

UNIVERSITI TEKNOLOGI MARA

**VECTOR AUTOREGRESSION TO
FORECAST CHURN BASED ON
STEAM REVIEW**

MUHAMAD FAIDI AKIF BIN MD ALI

**BACHELOR OF INFORMATION SYSTEMS
(HONS.)
INTELLIGENT SYSTEMS ENGINEERING**

FEBRUARY 2023

Universiti Teknologi MARA

**Vector Autoregression To Forecast
Churn Based On Steam Review**

Muhamad Faidi Akif Bin Md Ali

**Thesis submitted in fulfilment of the requirements
for Bachelor of information Systems (Hons.)
Intelligent Systems Engineering
College of Computing, Informatics and Media**

February 2023

SUPERVISOR APPROVAL

VECTOR AUTOREGRESSION TO FORECAST CHURN BASED ON STEAM REVIEW

By

MUHAMAD FAIDI AKIF BIN MD ALI
2021196337

This thesis was prepared under the supervision of the project supervisor, Assoc. Prof. Dr. Shuzlina Binti Abdul Rahman. It was submitted to the College of Computing, Informatics and Media, and was accepted in partial fulfilment of the requirements for the degree of Bachelor of Information Systems (Hons.) Intelligent Systems Engineering.

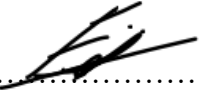
Approved by

.....
Assoc. Prof. Dr. Shuzlina Binti Abdul Rahman
Project Supervisor

FEBRUARY 3, 2023

STUDENT DECLARATION

I certify that this thesis and the project to which it refers is the product of my own work and that any idea or quotation from the work of other people, published or otherwise are fully acknowledged in accordance with the standard referring practices of the discipline.



.....
MUHAMAD FAIDI AKIF BIN MD ALI
2021196337

FEBRUARY 3, 2023

ACKNOWLEDGEMENT

All praise belongs to Allah, thanks to His unending blessings, His enlightenment, and His knowledge, I was able to complete my research.

First and foremost, I am extremely grateful and would like to express my deepest gratitude to my supervisor, Assoc. Prof. Dr. Shuzlina Binti Abdul Rahman for her guidance, patience, concerns, and wisdom from the beginning of this research to the end of its completion.

My deepest thanks towards my final year project lecturer, Dr Azliza Binti Mohd Ali for her guidance and feedback towards the project and the knowledge thought and shared. Moreover, thanks to my examiner Ts. Dr. Rashidah Mokhtar, it is my utmost appreciation when she carries out her duty as my examiner.

I would also like to thank my parents for supporting me and for their prayers throughout the progress of this research that have made my path clearer and easier.

My final thanks goes to my classmates and peers who were under the supervision of Assoc. Prof. Dr. Shuzlina Binti Abdul Rahman for providing moral support, sharing ideas and assisting each other, and sharing knowledge with one another

ABSTRACT

The importance in forecasting when a customer will stop from using a businesses' products or services is called churn forecasting. Business' do this because they can take action before losing their customers, similarly to video game developer and publisher who faces the same problem especially on Steam, the biggest video game digital distributor is bound to be extremely competitive. Hence, rectifying the need to see their current and future position in the market to avoid revenue from decreasing due to decline in number of customers. Sentiment analysis for video game is the process to determine if a review is positive or negative based on what is written by the reviewer that could help developers and publisher to improve their game based on the review's context. It is proven that sentiment is a valuable additional input to forecast churn. For example, increasing pattern in negative sentiment would indicate customers are not satisfied with the game, causing them to stop playing and vice versa for positive sentiment. Many studies have been conducted regarding churn forecasting using sentiment, however there is a lack in studies that utilizes Steam reviews, and with steam large-scale computer game there is an opportunity to analyse sentiment in a large scale that other study lacks. Hence, a churn forecasting model is designed and developed based on additional input from sentiment analysis model. The machine learning classifier called Support Vector Machine is used, whereas Vector Autoregression (VAR) is used for the churn forecasting model.

TABLE OF CONTENTS

CONTENT	PAGE
SUPERVISOR APPROVAL	ii
STUDENT DECLARATION	iii
ACKNOWLEDGEMENT	iv
ABSTRACT	v
TABLE OF CONTENTS	vi
LIST OF FIGURES	ix
LIST OF TABLES	xii
LIST OF ABBREVIATIONS	xiii
 CHAPTER ONE: INTRODUCTION	
1.1 Research Background	1
1.2 Problem Statement	3
1.3 Research Questions	4
1.4 Research Objectives	4
1.5 Research Scope	5
1.6 Research Significance	5
1.7 Summary on Introduction	6
 CHAPTER TWO: LITERATURE REVIEW	
2.1 Introduction	7
2.2 Steam Review	7
2.3 Machine Learning Classifiers	10
2.4 Levels of Sentiment Analysis	11
2.4.1 Document Level	11
2.4.2 Sentence Level	12
2.4.3 Entity and Aspect Level	12
2.5 Framework of SA Modelling	12
2.6 Discussion on Previous SA	13
2.6.1 Preprocessing Textual Data	13

2.6.2	Discussion on Related Work	16
2.6.3	Summary of Discussion in Sentiment Analysis	18
2.7	Framework of Time Series Forecasting Modelling	19
2.7.2	Determining Stationary	20
2.7.3	Granger Causality Analysis	20
2.7.4	Vector Autoregression	21
2.7.5	Forecasting Model	21
2.8	Discussion on Related Work for Prediction	21
2.9	Discussion on Related Work for Forecasting	22
2.10	Summary of Related Work for Churn Prediction and Forecasting	23
2.11	Churn Prediction Based on Sentiment Analysis	24
2.12	Summary on Literature Review	25

CHAPTER THREE: RESEARCH METHODOLOGY

3.1	Research Framework	27
3.2	Preliminary Study	29
3.3	Knowledge Acquisition	30
3.4	Data Collection	31
3.5	Data Preprocessing	33
3.6	Model Design	35
3.6.1	Sentiment Analysis Model	37
3.6.2	Churn Forecasting Model	41
3.7	Model development	43
3.8	Testing and Evaluation	60
3.9	Documentation	63
3.10	Summary on Research Methodology	63

CHAPTER FOUR: RESULTS AND FINDINGS

4.1	Dataset Review & Preprocessing Result	64
4.2	Sentiment Analysis Modelling Results	68
4.2.1	Performance Based on Different Kernels	68
4.2.2	Performance Based on Different Kernels with SMOTE	69
4.2.3	Performance Based on Tuned TF-IDF	71

4.2.4	Performance Based on Tuned TF-IDF + Tuned SVM	72
4.2.5	Using the Developed SA Model	73
4.3	Churn Forecast Modelling Results	74
4.3.1	Timeseries Consolidation	74
4.3.2	Stationary	77
4.3.3	GC results	79
4.3.4	AIC results	79
4.4	Performance on Proposed Churn Forecasting Model	80
4.5	Evaluating Significance on Proposed CF Model	84
4.5.1	BloonsTD6	85
4.5.2	Oxygen Not Included	87
4.5.3	Sid Meier's Civilization VI	89
4.5.4	Stellaris	91
4.6	Summary of Results and Findings	92
 CHAPTER FIVE: CONCLUSION		
5.1	Purpose of Project	94
5.1.1	Objective 1	94
5.1.2	Objective 2	95
5.2	Strengths	96
5.3	Limitations	96
5.4	Future Work and Recommendations	97
5.5	Summary on Conclusion	98
 REFERENCES		 99

LIST OF FIGURES

FIGURE	PAGE
1.1 Steam storefront	1
1.2 Steam game release by year	2
2.1 Mind map of literature review	7
2.2 Professional looking review	9
2.3 Short but informative review	9
2.4 Jokey review	9
2.5 General flow of SA modelling	12
2.6 Checkbox style review	15
2.7 General flow of time series forecasting modelling	19
2.8 Graph for trend and seasonality	20
3.1 Sample for lifetime number of players on SteamDB	31
3.2 Sample number of daily players since 2011	31
3.3 An example of Steam review and their API	32
3.4 Sample of scrapped raw review through Steam API	32
3.5 Sample of semi-preprocessed review saved in Excel	33
3.6 Model architecture	35
3.8 SVM kernel comparison	38
3.9 SVM kernel comparison with SMOTE	38
3.10 TF-IDF hyperparameter tuning using grid search	39
3.11 SVM hyperparameter tuning with hyperopt	40
3.12 Churn forecasting model	41
3.13 Sample data of consolidated dataset	42
3.14 Pseudo-code for scrapping algorithm	45
3.15 Pseudo-code for preprocessing algorithm	47
3.16 Pseudo-code for selecting reviews algorithm	48
3.17 Pseudo-code for merging reviews algorithm	49
3.18 Pseudo-code for reading file to get input data and label	50
3.19 Pseudo-code for train and test split algorithm	50
3.20 Pseudo-code for vectorizing transformed features	51

3.21	Pseudo-code for fitting into classifier	51
3.22	Pseudo-code for displaying result	52
3.23	Pseudo-code for modeling with default parameters	52
3.24	Pseudo-code for train test split and apply SMOTE	53
3.25	Pseudo-code for modelling with smote algorithm	53
3.26	Pseudo-code for creating pipeline for tf-idf hyperparameter tuning	54
3.27	Pseudo-code for grid search algorithm	54
3.28	Pseudo-code of function with tuned tf-idf with SMOTE	55
3.29	Pseudo-code for modeling with tuned tf-idf with SMOTE	55
3.30	Pseudo-code for objective function	56
3.31	Pseudo-code for setting hyperopt search space	56
3.32	Pseudo-code for starting hyperparameter tuning for classifier	56
3.33	Pseudo-code for modeling with tuned tf-idf and classifier with smote	57
3.34	Pseudo-code for adf test algorithm	57
3.35	Pseudo-code for kpss test algorithm	58
3.36	Pseudo-code for developing cf algorithm	58
3.37	Pseudo-code for granger causality algorithm	58
3.38	Pseudo-code for plotting test value vs forecasted value	59
3.39	Pseudo-code for evaluating with MAPE	59
4.1	Sample of reivew before preprocessing	64
4.2	Sample of review after preprocessing	65
4.3	Number of sentiment for each game before and after preprocessing	66
4.4	Sample for daily number of player	66
4.5	Timeseries of daily number of player for BloonsTD	67
4.6	Timeseries of daily number of player for ONI	67
4.7	Timeseries of daily number of player for SMC	67
4.8	Timeseries of daily number of player for Stellaris	67
4.9	Comparison between SVM kernels	68
4.10	Comparison between linear and linear with SMOTE	69
4.11	Comparison between RBF and RBF with SMOTE	69
4.12	Comparison between polynomial and polynomial with SMOTE	70
4.13	Comparison between SVM kernels with SMOTE	70
4.14	Sample predicted dataset	74

4.15	Calculating sent score and consolidating	74
4.16	Timeseries for consolidated BloonsTD	75
4.17	Timeseries for consolidated ONI	75
4.18	Timeseries for consolidated SMC	76
4.19	Timeseries for consolidated Stellaris	76
4.20	Differenced timeseries for BloonsTD	78
4.21	Differenced timeseries for ONI	78
4.22	Differenced timeseries for SMC	78
4.23	Actual vs forecasted result for BloonsTD	81
4.24	Undifferenced actual vs forecasted result for BloonsTD	81
4.25	Actual vs forecasted result for ONI	81
4.26	Undifferenced actual vs forecasted result for ONI	81
4.27	Actual vs forecasted result for SMC	82
4.28	Undifferenced actual vs forecasted result for SMC	82
4.29	Actual vs forecasted result for Stellaris	82
4.30	Timeseries comparison for BloonsTD	85
4.31	Model comparison of forecasted value for BloonsTD	86
4.32	Timeseries comparison for ONI	87
4.33	Model comparison of forecasted value for ONI	88
4.34	Timeseries comparison for SMC	89
4.35	Model comparison of forecasted value for SMC	90
4.36	Timeseries comparison for Stellaris	91
4.37	Model comparison of forecasted value for Stellaris	92

LIST OF TABLES

TABLE	PAGE
2.1 Summary of Previous Work on Sentiment Analysis	16
3.1 Research Framework	27
3.2 Attributes Selected based on the Scrapped Data	33
3.3 TF-IDF Hyperparameter Values	54
3.4 Defined Hyperopt Requirments	55
3.5 Search Space According to Kernel for Hyperopt	56
3.6 Summary of software used	60
3.7 Sample Confusion Matrix for Binary Label	61
4.1 Amount ff Review and Sentiment for Each Game Scraped	64
4.2 Amount of review for each game before and after preprocesing	65
4.3 Amount Of Review And Sentiment For Each Game After Preprocessing	66
4.4 SVM kernel performance result	68
4.5 SVM kernel with SMOTE performance result	71
4.6 Grid Search Result for TF-IDF Parameter	71
4.7 Default TF-IDF vs Tuned TF-IDF	72
4.8 Default TF-IDF vs Tuned TF-IDF	72
4.9 Amount of Sentiment for Preprocessed and Predicted Review	73
4.10 Legend explanation on timeseries	76
4.11 ADF test & KPSS pvalue	77
4.12 ADF test & KPSS after first differencing	77
4.13 Granger causality output for each game	79
4.14 Lag AIC for Each Game	79
4.15 MAPE output for each game	83
4.16 Number of Sentiment Used for Model Comparison	84
4.17 Model Comparison Metrics for BloonsTD	86
4.18 Model Comparison Metrics for ONI	88
4.19 Model Comparison Metrics for SMC	90
4.20 Model Comparison Metrics for Stellaris	92

LIST OF ABBREVIATIONS

SVC	Support Vector Classifier
SVM	Support Vector Machine
NB	Naïve Bayes
XGB	Extreme Gradient Boosting
MNB	Multinomial Naïve Bayes
LR	Linear Regression
MLP	Multi-layer Perceptron Classifier
DT	Decision Tree
RF	Random Forest
BoW	Bag of Word
TD-IDF	Term Frequency – Inverse Document Frequency
VAR	Vector Autoregression
SMOTE	Synthetic Minority Over-sampling Technique
MAPE	Mean Absolute Percentage Error
SA	Sentiment Analysis
CF	Churn Forecast
CV	Cross Validation
GC	Granger Causality
CSV	Comma-separated Values
SW	Sliding Window
ADF	Augmented Dickey-Fuller test
KPSS	Kwiatkowski-Phillips-Schmidt-Shin
AIC	Akaike Information Criterion
ONI	Oxygen Not Included
SMC	Sid Meier's Civilization VI
ts	timeseries
UGC	User generated content
API	Application Programming Interface

CHAPTER ONE

INTRODUCTION

This chapter provides the background and rationale for the study. It also gives details of the significance of churn prediction in business especially in the domain of video game, the issues and problems that led to this study.

1.1 Research Background

The video gaming industry have come a long way, from the first ever video game on a system called Brown Box that is only accessible to a person, to an online storefront for people to buy and update games whenever they want such as Steam.

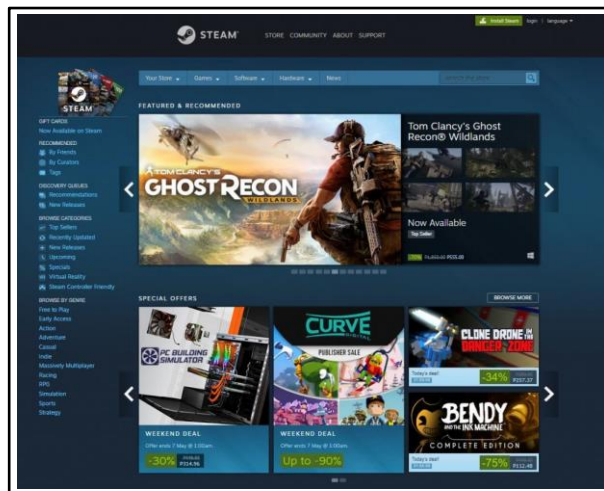


Figure 1.1 Steam storefront

Steam is a video game digital distribution service storefront by a company called Valve. This platform does not just sell games, but allows its user to trade in game items, and socialize with community that consist of diverse personality of people. In addition, Steam is very active to run sales to attract and maintain their users. To top it off, Steam is the leading video game distributor with 120 million active users and a catalogue over 50 thousand games in 2021 (Ammerman, 2021; Zain & Juleza, 2021).

Video game industry have contributed over 36.7 billion USD for desktop games in 2021 according to Wijman (2021). The industry is still growing in terms of number of players and number of games released which makes the industry more competitive than ever. According to SteamDB, a website that is not affiliated with valve that lists and tracks information regarding applications and packages available on Steam, the highest number of games released on Steam was in 2021 with a total of 11739 games.

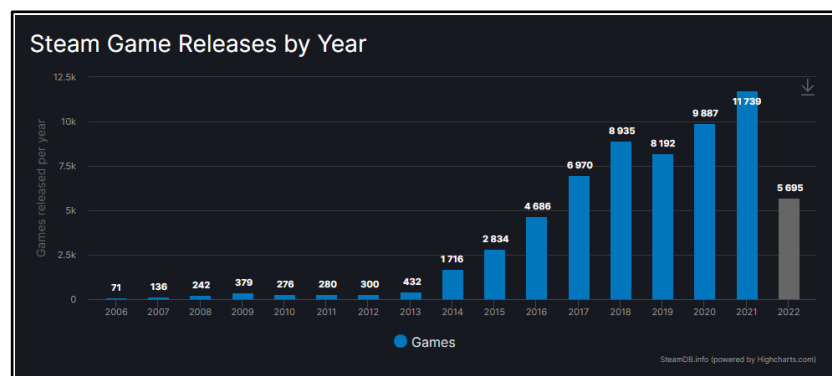


Figure 1.2 Steam game release by year
(Source: SteamDB)

Getting attention depends on the marketing and how well the company can advertise themselves. However, one of the most challenging things they face, like any industries, are customer churn. Customer churn is the percentage of customers that stopped using a company's product or service during a certain time frame.

Therefore, customer churn in video game is the action of customer to stop playing video game. It is more important to maintain current customers than gaining new one, and customer retention is the key to success for video game to become successful not just at the release of the game, but in the long term as well. Churn in video game is caused by its design such as graphics, audio, mechanic of the game that determines how playable it is, and how the game deals social interactions such as dealing with cheater.

The flaws in game that is either caused by social interactions or directly from the developer causes frustration towards the player. The frustration in return causes the game to lose its playerbase (the population of people who play their

game as a whole). According to Robinson in his talk “Why Players are Leaving Your Game” in 2013, “*On average, less than 40% of players return to a free-to-play game after just one session*”. Some examples of game downfalls that is caused by their design are “No Man’s Sky” that lacks content loses 78% of its playerbase in a week, “Cyberpunk 2077” that is too problematic to play because of too many bugs (something that causes the game to be unplayable) and loses 95% of its playerbase in two months according to video game statistics website called GitHyp.

Customer churn can be estimated by identifying players' feeling towards the game. The players may express their feeling about the game they play, either something positive or negative about the game by leaving a review. Reviews are useful for the developers because it contains text that are rich in information about the product where they identify the players’ opinion to further improve their game by fixing bugs or adding new contents. As a result, they can retain their customer, essentially people who play their game.

1.2 Problem Statement

Like all company, game company also face churn when publishing their game. Since Steam is an extremely competitive market, predicting churn is a must to see the status of their game in the market.

Studies regarding churn prediction in game have been conducted by Jang, Kim, & Yu (2021) and Kristensen & Burelli (2019). Most churn prediction relating to games relied on player log data such as their playtime per session. Usage of user generated content is not widely used in this area, which is the main concern of this study.

User generated content refers to review of a product. It is not just for a company or developer to read feedbacks on their game, but also for consumer to make decision on playing or buying a game. Therefore, for game developer to know the players’ opinion on their game, they need to perform sentiment analysis on review because of the huge amount of data.

Very few such as Kilimchi, Yoruk, & Akyokus (2020) incorporates sentiment in their prediction model for mobile games. Additionally, not a lot of research are focused on a large-scale computer game for predicting churn.

Moreover, there is a lack in churn prediction based on sentiment analysis for computer games specifically Steam reviews that also focuses on large-scale games. However, Steam has no player log data that is freely available which can be used to identify user behaviour. Therefore, instead of predicting churn of a specific user, this research is focussed on churn of a game that views it as a whole (the entire playerbase), by forecasting what the number of players will be in a time series (a set of data that tracks over time) manner.

Prediction is the estimation of what a label will be based on the subjective considerations such as a person behaviour, whereas forecast is to estimate what will happen based on historical data using statistic. Hence, there is a need to propose the churn forecasting model.

1.3 Research Questions

To solve the problem that have been discussed in problem statement, some research questions have been formed which are the following.

1. How to design and develop a churn forecasting model that utilizes sentiment analysis?
2. Is there a significance in utilizing sentiment analysis model for the churn forecasting model?

1.4 Research Objectives

To answer the research questions above, research objectives are developed.

1. To design and develop a churn forecasting model that utilizes sentiment analysis.
2. To compare the result between churn forecasting model that does not utilize sentiment analysis model and churn forecasting model that utilizes the sentiment analysis model.

1.5 Research Scope

To narrow down, games may yield different types of reviews based on their community and genre. Therefore, online Games as a Service (GaaS) is chosen. GaaS is a type of game that frequently gets updated, therefore the playerbase is significantly active compared to single player games that focuses on story which have less replay value. In addition, only a single type of genre is focused on which is strategy, since different types of genre may use different terminology that may cause problem to a machine learning classifier (Viggiato, Lin, Hindle, & Bezemer, 2021). The selected games were BloonsTD6, Oxygen Not Included (ONI), Stellaris, and Sid Meier's Civilization VI (SMC). These are strategy game that requires careful planning, where it is not affected by other players' action that can cause toxicity. In summary, the games selected have attribute in common which are heavy strategical planning that is focused highly on single player experience.

Steam review consist of multiple languages, only English language reviews were taken into consideration, since this research is not focused on translating other languages. Furthermore, a time frame for forecasting was selected for one year from 1st January 2020 to 31st December 2020.

Finally, the model developed is targeted for game developers since the forecasted churn is more useful to them to plan what is next that they should do for their game before a churn increases.

1.6 Research Significance

1. Developer able to make timely decision

By using the model, they can identify when churn will increase. By doing so, they can analyse what can be improved on their current status of the game that is causing the forecasted churn to increase. Hence, by acting in time they may prevent the churn from increasing further by listening to their playerbase feedback.

2. Incur marketing cost

When a developer fulfils their playerbase requirement, people will enjoy their game. Hence, they get free marketing such as word of mouth from people who genuinely enjoys the game rather than paid review. Hence, incurring marketing cost.

3. Gain customer loyalty

When developers listen to their consumer and produces quality games, people will trust their work. Therefore, people will be prone to buying other things or games that are produced by the same developer because of the consistency in quality and services that developers did to the game by listening to the playerbase.

1.7 Summary on Introduction

This chapter defines what is Steam, sentiment analysis, customer churn, prediction and how they are correlated to each other to address the problem that leads to this study and its significance.

CHAPTER TWO

LITERATURE REVIEW

This chapter provides the processes and results of related works that has been done by previous researchers. This chapter also explores on the nature of Steam and video game churn.

2.1 Introduction

This chapter discusses about what previous studies have been done and is separated into Sentiment Analysis (SA), Prediction, Forecasting and Churn Prediction based on Sentiment Analysis. Figure 2.1 below maps out all the authors that are closely related to this study.

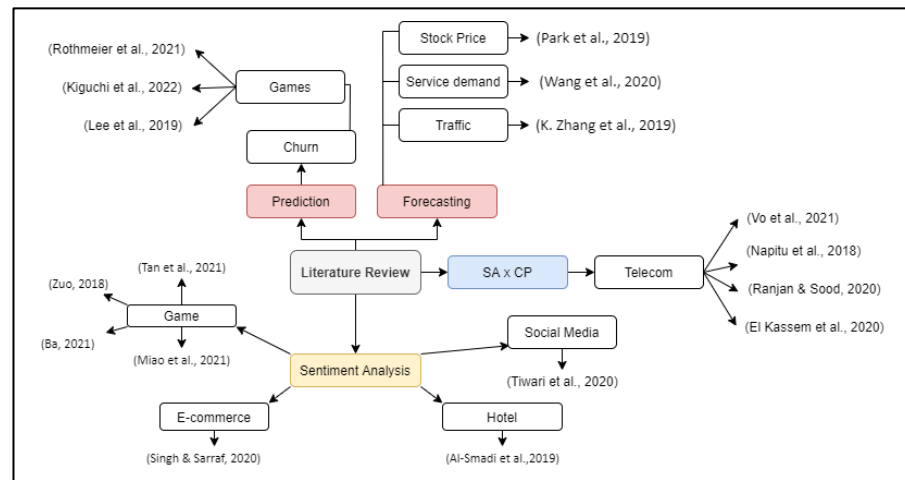


Figure 2.1 Mind map of literature review

2.2 Steam Review

The nature of Steam consists of people who are not professional. They write review for other people to understand easily. Hence, many features regarding game design may not be mentioned. However, their message still conveys meaning that people understand and that they enjoy the game through words that is closely related to the context of the game, in terms of its idea rather than

its design. For example, if a game is about war, then the word “war” will most likely be mentioned many times compared to “strategy”.

The method this research proposed does not just look in terms of the developer point of view where they need to identify aspects of the game that needs improvement, rather people enjoyment of the game, and this enjoyment or other emotions can be conveyed in many ways.

In an internet article by Lane (2018), states that Steam reviews consist of jokey one-liners, incoherent rants, and political rallying cries. The writer conducted an interview with several Steam reviewers. It is found that some reviewers write for Steam friends (so that it appears in their feed) and to force themselves to critically think about the game. This results in a professional looking review that developers really like to read, displayed in Figure 2.1. It is also discovered that other people write reviews in an informative and straightforward manner which can be seen in Figure 2.2. Meanwhile, some people write just to add an overall rating to the review, which is a requirement to give a rating to a game that results in short review that conveys joke, but still describe the game in a way, which is in Figure 2.3. Nonetheless, ratings are still emotion about the game whether they enjoy it or not.

In a research by (Viggiato et al., 2021). discovered some problems when it comes to SA regarding Steam, which are sarcasm, game comparison, negative terminology, and contrast conjunction. Surprisingly, sarcasm only contributes 6% to the problem while contrast conjunction (but, although, though, even though, and even if) has the highest of 30%. However, this root causes were not solved in the research and with time constraints, this problem cannot be solved explicitly in this research as well.

