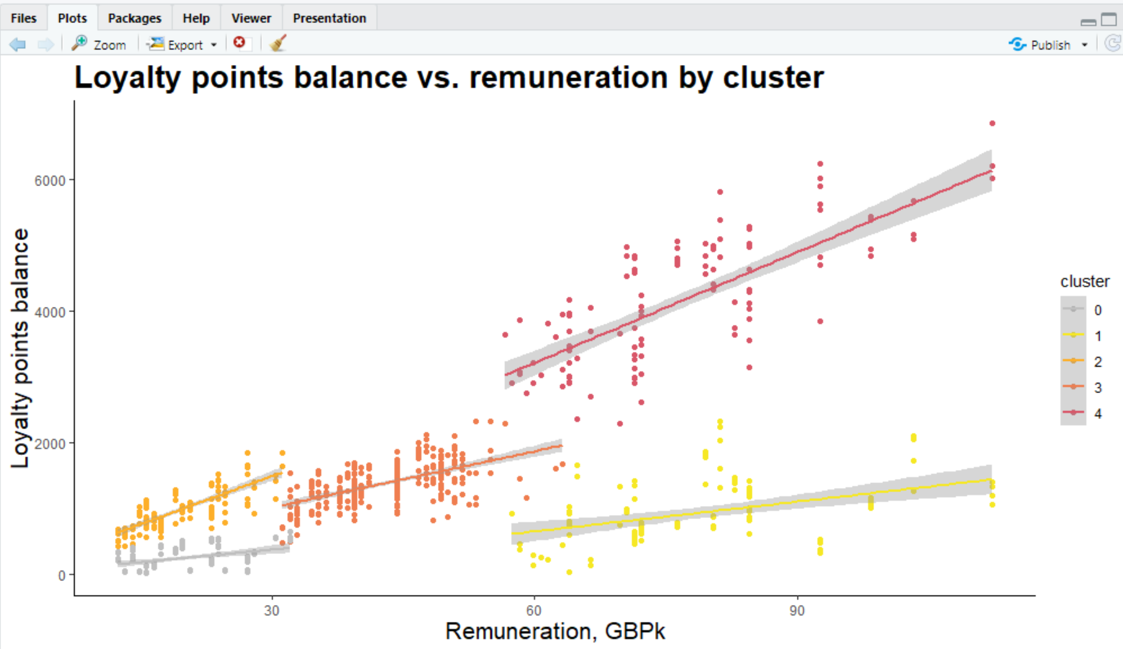
```
# Create a boxplot of remuneration by education
boxplot(remuneration ~ education,
        data = customers,
        main = "Boxplot of Remuneration by Education",
        cex.main = 2,
        xlab = "Education Level",
        ylab = "Remuneration")
```





Remuneration does not correlate with any numeric features except for loyalty points (Pearson correlation coefficients in the LR section) and does not vary by gender or education level.

```
> # Scatterplot: loyalty points vs  remuneration by cluster
> ggplot(customers, aes(x=remuneration, y=loyalty_points, col=cluster)) +
+   scale_color_manual(values = c('#BFBFBF', '#F6E825', '#FDAF2A', '#EF7E50', '#D95769')) +
+   geom_point() +
+   geom_smooth(method=lm) +
+   labs(x = "Remuneration, GBPk",
+        y = "Loyalty points balance",
+        title = "Loyalty points balance vs. remuneration by cluster") +
+   theme_classic() +
+   theme(
+     plot.title = element_text(size = 20, face = "bold"), # Increase title font size
+     axis.title.x = element_text(size = 16),              # Increase x-axis label font size
+     axis.title.y = element_text(size = 16)               # Increase y-axis label font size
+   )
`geom_smooth()` using formula = 'y ~ x'
>
```



The slope of the regression lines suggests that a unit increase in income leads to a proportionately greater rise in loyalty points for customers categorised in higher spending score clusters (specifically clusters 2 and 4) than for those in lower spending score clusters (clusters 0 and 1), i.e. loyalty points exhibit greater variability in response to income changes for customers with higher spending scores. Recognising this we fit three spending-score-category-specific LR models to achieve better predictive power (LR section).

We created and evaluated cluster-specific models in R (Appendix 4). We do not discuss details because they have limited practical value.

2.2. Predictive modelling

- Aalpha: 0.05
- Train/test split: 80/20, descriptive statistics are evaluated to exclude potential ‘data shift’.
- random_state: 42

2.2.1. Linear regression

We applied LR to achieve the following objectives:

- Identify significant features (MLR_all)
- Develop spending-score-category-specific models to predict loyalty points balances for scenario analysis

Initial feature selection was informed by descriptive statistics, Pearson correlation analysis, and visual inspection of potential relationships between numeric features and the dependent variable (loyalty_points).

Where descriptive statistics indicated non-normality of the dependent variable distribution (the key assessed metric Shapiro-Wilk) and/or visual inspection suggested potential heteroscedasticity (funnel-like pattern in X vs. Y scatterplots) or non-linear patterns, we considered log and root transformations (recognising the positive skew of Y-variable) to improve models' performance.

Transformations improved the normality of loyalty points distribution (S-W stat) with a slight improvement in correlation coefficients in most cases. What is more important, visual exploration suggested that root transformations result in patterns with more consistent variance at different points in X- vs. Y- variables scatterplots addressing inherent heteroscedasticity. These observations informed the type of transformation applied to the dependent variable.

To fit the initial MLR models we selected features with Pearson correlation coefficients > 30% as independent variables.

Fitted models summary

Model name	Spending_score_cat	Number of observations	Dependent variable transformation type	Intercept	Coefficients			Purpose
					'remuneration'	'spending_score'	'age'	
MLR_all	all	782	cube-root	2.5310	0.0804	0.8630	n/a	Identify meaningful features
LR_3_rem	high	248	5th root	3.5141	0.0224	n/a	n/a	Predict loyalty points
MLR_2	avg	283	square-root	- 5.6437	0.4600	0.3600	0.1200	Not used
MLR_1	low	251	cube-root	2.0526	0.6280	0.1352	n/a	Predict loyalty points
MLR_2_remss_test	avg	283	square-root	- 0.4373	0.4500	0.3600	n/a	Predict loyalty points

The independent variables' coefficients have expected signs, but are not interpretable for the transformation applied. Intercepts are not meaningful for the transformation (MLR_2 and MLR_2_remss_test negative in addition).

Model MLR_all

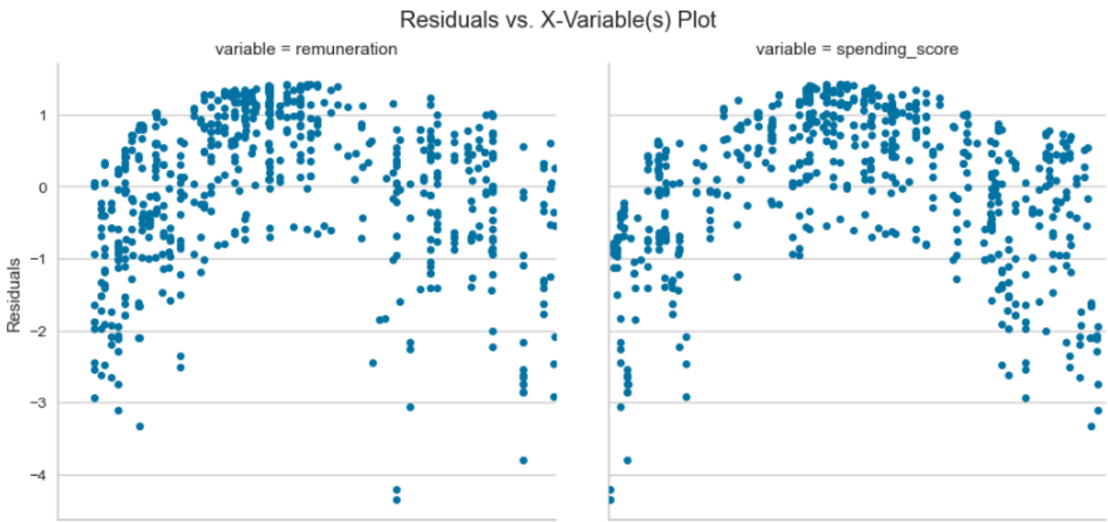
Descriptive statistics

	age	remuneration	spending_score	loyalty_points	loyalty_points_log	loyalty_points_2rt	loyalty_points_3rt
count	782.000000	782.000000	782.000000	782.000000	782.000000	782.000000	782.000000
mean	39.554987	46.058414	49.604859	1497.404092	6.890578	35.330002	10.514595
std	13.666746	25.249759	26.593610	1313.241715	1.065053	15.796016	3.244649
min	17.000000	12.300000	1.000000	25.000000	3.218876	5.000000	2.924018
25%	29.000000	23.165000	29.000000	701.000000	6.552508	26.476405	8.883266
50%	38.000000	44.280000	50.000000	1187.000000	7.079184	34.452866	10.588072
75%	49.000000	63.960000	73.000000	1658.000000	7.413367	40.718546	11.835724
max	72.000000	112.340000	99.000000	6847.000000	8.831566	82.746601	18.988913
range	55.000000	100.040000	98.000000	6822.000000	5.612690	77.746601	16.064895
IQR	20.000000	40.795000	44.000000	957.000000	0.860859	14.242142	2.952458
skewness (n=0)	0.599326	0.559234	-0.027514	1.660608	-1.076523	0.596465	0.096343
kurtosis (n=3)	2.778223	2.360189	2.035247	5.311685	4.466800	3.361124	3.172621
Shapiro-Wilk, stat (n=1)	0.952627	0.936880	0.965425	0.807552	0.909886	0.947999	0.966366
Shapiro-Wilk, p-value (>0.05)	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000

Pearson Correlation

	age	remuneration	spending_score	loyalty_points	loyalty_points_log	loyalty_points_2rt	loyalty_points_3rt
age	1.000000	0.006024	-0.124630	0.028904	0.000925	0.023254	0.017886
remuneration	0.006024	1.000000	-0.001645	0.630901	0.560484	0.631490	0.618104
spending_score	-0.124630	-0.001645	1.000000	0.632116	0.707846	0.696061	0.709969
loyalty_points	0.028904	0.630901	0.632116	1.000000	0.804473	0.963172	0.926228
loyalty_points_log	0.000925	0.560484	0.707846	0.804473	1.000000	0.930535	0.966382
loyalty_points_2rt	0.023254	0.631490	0.696061	0.963172	0.930535	1.000000	0.993217
loyalty_points_3rt	0.017886	0.618104	0.709969	0.926228	0.966382	0.993217	1.000000

The reverse ‘U-shaped’ pattern is observed in the “Residuals vs. X-Variable(s) Plot”, which suggests a deviation from linearity.

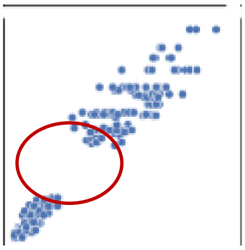


Model LR_3_rem

Descriptive statistics

	age	remuneration	spending_score	loyalty_points	count_review	loyalty_points_log	loyalty_points_2rt	loyalty_points_5rt
count	248.000000	248.000000	248.000000	248.000000	248.000000	248.000000	248.000000	248.000000
mean	37.125000	47.728629	80.798387	2549.862903	2.479839	7.559835	47.244729	4.591891
std	12.118364	30.793471	10.047241	1759.491415	2.573533	0.789125	17.862954	0.717598
min	17.000000	12.300000	61.000000	436.000000	1.000000	6.077642	20.880613	3.372076
25%	29.000000	17.220000	73.000000	932.500000	1.000000	6.837869	30.536860	3.925814
50%	34.000000	30.340000	79.000000	1658.000000	1.000000	7.413367	40.718546	4.404706
75%	39.000000	73.185000	90.000000	4036.250000	2.000000	8.303069	63.531448	5.262540
max	72.000000	112.340000	99.000000	6847.000000	8.000000	8.831566	82.746601	5.849248
range	55.000000	100.040000	38.000000	6411.000000	7.000000	2.753924	61.865988	2.477173
IQR	10.000000	55.965000	17.000000	3103.750000	1.000000	1.465200	32.994588	1.336726
skewness (n=0)	1.012570	0.291106	0.097448	0.428791	1.196393	-0.061515	0.186011	0.040580
kurtosis (n=3)	3.968043	1.530027	2.054574	1.726927	2.449440	1.450019	1.463295	1.422303
Shapiro-Wilk, stat (n=1)	0.912367	0.852759	0.964898	0.874407	0.547983	0.898062	0.889578	0.895662
Shapiro-Wilk, p-value (>0.05)	0.000000	0.000000	0.000009	0.000000	0.000000	0.000000	0.000000	0.000000

Note: SW stat for remuneration of 0.85 suggests a non-normal distribution. This is largely explained by the 'missing' average remuneration segment in the high spending score category and does not affect the model’s accuracy (same relates to the MLR_1).



