ANALYSIS OF APPS IN THE GOOGLE APP STORE

Siddharth Singh
17CO146
9980690119
siddharthsingh.171co146@nitk.edu.in

Chaitany Pandiya
17CO112
9834708844
chaitany.171co112@nitk.edu.in

Abbreviations & General Info

SS-Siddharth Singh
CP-Chaitany Pandiya
C- Common work done by both Siddharth and Chaitany
SVR-Support Vector Regressor
ML-Machine Learning
MSE-MEan squared Error
MAE-Mean absolute Error
MSLE-Mean squared log Error

This paper has been co-authored by two people SS(Siddharth Singh) and CP(Chaitany Pandiya). The paper follows a name terminology which means the part following SS is authored by Siddharth Singh on basis of his observations and implementation and the part authored by Chaitany Pandiya is preceded by CP which shows his observations and implementation.In general, the parts of the paper dealing with Installs and Machine Learning are authored by SS and part dealing with Rating and Machine Learning are authored by CP.

Abstract

Android operating system is one of the most popular operating system in the world. As a result, Android apps have flooded the smartphone application market. There are over **2 million** apps in the Google App (Play) store. This multitude of apps on the app store is largely a result of various organizations introducing proprietary apps for better user engagement and involvement. The phenomenon has resulted in intense competition among such organizations and corporations so as to better their apps and increase their number of installs. Any Android app is characterized by a number of attributes like the number of reviews, rating on the app store, price, category, genre, version, size, content rating, etc. These attributes have a combined impact on the number of installs that an app gets. Since all organizations wish to maximize the number of installs for their apps, it is worthwhile to know the optimal value of these characteristics which ensures maximal app installations. In particular, what is of interest is the task that given such parameters of an app can we roughly estimate the number of installs it might get. This is where the presence of a large number of such apps proves to be a boon. We can effectively do statistical analysis and develop and train machine learning models on the data pertaining to such apps. Thus the above-proposed prediction is now possible thanks to a large and diverse dataset together with appropriate Machine Learning algorithms to act on this dataset. This paper tries to deal with this problem and present a novel analysis using machine learning and deep learning, that given a set of parameters, what ratings and installs can an application get realistically with as low error as

possible. This we will try to accomplish by feeding into many classifiers of which major we will try to look at why Linear Regression, SVR, Random Forests still work decently to give us a decent accuracy. The first part of paper mostly deals with the insights from the datasets and how regular statistics also give us a good idea on the dataset. The performance of the proposed methodology is compared against baselines that utilize varying approaches to validate its utility. The results are finally summarized by reflecting on the effect of inclusion/exclusion of certain features and how they affect the results. We will construct and train an Artificial Neural Network(ANN) on the dataset of more than 7000 apps so as to get a good insight into the Android App store.

Keywords- Artificial Neural Network(ANN), Google App Store, Android Operating System

Introduction

Machine Learning (ML) is defined as the use of computational statistics and algorithms to learn from data without being explicitly programmed. It comes under artificial intelligence domain within computer science. The field came into prominence recently. This became possible due to the abundance of data, faster computers, and more efficient data storage. The very abundance of data today requires efficient data storage techniques and faster computers to enable us to draw meaningful insights from it. Also, most machine learning algorithms are compute intensive and are often characterized by large training times. Thus the advancement in technology has enabled us to faster process the ever-increasing data. However, the results produced by the machine learning algorithms will only be as good as the quality of data that is fed into them irrespective of the power and performance of the computers processing them. To sum up, the two things that matter a lot in machine learning are a large quantity of data and a good quality of that data. In my analysis of apps on the Google App store, We were able to obtain a large dataset of apps' data, thanks to the presence of a multitude of apps on the Play Store. As far as reliability of apps is concerned, the data cannot be more accurate if it is obtained by web crawling of the Google Play Store itself. The problem of predicting the number of installs of an app based on parameters like price, number of reviews, rating on the app store, category, etc. is a typical regression problem. As any regression problem requires, We have done a graphical analysis of these parameters versus the number of installs so as to spot any striking relationships prior to ML model construction and training. Such an analysis not only gives some initial insight but also helps to identify outliers in the dataset. Most machine learning algorithms are sensitive to the distribution

and range of parameter values in the input data. Outliers in input(training) data can mislead and skew the training process of ML algorithms resulting in significantly longer training times, less accurate models and ultimately poorer results. The ultimate objective is to discover the best deployment of ML algorithms to predict the number of installs of a given Android app based on the above-mentioned parameters. The results of a successful deployment of a machine learning model, trained on the apps' data can be of use to any individual or organization which is rolling out their app in the market and wish to estimate the number of installs. This prediction of the number of installs can help them to allocate resources more efficiently. The allocation of resources can range from setting up servers to cater to users' needs to hiring manpower for the same If we can gauge the load that can be expected beforehand we can be better prepared to balance our load. Eventually, this can lead to better planning and execution of the desired goals.

Approaches and Methods

1. Data Preparation

We now start discussing how do we approach the dataset and try to find out some basic statistics from the application data. This would have been pretty simple if the dataset was free from flaws, but unfortunately, this is not the case.

