DATA MINING AND ANALYSIS

Ayushi Shrivastava
MS Business Analytics (Spring 2019)
CSU, East Bay

SUMMARY

Mobile applications are increasingly playing a vital role in business with a larger customer base as well as workforce relying on the convenience of access on-demand to information & solutions. Tracking number of downloads can have an intense and helpful impact on evaluating an app's success rate and take appropriate measures in order to emerge out as a leader. Objective of our project is to extract data from google play store and predict number of downloads and understand which factors influence in downloading the apps.

The data for Android apps on Google Play Store was available from www.kaggle.com and it contained around 9360 records and 13 attributes. In our analysis we have decided to run five models in order to understand performance of this data set in each approach. Models used are: Multiple Regression, Regression Tree, Boosted Tree, Bootstrap forest, Neural Networks. Each model is then compared based on parameters such as R-square, RMSE/RASE and the model with best results would be chosen for making predictions of number of downloads.

INTRODUCTION

With constant advances in the field of technology, companies must adapt to the recent changes that affect the development of corporations, particularly the growth of the mobile web. A big portion of digital time is spent in mobile apps, and this presents an enormous opportunity for many companies to tap new clients and niche markets.

Based on this infographic, it can be understood how important it is for companies to keep their applications updated and use mobile analytics to track the success of an app. Since it is an opportunity to boost business, apps have to provide a distinctive customer experience in order to build an enduring inspiration. One of the measures is to see the ranking of your app via the Google Play store/Apple App store platform. The ranking factor that Google and Apple use, relies on 3 major factors:

Amount of downloads the apps has,

The average ratings

and the reviews the app has been able to rack up.

Seeing the rise in number of mobile app-based businesses and time spent on these apps , we decided to predict number of downloads in an app based on various performance measures available. The Play Store apps data has enormous potential to drive app-making businesses to success. Predictions from these can be used to gain insights and see where a particular app stands in comparison with their competitors.

Chapter 1: MAIN CHAPTER

i. Data Processing

Data Cleaning:

The original data set contained following variables:

App – Application Name
Category – category of the app
Rating – rating of the app(1 to 5)
Reviews – Number of reviews
Size – Size of the app
Installs – Number of downloads
Type – free /paid app
Price
Content Rating – Age group the app caters to
Genre
Last Updated – last date the app was updated
Current ver – current version of app
Android ver – version of mobile phone

In order to run the datasets in jmp, it needed certain amount of conditioning as the raw data had few missing values and several values not in proper format. We started with removing records with missing values and incorrect format and thus the data was reduced by approx. 1900 records. we removed few columns such as App, current version, Type, genre as the data from these columns were redundant. One of the important steps in cleaning the data is to remove outliers. Using jmp, few outliers were detected in our data set as shown below:

Number of Reviews & Installs is such that it can vary form few hundreds to millions and hence, range of these variables can be pretty wide. Similarly, for price, majority of apps are free having value 0, thus paid apps are detected as outliers. we decided not to remove these outliers as they can possibly be a correct data record.

Data Conditioning:

Data column Category consisted of 33 unique categories for all the records, and thus while running the data with such large number of unique categories caused and ill-conditioning error. Thus, we decided to reduce the number of categories by clubbing them to 7 main categories and been recorded in jmp. In order to study the effect of each category, we changed these categories to individual variable having nominal values.

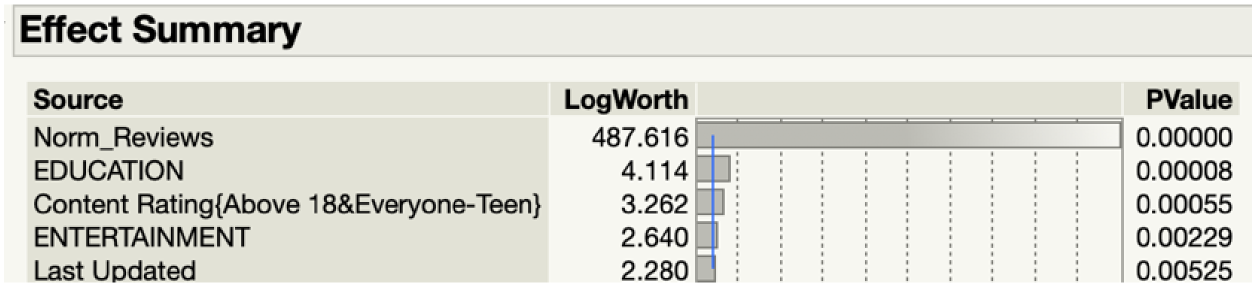
After cleaning & conditioning the data, the final list of variables is as below:

Variable	Description
QUOTIDIAN	Category is Quotidian? (Yes = 1, No = 0)
FAMILY	Category is Family? (Yes = 1, No = 0)
SOCIAL	Category is Social? (Yes = 1, No = 0)
BUSINESS	Category is Business? (Yes = 1, No = 0)
ENTERTAINMENT	Category is Entertainment? (Yes = 1, No = 0)
HEALTH & FITNESS	Category is Health & Fitness? (Yes = 1, No = 0)
EDUCATION	Category is Education? (Yes = 1, No = 0)
Rating	Overall User Rating of the app(out of 5)
Reviews	Number of user reviews for the app(Lacs)
Size	Size of the apps(MB)
Installs	Number of user downloads/installs for the app
Price	Price of the app($)
Content Rating	Age group the app is targeted at - Everyone/Mature 17+/Teen
Last Updated	Date when the app was last updated on Play Store(Year)
Android Version	Minimum required Android version

The final step in data processing was to *normalize the numerical variables* i.e Ratings, Reviews, Size, installs as all these variables have different unit and huge variations in values, it would be better to bring them to common scale in order to have precise predictions of the response variable.

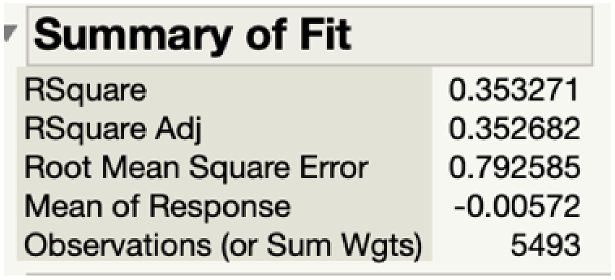
Chapter2: Multiple Regression Model

We created a regression model with variable Number of Installs as the response variable and all other variables as independent variable. To run the regression model, we tried all the methods such as Manual Elimination, Automated forward elimination , backward elimination and mixed stepwise. Out all these models, mixed stepwise model provided the best result. We obtained the following results using the Effect Summary, Summary of Fit, Analysis of Variance, Parameters Estimates, and Effect Tests with a threshold for p values as 0.05.



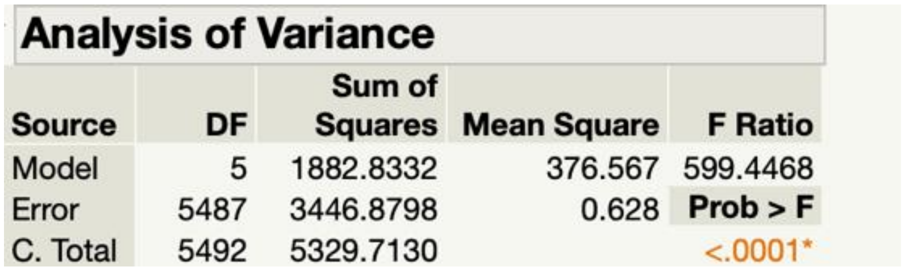
Effect Summary

Source	LogWorth	PValue
Norm_Reviews	487.616	0.00000
EDUCATION	4.114	0.00008
Content Rating{Above 18&Everyone-Teen}	3.262	0.00055
ENTERTAINMENT	2.640	0.00229
Last Updated	2.280	0.00525