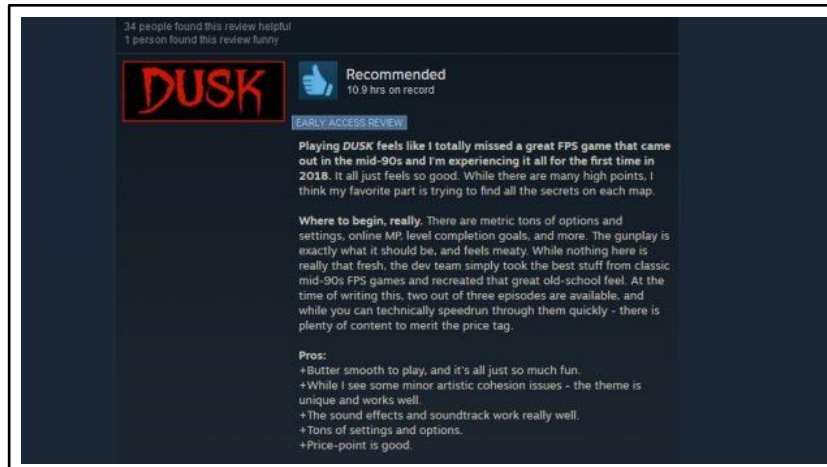


Figure 2.2 Professional looking review
(Source: Lane, 2018)



Figure 2.3 Short but informative review
(Source: Lane, 2018)

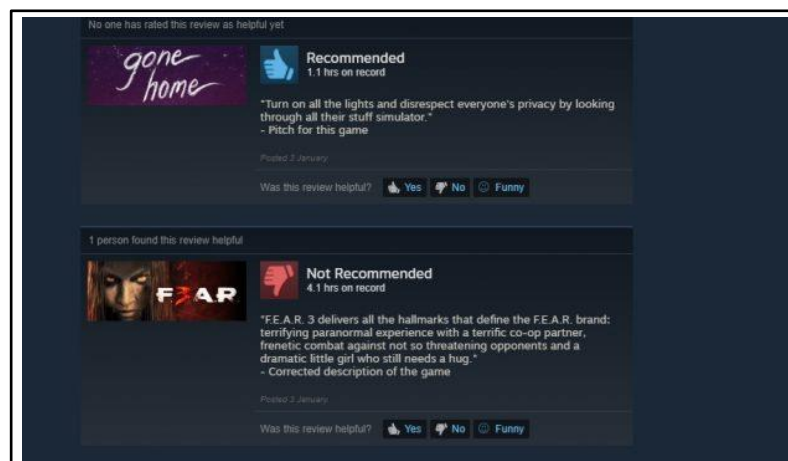


Figure 2.4 Jokey review
(Source: Lane, 2018)

Therefore, regardless of how the review is written, all reviews need to be considered to ensure that the classified sentiment will not cause the forecasting model to have bad results due skewed data that is caused by bad sentiment model during training.

2.3 Machine Learning Classifiers

Machine learning focuses on the use of data and algorithms, which we call them classifiers to simulate how human learn, gradually improving accuracy of learning by using statistical methods. Algorithms are trained to make predictions or classification.

Some classifiers explained here consist of base classifier such as Support Vector Machine and ensemble classifier. Ensemble classifier is the combination of multiple base classifiers to create a more robust classifier and usually have higher accuracy by learning from one another. The machine learning classifiers that are to be discussed in this section and further in the literature reviews are as follows.

a) Support Vector Machine (SVM)

A supervised machine learning algorithm best suited for classification that can solve linear and nonlinear problems. It creates a line in which it will separate data into classes. Support Vector Classifier (SVC) is an instance of SVM.

b) Naïve Bayes (NB)

A probabilistic machine learning algorithm based on the Bayes Theorem. This classification refers to the strong independence of each feature of a model.

c) Extreme Gradient Boosting (XGB)

Boosting refers to adding one classifier at a time to the ensemble and fit to correct the prediction errors made the prior models, it is a type of ensemble learning. Models are fit using any arbitrary differential loss function and gradient descent optimization algorithm.

d) Multinomial Naïve Bayes (MNB)

Is an instance of NB that uses multinomial distributions for each feature.

e) Logistic Regression (LR)

A statistical analysis method to predict a binary outcome such as yes or no based on prior observations of a data set by analysing relationship between one or more independent variables.

f) Multi-Layer Perceptron Classifier (MLP)

A classifier based on Neural Network. It is a feedforward artificial neural network that generates asset of output based on its input and uses backpropagation for training the network.

g) Decision Tree (DT)

Can be used to solve regression and classification problems by learning simple decision tree rules that is inferred from training data.

h) Random Forest (RF)

A model that is made up of multiple decision tree. An extension of an ensemble bagging method.

2.4 Levels of Sentiment Analysis

There are many levels of sentiment analysis such as document, sentence, and feature or aspect level which will be explained in here.

2.4.1 Document Level

Document level classifies an entire opinion document into a single sentiment polarity, therefore if there are other words that indicate another opinion it will not be counted which wrongly classifies the text. Hence this is not good for classifying sentiment polarity for Steam review since people can express their opinion on multiple things in a single review or piece of text.

2.4.2 Sentence Level

Sentence level classify each sentence into their sentiment polarity. Hence if a different opinion is mentioned in a different sentence, it can be detected. Therefore, this option is viable for classification that relates with review, specifically Steam review.

2.4.3 Entity and Aspect Level

Feature level is a finer grained method classify sentiment polarity on each aspect specifically. For example, if the aspects of a product which is laptop, would be screen, audio, material, etc, and any other word that comes up after that will determine if it is good or bad. This sentiment is useful if seller want to identify something specifically about their product.

2.5 Framework of SA Modelling

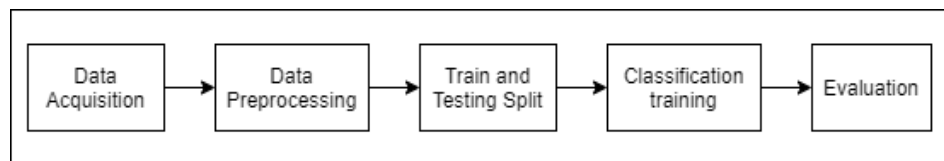


Figure 2.5 General flow of SA modelling

Figure 2.5 above display the general process of SA modelling and is explained on what each process does below.

a) Data Acquisition

Raw textual data are acquired either from reviews or from a dataset. These reviews can be obtained through scraping or using API whereas dataset can be obtained from websites such as GitHub or Kaggle. Data are not just limited to reviews but can also be in other form such as call logs that has been transcribed.

b) Data Preprocessing

After acquiring data, like any other process, requires preprocessing. This step consists of removing stop words, normalizing by changing all to lowercase, part of speech tagging, etc. However, the process depends on the goal of study.

This section is discussed in detailed in Subtopic 2.6.1.

c) Train and Testing Split

The cleaned dataset are split into training and testing where training set are used to train the classifier to produce the correct output, whereas testing set are used to evaluate the performance of the classifier.

d) Classification training

Next, training set are fed to a machine learning algorithm such as NB, SVM, DT, etc to classify the review. Classification also depends on the goal of study, such as classifying emotions such as sadness, joy, anger, etc or just classifying into positive, negative, or neutral.

e) Evaluation

Accuracy of model need to be evaluated and validated. Some of the most common evaluation method is by identifying the model's accuracy, precision and recall.

2.6 Discussion on Previous SA

In this subtopic, preprocessing that has been done and what machine learning classifiers have been done will be discussed.

2.6.1 Preprocessing Textual Data

The preprocessing methods mentioned here will only be the one that is related to preprocessing method on textual data in Chapter 3.

a) Tokenization

Tokenization is done to break up sentence into each specific individual words to bring context (Tan Jie, Sai, Chow, & Tan Chi, 2021).

b) Part of Speech (POS) tagging

Tag each token with their corresponding grammatical categories such as verb, determinant, adjective, etc. (Tan Jie et al., 2021).

c) Lemmatization

In simple words, it is the converting of a word to its base form, words such as plural, past tense, present tense is converted to their root form. (Tan Jie et al., 2021).

d) Removing hyperlinks

As discussed before, Steam is an open-ended which allows anyone to post any type of reviews. Hence there is also freedom to put anything in the reviews. It is common for people to include hyperlinks that lead to a video or tutorial or other non-related content, therefore it is removed since it brings no value to a classifier (Urriza & Clariño, 2021).

e) Remove most non-alphanumeric characters

Contains in reviews for aesthetic purposes, other than common symbols (%,-,*,etc) and punctuation marks, others are removed as they contain no values (Tan Jie et al., 2021; Urriza & Clariño, 2021; Zuo, 2018).

f) Remove emojis

Emojis have heavier impact on sentiment produced (Ayvaz & Shiha, 2017). Given the nature of Steam reviews, emojis are removed to avoid wrong classification. On the other hand, some uses them to create checkbox like reviews. In this case, ticked checkbox are kept since it contains the sentiment of the reviewer.



Figure 2.6 Checkbox style review

g) Remove stop words

Stop words are the most frequent type of words in human language, it is a low-level information that does not hold significant values to classifiers (Tan et al., 2021; Zuo, 2018). In addition, they are removed to reduce computation time.

h) Remove extra whitespace

Extra whitespace, such as accidental whitespace between two words, or spaces between paragraph are only for the human to read information easily. However, for classifiers, this will be noise entity hence it needs to be removed (Tan Jie et al., 2021)

i) Fix contractions

Contractions in natural language processing refers to shortened form of words that usus apostrophe such as “don’t” that stands for “do not” that brings challenges to tokenization and POS tagging.

j) Set to lowercase

Setting the words to lowercase is to normalize all the words so that same words with different capital will not count as a unique word (Zuo, 2018)

2.6.2 Discussion on Related Work

Table 2.1 Summary of Previous Work on Sentiment Analysis

Author	Level	Dataset	Features	Algorithm(s)	Result
Zuo 2018	-	Steam Review	<ul style="list-style-type: none"> • Information gain • TFIDF* • n-gram 	<ul style="list-style-type: none"> • DT* • NB 	<ul style="list-style-type: none"> • DT 75% accuracy • Extra feature increases accuracy
Tan 2021	-	Steam Review	TF-IDF	<ul style="list-style-type: none"> • SVC* • XGB • MNB • LR • MLP 	SVC 91% accuracy
Ba 2021	Aspect	Steam Review	TF-IDF	SVM	-
Ji 2019	-	Steam Review	Word2vec	<ul style="list-style-type: none"> • SVM • NB 	No clear winner
Miao 2021	-	Kaggle	-	<ul style="list-style-type: none"> • SVM • RF* • NB 	RF outperforms all, in terms of preprocessing libraries
Singh 2020	-	e-commerce	BoW	RF	-
Tiwari 2020	Sentence	Twitter	BoW	<ul style="list-style-type: none"> • RF* • DT* • SVM 	<ul style="list-style-type: none"> • DT 99.3% accuracy • RF 99.4% accuracy

Multiple SA research has been conducted regarding Steam review domain. Research by Zuo (2018) uses Steam API to obtain data as it is less prone to error compared to the traditional scrapping, and uses TF-IDF as feature extraction method, as well as Decision Tree and Naïve Bayes as two different classifier. It was found that Decision Tree with TF-IDF have better accuracy compared to Naïve Bayes with TF-IDF with 500 feature and extra feature. However, it was not a good comparison between the two classifiers since they both have different parameter settings, and no best classifier was concluded at the end of the research.

Another research by Tan Jie et al. (2021) compared five algorithms which are SVC, XGB, MNB, LR and MLP and trained them on their default parameter. Unlike previous research by Zuo (2018), this paper also include professional game reviews. Class imbalances are handled by using SMOTE and feature extraction used was TF-IDF. Tan Jie et al. (2021) research focuses on whether

oversampling and hyperparameter tuning can contribute to better classification. Oversampling with SVC and hyperparameter tuning for TF-IDF is found to be the best result out of all classifiers with an accuracy of 91%.

Research by Ji (2019) also obtain Steam review through Steam API and uses NB and SVM using python with a wider scope that focuses on four different games. It was mentioned that word2vec word embedding was used. It was found that both SVM and NB has its pros and cons however they both yield an overall similar accuracy. The main findings of this study was that if most game reviews are not positive, then the algorithm will give less accuracy, this problem was tackled by Tan Jie et al. (2021) by using SMOTE.

Furthermore, research by Miao, Jin, Zhang, Chen & Lai (2021) focuses on NB, SVM and RF. The data obtained for this research was from Kaggle. Instead of just focusing on the text, the research also focusses on other fields such as the “recommended” and “hour played” field. This extra field can be some sort of filter to determine which review will be helpful to the classifier, and if left unchecked, it can introduce noise to the classifier. The purpose of this research leans towards the preprocessing libraries used and how it effects the classifier. RF has the best accuracies with the highest F1 score around 89.5 for all preprocessing libraries used compared to NB and SVM.

Some aspect based sentiment analysis for games have also been studied by Urriza & Clariño (2021). Since aspect based are much finer grained, the study uses two different classifiers, to classify aspect and to classify sentiment. Some aspects extracted are in terms of the game’s graphics, audio and gameplay and to see the number of positive, negative, and neutral polarity for each aspect. However, this study’s approach is not too interested in aspect or entity level sentiment analysis.

There are also many sentiment analysis regarding other domain. Singh & Sarraf (2020) analysed customer reviews in terms of e-commerce due to inconsistency in product rating and product review. Therefore, the purpose of sentiment analysis here is to classify the review based on text of review rather

than ratings. Reviews were extracted from Flipkart.com by scraping the website. This research focuses on RF algorithm and uses Bag of Words (BoW). However, no clear conclusion can be made in this research.

Tiwari, Verma, Garg, & Bansal (2020) analysed sentiment analysis on twitter. This research compares RF, DT and SVM by using BoW on a sentence level sentiment analysis with a dataset of 40,000 collection gathered from GitHub and Kaggle. Instead of classifying sentiment in positive, neutral and negative, this paper classifies the sentiment in terms of emotions such as anger, boredom, empty, hate, love, etc. This study concluded that DT and RF are the best classifier with an accuracy of 99.3% and 99.4% respectively while SVM has 91.6%.

2.6.3 Summary of Discussion in Sentiment Analysis

It is important to emphasize on what Steam review is because although Steam review is a place such as on online e-commerce where people will talk about the reviews of the product, it is still denoted as an unprofessional site and sometime acts similarly as social media where people interact with each other rather than just read the review.

Therefore, aspect level is not looked thoroughly because it does not align with this research's goal. Whereas this study is focused to determine whether a person enjoy the game or not, regardless of the context reviewers leave and any additional feature such as hour played. Ergo, why this study is based on sentence level.

Furthermore, the closest related work chosen is Zuo (2018) by scrapping reviews through Seam API and to classify SA using the approach by Tan et al. (2021) were used in this research, however, negating the authors data on professional review. The preprocessing techniques can also be concluded based on earlier discussion.

2.7 Framework of Time Series Forecasting Modelling

Framework of churn prediction is not discussed in this topic because this study will not be using binary prediction.

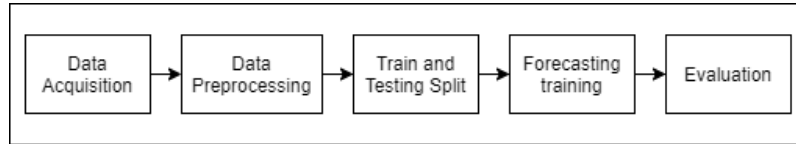


Figure 2.7 General flow of time series forecasting modelling

Figure 2.7 above display the general process of time series forecasting modelling and is explained on what each process does below.

a) Data Acquisition

Raw data contains date that acts as the index of time series and quantifiable values in the form of integers or float to be forecasted such as stock price, revenue, etc. This value can be single, that makes time series forecasting univariate, or multiple values that makes forecasting bivariate or multivariate, Time series data are usually historical data that spans over years.

b) Data Preprocessing

This step consists of determining stationary, transforming and performing granger causality. This section is discussed in detailed in Subtopics 2.7.2 and 2.7.3.

c) Train and Testing Split

The cleaned dataset is split into training and testing where training set are used train the classifier to produce the correct output, whereas testing set are used to evaluate its performance.

d) Forecasting training

Next, training set are fed to Vector Autoregression (VAR) where the main timeseries (ts) is related to its past values and past values of other ts to forecast and determine the lag order. Suitable lag can be found using Akaike

Information Criterion (AIC), and it is then fitted into the model.

e) Evaluation

Finally, the findings are evaluated. Some of the most common evaluation method is using Mean Average Percentage Error (MAPE).

2.7.2 Determining Stationary

When developing forecasting model, the time series is required to be stationary. Stationary refers to time series that does not rely on the time at which the series is observed such as time series with trends and seasonality.

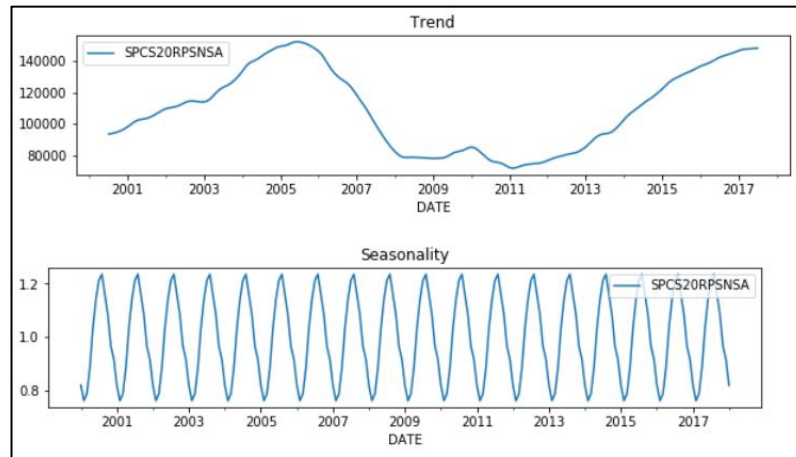


Figure 2.8 Graph for trend and seasonality

Stationary can be identified easier using unit root test called Augmented Dickey-Fuller test (ADF) and Kwiatkowski-Phillips-Schmidt-Shin (KPSS) test. Both of their purpose is to determine if time series is stationary or not by returning p-value based on a statistical equation. The difference is that they have different interpretation of p-value, opposite of each other. If time series is found to be non-stationary, it needs to be transformed by differencing operation.

2.7.3 Granger Causality Analysis

Granger causality analysis is a method to determine if a time series Granger cause another time series, in other words, whether a time series affect values

of other time series. It is a statistical concept that is based on prediction. According to Granger causality test, if a value X Granger cause value Y, then the past value of X should contain information that helps predict Y.

Granger can be tested with simple F-test with null hypotheses, which the null hypothesis being X does not Granger causes Y. Through F-test, p-value can be identified to determine if the correlated values have strong or weak correlation. In this case, if $p\text{-value} < \text{significance level of } 0.05$, then the hypothesis would be rejected which means, there are correlation between the two time series and what lag (a constant period that has passed) has the best correlation.

2.7.4 Vector Autoregression

Vector Autoregression (VAR) is platform of multivariate time series model that relates current observations with prior data of itself and other variables. It is a platform to perform Granger causality test. While Auto Regressive model is to forecasts based on only one variable based on its past value, VAR focuses on other variables that might affect another variable.

2.7.5 Forecasting Model

AIC is then used to determine the optimal lag. The value will be chosen to develop the forecasting model. Forecasting model depends on the lag that is identified. Besides VAR, other algorithm however can be chosen if it can do regression such as SVM or LSTM which are validated using MAPE (Park, Ma, & Leung, 2019; Zhang, Chuai, Gao, Liu, Maimaiti, & Si, 2019; Wang, Guo, Liu, & Fand, 2020; Napitu, Bijaksana, Trisetyarso, & Heryadi, 2018)

2.8 Discussion on Related Work for Prediction

Discussion regarding prediction will not be too in depth since later it is found that binary classification prediction is not to be used in this study.

A research by Rothmeier, Pflanzl, Hüllmann, & Preuss (2021) predicts player churn based on user activity. The study is about Freemium Online Strategy Game called The Settlers Online. It focuses on two challenges in prediction for game which are class labelling approach and what machine learning algorithm are best suited for the job. Churn definition for this game is found to be 14 days. The best model that uses RF classifier yield an accuracy more than 97% and AUC values more than 0.99.

Meanwhile, a study by Lee, Jang, Yoon, Jeon, Yang, Lee, Kim, Chen, Guitart, Bertens, Periañez, Hadiji Müller, Joo, Lee, Hwang, & Kim (2019) uses survival analysis for predicting whether a player would and when they would churn. The purpose of survival analysis is to predict the specific churn point of a player. Once the classification of whether a player would churn or not has been resolved, next involves a regression problem to predict the number of weeks a gamer would survive in the future.

On the other hand, Kiguchi, Saeed, & Medi (2022) domain was mostly on education purposes that incorporates online gaming, hence churn can still occur. A dataset from Japanese company that provides Digital Game-Based Learning service was used. Multiple classifiers such as LR, DT, and RF was used. Since the research is different in terms of game domain, but relate to education, time frame of churn such as 7, 14 or 30 days cannot be considered because education uses longer time frame. The research uncovers specific time frame for each user, and found that churn rate of Japanese service between churners and non-churners display different behaviour through retention analysis.

2.9 Discussion on Related Work for Forecasting

Research by Park et al. (2019) uses non-linear granger causality approach based on SVM to forecast stock price with sentiment fusion based on twitter. The sentiment that is obtained took into other feature such as retweet count and number of followers as sentiment weighting factor. This sentiment weighting factor is used to calculate the impact of the tweet either positive or negative.

Granger causality is used to find correlation between stock price and sentiment. Hence, the overall sentiment tweet is calculated daily instead of separating positive and negative neural using their formula.

Their equation then was input together with the historical data of stock price to forecast the next day's stock price. This study also finds the optimal SVM lag to be used for the forecasting model. The best lag was identified using p-value that is less than 0.1 for four different companies. It is concluded that incorporating nonlinear sentiment achieved the best performance with 1.15% error in MAPE.

Another study by Zhang et al. (2019) and Wang et al. (2020) uses LSTM in their research forecasting traffic in urban wireless communication network and service demand respectively. Research by Zhang et al. (2019) transform the problem into time series while retaining its spatial-temporal characteristics. A multivariate LSTM was used to forecast traffic and as a result from granger causality, it improved the accuracy of multivariate time series forecasting. Whereas Wang et al. (2020) discovered indicators by conducting Granger causality test, the indicators are trained based of historical data.

2.10 Summary of Related Work for Churn Prediction and Forecasting

It is observed that most churn prediction on telecom sector will use their customer profile to identify their behaviour, hence they are able to predict whether a customer will churn or not. Similarly, for games, researchers will use player log data which tells when a how long does a person play a game per day, how long is their playtime session interval, purchase behaviour if there are any, etc. In this case, it is not possible for this study because player log data is not accessible towards the public. To compensate, a different variable is used which is data of daily number of player within for a game.

In terms of churn prediction for games, there are no clear definition of churn unlike telecom company that has customer churn, or no churn status based on

their subscription. There are no explicit event that signals whether user stops using the service (Kristensen & Burelli, 2019). Typically, definition of churn for games are more than 6 days, and even 9 days, (Kiguchi et al., 2022; Perišić, Jung, & Pahor, 2022; Rothmeier et al., 2021). Therefore, it is safe to say that time taken for players to churn for video game are within a week and more, where this value is used for determining lag.

In addition, the churn prediction does not actually assist the scope of this proposed study, since there are restrictions to determine each specific player of a game and because this study is focused on daily number of players as an input, it is much more useful to forecast churn rate rather than churn prediction of a user. However, the valuable input for churn prediction literature would be to get the time frame of churn since there are no explicit definition of when a gamer will churn.

2.11 Churn Prediction Based on Sentiment Analysis

A study by El Kassem, Hussein, Abdelrahman, & Alsheref (2020) conducts sentiment analysis to analyse user opinion based on lexicons regarding telecom. The sentiment that is classified determines the probability whether a user will churn or not. No behavioural data is used in this study, rather the author create and analyse questionnaire as to what will causes a churn based on customer profile. Through the questionnaire, author find the weightage if each question and solution as to what cause them to churn most by using Deep learning, NB and LR.

Whereas Vo, Liu, Li, & Xu (2021) uses multi-stacking ensemble churn prediction to calculate churn score to indicate churn status for telecom company based on their customer profile and call log data. The call logs features extracted was term importance, phase embedding, lexical information and detecting personality traits. The result shows that customer call logs can be used to generate meaningful insights and best model that is proposed by them using XGB and multi-stacking performed the best in comparison to BERT by using all the features mentioned earlier with AUC score of 0.8124.

Furthermore, research by Napitu et al. (2018) predicts customer churn rate for a telecom company. The research obtained dataset on churn rate from the company that it was studying on, and sentiments related to the company from twitter. The churn rate and frequency of positive, negative and neutral was calculated per month. Granger causality found that the most correlated out of positive, negative, and neutral values with churn rate are negative sentiment analysis whereas positive and neutral sentiment has no significant effect. By using LSTM, they are able to forecast the churn rate with MAPE of 1.47% with lag that occurs 3 months later.

Research by Ranjan & Sood (2020) used only sentiment analysis to predict churn for telecom company in India for five different companies, however the result shown are actually the growth rate. This can be explained because the churn is under the assumption that when a company losses or gain customers, it can be inferred whether customer leave to subscribe another telecom company or go to their company. NB was used to classify the sentiment polarity, and prediction model utilized the overall sentiment score and predict customer churn on a monthly basis using a base value. The number of classified sentiments were calculated using their corresponding weights. Growth or churn depends on the previous month sentiment score. The results were validated by applying correlation analysis were within significance limits.

2.12 Summary on Literature Review

Overall, it is proven that sentiment improves accuracy of prediction (Park et al., 2019; Vo et al., 2021; Napitu et al., 2018). It is found that Napitu et al. (2018) research is a form of forecasting by utilizing sentiment analysis, the same with Park et al. (2019). It can be concluded that sentiment brings an insight towards the classifier and this approach can be used in this proposed study. Therefore, both sentiment analysis and churn forecasting need to be merged into synchronising time series.

Sentiment that is to be collected from Steam review through its API is then preprocessed and classified into positive or negative. This sentiment is then

calculated in terms of their respective weight. The purpose of determining weight is to determine the influence of the user. Therefore, the weight of sentiment in terms of Steam review need to deliver a similar purpose as in by Park et al. (2019) by calculating daily sentiment score using the weights. This weight can be obtained based on a variable that is determined by Steam themselves while scrapping the reviews called score.

SVM is a great classifier for this research because it can yield high accuracy even for unpredictable dataset such as Steam review because of SVM robustness nature therefore is suitable for the situation.

To forecast the daily player there needs to be a correlation between it and sentiment score. Once correlation is confirmed using Granger causality, only then optimal lag can be found, fed into the forecasting model and validated using MAPE.

Although previous work on churn prediction does not have significant impact on this research, it does serve as a measure of what time frame can be estimated for churn in game which is more than a week. This value is used to determine find best lag in Chapter 4.

CHAPTER THREE

RESEARCH METHODOLOGY

This chapter gives description of all the research phases from start to end in a detailed manner.

3.1 Research Framework

This section shows all the table methodologies that describe phases, activities, sources, deliverables for each objective.