We start by analysing the dataset properly, the first thing to be done is to load the dataset into something that would make analysing it much easier. For this paper, we have used pandas library and the programming language used is Python. By quickly running a built-in function **df.info()** we get to know the general details of the dataset

The figure below gives a glance of the first few rows of the dataset.

```
dataset=pd.read_csv("../input/googleplaystore.csv")
dataset.head()
```

	App	Category	Rating	Reviews	Size	Installs	Type	Price	Content Rating	Genres	Last Updated	Current Ver	Androi Ver
0	Photo Editor & Candy Camera & Grid & ScrapBook	ART_AND_DESIGN	4.1	159	19M	10,000+	Free	0	Everyone	Art & Design	January 7, 2018	1.0.0	4.0.3 and up
1	Coloring book moana	ART_AND_DESIGN	3.9	967	14M	500,000+	Free	0	Everyone	Art & Design;Pretend Play	January 15, 2018	2.0.0	4.0.3 and up
2	U Launcher Lite – FREE Live Cool Themes, Hide ...	ART_AND_DESIGN	4.7	87510	8.7M	5,000,000+	Free	0	Everyone	Art & Design	August 1, 2018	1.2.4	4.0.3 and up
3	Sketch - Draw & Paint	ART_AND_DESIGN	4.5	215644	25M	50,000,000+	Free	0	Teen	Art & Design	June 8, 2018	Varies with device	4.2 an up
4	Pixel Draw - Number Art Coloring Book	ART_AND_DESIGN	4.3	967	2.8M	100,000+	Free	0	Everyone	Art & Design;Creativity	June 20, 2018	1.1	4.4 an up

Fig (1)

(SS) To begin with, some of the columns had missing entries. Such values are displayed as NaN in the data frame. Most of the missing entries were found in the ratings column. For removing these missing values We used **dropna**() method of the Pandas library. This method drops the Rows/Columns with null values in different ways. After this step, the dataset was left with 9360 apps' data.

```
dataset.dropna(how='any',inplace=True)
dataset.shape
```

```
(9360, 13)
```

Machine learning algorithms work better on numbers than they do on text, and we observe that most of the columns have the data type as object which means we cannot process it effectively, so we try to make each column to its suitable data type to preprocess the data.From the categorical column, we converted each category into an individual number. This will help in the implementation part as, when we do apply machine learning, two methods will be applied to the code, being integer encoding(which we are doing now) and one-hot encoding, aka dummy variables.
The main reason as to why I understand we do this transformation is mainly because integer encoding relies on the fact that there's a relationship between each category(e.g. think age range vs types of animals). In this case, however, it's hard to really determine such a relationship, hence dummy/one-hot encoding might help provide better predictive accuracy.

(SS) Next, it can be seen in Fig(1) that the installs column has values appended with a '+' to signify that the number of installs is more than the numerical value but still close to it. However, this notation is not suitable for feeding to machine learning models. So, for my next step I removed the '+' character from the Installs column.Also, the commas in the entries of installs column had to be removed. The price column had some values with '$' appended to them. This again had to be removed and their datatype converted to Float. The size column had some entries in KB with the character 'K' appended to the and some entries in MB with the character 'M' appended to them. I dropped these characters and converted the entries into bytes for uniformity. Some 'Varies with device' entries in the size column were also dropped.

(CP) We also convert Free and Paid to integers where 0 stands for 'Free' and 1 stands for 'Paid'. We also convert the content rating to a unique integer. We drop the 'Last Updated','Current Ver', 'Android Ver', 'App' from our dataframe as these will not make any useful contribution in influencing the results.I also made a new column 'Category_c' which contains integer encoding of the categories ranging from 0 to 32.As the number of genres is substantial(115) we won't do dummy encoding here, as it will lead to a huge increase in independent variables instead, I will just use integer encoding and name it as Genres_c. We would do two separate regressions, one including and one excluding such genre data. When

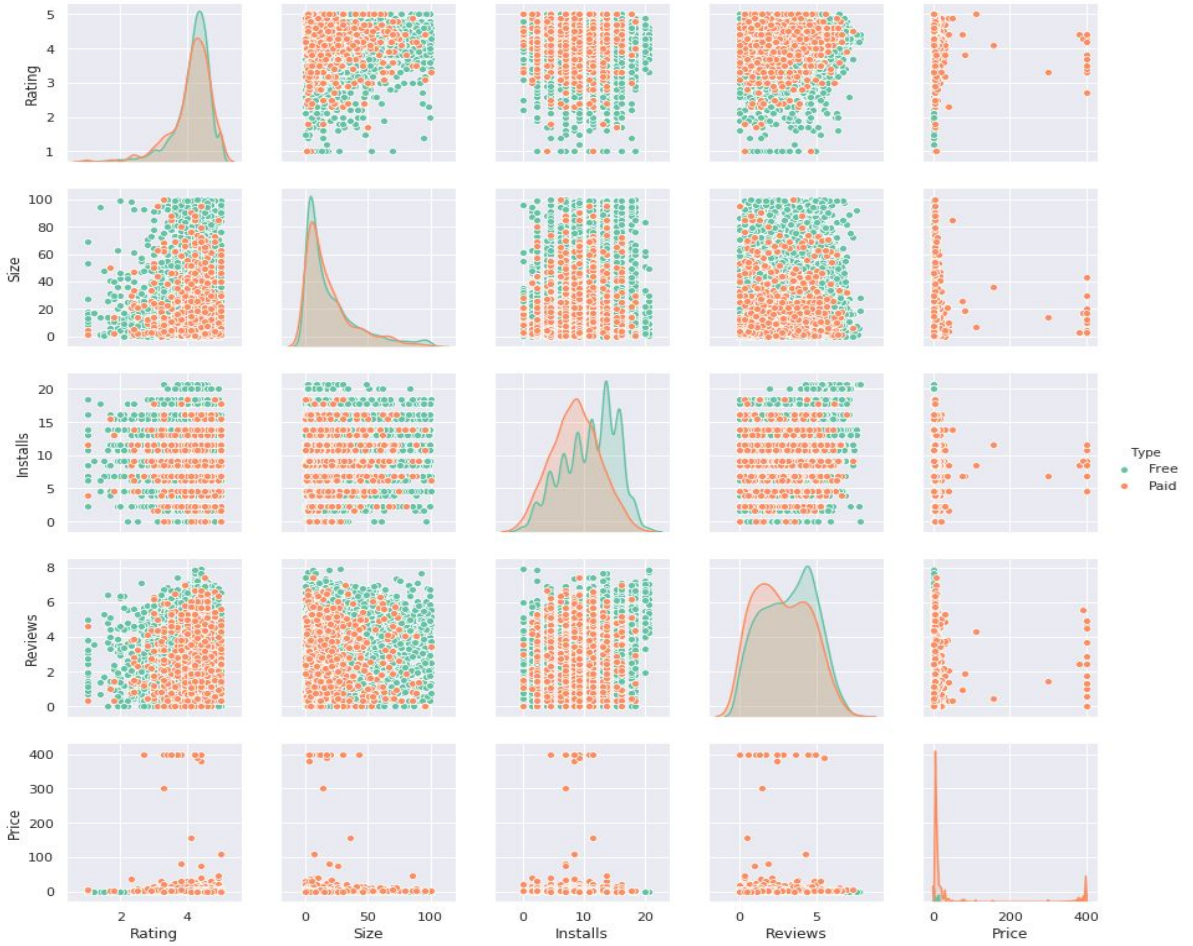
including the data, we only considered in the impact/information provided via the genre section purely based on its numeric value.I also converted the columns to their correct data type.