Pearson correlation

	age	remuneration	spending_score	loyalty_points	count_review	loyalty_points_log	loyalty_points_2rt	loyalty_points_5rt
age	1.000000	0.107074	-0.009535	0.186361	-0.254683	0.194856	0.188230	0.191557
remuneration	0.107074	1.000000	0.076360	0.965946	0.095829	0.958396	0.970692	0.965560
spending_score	-0.009535	0.076360	1.000000	0.229860	0.033976	0.216428	0.221904	0.218283
loyalty_points	0.186361	0.965946	0.229860	1.000000	0.090129	0.970549	0.992954	0.981466
count_review	-0.254683	0.095829	0.033976	0.090129	1.000000	0.120599	0.108262	0.116512
loyalty_points_log	0.194856	0.958396	0.216428	0.970549	0.120599	1.000000	0.992001	0.998657
loyalty_points_2rt	0.188230	0.970692	0.221904	0.992954	0.108262	0.992001	1.000000	0.997188
loyalty_points_5rt	0.191557	0.965560	0.218283	0.981466	0.116512	0.998657	0.997188	1.000000

Model MLR_2 and MLR2_remss

Descriptive statistics

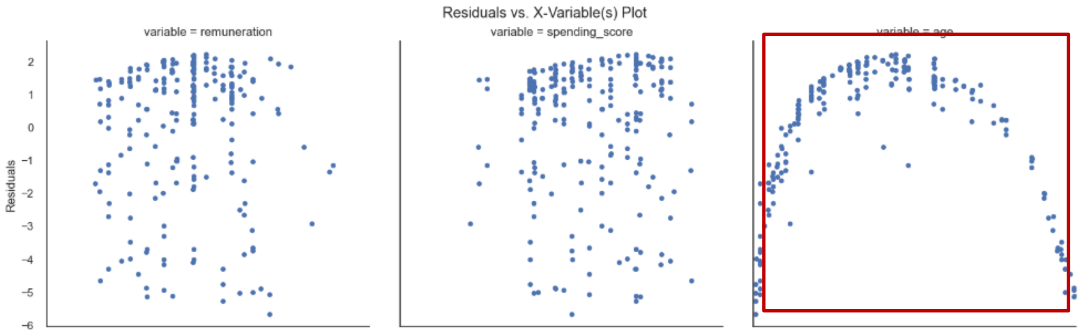
	age	remuneration	spending_score	loyalty_points	count_review	loyalty_points_log	loyalty_points_2rt	loyalty_points_3rt
count	283.000000	283.000000	283.000000	283.000000	283.000000	283.000000	283.000000	283.000000
mean	40.310954	42.947138	49.893993	1389.890459	2.826855	7.208402	37.023824	11.090693
std	14.664164	6.497970	6.579255	318.982767	2.790889	0.247010	4.381182	0.886196
min	17.000000	31.160000	34.000000	478.000000	1.000000	6.169611	21.863211	7.818846
25%	29.000000	37.720000	46.000000	1176.000000	1.000000	7.069874	34.292853	10.555263
50%	38.000000	44.280000	50.000000	1395.000000	1.000000	7.240650	37.349699	11.173556
75%	49.000000	48.380000	55.000000	1619.500000	7.000000	7.389872	40.243000	11.743391
max	72.000000	63.140000	65.000000	2332.000000	9.000000	7.754482	48.290786	13.260997
range	55.000000	31.980000	31.000000	1854.000000	8.000000	1.584871	26.427575	5.442152
IQR	20.000000	10.660000	9.000000	443.500000	6.000000	0.319998	5.950147	1.188128
skewness (n=0)	0.462103	0.151877	-0.104848	0.036918	0.882672	-0.806980	-0.350095	-0.492351
kurtosis (n=3)	2.442123	2.486520	2.350634	3.047252	1.805134	4.398668	3.333056	3.581760
Shapiro-Wilk, stat (n=1)	0.956272	0.969325	0.978556	0.996618	0.591866	0.965371	0.990215	0.984169
Shapiro-Wilk, p-value (>0.05)	0.000000	0.000010	0.000296	0.812760	0.000000	0.000003	0.054947	0.003217

Pearson Correlation

	age	remuneration	spending_score	loyalty_points	count_review	loyalty_points_log	loyalty_points_2rt	loyalty_points_3rt
age	1.000000	-0.045529	-0.045160	0.318068	0.130337	0.333694	0.327365	0.329853
remuneration	-0.045529	1.000000	-0.165671	0.576787	0.208179	0.573348	0.577527	0.576747
spending_score	-0.045160	-0.165671	1.000000	0.428491	-0.069948	0.424942	0.427699	0.427024
loyalty_points	0.318068	0.576787	0.428491	1.000000	0.070479	0.982700	0.996004	0.992709
count_review	0.130337	0.208179	-0.069948	0.070479	1.000000	0.067092	0.068362	0.067847
loyalty_points_log	0.333694	0.573348	0.424942	0.982700	0.067092	1.000000	0.995256	0.997823
loyalty_points_2rt	0.327365	0.577527	0.427699	0.996004	0.068362	0.995256	1.000000	0.999503
loyalty_points_3rt	0.329853	0.576747	0.427024	0.992709	0.067847	0.997823	0.999503	1.000000

While the Pearson correlation coefficient for the ‘loyalty_points’¬’age’ pair is > 30%, the “Residuals vs. X-Variable(s) Plot” suggests that the assumption of linearity for the MLR_2 model concerning the independent variable ‘age’ is violated (reverse ‘U-shaped’ pattern). Recognising this we excluded ‘age’-variable and fitted

MLR_2_remss instead.



Model MLR_1

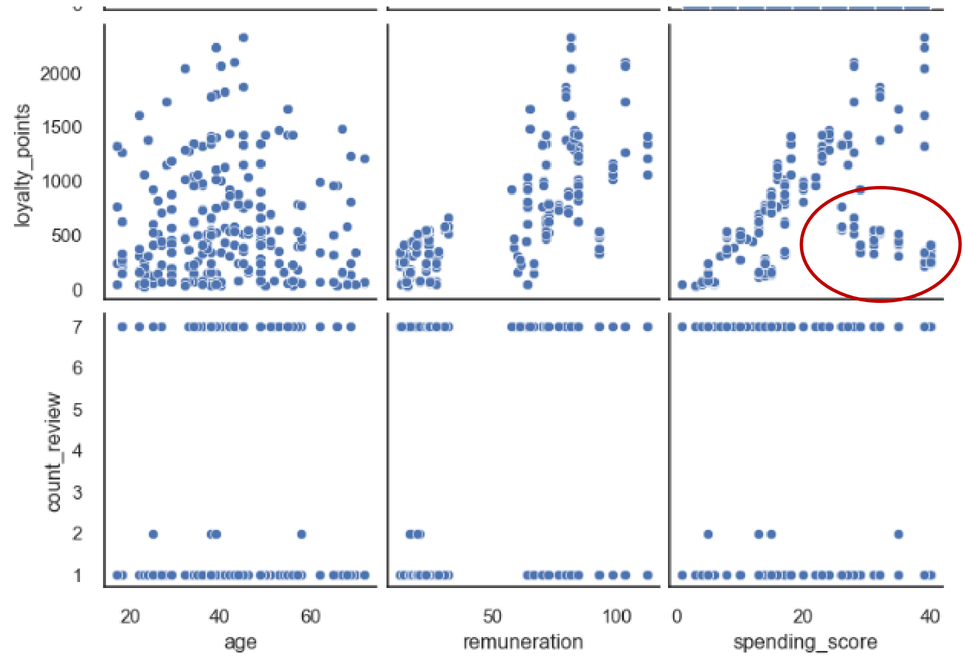
Descriptive statistics

	age	remuneration	spending_score	loyalty_points	count_review	loyalty_points_log	loyalty_points_2rt	loyalty_points_3rt
count	251.000000	251.000000	251.000000	251.000000	251.000000	251.000000	251.000000	251.000000
mean	41.103586	47.916096	18.458167	578.745020	2.330677	5.870976	21.647915	7.550284
std	13.672353	31.464744	11.020401	504.999547	2.481575	1.121244	10.514430	2.558080
min	17.000000	12.300000	1.000000	25.000000	1.000000	3.218876	5.000000	2.924018
25%	32.000000	17.220000	10.000000	157.000000	1.000000	5.056246	12.529964	5.394691
50%	39.000000	27.880000	15.000000	450.000000	1.000000	6.109248	21.213203	7.663094
75%	49.000000	77.900000	28.000000	845.500000	1.000000	6.739473	29.074176	9.454980
max	72.000000	112.340000	40.000000	2325.000000	7.000000	7.751475	48.218254	13.247715
range	55.000000	100.040000	39.000000	2300.000000	6.000000	4.532599	43.218254	10.323698
IQR	17.000000	60.680000	18.000000	688.500000	0.000000	1.683227	16.544212	4.060290
skewness (n=0)	0.395143	0.319524	0.389527	1.115255	1.349796	-0.527306	0.322820	0.041834
kurtosis (n=3)	2.546315	1.501808	2.029907	3.730760	2.832367	2.286762	2.261839	2.093801
Shapiro-Wilk, stat (n=1)	0.967704	0.845462	0.935699	0.886047	0.518889	0.946445	0.964708	0.971953
Shapiro-Wilk, p-value (>0.05)	0.000019	0.000000	0.000000	0.000000	0.000000	0.000000	0.000008	0.000074

Pearson Correlation

	age	remuneration	spending_score	loyalty_points	count_review	loyalty_points_log	loyalty_points_2rt	loyalty_points_3rt
age	1.000000	-0.056461	-0.080117	0.010688	0.037891	-0.015383	0.001712	-0.003288
remuneration	-0.056461	1.000000	-0.032804	0.748220	0.075611	0.707259	0.763212	0.753119
spending_score	-0.080117	-0.032804	1.000000	0.480902	-0.037886	0.603741	0.544490	0.566558
loyalty_points	0.010688	0.748220	0.480902	1.000000	0.060397	0.875808	0.971978	0.947687
count_review	0.037891	0.075611	-0.037886	0.060397	1.000000	0.057832	0.060025	0.059590
loyalty_points_log	-0.015383	0.707259	0.603741	0.875808	0.057832	1.000000	0.962732	0.982824
loyalty_points_2rt	0.001712	0.763212	0.544490	0.971978	0.060025	0.962732	1.000000	0.996024
loyalty_points_3rt	-0.003288	0.753119	0.566558	0.947687	0.059590	0.982824	0.996024	1.000000

Visual inspection



We note that there is a small group of observations which lies aside from the general linear pattern observed in the spending_score¬loyalty_points scatterplot. We recommend investigating whether those observations can be seen as outliers and excluded from the analysis. This must result in a demonstrably improved R-squared.

Linear Regression Metrics Summary

	MLR_all	LR_3_rem	MLR_2	MLR_1	MLR_2_remss
Durbin-Watson statistic (1.5-2.5)	1.933877	2.298320	1.982308	1.805602	2.094445
F statistic p-value (< 0.05)	0.000000	0.000000	0.000000	0.000000	0.000000
JB Probability (> 0.05)	0.000000	0.192341	0.000000	0.000000	0.000000
LM Test p-value (> 0.05)	0.578577	0.165024	0.126928	0.291067	0.327871
Resid. vs age corr	NaN	NaN	0.00	NaN	NaN
Resid. vs remuneration corr	-0.00	-0.00	-0.00	-0.00	-0.00
Resid. vs spending_score corr	-0.00	NaN	-0.00	0.00	-0.00
VIF Factor: age (<10)	NaN	NaN	1.000956	NaN	NaN
VIF Factor: remuneration (<10)	1.000195	1.000000	1.042453	1.000360	1.041479
VIF Factor: spending_score (<10)	1.000195	NaN	1.041482	1.000360	1.041479

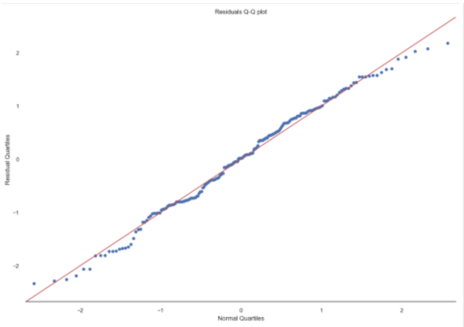
Linear Regression Assumptions Assessment Summary

Assumption	Metric	MLR_all	LR_3_rem	MLR_2	MLR_1	MLR_2_remss
Linearity	Scatterplot of X and Y					
Linearity	Residuals vs. X-Variable(s) Plot					
Linearity	Residuals vs. Fitted Values Plot					
No autocorrelation of residuals	Durbin-Watson statistic (1.5-2.5)					
No autocorrelation of residuals	Residuals vs. Fitted Values Plot					
Exogeneity	Pearson correlation: X vs. Residual					
Homoscedasticity	LM Test p-value					
Homoscedasticity	Residuals vs. Fitted Values Plot					
No perfect multicollinearity	VIF					
Normality of error terms (optionsl)	JB, Q-Q plot					

Legend:
Violated
Satisfied

F-statistic p-value < 0.05 indicates that all the regression models are statistically significant.

Note: LR_3_rem is the only model with the optional Normality of error terms satisfied (JB Probability > 0.05). While OLS does not require that the error term follows a normal distribution to produce unbiased estimates, satisfying this assumption allows statistical hypothesis testing and reliable confidence intervals.



Goodness of Fit Metrics Summary (test dataset)

	MLR_test	LR_3_rem_test	MLR_1_test	MLR_2_remss_test
R2	0.906427	0.869809	0.899409	0.603620
R2 (train)	0.891825	0.932260	0.917904	0.614948
Adj. R2 (train)	0.891478	0.931914	0.917070	0.611495
RMSE	393.538048	717.926025	128.713884	196.764224
MAE	318.451350	537.882979	83.708104	150.250712
MAPE, %	26.413167	19.523268	16.627255	10.508687
q1 (y_real)	777.000000	1010.250000	149.500000	1215.000000
q3 (y_real)	1686.000000	4798.750000	700.000000	1592.000000
mean (y_real)	1529.000000	2867.920000	505.098039	1389.087719

Given the linear regression assumptions are satisfied, models can be used to predict loyalty points based on customers' remuneration and target spending scores. MLR, LR_3_rem and MLR_1 models have very strong predictive power with R-squared of

91%, 87% and 90%, respectively. MLR_2_remss has strong predictive power with an R-squared of 60%. For predictions vs. observations plots please refer to Appendix 7.

We used spending-score-category-specific models to predict loyalty points' balances in response to the change in customer purchasing behaviour, recognising their stronger error metrics:

- Scenario 1: Cluster 1 spending score category = '2' ('medium'); Cluster 2 – '3' ('high')
- Scenario 2: Cluster 1 and Cluster 2 spending score category = '3' ('high')

Scenario 1 and Scenario 2 assumptions

cluster	remuneration	spending_score	spending_score_s1	spending_score_s2
0	19.574394	19.037879	19.037879	19.037879
1	79.353950	17.815126	49.893993	80.798387
2	19.311969	79.889764	79.889764	79.889764
3	42.947138	49.893993	80.798387	80.798387
4	77.554380	81.752066	81.752066	81.752066

Predicted average and total loyalty points per cluster

cluster	loyalty_points_avg	loyalty_points_sc1_avg	loyalty_points_sc2_avg	loyalty_points_sum	loyalty_points_sc1_sum	loyalty_points_sc2_sum
0	246.643939	246.643939	246.643939	32557	32557.000000	32557.000000
1	947.126050	2909.952785	4281.656779	112708	346284.381427	509517.156735
2	980.519685	980.519685	980.519685	124526	124526.000000	124526.000000
3	1389.890459	1816.040563	1816.040563	393339	513939.479192	513939.479192
4	4197.024793	4197.024793	4197.024793	507840	507840.000000	507840.000000

```
loyalty_points_sum        1170970.000000
loyalty_points_sc1_sum    1525146.860619
loyalty_points_sc2_sum    1688379.635927
dtype: float64
```

If the target spending score levels specific to each scenario are achieved, the total loyalty points balance is projected to increase by 30.2% in Scenario 1 and 10.7% in Scenario 2. Since loyalty points represent the value of purchases, we extrapolate these results to revenue.

