School of Informatics & IT
TEMASEK POLYTECHNIC

Big Data Programming
CBG1C04
Assignment

Submitted by

MEENAKSHI DEKSHINAMURTHY (2080548G)

04/03/2021

Contents

1. THE APPLICATION DOMAIN

- Hyper Casual games are easy to play and free to play. Their revenues are based mainly on Ads and in game purchases.
- In the Fast-moving world, the hyper casual games are addicting players of all age through their simple user interface and short gaming duration and no pre-requisite for any skill.
- With increasing users, a huge data is available, which can be analysed to derive and predict meaningful information.

2. BACKGROUND:

- An IT company releasing a variety of themes on hyper causal games is interested to find out the customer segments based on interests so that they can increase revenue by investing on games and features that interests the customer. It helps to introduce in game purchases and ads to suit players age and preference which increases game monetisation

3. RESEARCH GOALS

- Customer Profile based on location
- Games of interest by age

4. VALUES

- Improve profit by investing in customer choice of theme for the games
- Implement in game purchases by analysing customer segmentation to generate revenue
- Find the age group of customer with low bounce rate or more Avg Session Time for a new game, so that appropriate ads and features can be hosted to improve gamer retention

5. DATA SOURCE AND SCHEMA

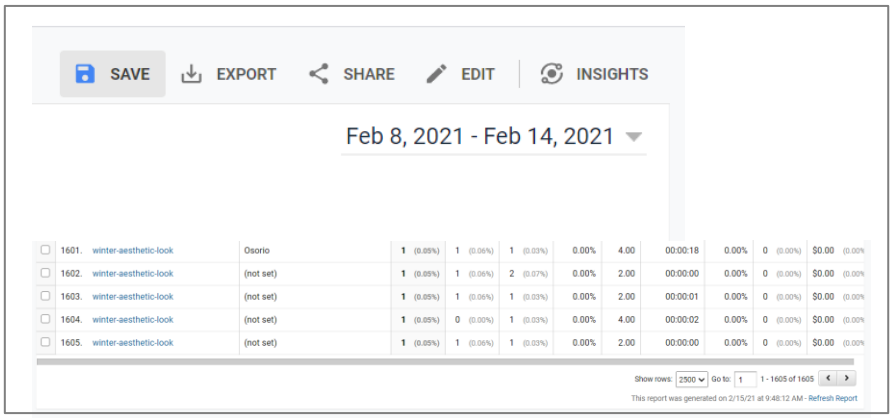
5.1 Data Source

https://analytics.google.com/analytics/web - For cutedressup.com
The data is filtered from google analytics by date and exported to Excel.
Downloaded the data for the month of January and February 2021 for data on location of gamers(city_campaign.csv).
Age Data filtered for the last six months.(Age_campaign.csv)

Campaign(String)	The Game Theme accessed by the player
City(String)	The city from which the player access the site
Users (String)	Existing or returning players
Pages_Session(Double)	New players
Bounce rate(Double)	Rate at which players leave the site without clicking anything on the gaming site
Avg_Session_Duration(Double)	Duration for which the players use the site

Campaign(String)	The Game theme accessed by the player
Age(String)	The city from which the player access the site
Users (String)	Existing or returning players
Pages_Session (Double)	No. of pages visited in a session
Bounce rate(Double)	Rate at which players leave the site without clicking anything on the gaming site
Avg_Session_Duration(Double)	Duration for which the players use the site

5.2 Data Extraction



	Age	Acquisition			Behavior			Conversions		
		Users	New Users	Sessions	Bounce Rate	Pages / Session	Avg. Session Duration	Goal Conversion Rate	Goal Completions	Goal Value
		1,279 % of Total: 39.63% (3,227)	1,145 % of Total: 40.82% (2,805)	1,718 % of Total: 36.76% (4,674)	0.23% Avg for View: 0.62% (-62.47%)	5.81 Avg for View: 6.09 (-4.50%)	00:04:31 Avg for View: 00:04:57 (-8.85%)	0.00% Avg for View: 0.00% (0.00%)	0 % of Total: 0.00% (0)	$0.00 % of Total: 0.00% ($0.00)
☑	1. 18-24	403 (31.51%)	366 (31.97%)	545 (31.72%)	0.00%	5.90	00:04:18	0.00%	0 (0.00%)	$0.00 (0.00%)
☑	2. 25-34	279 (21.81%)	245 (21.40%)	362 (21.07%)	0.00%	5.85	00:04:26	0.00%	0 (0.00%)	$0.00 (0.00%)