	App	Category	Rating	Reviews	Size	Installs	Type	Price	Content Rating	Genres	Last Updated	Current Ver	Android Ver
0	Photo Editor & Candy Camera & Grid & ScrapBook	ART_AND_DESIGN	4.1	159	19000000.0	10000	Free	0.0	Everyone	Art & Design	January 7, 2018	1.0.0	4.0.3 and up
1	Coloring book moana	ART_AND_DESIGN	3.9	967	14000000.0	500000	Free	0.0	Everyone	Art & Design;Pretend Play	January 15, 2018	2.0.0	4.0.3 and up
2	U Launcher Lite – FREE Live Cool Themes, Hide ...	ART_AND_DESIGN	4.7	87510	8700000.0	5000000	Free	0.0	Everyone	Art & Design	August 1, 2018	1.2.4	4.0.3 and up
3	Sketch - Draw & Paint	ART_AND_DESIGN	4.5	215644	25000000.0	50000000	Free	0.0	Teen	Art & Design	June 8, 2018	Varies with device	4.2 and up
4	Pixel Draw - Number Art Coloring Book	ART_AND_DESIGN	4.3	967	2800000.0	100000	Free	0.0	Everyone	Art & Design;Creativity	June 20, 2018	1.1	4.4 and up

Fig (2) shows the dataset after applying the above-mentioned data preparation steps.

(SS) Label Encoding and One hot Encoding- Since the entries in the Category column are of type String, they cannot be effectively fed to the machine learning models. For that reason, I decided to Encode them into numerical values. The two approaches available to me were Label Encoding and One hot Encoding. Label Encoding substitutes the unique entries with numbers from 0 to number of unique entries in ascending order as and when they are encountered. However, this encoding poses a problem. The problem with the first approach is that, since there are different numbers in the same column, the model will misinterpret the data to be in some kind of order. But this definitely is not the case. To get around this problem, I used One Hot Encoding . This approach takes a column which has categorical data, which has been label encoded, and then splits the column into multiple columns concatenated together. The numbers are replaced by 1s and 0s, depending on which column has what value.
There were a total of 33 unique entries in the Category column. Thus, this step resulted in addition of 33 new columns to our dataset.

2. Explorative Data Analysis (EDA)



(SS) Exploratory Data Analysis refers to the process of performing initial investigations on data so as to discover certain striking patterns, to spot anomalies, to test hypothesis and to check assumptions with the help of statistics and graphical representations.It is recommended to understand the data first and try to gather significant insights from it. EDA is all about making sense of data in hand, before we start deploying learning algorithms on them.

I plotted various columns of the dataset using seaborn to visualize their dependencies of on each other. The figure above shows the pair plot between Installs, Size, Rating, Reviews and Price. After visualizing the dependencies between various traits, I started exploring the correlations between them. I came across an interesting correlation between Number of Installs and Number of Reviews when these values are log scaled (Base 10). This correlation is best represented by the graph below.

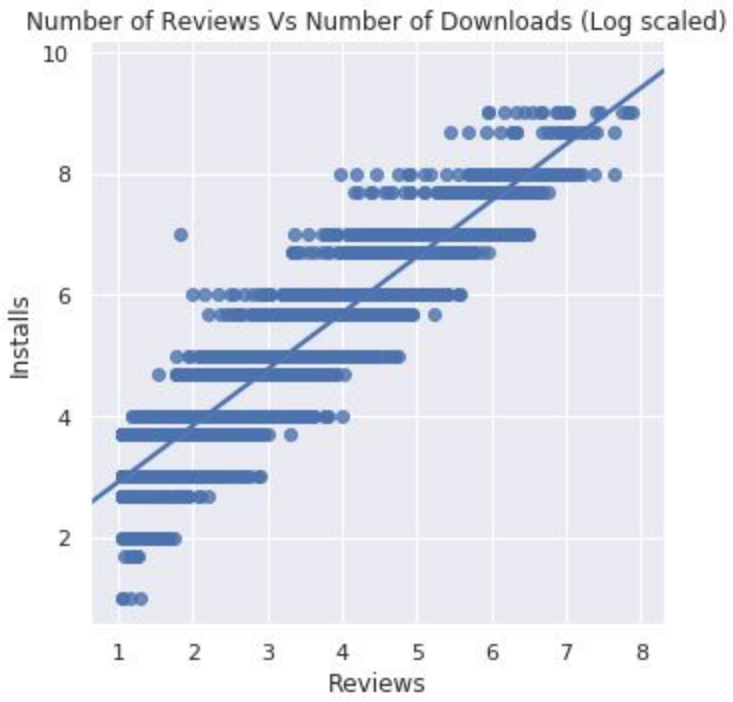


Fig (3)

3. Feature Engineering and Feature scaling

(SS) Feature engineering is the process of using **domain knowledge** of the data to create features that make machine learning algorithms work. If feature engineering is done correctly, it increases the predictive power of machine learning algorithms by creating features from raw data that help facilitate the machine learning process.