2.2.2. Decision tree

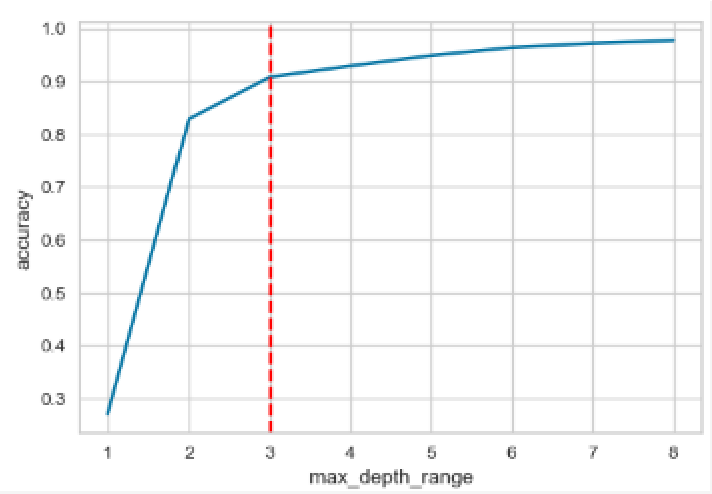
Decision tree analysis is complementary to LR allowing for the evaluation of the significance of categorical variables (like gender and education). Moreover, DT handles non-linear relationships, which are not captured in LR.

We pre-processed categorical features utilising OneHotEncoding and eliminated multicollinearity (VIF<10).

VIF factor: Selected Features

feature	VIF
education_Pre-school	1.070104
education_High-school	1.281899
education_Postgraduate	1.440967
education_PhD	1.548076
gender_Male	1.784633
spending_score	3.329248
remuneration	3.686249
age	5.340723

We optimised max_depth hyperparameter based on R-squared in the range of [1, 9). Selected max_depth=3



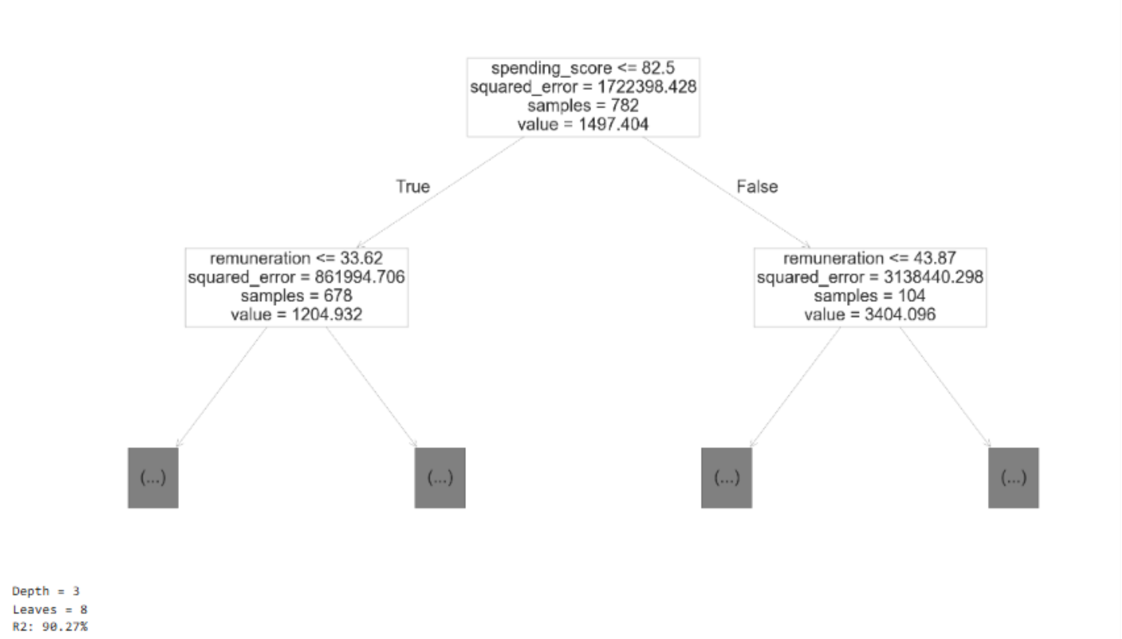
We evaluated the pruned model performance (see the summary below) and retrained it on the entire dataset to leverage all the available data.

Goodness of Fit Metrics Summary (train & test dataset)

	DT_prunned_train	DT_prunned_test	DT_fin	DT_fin_test	MLR_test
R2	0.912005	0.908051	0.889998	0.902689	0.906427
R2 (train)	None	None	None	None	0.891825
Adj. R2 (train)	None	None	None	None	0.891478
RMSE	391.181549	390.108166	435.277473	401.321993	393.538048
MAE	266.432613	274.656903	314.917333	296.629368	318.451350
MAPE, %	24.385323	23.133588	29.977913	27.254099	26.413167
q1 (y_real)	684.000000	777.000000	701.000000	777.000000	777.000000
q3 (y_real)	1637.000000	1686.000000	1658.000000	1686.000000	1686.000000
mean (y_real)	1489.467200	1529.000000	1497.404092	1529.000000	1529.000000

The pruned tree R2 of 90.8% on the test dataset is very high and compares well to that on the train dataset of 91.2%. Overall, test metrics (including errors) align with the train, indicating that the model generalises well to unseen data. At the same time, the final model performance is slightly weaker, which might be associated with overfitting. We note that decision trees are inherently prone to high variability. Cross-validation techniques can be used to assess how much the model changes across folds and to ensure that variability is minimised.

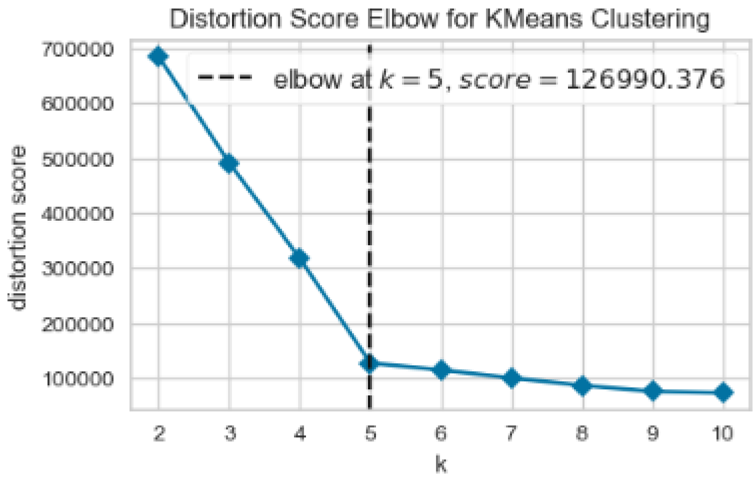
DT suggests remuneration and spending scores best explain the variance in loyalty point balances, which aligns with MLR output.



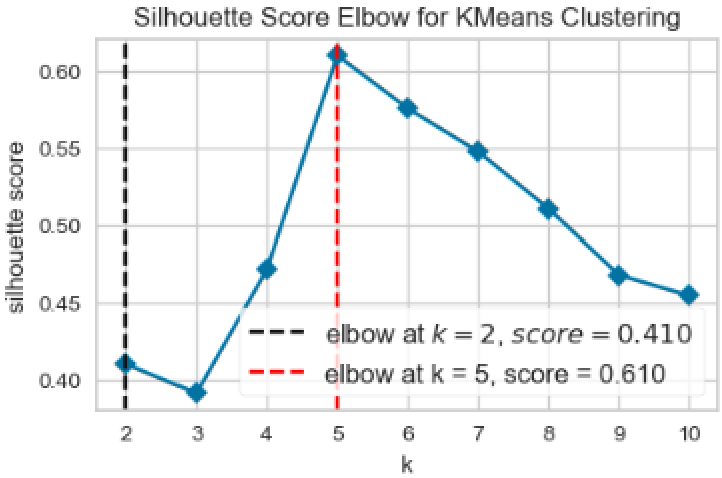
2.2.3. Clustering using k-means

Based on visual inspection (Appendix 5) and regression analysis (4.1-2) we selected 'remuneration', 'spending_score' as the basis for clustering analysis.

To determine the optimal k-value we employed the KElbowVisualizer from the yellowbrick library utilising the Elbow (metric='distortion') and the Silhouette (metric='silhouette') methods.



The Elbow method suggests the optimal k-value of 5 (for k > 5 the decrease in WSS is marginal).



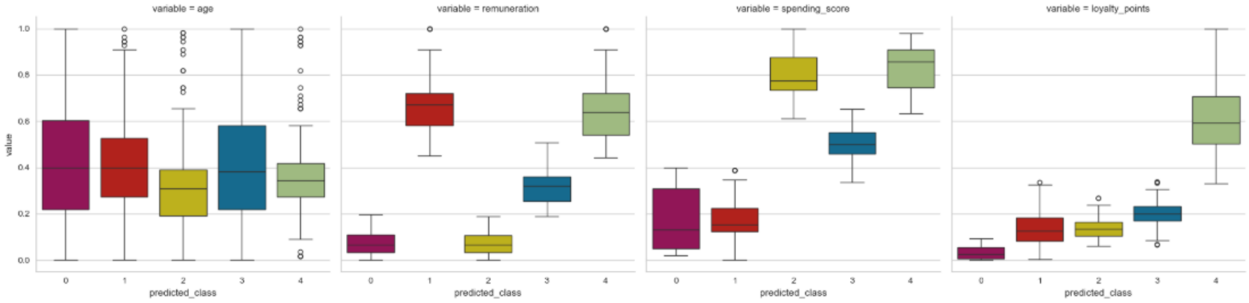
The silhouette score technique suggests the optimal k-value of 5 as the cut-off point that maximises the silhouette score. We evaluated models for k-value of 5 - 7:

	k-value=5	k-value=6	k-value=7
Observations per cluster			
min	119.000000	43.000000	43.000000
mean	156.400000	130.333333	111.714286
max	283.000000	282.000000	281.000000

k-value of 5 was selected as the optimal recognising lower variability in the observation counts per cluster and based on visual analysis (Appendix 8)

We utilised EDA and qualitative techniques to interpret clusters:

- Clusters 2 and 4 are associated with high sending scores, cluster 3 - with moderate, and clusters 0 and 1 - with low.
- Clusters 1 and 4 are associated with high remuneration, cluster 3 - with moderate, and clusters 0 and 2 - with low.



predicted_class	avg_remuneration	avg_spending_score	avg_loyalty_points	count_customers	remuneration_cat	spending_score_cat
0	19.574394	19.037879	246.643939	132	1	1
1	79.353950	17.815126	947.126050	119	3	1
2	19.311969	79.889764	980.519685	127	1	3
3	42.947138	49.893993	1389.890459	283	2	2
4	77.554380	81.752066	4197.024793	121	3	3

2.3. NLP

Model: VADER

Granularity: each 'review' is split into sentence tokens to improve accuracy for longer reviews (Appendix 10).

In addition to basic pre-processing, we removed words, where the sentiment assigned by the model does not make sense in gaming context.

```
words_to_exclude = ['enemies', 'die', 'playing', 'killed', 'helps', 'attacked', 'died',
'play', 'helping', 'killer', 'plays', 'killing', 'helped', 'played', 'cuts', 'villains', 'help'
, 'villain', 'paying', 'kill', 'attacking', 'battle', 'anger', 'battles', 'kills', 'pay', 'cuttin
g', 'pays', 'cut', 'attacks', 'enemy', 'attack']
```

We replaced frequent word collocations with implied strong sentiment, where the mo del assigned 'neutral', with alternative expressions conveying expected sentiment:

```
replace_dict =
{'zero stars': 'horrifying', 'one stars': 'bad', 'two stars': 'appalling', 'three
stars': 'neutral', 'four stars': 'fine', 'five stars': 'superb', 'six stars': 'superb',
'zero star': 'horrifying', 'one star': 'bad', 'two star': 'appalling', 'three star':
'neutral', 'four star': 'fine', 'five star': 'superb', 'six star': 'superb'}
```

These decisions were informed by the most frequent word analysis (Appendix 11).

There is no noticeable difference in patterns of sentiment score distribution across clusters and spending score categories (Appendix 12).

Average sentiment score per category (compound)

sentiment_cat	compound
negative	-0.360573
neutral	0.000000
positive	0.553523

Count by sentiment category (compound)

sentiment_cat	compound
negative	1003
neutral	2035
positive	4836

```
reviews_subset['compound'].describe()
```

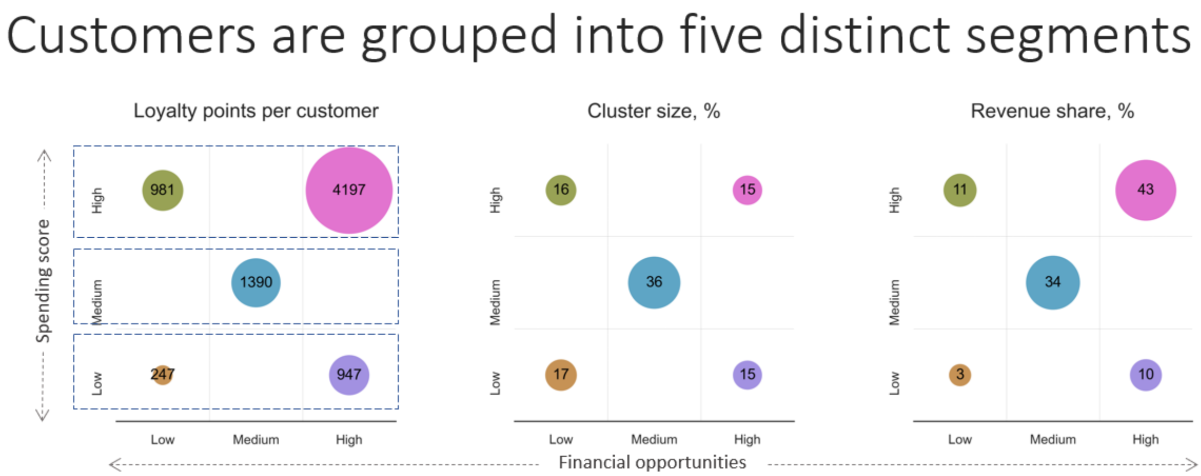
```
count    7874.000000
mean        0.294028
std         0.382643
min        -0.932500
25%         0.000000
50%         0.367700
75%         0.624900
max         0.995900
Name: compound, dtype: float64
```

III. Visualisation Approach

The intended target audience is the Turtle Games management, including CMO. Visualisations are designed to support conclusions. The choice of visualisation types is driven by the key messages they are intended to convey, i.e.:

- Scatterplot - to visualise relationships between continuous features and categorical (colour-coding)
- Histogram – to visualise distribution (e.g. sentiment score)

Example: used to explain the results of clustering analysis



Please refer to Appendix 13 to review the code used to prepare this visualisation.