Table 3.1 Research Framework

OBJECTIVES	PHASES	ACTIVITIES	SOURCES	DELIVARABLES
To design and develop a churn forecasting model that utilizes sentiment analysis.	Preliminary Study	Discuss with supervisor		<ul style="list-style-type: none">• Project title• Definition of domain• Problem statement• Research Questions• Research Objectives• Scope• Significance
		Determine problem domain	Online sources	
	Knowledge acquisition	Research and understand the problem	UiTM E-Resources Online Databases	<ul style="list-style-type: none">• How to collect data• Type machine learning algorithm to be used.• Understanding and idea of how to design model.• Literature review
		Research on techniques applied to solve the problem		

Table 3.1 (continued)

OBJECTIVES	PHASES	ACTIVITIES	SOURCES	DELIVARABLES
To design and develop a churn forecasting model that utilizes sentiment analysis.	Data collection	Retrieve Steam review	Steam platform using API	Raw data
		Retrieve daily player	SteamDB website	
	Data Preprocessing	Steam Review Remove: <ul style="list-style-type: none"> • Hyperlink • Most non alphanumeric characters • extra whitespace • stop words • Set to lowercase • Fix contractions • Tokenization • POS • Lemmatization • Change score 0 to 0.1 	Python Package <ul style="list-style-type: none"> • NLTK • re • contractions 	Cleaned data
		Time series <ul style="list-style-type: none"> • Difference transformation 		

Table 3.1 (continued)

OBJECTIVES	PHASES	ACTIVITIES	SOURCES	DELIVARABLES
To design and develop a churn forecasting model that utilizes sentiment analysis.	Model design	Design SA model with hyperparameter tuning method	SVM	Model architecture for SA
		Design CF model with the SA model	VAR	Model architecture for CF
	Model development	Develop SA model based on hyperparameter result	python	SA model with best parameter
		Develop CF model that utilizes the SA model		CF model that utilizes the SA model
To compare the result between churn forecasting model that does not utilize sentiment analysis model and churn forecasting model that utilizes the sentiment analysis model.	Testing and Evaluation	Evaluate the churn forecasting model	confusion matrix MAPE	Evaluated result Significance of CF model that utilizes the SA model
	Documentation	Write full report	Word doc	Final year report

3.2 Preliminary Study

This is the first phase to first objective began by carrying out activities such as discussion with supervisor and determining problem domain. Discussions were carried with supervisor with some inputs from online sources to specify problem domain that can be tackled to deliver project title, definition of domain, problem statement, research questions, research objectives, scope, and significance.

3.3 Knowledge Acquisition

The second phase of the first objective, is research and understand the problem domain that were decided in the previous phase and to research on the techniques applied to solve the problem. It was decided that the problem domain to be focussed on is player churn of games on Steam platform. Many research papers was studied, and some anchor paper concluded, were (Napitu et al., 2018), (Tan Jie et al., 2021) ,(Zuo, 2018), etc.

Furthermore, this paper proposed a sentiment analysis approach to forecast churn, hence the primary objective of this phase is to find out what causes a person to churn or not churn in games in terms of a person's emotion about a game rather than pattern of activity in game. Functional aspects and non-functional aspect, that also exist in other things, exist in games. Some examples are audio, graphics, etc. These are the aspects or entity that stirs an emotion when playing games. Although functional aspect has more impact to emotion, non-functional aspects cannot be excluded since a game that can bring enjoyment is an objective to any game, which requires both aspects to strive. These aspects create the overall design of a game. However, as discussed earlier not all reviews in steam may contain this information, in fact some may not contain any of this information. Therefore, to focus solely on these aspects may cause the data to be not sufficient to be fed into the forecasting model.

When it comes to predicting player churn for game, it is more complex since we do not have an explicit way to tell if a person churns or not churn since there are no subscription indication like telecom services has. Therefore, a time frame for when a player leaves a game must be initialized. When it comes to games, time frame varies as it has been discussed in literature review. It also varies for different games. Therefore, multiple forecasting model for each game may be needed. Finally, extensive research about the method on how to collect data, type of model to be used, and the general idea of how to create both SA and CF model were written and summarize in Subtopic 2.12 in the literature review summary.

3.4 Data Collection

Data that will be collected consist of structured and unstructured data. The reason why we use both structured and unstructured is explained in Subtopic 3.6. Structured data will be collected from a third-party tool called SteamDB that listen to the firehose of data the regular Steam client is sent. This data was obtained in an Excel file. However, a Steam account is required to access the data. The reason why SteamDB was selected is because it provides historical data for daily numbers of players for all games that is available in Steam which is important for forecasting purposes. Figure 3.3 and Figure 3.4 below show a sample of raw structured data before and after collection respectively.

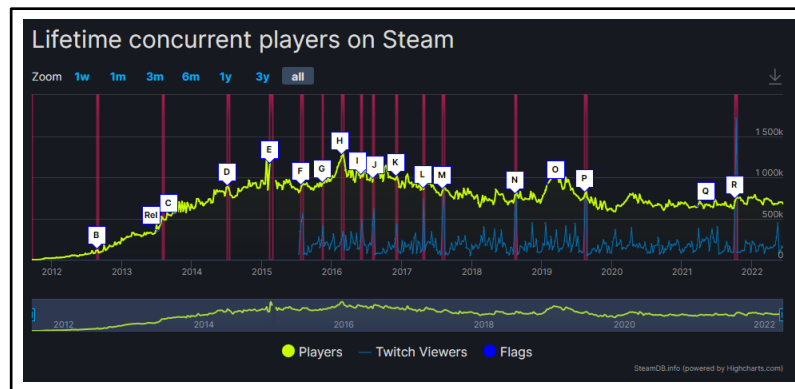


Figure 3.1 Sample for lifetime number of players on SteamDB
(Source: SteamDB)

	A	B	C	D	E	F	G
1	DateTime	Players	Twitch Viewers	Flags			
2	17/08/2011 00:00						
3	22/09/2011 00:00	194					
4	23/09/2011 00:00	240					
5	24/09/2011 00:00						
6	25/09/2011 00:00	233					
7	26/09/2011 00:00	222					
8	27/09/2011 00:00	303					
9	28/09/2011 00:00	312					
10	29/09/2011 00:00	299					
11	30/09/2011 00:00						
12	01/10/2011 00:00						
13	02/10/2011 00:00	376					
14	03/10/2011 00:00	344					
15	04/10/2011 00:00	389					
16	05/10/2011 00:00	466					

Figure 3.2 Sample number of daily players since 2011

Note that only “DateTime” and “Players” columns are considered. Meanwhile unstructured data will be obtained from Steam reviews through Steam API. By using Steam API, specific data that is required can be extracted with ease.

Steam API is selected compared to web scrapping is because it is less prone to error and takes less time to process. Figure 3.3 and Figure 3.4 below show a sample of raw unstructured data before and after collection respectively. Valve, owner of Steam, provide these for the sole purpose of using data from Steam in an interesting way.



Figure 3.3 An example of Steam review and their API

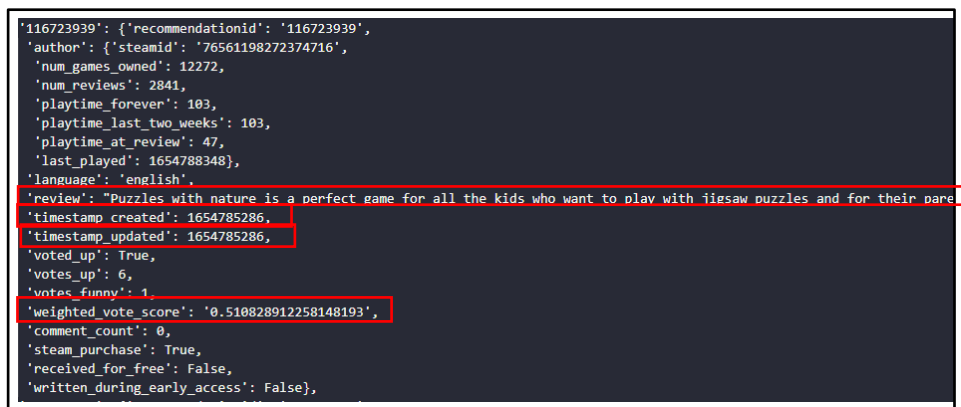


Figure 3.4 Sample of scrapped raw review through Steam API

Finally, the scrapped data were collected from 1st January 2020 to 31st December 2020, where it was stored into an excel file for easy access on later purposes as it saves time because there was no need to scrape multiple times.

review	sentiment	score	time
MONKE GO BRRRRRRRRRRRRR	TRUE	0	1609419231
gg	TRUE	0	1609419084
Fun little strategy game. Simple a	TRUE	0	1609418664
Recommended for toaster/potato p	TRUE	0	1609418509
j	TRUE	0	1609418509
hmm, monke	TRUE	0.5238095	1609418248
i just bought this and am now re liv	TRUE	0	1609416161
best game ive ever played	TRUE	0.5435684	1609415927
I recomend this game to people wh	TRUE	0	1609415777
Good	TRUE	0	1609415326
Very unrewarding :)	TRUE	0	1609415274
It was poggers	TRUE	0	1609415170
fun	TRUE	0.4761904	1609415125
Banana	TRUE	0	1609414842
Monke is sick of getting invaded by	TRUE	0.5238095	1609414412
very interactive, sometimes have to	TRUE	0	1609414186
fun	TRUE	0	1609414051
Just like Bloons TD 4 and 5, but m	TRUE	0.5238095	1609413942

Figure 3.5 Sample of semi-preprocessed review saved in Excel

Table 3.2 Attributes Selected based on the Scrapped Data

Attribute Raw	Attribute Semi-processed	Explanation
review	review	Textual data written by the user
timestamp_created	time	Date of the review posted in epoch date format
voted_up	sentiment	Boolean value, True indicating recommended and False indicating not recommended
weighted_vote_score	score	Value given by Steam to indicate the review's helpfulness, note that this attribute is not displayed in Figure 3.3 as it is only accessible through scrapping. This value is used to calculate the sentiment score

3.5 Data Preprocessing

Data preprocessing is important when it comes to anything that requires data processing. Unimportant text that will cause noise, and inefficiency which gives the classifier a hard time is removed. Below are the lists of data preprocessing methods that were applied, which goes in hand with the characteristics of Steam reviews.

- Remove hyperlink: Reviews that contains hyperlinks that leads to video or other sources does not contain any sentiment.

- b) Contraction: Fix contraction such as “don’t” by expanding it into its original word such as “do not”.
- c) Remove emoji: Emojis have impact on classifying sentiment. However, reviews on Steam may use emoji that contradicts with their review. For example, writing “Game is very bad :)” where “bad” indicates negative sentiment but “:)” indicates positive sentiment. In addition, this study does not consider emojis.
- d) Set to lowercase: To normalize size of words.
- e) Remove most non alphanumeric characters: Remove special characters with the exclusion of punctuation marks and commonly used symbols as they do not contain actual value.
- f) Remove extra whitespace: Accidental extra white space or to show paragraphs is only for readers purposes and is not needed for processing.
- g) Remove stop word: Remove most stop words except negations such as “no” and “not” since it would invert the sentiment of the review.
- h) Tokenization: Helps to understand the context of words by breaking the apart to be analysed in sequence
- i) Part-of-Speech tagging: Give context to each token
- j) Lemmatization: Receive POS tags so that lemmatization is carried out based on context of tokens.
- k) This study proposed to use the sentiment score in a time series like manner. Score of 0 cannot be overlooked. If score is 0, then it is changed to 0.01, to consider the sentiment rather than ignoring it completely.

Meanwhile, timeseries is transformed via differencing if it is not stationary which is determined using Augmented Dickey-Fuller test (ADF) and

Kwiatkowski-Phillips-Schmidt-Shin (KPSS).

3.6 Model Design

This section discusses about the fifth phase which is the model design. The primary goal of this research is to develop a churn forecasting model that utilizes sentiment analysis. However, there is a lack in usage of number of players as an input to forecasting churn in video game domain. Hence this study also determines whether a correlation between the two inputs exist or not.

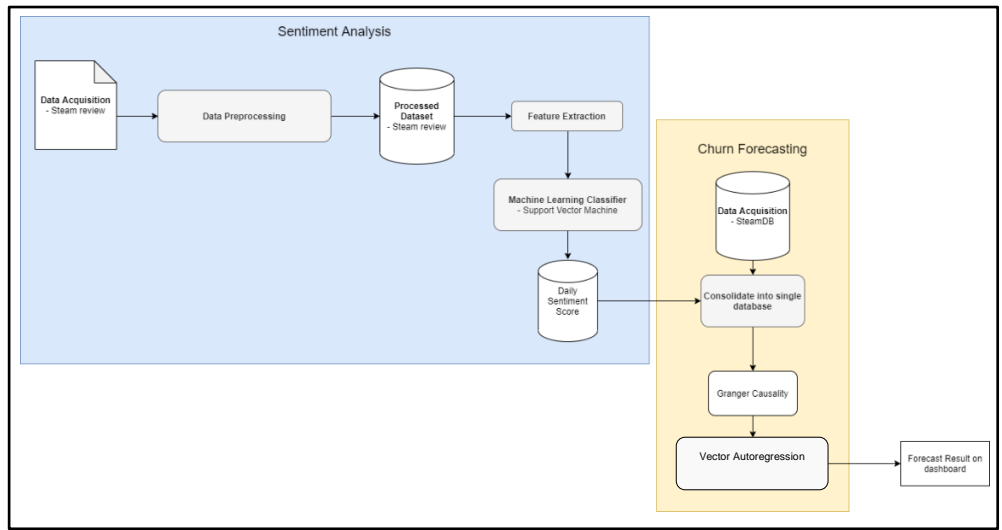


Figure 3.6 Model architecture

Figure 3.6 above shows the model which can be viewed from two separate process. After classifying Steam review to determine its sentiment, which is highlighted in the blue box, it is then calculated into daily sentiment score, *dSentScore*, using the following formula similar to the research by Park et al. (2019) as discussed in the literature review, starting from beginning of the collection date starting from 1st January 2020 to 31st December 2020. The formula is written as formula number 1.

$$dSentScore = \log \left(\frac{1 + \sum wPos}{1 + \sum wNeg} \right) \quad (1)$$

where,

wPos	score of positive review
wNeg	score of negative review

Note that score is taken from the “score” column in the semi-preprocessed Excel file that acts as the weight to determine helpfulness of review as discussed in Subtopic 2.12.

Next, both daily values of number of players from SteamDB dataset and daily sentiment score calculated were inserted into the consolidated into a single dataset. Depending on the timeseries, differencing was performed using the following formula.

$$differencedDate = currentDay - nextDay \quad (2)$$

Upon differencing, the 1st day (1st January 2020) did not hold a value, and only 2nd January 2020 until 31st December holds a value.

Only then multiple churn forecasting model was developed based on the games identified using vector autoregression.

3.6.1 Sentiment Analysis Model

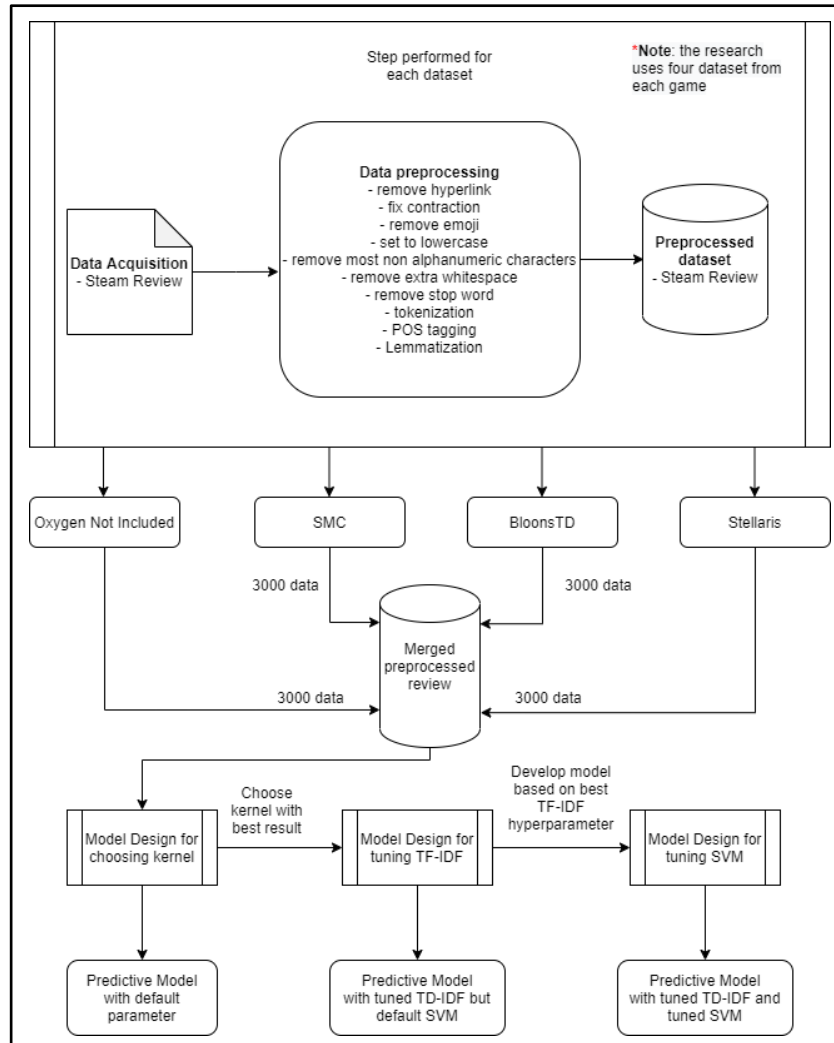


Figure 3.7 Sentiment analysis model

Figure 3.7 shows the sentiment analysis model. After data were acquired and preprocessed from Steam based on the preprocessing step in subtopic 3.5 for the sentiment part, it was then labelled by using user given review based on “voted_up” attribute from Table 3.2. The preprocessed data was saved into their own repository for each game. These steps were repeated for all four games.

A number of 3000 of the preprocessed data from each games were selected and merged, totalling to 12000 data in a single dataset, to increase the diversity of game vocabulary for better training. All negative reviews were selected before positive reviews to prevent positive reviews from filling up the entire dataset.

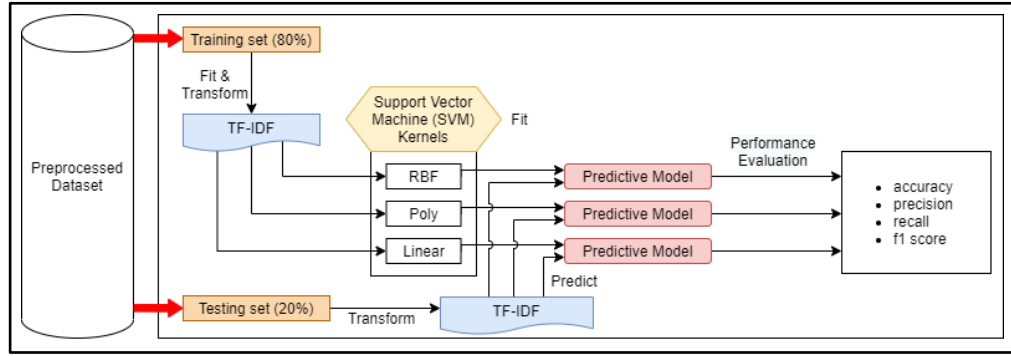


Figure 3.8 SVM kernel comparison

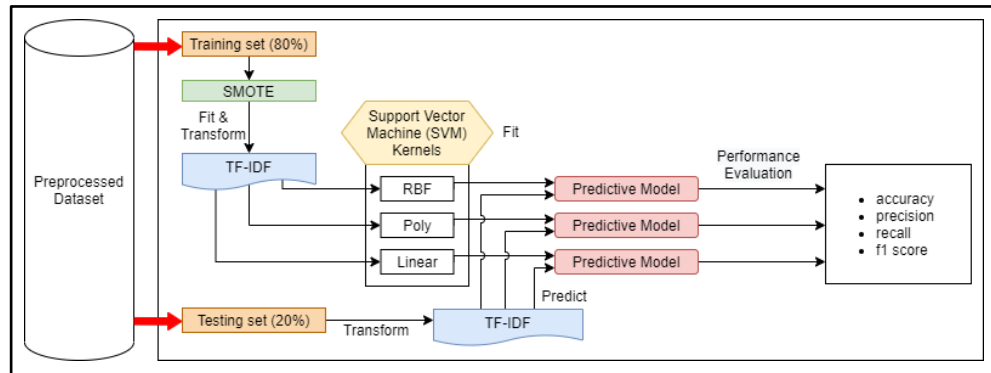


Figure 3.9 SVM kernel comparison with SMOTE

Next, data was split into 80:20 ratio after data is shuffled, where it was sent to Term Frequency-Inverse Document Frequency (TF-IDF) for converting the reviews to a matrix of TF-IDF features and putting them in a vector. This was the standard process for SA modelling throughout this research as displayed in Figure 3.8, 3.9, 3.10 and 3.11.

In Figure 3.8 and 3.9, Support Vector Machine (SVM) with linear, radial basis function (RBF), and polynomial (poly) kernel was deployed to compare and evaluate against one another where all these comparisons had default parameter for TF-IDF and SVM. In addition, Figure 3.9 displays the use of Synthetic Minority Over-sampling Technique (SMOTE) for the same process to see if oversampling were able to increase the recall of the model by handling class imbalance through distribution adjustment. This section is the “Model Design for choosing kernel” in Figure 3.7

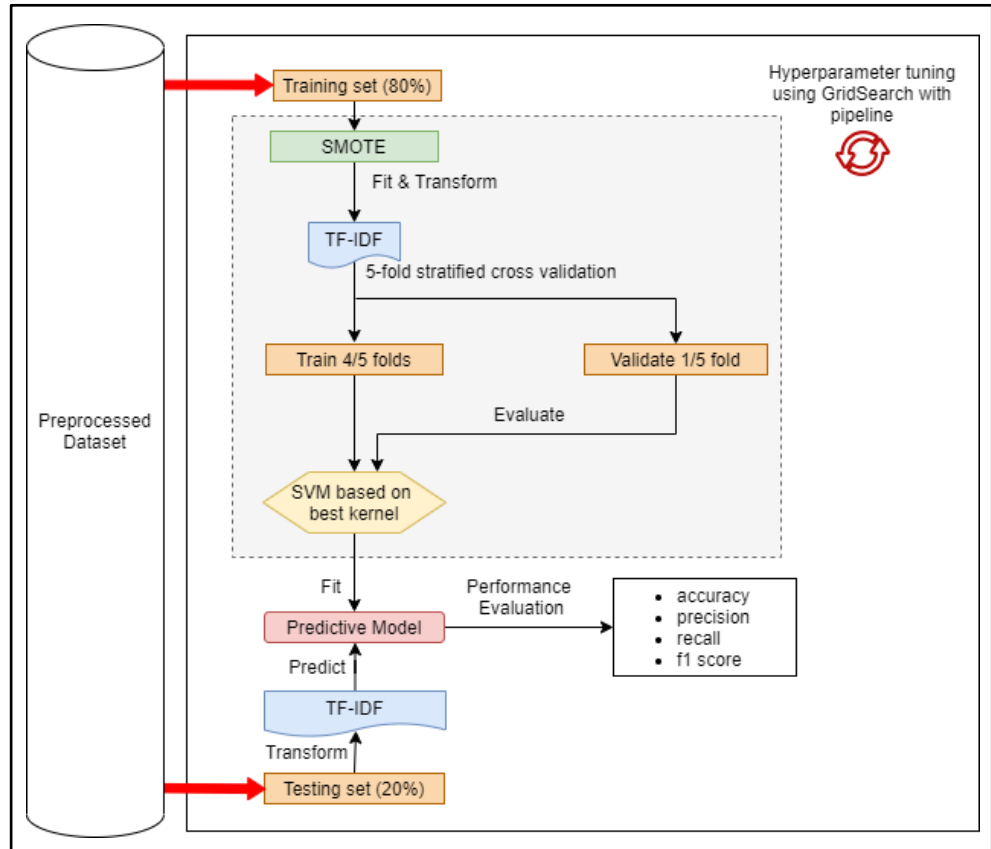


Figure 3.10 TF-IDF hyperparameter tuning using grid search

From now on, this research will use “best kernel” as reference to SVM kernel with best performance evaluated. Best kernel was selected for SVM and TF-IDF hyperparameter tuning.

It is common that cross validation (CV) is used with hyperparameter tuning. CV is a statistical method to evaluate the accuracy of model. It is an improvement of the conditional training and testing split where test data will not be trained. CV will iterate based on number of folds. Every data will be trained and tested (Elgeldawi, Sayed, Galal, & Zaki, 2021). For example, while 9 folds are trained with tuned hyperparameter, 1-fold is evaluated.

Based on Figure 3.10, grid search trained the model based on every possible combination of hyperparameter defined (Elgeldawi et al., 2021). Therefore, an optimized values were able to be obtained. It is used for TF-IDF because it has very little combination of hyperparameter for max feature, max df and ngram range. A pipeline is a series of sequence to be processed and is used because it provides a structure especially for grid search hyperparameter tuning, where

the grey box with dotted line is the pipeline. This section is the “Model Design for tuning TF-IDF” in Figure 3.7

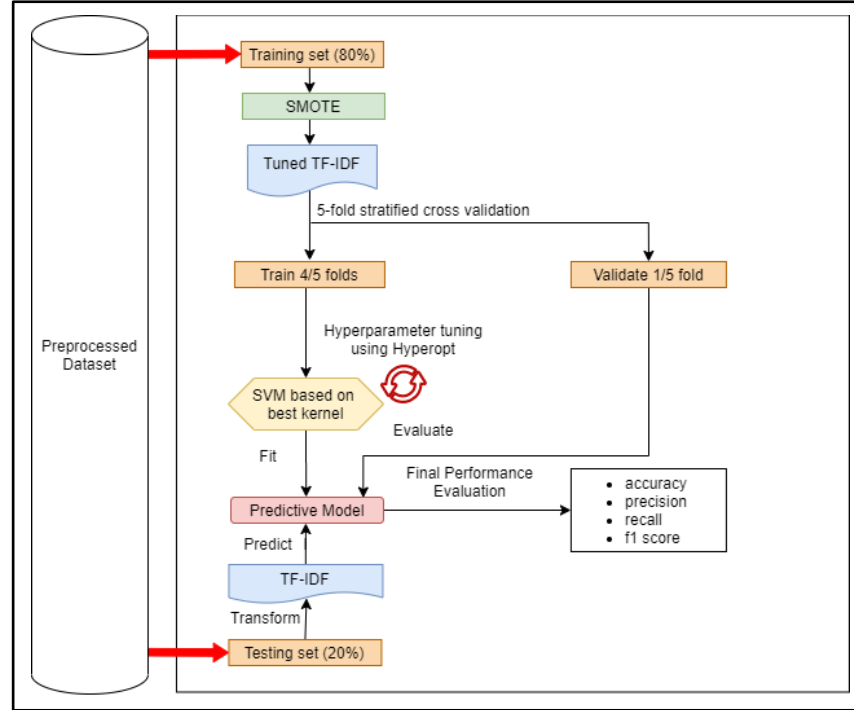


Figure 3.11 SVM hyperparameter tuning with hyperopt

On the other hand, hyperopt embedded with Bayesian optimization were used for SVM as displayed in Figure 3.11. It was used by defining objective function, search space, and search algorithm. Objective function goal used were to minimize lost therefore finding the best parameter to its ability by finding the parameter with smallest possible value of the lost function. Since parameter such as ‘C’ and ‘gamma’ have wide range of values, stochastic expressions such as ‘qloguniform’ is most suitable since the samples generated will be distributed uniformly in log-space that causes search to be smooth. Lastly search algorithm used were Tree-structured Parzen Estimator as it was proven to be able to perform well in a wide range of problems. This section is the “Model Design for tuning SVM” in Figure 3.7. In comparison to grid search, hyperopt does not require a pipeline. Both hyperopt and grid search hyperparameter tuning uses cross-validation of 5 folds due to computation time constraints.