Feature scaling helps to normalise the data within a particular range.It is a step of Data Pre Processing which is applied to independent variables or features of data.

The observations from Fig(3) prompted me to look for relationships between quantities when parameters with high variance such as number if installs and number of reviews are log scaled(Base 10). Thus I converted the values in columns ‘Installs’ and ‘Reviews’ to their logarithms base 10. This was done to improve the predictive power of the machine learning algorithms on the given problem statement. This log scaling helps to minimise the variance in general.

Based on the EDA performed by me, I have decided to go ahead and train my ML models to predict log (Base 10) of **the number of installs** based on the **price**, **rating** and **log(Base 10) of the number of reviews.**

(CP) After our final checks for the preprocessing of our data, we can now start doing some analysis on it, so the next question is what exactly are we doing and how are we doing it.
So the goal of this instance is to see if we can use existing data provided(e.g. Size, no of reviews) to predict the ratings of the Google applications. **In other words, our dependent variable Y would be the rating of the apps.** One important factor to note is that the dependent variable Y, is a continuous variable(aka infinite no of combinations), as compared to a discrete variable. Naturally, there are ways to convert our Y to a discrete variable but **I decided to keep Y as a continuous variable** for the purposes of this machine learning session.

3. Train-Test split

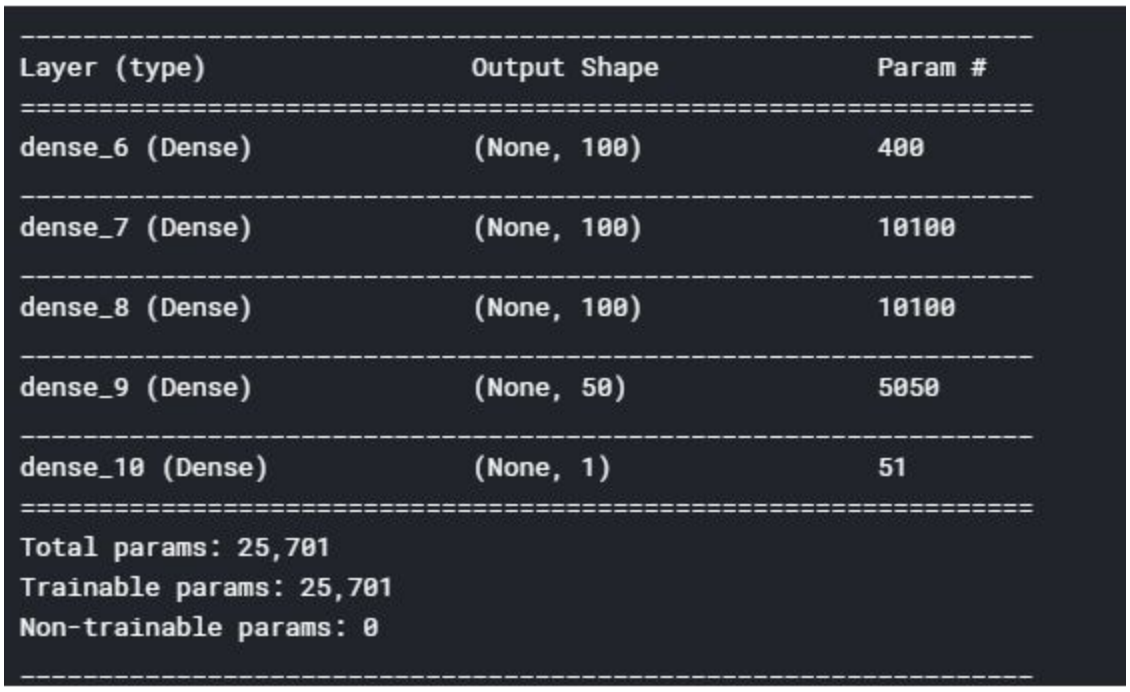
(C) We have split the entire dataset into Training and Testing datasets such that 20 per cent of entire dataset is for testing the performance of the Machine Learning models. Testing data is kept separate while training so that overfitting, if present, can be detected during testing. Also, ultimately our models should perform decently on unseen data. So, it makes sense to keep a certain section of data separate from the training dataset.

4. Machine Learning model selection

(SS) The below mentioned approaches were tried by me for predicting the number of installs of a given Android app.
First Approach: Multiple Linear Regression
Multiple Linear Regression: Multiple linear regression attempts to model the relationship between two or more explanatory variables and a response variable by fitting a linear equation to observed data. Every value of the independent variable x is associated with a value of the dependent variable y. In other words, multiple linear regression explains the relationship between **one continuous dependent variable** (y) and **two or more independent variables** ($x1, x2, x3\ldots$ etc)**.** The continuity of the dependent variable y, number of installs in this case, is ensured because we are taking logarithm of the entries. Since this problem specifically belongs to the class of regression problems, it is worthwhile to try multiple linear regression as the first approach. The only factor that might deem multiple linear regression unfit for this problem is the fact that there might not be the best fit line that predicts the number of installs based on the number of reviews, rating and price. This, if true can bring down the accuracy of our model. For this reason, We decided to with a second and more robust approach.

Second Approach: Artificial Neural Network(s)
In simple terms, an Artificial Neural Network is a mathematical model based on the biological neural network. A neural network has a set of connected neurons and each and every one of these

neurons have weights that can be adjusted during the learning phase. If we describe further on the learning phase of the neural network this is when the inputs are fed into the system and the initial weights assigned to the neuron connections are thus adjusted such that the final error given by the model is minimized.
We constructed a neural network with 4 hidden layers in addition to an input and an output layer. The input layer has 3 neurons followed by three hidden layers with 100 neurons each. Another hidden layer is present with 50 neurons followed by the output layer with a single neuron.