Summary of Fit

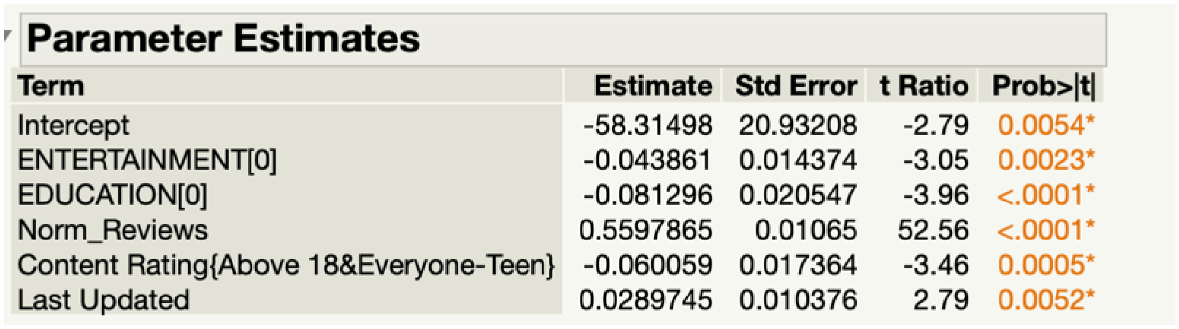
RSquare	0.353271
RSquare Adj	0.352682
Root Mean Square Error	0.792585
Mean of Response	-0.00572
Observations (or Sum Wgts)	5493



Analysis of Variance

Source	DF	Sum of Squares	Mean Square	F Ratio
Model	5	1882.8332	376.567	599.4468
Error	5487	3446.8798	0.628	Prob > F
C. Total	5492	5329.7130		<.0001*

Model has a decent R-square of 0.35 which means 35% of variation in number of installs can be explained by variation in predictors Entertainment, Education, Number of reviews, Content Ratings & last updated Ratio has a probability of <0.0001 (a chance that the model is not linear). P-values for the predictors' regression coefficients (Estimates) for are less than 0.05, and thus these coefficients are statistically significant.

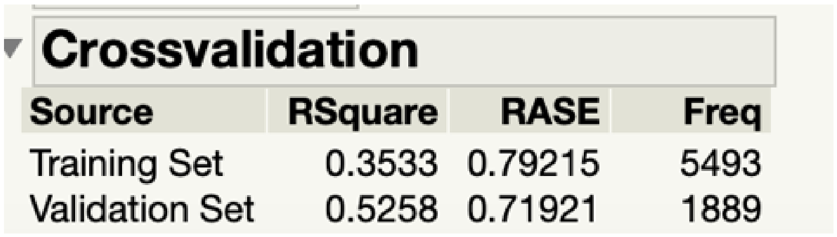


Parameter Estimates

Term	Estimate	Std Error	t Ratio	Prob>|t|
Intercept	-58.31498	20.93208	-2.79	0.0054*
ENTERTAINMENT[0]	-0.043861	0.014374	-3.05	0.0023*
EDUCATION[0]	-0.081296	0.020547	-3.96	<.0001*
Norm_Reviews	0.5597865	0.01065	52.56	<.0001*
Content Rating{Above 18&Everyone-Teen}	-0.060059	0.017364	-3.46	0.0005*
Last Updated	0.0289745	0.010376	2.79	0.0052*

The regression equation is :

Norm_Installs = -58.31 – 0.0438 * ENTERTAINMENT[0] + 0.0438*ENTERTAINMENT[1] – 0.081 * EDUCATION[0] + 0.081 * EDUCATION[1] + 0.5597 * Norm_Reviews – 0.060 * Content rating + 0.0289 * Last Updated



Crossvalidation

Source	RSquare	RASE	Freq
Training Set	0.3533	0.79215	5493
Validation Set	0.5258	0.71921	1889

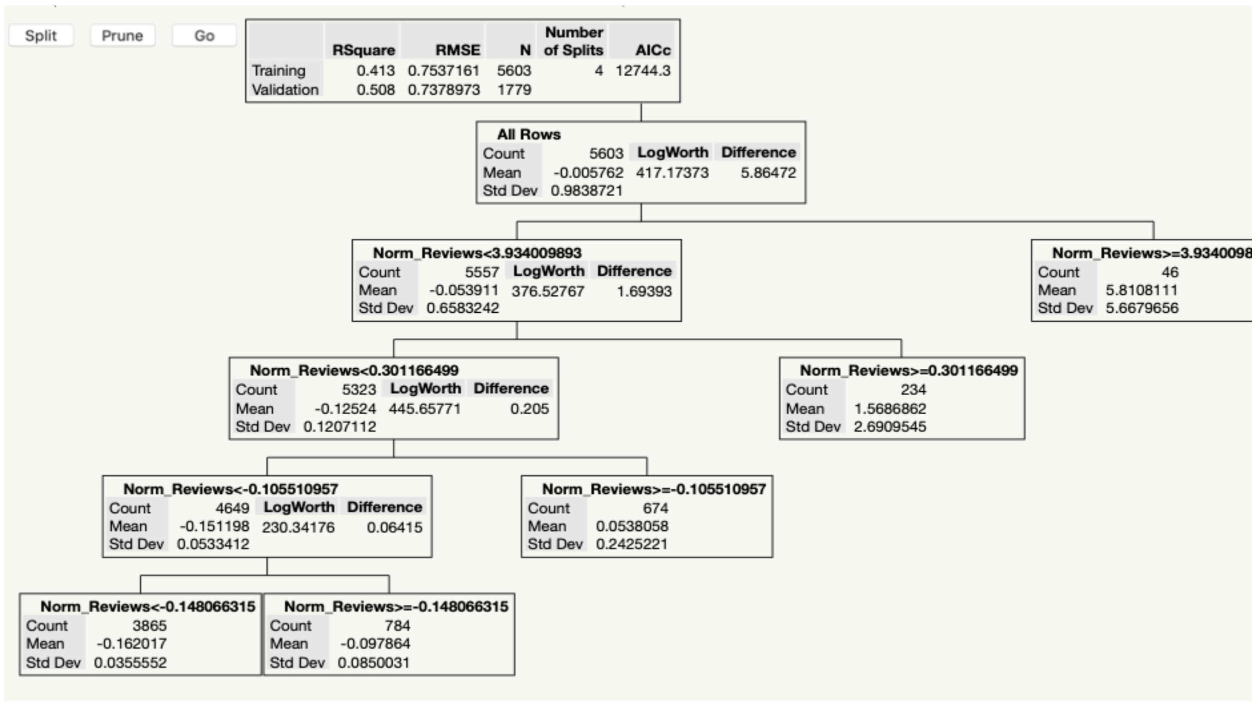
Based on cross validation table we can see that for R square, change in validation & training set is ((0.5258-0.3533)/0.3533) ~ 0.488 ~48.8%. Similarly, for RASE change in validation set is ((0.79215-0.71921)/0.79215) ~ 9.2%.Since these values are under 50%, we can say that the regression equation fits well into the validation data set and there is no issue of overfitting. Overall the model fits well and is statistically significant and hence can be used to make predictions.

2b) <u>REGRESSION TREE</u>

Fit Details

The goal of Decision tree is to create a model that predicts the value of a target variable based on several input variables. A tree is built by splitting the source set, constituting the root node of the tree, into subsets - which constitute the successor children. The splitting is based on a set of splitting rules. This process is repeated on each derived subset in a recursive manner called recursive partitioning. The recursion is completed when the subset reaches homogeneity, or when splitting no longer adds value to the predictions.

A regression tree was also trained on the Google play store dataset and the fit details of the model are seen below. As seen in the table, we observe a RSquare value for training and validation data at 0.426 and 0.471 respectively and the change in RSquare is 10.5%, which tell us that this model does not suffer from overfitting. Moreover, we see that there were a total 4 splits used for the final regression tree. Also, we observe that Training and Validation data RMSE values are at 0.7463& 0.7596; and the change in RMSE is only 5.5%. This indicates that the chance of model overfitting is considerably low.



