Chapter 1

Introduction and Overview

1.1 Introduction

This chapter gives introduction to Board game review Prediction, motivation behind the project, list of existing systems and their limitations, the proposed system and organization of report.

1.2 Board game prediction

Board game review prediction is the act of determining the user reviews of a particular game. Since user review generates the average rating of a game which intern help people to understand the quality of the game by seeing the average rating.

1.3 Motivation behind the project

Research is needed to examine how rating in the game increases related to user reviews and how they affect the users.

1.4 Existing system

Existing systems of Board game prediction is not reliable. As rating keeps on changing finding the exact and clear solution to it is not possible manually.

1.5 Proposed system

The proposed system probes into prediction of average rating in the game more precisely using new technology. This system should give much better results when compared to existing system.

1.6 Organization of report

Chapter 2 contains the literature survey and the technologies that are needed. Chapter 3 addresses the System Requirements Specification. Chapter 4 presents the design of the system in terms of modules and the approach used. Chapter 5 shows the implementation details of each of the major modules and analysis drawn from implementation. Chapter 6 explains the system testing. Chapter 7 explains results. Finally, the conclusion of the work and future enhancements is presented in Chapter 8.

Chapter 2

Literature survey

2.1 Introduction

This chapter covers the literature in the field of Linear Regression and Random Forest Regression. Predictive analysis using machine learning has been gaining popularity in recent times. In this paper, the Random Forest regression model and Linear Regression is used to predict average rating of board games from the given data set. The performance of the Random Forest model and Linear Regression is investigated and compared with to each other. Impact of regularization, correlation, high bias/high variance and feature selection on the learning models are also studied.

2.2 Literature survey

Studies have shown that when testing structural equation models, researchers attempt to establish a model that will generalize to other samples from the same population. Unfortunately, researchers tend to test and respecify models during this attempt, capitalizing on the characteristics inherent within the sample data in which the model is being developed. Several measures of model fit exist to aid researchers when trying to select a model that fits the sample data well. However, these measures fail to consider the predictive validity of a model, or how well it will generalize to other samples from the same population.

Predictive analysis has been vital to the growth of companies like Target, Amazon and Netflix in identifying customer behavior through Business Intelligence models. These instances suggest clearly that there are business opportunities to be harnessed in the field of predictive analysis. Applying predictive analysis to articles on the web can be vital in extending a creator into a large-scale distributor of his/her content.

Current marketing techniques make use of a business intelligence approach called Decision Support Systems (DSS) to predict popularity before the actual marketing of the product. DSS is an information system that supports business or decision-making activities for organizations. Adaptive Business Intelligence (ABI) provides the power of prediction and optimization working together to increase popularity of articles on the web. Thus, knowing in advance the game which are likely to become popular, based on the game content, is vital for business intelligence approaches. A low share count is a clear indication to make changes to the game content to improve popularity.

2.3 Technologies used

Various software's required for project are python, Anaconda and Jupyter Notebook.

Chapter 3

System Requirements

3.1 Introduction

System requirements cover all of the requirements at the system level, which describe the functions the research as a whole should fulfil. It is expressed in an appropriate combination of textual statements, views and non-functional requirements; the latter expressing the levels of speed, reliability, scalability and standards to meet.

3.2 Software requirements

The software required to implement this project can be installed from Anaconda website which contains Jupyter Notebook for Mac, windows or Linux and usage of each has be mentioned in Table 3.1

Software	Use
Windows, Mac, Linux	Operating System
Python	Programming language used to code biological properties, constraints and operations.
Anaconda	It’s a python distribution that makes easy to install and use Jupyter notebook.
Jupyter Notebook	It’s an open source web application that allows to data cleaning and transformation, numerical simulation, statistical modelling, data visualization, machine learning, and much more.