Layer (type)	Output Shape	Param #
dense_6 (Dense)	(None, 100)	400
dense_7 (Dense)	(None, 100)	10100
dense_8 (Dense)	(None, 100)	10100
dense_9 (Dense)	(None, 50)	5050
dense_10 (Dense)	(None, 1)	51

Total params: 25,701
Trainable params: 25,701
Non-trainable params: 0

Fig(4)

Fig(4) gives a summary of the neural network constructed by me. The large number of hidden layer neurons are introduced by me to take care of non-linearity as effectively as possible. Plots of non-linear functions are characterised by a curvature. Such functions have a degree more than one. The presence of such non-linear functions puts a practical limit to the performance of a Linear regression model since it tries to find the best fit line when there is no linear relationship.

Activation Function: This function decides whether a neuron is activated or not by calculating the weighted sum and adding a bias to it. Activation function introduces non-linearity into the output of a neuron. We have decided to use ReLU(Rectified Linear Unit) as my activation function.
ReLU is half rectified(from bottom).f(z) is zero when z is less than zero and f(z) is equal to z when z is above or equal to zero.
Range [0 to infinity)
The function and its derivative both are monotonic.
Optimizer

Optimization algorithms helps us to ***maximise (or minimise)***an **Objective** function (*another name for* ***Error*** *function*) **E(x)** which is simply a mathematical function dependent on the Model's internal **learnable parameters.** These parameters are used in computing the target values(**Y**) from the set of *predictors*(**X**) used in the model. For example — we call the **Weights(W)** and the **Bias(b)** values of the neural network as its internal learnable *parameters* which are used in computing the output values and are learned and updated in the direction of optimal solution i.e minimizing the **Loss** by the network's training process and also play a major role in the ***training*** process of the Neural Network Model .

We have decided to use Adam as my optimization function. Adam optimization function is an extension to stochastic gradient descent.The benefits of using Adam are multifold. It is straightforward to implement, little memory requirements, computationally efficient, well suited for problems that are large in terms of data and/or parameters, Appropriate for non-stationary objectives, Appropriate for problems with very noisy/or sparse gradients, Invariant to diagonal rescale of the gradients, Hyper-parameters have intuitive interpretation and typically require little tuning. Also, Empirical results demonstrate that Adam works well in practice and compares favorably to other stochastic optimization methods. As of today the two recommended optimizers to use are *SGD+Nesterov Momentum or Adam.*

Epochs: One Epoch is when an entire dataset is passed forward and backward through the neural network only once. A single epoch may result in underfitting. That is the reason why multiple epochs are advisable. However, we need to be a bit careful. The more the number of epochs, the more the number of times the weights are changed in the neural network and the curve goes from **underfitting** to **optimal** to **overfitting** curve.

We have decided to go ahead with 100 epochs for my neural network.

Loss function: Loss function is a method of evaluating how well the algorithm models the dataset. If the predictions are pretty good, it'll output a lower number.If they are totally off, then the loss function will output a higher number.

I am using MSE as my loss function. MSE loss, or MSE for short, is calculated as the average of the squared differences between the predicted and actual values.

(CP) Model-wise, I'm not too sure as well as there are like a ton of models out there that can be used for machine learning. Hence, I basically just chose the 3 most common models that I use, being linear regression, SVR, and random forest regressor, we will check how these perform on the dataset.
We technically run 4 regressions for each model used, as we consider one-hot vs integer encoded results for the category section, as well as including/excluding the genre section.

We then evaluate the models by comparing the predicted results against the actual results graphically, as well as use the MSE, MAE and MSLE as a possible benchmark. My first approach is using Linear Regression which is described above.

My second approach is using SVR

Support Vector Machine can also be used as a regression method, maintaining all the main features that characterize the algorithm (maximal margin). However, the main idea is always the same: to minimize error, individualizing the hyperplane which maximizes the margin, keeping in mind that part of the error is tolerated

My third approach is using Random Forest Regressor

The Random Forest is the most effective of all approaches discussed here.

Background
The random forest model is an ensemble model that makes predictions by combining decisions from a sequence of base models. More formally we can write this class of models as:

$h(x) = g0(x) + g1(x) + g2(x)+...$
Here h is the resultant model which is made after combining the sum of g(i) which are basic models. Here, each base classifier is a simple decision tree. This broad technique of using multiple models to obtain better predictive performance is called model ensembling. In random forests, all the base models are constructed independently using a different subsample of the data.

Why should we choose Random Forest Regressor?

Different kinds of models have different advantages. The random forest model is very good at handling tabular data with numerical features, or categorical features with fewer than hundreds

of categories. Unlike linear models, random forests are able to capture non-linear interaction between the features and the target.

The Random forest regressor is best out of all the previous model that we applied and hence this is reflected in the results also.

Results

(C) Some important insights we found are:

- The average app rating is 4.17.
- Family and Game apps have the highest Market prevalence.
- Almost all app categories perform decently. Health and Fitness and Books and Reference produce the highest quality apps with 50% apps having a rating greater than 4.5. This is extremely high!
- On the contrary, 50% of apps in the Dating category have a rating lesser than the average rating.
- A few junk apps also exist in the Lifestyle, Family and Finance category.
- Most top rated apps are optimally sized between 2MB to 40MB - neither too light nor too heavy.
- Most top rated apps are optimally priced between 1$ to 30$.