Every time the model is tuned, it was evaluated, and the evaluations were compared. Based on the comparison, the final model with best result were selected as the sentiment model.

To summarize, SVM and TF-IDF used hyperparameter tuning to find the model with best parameter, whilst SMOTE was applied to balance the distribution. Several different models became result of this process which were used in the evaluation phase.

3.6.2 Churn Forecasting Model

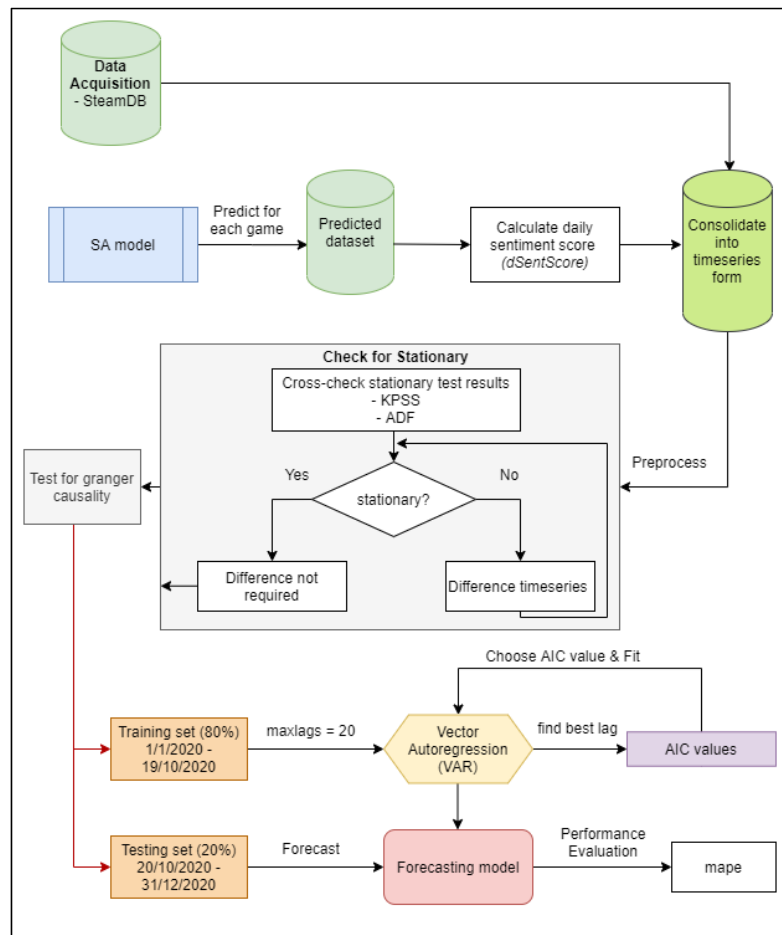


Figure 3.12 Churn forecasting model

Figure 3.12 shows the churn forecasting model. After using the sentiment analysis (SA) model to classify the reviews, the formula for calculating daily sentiment score were applied in the collected dataset producing daily sentiment score from 1st January 2020 to 31st December 2020. Then, data of daily player

that have been collected from SteamDB and daily sentiment score calculated were consolidated into a single repository in a timeseries like manner where it has three columns, the first and second columns are daily sentiment score calculated and daily number of player respectively, whereas the third column is date time corresponding to the data consolidated such as 2020-1-1- to indicate 1st day January 2020. It was consolidated by calculating the sentiment score obtained by identifying the same date to use *dSentScore* formula so that it only has one score per day (sentscore), then was matched with daily player data from SteamDB that has the same date so it will be on the same row as in Figure 3.13 (Refer Figure 4.15 for clearer process). This was applied for each game within this research scope.

sentscore	peakplayer	datetime
1.875667	30092	2020-12-31
1.901211	32274	2020-12-30
1.844825	29997	2020-12-29
1.789881	29605	2020-12-28
1.784099	28667	2020-12-27
1.771479	25662	2020-12-26
1.611653	22622	2020-12-25
1.670142	19776	2020-12-24
1.56526	17575	2020-12-23
1.416677	13437	2020-12-22
1.239139	11318	2020-12-21
1.258936	11800	2020-12-20
1.260155	11553	2020-12-19
1.248596	11357	2020-12-18
1.359532	11382	2020-12-17
1.098309	11612	2020-12-16

Figure 3.13 Sample data of consolidated dataset

Both time-series were checked whether it was stationary, if not, then the time-series was transformed by conducting difference operation until it became stationary. Stationary was checked using ADF and KPSS where the results from both tests were cross checked to ensure it was truly stationary.

After stationary was conducted, Granger Causality (GC) test was conducted to determine whether sentiment score and number of players has correlation. It is a test to see if one time series have a predictive power of another time series, in this case it is to investigate whether time series of sentiment score was able to assist in forecasting number of player.

The null hypotheses of Granger causality state that if X does not Granger Cause time series Y, then there is no correlation. Probability value (p-value) can be used to reject the null hypothesis, if p-value is less than significance level of 0.05, then the null hypothesis is rejected which means time series X is useful to forecast time series Y. In this case, if time series sentiment score Granger causes time series number of player, the null hypothesis can be rejected.

All timeseries was split into 80:20 ratio for training and testing. Unlike SA, train and test split data was not shuffled due to it being a timeseries data, therefore the train data was split into training set starting from 1st January 2020 until 18th October 2020, whereas testing set begin from 19th October 2020 until 31st December 2020. However, when a shift occurred because of differencing, training and testing data began on 2nd Jan until 19th Oct, and 20th Oct until 31st December respectively.

Lag is a fixed time frame in a time series, such as 1 day, and increment on that number for lag-2 to be 2 days, if lag is 1 week, then lag-2 will be 2 weeks, and so on. Train dataset was fed into VAR model with lag of 20 to find the best lag and a result of best lag was observed based on AIC. Lag 20 was selected to find best lag because the retention rate for game last for up to 10 days. In addition, using higher lag was able to train the VAR model better as higher lag means more complex computation, therefore VAR punishes model with higher lag where optimal lower lag is produced. The model was fitted with the best lag found based on AIC. This process was repeated for all the other game.

3.7 Model development

In the model development phase, the model was developed based on their design that has been explained earlier, a couple of software was identified and used for specific purposes. Both SA and CF model was developed using completely using python and its packages and modules. Algorithms in the form of pseudocode are also displayed in this section.

Both SA and CF model requires data to be imported from Excel file which was done using pandas package. The data was transformed into data frame where data were easier to access. This process was used many times at the beginning of processes such as before preprocessing for SA, before splitting training and testing, and merging preprocessed data. To save any changes to the data frame into Excel file, xlwt package was used.

Raw data was scrapped by using request package that accessed the url through 'https://store.steampowered.com/appreviews/' to scrape the data and saved into Excel file as semi-processed data in Figure 3.14. The saved semi preprocessed data (only certain data was selected to be saved) was preprocessed using nltk, re and contractions packages and saved into new excel file as shown in Figure 3.15.

```
Input:
  appid: game ID
  n: number of reviews to be scraped
  gameName: name of the game for easy saving purposes
Output:
  review, date time in epoch format, score, and sentiment saved in excel file

ALGORITHM:

Begin
SET reviews TO []
SET cursor TO '*'
SET params TO {
SET d0 TO date(2020,1,1) #start date to collect data
SET d1 TO date.today()
SET delta TO d1 - d0 #duration of days to collect data

    'json' : 1,
    'filter' : 'recent',
    'day_range' : delta.days, #collect data from 1/1/2020
    'review_type' : 'all',
    'language' : 'english',
    'purchase_type' : 'all'
}

WHILE n > 0:
  SET params['cursor'] TO cursor.encode()
  SET params['num_per_page'] TO 100

  n -= 100
  SET url TO 'https://store.steampowered.com/appreviews/'
  SET response TO requests.get(url=url+appid, params= 'json':1, headers={'User-Agent': 'Mozilla/5.0'})
```

```

SET response TO response.json()
SET cursor TO response['cursor']
reviews += response['reviews']
IF len(response['reviews']) < 100: break
SET json_data TO reviews

IMPORT xlwt
from xlwt IMPORT Workbook
SET wb TO Workbook()
SET sheet1 TO wb.add_sheet('Sheet 1')
sheet1.write(0,0,"review")
sheet1.write(0,1,"sentiment")
sheet1.write(0,2,"score")
sheet1.write(0,3,"time")

SET row TO 0
#GET RAW DATA , INSERT , AND SAVE TO EXCEL FILE SIMULTANEOUSLY
FOR record IN json_data:
    IF record["timestamp_created"] >= 1577840461 and record["timestamp_created"] <=
1609419599 : #timestamp between 1/1/2020 and 31/12/2020
        row += 1
        sheet1.write(row,0,record["review"])
        sheet1.write(row,1,record["voted_up"])
        sheet1.write(row,2,record["weighted_vote_score"])
        sheet1.write(row,3,record["timestamp_created"])
    SET saveFile TO gameName + ".xlsx"
    wb.save(saveFile)
End

```

Figure 3.14 Pseudo-code for scrapping algorithm

```

Input:
    rawFile: name of the file that contains the raw collected data.
Output:
    saved preprocessed review of the game in xls file

ALGORITHM:

Begin
SET Corpus TO pd.read_excel(rawFile + ".xlsx")
# iterate through excel

#for saving into excel

SET wb TO Workbook()

SET sheet1 TO wb.add_sheet('Sheet 1')

SET i TO 0

sheet1.write(0,0,"review")
sheet1.write(0,1,"sentiment")
sheet1.write(0,2,"score")
sheet1.write(0,3,"time")

```



```

SET word TO re.sub(r'^A-Za-z0-9 ]+', '', word)
IF word not IN stop_words and word.isalpha():
    SET word_Final TO word_Lemmatized.lemmatize(word,tag_map[tag[0]])
    Final_words.append(word_Final)

SET review TO str(" ".join(Final_words))
SET sentiment TO Corpus['sentiment'][index]
SET score TO Corpus['score'][index]
SET time TO Corpus['time'][index]

IF review EQUALS "nan":
    SET review TO ""

IF len(review) != 0 and len(review) >= 6:
    sheet1.write(i+1,0,review)
    sheet1.write(i+1,1,bool(sentiment))
    sheet1.write(i+1,2,score)
    sheet1.write(i+1,3,int(time))
    i += 1

wb.save("Preprocessed" + rawFile + "v4.xls")
End

```

Figure 3.15 Pseudo-code for preprocessing algorithm

Before merging process, reviews from each game were selected and saved into a different Excel file based on Figure 3.16, only then the saved Excel file is imported for merging based on Figure 3.17.

```

Input:
    reviewWant: amount of view selected
    gameName: name of the game for easy saving purposes
Output:
    review, and sentiment saved in excel file

ALGORITHM:

Begin
SET wb TO Workbook()
SET df TO pd.ExcelFile("Preprocessed"+gameName+".xls").parse()

SET sheet1 TO wb.add_sheet('Sheet 1')
SET le TO LabelEncoder()
SET df['sentiment'] TO le.fit_transform(df['sentiment'])

SET sentVal TO Counter(df['sentiment'])
SET sent TO sentVal.most_common()
SET numTrue TO sent[0][1] #get number of true sentiment
SET numFalse TO sent[1][1] #get number of false sentiment

#number of true review to be collected. only collect when numFalse is completely
collected.

```

```

SET numTrue TO reviewWant - numFalse
SET tr TO 0
SET fl TO 0
SET excelRow TO 0
sheet1.write(0, 0, "review")
sheet1.write(0, 1, "sentiment")

FOR i IN range(len(df)):
    SET review TO (df['review'][i]) #col 1 to get review
    SET sentiment TO (df['sentiment'][i]) #col 1 to get review
    IF sentiment EQUALS 0 and fl < numFalse:
        IF fl <= numFalse:
            fl += 1
            sheet1.write(excelRow+1, 0, str(review))
            sheet1.write(excelRow+1, 1, str(sentiment))
            excelRow += 1
        ELSEIF sentiment EQUALS 1:
            IF tr < numTrue:
                tr += 1
                sheet1.write(excelRow+1, 0, str(review))
                sheet1.write(excelRow+1, 1, str(sentiment))
                excelRow += 1
wb.save("selected" + gameName + ".xls")
End

```

Figure 3.16 Pseudo-code for selecting reviews algorithm

```

Input:
    doc1: selected review for BloonsTD6 in Excel file
    doc2: selected review for Oxygen Not Included in Excel file
    doc3: selected review for Stellaris in Excel file
    doc4: selected review for Civilization VI in Excel file
Output:
    merged review in Excel file

ALGORITHM:

IMPORT pandas as pd
IMPORT seaborn as sns

from collections IMPORT Counter
from sklearn.preprocessing IMPORT LabelEncoder

IMPORT xlwt
from xlwt IMPORT Workbook

SET wb TO Workbook()
SET sheet1 TO wb.add_sheet('Sheet 1')

sheet1.write(0, 0, "review")
sheet1.write(0, 1, "sentiment")

SET excelRow TO 0
SET doc TO 0

WHILE doc < 4:
    IF doc EQUALS 0:

```

```

    SET df TO pd.ExcelFile(doc1).parse()
ELSEIF doc EQUALS 1:
    SET df TO pd.ExcelFile(doc2).parse()
ELSEIF doc EQUALS 2:
    SET df TO pd.ExcelFile(doc3).parse()
ELSE:
    SET df TO pd.ExcelFile(doc4).parse()

FOR i IN range(len(df)):
    SET review TO (df['review'][i]) #col 1 to get review
    SET sentiment TO (df['sentiment'][i]) #col 1 to get review
    sheet1.write(excelRow+1, 0, str(review))
    sheet1.write(excelRow+1, 1, str(sentiment))
    excelRow += 1

doc += 1

#name of merged file
wb.save("merged.xls")

```

Figure 3.17 Pseudo-code for merging reviews algorithm

The merged preprocessed was splitted using sklearn `train_test_split()` where train and test values for review and sentiment were obtained which then the features were extracted. Review contains the input text data whereas sentiment was the corresponding label to the text data.

Features extraction (TF-IDF) was performed using sklearn python library which is `TfidfVectorizer`. The extracted feature were transformed into array, therefore there was a need to transform sentiment label into array aswell using numpy. SVM with linear, polynomial, and RBF that is available in sklearn was deployed.

Since many testings were required to be conducted, the general flow from training and testing split to displaying results were defined into separate functions for reusable purposes.

```

Input:
    merged review filename
Output:
    X: input data
    Y: label data

ALGORITHM:

DEFINE FUNCTION readFile(fileName):

```

```

SET df TO pd.read_excel(fileName)
SET X TO df['review']      #getting review column
SET Y TO df['sentiment']   #getting sentiment column (label)

RETURN X,Y

```

Figure 3.18 Pseudo-code for reading file to get input data and label

```

Input: merged review filename
Output:
word_train: train set of input data
word_test: test set of input data
sentiment_train: train set of label data
sentiment_test: test set of label data

ALGORITHM:

DEFINE FUNCTION training(fileName):

    SET X,Y TO readFile(fileName)

    SET word_train, word_test, sentiment_train, sentiment_test TO train_test_split(X, Y,
        test_size=0.2, random_state=0, stratify=Y)

    # Check the number of records IN training and testing dataset.

    RETURN word_train, word_test, sentiment_train, sentiment_test

```

Figure 3.19 Pseudo-code for train and test split algorithm

```

Input:
filename: merged review filename
vect: instantiation of TfidfVectorizer
word_train: train set of input data
word_test: test set of input data
sentiment_train: train set of label data
sentiment_test: test set of label data

Output:
word_train_vec: transformed and vectorized train set of input data
word_test_vec: transformed and vectorized test set of input data
sentiment_train_vec: vectorized train set of label data
sentiment_test_vec: vectorized test set of label data
vecWord: transformed and vectorized of whole input data
vecSentiment: vectorized of whole label data

ALGORITHM:
DEFINE FUNCTION vectorizer(fileName, vect, word_train, word_test, sentiment_train,
    sentiment_test):

    SET X,Y TO readFile(fileName)

    SET word_train_vec TO vect.transform(word_train)
    SET sentiment_train_vec TO np.array(sentiment_train)

    SET word_test_vec TO vect.transform(word_test)
    SET sentiment_test_vec TO np.array(sentiment_test)

```



```

SET vecWord TO vect.transform(X)
SET vecSentiment TO np.array(Y)

OUTPUT("n_samples:" , len(word_train) , "n_features:" ,
      len(vect.get_feature_names()))

RETURN word_train_vec, sentiment_train_vec, word_test_vec , sentiment_test_vec,
vecWord , vecSentiment

```

Figure 3.20 Pseudo-code for vectorizing transformed features

```

Input:
word_train: transformed and vectorized train set of input data
sentiment_train: vectorized train set of label data
model: instantiation of classifier

Output: fitted classifier into model

ALGORITHM:
DEFINE FUNCTION classifier(word_train, sentiment_train, model):
    model.fit(word_train, sentiment_train)

```

Figure 3.21 Pseudo-code for fitting into classifier

```

Input:
wordCount: transformed and vectorized of whole input data
sentimentValues: vectorized of whole label data
y_test: vectorized test set of label data
X_train: transformed and vectorized train set of input data
X_test: transformed and vectorized test set of input data
y_train: vectorized train set of label data
modelFit: model with fitted train input and label data
Output: Display metrics within confusion matrix

ALGORITHM:

DEFINE FUNCTION
displayResult(wordCount,sentimentValues,y_test,X_train,X_test,y_train, modelFit):

    OUTPUT(Counter(y_train))
    OUTPUT(sns.countplot(y_train))

    SET X TO wordCount
    SET Y TO sentimentValues

    SET kfold TO KFold(shuffle=True, n_splits=10)
    SET cv_score TO cross_val_score(modelFit,X,Y, cv TO kfold)
    SET meanAccuracy TO round((sum(cv_score)/len(cv_score))*100,2)

    OUTPUT("Mean Accuracy:" , meanAccuracy)

    SET word_Testprediction TO modelFit.predict(X_test)
    OUTPUT(classification_report(y_test, word_Testprediction))
    SET diff TO cv_score.mean() - modelFit.score(X_test, y_test)
    SET SD TO diff / cv_score.std()

    OUTPUT(f"Training Score:{modelFit.score(X_train, y_train)}")
    OUTPUT(f"Cross V Score: {cv_score.mean()} +/- {cv_score.std()}")

```

```

OUTPUT(f"Testing Score: {modelFit.score(X_test, y_test)}")
OUTPUT(f"Cross & Test Diff: {diff}")
OUTPUT(f"Standard Deviations Away: {SD}")
OUTPUT(confusion_matrix(y_test, word_Testprediction))

```

Figure 3.22 Pseudo-code for displaying result

All the functions from Figure 3.18 to Figure 3.22 were used in a single block of code for testing where each block changes the type of kernel as displayed in Figure 3.23.

```

Input
  Filename
Output:
  Developed the model
  Display validation of the model

SET word_train, word_test, sentiment_train, sentiment_test TO training(fileName)

SET vect TO TfidfVectorizer().fit(word_train)

SET word_train_vec, sentiment_train_vec, word_test_vec, sentiment_test_vec, vecWord,
vecSentiment TO vectorizer(fileName, vect, word_train, word_test, sentiment_train,
sentiment_test)

#model was defined based on the kernel used.
SET defModel TO svm.SVC(kernel='linear', random_state=0) #note, kernel is changed
for poly and RBF for other testing

CALL classifier(word_train_vec, sentiment_train_vec, defModel)
CALL displayResult(vecWord, vecSentiment,
sentiment_test_vec, word_train_vec, word_test_vec, sentiment_train_vec, defModel)

```

Figure 3.23 Pseudo-code for modeling with default parameters

For further testing to implement SMOTE using imblearn, another function was required to be defined unlike the function in Figure 3.19.

```

Input: merged review filename
Output:
  word_train: train set of input data
  word_test: test set of input data
  sentiment_train: train set of label data
  sentiment_test: test set of label data

ALGORITHM:

DEFINE FUNCTION trainingSmote(fileName):
  SET smt TO SMOTE(random_state=0)

```

```

#ONLY ON training data, not test data.
SET word_train, word_test, sentiment_train, sentiment_test TO training(fileName)

#note this value is same as previous tfidf
SET vect TO TfidfVectorizer().fit(word_train)

SET word_Smote_vec, sentiment_Smote_vec, word_test_vec , sentiment_test_vec,
vecWord , vecSentiment TO vectorizer(fileName, vect,word_train, word_test,
sentiment_train, sentiment_test)

SET word_Smote_vec , sentiment_Smote_vec TO smt.fit_resample(word_Smote_vec,
sentiment_Smote_vec)

RETURN word_Smote_vec, sentiment_Smote_vec, word_test_vec ,
sentiment_test_vec, vecWord , vecSentiment

```

Figure 3.24 Pseudo-code for train test split and apply SMOTE

Similar to the process in Figure 3.23, the “training” function in Figure 3.24 is replaced with “trainingSmote” function based in Figure 3.24. Kernel was still changed and tested for linear, poly, and RBF in Figure 3.25.

```

SET smoteModel TO classifier with default value

SET word_Smote_vec, sentiment_Smote_vec, word_test_vec , sentiment_test_vec,
vecWord , vecSentiment TO trainingSmote(fileName)

CALL classifier(word_Smote_vec, sentiment_Smote_vec,smoteModel)

CALL displayResult(vecWord, vecSentiment,
sentiment_test_vec,word_Smote_vec,word_test_vec,sentiment_Smote_vec,smoteModel)

```

Figure 3.25 Pseudo-code for modelling with smote algorithm

Regarding hyperparameter tuning, TF-IDF was fed into a pipeline using imblearn. Pipeline is a method to streamline the process which makes hyperparameter tuning for TFIDF easier. The values that were trained on GridSearch is displayed in Table 3.3 below Whereas the pseudocode is displayed in Figure 3.27.

Table 3.3 TF-IDF Hyperparameter Values

Hyperparameter	Test Values
Max_features	2000, 2500,5000,10000,15000
Max_df	0.25,0.5,0.75,1
Ngram_range	(1,1), (1,2), (1,3)

```

SET pipeline TO Pipeline([
    ('tfidf', TfidfVectorizer()),

    #SMOTE BECAUSE WE WILL BE USING SMOTE DATA
    ('sampling', SMOTE(random_state=0)) ,

    #USE THE TUNED Classifier
    ('clf', tuned classifier
])

SET parameters TO [{
    'tfidf__max_features': (2000, 2500, 5000, 10000, 15000),
    'tfidf__max_df': (0.25, 0.5, 0.75),
    'tfidf__ngram_range': [(1, 1), (1,2), (1, 3)]

}]

```

Figure 3.26 Pseudo-code for creating pipeline for tf-idf hyperparameter tuning

```

Input
    pipeline: defined pipeline
    parameters:
Output: Best parameter
SET gs TO GridSearchCV(pipeline, parameters, cv=5, n_jobs=-1, verbose=10)

gs.fit(word_train,sentiment_train)

# best estimator to fit
OUTPUT(gs.best_estimator_.steps)
SET best_clf TO gs.best_estimator_

```

Figure 3.27 Pseudo-code for grid search algorithm

After obtaining the best parameter based on grid search, it was inserted into the TF-IDF hyperparameter as displayed in Figure 3.28 and was used for modelling in Figure 3.29.

```

DEFINE FUNCTION trainingSmoteVectorizerTUNED (fileName):

    SET smt TO SMOTE(random_state=0)
    SET word_train, word_test, sentiment_train, sentiment_test TO training(fileName)

```

```

# use found tfidf parameter values
SET vect TO TfidfVectorizer(best parameter found).fit(word_train)

SET word_Smote_vec, sentiment_Smote_vec, word_test_vec , sentiment_test_vec,
vecWord , vecSentiment TO vectorizer(fileName, vect,word_train, word_test,
sentiment_train, sentiment_test)

SET word_Smote_vec , sentiment_Smote_vec TO smt.fit_resample(word_Smote_vec,
sentiment_Smote_vec)

#major difference here we RETURN vect because need to use the vect.transform for
prediction

RETURN word_Smote_vec, sentiment_Smote_vec, word_test_vec ,
sentiment_test_vec, vecWord , vecSentiment, vect

```

Figure 3.28 Pseudo-code of function with tuned tf-idf with SMOTE

```

SET classifierTunedVect TO defaultClassifier with best kernel

SET word_Smote_vec, sentiment_Smote_vec, word_test_vec , sentiment_test_vec,
vecWord , vecSentiment , vectSMOTETFIDF TO trainingSmoteVectorizerTUNED
(fileName)

CALL classifier(word_Smote_vec, sentiment_Smote_vec,classifierTunedVect)

CALL displayResult(vecWord, vecSentiment,
sentiment_test_vec,word_Smote_vec,word_test_vec,sentiment_Smote_vec,classifierTune
dVect

```

Figure 3.29 Pseudo-code for modeling with tuned tf-idf with SMOTE

Meanwhile, hyperparameter tuning for SVM was trained using hyperopt which required some requirement to be defined. The requirements defined are displayed in Table 3.4 below.

Table 3.4 Defined Hyperopt Requirments

Requiereement	Defined Values
Objective function	Minimize loss function
Search algorithm	Tree Parzen Estimators
Search space	Refer table 3.4

As discussed earlier, the SVM with different kernels were compared to find the kernel with best performance. When it is found, the parameters to be tuned was

depended on the kernel. For linear and polynomial kernel, only ‘C’ is able to be tuned, whereas RBF is both ‘C’ and ‘gamma’ where the value obtained from search space used logsuniform. The table 3.4 below displays the option of parameter to be tuned based on the best kernel obtained. In simple terms, if linear kernel is considered as best kernel based on its performance, then only hyperparameter tuning on linear kernel is considered where the values in table 3.5 regarding only linear kernel is used.