Please refer to 4.3 for the biases and assumptions that might impact the interpretation.

IV. Insights patterns and recommendations

4.1. Observations and predictions

- Customers can be grouped into five clusters based on spending score & remuneration and three groups based on spending score category:
    - 'High' spending score:
        - Cluster 4: High-Value Customers (high-income)
        - Cluster 2: Limited-Potential Customers (low-income)
    - 'Moderate' spending score:
        - Cluster 3: Moderate spenders (average income)
    - 'Low' spending score:
        - Cluster 1: High-Potential Customers (high-income)
        - Cluster 0: Low-Value Customers (low-income)
- The 30-40 age group is the most common across all clusters. Customers over 40 appear underrepresented particularly in higher-earning clusters.
- Remuneration does not vary by gender, age or education level.
- MLR and DT suggest that remuneration and spending scores have a very strong positive correlation with loyalty points and best explain the variance in loyalty point balances.
- Spending-score-category-specific LR models satisfy mandatory LR assumptions and as such can produce unbiased estimates of loyalty points.
- The lines of the best fit in clusters with higher spending scores have higher independent variable coefficients (higher slope) and for this reason exhibit greater income elasticity.
- The best predictive power and accuracy are achieved when analysing spending score segments in isolation from each other. We recommend using ensemble techniques to predict loyalty points, incorporating both clustering and linear regression.
- Regression analysis indicates that increasing spending scores in cluster 1 and cluster 3 by one spending score category up to 50 and 80 points respectively can drive average loyalty points balances up to 2,910 and 1816, respectively, resulting in c.30% increase in total loyalty points.
-
- Sentiment analysis shows a wide range of scores, from -0.93 to 0.99, but positive sentiment dominates at 61%, with an average score of 0.55 in this group
- Customer sentiment scores and distribution patterns seem to be consistent across clusters and spending score categories.

4.2. Recommendations

- Existing Customer Campaigns
    - Focus on higher- and average-income clusters with lower spending scores targeting an increase in engagement:
        - Cluster 1: High growth potential with opportunities to increase loyalty points balances by 3–4 times.

        - Cluster 3 (Strategic Segment): This segment represents one-third of the customer base and revenue, warranting prioritisation despite moderate growth potential (up to 30%).
    - Cluster 4 (High-Value): Use premium loyalty programmes and personalised strategies to maximise retention and lifetime value.
    - Cluster 2 (Price-Sensitive): Deploy cost-saving promotions to engage this low-growth audience.
    - Cluster 0 (Low-Value): Limit marketing investment due to minimal purchasing power and growth potential.
- Leverage referral campaigns via Clusters 4 and 3 to drive cost-effective customer acquisition through existing networks.
- Monitor sentiment scores and Net Promoter Score (NPS) to evaluate campaign impact and loyalty trends.
- Use surveys to address negative feedback and recurring concerns including product quality, unclear rules, and complexity and to guide product improvements.

4.3. Recommendations for future data analysis, data collection

Sampling & biases:

- The non-random sample composition (10 reviews per product) may introduce bias, as it might not reflect the actual sales structure by cluster/product and misrepresent certain demographic groups or products (Appendix 3).
- The sample is limited to customers who left reviews, which may not represent the opinions of all relevant groups.

Data collection:

- There may be a technical issue in the product column. A brief analysis suggests that the product feature appears inconsistent, with a single ID being assigned to multiple different product types (Appendix 3).
- Introducing time-stamped customer reviews will allow Turtle Games to monitor changes over time (e.g., how sentiment evolves) and evaluate the impact of campaigns on customer satisfaction more effectively by comparing pre- and post-campaign data.
- When developing a marketing strategy by customer cluster, it is crucial to consider not only cluster characteristics but also cluster size in terms of customer count and revenue share. A more effective approach would involve analysing the contribution margin level to account for profitability.

Appendix 1: Libraries Utilised

```r
# Import the necessary libraries.
library(tidyverse)
library(skimr)
library(DataExplorer)
library(moments)
library(psych)
```

```python
# Limit the number of threads used by MKL to a manageable value (e.g., 4)
import os
os.environ['OMP_NUM_THREADS'] = '4'
```

```python
# Imports
import numpy as np
import pandas as pd
import math

# scipy
from scipy.stats import skew
from scipy.spatial.distance import cdist

# sklearn
import sklearn
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LinearRegression
from sklearn import tree
from sklearn.tree import DecisionTreeRegressor, plot_tree
from sklearn.cluster import KMeans
from sklearn.preprocessing import MinMaxScaler

from sklearn import metrics
#from sklearn.metrics import silhouette_score
#from sklearn.metrics import r2_score
#from sklearn.metrics import accuracy_score, confusion_matrix, ConfusionMatrixDisplay
#from sklearn.metrics import classification_report

# yellowbrick
from yellowbrick.cluster import KElbowVisualizer, SilhouetteVisualizer

# statsmodels
import statsmodels.api as sm
from statsmodels.formula.api import ols
from statsmodels.stats.outliers_influence import variance_inflation_factor
from statsmodels.graphics.gofplots import qqplot
import statsmodels.stats.api as sms

# visualisation
import seaborn as sns
import matplotlib.pyplot as plt
from mpl_toolkits.mplot3d import Axes3D
from matplotlib.ticker import Locator, MultipleLocator


#from IPython.display import HTML
```

```python
# Import all the necessary packages
import warnings

# Settings for the notebook.
warnings.filterwarnings("ignore")
```

```python
# display all numeric values in the DataFrame in standard numeric format
pd.set_option('display.float_format', '{:.6f}'.format)
```

d

```python
# Import all the necessary packages.

# general use
import pandas as pd
import numpy as np
import re
#from scipy.stats import norm
#import os

# visualisation
import seaborn as sns
import matplotlib.pyplot as plt


# pre-processing
# nltk
import nltk
from nltk.tokenize import sent_tokenize
from nltk.tokenize import word_tokenize
from nltk.corpus import stopwords
from nltk.corpus import words
from nltk.corpus import wordnet
from nltk.stem.wordnet import WordNetLemmatizer

# other pre-processing
from num2words import num2words
import contractions
import emoji

# Sentiment Analysis.
from nltk.sentiment.vader import SentimentIntensityAnalyzer
from textblob import TextBlob

# word clouds
from wordcloud import WordCloud


# Import Counter.
#from nltk.probability import FreqDist
#from collections import Counter

# Other
import warnings
```

Appendix 2: User-defined functions

```
# returns a summary of the goodness of fit metrics for a model.
def goodness_of_fit(y, y_pred, mlr_model, model_name_str):
    df = pd.DataFrame({'R2': [metrics.r2_score(y, y_pred)]})
    try:
        df['R2 (train)'] = mlr_model.rsquared
    except AttributeError:
        df['R2 (train)'] = None
    try:
        df['Adj. R2 (train)'] = mlr_model.rsquared_adj
    except AttributeError:
        df['Adj. R2 (train)'] = None
    df['RMSE'] = np.sqrt(metrics.mean_squared_error(y, y_pred))
    df['MAE'] = metrics.mean_absolute_error(y, y_pred)
    df['MAPE, %'] = np.mean(np.abs((y -  y_pred) / y_pred)) * 100
    df['q1 (y_real)'] = np.percentile(y, 25)
    df['q3 (y_real)'] = np.percentile(y, 75)
    df['mean (y_real)'] = y.mean()
    df = df.T
    df.columns = [model_name_str]
    return df
```

```
# Creates a descriptive statistics summary for a pandas DataFrame
def desc_stat_summary(df):
    numeric_df = df.select_dtypes(include=['number']).copy()
    summary = numeric_df.describe().T
    summary['range'] = summary['max'] - summary['min']
    summary['IQR'] = summary['75%'] - summary['25%']
    summary['skewness (n=0)'] = numeric_df.apply(stats.skew)
    # Standard Kurtosis
    summary['kurtosis (n=3)'] = numeric_df.apply(lambda x: stats.kurtosis(x, fisher=False))
    summary['Shapiro-Wilk, stat (n=1)'] = numeric_df.apply(lambda x: stats.shapiro(x)[0])
    summary['Shapiro-Wilk, p-value (>0.05)'] = numeric_df.apply(lambda x: stats.shapiro(x)[1])
    summary = summary.T
    return summary
```

```
# returns a summary of metrics to review LR assumptions.
def LR_statistics(mlr_model, X_train, model_name_str):
    # call the Durbin-Watson statistic (check for autocorrelation)
    dw_statistic = durbin_watson(mlr_model.resid)
    df = pd.DataFrame({'Durbin-Watson statistic (1.5-2.5)': [dw_statistic]})
    # Pearson correlation coefficients for each of the x-variables and residuals (check for heterogeneity)
    for i in X_train.columns:
        df[f"Resid. vs {i} corr"] = "{:.2f}".format(mlr_model.resid.corr(X_train[i]))
    # LM Test (heteroscedasticity)
    df['LM Test p-value (> 0.05)'] = sms.het_breuschpagan(mlr_model.resid, mlr_model.model.exog)[1]
    # check for multicollinearity
    for i in range(mlr_model.model.exog.shape[1]):
        df[f"VIF Factor: {mlr_model.model.exog_names[i]} (<10)"] = variance_inflation_factor(mlr_model.model.exog, i)
    # drop as not informative
    df = df.drop(columns = 'VIF Factor: const (<10)')
    # JB probability (normality of error terms)
    df['JB Probability (> 0.05)'] = jarque_bera(mlr_model.resid)[1]
    df['F statistic p-value (< 0.05)'] = mlr_model.f_pvalue
    df = df.T
    df.columns = [model_name_str]
    return df
```

(continued on the next page)

```python
# Provided function.
def generate_polarity(comment):
    '''Extract polarity score (-1 to +1) for each comment'''
    return TextBlob(comment).sentiment[0]
```

```python
# sentense tokens preprocessing: takes a string as an input and returns a 'clean' string,
# i.e. lowercasing, contractions, emojis, etc.
def preprocess_sent(text):
    # Convert to lowercase (VADER is case-sensitive)
    text = text.lower()
    # Expand contractions
    text = contractions.fix(text)
    # Convert emojis to text
    text = emoji.demojize(text)
    # Remove URLs
    text = re.sub(r'\s* \s*|<a [^>]*>(.*?)<\/a>', f" {r'\1'} ", text)
    text = re.sub(r'http[^ ]+', ' ', text)
    # Replace hyphens and slashes with spaces
    text = re.sub(r'[-/]', ' ', text)
    # Convert numbers to words
    text = " ".join(num2words(int(x)) if x.isdigit() else x for x in word_tokenize(text))
    # Remove extra whitespace
    text = re.sub(r'\s+', ' ', text).strip()
    # remove quotations
    text = re.sub(r'[\'"]', '', text)
    # spelling correction
    #text = str(TextBlob(text).correct())
    return text
```

```python
# word tokens preprocessing: takes a list of word tokens as input and
# returns a list of word tokens w/o stopwords and digits.
def preprocess_words(w_list):
    # Remove stopwords from word_tokens_clean
    w_list = [word for word in w_list if word not in english_stopwords]
    # remove digits
    w_list = [word for word in w_list if not re.search(r'\d', word)]
    return w_list
```

```python
# Function to map NLTK POS tags to WordNet POS tags
def get_wordnet_pos(treebank_tag):
    if treebank_tag.startswith('J'):  # Adjective
        return wordnet.ADJ
    elif treebank_tag.startswith('V'):  # Verb
        return wordnet.VERB
    elif treebank_tag.startswith('N'):  # Noun
        return wordnet.NOUN
    elif treebank_tag.startswith('R'):  # Adverb
        return wordnet.ADV
    else:
        return None
```

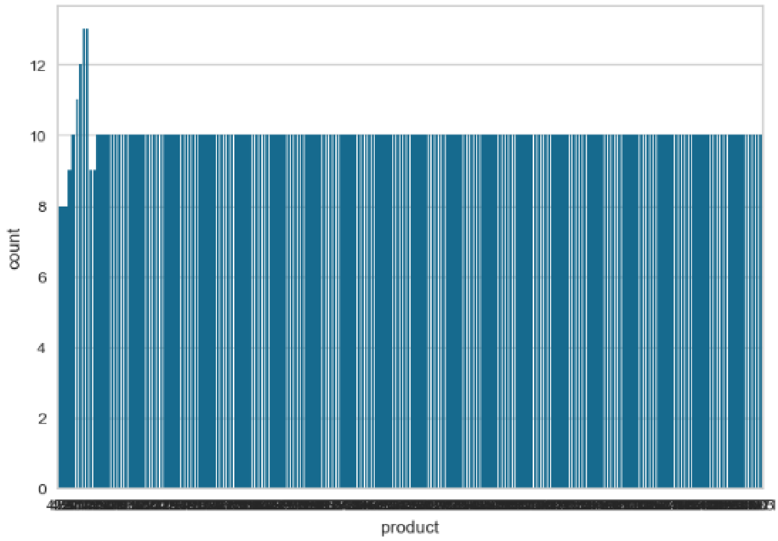
```python
# Define a function to replace multiple words in a string based on a dictionary
def replace_all(text, replace_dict):
    for key, value in replace_dict.items():
        text = text.replace(key, value)
    return text
```

Appendix 3: Product Feature

```
reviews_full[reviews_full['product'] == '10241'][['product', 'review']].style.set_properties(subset=['review'], **{'width': '1200px'})
```

	product	review
137	10241	This little Dover book is like the others of the series - fun and entertaining. I was not happy to see that the price on the back was 1.50andIpaid2.51 with Amazon. I am used to getting a better deal than the stores, but this certainly was not!
336	10241	Awesome fun with both my son and daughter. Thank you
536	10241	Great game that kids love to play, it's like Uno so most kids know how to play it already. Recommend.
733	10241	Consistently good quality product by this company. Nice graphics and sturdy enough for use by a young child.
929	10241	Let me start by saying,"This is a great game".I recomend it to anyone who enjoys board games or role playing games. It can be a family game also,for "Game Night",because it encourages cooperation with all players to win.(actually it forces cooperation in order to win).There are some things I would change,but I think everyone will have opinions on different ways to spawn monsters,events,etc....I just hope they develope it with every new edition. My game was missing the 20 sided die,I sent Wizards of the Coast an email and they responded right away and sent me one within a few days with a letter of apology. You can't ask for a better response. Keep up the good work. Bakis.
1124	10241	Great booster! Me and my friends play this game. A must have if you like axis and allies land miniatures. I got the priest tank Japanese engineer German motorcycle a rifleman and a Hungarian MG team.
1318	10241	A friend bought me Lords of Waterdeep as a gift and I loved it. Right away bought this expansion. This adds more spaces for meeples as well as corruption. However, you do not need to use all of it. We started without using the corruption and then added it in the next game. Depending how we feel, corruption will be added in or not. Very good game all in all. Wish there were more expansions.
1511	10241	very good stickers...but no mummy pig or daddy pig, but still very cute.
1700	10241	Kids of all ages like this learning toy.
1909	10241	We really did not enjoy this game.

```
<Axes: xlabel='product', ylabel='count'>
```



Observation:

The number of reviews per product ranges from 8 to 13 in the sample with the mode (most common) number of reviews left per product in the sample of 10.