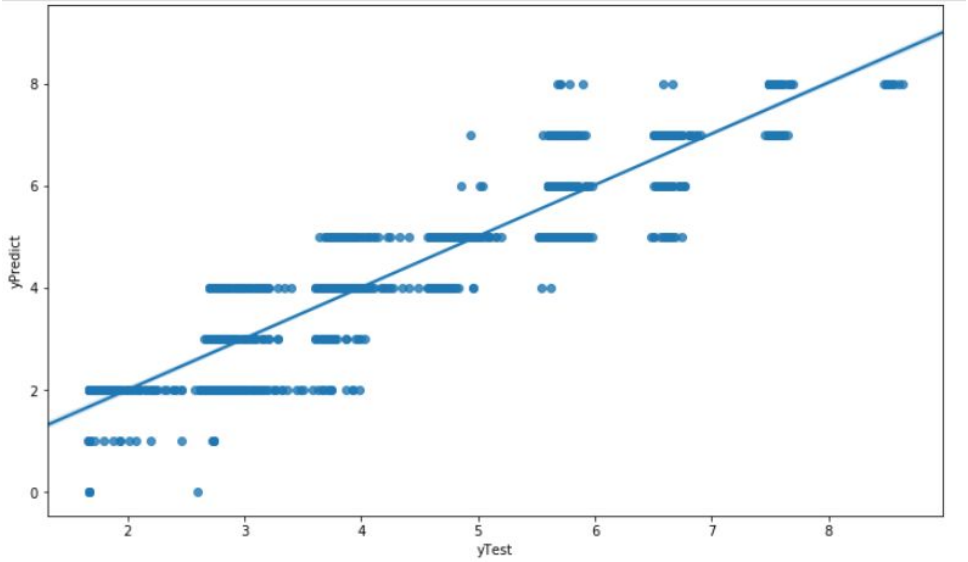
1. **(SS) Linear Regression for predicting the number of installs**
   Since Linear Regression tries to find the best fit line, trying to predict the number of installs based on a lot of parameters may result in poor accuracy. This is due to the possibility that the presence of a large number of parameters may introduce significant non-linearity. Based on the Explorative data analysis conducted by me, I found that the number of Installs is significantly affected by price, the number of reviews and rating. I decided to train the Linear Regression model to predict the number of Installs based on these parameters alone.

   The metrics used by me were R2 score.

R2 score is a statistical measure that's used to assess the goodness of fit of our regression model. Here, the model constructed by us is compared to a baseline model. The baseline model is the worst model. It doesn't make use of any independent variables to predict the value of dependent variable Y. Instead, it uses the mean of the observed responses of dependent variable Y and always predicts this mean as the value of Y.
Best possible score on this metric is 1.0 and it can be negative (because the model can be arbitrarily worse). A constant model that always predicts the expected value of y, disregarding the input features, would get an R2 score of 0.0.
The R2 score obtained on this Linear regression model was 0.85.
The below graph shows a regplot of predicted and Test values using Linear regression.

```
mean squared error 0.4297580215423224
mean absolute error 0.5400868152421135
```

This graph corresponds to testing on the test dataset.

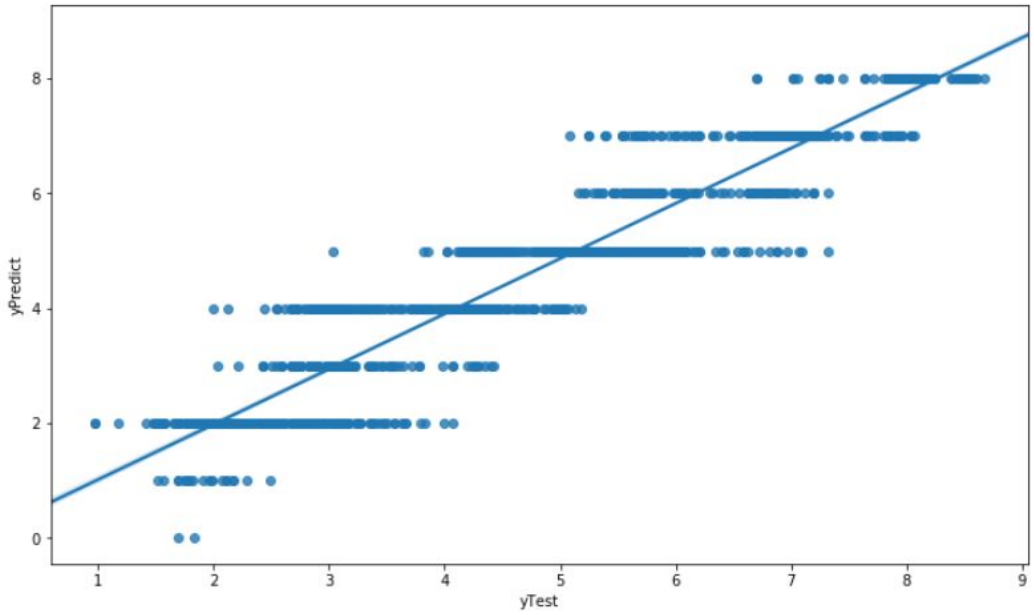


2. Neural Network Model for predicting the number of installs

I trained the neural network constructed by me for 100 epochs on the training data. The input parameters were Rating, number of reviews, price, and category. The category was one hot encoded rather than label encoded to avoid any bias towards any particular category. The metrics used by me were MSE(mse). The mse after 100 epochs came close to 0.32.
The below graph shows a regplot of predicted and Test values using Neural networks. Applying MSE and MAE metrics on test data for neural network model I got

```
mean squared error 0.3577414849659389
mean absolute error 0.45713472207132566
```