Table 3.5 Search Space According to Kernel for Hyperopt

Kernel	Parameter	Test Values
SVM	C	0.001 – 10 ⁵
Polynomial		
RBF	gamma	0.001 – 10 ³

```

SET def objective(params, X_train TO word_Smote_vec, y_train TO
sentiment_Smote_vec):

    SET clf TO svm.SVC(**params,random_state=0, kernel TO 'linear')
    SET scores TO cross_val_score(clf, X_train, y_train, cv=5)

    # Extract the best score
    SET best_score TO max(scores)

    # Loss must be minimized
    SET loss TO 1 - best_score

    # Dictionary with information FOR evaluation
    RETURN {'loss': loss, 'params': params, 'status': STATUS_OK}

```

Figure 3.30 Pseudo-code for objective function

```

SET param_grid TO {
    Set search space  }
SET MAX_EVALS TO 50

```

Figure 3.31 Pseudo-code for setting hyperopt search space

```

SET tpe_algorithm TO tpe.suggest
SET bayes_trials TO Trials()

SET best TO fmin(fn TO objective, space TO param_grid, algo TO tpe.suggest,
max_evals TO MAX_EVALS, trials TO bayes_trials)

```

Figure 3.32 Pseudo-code for starting hyperparameter tuning for classifier

```

#just use this and insert best value, no need to run hyperopt everytime

#note kerner and c is based on hyperopt hyperparameter tuning
SET classifierTuned TO classifier with best parameter found

SET word_Smote_vec, sentiment_Smote_vec, word_test_vec , sentiment_test_vec,
vecWord , vecSentiment TO trainingSmoteVectorizerTUNED (fileName)

CALL classifier(word_Smote_vec, sentiment_Smote_vec,classifierTuned)

CALL displayResult(vecWord, vecSentiment, sentiment_test_vec, word_Smote_vec,
word_test_vec,sentiment_Smote_vec,classifierTuned)

```

Figure 3.33 Pseudo-code for modeling with tuned tf-idf and classifier with smote

Finally, when the best parameters were obtained, it was inserted into the parameter of instantiated classifier as shown in Figure 3.33, therefore, the final modelling step for SA is completed.

In comparison to modelling for SA, modelling for CF was much easier. To check stationary by using ADF and KPSS, granger causality, finding lag, and fitting into VAR was all included in python statsmodel package. Python read the .csv file containing the consolidated time series data, transform it into data frame, and performs all the processes that was discussed earlier using the package mentioned.

```

Input:
    timeseires: separate column for the time series
Output: Display if time series is stationary or not

Algorithm:
# AIC aims to minimize corresponding information
SET dfctest TO adfuller(timeseries, autolag='AIC')
SET dfcoutput TO pd.Series(dfctest[0:4], index=['Test Statistic','p-value','#Lags
Used','Number of Observations Used'])
SET pvalue TO dfcoutput['p-value']
# OUTPUT("Pvalue:" , type() "{:f}".format())
OUTPUT (dfcoutput)

IF pvalue < 0.05:
    OUTPUT('Time series is stationary')
ELSE:
    OUTPUT('Time series is NOT stationary')

```

Figure 3.34 Pseudo-code for adf test algorithm

```

Input:
    timeseires: separate column for the time series
Output: Display if time series is stationary or not

```

```

Algorithm:
IF sm.tsa.stattools.kpss(timeseries, regression='ct')[1] > 0.05:
    OUTPUT('Time series is stationary')
ELSE:
    OUTPUT('Time series is NOT stationary')

```

Figure 3.35 Pseudo-code for kpss test algorithm

```

Input:
    timeseries: separate column for the time series
Output: Display if time series is stationary or not

Algorithm:
SET train_df TO dfDiff[:int(0.8*(len(dfDiff)))]
SET test_df TO dfDiff[int(0.8*(len(dfDiff))):]

SET mod TO VAR(train_df)
sorted_order=mod.select_order(maxlags=20)

SET lag TO sorted_order.aic
SET res TO mod.fit(lag)

```

Figure 3.36 Pseudo-code for developing cf algorithm

```

Input:
    data: data frame of time series
    maxlags: number of lags to see does it correlate until that lag
Output: Display p-value to identify if there are correlation

ALGORITHM:
variables=data.columns
SET matrix TO pd.DataFrame(np.zeros((len(variables), len(variables))),
                           columns=variables, index=variables)

FOR col IN matrix.columns:
    FOR row IN matrix.index:
        SET test_result TO grangercausalitytests(data[[row, col]], maxlag=maxlag,
                                                  verbose=False)
        SET p_values TO [round(test_result[i+1][0]['ssr_chi2test'][1],4) FOR i IN
                        range(maxlag)]
        SET min_p_value TO np.min(p_values)
        SET matrix.loc[row, col] TO min_p_value
    SET matrix.columns TO [var + '_x' FOR var IN variables]
    SET matrix.index TO [var + '_y' FOR var IN variables]
OUTPUT(matrix)

```

Figure 3.37 Pseudo-code for granger causality algorithm

```

Input:
    dayForecast: duration in days to forecast
Output: Plot of actual value vs forecasted value

SET forecast_res TO res.forecast(dfDiff.values[-lag:], dayForecast)
SET getLastDate TO test_df.index[len(test_df)-1]
SET getstartDate TO test_df.index[0]

```



```

SET durationDays TO (getLastDate - getstartDate)
SET durationDays TO durationDays.days

#use train_df if you want to see your difference between test and actual result

SET test_res TO res.forecast(train_df.values[-lag:], durationDays)
SET startDate TO test_df.index[0].strftime("%Y-%m-%d")
SET indx TO pd.date_range(startDate, periods=durationDays)

SET forecastTest_df TO pd.DataFrame(test_res, index=indx,
                                   columns=['f_sentscore','f_peakplayer'])

test_vs_predScore=pd.concat([test_df.peakplayer,forecastTest_df.f_peakplayer],axis=1)
test_vs_predScore.plot(figsize=(12,5))

```

Figure 3.38 Pseudo-code for plotting test value vs forecasted value

```

Input
  Invdy: de-difference of time series
Output: MAPE value

Algorithm:
SET train_dfNotdiff TO df[:int(0.8*(len(df)))]
SET test_dfNotdiff TO df[int(0.8*(len(df))):]

invdy=np.cumsum((test_dfNotdiff['peakplayer'][1],test_df['peakplayer'])) # inverse first
differences

SET APE TO []

# Iterate over the list values
FOR day IN range(len(invdy)):
  # Calculate percentage error

  SET per_err TO (test_dfNotdiff['peakplayer'][day+1] - invdy[1][day]) /
    test_dfNotdiff['peakplayer'][day+1]

  SET per_err TO abs(per_err)
  APE.append(per_err)

# Calculate the MAPE
SET MAPE TO sum(APE)/len(APE)

# Print the MAPE value and percentage

OUTPUT(f"
MAPE : { round(MAPE, 2) }
MAPE % : { round(MAPE*100, 2) } %
")

```

Figure 3.39 Pseudo-code for evaluating with MAPE

Lastly, the forecasted results were used to compare between churn forecasting model only and churn forecasting model that utilizes sentiment analysis.

Table 3.6 Summary of software used

Software	Used on	Packages and Modules	Purpose
Python	SA Model & CF Model	xlwt	Saving data into Excel
		pandas	Transform data into data frame for easy manipulation
	SA Model	request	Request from website to scrape data
		nltk	Consist of most preprocessing method such as tokenization
		re	Perform preprocessing method on regular expression such as removing punctuation
		contractions	Fix contraction in text
		sklearn	Main part for sentiment analysis for feature extraction, model selection such as svm, calculating cross validation score, grid search hyperparameter tuning, etc.
		numpy	Transform Y values into array that aligns with X values that was transformed by tfidf
		imblearn	SMOTE and making pipeline
		hyperopt	Hyperparameter tuning using hyperopt
	CF Model	statsmodel	Main part for time series forecasting for ADF, VAR, KPSS, granger causality, etc.

3.8 Testing and Evaluation

Although this research main concern is churn forecasting result, evaluating the performance of sentiment analysis also cannot be overlooked. Sentiment analysis was evaluated by comparing different kernels and when SMOTE is applied, the fined-tuned TF-IDF with fined-tuned SVM and fined-tuned TF-IDF model in terms of each models' performance metrics. The metrics compared were accuracy, weighted precision, weighted recall, weighted F1-score for each classified polarity which are positive and negative, which were all calculated using python using the following formula.

Given the label in this research is binary, the confusion matrix would look as the following Table 3.7

Table 3.7 Sample Confusion Matrix for Binary Label

		Predicted Values	
		Positive	Negative
Actual Values	Positive	True Positive (TP)	False Negative (FN)
	Negative	False Positive (FP)	True Negative (TN)

where,

TP	Number of positive labels classified correctly
TN	Number of negative labels classified correctly
FP	Number of positive labels classified incorrectly
FN	Number of negative labels classified incorrectly

Accuracy was used to the overall accuracy of the model and was calculated using the formula number 3.

$$Accuracy = \frac{TP + TN}{TP + TN + FN + FP} \quad (3)$$

Formula number 4 describes the precision formula. Precision is a metric to see what percentage of the results were truly positive.

$$Precision = \frac{TP}{TP + FP} \quad (4)$$

Recall is the percentage of predicted result (positive and negative) were the same as true positive rate. It was calculated using formula number 5.

$$Recall = \frac{TP}{TP + FN} \quad (4)$$

F1-score is the harmonic measurement of precision and recall by combining them into a single metric calculated using the formula number 5. The metrics that are weighted simply means the average of all class positive and negative combined.

$$F1score = 2 \times \frac{(Precision \times Recall)}{(Precision + Recall)} \quad (5)$$

Meanwhile, this research main concern regarding churn forecast was tested based on Mean Absolute Percentage Error (MAPE). MAPE is a statistical measure to define the accuracy of a time series forecasting. It is considered as a loss function to determine the error termed by the model evaluation. It can be defined by the following formula.

$$M = \frac{1}{n} \sum_{t=1}^n \left| \frac{A_t - F_t}{A_t} \right| \quad (6)$$

where,

A_t	actual value
F_t	forecasted value
n	frequency of summation iteration

Where the absolute difference between actual value, A_t and forecast value, F_t was calculated and a mean function was applied to get the result of MAPE value. Lower MAPE indicates a better model.

Finally with these metrics, the significance of the proposed churn forecasting model was able to be evaluated and identified whether there is a need to incorporate SA into the proposed model or was it sufficient to only use UGC.

3.9 Documentation

The final phase is documentation where procedures are compiled to view and traceback if there are any mistake to be fixed. All findings, results, weakness, coding of proposed model are included. It is a continuous phase since new idea and new problems to be solved were discovered, hence the content might change from time-to-time varying different version of documentation. Nonetheless, it needs to be systematic so that reader will be able to digest information easier.

3.10 Summary on Research Methodology

To conclude, this chapter discusses 8 phases that was proposed to be followed which are preliminary study, knowledge acquisition, data collection, data preprocessing, model design, model development, testing and evaluation, and documentation. By following these phases, the objective can be reached by the end of the semester.

Preliminary study and knowledge are phases that focuses on the beginning where outline and what is required for this proposed project is prepared. This phase solves the first objective with analysis from literature review. Data collection and data preprocessing are phases that was required for input of the proposed model. Second objective can be achieved by following these phases. The final objective can be achieved by following these four phases which are model design, model development, testing and development, and documentation where all of these phases focus on developing the model as well as documenting the results, hence the best model among other models that is tested can be concluded, therefore accomplishing the goal of this proposed research. The final phase was designed and developed with the aid of literature review.

CHAPTER FOUR

RESULTS AND FINDINGS

This chapter shows the dataset obtained, discusses and evaluate about the sentiment analysis model and churn forecasting model developed, and finds whether the model proposed has significance.

4.1 Dataset Review & Preprocessing Result

The number of data collected for sentiment analysis (SA) from 1st January 2020 to 31st December 2020 are displayed in Table 4.1 below.

Table 4.1 Amount of Review and Sentiment for Each Game Scraped

Game Title	Amount of Data	True	False
BloonsTD	26506	25730	776
Oxygen Not Included (ONI)	8460	8197	263
Sid Meier's Civilization VI (SMC)	14870	14326	544
Stellaris	16777	15498	1279

review	sentiment	score	time
Good game. Simple to start, hard to master.	TRUE	0	1609418341
It's not my kind of game but I can't say it's bad	TRUE	0	1609415285
The title is a lie	TRUE	0	1609414390
무조건하세요 ㅅㅂ 벌써 새해임 ?	TRUE	0	1609411099
fun to play but complicated in a good way.	TRUE	0	1609407544
Space economy simulation where management	TRUE	0.6574779	1609405181
Super addictive. I love the layers of complexity t	TRUE	0	1609402840
This game is basically a huge sandbox but the	TRUE	0	1609401861
.	TRUE	0	1609396778
Amazing	TRUE	0	1609396436
this game will keep you coming back too it	TRUE	0	1609395024
GREAT GAME!!! (watch a guide to get started y	TRUE	0	1609394972
.			
.			
.			
So simple yet so complex.	TRUE	0	1577854284
Strangely addictive.	TRUE	0	1577851707
This game is fantastic. In essence it is Sim City	TRUE	0	1577851189
A very interesting game, not gonna lie. Fun with	TRUE	0.4692460	1577845221
this game is either really good or so far up my a	TRUE	0	1577844876
played Klei's Dont Starve and this delivers the s	TRUE	0	1577844252
An overall fun experience	TRUE	0	1577843338
This game is infuriating, but I can't stop playing	TRUE	0	1577842967

Figure 4.1 Sample of review before preprocessing

review	sentiment	score	date
good simple start hard master	TRUE	0.1	2020-12-31
not kind not say bad like simulation building thing	TRUE	0.1	2020-12-31
title lie	TRUE	0.1	2020-12-31
fun complicate good way	TRUE	0.1	2020-12-31
space economy simulation management heat pres	TRUE	0.657478	2020-12-31
super addictive love layer complexity uncover find r	TRUE	0.1	2020-12-31
basically huge sandbox sandbox time bomb order	TRUE	0.1	2020-12-31
keep come back	TRUE	0.1	2020-12-31
.			
.			
.			
simple yet complex	TRUE	0.1	2020-01-01
strangely addictive	TRUE	0.1	2020-01-01
fantastic essence sim city gravity automation elem	TRUE	0.1	2020-01-01
interesting not go lie fun amount stuff not feel tedio	TRUE	0.469246	2020-01-01
either really good far alley love point afraid take clo	TRUE	0.1	2020-01-01
play klei not starve deliver survival feel streamlined	TRUE	0.1	2020-01-01
overall fun experience	TRUE	0.1	2020-01-01
infuriate not stop playing	TRUE	0.1	2020-01-01

Figure 4.2 Sample of review after preprocessing

Note that date was in epoch format, which was previously mentioned in Table 3.2 under Subtopic 3.4. The table also explained about attributes that were scraped. Converting epoch into readable datetime format was not mentioned as a preprocessing step during sentiment analysis process because it was not closely related to the process. However, an additional step of transforming the data into human readable form was done for the sake of readability. Figure 4.1 is almost similar to 3.5, however it was written again for easy comparison with Figure 4.2.

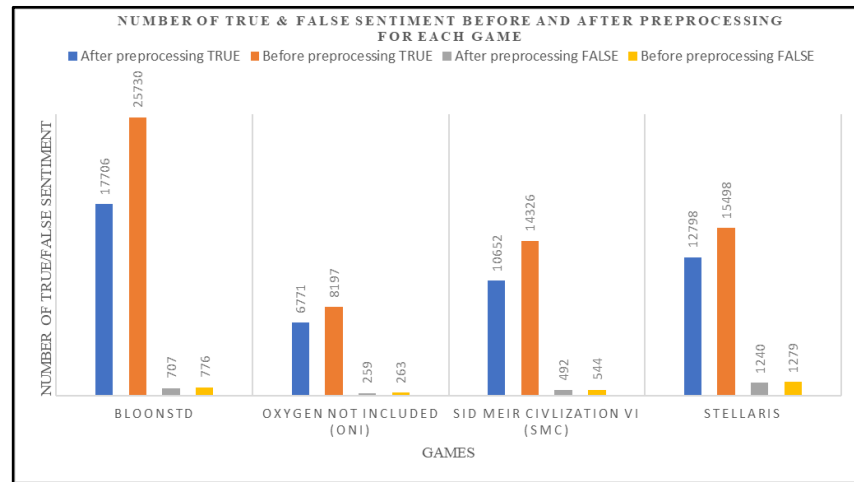
The dataset was successfully preprocessed by following the outlined preprocessing steps in Chapter 3. Table 4.2 below shows the amount of data before and after preprocessing.

Table 4.2 Amount of review for each game before and after preprocessing

Game Title	Before preprocessing	After preprocessing
BloonsTD	26506	18413
Oxygen Not Included (ONI)	8460	7030
Sid Meier's Civilization VI (SMC)	14870	11144
Stellaris	16777	14038

Table 4.3 Amount Of Review And Sentiment For Each Game After Preprocessing

Game Title	Amount of Data	True	False
BloonsTD	18413	17706	707
Oxygen Not Included (ONI)	7030	6771	259
Sid Meier's Civilization VI (SMC)	11144	10652	492
Stellaris	14038	12798	1240

**Figure 4.3** Number of sentiment for each game before and after preprocessing

After selecting and merging the dataset, the total number of review with true and false sentiment consist of 9302 and 2698 respectively totalling to 12000 reviews.

DateTime	Players
01/02/2020 00:00	9103
02/02/2020 00:00	9436
03/02/2020 00:00	7195
04/02/2020 00:00	7349
05/02/2020 00:00	7293
06/02/2020 00:00	7433
07/02/2020 00:00	7814

Figure 4.4 Sample for daily number of player

Next, the dataset for number of daily players are straightforward as it consist of two column based on Figure 4.4. The column “Players” indicates the highest number of players on a date which is indicated by the “DateTime” column. The transformation of differencing does not occur until data is consolidated which is explained in later section. The number of daily players collected (peakplayer) are plotted into a line graph as follows from Figure 4.5 to Figure 4.8.

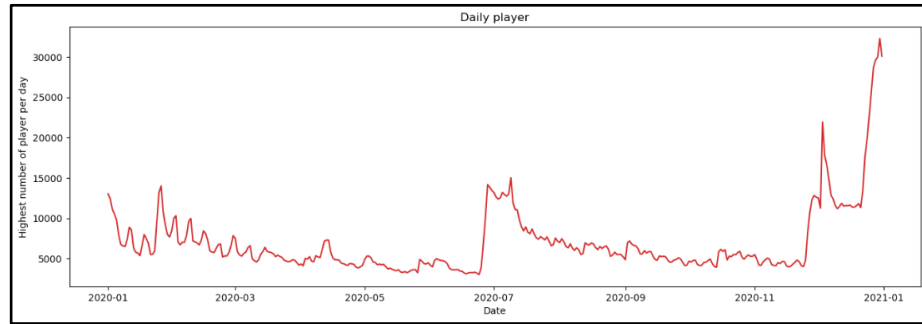


Figure 4.5 Timeseries of daily number of player for BloonsTD

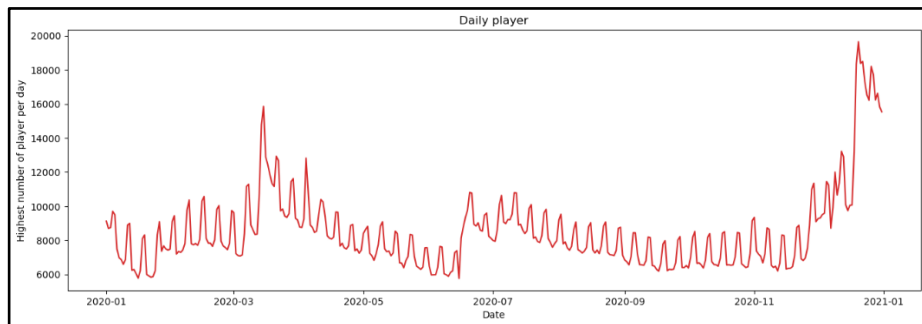


Figure 4.6 Timeseries of daily number of player for ONI

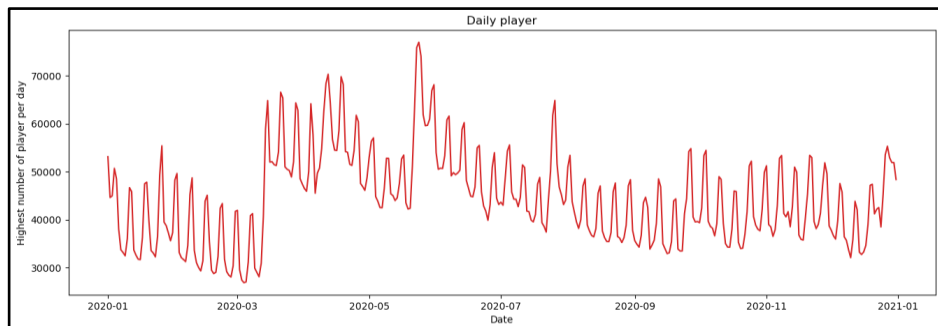


Figure 4.7 Timeseries of daily number of player for SMC

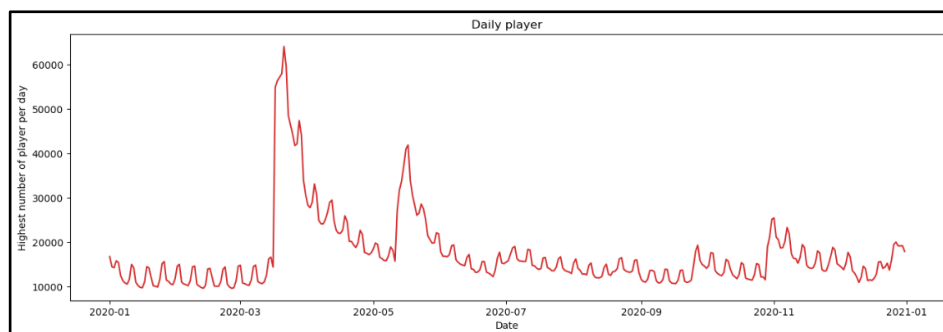


Figure 4.8 Timeseries of daily number of player for Stellaris

4.2 Sentiment Analysis Modelling Results

The sentiment analysis evaluation throughout this section is based on the result of merged dataset that was trained. The results obtained metrics such as accuracy, precision, recall, and F1 score. The values used to compare were in percentages in decimal form, 1 being 100% and 0.1 being 10%.

4.2.1 Performance Based on Different Kernels

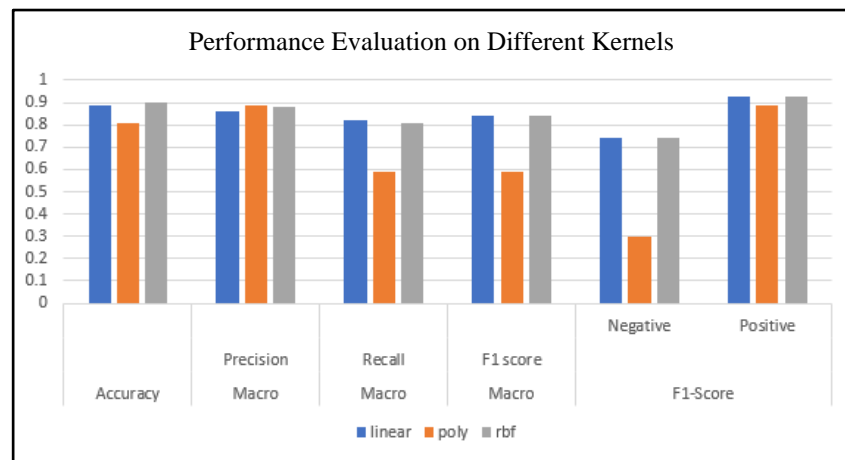


Figure 4.9 Comparison between SVM kernels

Table 4.4 SVM kernel performance result

Kernel	Accuracy	Macro Precision	Macro Recall	Macro F1 score	F1-Score	
					Negative	Positive
linear	0.89	0.86	0.82	0.84	0.74	0.93
poly	0.81	0.89	0.59	0.59	0.30	0.89
RBF	0.90	0.88	0.81	0.84	0.74	0.93

The very first testing conducted were to see the performance of each kernel without applying SMOTE, and with default values for TF-IDF and SVM parameters. Using confusion matrix and its formula, Table 4.4 is obtained. Throughout this research testing and result, this table for evaluating are similar to one another using the same metrics.

Based on Figure 4.9 obtained by mapping out based on Table 4.4, it observed that linear and RBF kernel performs relatively the same and is better in comparison to poly kernel, due to the sentiment being a binary value of “true” and “false.

4.2.2 Performance Based on Different Kernels with SMOTE

SMOTE is then applied to the training data set, where it is then fit into the classifiers with different kernel to see whether oversampling is able to give a better result when tested. Hyperparameter for TF-IDF and SVM are still default.

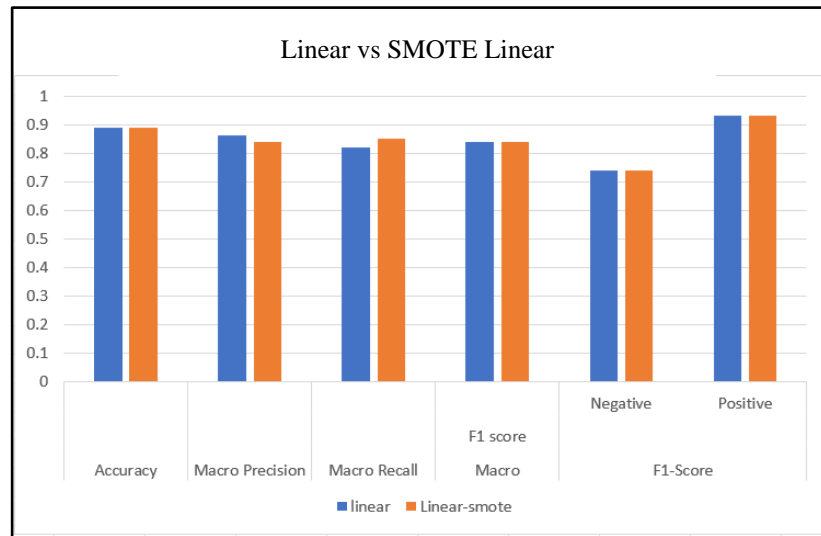


Figure 4.10 Comparison between linear and linear with SMOTE

The results obtained for linear kernel when SMOTE was applied is relatively the same based on Figure 4.10, however, the changes that can be seen is the decrease in precision and increase in recall when SMOTE is applied.

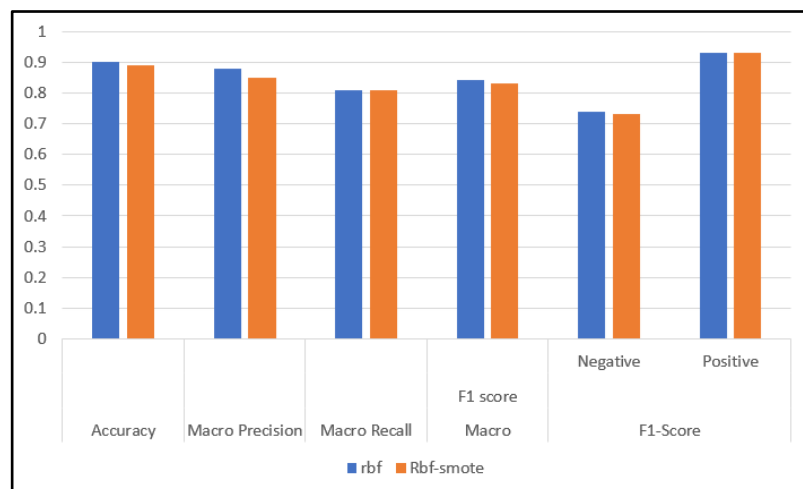


Figure 4.11 Comparison between RBF and RBF with SMOTE

Based on Figure 4.11 for RBF kernel, most of the results is seen to be clearly decreased from using default training data to training data that uses SMOTE. Recall also did not change, defeating the purpose of why SMOTE is applied.