Table 3.1: Softwares used

3.3 Hardware requirements

Hardware requirements define the minimal and optimal configurations for the system. Following are the hardware requirements for this project:

Processor : 1.5 GHz

Physical memory : 1 GB

Secondary memory : 3 GB

3.4 Functional requirements

These are statements of services the system should provide, how the system should react to particular inputs, and how the system should behave in particular situations. In some cases, the functional requirements may also explicitly state what the system should not do.

3.5 Non-functional requirements

Non Functional Requirements describe the constraints on development process and standards to be followed in development process.

Chapter 4

System Analysis and Design

4.1 Introduction

Systems analysis is a problem solving technique that decomposes a system into its component pieces for the purpose of the studying how well those component parts work and interact to accomplish their purpose. Systems design is the process of defining the architecture, components, modules, interfaces and data for a system to satisfy specified requirements. This chapter explains both the model of linear regression and forest regression.

4.2 Objectives of the system

The system should read the parameters from a configuration file and normalize the given dataset into test set and training set and remove all the redundant data from the give dataset. This test sets are fit into Linear Regression and Random Forest repressors for prediction and comparison of the two algorithms performance in prediction.

4.3 Modules design

4.3.1 Normalization of dataset

Normalization of dataset means removing the redundant values from the given datasets which will not contribute to the Linear Regression and Random Forest Regression.

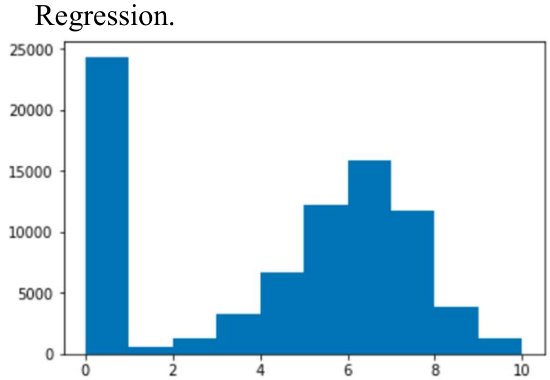

Fig 4.1 Histogram for average_rating.

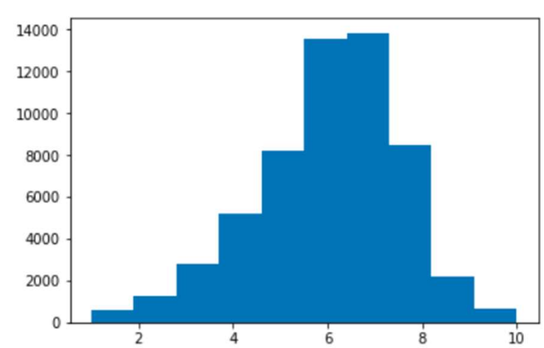

Fig 4.2 Histogram for Normalised average_rating

4.3.2 Correlation matrix for Normalized dataset

In order to determine how strong the relationship is between two variables, a formula must be followed to produce what is referred to as the coefficient value. The coefficient value can range between -1.00 and 1.00. If the coefficient value is in the negative range, then that means the relationship between the variables is negatively correlated, or as one value increases, the other decreases. If the value is in the positive range, then that means the relationship between the variables is positively correlated, or both values increase or decrease together.