	RSquare	RMSE	N	Number of Splits	AICc
Training	0.413	0.7537161	5603	4	12744.3
Validation	0.508	0.7378973	1779		

All Rows

		LogWorth	Difference
Count	5603		
Mean	-0.005762	417.17373	5.86472
Std Dev	0.9838721		

Norm_Reviews<3.934009893

		LogWorth	Difference
Count	5557		
Mean	-0.053911	376.52767	1.69393
Std Dev	0.6583242		

Norm_Reviews>=3.9340098

Count	46
Mean	5.8108111
Std Dev	5.6679656

Norm_Reviews<0.301166499

		LogWorth	Difference
Count	5323		
Mean	-0.12524	445.65771	0.205
Std Dev	0.1207112		

Norm_Reviews>=0.301166499

Count	234
Mean	1.5686862
Std Dev	2.6909545

Norm_Reviews<-0.105510957

		LogWorth	Difference
Count	4649		
Mean	-0.151198	230.34176	0.06415
Std Dev	0.0533412		

Norm_Reviews>=-0.105510957

Count	674
Mean	0.0538058
Std Dev	0.2425221

Norm_Reviews<-0.148066315

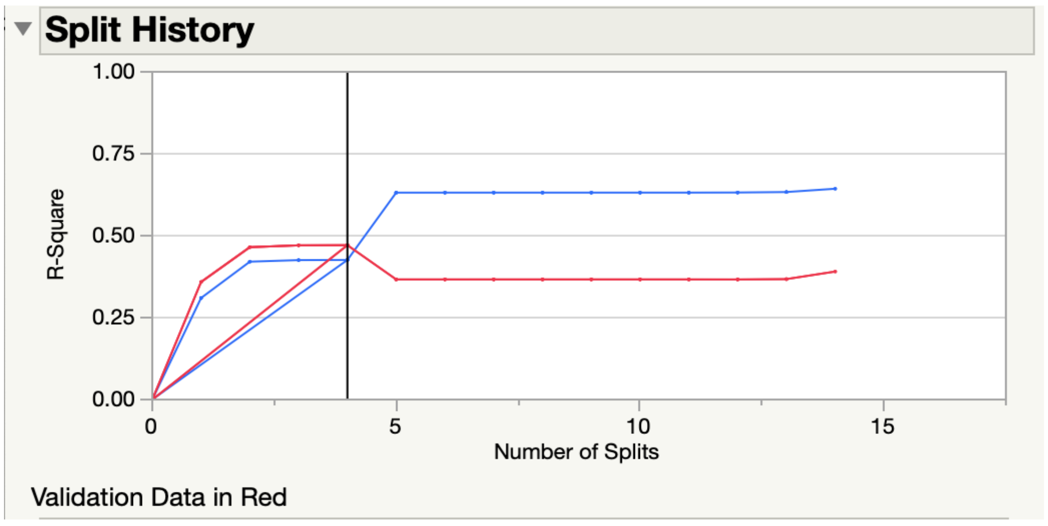
Count	3865
Mean	-0.162017
Std Dev	0.0355552

Norm_Reviews>=-0.148066315

Count	784
Mean	-0.097864
Std Dev	0.0850031

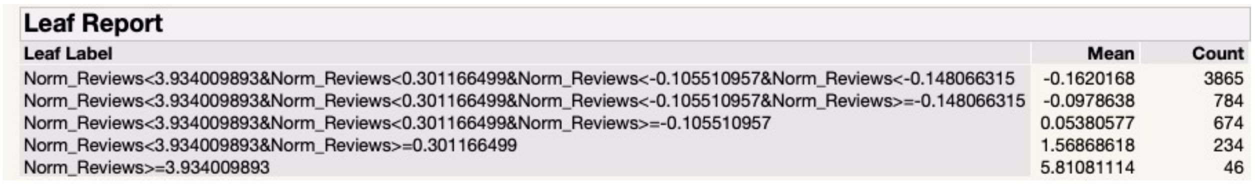
Split History

From split history, we can observe that the validation RSquare reaches its maximum (47.1%) at n=4 i.e., as it represents the most optimal number of splits. It is seen that although the number of splits increases for the training data, this is not to be used to avoid overfitting for which regression trees are prone to. Validation data is indicated in red.



Leaf Report

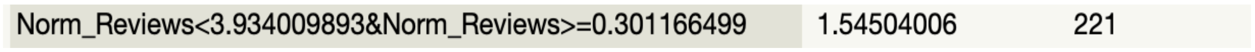
The leaf report provides a summary of the regression tree, with rules for predicting the response. This leaf report allows users to quickly derive rules that are based on features included as splits. The availability of a leaf report is one of the main advantages of running a regression tree as compared to other models which have a more black-box nature.



Leaf Report

Leaf Label	Mean	Count
Norm_Reviews<3.934009893&Norm_Reviews<0.301166499&Norm_Reviews<-0.105510957&Norm_Reviews<-0.148066315	-0.1620168	3865
Norm_Reviews<3.934009893&Norm_Reviews<0.301166499&Norm_Reviews<-0.105510957&Norm_Reviews>=-0.148066315	-0.0978638	784
Norm_Reviews<3.934009893&Norm_Reviews<0.301166499&Norm_Reviews>=-0.105510957	0.05380577	674
Norm_Reviews<3.934009893&Norm_Reviews>=0.301166499	1.56868618	234
Norm_Reviews>=3.934009893	5.81081114	46

An example of a rule derived from the leaf report would be the following:

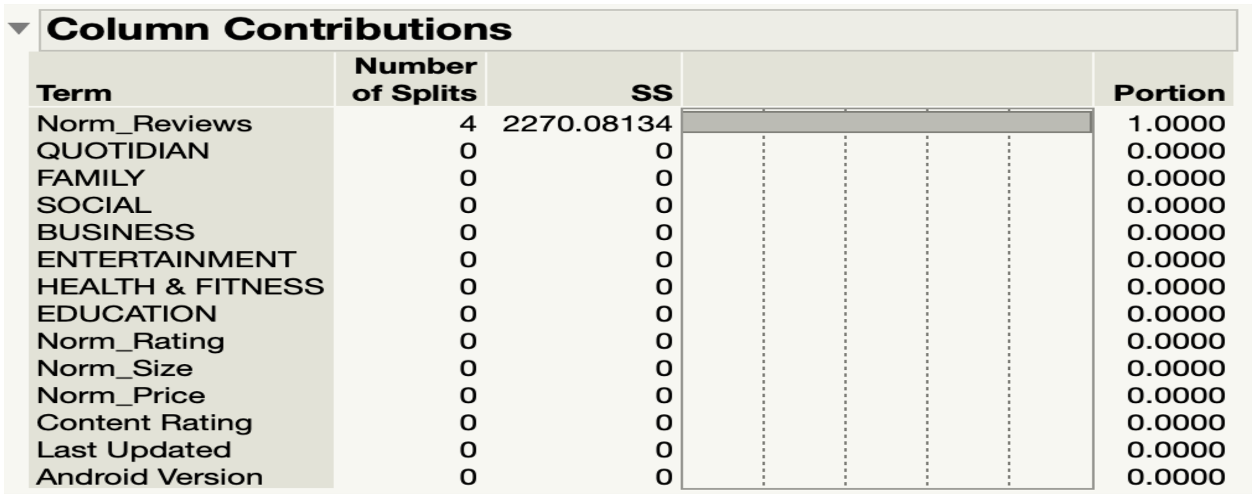


Norm_Reviews<3.934009893&Norm_Reviews>=0.301166499	1.54504006	221

This means that if the reviews are higher than 0.3 (Norm_Reviews>=0.3), and less than 3.93 (Norm_Reviews< 3.93) then the average number of installs will be 1.545.

Column Contributions

From the column contributions, we can observe the predictor variables that contributed to the splits in the regression tree. In the below table, number of reviews is the only contributor to the number of installs.