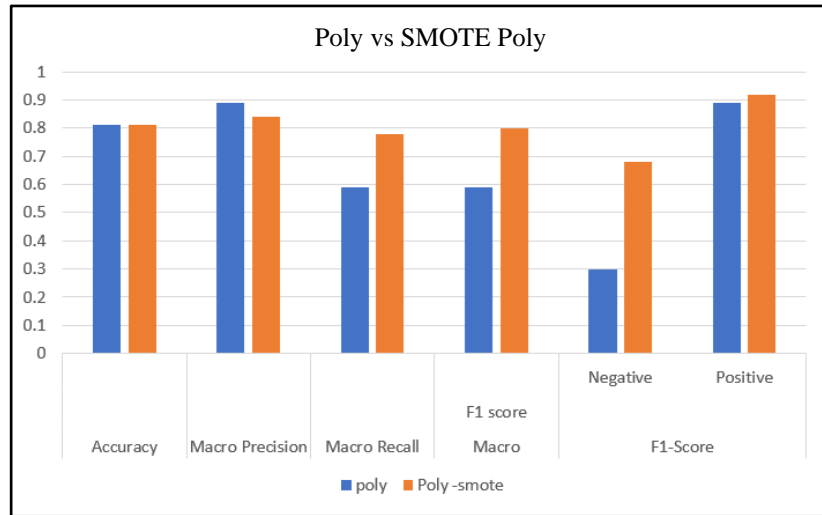


Figure 4.12 Comparison between polynomial and polynomial with SMOTE

On the other hand, polynomial kernel (poly) was able to perform better with SMOTE as displayed on Figure 4.12. Although precision decreases, most of the metrics such as recall, and F1-score increases a lot while maintaining accuracy.

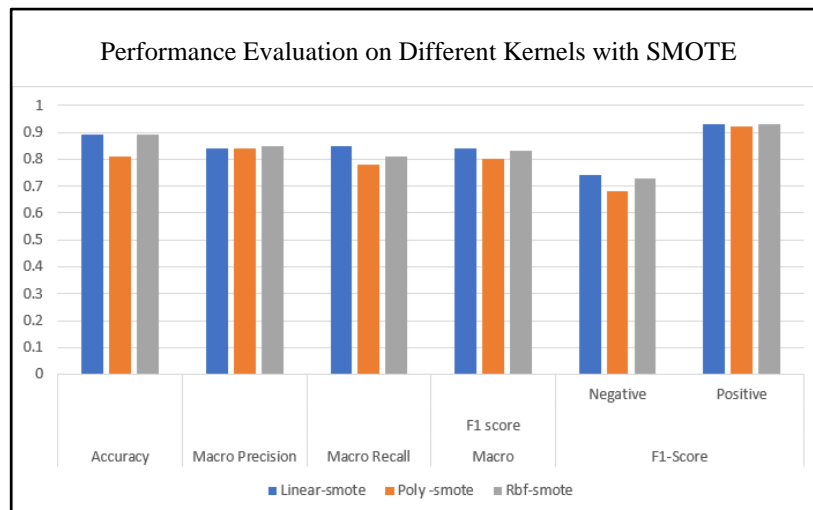


Figure 4.13 Comparison between SVM kernels with SMOTE

Table 4.5 SVM kernel with SMOTE performance result

Kernel	Accuracy	Macro Precision	Macro Recall	Macro F1 score	F1-Score	
					Negative	Positive
linear	0.89	0.84	0.85	0.84	0.74	0.93
poly	0.81	0.84	0.78	0.80	0.68	0.92
RBF	0.89	0.85	0.81	0.83	0.73	0.93

It can be said that SMOTE increases the performance of classifiers for most of the SVM kernel. Hence, it was still considered to continue using SMOTE for linear and poly. Furthermore, a justified comparison was performed by comparing SMOTE results with other SMOTE results.

Table 4.5 compares the result of all kernels that uses SMOTE and Figure 4.13 displays the bar chart based on Table 4.5. Linear and RBF were close competitor as they performed well on accuracy, precision, recall and F1 score. However, RBF's recall was slightly off compared to linear kernel with 0.04 difference while having only 0.01 difference in precision. Meanwhile poly was behind linear and RBF by a decent margin. Therefore, the linear chosen as “best kernel” as discussed in Subtopic 3.6.1 is the linear kernel with SMOTE applied, where this linear was used throughout the model development.

4.2.3 Performance Based on Tuned TF-IDF

After best kernel was identified, grid search hyperparameter tuning was performed while using SVM linear kernel with default parameter. The best TF-IDF parameter obtained according to grid search hyperparameter tuning is displayed in Table 4.6 below.

Table 4.6 Grid Search Result for TF-IDF Parameter

Hyperparameter	Test Values	Best Combination of Parameter
Max_features	2000, 2500, 5000, 10000, 15000	15000
Max_df	0.25, 0.5, 0.75, 1	0.05
Ngram_range	(1,1), (1,2), (1,3)	(1,1)

The performance of tuned TF-IDF with default SVM model is compared against default TF-IDF with default SVM.

Table 4.7 Default TF-IDF vs Tuned TF-IDF

Model	Accuracy	Macro Precision	Macro Recall	Macro F1 score	F1-Score	
					Negative	Positive
Linear + SMOTE	0.89	0.84	0.85	0.84	0.76	0.93
Linear + SMOTE + Tuned TF-IDF	0.89	0.84	0.85	0.84	0.76	0.93

Based on Table 4.7 all metrics we're the same for default TF-IDF against tuned TF-IDF. This is because the default values we're sufficient to deliver a satisfactory performance, and that no other combination of TF-IDF hyperparameter tuning was able to give a better result.

4.2.4 Performance Based on Tuned TF-IDF + Tuned SVM

Hyperparameter tuning using hyperopt for Support Vector Machine (SVM) was conducted to find the best value of the parameter 'C'. Best value of 'C' obtained were 15.576.

This value was then used when defining the SVM, with the value of tuned TF-IDF to create the final model for sentiment analysis with the research hope for it to produce a better result, as theoretically hyperparameter tuning should increase a model's performance. The results after testing the model are displayed in Table 4.8.

Table 4.8 Default TF-IDF vs Tuned TF-IDF

Model	Accuracy	Macro Precision	Macro Recall	Macro F1 score	F1-Score	
					Negative	Positive
Linear	0.89	0.86	0.82	0.84	0.74	0.93
Linear + SMOTE	0.89	0.84	0.85	0.84	0.76	0.93
Linear + SMOTE + Tuned TF-IDF	0.89	0.84	0.85	0.84	0.76	0.93
Linear + SMOTE + Tuned TF-IDF + Tuned SVM	0.88	.081	0.82	0.81	0.71	0.91

Notice that the model without SMOTE is also included in Table 4.8 because a comparison between all model after determining the best classifier is a good

practice before choosing the final model as the SA model.

In comparison with Linear + SMOTE (model from Subtopic 4.2.2) and Linear + SMOTE + Tuned TF-IDF (model from Subtopic 4.2.3), the final model obtained performed worse than previous model whereby none of the metrics increase in value. Although linear without SMOTE obtained relatively good results, a higher recall was preferred because of the unbalanced class as portrayed in Table 4.3.

Therefore, the final model chosen was open to two option which are the Linear + SMOTE or Linear + SMOTE + Tuned TF-IDF. It was too early to determine whether it might have different impact or not. Linear + SMOTE + TF-IDF was chosen as there is no high significance and any option can be chosen.

4.2.5 Using the Developed SA Model

The model chosen (Linear + SMOTE + TF-IDF) was used to predict the game that this research selected to determine its sentiment and to see whether the predicted values will be a use to the churn forecasting model in the later Subtopic.

Table 4.9 Amount of Sentiment for Preprocessed and Predicted Review

Game Title	Preprocessed		Predicted	
	True	False	True	False
BloonsTD	17706	707	16235	2178
ONI	6771	259	6403	627
SMC	10652	492	9910	1234
Stellaris	12798	1240	12532	1506

When using this model the predicted sentiment using the model, as well as “score” and “date” from Figure 4.2 is also saved to be used for the timeseries churn forecasting except for the “sentiment” column since it is not required for calculation purposes. A sample of saved predicted dataset is shown in Figure 4.14 below.

score	sentiment	time
0.1	TRUE	2020-12-31
0.1	FALSE	2020-12-31
0.1	TRUE	2020-12-31
0.1	TRUE	2020-12-31
0.657478	TRUE	2020-12-31
0.1	TRUE	2020-12-31
0.1	TRUE	2020-12-31
0.1	TRUE	2020-12-31
0.1	TRUE	2020-12-31
0.462707	TRUE	2020-12-31
0.1	TRUE	2020-12-31
0.1	TRUE	2020-12-31
0.1	TRUE	2020-12-31
0.1	TRUE	2020-12-31
0.1	TRUE	2020-12-31
0.1	TRUE	2020-12-31
0.1	TRUE	2020-12-31
0.47619	TRUE	2020-12-31
0.1	TRUE	2020-12-31
0.1	TRUE	2020-12-31
0.1	TRUE	2020-12-31
0.1	TRUE	2020-12-31
0.1	TRUE	2020-12-30
0.488764	TRUE	2020-12-30
0.1	TRUE	2020-12-30
0.1	TRUE	2020-12-30
0.1	FALSE	2020-12-30

Figure 4.14 Sample predicted dataset

The predicted sentiment values were used to calculate the daily sentiment score as the guideline proposed in Chapter 3.

4.3 Churn Forecast Modelling Results

4.3.1 Timeseries Consolidation

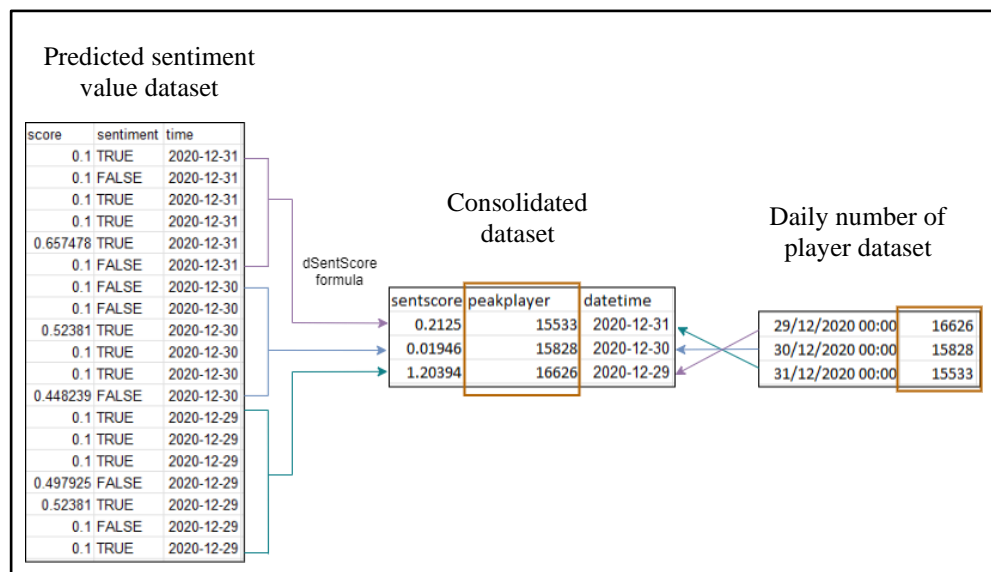


Figure 4.15 Calculating sentscore and consolidating

Figure 4.15 visually show how the process of consolidation occur by calculating daily sentiment score using formula number 1, $dSentScore$ in Subtopic 3.6, and combining with daily number of player dataset which was performed on every game to transform into timeseries manner. A sample of consolidated dataset was shown in Figure 3.13.

A total of four timeseries was consolidated, which were the four games selected (BloonsTD, ONI, SMC, Stellaris) and displayed in Figure 4.16 until Figure 4.19.

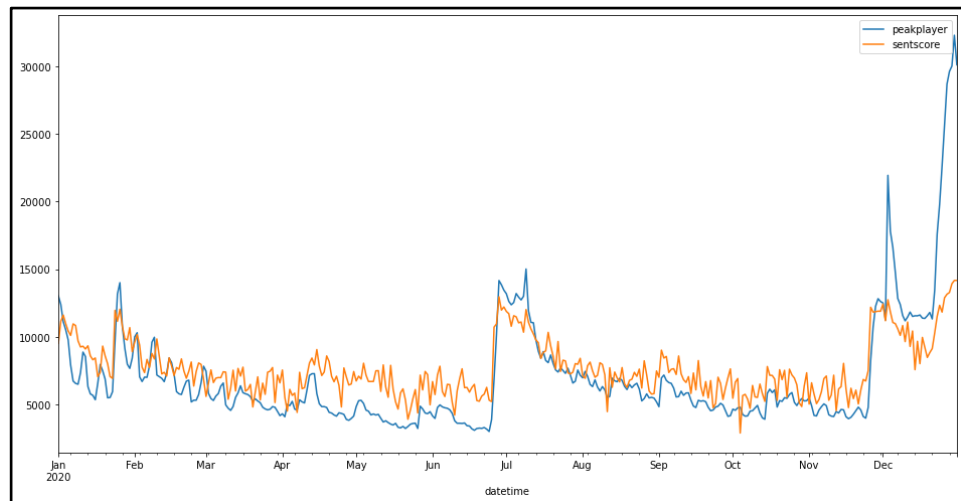


Figure 4.16 Timeseries for consolidated BloonsTD

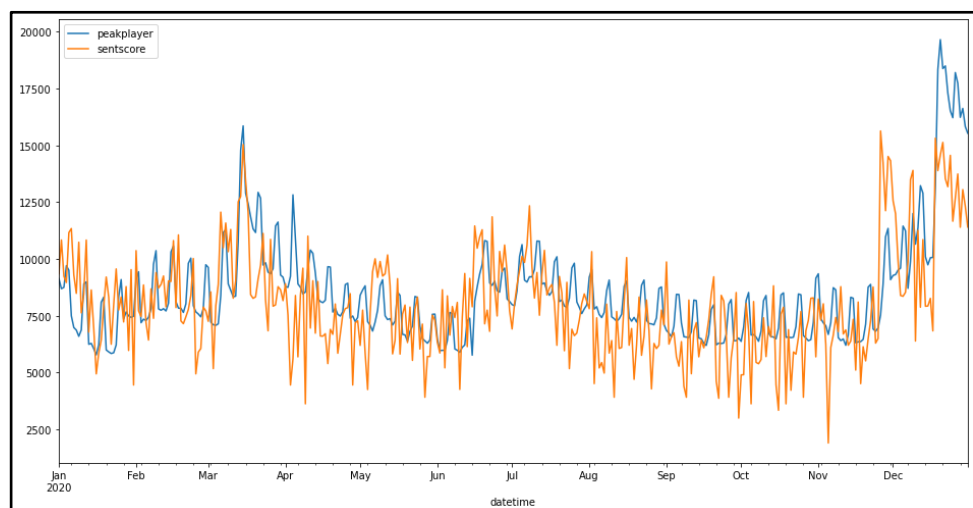


Figure 4.17 Timeseries for consolidated ONI

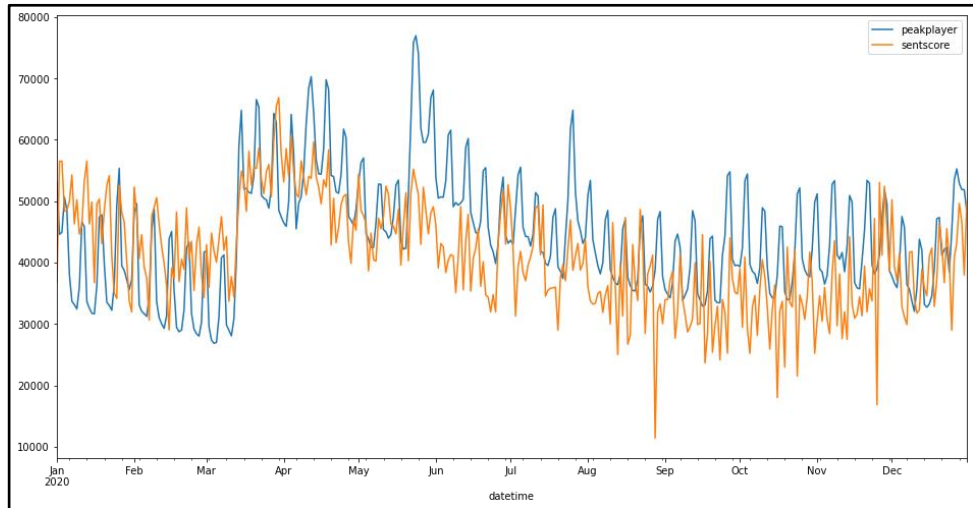


Figure 4.18 Timeseries for consolidated SMC

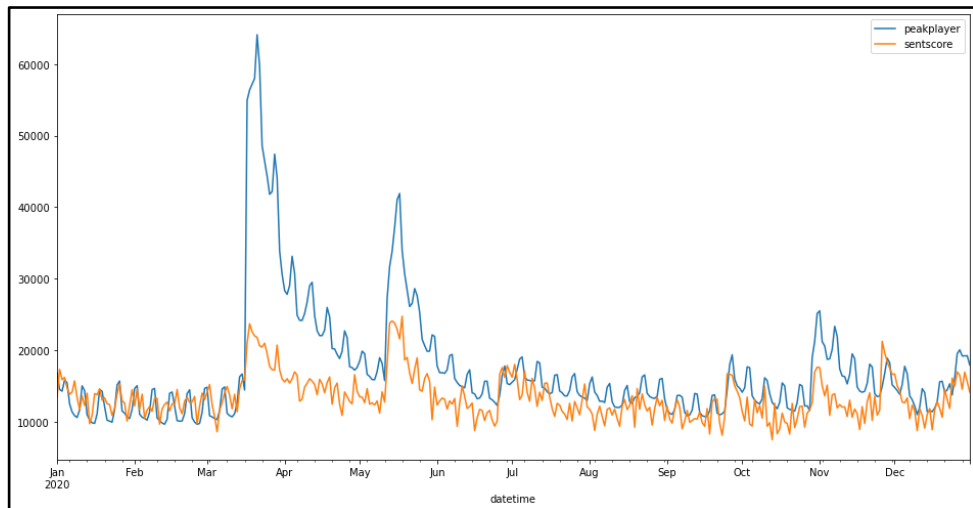


Figure 4.19 Timeseries for consolidated Stellaris

Table 4.10 Legend explanation on timeseries

Legend	Explanation
peakplayer	Daily number of player obtained from SteamDB
sentscore	The calculated daily sentiment score using formula number 1, $dSentScore$

Based on Figure 4.16 until Figure 4.19, both sentscore and peakplayer timeseries consolidated has similar shape and shows correlation for all games selected.

4.3.2 Stationary

Before checking for Granger Causality, ADF test and KPSS was used as a method to determine whether a timeseries is stationary or not as discussed in this research. Based on ADF test, if p-value is more than 0.05, then it is not stationary, vice versa to KPSS if p-value is less than 0.05, then it is not stationary. Both timeseries was checked, if one was not stationary, then both was differenced for each game.

Table 4.11 ADF test & KPSS pvalue

Game Title	p-value			
	ADF (p-value < 0.05)		KPSS (p-value > 0.05)	
	peakplayer	sentscore	peakplayer	sentscore
BloonsTD	0.9062	0.1254	0.0172	0.0368
ONI	0.9417	0.000019	0.02362	0.01730
SMC	0.219566	0.355038	0.01	0.027835
Stellaris	0.020934	0.00011	0.0692199	0.0909717

Table 4.12 ADF test & KPSS after first differencing

Game Title	p-value			
	ADF (p-value < 0.05)		KPSS (p-value > 0.05)	
	peakplayer	sentscore	peakplayer	sentscore
BloonsTD	4.710	6.5459	0.1	0.1
ONI	0.000009	6.9665e-17	0.1	0.1
SMC	2.088254e-07	1.390893e-16	0.1	0.1

Based on Table 4.11, BloonsTD and SMC timeseries (ts) for sentscore and peakplayer were not stationary because of the p-value from ADF test and KPSS obtained were more than 0.05 and less then 0.05 respectively. Therefore, a difference transformation was performed on them. As for ONI, only the sentscore based on ADF test were stationary. Nonetheless, the rest were not, hence differencing was still performed. Finally, Stellaris passed all the tests for stationary therefore differencing was not required and is not included in Table 4.12.

Table 4.12 shows the results of ADF test and KPSS test after differencing BloonsTD, ONI and SMC for the first time in which it shows that all of the games have passed. Hence there were no need to difference any ts no longer. Figure 4.20 until 4.22 shows the graph of daily differenced timeseries.

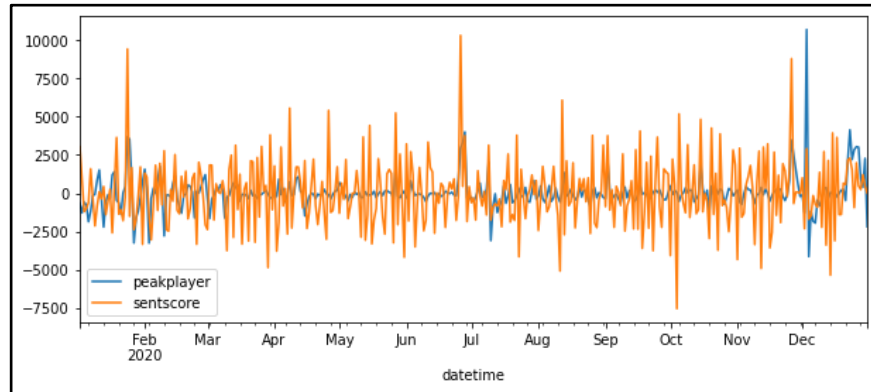


Figure 4.20 Differenced timeseries for BloonsTD

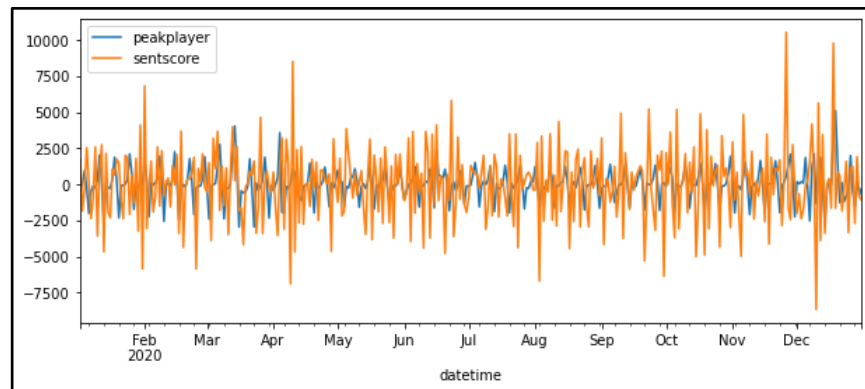


Figure 4.21 Differenced timeseries for ONI

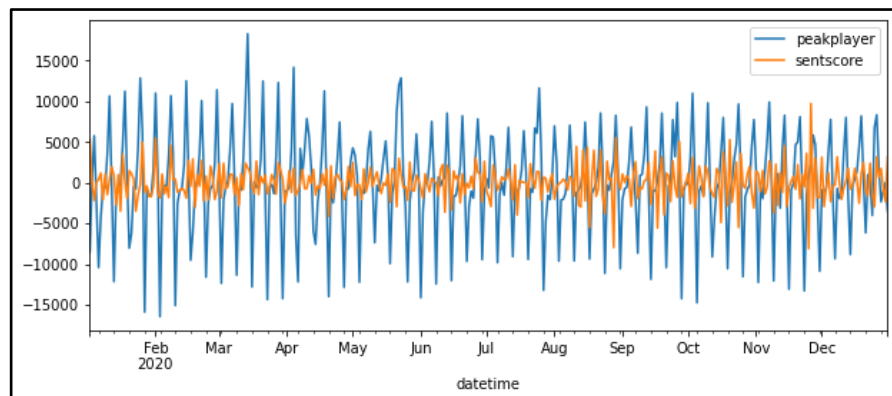


Figure 4.22 Differenced timeseries for SMC

4.3.3 GC results

Granger Causality (GC) was then performed to see if there were strong correlation between peakplayer and sentscore timeseries for all games.

Table 4.13 Granger causality output for each game

Game Title		sentscore_x	peakplayer_x
BloonsTD	sentscore_y	1.0	0.0
	peakplayer_y	0.012	1.0
ONI	sentscore_y	1.0	0.0
	peakplayer_y	0.0	1.0
SMC	sentscore_y	1.0	0.0
	peakplayer_y	0.0014	1.0
Stellaris	sentscore_y	1.0	0.0
	peakplayer_y	0.0216	1.0

Granger causality operates based on the p-value obtained, which are the values in Table 4.13. sentscore_x is the indicator whether it will likely be helpful in predicting peakplayer_y, and whether peakplayer_x will be helpful in predicting sentscore_y. The value of sentscore_x for all games were less than 0.05. Given the significance level is 0.05, it was concluded that sentscore will be helpful to assist in forecasting peaklayer to determine churn.

4.3.4 AIC results

Akaike Information Criterion (AIC) was then determined based on the training data fed into VAR with maximum lag of 20 after training and testing split. The lowest value of AIC indicated the most suitable lag which was fitted into VAR. The following Table 4.14 shows the result of finding AIC for each game.

Table 4.14 Lag AIC for Each Game

Lag	BloonsTD	ONI	SMC	Stellaris
0	28.79	29.50	32.54	33.55
1	28.49	29.20	32.29	31.19
2	28.32	28.94	31.99	31.14
3	28.25	28.90	31.97	31.15
4	28.22	28.78	31.75	31.16
5	28.19	28.34	31.43	31.19
6	28.19	28.32	31.45	31.19

Table 4.14 Lag AIC for Each Game (continued)

Lag	BloonsTD	ONI	SMC	Stellaris
7	28.06	27.57	30.80	31.21
8	28.03	27.56	30.81	31.13*
9	28.01	27.53	30.75	31.15
10	28.03	27.53	30.68	31.17
11	28.06	27.55	30.68	31.20
12	28.04	27.55	30.70	31.21
13	28.02	27.55	30.72	31.20
14	27.99*	27.46*	30.63	31.22
15	27.99	27.47	30.64	31.22
16	28.00	27.47	30.61	31.26
17	28.01	27.49	30.60*	31.28
18	28.01	27.51	30.61	31.31
19	28.02	27.54	30.64	31.31
20	28.05	27.56	30.66	33.55

The result in Table 4.14 shows that each game have different lag values suitable to forecast their timeseries. This is due to each game having different form of timeseries (peakplayer and sentscore) and their different amount of playerbase, as well as how their playerbase left the review, effecting the sentscore that could've affected the AIC.

4.4 Performance on Proposed Churn Forecasting Model

There was a total of four churn forecasting model, one for each game based on the lag found using AIC that was fitted into VAR.

The results of forecasted data on the testing set were obtained. Before evaluation, dataset that was differenced were required to be un-difference to change it into its original form, otherwise false result will be given.