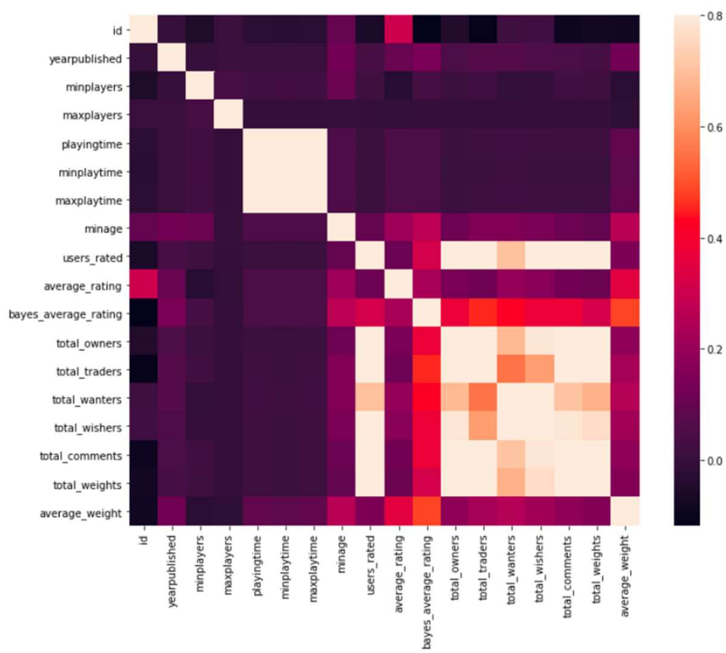

Fig 4.3 Correlation matrix for normalized dataset.

Chapter 5

System Implementation

5.1 Introduction

This chapter explains the implementation of Forest Regression and Linear Regression using python with data-sets

5.2 Data set

	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U
1	id	type	name	yearpubli	minplayer	maxplaye	playingtin	minplayti	maxplayti	minage	users_rate	average_r	bayes_ave	total_own	total_trad	total_wan	total_wish	total_com	total_wei	average_weight	
2	12333	boardgam	Twilight S	2005	2	2	180	180	180	13	20113	8.33774	8.22186	26647	372	1219	5865	5347	2562	3.4785	
3	120677	boardgam	Terra Mys	2012	2	5	150	60	150	12	14383	8.28798	8.14232	16519	132	1586	6277	2526	1423	3.8939	
4	102794	boardgam	Caverna: T	2013	1	7	210	30	210	12	9262	8.28994	8.06886	12230	99	1476	5600	1700	777	3.7761	
5	25613	boardgam	Through tl	2006	2	4	240	240	240	12	13294	8.20407	8.05804	14343	362	1084	5075	3378	1642	4.159	
6	3076	boardgam	Puerto Ric	2002	2	5	150	90	150	12	39883	8.14261	8.04524	44362	795	861	5414	9173	5213	3.2943	
7	31260	boardgam	Agricola	2007	1	5	150	30	150	12	39714	8.11957	8.03847	47522	837	958	6402	9310	5065	3.616	
8	124742	boardgam	Android: I	2012	2	2	45	45	45	14	15281	8.1676	7.97822	24381	680	627	3244	3202	1260	3.3103	
9	96848	boardgam	Mage Knig	2011	1	4	150	150	150	14	12697	8.15901	7.96929	18769	367	1116	5427	2861	1409	4.1292	
10	84876	boardgam	The Castle	2011	2	4	90	30	90	12	15461	8.07879	7.95011	20558	215	929	3681	3244	1176	3.0442	
11	72125	boardgam	Eclipse	2011	2	6	200	60	200	14	15709	8.07933	7.93244	17611	273	1108	5581	3188	1486	3.6359	
12	2651	boardgam	Power Gri	2004	2	6	120	120	120	12	34422	7.9888	7.91794	38633	550	1171	6157	7531	3998	3.2911	
13	164153	boardgam	Star Wars:	2014	2	5	90	90	90	0	3980	8.43944	7.91643	8477	57	701	2970	736	360	3.225	
14	115746	boardgam	War of the	2012	2	4	150	150	150	13	3870	8.35044	7.88643	6257	71	677	2431	771	288	3.9375	
15	121921	boardgam	Robinson	2012	1	4	180	90	180	14	10539	8.09283	7.88503	15896	217	1379	5821	2109	896	3.6328	
16	35677	boardgam	Le Havre	2008	1	5	200	100	200	12	15774	7.99115	7.88172	16429	205	1343	5149	3458	1450	3.7531	
17	28720	boardgam	Brass	2007	3	4	180	120	180	13	8785	8.03071	7.85824	9171	149	798	2858	2259	1012	3.8646	
18	126163	boardgam	Tzolk'in: T	2012	2	4	90	90	90	13	12143	7.98673	7.83148	13958	120	1056	3945	2144	933	3.5595	