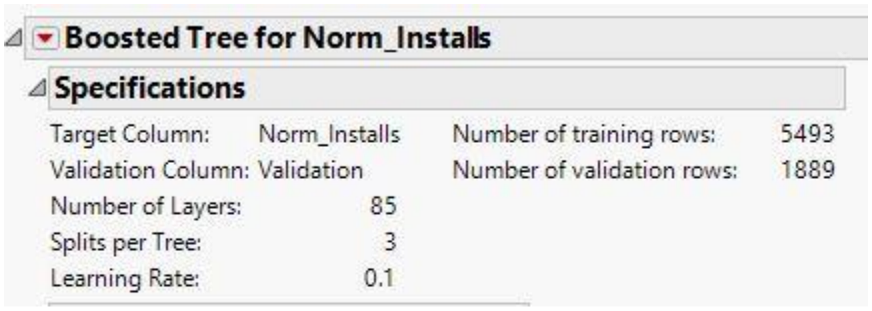


Column Contributions

Term	Number of Splits	SS	Portion
Norm_Reviews	4	2270.08134	1.0000
QUOTIDIAN	0	0	0.0000
FAMILY	0	0	0.0000
SOCIAL	0	0	0.0000
BUSINESS	0	0	0.0000
ENTERTAINMENT	0	0	0.0000
HEALTH & FITNESS	0	0	0.0000
EDUCATION	0	0	0.0000
Norm_Rating	0	0	0.0000
Norm_Size	0	0	0.0000
Norm_Price	0	0	0.0000
Content Rating	0	0	0.0000
Last Updated	0	0	0.0000
Android Version	0	0	0.0000

2c) BOOSTED TREE

The boosted tree is used to improve the predictive ability of the regression tree. It produces an additive decision tree model that is based on many smaller decision trees that are constructed in layers. Each layer tries to reduce the prediction error. It is a concept of layer improvement rather than how many trees are built.

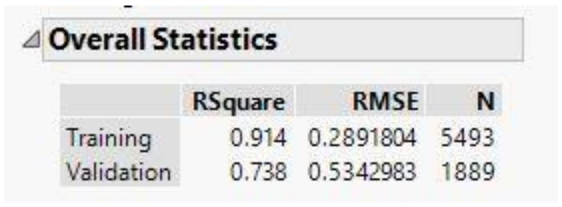


Boosted Tree for Norm_Installs

Specifications

Target Column:	Norm_Installs	Number of training rows:	5493
Validation Column:	Validation	Number of validation rows:	1889
Number of Layers:	85		
Splits per Tree:	3		
Learning Rate:	0.1		

From the above **Specification table**, we can see that our target variable is "Install", we have used validation, and 85 layers were used for the model with 3 splits per tree, and learning rate is 0.1 which tell us about the percentage of error propagated into the next layer.

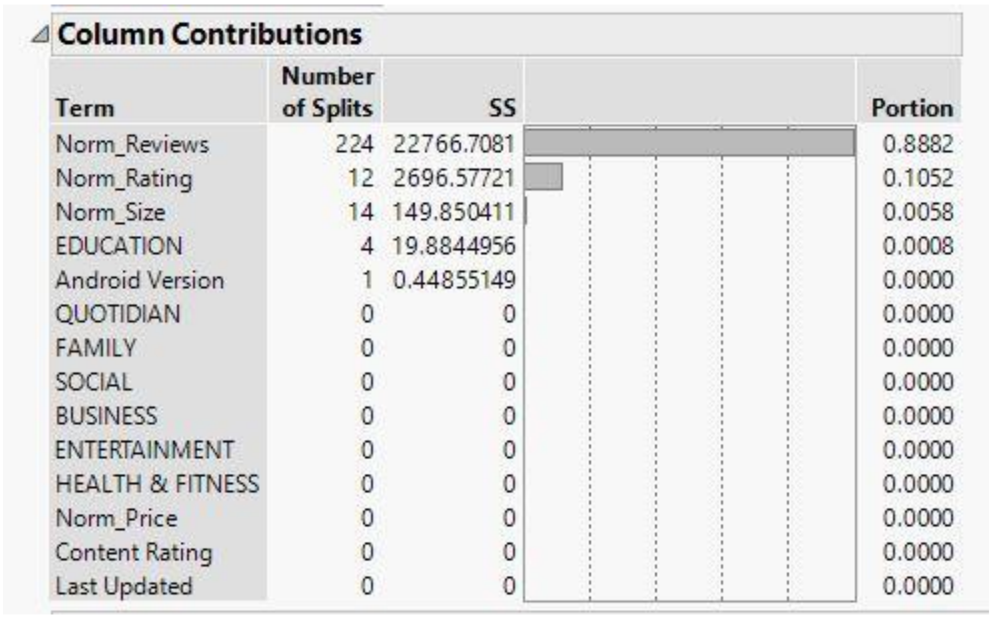


Overall Statistics

	RSquare	RMSE	N
Training	0.914	0.2891804	5493
Validation	0.738	0.5342983	1889

Based on **Overall Statistics** table, we say that RSquare Validation is lesser than RSquare Training by 0.176 with ~19% decrease. So, in this case we can clearly see that there is no problem of overfitting our training data fits well with our validation data.

However, in RMSE we can clearly see **Overfitting** as validation data is greater than training data by 0.2451 with ~84% increase. So, in this case we can say that our training data doesn't fits well with our validation data.


Column Contributions

Term	Number of Splits	SS	Portion
Norm_Reviews	224	22766.7081	0.8882
Norm_Rating	12	2696.57721	0.1052
Norm_Size	14	149.850411	0.0058
EDUCATION	4	19.8844956	0.0008
Android Version	1	0.44855149	0.0000
QUOTIDIAN	0	0	0.0000
FAMILY	0	0	0.0000
SOCIAL	0	0	0.0000
BUSINESS	0	0	0.0000
ENTERTAINMENT	0	0	0.0000
HEALTH & FITNESS	0	0	0.0000
Norm_Price	0	0	0.0000
Content Rating	0	0	0.0000
Last Updated	0	0	0.0000

From the **Column Contributions** table above, we can say that the predictor ‘Norm_Reviews’ has played a significant role in building the boosted tree by participating in 224 splits and it has contributed for 88.82% out of all the contributions made by the predictors.

Also, the variable Norm_Rating has contributed to the boosted tree by participating in 12 splits with 10.52% contributions made by the predictors.

The predictor ‘Android Version’ has contributed the least by participating in just 1 split.

2d) BOOTSTRAP FOREST

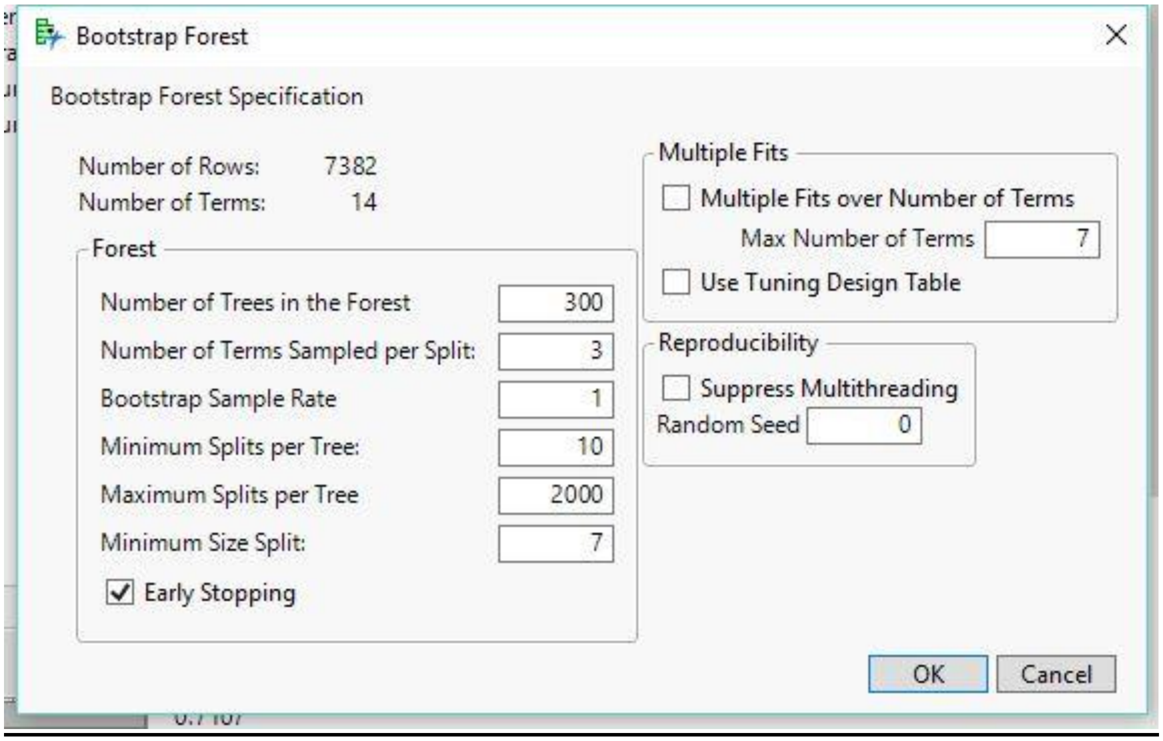
The Bootstrap Forest platform fits an ensemble model by averaging many decision trees each of which is fit to a bootstrap sample of the training data. Each split in each tree considers a random subset of the predictors. In this way, many weak models are combined to produce a more powerful model. The final prediction for an observation is the average of the predicted values for that observation over all the decision trees. It is based on an idea of developing multiple trees using **Recursive Partitioning Methodology**. Each tree is built on the training data or a sample of training data. This sample is called **Bootstrap Sample** wherein sample of observations are drawn with replacement. **Replacement** means we might consider similar sample data set for our next sample. In addition to this, predictors are sampled at each split.