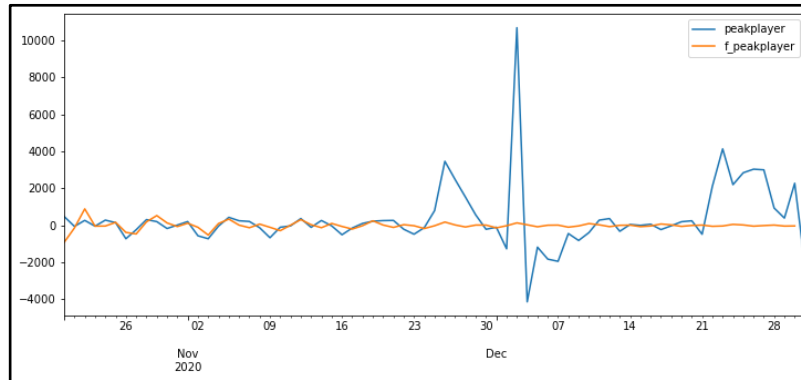


Figure 4.23 Actual vs forecasted result for BloonsTD

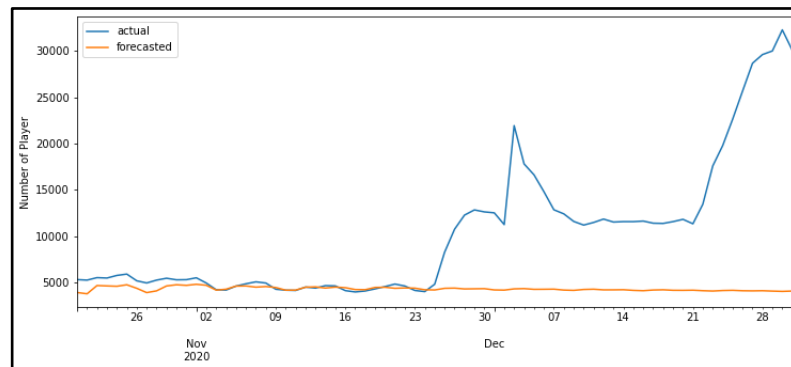


Figure 4.24 Undifferenced actual vs forecasted result for BloonsTD

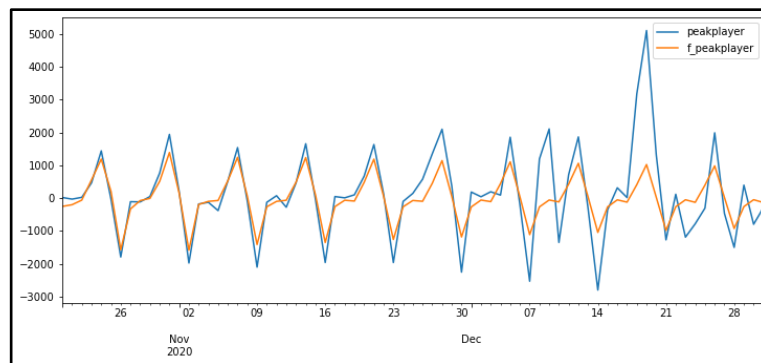


Figure 4.25 Actual vs forecasted result for ONI

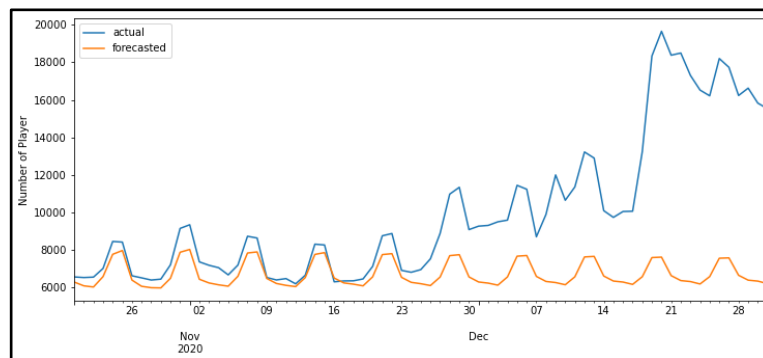


Figure 4.26 Undifferenced actual vs forecasted result for ONI

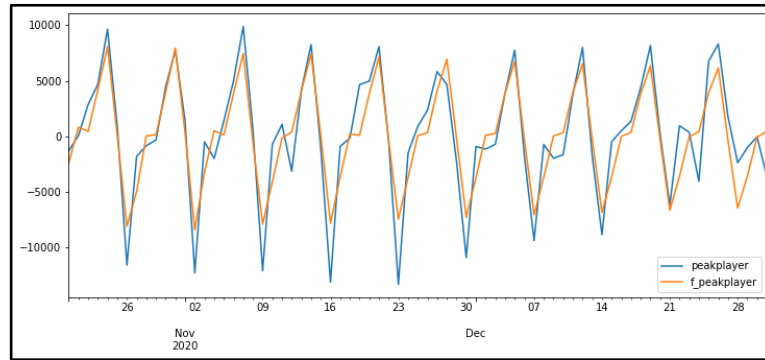


Figure 4.27 Actual vs forecasted result for SMC

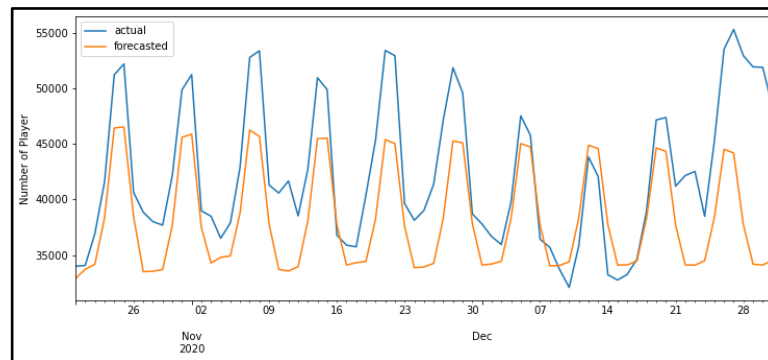


Figure 4.28 Undifferenced actual vs forecasted result for SMC

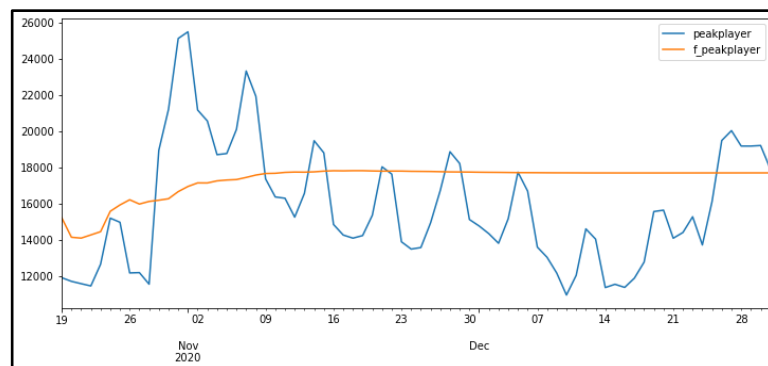


Figure 4.29 Actual vs forecasted result for Stellaris

Figure 4.23 until 4.29 shows the result of forecasted data for each game. Note that “peakplayer” means actual values, whereas “f_peakplayer” means the forecasted value. Stellaris have no figure to display the undifferenced data because it was not differenced in the first place, hence a figure to display undifferenced result was not required.

The performance is evaluated using Mean Absolute Percentage Error (MAPE). A MAPE less than 25% indicates an acceptable model (Swanson, 2015).

Table 4.15 MAPE output for each game

Game Title	Error (%)
BloonsTD	39.3
ONI	25.49
SMC	10.37
Stellaris	20.81

The figures about results of forecasted number of players indicates that certain models were able to perform better than the rest. A timeseries that had less of consistent values were harder to forecast, such as BloonsTD, ONI, and Stellaris. Figure 4.23 shows that BloonsTD had a major spike around 30th November increasing the number of player ten times the usual value whereas ONI had a minor spide around 21st December in Figure 4.25. Unlike differenced data that BloonsTD, ONI, and SMC had which made the timeseries appeared to be consistent, Figure 4.29 shows Stellaris were harder to forecast because of the inconsistency. SMC performed the best out of four model as Figure 4.27 displays the timeseries to be consistent.

As a result, MAPE was obtained using formula number 6. Because BloonsTD had the most difference between forecasted value and actual value in Figure 4.24, a MAPE of 39.3% calculated indicates that BloonsTD model is unacceptable as a forecasting model. Similar to ONI and Stellaris, a MAPE with value of 25.49% and 20.81% was calculated where ONI forecasting model was barely acceptable whereas Stellaris is an acceptable model according to MAPE yet Figure 4.29 shows a major difference between forecasted and actual value therefore it cannot be said that it is an acceptable model. ONI forecasting model performed the best as Figure 4.28 shows minor differences between forecasted result and actual value, and MAPE of 10.37% was calculated making ONI a good forecasting model.

It can be seen that VAR were unable to forecast if the value of timeseries goes out of their normal range. This is a major problem because the purpose of forecasting model is to be able to forecast drop or increase in number of players. As of now, the model this research proposed is only able to forecast churn if the number of players does not have sudden spike or drop in number

of player or an oddly shaped timeseries. For example, based on Figure 4.28 in terms of analysing what the churn will be, it is observed that players will drop from around 40,000 to somewhere below 35,000 on 26 November 2020.

Despite that, further comparisons were conducted to see if this model is better in its current standing or worse in comparisons.

4.5 Evaluating Significance on Proposed CF Model

To determine if there is major significance in utilizing sentiment analysis model for churn forecasting, the current results of churn forecasting model for each game (CF x SA) is compared with the churn forecasting model that does not use the sentiment analysis model (CF / SA). This does not mean that no sentiment was used. The sentiment data that was scraped from Steam was still used, but there was no preprocessing including changing score 0 to 0.1 on it was performed neither a sentiment analysis model was modelled, nor the data was fed into the sentiment analysis model in Subtopic 4.2 was created. From this point and out, this research will refer “CF x SA” as the CF model that utilized the SA model developed in Subtopic 4.2, whereas “CF / SA” as is referring to the CF model that does not utilizes the SA model developed.

However, the step to develop the model CF / SA still had undergone the steps mentioned in Subtopic 4.4.1, 4.4.2, 4.4.3 and 4.4.4 which are calculating daily sentiment score and consolidation of dataset, check for stationary, granger causality, and finding optimal lag with AIC respectively, henceforth creating a new dataset that for CF / SA.

Table 4.16 Number of Sentiment Used for Model Comparison

Game Title	Model	True	False	Total
BloonsTD	CF x SA	16235	2178	18413
	CF / SA	5416	553	5969
ONI	CF x SA	6403	627	7030
	CF / SA	1459	231	1690
SMC	CF x SA	9910	1234	11144
	CF / SA	1597	326	1923
Stellaris	CF x SA	12532	1506	14038
	CF / SA	3258	558	3816

To clarify on Table 4.16, sentiment values for CF x SA used the preprocessed dataset whereas CF / SA uses the dataset before preprocessing which results in different amount of data. In addition, the difference in sentiment data was also because of the amount true and false sentiment that were considered during calculating sentiment score. Since CF x SA changed the score of 0 to 0.1, all true and false values after preprocessing were considered. As for CF / SA, not all values were considered from the total number of true and false values from the data scraped because some reviews had a score of 0.

4.5.1 BloonsTD6

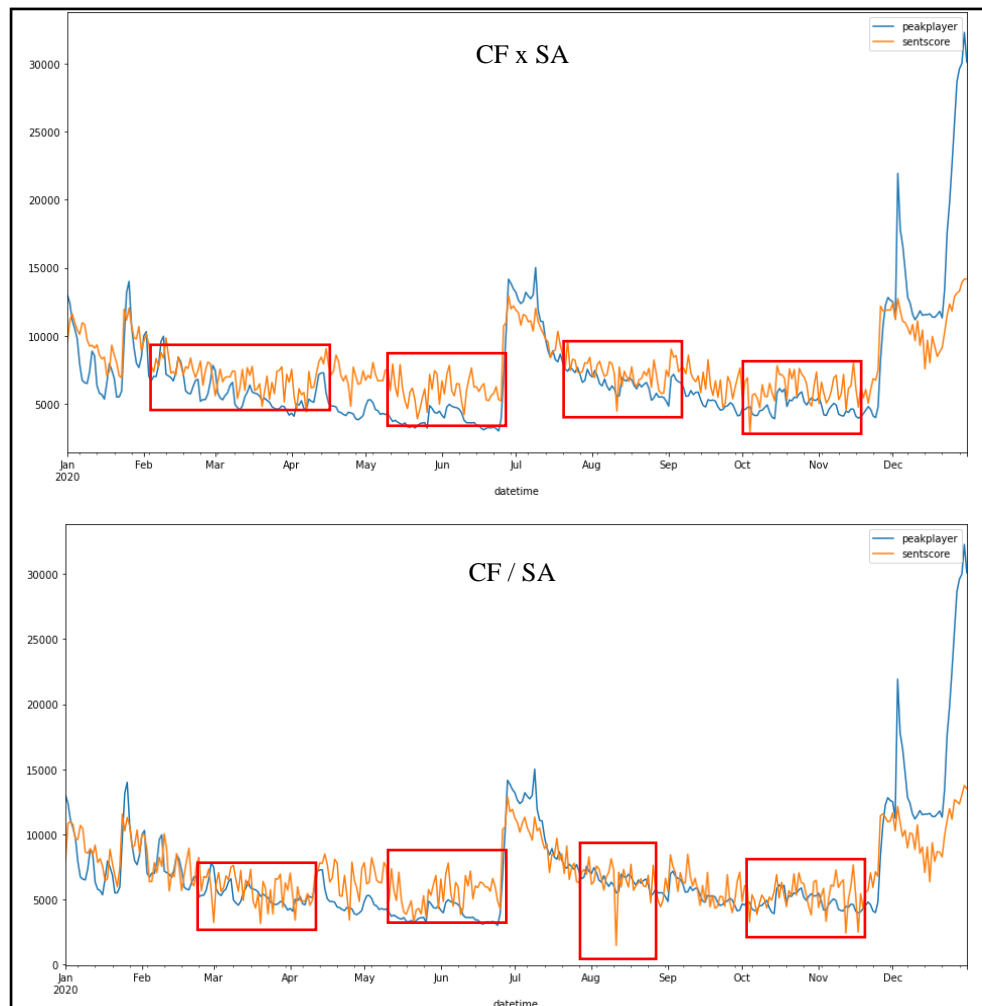


Figure 4.30 Timeseries comparison for BloonsTD

Figure 4.30 above shows the difference between the consolidated timeseries of CP x SA and CP / SA for BloonsTD. It is observed that that timeseries sentscore is much bigger and wider in range at certain locations for CP / SA

compared to CP x SA. Despite the amount of sentiment from Table 4.16 for BloonsTD is different, it still managed to have some form of similarity even without using the SA model.

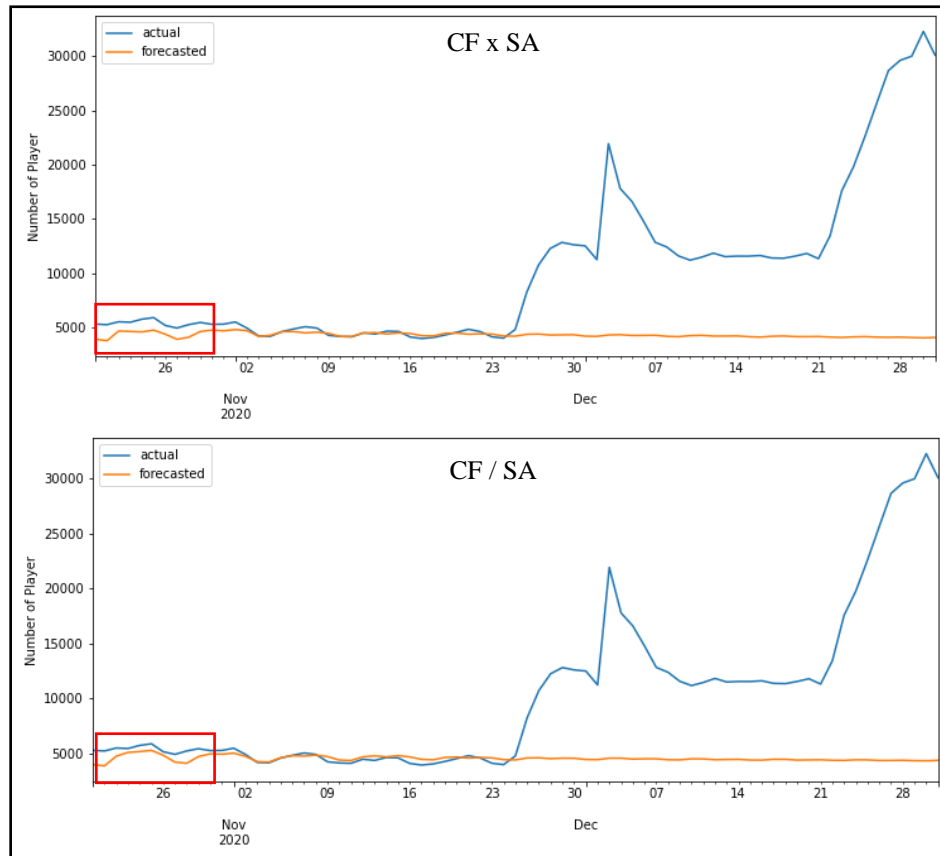


Figure 4.31 Model comparison of forecasted value for BloonsTD

Table 4.17 Model Comparison Metrics for BloonsTD

Metric	CF / SA	CF x SA
Granger (sentscore_x with peakplayer_y)	0.0012	0.012
MAPE (%)	37.81	39.3

Figure 4.31 shows the result of CF x SA and CF / SA forecasted values. The noticeable difference is the value predicted in the earlier date where the forecasted value for CF / SA were closer to its actual value in comparison to CF x SA. Based on Table 4.17, MAPE obtained for CF / SA is lower than CF x SA, with the support of granger obtain of 0.0012 lesser than 0.012, which shows that it is a better model than CF x SA. However, both models are still unacceptable in terms of evaluation.

4.5.2 Oxygen Not Included

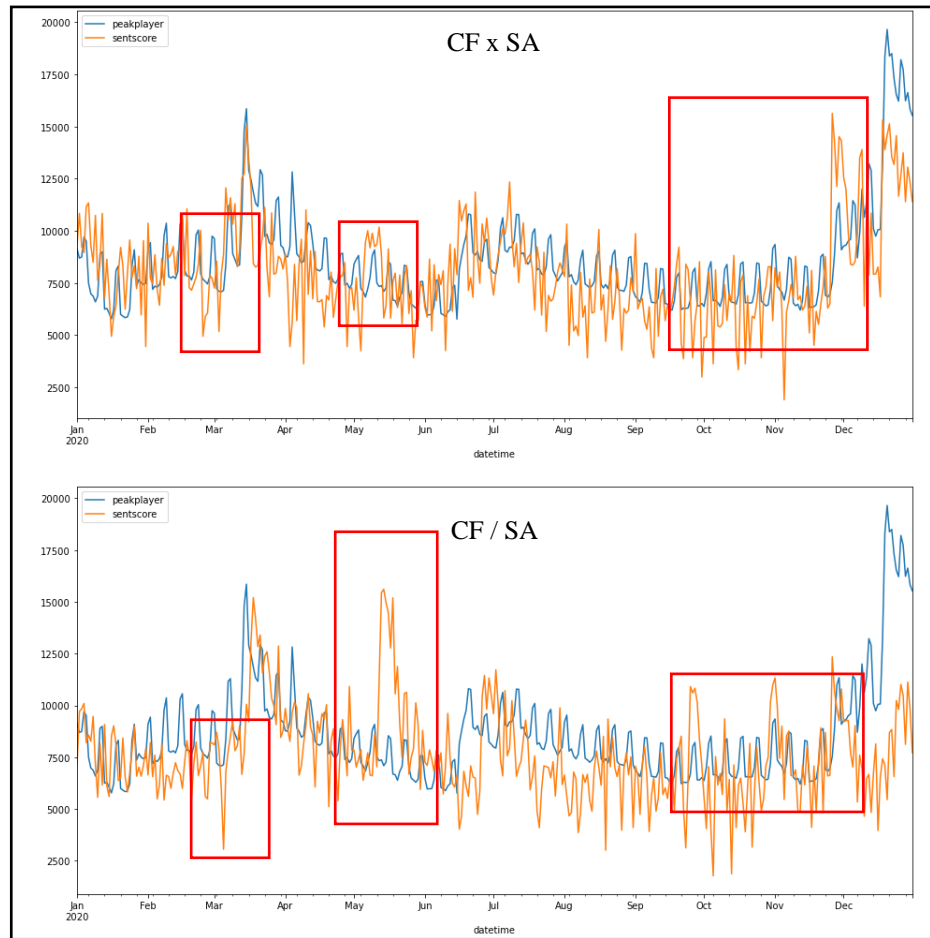


Figure 4.32 Timeseries comparison for ONI

Figure 4.31 above shows the difference between the consolidated timeseries of $CP \times SA$ and CP / SA for ONI. While some parts remained had a steady distribution, there were major differences on certain locations for sentscore with the highest spike difference four times the size for CF / SA compared to $CF \times SA$ that had a much more controlled sentscore timeseries.

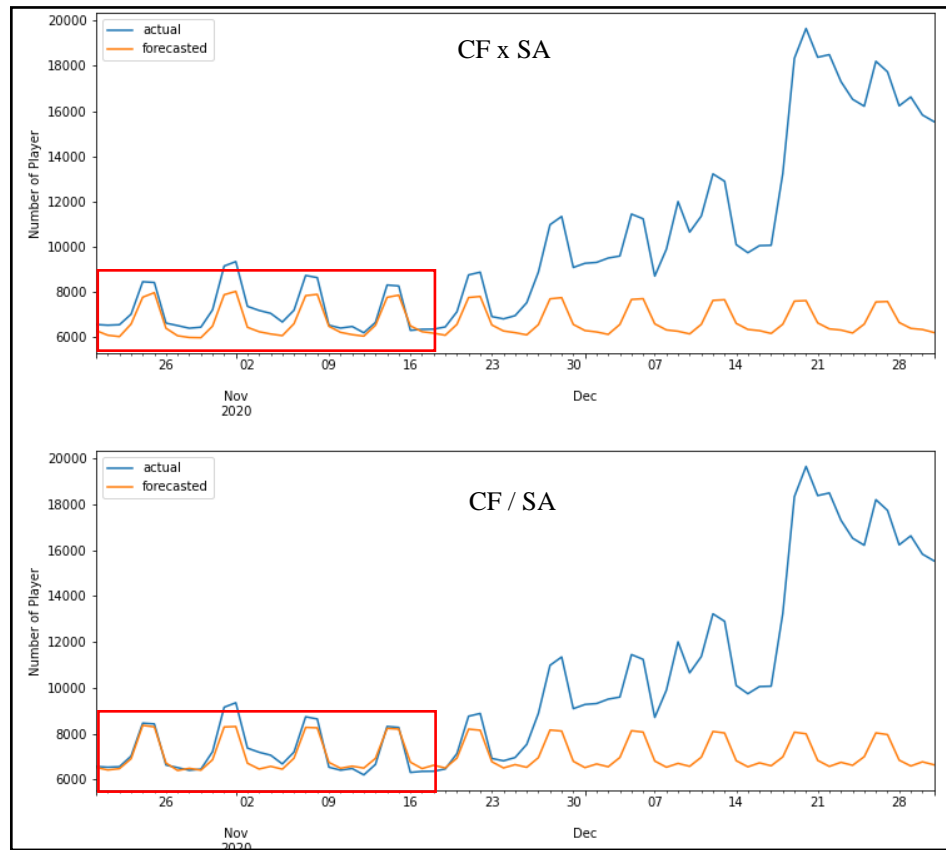


Figure 4.33 Model comparison of forecasted value for ONI

Table 4.18 Model Comparison Metrics for ONI

Metric	CF / SA	CF x SA
Granger (sentscore_x with peakplayer_y)	0.0597	0.0
MAPE (%)	22.28	25.49

Figure 4.33 also shows a similar improvement in the earlier forecasting result. However, despite granger obtaining lower than CF / SA for CF x SA of 0.0, it still had a higher MAPE of 25.49 in comparison to CF / SA with 22.28, making it an acceptable model.

4.5.3 Sid Meier's Civilization VI

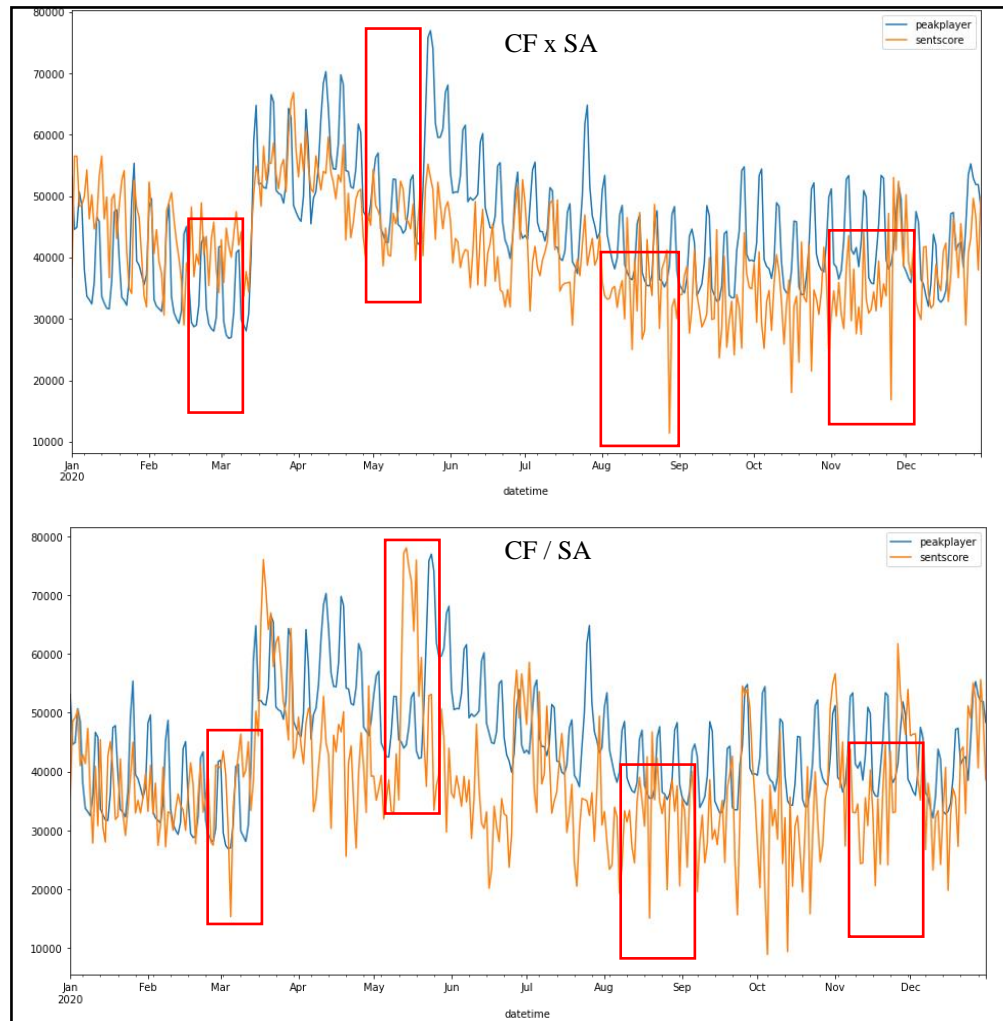


Figure 4.34 Timeseries comparison for SMC

Figure 4.34 above shows the difference between the consolidated timeseries of $CP \times SA$ and CP / SA for SMC. It is observed that there are major differences on certain locations with increasing and decreasing spike. This was due to the amount of sentiment with score was not removed, and as a result the timeseries for CP / SA had less similarity between peakplayer and sentscore.