Figure 5.1 Snapshot of dataset

5.3 Algorithm

Pandas module is used to read the dataset from csv file into data frame. There are some unused column in the data frame, hence those are to be removed. The column average_rating has some values as 0 since 0 doesn’t contribute to Linear Regression and Random Forest regression we have removed rows where average_rating and user_rating. The correlation co-efficient is calculated using dataframe.corr()(a method available in pandas).This co-efficient is used to find the related columns in the data. In Heat plot the plot is used to find the relation and the columns highly related are removed such as bayes_average_rating, type, name, and id. The prediction column average_rating should as be removed since it is output.

Data should be split into train and test, pandas module has sample() module which takes input as frac(fraction) 80% train and 20% test is split. Sklearn module has many algorithms such as Linear Regression and Forest Regression.

Linear regression of a single independent variable is used to predict the value of a dependent variable, in multiple linear regression two or more independent variables are used to predict the value of a dependent variable.

RandomForestRegressor(n_estimators=10, criterion='mse', max_depth=None, min_samples_split=2, min_samples_leaf=1, min_weight_fraction_leaf=0.0, max_features='auto', max_leaf_nodes=None, min_impurity_decrease=0.0, min_impurity_split=None, bootstrap=True, oob_score=False, n_jobs=1, random_state=None, verbose=0, warm_start=False). A random forest is a meta estimator that fits a number of classifying decision trees on various sub-samples of the dataset and use averaging to improve the predictive accuracy and control over-fitting.

LinearRegression.fit() is used to train the model using train data and RandomForestRegressor.fit() is used to train the model using train data, predict method is used to predict the trained model.

mean squared error of an estimator measures the average of the squares of the error or deviation that is, the difference between the estimator and actual value of what is estimated. Mean_squared_error(predicted,test) gives the mean squared error

Chapter 6

System Testing

6.1 Introduction

System testing of software is testing conducted on a complete, integrated system to evaluate the system's compliance with its specified requirements. This chapter covers unit and integration testing. Unit testing can be done by testing each of the algorithm separately. Integration testing combines test of both the algorithm performance and prediction of average rating of the board game.

6.2 Unit testing

In unit testing the smallest testable parts of an application, called units, are individually and independently scrutinized for proper operation. Unit test were conducted on units read from dataset, Normalize the dataset, plot the graph of correlation, predict the average rating, calculate mean squared error save results and quantify using both the algorithm independently. Each unit performing its task as expected.

6.2.1 Read Dataset

The read dataset module should read values from dataset file.

6.3 Integration testing

Integration testing is a level of software testing where individual units are combined and tested as a group. Linear regression and Random Forest Regression were trained using the dataset and tested using test dataset simultaneously. Calculation of mean squared error and prediction was as expected.

Chapter 7

Results

7.1 Introduction

This chapter cover's the experimental value and the exact value predicted by the regression models.

7.2 Accuracy

Due to huge dataset it's not accurate for a linear model to predict accurately, whereas non-linear model such as Random forest regression it show's to be more accurate.

Here the following code shows the performance of both the model in the given dataset.

```
#accuracy for Linear Regression
from sklearn.linear_model import LinearRegression
from sklearn.metrics import mean_squared_error
LR = LinearRegression()
LR.fit(train[columns],train[target])
o/p:LinearRegression(copy_X=True, fit_intercept=True, n_jobs=1, normalize=False)
predictions = LR.predict(test[columns])
mean_squared_error(predictions, test[target])
o/p:2.078819032629326
#accuracy for Random forest regressor
from sklearn.ensemble import RandomForestRegressor
RFR = RandomForestRegressor(n_estimators = 100, min_samples_leaf =20, random_state=1)
RFR.fit(train[columns],train[target])
o/p:RandomForestRegressor(bootstrap=True, criterion='mse', max_depth=None,
        max_features='auto', max_leaf_nodes=None,
        min_impurity_decrease=0.0, min_impurity_split=None,
        min_samples_leaf=20, min_samples_split=2,
        min_weight_fraction_leaf=0.0, n_estimators=100, n_jobs=1,
        oob_score=False, random_state=1, verbose=0, warm_start=False)
predictions = RFR.predict(test[columns])
mean_squared_error(predictions, test[target])
```