Assumption:

This is likely a sample generated by selecting 10 reviews per product.

LIMITATION:

- We do not know if all the products are included
- We do not know the sales structure by product
- We do not know if this is representative of the population

WE SUSPECT Sample selection bias (non-random data is selected for statistical analysis);

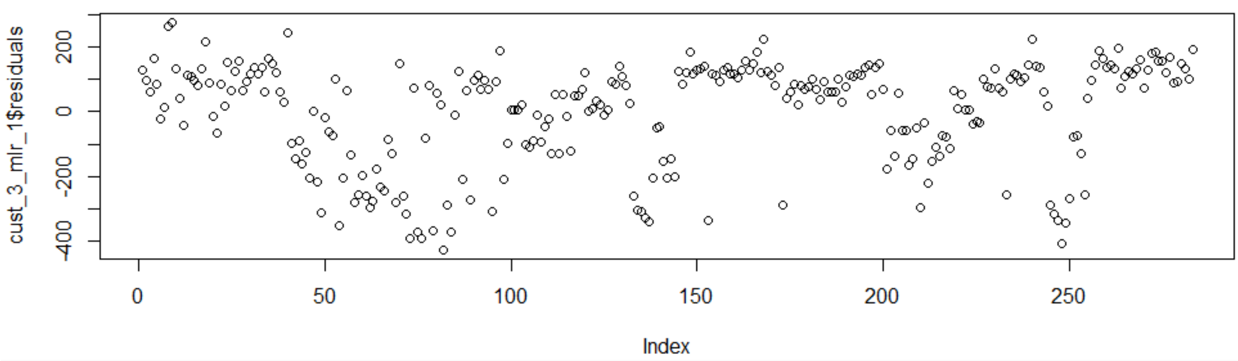
RECOMMEND: To perform inferential analysis (a statistical technique used to make conclusions or predictions about a population based on a sample of data) using hypothesis testing and confidence intervals.

Appendix 4: Linear regression in R

```
> # Create am MLR model
> cust_3_mlr_1 <- lm(loyalty_points ~ remuneration + spending_score + age,
+                    data=cust_3)
> summary(cust_3_mlr_1)

Call:
lm(formula = loyalty_points ~ remuneration + spending_score +
    age, data = cust_3)

Residuals:
    Min      1Q  Median      3Q     Max
-426.49  -96.04   62.44  117.03  275.58

Coefficients:
                 Estimate Std. Error t value Pr(>|t|)
(Intercept)    -1738.3798   109.7283  -15.84   <2e-16 ***
remuneration      33.6987     1.4782   22.80   <2e-16 ***
spending_score    27.1086     1.4599   18.57   <2e-16 ***
age                8.1479     0.6466   12.60   <2e-16 ***
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

Residual standard error: 158.8 on 279 degrees of freedom
Multiple R-squared:  0.7547,	Adjusted R-squared:  0.752
F-statistic: 286.1 on 3 and 279 DF,  p-value: < 2.2e-16

> # plot residuals
> plot(cust_3_mlr_1$residuals)
> |
```

Files | Plots | Packages | Help | Viewer | Presentation

Zoom | Export | Publish

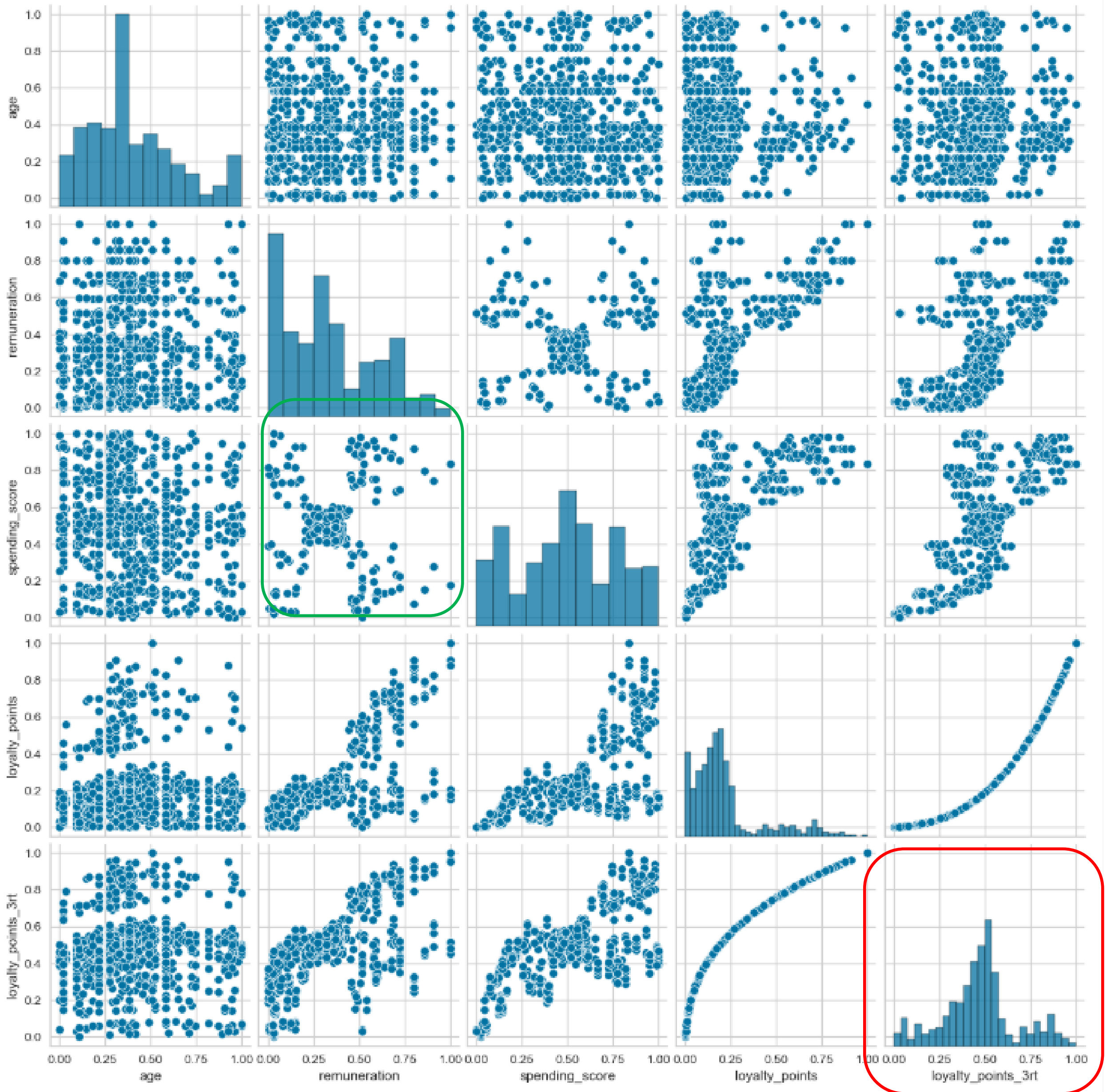



```
# A tibble: 15 × 6
   Statistic               cust_0  cust_1  cust_2  cust_3  cust_4
   <chr>                    <dbl>   <dbl>   <dbl>   <dbl>   <dbl>
 1 intercept              -254.   -733.  -1186.  -1738.  -4614.
 2 r_squared                 0.884   0.905   0.870   0.755   0.854
 3 adjusted_r_squared        0.882   0.904   0.867   0.752   0.850
 4 mae                      47.4   103.     80.0    130.    260.
 5 rmse                     60.5   152.    110.     158.    367.
 6 mean                    247.    947.    981.    1390.   4197.
 7 min                      25      40     436     478    2289
 8 max                     664    2325    1851    2332    6847
 9 q1                       69.8   586     734.   1176    3455
10 median                  198.    894     942    1395    4071
11 q3                      406    1278.   1150    1620.   4844
12 IQR                     336.    692.    416.    444.   1389
13 remuneration             13.7    10.2    49.7    33.7    54.3
14 spending_score           12.2    48.9    11.9    27.1    47.2
15 age                      NA      NA       7.02    8.15   19.5
>
```

Appendix 5: EDA

```
# visualise the distributions & relationships beetween numeric variables
sns.pairplot(customers_scaled)
```

```
<seaborn.axisgrid.PairGrid at 0x20d68b740e0>
```



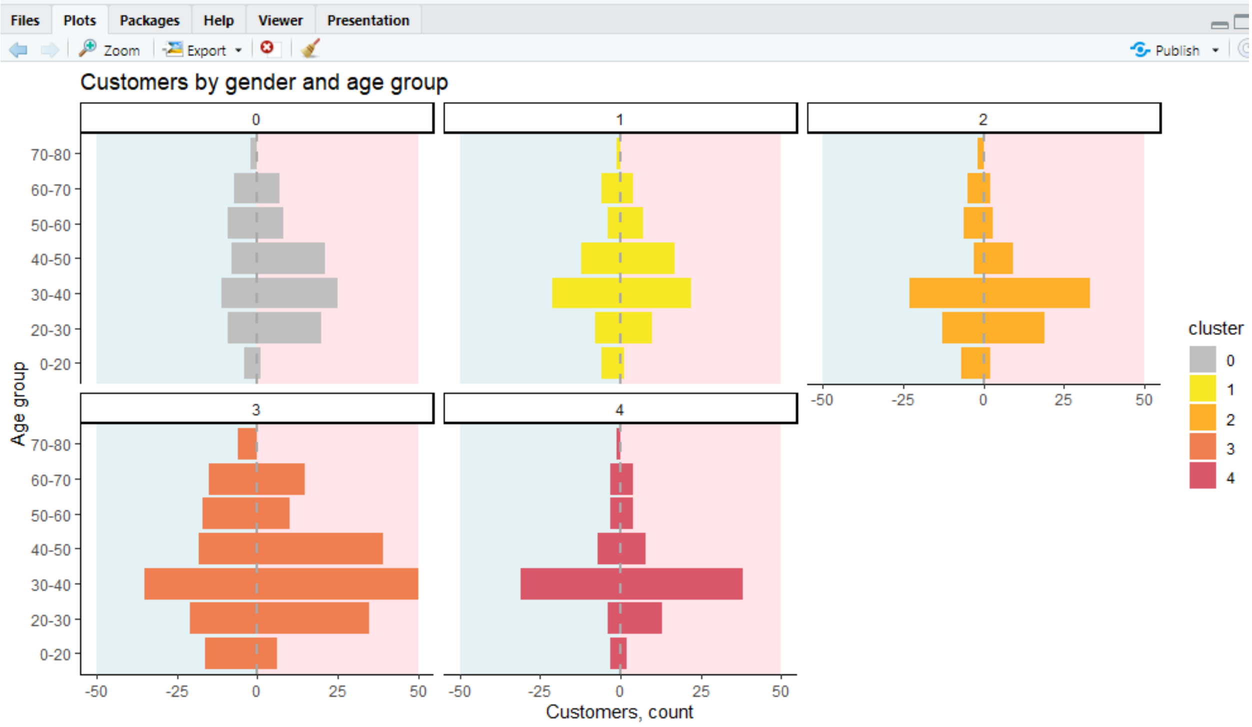
Clusters

Closer-to normal dist after cube-root transformation

Appendix 6: Cluster EDA

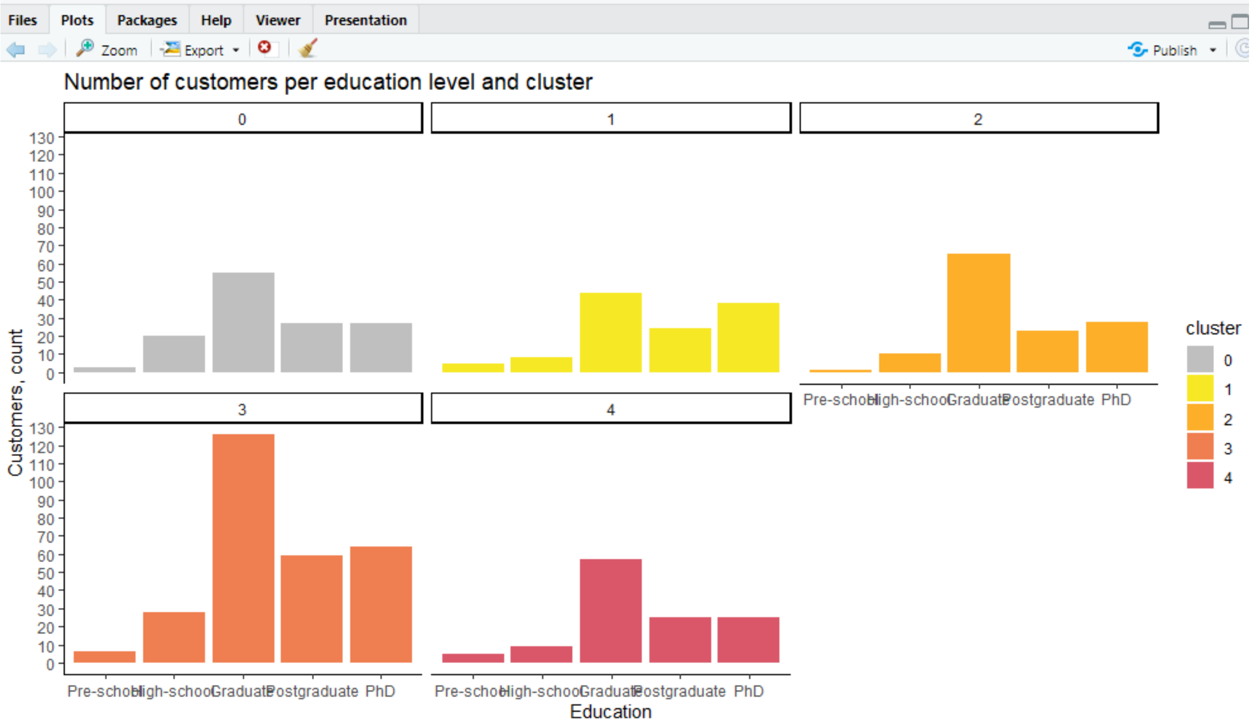
```r
## add gender_viz columns with male = -1 and female = 1
customers$gender_viz[which(customers$gender == 'Male')] <- -1
customers$gender_viz[which(customers$gender == 'Female')] <- 1
```

```r
> # Aggregate the data by age group and gender, summing the gender_viz values
> agg_data <- aggregate(gender_viz ~ age_group + gender + cluster, data = customers, sum)
> # Create the plot with aggregated data and custom background colours
> ggplot(agg_data, aes(x = age_group, y = gender_viz, fill = cluster)) +
+   # Add background colours first (behind the bars)
+   geom_rect(aes(xmin = -Inf, xmax = Inf, ymin = -50, ymax = 0), fill = "lightblue", alpha = 0.03) +
+   geom_rect(aes(xmin = -Inf, xmax = Inf, ymin = 0, ymax = 50), fill = "lightpink", alpha = 0.03) +
+   # Add the bars on top of the background
+   geom_bar(stat = "identity", position = "dodge") +
+   scale_fill_manual(values = c('#BFBFBF', '#F6E825', '#FDAF2A', '#EF7E50', '#D95769')) +  # Custom fill colours for clu
ster
+   facet_wrap(~cluster) +
+   labs(title = "Sum of Gender Viz by Age Group",
+        x = "Age Group",
+        y = "Sum of Gender Viz") +
+   coord_flip() +  # Flip the x-axis and y-axis
+   geom_hline(yintercept = 0, linetype = "dashed", color = "darkgrey", size = 1) +
+   # Set y-axis limits from -40 to 40
+   scale_y_continuous(limits = c(-50, 50)) +
+   labs(x = "Age group",
+        y = "Customers, count",
+        title = "Customers by gender and age group") +
+   theme_classic()
```