A Bootstrap Forest was trained on the Google Play Store Dataset and results were based on following algorithm:

1. For each tree, select a bootstrap sample of observations.
2. Fit the individual decision tree, using recursive partitioning, as follows:
    - Select a random set of predictors for each split.
    - Continue splitting until a stopping rule that is specified in the Bootstrap Forest Specification window is met.

3. Repeat step 1 and step 2 until the number of trees specified in the Bootstrap Forest Specification window is reached or until Early Stopping occurs.

Specification Window:



Bootstrap Forest Specification

Number of Rows: 7382
Number of Terms: 14

Forest
Number of Trees in the Forest 300
Number of Terms Sampled per Split: 3
Bootstrap Sample Rate 1
Minimum Splits per Tree: 10
Maximum Splits per Tree 2000
Minimum Size Split: 7
[✓] Early Stopping

Multiple Fits
[ ] Multiple Fits over Number of Terms
Max Number of Terms 7
[ ] Use Tuning Design Table

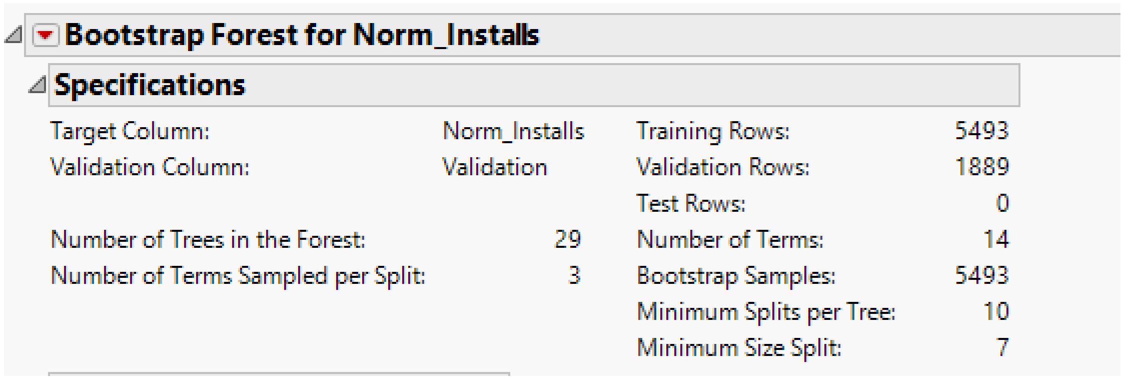
Reproducibility
[ ] Suppress Multithreading
Random Seed 0

Panels	**Description**	**Google Play Store Model Value**
Number of Rows	The number of rows in the data table	**7382**
Number of Terms	The number of columns that are specified as predictors	**14**
Number of Trees in the Forest	Number of trees to grow and then average	**300**
Number of Terms in the Forest	Number of predictors to consider as splitting candidates at each split. For each split, a new random sample of predictors is taken as the candidate set.	**3**
Bootstrap Sample Rate	Proportion of observations to sample (with replacement) for growing each tree. A new random sample is generated for each tree.	**1**
Minimum Splits Per Tree	Minimum number of splits for	**10**

	each tree.	
Maximum Splits Per Tree	Maximum number of splits for each tree.	**2000**
Minimum Split Size	Minimum number of observations needed on a candidate split.	**7**

We have taken into consideration validation column, hence, **early stopping** option was used. If selected, the process stops growing additional trees if the additional trees do not improve the validation statistic. In our model's case, RSquare is the validation statistic.

Specifications after running the model:


Bootstrap Forest for Norm_Installs

Specifications

Target Column:	Norm_Installs	Training Rows:	5493
Validation Column:	Validation	Validation Rows:	1889
		Test Rows:	0
Number of Trees in the Forest:	29	Number of Terms:	14
Number of Terms Sampled per Split:	3	Bootstrap Samples:	5493
		Minimum Splits per Tree:	10
		Minimum Size Split:	7

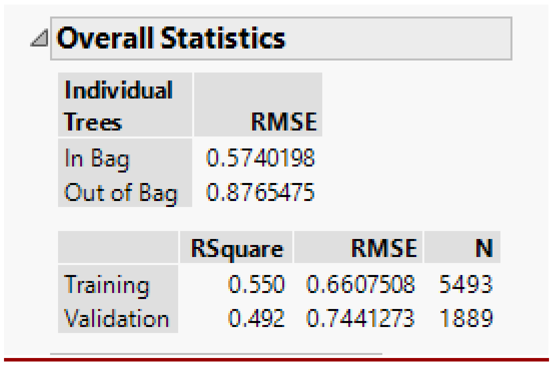
The above table shows the settings used in fitting the model.

Number of Trees in the Forest: 29

Why only 29?

It is because it checked the RSquare Validation for each tree and found it to be maximum at 29th tree. After reaching 29th tree, it further checked RSquare value for 10-20 tress and the values were not found to be greater than RSquare Validation value for 29th tree, hence, it pruned back to the tree having highest RSquare validation value, which in our model is 29 trees.

Overall Statistics:



Overall Statistics

Individual Trees	RMSE
In Bag	0.5740198
Out of Bag	0.8765475

	RSquare	RMSE	N
Training	0.550	0.6607508	5493
Validation	0.492	0.7441273	1889

Individual Tree Report:

The Overall Statistics table gives RMSE values, which are averaged over all trees, for In Bag and Out of Bag observations.

- Training set observations that are used to construct a tree are called **In-Bag** observations. RMSE for In Bag observations is 57%.
- Training observations that are not used to construct a tree are called **Out-of-Bag** (OOB) observations. RMSE for Out of Bag observations is 88%.

RSquare and RMSE Report:

The above table gives RSquare, root mean square error, and the number of observations for the training set, and for the validation and test sets, if they are defined.

The comparison of the training and validation RSquare for Bootstrap Forest shows that the decrease is only 12.8% ((0.560-0.488)/0.560 = 0.128).