5.3 Data Cleaning

The Column names were changed to remove Spaces and Special Characters (“.”, “ “). The values of the BounceRate Column is cleaned by removing the % sign

```
from pyspark.sql import functions as fn
df1 = gamesDF.withColumnRenamed('Pages / Session', 'Pages_Session')
df2 = df1.withColumnRenamed('Bounce Rate', 'BounceRate')
df3 =  df2.withColumn('BounceRate', fn.regexp_replace('BounceRate', '%', ''))
clean_df = df3.toDF(*(c.replace('.', '_') for c in df3.columns))
final_df = clean_df.withColumnRenamed("Avg_ Session Duration", "AvgSessDuration")
```

The String features were converted to numeric values since the machine learning algorithms are based on algebra and need numeric values

The no. of users was stored as string. It was transformed to type Double
gamesDF = interDF.withColumn("Users", interDF.Users.cast("double"))

BounceRate which was a String is converted to Double
df = final_df.withColumn("BounceRate", final_df.BounceRate.cast("double"))

Categorical Values like City and Gametheme are also converted to string using StringIndexers which help to represent the categories in numbers

6. CASE STUDY 1 – PREDICTING BOUNCE RATE

The set of data capturing the age of players and the gametheme they are playing is first visualized and analyzed to arrive at a model to predict Bounce rate for a particular age group.

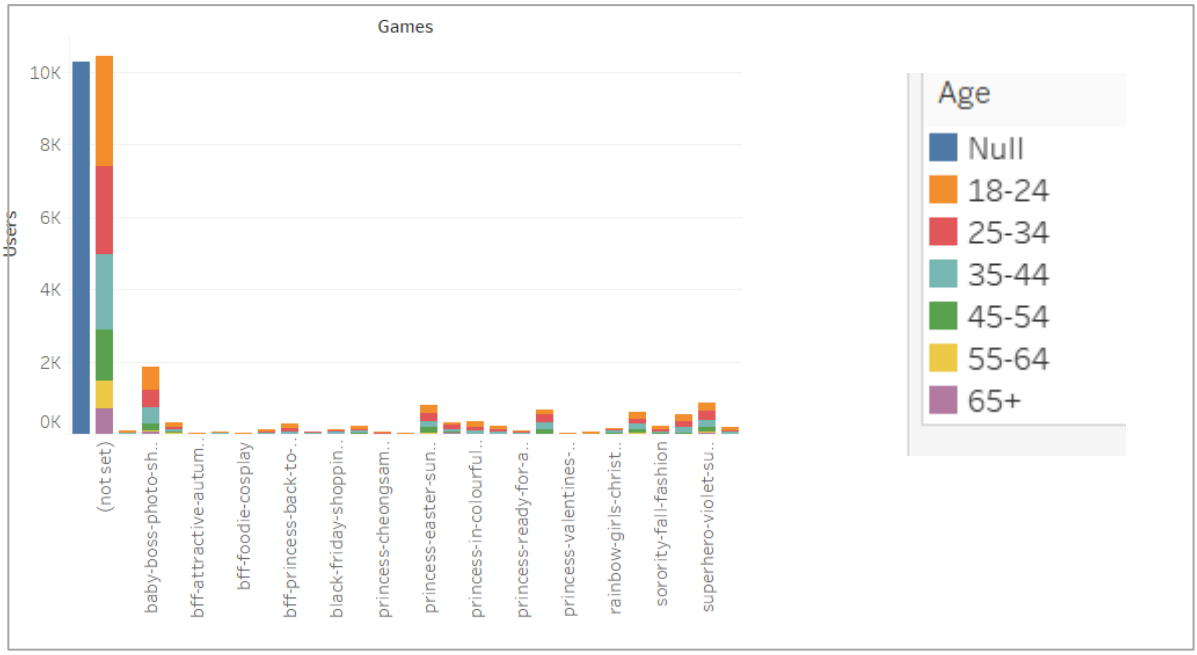
Bounce rate is the rate at which the player leaves the gaming site without even clicking at it. It has to be preferably zero or as lower as possible