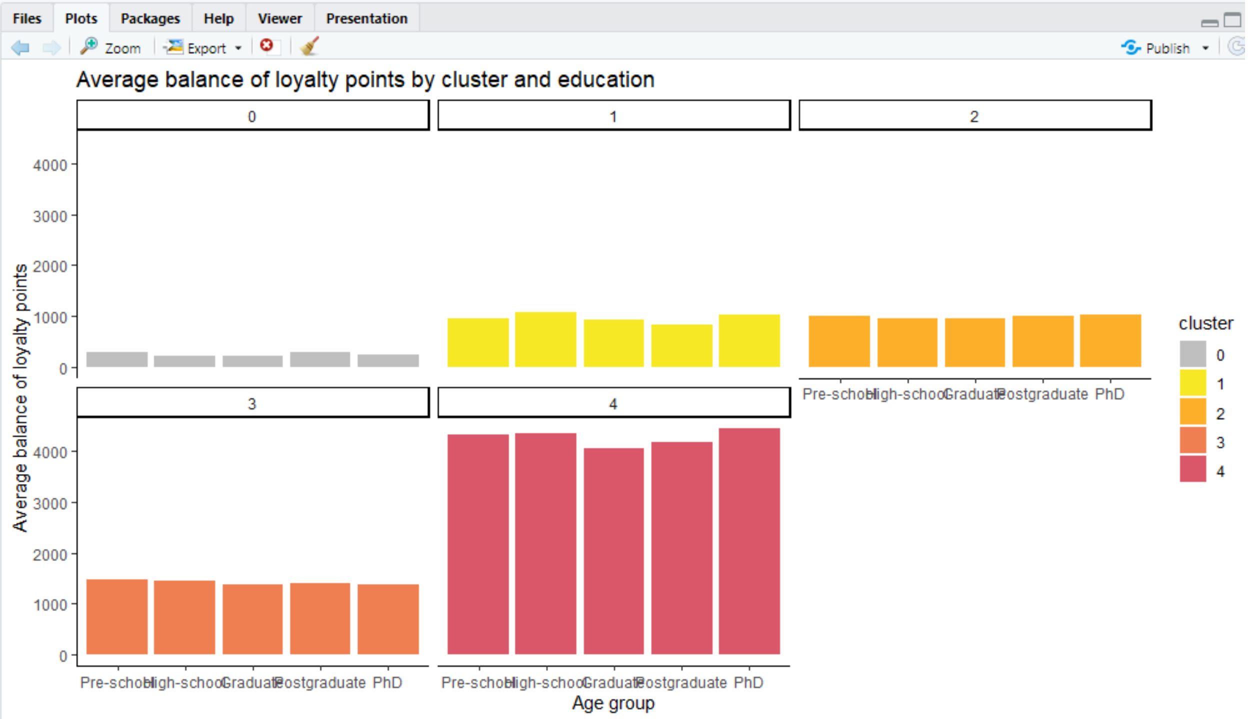
Clusters 1, 2, and 4 have a balanced gender distribution, while in clusters 0 and 3 we observe a larger share of women.

```
> # plot
> ggplot(customers, aes(x = education, fill = cluster)) +
+   geom_bar(position='dodge') +
+   facet_wrap(~cluster) +
+   scale_fill_manual(values = c('#BFBFBF', '#F6E825', '#FDAF2A', '#EF7E50', '#D95769')) +
+   scale_y_continuous(breaks = seq(0, 350, 10), "Customers, count") +
+   labs(title = "Number of customers per education level and cluster", x = "Education") +
+   theme_classic()
> 
```



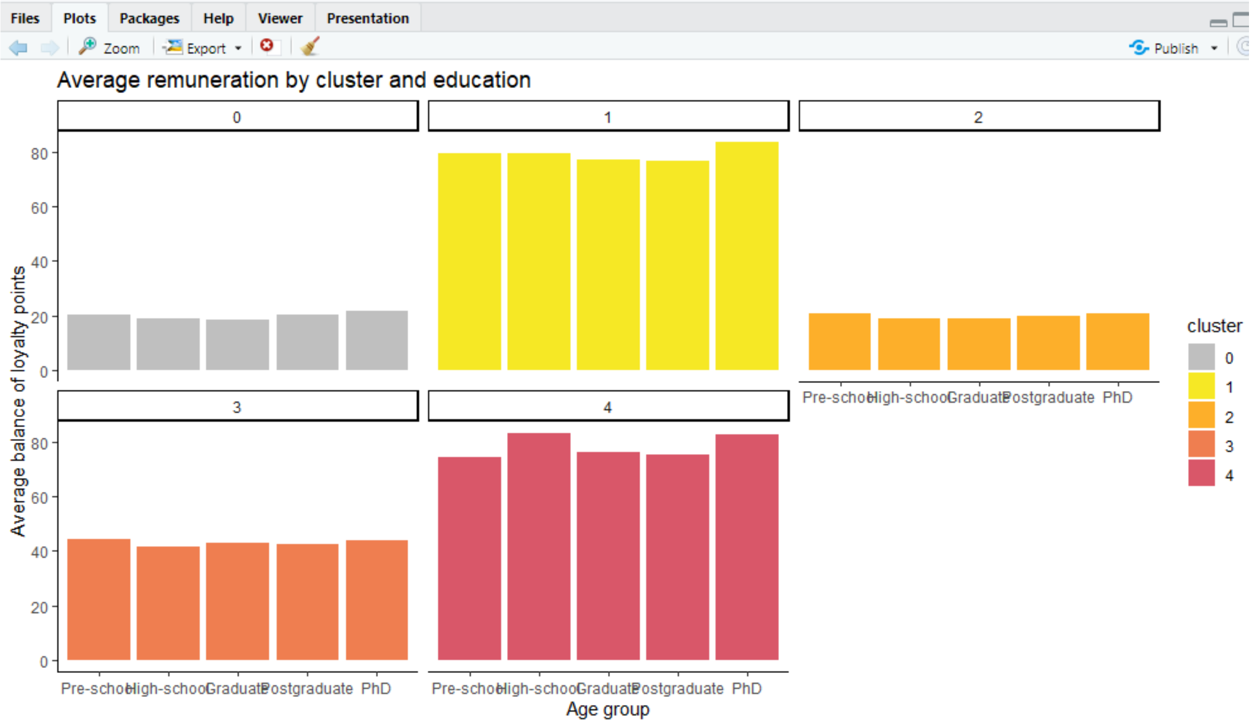
Most of the customers have a graduate degree or above. There is no notable difference in education level by cluster.

```
> # Aggregate the data by education and cluster, calculating average for the loyalty_points values
> agg_data <- aggregate(loyalty_points ~ education + cluster, data = customers, mean)
> # Create the plot with aggregated data and custom background colours
> ggplot(agg_data, aes(x = education, y = loyalty_points, fill = cluster)) +
+   # Add background colours first (behind the bars)
+   #  geom_rect(data = subset(agg_data, cluster == 1), aes(xmin = -Inf, xmax = Inf, ymin = -Inf, ymax = Inf), fill = '#F
DAF2A', alpha = 0.1) +
+   #  geom_rect(data = subset(agg_data, cluster == 4), aes(xmin = -Inf, xmax = Inf, ymin = -Inf, ymax = Inf), fill = '#D
95769', alpha = 0.1) +
+   #  geom_rect(aes(xmin = -Inf, xmax = Inf, ymin = 0, ymax = 40), fill = "lightpink", alpha = 0.03) +
+   # Add the bars on top of the background
+   geom_bar(stat = "identity", position = "dodge") +
+   scale_fill_manual(values = c('#BFBFBF', '#F6E825', '#FDAF2A', '#EF7E50', '#D95769')) +  # Custom fill colours for clu
ster
+   facet_wrap(~cluster) +
+   labs(title = "Sum of Gender Viz by Age Group",
+        x = "Age Group",
+        y = "Sum of Gender Viz") +
+   #  geom_hline(yintercept = 0, linetype = "dashed", color = "darkgrey", size = 1) +
+   labs(x = "Age group",
+        y = "Average balance of loyalty points",
+        title = "Average balance of loyalty points by cluster and education") +
+   theme_classic()
>
```



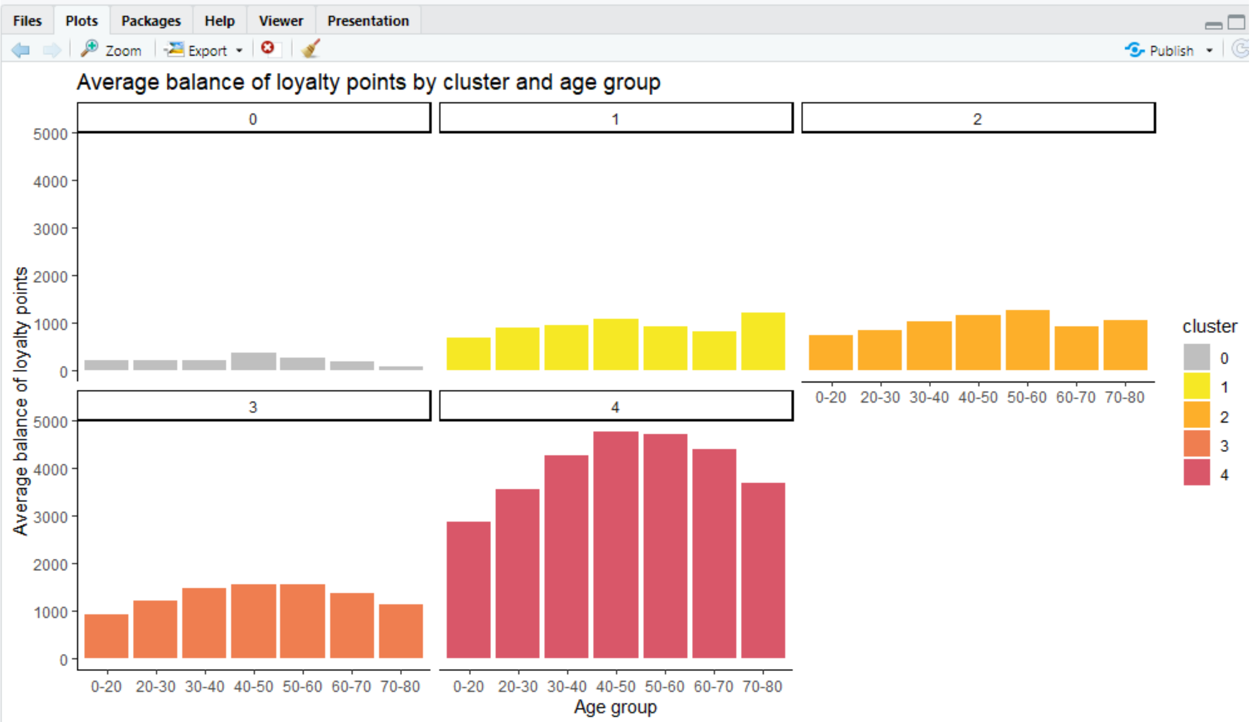
No variability in loyalty points by education within clusters. Loyalty points do not depend on education level (rather clusters).

```
> agg_data <- aggregate(remuneration ~ education + cluster, data = customers, mean)
> # Create the plot with aggregated data and custom background colours
> ggplot(agg_data, aes(x = education, y = remuneration, fill = cluster)) +
+   # Add background colours first (behind the bars)
+   #  geom_rect(data = subset(agg_data, cluster == 1), aes(xmin = -Inf, xmax = Inf, ymin = -Inf, ymax = Inf), fill = '#F
DAF2A', alpha = 0.1) +
+   #  geom_rect(data = subset(agg_data, cluster == 4), aes(xmin = -Inf, xmax = Inf, ymin = -Inf, ymax = Inf), fill = '#D
95769', alpha = 0.1) +
+   #  geom_rect(aes(xmin = -Inf, xmax = Inf, ymin = 0, ymax = 40), fill = "lightpink", alpha = 0.03) +
+   # Add the bars on top of the background
+   geom_bar(stat = "identity", position = "dodge") +
+   scale_fill_manual(values = c('#BFBFBF', '#F6E825', '#FDAF2A', '#EF7E50', '#D95769')) +  # Custom fill colours for clu
ster
+   facet_wrap(~cluster) +
+   labs(title = "Sum of Gender Viz by Age Group",
+        x = "Age Group",
+        y = "Sum of Gender Viz") +
+   #  geom_hline(yintercept = 0, linetype = "dashed", color = "darkgrey", size = 1) +
+   labs(x = "Age group",
+        y = "Average balance of loyalty points",
+        title = "Average remuneration by cluster and education") +
+   theme_classic()
> |
```



Average remuneration by education level within clusters does not demonstrate any patterns.