The comparison of the training and validation RMSE for Bootstrap Forest shows that the increase is only 14.3% ((0.747064-0.653314)/0.653314 = 0.143)

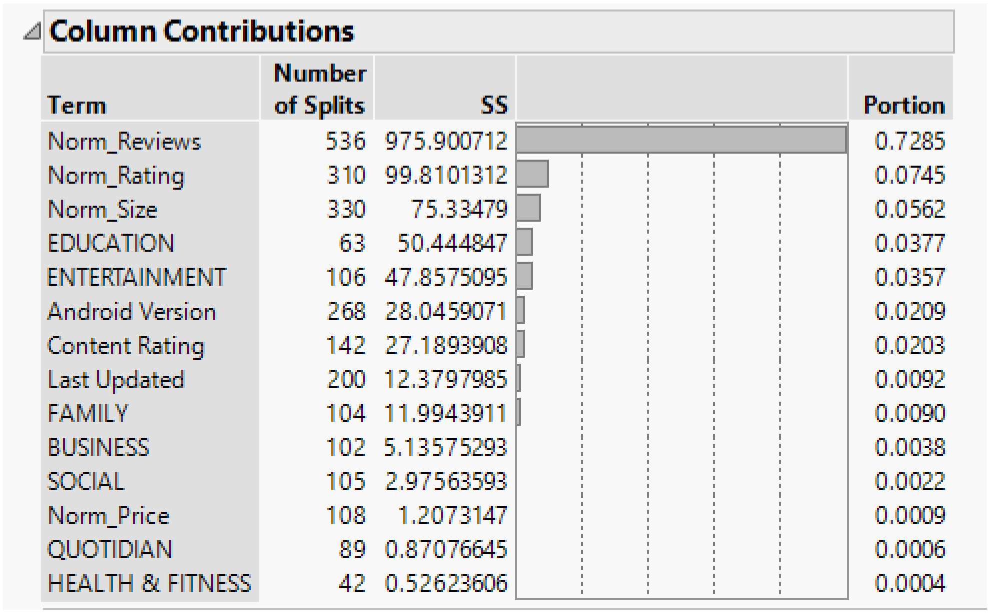
So, considering these two measurements, the Bootstrap Forest fits well into the validation set, and there is no overfitting issue. It fits well for the randomly picked observation.

Column Contribution:

We observe from the column contributions the predictors that have been used in the regression tree. SS(Sum of Squared Deviation) statistic provides a measure of the difference or dissimilarity in the proportions at the cut point for each variable. Portion shows the amount of contribution that each of the predictors have made in growing the regression tree.

We can see that the Norm_reviews predictor has the highest contribution to the Number of Installs of the App

The predictors Norm_Reviews, Norm_Rating, Norm_Size, Education, Entertainment, Android Version, Content rating, Last Updated, Family have participated in growing the tree. The predictor '**Norm_Reviews**' has contributed for 72.85%(by participating in 536 splits) of all the contributions made by predictors followed by Norm_Rating, Norm_Size and so on. We can say that Last Updated and Family has the least contribution in growing the trees with 0.90% and 0.92% contribution respectively.



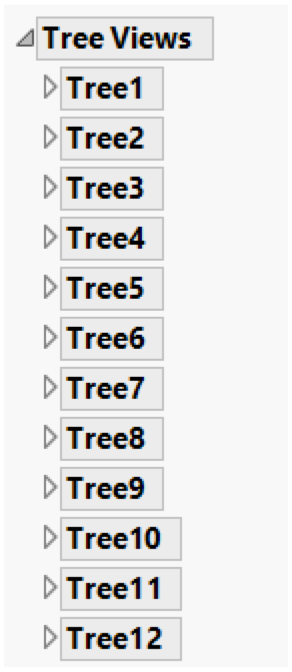
Column Contributions

Term	Number of Splits	SS	Portion
Norm_Reviews	536	975.900712	0.7285
Norm_Rating	310	99.8101312	0.0745
Norm_Size	330	75.33479	0.0562
EDUCATION	63	50.444847	0.0377
ENTERTAINMENT	106	47.8575095	0.0357
Android Version	268	28.0459071	0.0209
Content Rating	142	27.1893908	0.0203
Last Updated	200	12.3797985	0.0092
FAMILY	104	11.9943911	0.0090
BUSINESS	102	5.13575293	0.0038
SOCIAL	105	2.97563593	0.0022
Norm_Price	108	1.2073147	0.0009
QUOTIDIAN	89	0.87076645	0.0006
HEALTH & FITNESS	42	0.52623606	0.0004

Per-Tree Summaries:

The Per-Tree Summaries report involves the concepts of in-bag and out-of-bag observations. For an individual tree, the bootstrap sample of observations used in fitting the tree is drawn with replacement. Even if you specify that 100% of the observations are to be sampled, because they are drawn with replacement, the expected proportion of unused observations is 1/e. For each individual tree, the unused observations are called the out-of-bag observations. The observations used in fitting the tree are called in-bag observations.

Below is the screenshot capturing only 12 trees, but our model has 29 trees.


Tree Views
Tree1
Tree2
Tree3
Tree4
Tree5
Tree6
Tree7
Tree8
Tree9
Tree10
Tree11
Tree12

2e) NEURAL NETWORK

Neural network technology mimics the brain's own problem-solving process. Just as humans apply knowledge gained from past experience to new problems or situations, a neural network takes previously solved examples to build a system of "neurons" that makes new decisions, classifications, and forecasts.

In its simplest form, the layers of neurons at either end make the main inputs and outputs, whereas all in the middle are linked such that their outputs become the input for the subsequent layer.

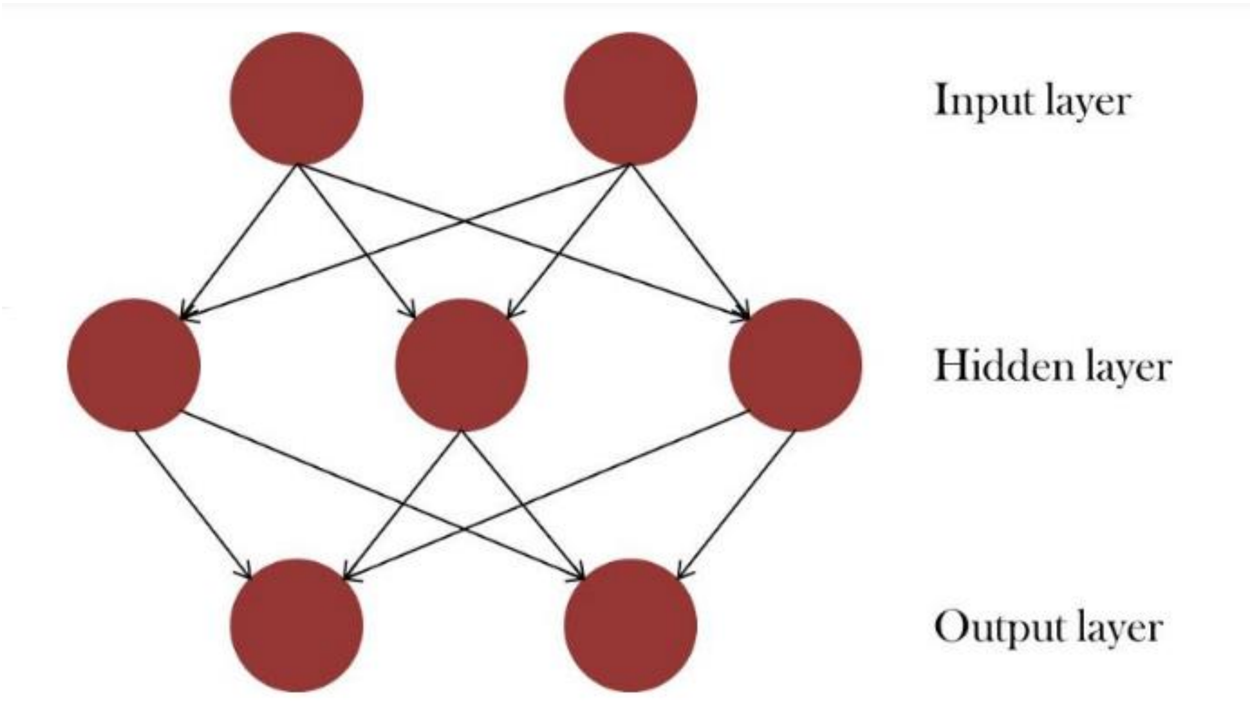


Figure 1: Illustration of a simple neural network

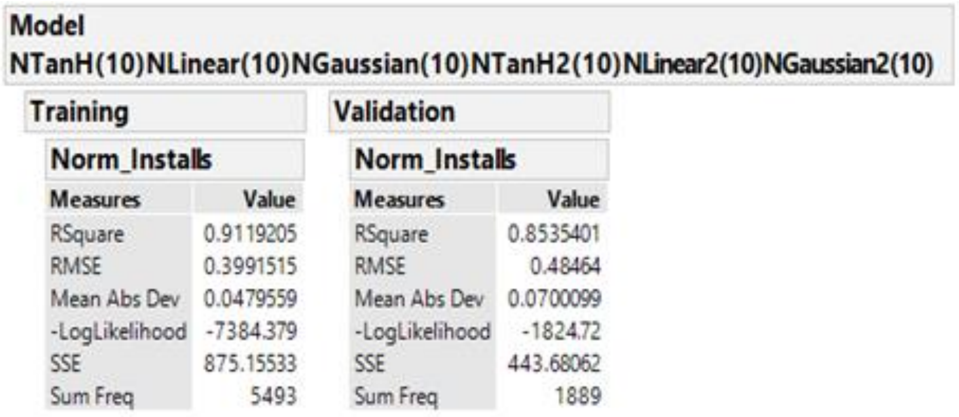
Fit Details

The very last model that we implemented was neural networks. Neural networks work well for both classification and prediction. We used neural models to predict *"Installs"(*Number of

Installs). We ran them four times varying parameter and found that the best performance is attained when using 10 nodes for the hidden layer for each activation function TanH, Linear and Gaussian.