6.1 Choosing the Algorithm

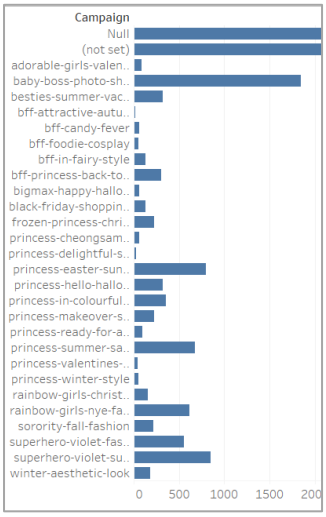
The Campaign or GameTheme and the age group is taken as input variable (set of independent variable) and the bounce rate is predicted as output (dependent target variable). Since a numeric continuous variable is being predicted based on two independent variables , the multivariate regression is chosen.The data available / collected is relatively smaller. So

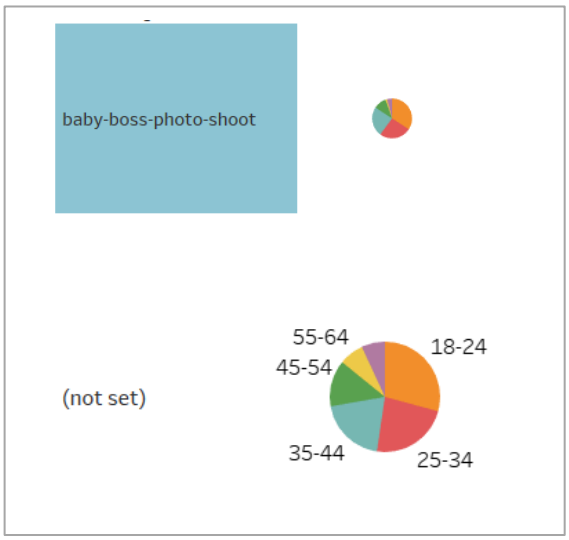
an algorithm of high bias/low variance like Multi-variate Linear regression is a good choice. A much wider data is needed to have a still accurate model.

6.2 Data Visualization - age-campaign.csv



The bar chart shows the distribution of data for the different games played by users of different age groups