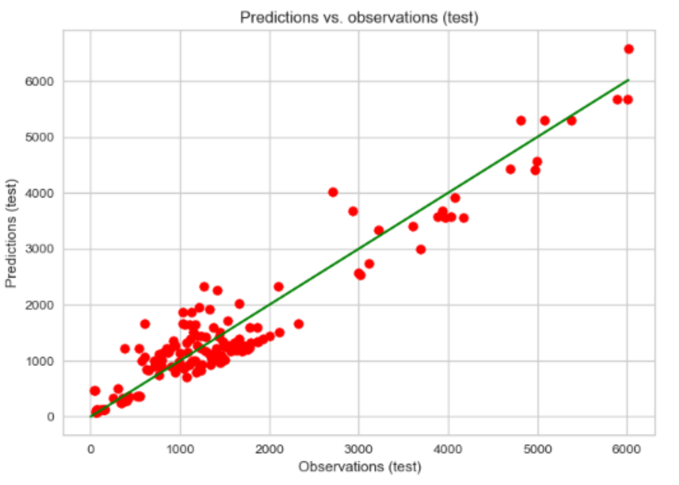
```
> # Aggregate the data by age group and cluster, calculating average for the loyalty_points values
> agg_data <- aggregate(loyalty_points ~ age_group + cluster, data = customers, mean)
> # Create the plot with aggregated data and custom background colours
> ggplot(agg_data, aes(x = age_group, y = loyalty_points, fill = cluster)) +
+   # Add background colours first (behind the bars)
+   #  geom_rect(data = subset(agg_data, cluster == 1), aes(xmin = -Inf, xmax = Inf, ymin = -Inf, ymax = Inf), fill = '#F
DAF2A', alpha = 0.1) +
+   #  geom_rect(data = subset(agg_data, cluster == 4), aes(xmin = -Inf, xmax = Inf, ymin = -Inf, ymax = Inf), fill = '#D
95769', alpha = 0.1) +
+   #  geom_rect(aes(xmin = -Inf, xmax = Inf, ymin = 0, ymax = 40), fill = "lightpink", alpha = 0.03) +
+   # Add the bars on top of the background
+   geom_bar(stat = "identity", position = "dodge") +
+   scale_fill_manual(values = c('#BFBFBF', '#F6E825', '#FDAF2A', '#EF7E50', '#D95769')) +  # Custom fill colours for clu
ster
+   facet_wrap(~cluster) +
+   labs(title = "Sum of Gender Viz by Age Group",
+        x = "Age Group",
+        y = "Sum of Gender Viz") +
+   #  geom_hline(yintercept = 0, linetype = "dashed", color = "darkgrey", size = 1) +
+   labs(x = "Age group",
+        y = "Average balance of loyalty points",
+        title = "Average balance of loyalty points by cluster and age group") +
+   theme_classic()
> |
```



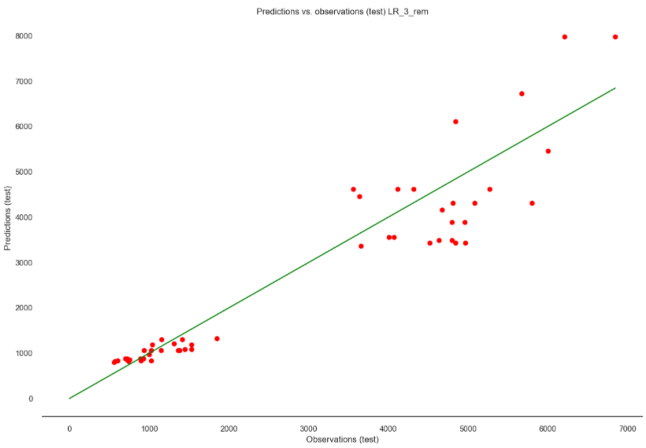
There is some variability in loyalty points by age groups within clusters, particularly in cluster 4, but in cluster 4 all the age groups except for 30-40 are insignificant in terms of customer count and for this reason this pattern might be not meaningful.

Appendix 7: Predictions and observations plots

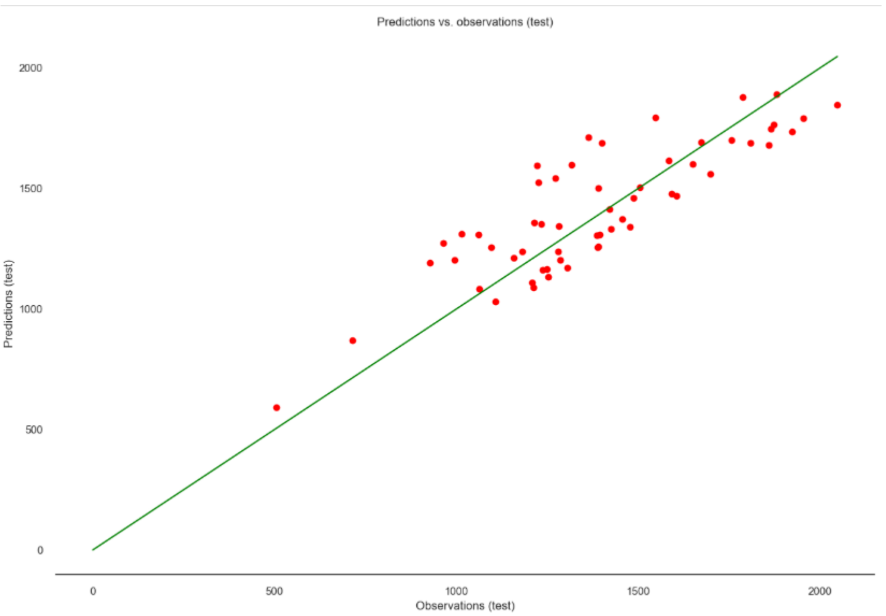
MLR_all



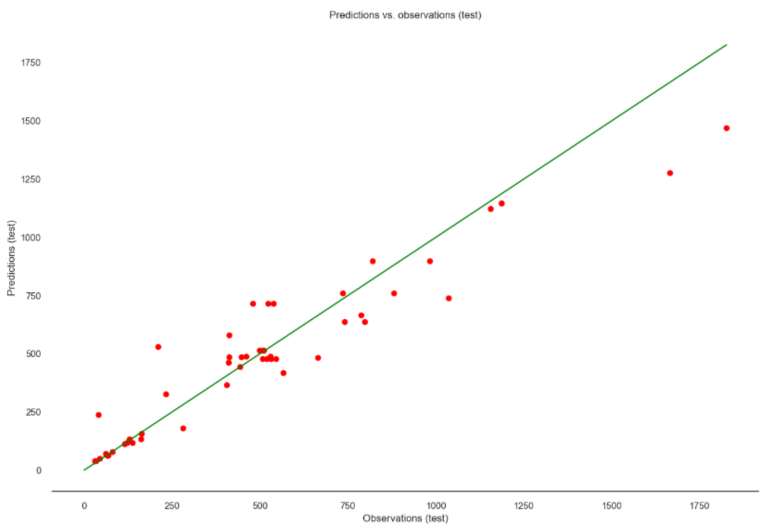
LR_3_rem



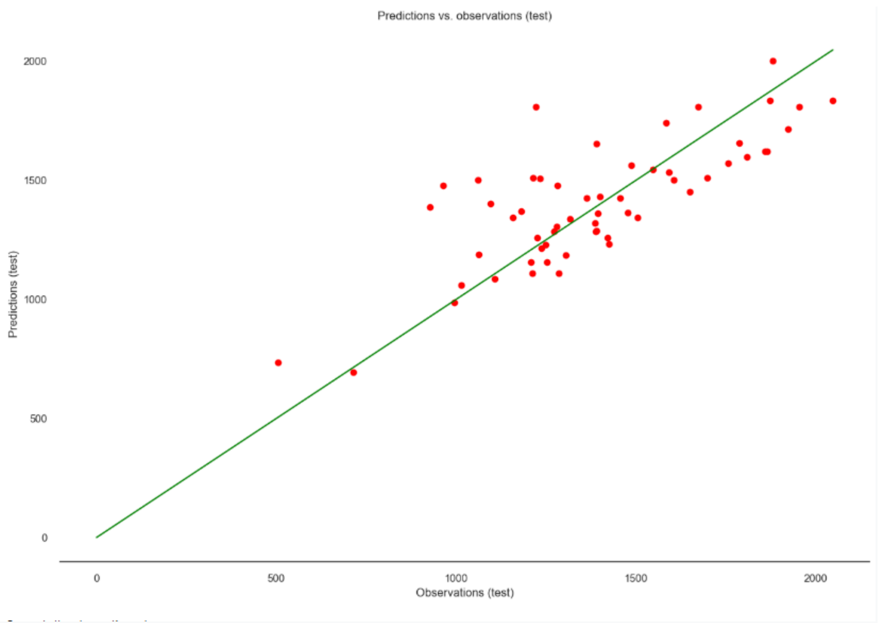
MLR_2



MLR_1

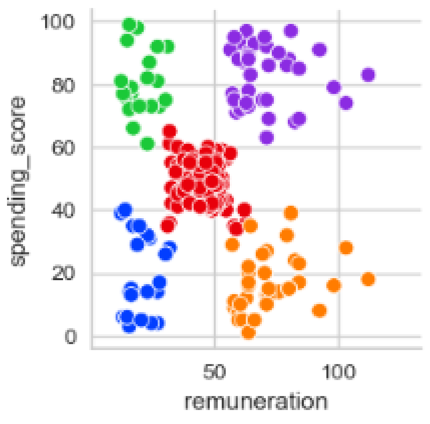


MLR_2_remss_test

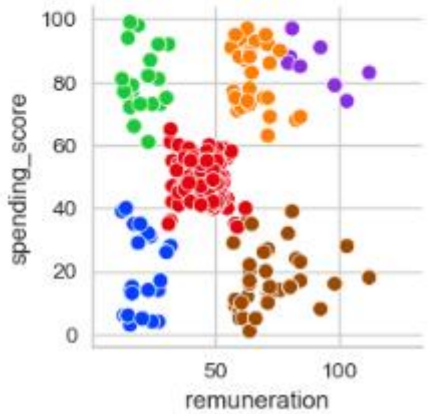


Appendix 8: The optimal k-value analysis

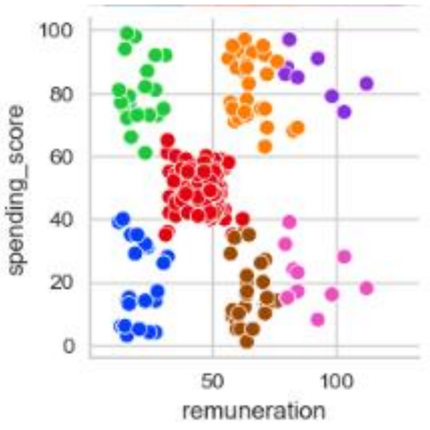
k-value = 5



k-value = 6



k-value=7



Appendix 10: Selected review exhibit

```
# Transform the DataFrame: explode
reviews_subset = reviews_subset.explode('sent_tokens', ignore_index=True)

# Display the transformed DataFrame
reviews_subset.head(4).style.set_properties(subset=['review', 'sent_tokens'], **{'width': '1200px'})
```

	cluster	review_id	review	sent_tokens
0	0	0	When it comes to a DM's screen, the space on the screen itself is at an absolute premium. The fact that 50% of this space is wasted on art (and not terribly informative or needed art as well) makes it completely useless. The only reason that I gave it 2 stars and not 1 was that, technically speaking, it can at least still stand up to block your notes and dice rolls. Other than that, it drops the ball completely.	When it comes to a DM's screen, the space on the screen itself is at an absolute premium.
1	0	0	When it comes to a DM's screen, the space on the screen itself is at an absolute premium. The fact that 50% of this space is wasted on art (and not terribly informative or needed art as well) makes it completely useless. The only reason that I gave it 2 stars and not 1 was that, technically speaking, it can at least still stand up to block your notes and dice rolls. Other than that, it drops the ball completely.	The fact that 50% of this space is wasted on art (and not terribly informative or needed art as well) makes it completely useless.
2	0	0	When it comes to a DM's screen, the space on the screen itself is at an absolute premium. The fact that 50% of this space is wasted on art (and not terribly informative or needed art as well) makes it completely useless. The only reason that I gave it 2 stars and not 1 was that, technically speaking, it can at least still stand up to block your notes and dice rolls. Other than that, it drops the ball completely.	The only reason that I gave it 2 stars and not 1 was that, technically speaking, it can at least still stand up to block your notes and dice rolls.
3	0	0	When it comes to a DM's screen, the space on the screen itself is at an absolute premium. The fact that 50% of this space is wasted on art (and not terribly informative or needed art as well) makes it completely useless. The only reason that I gave it 2 stars and not 1 was that, technically speaking, it can at least still stand up to block your notes and dice rolls. Other than that, it drops the ball completely.	Other than that, it drops the ball completely.

```
reviews_subset.shape
```

```
(7874, 9)
```

```
# call VADER polarity scores and store in separate variables
reviews_subset['vader'] = reviews_subset['sent_tokens_clean'].apply(lambda x: list(sia.polarity_scores(x).values()))
reviews_subset[['neg', 'neu', 'pos', 'compound']] = reviews_subset['vader'].apply(pd.Series).values
reviews_subset.drop(columns='vader', inplace=True)

# Display the transformed DataFrame
reviews_subset[['cluster', 'review_id', 'sent_tokens', 'sent_tokens_clean', 'word_tokens_clean', 'neg', 'neu', 'pos', 'compound']].head(3)\
.style.set_properties(subset=['sent_tokens', 'sent_tokens_clean', 'word_tokens_clean'], **{'width': '1200px'})
```

	cluster	review_id	sent_tokens	sent_tokens_clean	word_tokens_clean	neg	neu	pos	compound
0	0	0	When it comes to a DM's screen, the space on the screen itself is at an absolute premium.	when it comes to a dm s screen , the space on the screen itself is at an absolute premium .	['comes', 'dm', 'screen', 'space', 'screen', 'absolute', 'premium']	0.000000	1.000000	0.000000	0.000000
1	0	0	The fact that 50% of this space is wasted on art (and not terribly informative or needed art as well) makes it completely useless.	the fact that fifty % of this space is wasted on art (and not terribly informative or needed art as well) makes it completely useless .	['fact', 'fifty', 'space', 'wasted', 'art', 'terribly', 'informative', 'needed', 'art', 'well', 'makes', 'completely', 'useless']	0.201000	0.639000	0.160000	-0.311400
2	0	0	The only reason that I gave it 2 stars and not 1 was that, technically speaking, it can at least still stand up to block your notes and dice rolls.	the only reason that i gave it appalling and not one was that , technically speaking , it can at least still stand up to block your notes and dice rolls .	['reason', 'gave', 'two', 'stars', 'one', 'technically', 'speaking', 'least', 'still', 'stand', 'block', 'notes', 'dice', 'rolls']	0.172000	0.828000	0.000000	-0.659700