We observe that we have lower RMSE values for both training and validation with increased RSquare values for both training and validation data. This lower RMSE for validation data tells us that the neural network performs the best as compared to other models. Even, we can see that the RSquare values for training(.8535401) and validation(.9119205) do not have a significant difference with for training and for validation which hints that our model is not suffering from over-fitting.

- The training and validation results have very high *RSquare*. The validation *RSquare* is only slightly lower than the *RSquare* for the training dataset ((0.9119205-0.8535401)/0.9119205 = 6 .4%).
- The *RMSE* for the validation dataset 21.26% higher (.48464-.3991515)/.3991515) than that of the training data.


Model
NTanH(10)NLinear(10)NGaussian(10)NTanH2(10)NLinear2(10)NGaussian2(10)

Training

Norm_Installs

Measures	Value
RSquare	0.9119205
RMSE	0.3991515
Mean Abs Dev	0.0479559
-LogLikelihood	-7384.379
SSE	875.15533
Sum Freq	5493

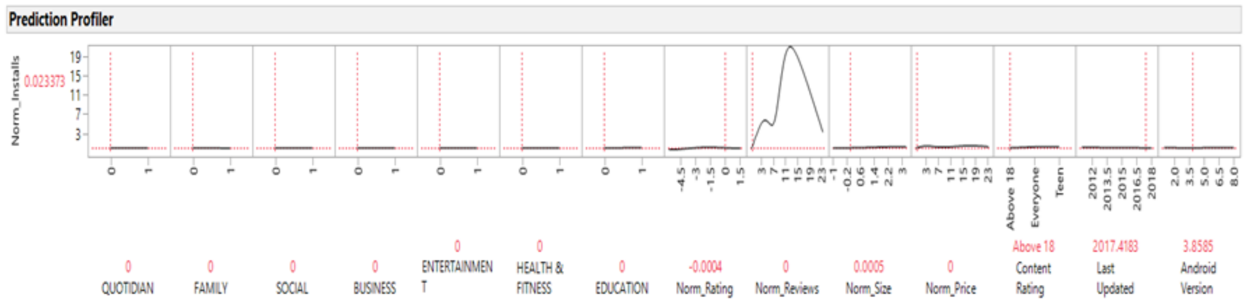
Validation

Norm_Installs

Measures	Value
RSquare	0.8535401
RMSE	0.48464
Mean Abs Dev	0.0700099
-LogLikelihood	-1824.72
SSE	443.68062
Sum Freq	1889

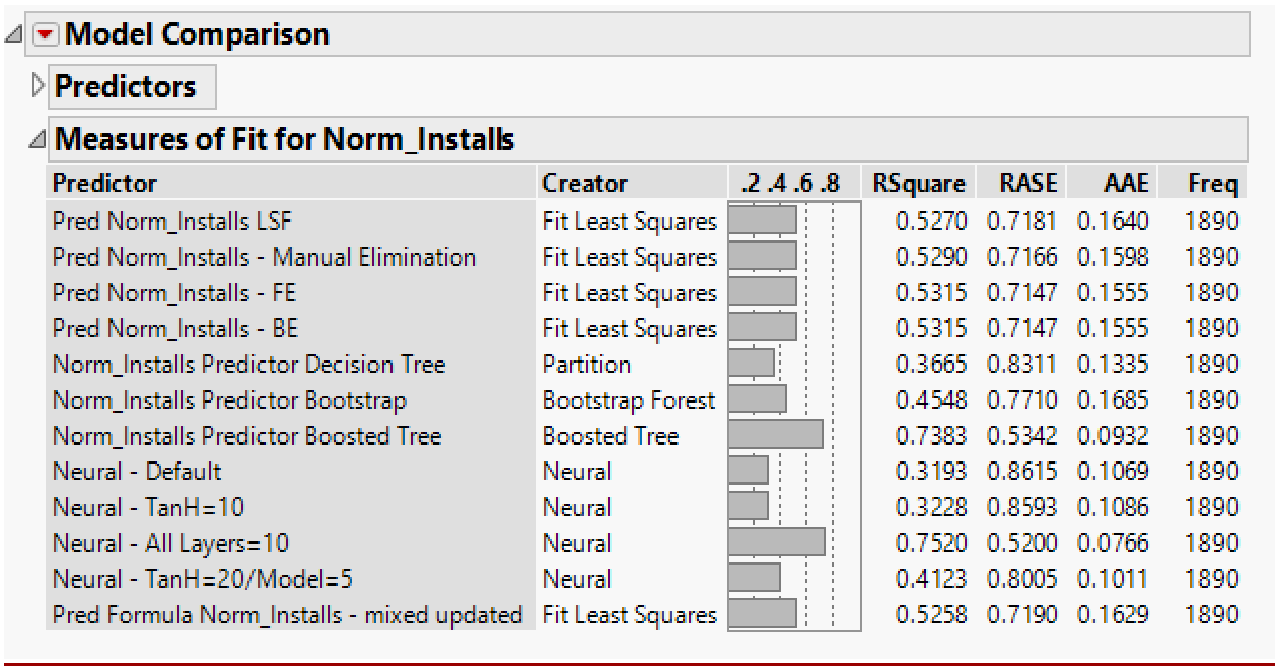
Profiler:

The Prediction Profiler of the input predictors show that input predictor "*Reviews*" have a non-linear relationship with "*installs*". However, all other input predictors have nearly no relationship with the *"installs"*.



Chapter 3: MODEL COMPARISON

- Based on the Model Comparison option, we compared the RSquare and RASE, of all the models.
- We have 2 models with highest RSquare and lowest RASE.
- Boosted tree with RSquare= 0.7383 and RASE=0.5342
- The Neural Networks model with 10 nodes for all the functions in both the layers has the second highest RSquare(0.7520) and relatively low RASE(0.5200).
- Since there is an overfitting issue in the Boosted tree, we select Neural networks model as the best model.