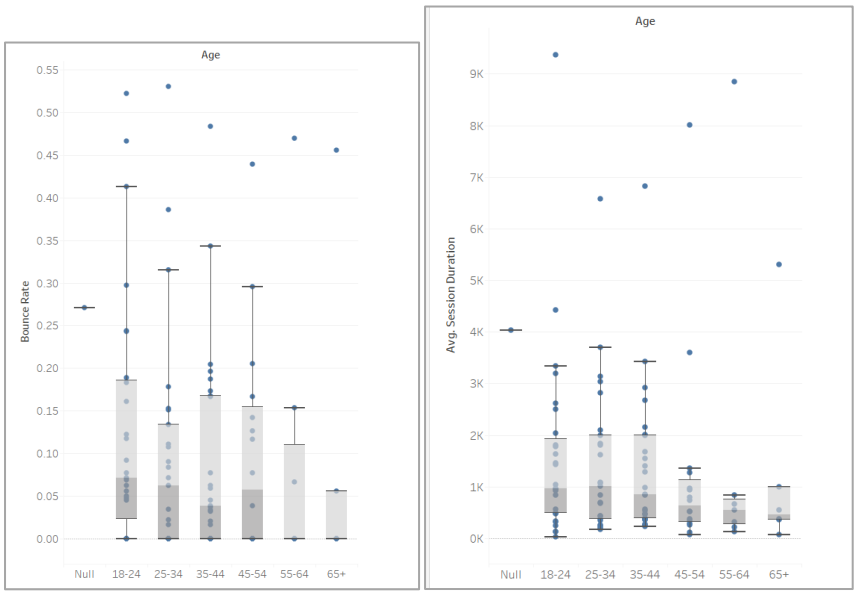


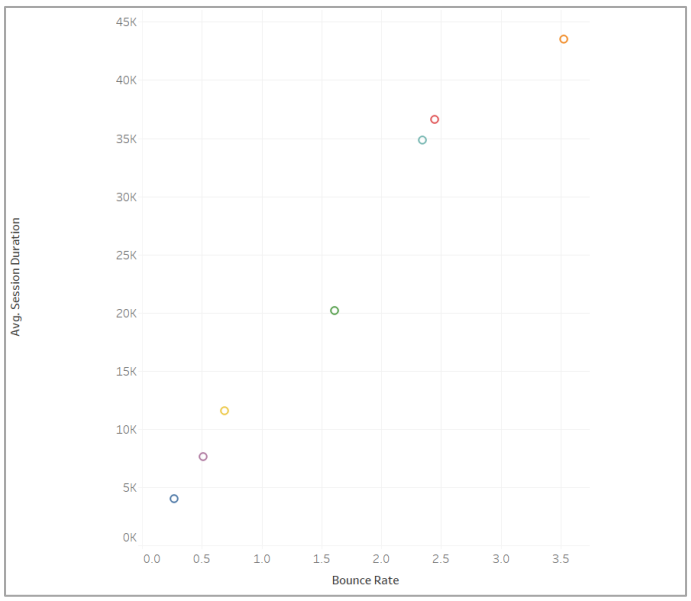
There are about 25 categories of games and 5 age groups across which the data is distributed. some values are null and some are "(not set)".

```
In [25]: import pyspark.sql.functions as fn

df.agg( * [ (1 - (fn.count(c) / fn.count('*')) ).alias(c + '_missing') for c in df.columns] ) \
    .show(vertical=True)

-RECORD 0-------------------------------------
 Age_missing             | 0.025974025974025983
 Campaign_missing        | 0.025974025974025983
 Users_missing           | 0.0
 Pages_Session_missing   | 0.0
 AvgSessDuration_missing | 0.0
 BounceRate_missing      | 0.0
```

There is only about 2% missing values which is negligible. But it can be filled with based on the trend seen for the most played game. From the data visualization above, the most played game is identified as “baby-boss-photo-shoot” and the frequent playing group is “18-24”. So the null values were modified to be filled with “baby-boss-photo-shoot” for Campaign and “18-24” for Age-group.

The box plot is drawn to see any outlier values for BounceRate and the Average Session time. The outliers are dropped





The scatter plot shows the possible linear fitting relation between Session duration and bounce rate and age-group

6.3 Data Exploration: age-campaign.csv

The data is collected over a period of six months for the gamethemes released. The following are the features taken in from the data: Age, Campaign, Users, Session, Pages/Session, Avg Session Duration

summary	Users	BounceRate	Pages_Session	AvgSessDuration
count	539	539	539	539
mean	48.32282003710575	1.9077736549165116	5.832708719851575	250.46703153988898
stddev	121.37747068158338	3.376550264463662	1.7838340799142733	173.6730686418285
min	10.0	0.0	2.54	12.0
max	1149.0	37.93	19.17	1393.13

Almost well distributed data. The coefficient of variation (the ratio of mean to standard deviation) is very high (close or greater than 1), suggesting a wide spread of observations for AvgSessDuration. BounceRate does not seem to have a wider spread or distribution

	Users	BounceRate	Pages_Session	AvgSessDuration
Users	1	-0.0226123	0.123021	0.156262
BounceRate		1	-0.0885242	-0.0738906
Pages_Session			1	0.663893
AvgSessDuration				1

There is not much of collinearity between the features except for Pages_Session and AvgSessDuration

skewness(BounceRate)	skewness(Users)	skewness(AvgSessDuration)
4.401636589568072	2.3479980154831823	1.7739497798833286

Positively skewed. The data can be scaled and used to mitigate the effect of positive skewness

6.4 Model Insight and Evaluation

When the model was done to predict BounceRate, a negative r2 was got on model evaluation. Possible reason could be a small data set and non-availability of wide distribution of data on BounceRate. Further more data can be collected to provide a better spread of data for training. With a refined model for BounceRate Prediction, the company can predict the bounce rate for a given game and age group to either retain the users with new features/themes or increase revenue with in game purchases

Though the data set is small, the AvgSessDuration is well distributed from the statistics.
Hence , the model was done again with AvgSessDuration as the predicted target value. The Company can predict the Average Session Duration for a given game and Age group.
If the Session Duration is high or considerable, the company can try to add in game purchases to increase revenue. If the SessionDuration low, the company can introduce other game themes to suit the players or refine the advertisements.