Appendix 11: Use-case-specific keywords analysis

```
# create a frequency distribution table with words, lemmas and sentiment
# create a variable for word_tokens before lemmatisation
words = reviews_subset['word_tokens_clean'].copy()

# derive the number of unique words and store in a separate column
words = words.reset_index()
words.columns = ['count', 'word']
words = words.explode('word', ignore_index=True)
words = words.dropna()
words = words.groupby(by='word').count().sort_values(by='count', ascending=False).reset_index()

# populate the lemma column
words['lemma'] = words['word'].apply(lambda x: lemmatiser.lemmatize(x, pos=get_wordnet_pos(nltk.pos_tag([x])[0][1])) \
                  if get_wordnet_pos(nltk.pos_tag([x])[0][1]) else lemmatiser.lemmatize(x))

# call VADER polarity scores and store in separate variables based on word (not lemma)
words['vader'] = words['word'].apply(lambda x: list(sia.polarity_scores(x).values()))
words[['neg', 'neu', 'pos', 'compound']] = words['vader'].apply(pd.Series).values
words.drop(columns='vader', inplace=True)

# reorder columns
words = words[['word', 'lemma', 'count', 'neg', 'neu', 'pos', 'compound']]

# view
words.head(2)
```

```
# view most frequent negative words
words[words.neg == 1].head(25)
```

	word	lemma	count	neg	neu	pos	compound
76	anger	anger	99	1.0	0.0	0.0	-0.5719
107	hard	hard	80	1.0	0.0	0.0	-0.1027
199	difficult	difficult	50	1.0	0.0	0.0	-0.3612
284	disappointed	disappointed	38	1.0	0.0	0.0	-0.4767
361	bad	bad	31	1.0	0.0	0.0	-0.5423
442	problem	problem	25	1.0	0.0	0.0	-0.4019
465	boring	boring	24	1.0	0.0	0.0	-0.3182
487	lost	lose	23	1.0	0.0	0.0	-0.3182
551	alone	alone	20	1.0	0.0	0.0	-0.2500
552	attack	attack	20	1.0	0.0	0.0	-0.4767
571	risk	risk	19	1.0	0.0	0.0	-0.2732
577	cut	cut	19	1.0	0.0	0.0	-0.2732
601	frustrating	frustrate	18	1.0	0.0	0.0	-0.4404
618	battle	battle	17	1.0	0.0	0.0	-0.3818
632	bored	bore	17	1.0	0.0	0.0	-0.2732
641	mess	mess	17	1.0	0.0	0.0	-0.3612
644	die	die	17	1.0	0.0	0.0	-0.5994
665	limited	limited	16	1.0	0.0	0.0	-0.2263
683	low	low	16	1.0	0.0	0.0	-0.2732
704	missing	miss	15	1.0	0.0	0.0	-0.2960
717	wrong	wrong	15	1.0	0.0	0.0	-0.4767
743	pay	pay	14	1.0	0.0	0.0	-0.1027
757	damage	damage	14	1.0	0.0	0.0	-0.4939
758	negative	negative	14	1.0	0.0	0.0	-0.5719
766	disappointing	disappoint	14	1.0	0.0	0.0	-0.4939

```
# view most frequent positive words
words[words.pos == 1].head(25)
```

	word	lemma	count	neg	neu	pos	compound
1	great	great	596	0.0	0.0	1.0	0.6249
3	fun	fun	553	0.0	0.0	1.0	0.5106
4	play	play	502	0.0	0.0	1.0	0.3400
5	like	like	414	0.0	0.0	1.0	0.3612
7	love	love	331	0.0	0.0	1.0	0.6369
12	good	good	294	0.0	0.0	1.0	0.4404
16	well	well	262	0.0	0.0	1.0	0.2732
26	playing	play	224	0.0	0.0	1.0	0.2023
28	played	played	207	0.0	0.0	1.0	0.3400
49	loves	love	146	0.0	0.0	1.0	0.5719
51	easy	easy	137	0.0	0.0	1.0	0.4404
52	nice	nice	131	0.0	0.0	1.0	0.4215
54	better	well	129	0.0	0.0	1.0	0.4404
55	recommend	recommend	129	0.0	0.0	1.0	0.3612
64	cute	cute	113	0.0	0.0	1.0	0.4588
73	loved	love	100	0.0	0.0	1.0	0.5994
74	gift	gift	100	0.0	0.0	1.0	0.4404
80	want	want	98	0.0	0.0	1.0	0.0772
89	enjoy	enjoy	92	0.0	0.0	1.0	0.4939
92	friends	friend	89	0.0	0.0	1.0	0.4767
94	best	best	88	0.0	0.0	1.0	0.6369
100	pretty	pretty	85	0.0	0.0	1.0	0.4939
108	help	help	80	0.0	0.0	1.0	0.4019
118	definitely	definitely	74	0.0	0.0	1.0	0.4019
128	worth	worth	72	0.0	0.0	1.0	0.2263

```
[ ]: # review words that came to our attention [monster, enemy, kill, hit / hitting]

     # List of words to filter
     filter_words = ['monst', 'villain', 'enem', ' kill', ' hit ', 'hitting']

     # Create a regex pattern from the filter words
     pattern = '|'.join(filter_words)

     # Filter the DataFrame using str.contains
     words[words['word'].str.contains(pattern, regex=True)]
```

[ ]:

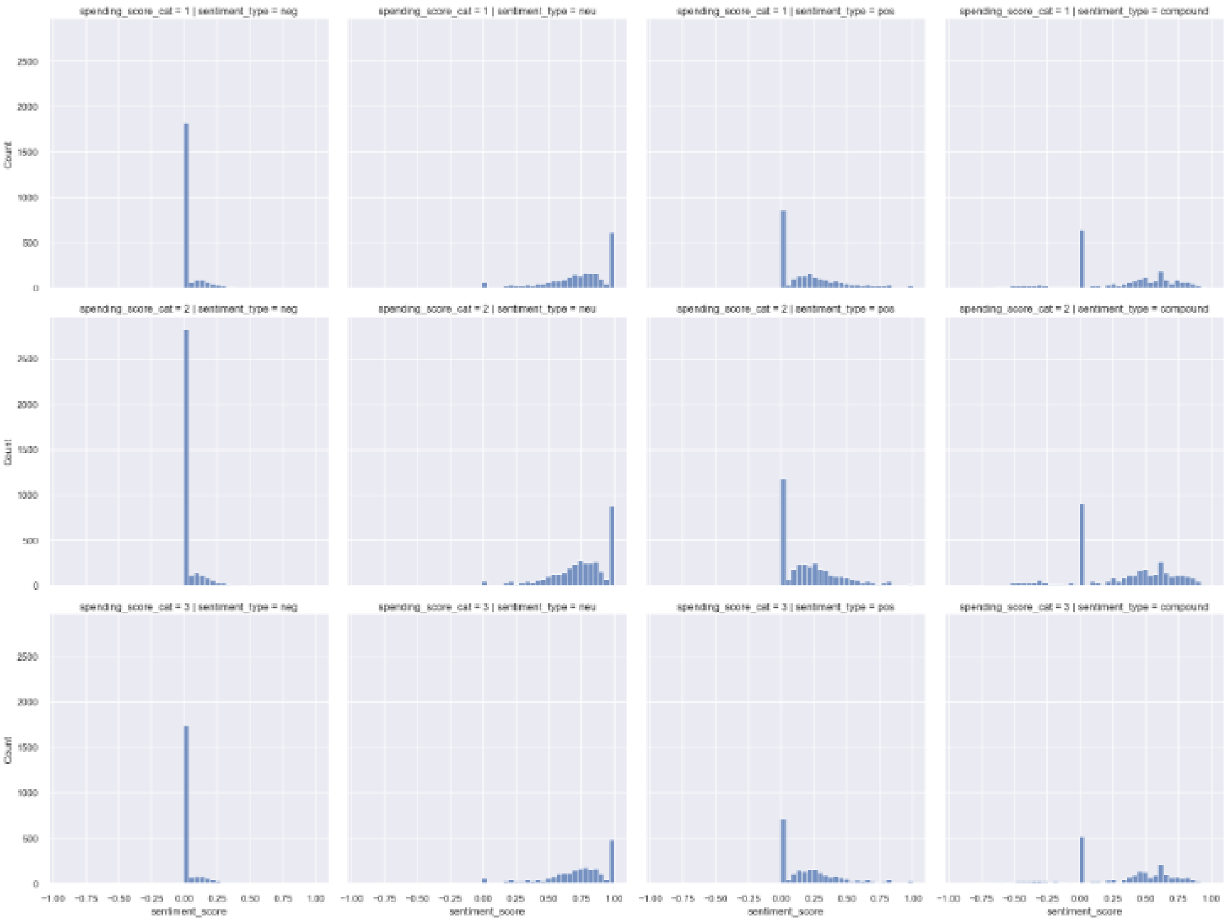
	word	lemma	count	neg	neu	pos	compound
182	monsters	monster	55	0.0	1.0	0.0	0.0000
225	monster	monster	46	0.0	1.0	0.0	0.0000
896	enemies	enemy	11	1.0	0.0	0.0	-0.4939
2991	enemy	enemy	2	1.0	0.0	0.0	-0.5423
3020	villain	villain	2	1.0	0.0	0.0	-0.5574
4035	villains	villain	1	1.0	0.0	0.0	-0.6597
4870	demonstration	demonstration	1	0.0	0.0	1.0	0.1027
4871	demonstrating	demonstrate	1	0.0	1.0	0.0	0.0000
5986	monstervillain	monstervillain	1	0.0	1.0	0.0	0.0000
5993	monsterseventsetci	monsterseventsetci	1	0.0	1.0	0.0	0.0000
7002	hitting	hit	1	0.0	1.0	0.0	0.0000

```
[554]: i = 'gave two stars'

[555]: sia.polarity_scores(i)

[555]: {'neg': 0.0, 'neu': 1.0, 'pos': 0.0, 'compound': 0.0}

[558]: i = 'gave five stars'

[559]: sia.polarity_scores(i)

[559]: {'neg': 0.0, 'neu': 1.0, 'pos': 0.0, 'compound': 0.0}
```

Appendix 12: Sentiment distribution

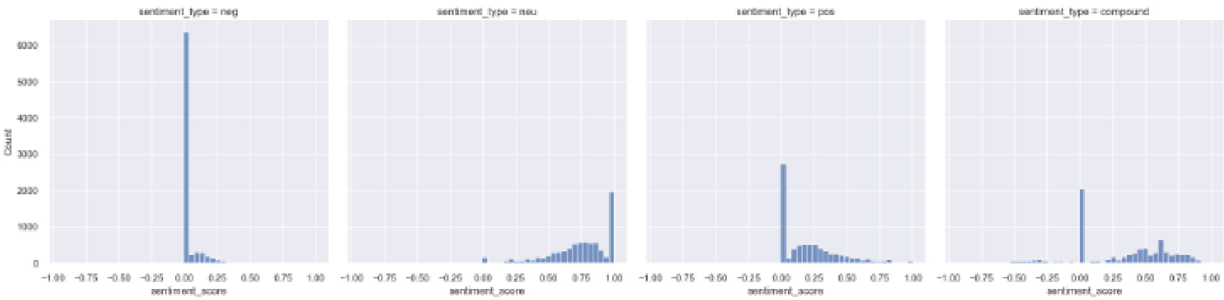
```
# sentiment score distribution by sentiment type and spending_score-category
sns.displot(data = reviews_subset_long, \
            x = 'sentiment_score', row = 'spending_score_cat', col= 'sentiment_type' , kind='hist')
```

```
<seaborn.axisgrid.FacetGrid at 0x1cf8d3423f0>
```



```
# sentiment score distribution by sentiment type
sns.displot(data = reviews_subset_long, x = 'sentiment_score', col= 'sentiment_type' , kind='hist')
```

```
<seaborn.axisgrid.FacetGrid at 0x1cf843c7230>
```



Appendix 13: Selected visualisation code

```python
# Visualise average loyalty points per customer by cluster
# Plot the 5 observation points (average loyalty points per customer by cluster) on a scatterplot; point size is reflective of the variable
fig, ax = plt.subplots(figsize=(5, 5))
param = 'avg_loyalty_points'

sns.scatterplot(data=summary, x='remuneration_cat', y='spending_score_cat', size=param, sizes=(500, 8000), legend=False,\
                hue='predicted_class',  hue_order=summary.predicted_class.values, palette=palette)

# Add labels to each point displaying 'avg_loyalty_points'
for i in range(len(summary)):
    label_color = 'white' if summary[param][i] >= 5000 else 'black'
    plt.text(
        summary['remuneration_cat'][i],         # x position of the label
        summary['spending_score_cat'][i],       # y position of the label
        f'{"{:.0f}".format(summary[param][i])}',  # label text: avg_loyalty_points value
        horizontalalignment='center',
        verticalalignment='center',
        fontsize=15,
        color=label_color
    )

# Set the x and y axis limits and tick positions and names
plt.xlim(0.5, 3.5)
plt.ylim(0.5, 3.5)
plt.xticks([1, 2, 3])
plt.yticks([1, 2, 3])
plt.xticks([1, 2, 3], ['Low', 'Medium', 'High'], fontsize=13)  # Custom x-axis tick marks
plt.yticks([1, 2, 3], ['Low', 'Medium', 'High'], rotation=90, fontsize=13)  # Custom y-axis tick marks

# Remove gridlines
plt.grid(False)


# Add vertical and horizontal lines
plt.axvline(x=1.5, color='grey', linestyle='--', linewidth=0.2)
plt.axvline(x=2.5, color='grey', linestyle='--', linewidth=0.2)
plt.axhline(y=1.5, color='grey', linestyle='--', linewidth=0.2)
plt.axhline(y=2.5, color='grey', linestyle='--', linewidth=0.2)

# Rename x and y axes add chart title
plt.xlabel('', fontsize=15)  # x-axis label
plt.ylabel('', fontsize=15)  # y-axis label
plt.title('Loyalty points per customer\n', fontsize=20)

plt.savefig(f"{param}.png", dpi=300, bbox_inches='tight')  # Save as PNG with high resolution
```