Model Comparison

Predictors

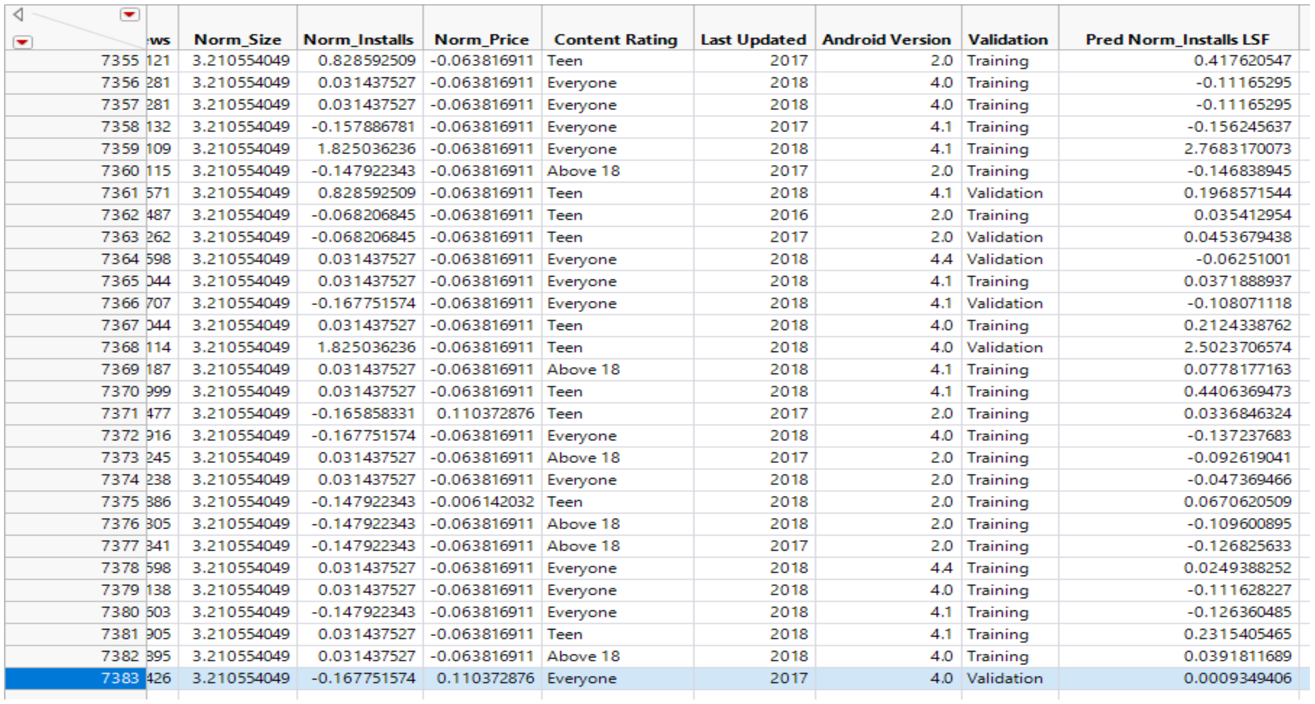
Measures of Fit for Norm_Installs

Predictor	Creator	.2 .4 .6 .8	RSquare	RASE	AAE	Freq
Pred Norm_Installs LSF	Fit Least Squares		0.5270	0.7181	0.1640	1890
Pred Norm_Installs - Manual Elimination	Fit Least Squares		0.5290	0.7166	0.1598	1890
Pred Norm_Installs - FE	Fit Least Squares		0.5315	0.7147	0.1555	1890
Pred Norm_Installs - BE	Fit Least Squares		0.5315	0.7147	0.1555	1890
Norm_Installs Predictor Decision Tree	Partition		0.3665	0.8311	0.1335	1890
Norm_Installs Predictor Bootstrap	Bootstrap Forest		0.4548	0.7710	0.1685	1890
Norm_Installs Predictor Boosted Tree	Boosted Tree		0.7383	0.5342	0.0932	1890
Neural - Default	Neural		0.3193	0.8615	0.1069	1890
Neural - TanH=10	Neural		0.3228	0.8593	0.1086	1890
Neural - All Layers=10	Neural		0.7520	0.5200	0.0766	1890
Neural - TanH=20/Model=5	Neural		0.4123	0.8005	0.1011	1890
Pred Formula Norm_Installs - mixed updated	Fit Least Squares		0.5258	0.7190	0.1629	1890

WORKING EXAMPLE

- These results can be considered as a threshold for number of downloads. Companies can track the progress with the help of these parameters and use insights from these accordingly.

- Reviews are also important in terms of they allow your user base to give constructive feedback which will allow you to improve or add features that may get you even more downloads.



	ws	Norm_Size	Norm_Installs	Norm_Price	Content Rating	Last Updated	Android Version	Validation	Pred Norm_Installs LSF
7355	121	3.210554049	0.828592509	-0.063816911	Teen	2017	2.0	Training	0.417620547
7356	281	3.210554049	0.031437527	-0.063816911	Everyone	2018	4.0	Training	-0.11165295
7357	281	3.210554049	0.031437527	-0.063816911	Everyone	2018	4.0	Training	-0.11165295
7358	132	3.210554049	-0.157886781	-0.063816911	Everyone	2017	4.1	Training	-0.156245637
7359	109	3.210554049	1.825036236	-0.063816911	Everyone	2018	4.1	Training	2.7683170073
7360	115	3.210554049	-0.147922343	-0.063816911	Above 18	2017	2.0	Training	-0.146838945
7361	571	3.210554049	0.828592509	-0.063816911	Teen	2018	4.1	Validation	0.1968571544
7362	487	3.210554049	-0.068206845	-0.063816911	Teen	2016	2.0	Training	0.035412954
7363	262	3.210554049	-0.068206845	-0.063816911	Teen	2017	2.0	Validation	0.0453679438
7364	598	3.210554049	0.031437527	-0.063816911	Everyone	2018	4.4	Validation	-0.06251001
7365	044	3.210554049	0.031437527	-0.063816911	Everyone	2018	4.1	Training	0.0371888937
7366	707	3.210554049	-0.167751574	-0.063816911	Everyone	2018	4.1	Validation	-0.108071118
7367	044	3.210554049	0.031437527	-0.063816911	Teen	2018	4.0	Training	0.2124338762
7368	114	3.210554049	1.825036236	-0.063816911	Teen	2018	4.0	Validation	2.5023706574
7369	187	3.210554049	0.031437527	-0.063816911	Above 18	2018	4.1	Training	0.0778177163
7370	999	3.210554049	0.031437527	-0.063816911	Teen	2018	4.1	Training	0.4406369473
7371	477	3.210554049	-0.165858331	0.110372876	Teen	2017	2.0	Training	0.0336846324
7372	916	3.210554049	-0.167751574	-0.063816911	Everyone	2018	4.0	Training	-0.137237683
7373	245	3.210554049	0.031437527	-0.063816911	Above 18	2017	2.0	Training	-0.092619041
7374	238	3.210554049	0.031437527	-0.063816911	Everyone	2018	2.0	Training	-0.047369466
7375	386	3.210554049	-0.147922343	-0.006142032	Teen	2018	2.0	Training	0.0670620509
7376	305	3.210554049	-0.147922343	-0.063816911	Above 18	2018	2.0	Training	-0.109600895
7377	341	3.210554049	-0.147922343	-0.063816911	Above 18	2017	2.0	Training	-0.126825633
7378	598	3.210554049	0.031437527	-0.063816911	Everyone	2018	4.4	Training	0.0249388252
7379	138	3.210554049	0.031437527	-0.063816911	Everyone	2018	4.0	Training	-0.111628227
7380	503	3.210554049	-0.147922343	-0.063816911	Everyone	2018	4.1	Training	-0.126360485
7381	905	3.210554049	0.031437527	-0.063816911	Teen	2018	4.1	Training	0.2315405465
7382	395	3.210554049	0.031437527	-0.063816911	Above 18	2018	4.0	Training	0.0391811689
7383	426	3.210554049	-0.167751574	0.110372876	Everyone	2017	4.0	Validation	0.0009349406

CONCLUSION

After understanding our data and exploring the various predictors involved to predict Installs, we have arrived at the conclusion that the Neural Network Model has the best performance upon training on the Google Play Store dataset. Moreover, we recommend that this model be used because it has the lowest RMSE value and the highest RSquare value for the validation data as compared to all other models. Both statistical measures, which are useful for comparison of performance of machine learning models, both indicate that the neural network has the best performance.

Throughout our analysis, we tried various models to understand which factors contribute in number of installs and it seems that number of reviews has the most influence and in many cases the only factor contributing in predictions. Hence it can be said that when it comes to implementing strategies to increase number of installs of any mobile application , companies should focus on getting reviews from user base in order to be successful. More than often your users will share any problems they have or suggestions in a form of review instead of contacting you via your developer email. The predictions of number of installs from this model can be used as a threshold for tracking the installs i.e if an app has certain number of reviews and the corresponding installs are not as predicted, the company can investigate the reason for that and work on their strategy accordingly. From all the models used in analysis, Neural Network with 2 Hidden layers and all the activation functions used, provides the most precise results and hence we would recommend that for predictions.

BIBILIOGRAPHY

Infographics : **https://www.business2community.com/infographics/mobile-app-development-2018-infographic-02017135**

Kaggle:

https://www.kaggle.com/tanetboss/how-to-get-high-rating-on-play-store/data