For Bounce Rate as Predicted Value:

```
In [140]: evaluator = ev.RegressionEvaluator(predictionCol="prediction", labelCol="BounceRate")

          print(evaluator.evaluate(predictions, {evaluator.metricName: 'rmse'}))
          print(evaluator.evaluate(predictions, {evaluator.metricName: 'r2'}))

          3.1540716435087535
          -0.1287305533821872

In [141]: schema = ["Campaign", "Age"]
          test_record = [ ( "adorable-girls-valentino-fashion", "45-54" ) ]
          df = spark.createDataFrame(test_record, schema=schema)
          model.transform(df).first().prediction

Out[141]: 2.060612142182409
```

The negative value of r2 seen with BounceRate Prediction is due to the small size of the data set.

For AvgSessDuration as Predicted Value: --In the same file Predict_bouncerate.ipynb , edit to change the labelCol as AvgSessDuration and retrain the model. Acceptable values of RMSE and R2 are seen.

```
In [130]: evaluator = ev.RegressionEvaluator(predictionCol="prediction", labelCol="AvgSessDuration")

          print(evaluator.evaluate(predictions, {evaluator.metricName: 'rmse'}))
          print(evaluator.evaluate(predictions, {evaluator.metricName: 'r2'}))

          140.0791273112905
          0.013126303474496837

In [134]: schema = ["Campaign", "Age"]
          test_record = [ ( "adorable-girls-valentino-fashion", "45-54" ) ]
          df = spark.createDataFrame(test_record, schema=schema)
          model.transform(df).first().prediction

Out[134]: 248.98193272334112
```

6.5 Value Proposal and Model Deployment

The Company can invest to export the data from Google Analytics to a database and store it for further analysis. Being able to predict the Bounce rate/Avg Sess Duration , will help the company to Identify which age-group is engaged with the game and hence deploy appropriate ads and features to attract them and retain them..

With a wider data, the model can be deployed

1. Save the model in a directory so that it can be launched. Design a front end to feed input from the developer of game/company to the model. The model estimates the Bounce Rate or other measurable parameter to gauge the gamer interest
2. Keep updating the back end database with data from Google Analytics and relearn model to track the gamer choice data.
   By predicting the bounce rate for a particular game theme, before the game theme can be developed and released, the company can invest wisely in themes of interest to the gamers.

7. CASE 2 – CLUSTERING OF GAMERS

City-Campaign.csv – the data set contains features “City”, Campaign”,Users, Sessions, Bounce rate, Pages/Session, Avg. Sess Duration. The aim is to cluster the gamers by their location and measurable metrics like BounceRate, sess Duration, no.of users

7.1 Choosing the Algorithm

K means algorithm is employed to do the clustering. It is one of the famous algorithms for unsupervised learning There is no response variable. Aim is to find the intrinsic pattern and hidden structures in the data (Similar observations from different groups). Once these groups are identified, the company can invest in in game purchases and Ads to suit the gamers. They can also introduce new games based on the segment interest