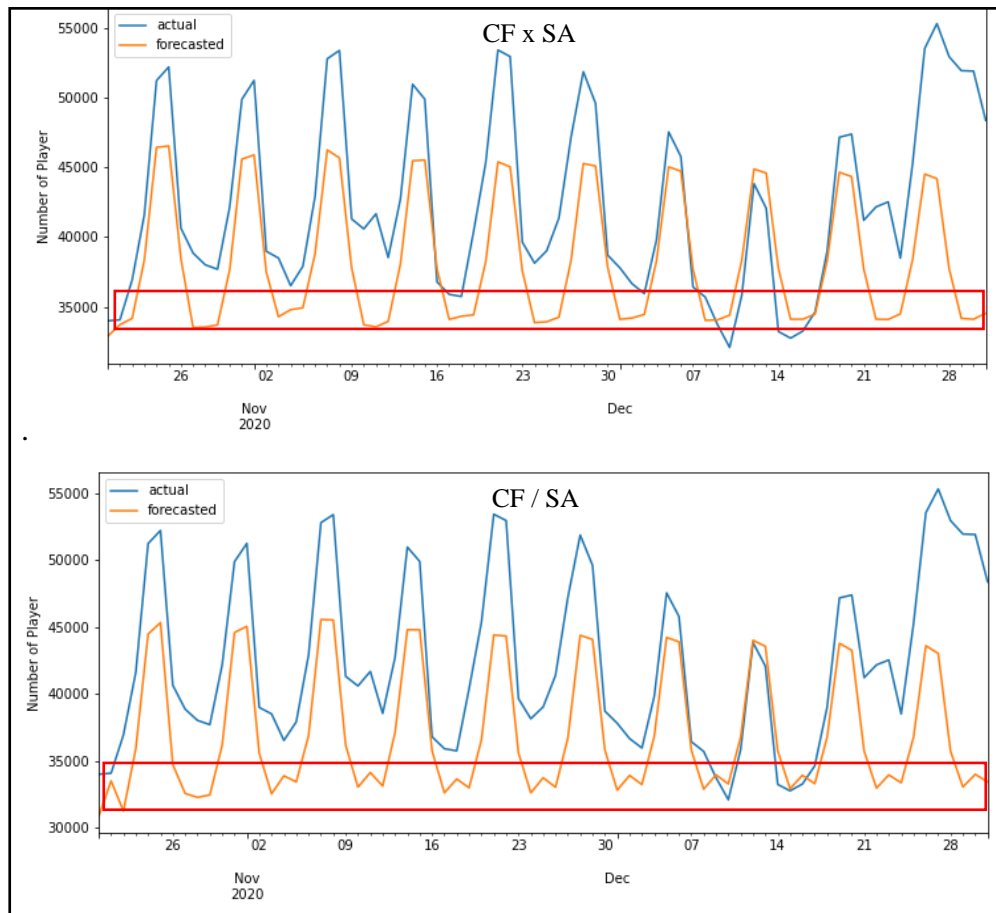


Figure 4.35 Model comparison of forecasted value for SMC

Table 4.19 Model Comparison Metrics for SMC

Metric	CF / SA	CF x SA
Granger (sentscore_x with peakplayer_y)	0.113	0.0014
MAPE (%)	12.55	10.37

Table 4.19 shows that with granger of 0.0014 for CF x SA, a MAPE of 10.37% was produced. Whereas CF / SA resulted in MAPE of 12.55% even with granger that is more than 0.05. Both models are good forecasting model according to MAPE, however Figure 4.35 suggested they are not that good of a model because they were unable to forecast if there is slight change in peaks. The difference between CF x SA and CF / SA are very minor but occurred throughout the test set. This maybe because of the bigger spikes in Figure 4.35 that gave a bigger range when training the VAR model.

4.5.4 Stellaris

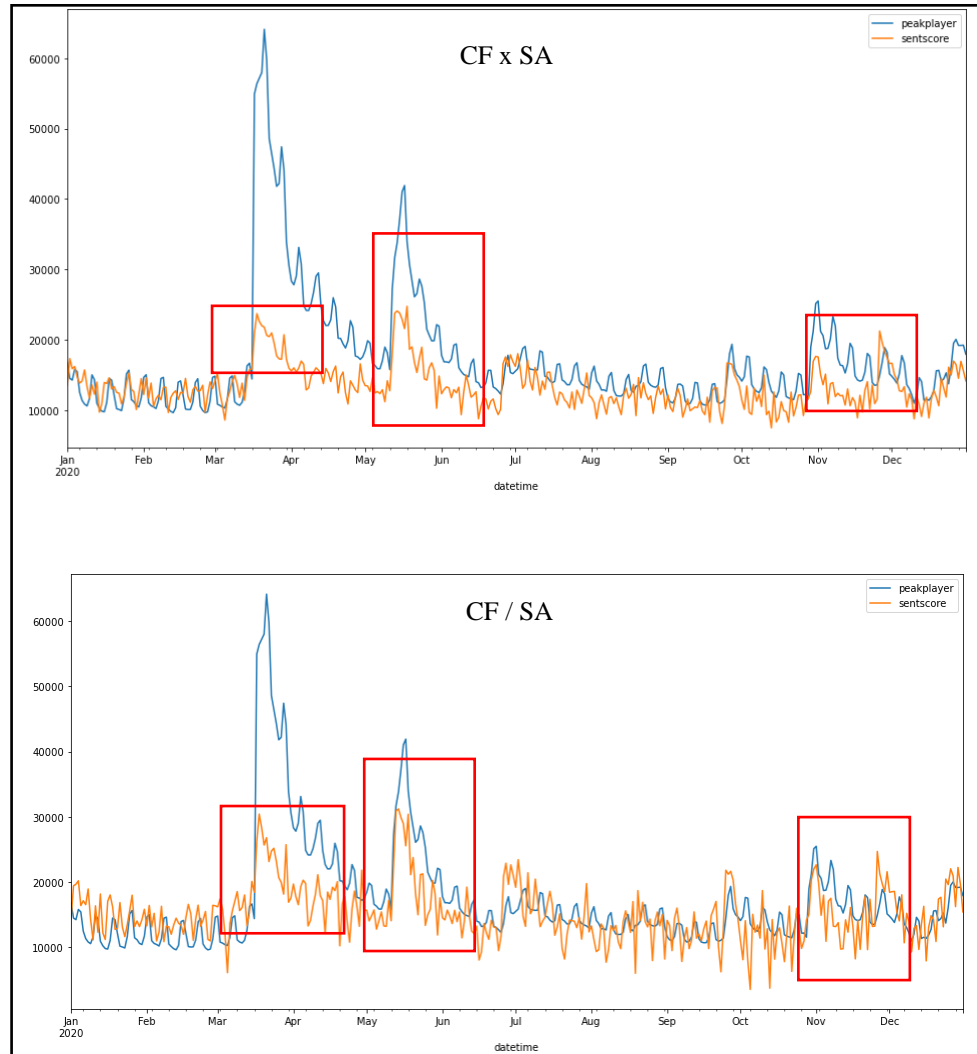


Figure 4.36 Timeseries comparison for Stellaris

As for Stellaris, Figure 4.36 shows both CP / SA and CP x SA has a similar shape for peakplayer and sentscore without major differences, and less spiking in sentscore. In addition, certain parts become closer between peakplayer and sentscore for CP/SA.

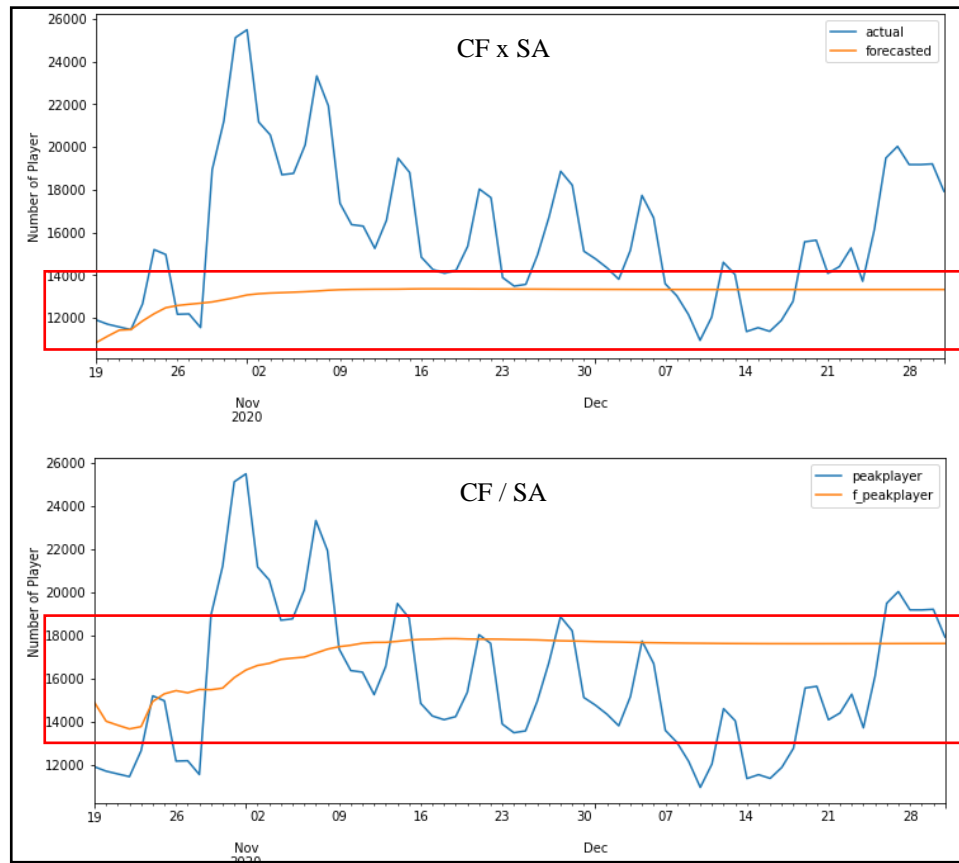


Figure 4.37 Model comparison of forecasted value for Stellaris

Table 4.20 Model Comparison Metrics for Stellaris

Metric	CF / SA	CF x SA
Granger (sentscore_x with peakplayer_y)	0.018	0.0216
MAPE (%)	20.46	20.81

The result obtained for Stellaris displayed in Table 4.20 shows very minor change in MAPE, with only 0.36 differences in improvement towards CF / SA. Each model is considered an acceptable model according to MAPE as both values obtained were below 25%. The value forecasted for CF / SA obtained is slightly higher compared to value forecasted for CF x SA, yet they model maintained a similar shape of forecasted value. CF/SA obtained a better result because of the spike explained in Figure 4.36

4.6 Summary of Results and Findings

To summarize, this chapter showed and discussed about the scraped reviews from Steam and number of daily players from SteamDB, number sentiments

before and after preprocessing and what the timeseries for daily player looked like. The results of sentiment analysis and churn forecasting were also discussed and evaluated.

Sentiment analysis modelling was performed in a level like manner until the model was derived. Multiple kernels were compared with and without SMOTE and found that Linear + SMOTE was the best. It was then applied with TF-IDF GridSearch where the best parameter found were 15000, 0.05, and (1,1) for max feature, max df, and ngram range respectively where this value were also applied on the final step which were SVM hyperopt. A value of 15.576 for the parameter C was found. The three models from each level were compared and the model with tuned TF-IDF was chosen as the predictive model with an accuracy of 89%, precision of 84%, recall of 85% and F1 score of 84%. The model was used to create a dataset for the CF model that utilizes the SA model.

Churn forecasting used the predicted result from the SA model created and it was found that both daily number of player and sentiment score calculated showed relation based on the timeseries. Each game's timeseries undergone the process of checking for stationary, GC, and finding optimal lag. Each game required their own forecasting model with different lag based on the AIC. A MAPE of 39.3%, 25.49%, 10.37%, and 20.81% was calculated for BloonsTD, ONI, SMC and Stellaris respectively.

Finally, these results were compared with a churn forecasting model that does not uses the sentiment analysis model and found that majority of it performed better than the proposed model. However, the results suggested that if the SteamDB data has a more consistent data distribution, it would perform better, such as a constant spike rather than one big spike. In conclusion, there is no major significance in using SA model to forecast churn on Steam because it was sufficient to only use user generated content as shown by CF / SA models that performed well and better for most of the time.

CHAPTER FIVE

CONCLUSION

This chapter concludes the purpose of the project with explanation of achieved objectives, strengths, limitation, future work and recommendation of this completed research.

5.1 Purpose of Project

To recap, the purpose of this project was to design and develop a churn forecasting model that utilizes the sentiment analysis model created so that stakeholders such as game developers are able to make timely decisions and avoid customer from churning by taking necessary action earlier. Game titles that were selected for this research were related to strategy games which was BloonsTD, Oxygen Not Included, Sid Meier's Civilization VI, and Stellaris.

5.1.1 Objective 1

The first was a form of composite objective, accomplishing two goal in a single task. It was documented as a single objective for easy documentation purposes. The purpose was to design and develop a churn forecasting model that utilizes sentiment analysis. This objective was achieved by going through processes such as preliminary study, knowledge acquisition where understanding of how to design the model, how to collect data, and what type of algorithm to be used were identified. A method of calculating daily sentiment score based on the sentiment obtained from each review were proposed.

The major processes were data collection, data preprocessing, model design and finally model development. Data were collected from two sources which were Steam via scraping through its API to obtain reviews and from SteamDB to obtain the daily number of players. The proposed model was designed so that it would take the predicted sentiment classification. Therefore, the model

design was separated into two parts, which were SA model and CF model. SA used GridSearch and hyperopt for TF-IDF and SVM respectively with linear kernel and SMOTE to create the SA model with hyperparameter tuning method. Using the SA model, the predicted values were then used by the CF model where it was calculated into daily sentiment score and consolidated with daily number of players which were then trained using VAR.

The models were developed using python with various libraries such as nltk, sklearn, imblearn, hyperopt, statsmodel, etc. with extensive algorithm written in the form pseudocode. The model for SA were developed based on the best hyperparameter result found by evaluating its accuracy, precision, recall and F1-score. It was used for creating a dataset that consisted of scores and sentiment from each review for the CF model. CF models were successfully developed for each game.

5.1.2 Objective 2

The second objective was done by to comparing the result between CF model that does not utilize SA model (CF / SA) and CF model that utilizes the SA model (CF x SA). As a clarification, CF x SA used the predicted values from SA model developed to calculate the daily sentiment score whereas CF / SA did not do so by calculating daily sentiment score straight from the scraped review dataset.

The timeseries of CF / SA still showed some form of similarity based on the visualisation despite not using the SA model. Games such as BloonsTD, ONI, and Stellaris performed slightly better whereas SMC performed slightly worse for CF / SA model. The difference in shape of the timeseries for BloonsTD, ONI, and Stellaris were not so different with SMC in terms of all of them have similar look when it is visualized for CF / SA in comparison to their look when visualized with CF x SA. Because there is lack in amount of data for comparison, this study could not conclude as to why CF x SA performed better or worse when compared to CF / SA with the game scope selected. However, according to SMC's MAPE of 10.37% it was proven that the method of

calculating daily sentiment score assisted in forecasting number of player as they both have a form of correlation.

5.2 Strengths

1. Hyperparameter tuning

Besides many comparisons were done to determine what was the best model that can be developed based on the results such as comparing kernels, tuned TF-IDF or SVM were also compared for SA model to be developed based on hyperparameter tuning. Hyperparameter tuning on both TF-IDF and SVM was able slightly increase the performance of the model by optimizing the model's parameter thereby giving it a better performance.

2. Sentiment Analysis model uses merged dataset and SMOTE

One of the problem of sentiment analysis is having an imbalanced class that can reduce the recall of a model. Merging the dataset was able to increase the total amount minority class, but it was still low. Therefore, by applying SMOTE, recall was able to be increased. It is important to determine negative sentiment in game as more valuable information that indicates churn were within the review.

3. Uses UGC to forecast churn

In terms of timeseries forecasting, a basic approach such as univariate timeseries forecasting would produce less accurate result in comparison to multivariate timeseries forecasting such as this research conducted. In addition, it was proven that using sentiment score was a viable method to support the forecasting of daily number of players.

5.3 Limitations

1. Computational power and time constraints

The method to tune TF-IDF and SVM was done in a sequential manner where only after obtaining TF-IDF parameter can the SVM be tuned with tuned TF-IDF. As a result, a potentially better parameter values we're missed due to

computational power.

2. Source of daily number of player data

The source of daily number of player data from SteamDB may not be accurate the entire time as it is a third-party website. An accurate data fed to the forecasting model resulted in an unacceptable model for certain games.

3. Size of data for timeseries

There might have been certain days that will usually have higher numbers of players during certain days or month such as Christmas holiday. This part was cut out as a testing set during splitting and as a result the VAR model were not able to train properly as the amount of training set was only from January to the third week of November.

5.4 Future Work and Recommendations

1. Use cloud to run hyperparameter tuning

Due to the constrain of computational power for hyperparameter tuning that can slows down research, running the hyperparameter tuning programme on a cloud-based system such as Cloud Tensor Processing Units (TPUs) is an option to enable maximum performance.

2. Data directly from developer

A more accurate data of daily number of players that came from Steam should be obtained from the game developer themselves to make sure that data are closely and accurate as possible for forecasting.

3. Larger dataset

Larger dataset that covers at least 2 years must be used for a complete training for the timeseries data to reduce the inaccuracies that is produced by VAR.

4. Include neutral label

This study used the label given by user for sentiment analysis. This label can be either positive or negative only. Including a neutral label for future work to

determine what reviews can be omitted before calculating daily sentiment score may produce better results.

5. Develop better scoring system

The fundamental structure to the forecasting model this research proposed is the usage of sentiment score from Steam. However, based on the review dataset obtained, majority of the review had a score of 0 which were required to be turned into 0.1. There may have been potentially more review that deserved a score but was not given one. Therefore, a better scoring system may help in getting a closer correlation towards the daily player timeseries.

5.5 Summary on Conclusion

This chapter summarizes the reason this study was conducted with its two objectives that have been concluded which are designing and developing a churn forecasting model that utilizes sentiment analysis and finding its significance by comparing the proposed model with a model that does not utilize sentiment analysis.

Some strength that were identified includes the hyperparameter tuning done on SA model, the usage of SMOTE and merging of dataset, and the usage of reviews to forecast churn. Whereas limitations that were identified are the computational power required to run hyperparameter tuning, quality of data and the volume used timeseries forecasting.

Future work and recommendations for this research are to use cloud-based system for hyperparameter tuning, getting data directly from developer, using larger dataset, include neural label and develop a better scoring system.

REFERENCES

- Ammerman, J. (2021). *Steam Reaches 50,000 Game Listings*.
<https://gamerant.com/steam-reaches-50000-game-listings/>
- Ayvaz, S., & Shiha, M. O. (2017). The Effects of Emoji in Sentiment Analysis. *International Journal of Computer and Electrical Engineering*, 9(1), 360–369.
<https://doi.org/10.17706/ijcee.2017.9.1.360-369>
- El Kassem, E. A., Hussein, S. A., Abdelrahman, A. M., & Alsheref, F. K. (2020). Customer churn prediction model and identifying features to increase customer retention based on user generated content. *International Journal of Advanced Computer Science and Applications*, 11(5), 522–531.
<https://doi.org/10.14569/IJACSA.2020.0110567>
- Elgeldawi, E., Sayed, A., Galal, A. R., & Zaki, A. M. (2021). Hyperparameter tuning for machine learning algorithms used for arabic sentiment analysis. *Informatics*, 8(4), 1–21. <https://doi.org/10.3390/informatics8040079>
- Jang, K., Kim, J., & Yu, B. (2021). On analyzing churn prediction in mobile games. *ACM International Conference Proceeding Series*, 20–25.
<https://doi.org/10.1145/3468891.3468895>
- Ji, F. (2019). *Sentiment Analysis and Opinion Extraction of Game Reviews on Steam. October*.
- Kiguchi, M., Saeed, W., & Medi, I. (2022). Churn prediction in digital game-based learning using data mining techniques: Logistic regression, decision tree, and random forest. *Applied Soft Computing*, 118, 108491.
<https://doi.org/10.1016/j.asoc.2022.108491>
- Kilimci, Z. H., Yoruk, H., & Akyokus, S. (2020). Sentiment Analysis Based Churn Prediction in Mobile Games using Word Embedding Models and Deep Learning Algorithms. *INISTA 2020 - 2020 International Conference on INnovations in Intelligent SysTems and Applications, Proceedings*.
<https://doi.org/10.1109/INISTA49547.2020.9194624>

- Kristensen, J. T., & Burelli, P. (2019). Combining sequential and aggregated data for churn prediction in casual freemium games. *IEEE Conference on Computational Intelligence and Games, CIG*, 2019-Augus. <https://doi.org/10.1109/CIG.2019.8848106>
- Lane, R. (2018). *Why users write Steam reviews* / *Rock Paper Shotgun*. <https://www.rockpapershotgun.com/why-users-write-steam-reviews>
- Lee, E., Jang, Y., Yoon, D. M., Jeon, J., Yang, S. Il, Lee, S. K., Kim, D. W., Chen, P. P., Guitart, A., Bertens, P., Periañez, Á., Hadiji, F., Müller, M., Joo, Y., Lee, J., Hwang, I., & Kim, K. J. (2019). Game data mining competition on churn prediction and survival analysis using commercial game log data. *IEEE Transactions on Games*, 11(3), 215–226. <https://doi.org/10.1109/TG.2018.2888863>
- Miao, Y., Jin, Z., Zhang, Y., Chen, Y., & Lai, J. (2021). *Compare Machine Learning Models in Text Classification Using Steam User Reviews*. 40–45. <https://doi.org/10.1145/3507473.3507480>
- Napitu, F., Bijaksana, M. A., Trisetyarso, A., & Heryadi, Y. (2018). Twitter opinion mining predicts broadband internet's customer churn rate. *2017 IEEE International Conference on Cybernetics and Computational Intelligence, CyberneticsCOM 2017 - Proceedings, 2017-Novem*, 141–145. <https://doi.org/10.1109/CYBERNETICSCOM.2017.8311699>
- Park, J., Ma, K., & Leung, H. (2019). Prediction of stock prices with sentiment fusion and SVM granger causality. *Proceedings - IEEE 17th International Conference on Dependable, Autonomic and Secure Computing, IEEE 17th International Conference on Pervasive Intelligence and Computing, IEEE 5th International Conference on Cloud and Big Data Computing, 4th Cyber Scienc*, 207–214. <https://doi.org/10.1109/DASC/PiCom/CBDCOM/CyberSciTech.2019.00046>
- Perišić, A., Jung, D. Š., & Pahor, M. (2022). Churn in the mobile gaming field: Establishing churn definitions and measuring classification similarities. *Expert Systems with Applications*, 191(November 2021). <https://doi.org/10.1016/j.eswa.2021.116277>
- Ranjan, S., & Sood, S. (2020). *Sentiment Analysis Based Telecom Churn Prediction*. January, 6–12. www.stmjournals.com

- Robinson, M. (2013). *GDC Vault - Why Players are Leaving Your Game*. <https://www.gdcvault.com/play/1020698/Why-Players-are-Leaving-Your>
- Rothmeier, K., Pflanzl, N., Hullmann, J. A., & Preuss, M. (2021). Prediction of Player Churn and Disengagement Based on User Activity Data of a Freemium Online Strategy Game. *IEEE Transactions on Games*, 13(1), 78–88. <https://doi.org/10.1109/TG.2020.2992282>
- Singh, S. N., & Sarraf, T. (2020). Sentiment analysis of a product based on user reviews using random forests algorithm. *Proceedings of the Confluence 2020 - 10th International Conference on Cloud Computing, Data Science and Engineering*, 112–116. <https://doi.org/10.1109/Confluence47617.2020.9058128>
- Swanson, D. A. (2015). On the Relationship Among Values of the Same Summary Measure of Error when it is Used Across Multiple Characteristics at the Same Point in Time: An Examination of MALPE and MAPE. *Review of Economics & Finance*, 5(1), 1–14, ISSN(p) 1923-7529, ISSN(e) 1923-8401. <https://escholarship.org/uc/item/1f71t3x9>
- Tan Jie, Y., Sai, A., Chow, K., & Tan Chi, W. (2021). *Sentiment Analysis on Game Reviews: a Comparative Study of Machine Learning Approaches*. October, 25–26.
- Tiwari, S., Verma, A., Garg, P., & Bansal, D. (2020). Social Media Sentiment Analysis on Twitter Datasets. *2020 6th International Conference on Advanced Computing and Communication Systems, ICACCS 2020*, 925–927. <https://doi.org/10.1109/ICACCS48705.2020.9074208>
- Urriza, I. M., & Clariño, M. A. A. (2021). *ASPECT-BASED SENTIMENT ANALYSIS OF USER CREATED GAME REVIEWS*. January 2020, 76–81.
- Viggiato, M., Lin, D., Hindle, A., & Bezemer, C. P. (2021). What Causes Wrong Sentiment Classifications of Game Reviews. *IEEE Transactions on Games*, 1–14. <https://doi.org/10.1109/TG.2021.3072545>
- Vo, N. N. Y., Liu, S., Li, X., & Xu, G. (2021). Leveraging unstructured call log data for customer churn prediction. *Knowledge-Based Systems*, 212, 106586. <https://doi.org/10.1016/j.knosys.2020.106586>

- Wang, N., Guo, J., Liu, X., & Fang, T. (2020). A service demand forecasting model for one-way electric car-sharing systems combining long short-term memory networks with Granger causality test. *Journal of Cleaner Production*, 244, 118812. <https://doi.org/10.1016/j.jclepro.2019.118812>
- Wijman, T. (2021). *The Games Market and Beyond in 2021: The Year in Numbers / Newzoo*. <https://newzoo.com/insights/articles/the-games-market-in-2021-the-year-in-numbers-esports-cloud-gaming>
- Zain, J., & Juleza, J. P. (2021). *Southeast Asia Game Industry Report 2021*.
- Zhang, K., Chuai, G., Gao, W., Liu, X., Maimaiti, S., & Si, Z. (2019). A new method for traffic forecasting in urban wireless communication network. *Eurasip Journal on Wireless Communications and Networking*, 2019(1). <https://doi.org/10.1186/s13638-019-1392-6>
- Zuo, Z. (2018). Sentiment Analysis of Steam Review Datasets using Naive Bayes and Decision Tree Classifier. *Student Publications and Research - Information Sciences*. <https://analytics.twitter.com>