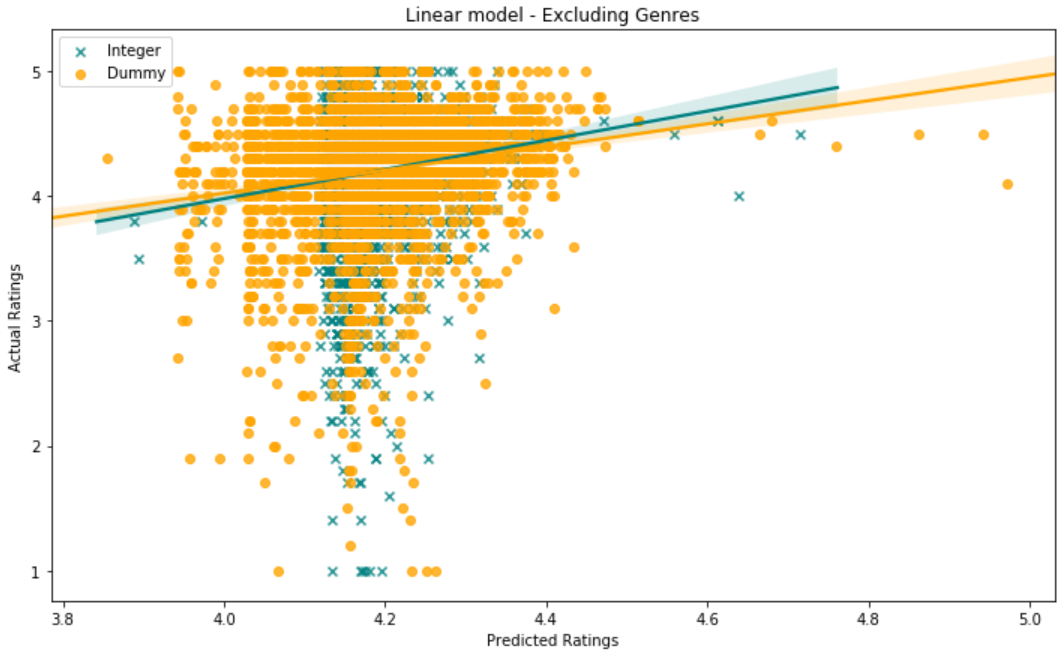
NOTE: The value predicted by my models are log scaled(base 10). To get the predicted number of installs one needs to take anti-log of the predicted values.
This graph corresponds to testing on the test dataset.



(CP)
The following are my results when my approaches were applied to different models and I have calculated my results by comparing the errors and plotting a graph of Actual vs Predicted Ratings which provides a visual feel for the results.Though as the applications are around 10000 we cannot see the density and how it affects the standard deviation for both integer and dummy. Thus we will also calculate the mean rating and standard deviation to find which is more accurate, using integer encoding or one-hot encoding

Linear Regression excluding genres :

We apply Linear Regression to the dataset without including the genres label



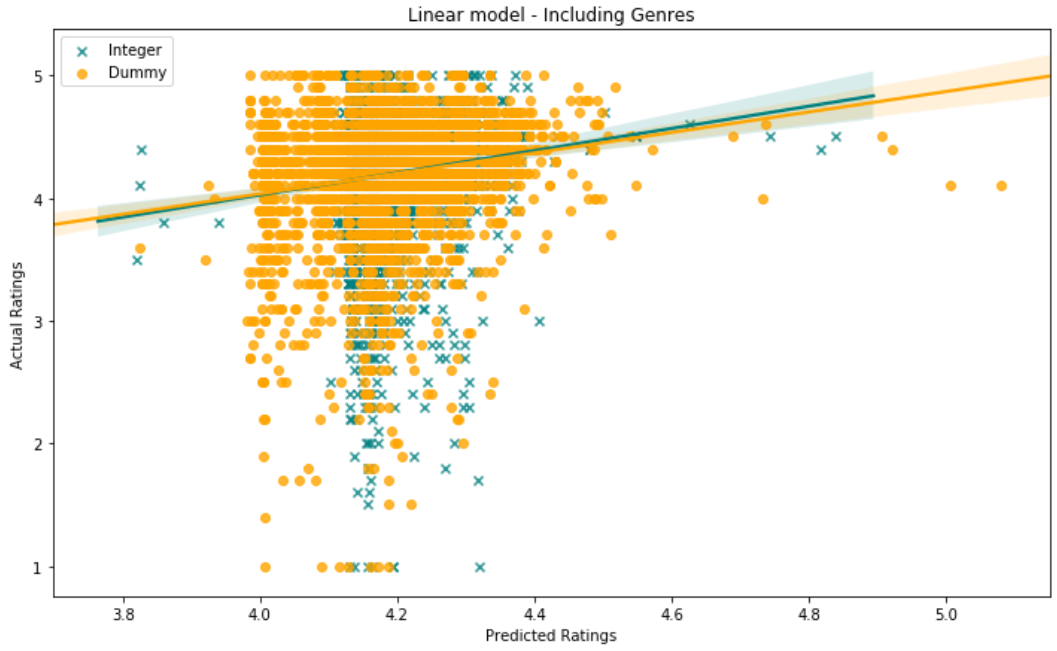
Actual mean of population:4.191837606837612
Integer encoding(mean) :4.1873290708390165
Dummy encoding(mean) :4.190100388213491
Integer encoding(std) :0.05393016592892948
Dummy encoding(std) :0.10141339123507354
Linear Regression For Integer Encoding:
MSE: 0.2593974048380525
MAE: 0.3586602884164144
MSLE: 0.012648173714083239
For dummy encoding:
MSE: 0.23326091141757077
MAE: 0.3425507229317599
MSLE: 0.01090254822607589

At first glance, it's hard to really see which model(dummy vs one-hot) is better in terms of predictive accuracy. What is striking, however, is the that at first glance, the dummy model seems favours the outcome of a lower rating compared to the integer model.

Although if we look at the actual mean of the predictive results, both are approximately the same, however, the dummy encoded results have a much larger standard deviation as compared to the integer encoded model.

Next is looking at the linear model including the genre label as a numeric value.