o/p:1.465461608331042

7.3 Results Analysis

	Without	Processed
LR	6.236480747475232	2.078819032629326
RFR	0.9942627042563742	1.465461608331042

Table 7.1 : Mean square error analysis

7.4 Main results

Prediction are accurate for some dataset for Linear regression whereas in-accurate for Random forest and vise-versa.

The following snapshot give's the brief view of the project.

```
In [75]: test[columns].iloc[5]

Out[75]: yearpublished      2012.0000
         minplayers            2.0000
         maxplayers            5.0000
         playingtime          60.0000
         minplaytime          60.0000
         maxplaytime          60.0000
         minage               12.0000
         users_rated       19864.0000
         total_owners      24419.0000
         total_traders       257.0000
         total_wanters       995.0000
         total_wishers      4706.0000
         total_comments     3898.0000
         total_weights      1493.0000
         average_weight        2.5177
         Name: 27, dtype: float64

In [76]: rating_LR = LR.predict(test[columns].iloc[5].values.reshape(1,-1))
         rating_RFR = RFR.predict(test[columns].iloc[5].values.reshape(1,-1))

         print(rating_LR)
         print(rating_RFR)

         [8.21185069]
         [7.63817745]

In [77]: test[target].iloc[5]

Out[77]: 7.82181
```

Figure 7.1: RFR predicts accurately than LR

```
In [36]: test[columns].iloc[0]

Out[36]: yearpublished      2011.0000
         minplayers            2.0000
         maxplayers            6.0000
         playingtime         200.0000
         minplaytime          60.0000
         maxplaytime         200.0000
         minage               14.0000
         users_rated       15709.0000
         total_owners      17611.0000
         total_traders       273.0000
         total_wanters      1108.0000
         total_wishers      5581.0000
         total_comments     3188.0000
         total_weights      1486.0000
         average_weight        3.6359
         Name: 9, dtype: float64

In [37]: rating_LR = LR.predict(test[columns].iloc[0].values.reshape(1,-1))
         rating_RFR = RFR.predict(test[columns].iloc[0].values.reshape(1,-1))

         print(rating_LR)
         print(rating_RFR)

         [8.12061283]
         [7.85919904]

In [38]: test[target].iloc[0]

Out[38]: 8.07933
```

Figure 7.2: LR predicts accurately than RFR

Conclusion and Future work

In this work, the Random Forest regression model and Linear regression was applied on the board game data set. Through optimization, a more accurate RFR model and LR was obtained. The optimized RFR and LR model predicted average rating of a board game.

For our given dataset we get mean square error as which is not accurate for a linear model the accurate value will be 0 for linear regression, but non-linear regression such as Random forest regression method predict 1 which is more accurate than linear regression.

For some input dataset to the regression's model we find Linear to be more accurate than Random forest regression.

In the future, the optimized regression model can be applied on tweets to predict popularity. Clustering tweets on the sentiments of their text and other advanced features could potentially lead to obtaining new insights into their distributions.

References

1. L. Breiman, J. Friedman, R. Olshen, and C. Stone, Classification and Regression Trees. Monterey, CA: Wadsworth and Brooks, 1984.
2. L. Breiman, "Random forests," Mach. Learn., vol. 45, no. 1, pp. 5-32, oct 200l.
3. Linear Regression, Wikipedia
4. G.A.FSeber, Linear Regression Analysis, Wiley,2003
5. Sean Back, Scrapers, boardgamegeek, GitHub, 2015