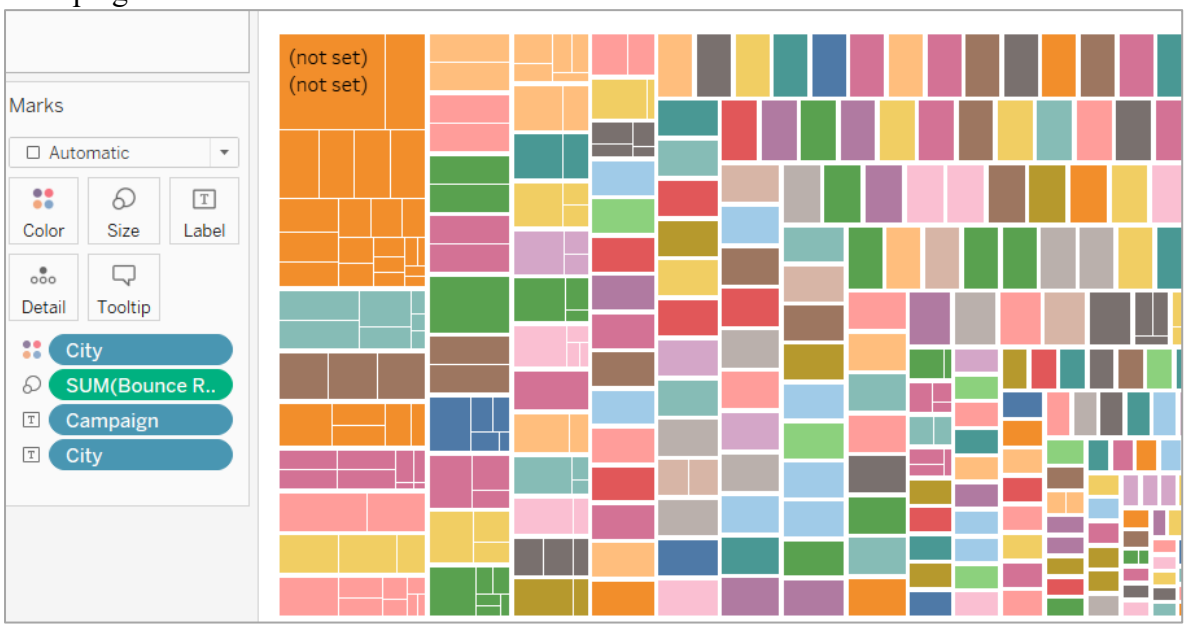
Input: After data analysis only the following variables were decided to be included in the feature set: "campaign-num","city-num","BounceRate","AvgSessDuration" as too many features may not help in good clustering

The no. of clusters k is chosen to be 5.
Other values of k yielded less accuracy.

Output: the entire dataset is segmented to 5 clusters.

7.2 Data Visualization:City-Campaign.csv

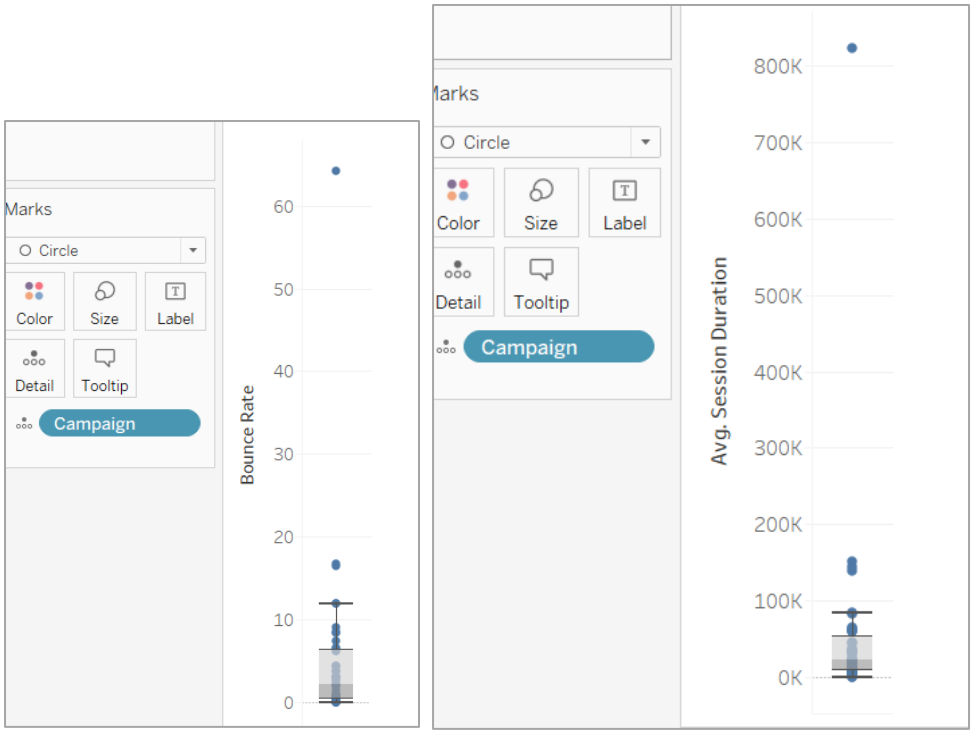
A heat map shows the visual summary or distribution among the categories 'City' and 'Campaign'



Few rows of data with Campaign and City as "(not set)" are present.

Outliers or noisy data is present for both Bounce Rate and Avg Session Duration which can be dropped to improve the clusters

The rows of Campaign data with "(not set)" is first replaced with the most played game "baby-boss-photo-shoot" .




```
quantiles = DF3.approxQuantile("BounceRate", [0.25, 0.75], 0.05)
IQR = quantiles[1] - quantiles[0]
bounds = [quantiles[0] - 1.5 * IQR, quantiles[1] + 1.5 * IQR]

DF4 = DF3.where( (DF3.AvgSessDuration > bounds[0]) | (DF3.AvgSessDuration < bounds[1]))
DF4.count()
```

```
7733
```

7.3 Data Exploration

The correlation between numeric values is studied to remove highly correlated values from the model.

	Users	BounceRate	Pages_Session	Sessions	AvgSessDuration
Users	1	0.0794723	-0.0513615	0.957528	-0.0812279
BounceRate		1	-0.0456758	0.0781049	-0.047416
Pages_Session			1	-0.0530275	0.583446
Sessions				1	-0.0862141
AvgSessDuration					1

- No. of Users and Sessions is highly correlated. So will drop sessions from the set Highly correlated features like Sessions and Users when removed provided better

accuracy. Correlated values were not useful for Machine Learning Segmentation. The accuracy drops when using correlated variables

The Distribution of data is studied through the mean and std. deviation values. The skewness of the data is studied. To minimise the effect of distribution and skewness of data, normalization of data is done

The outliers for the column "AvgSessDuration" were removed and improvement of Accuracy is noted

```
quantiles = DF2.approxQuantile("AvgSessDuration", [0.25, 0.75], 0.05)
IQR = quantiles[1] - quantiles[0]
bounds = [quantiles[0] - 1.5 * IQR, quantiles[1] + 1.5 * IQR]

DF3 = DF2.where( (DF2.AvgSessDuration < bounds[0]) | (DF2.AvgSessDuration > bounds[1]) ).drop()
```

The missing data in the data set is replaced with the most played game "baby-boss-photo-shoot"

7.4 Model Evaluation and Insight

Visualize the clusters in a graph or table to identify the features connecting them.
A domain expert can add more value here.

```
In [32]: import pyspark.ml.evaluation as ev

         predictions = model.transform(DF4)
         evaluator = ev.ClusteringEvaluator()
         print(evaluator.evaluate(predictions))

         0.6071640728197522

In [393]: km = model.stages[4]

          summary = km.summary
          print("Number of clusters: ", summary.k)
          print("Size of (number of data points in) each cluster: ", summary.clusterSizes)
          print("The cluster centers are: ", km.clusterCenters())

          Number of clusters:  5
          Size of (number of data points in) each cluster:  [487, 4194, 1415, 121, 1516]
          The cluster centers are:  [array([0.66201807, 0.84843219, 0.00433509, 3.54345163]), array([0.32245537, 0.41413074,
          0.0473323 , 0.33184432]), array([4.00450046e-01, 2.58636362e+00, 1.75944274e-03, 3.02789681e-01]), array([0.670906
          84, 0.4164316 , 7.41594766, 0.23115123]), array([2.40281348, 0.51901095, 0.0179725 , 0.34700184])]
```

The gaming company can focus on the groups formed as clusters and implement in game purchases and other features specific to these clusters. The clusters are meaningful only when the points within the cluster are close together and far from other clusters. This can be measured by the Silhouette score. Closer the score is to 1 better the accuracy. The model attained a pretty good score of 0.607

7.5 Value Proposal and Model Deployment

As new gamers join and some leave, the clustering has to be performed on a regular basis. So this clustering model can be deployed into a database on a big data cluster. Put the python script in a SQL stored procedure. The model then executes in the database and can easily be trained/updated against data stored in Database. The Company can further invest to capture user clicks and identify the user behavior online and adjust gaming features to retain gamers. The domain expert will be able to identify the cluster interests.