Linear Regression including genres:



Integer encoding(mean) :4.189955371309177
Dummy encoding(mean) :4.190338063176179
Integer encoding(std) :0.06135898515969693
Dummy encoding(std) :0.10722254835317722
Linear Regression For Integer Encoding
MSE: 0.2732221986940644
MAE: 0.3657990356410864
MSLE: 0.01329746250337762

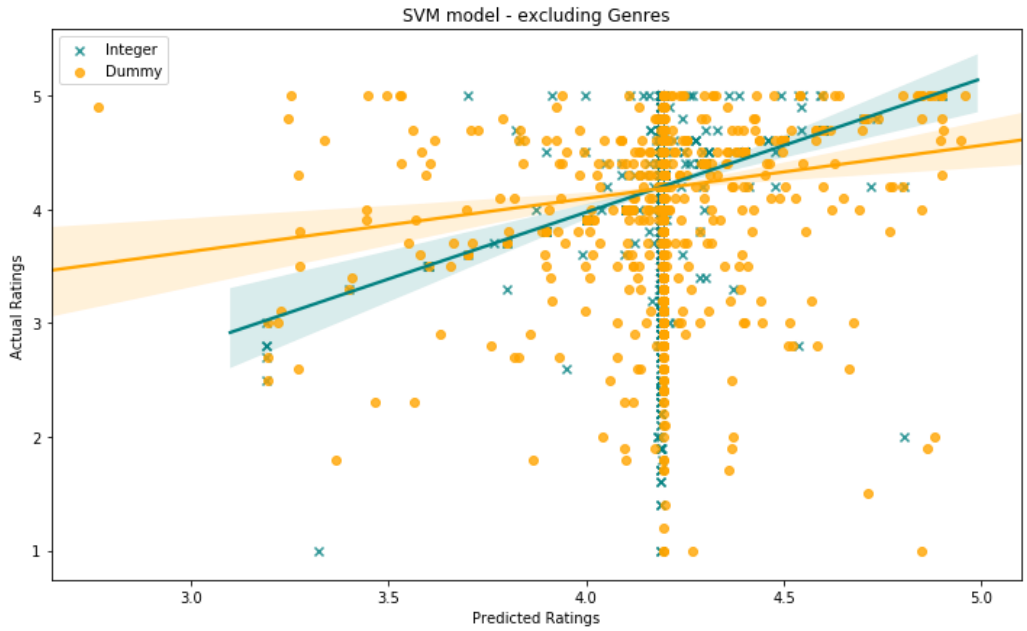
Linear Regression For dummy encoding

MSE: 0.24617550042442682
MAE: 0.35425815682526673
MSLE: 0.011486938308757627

When including the genre data, we see a slight difference in the mean between the integer and dummy encoded linear models. The dummy encoded model's std is still higher than the integer encoded model.

What's striking to me personally is that the dummy encoded regression line in the scatterplot is now flatter than the integer encoded regression line, which might suggest a "worse" outcome, given that usually, you would want your regression's beta value to be closer to 1 than to 0.

SVR Model excluding genres:



Integer encoding(mean) :4.19263281320923
Dummy encoding(mean) :4.197715672997972
Integer encoding(std) :0.08095380235939799
Dummy encoding(std) :0.1369358840558245

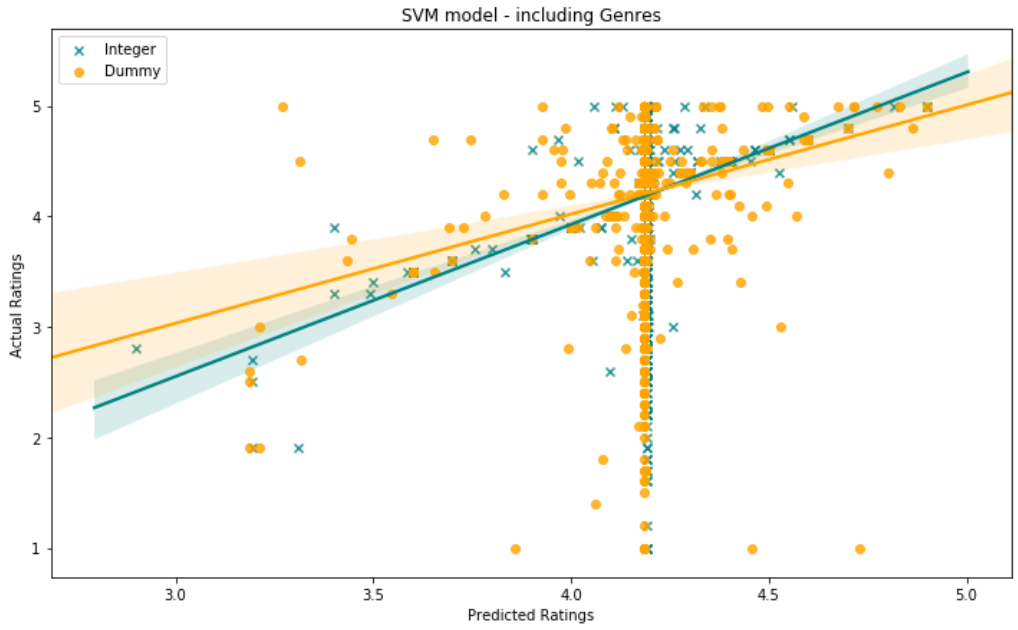
SVR excluding genres integer:
MSE: 0.23709569820965112

MAE: 0.34483086727311474
MSLE: 0.011201787228092406
SVR excluding genres dummy:
MSE: 0.2659516344386481
MAE: 0.35275167921998
MSLE: 0.012922274151660166

The results are quite interesting. Overall the model predicted quite a bit of ratings to be approximately at 4.2, even though the actual ratings were not. Looking at the scatterplot, the integer encoded model seems to have performed better in this instance.

As usual, the dummy encoded model has a higher std than the integer encoded model.

SVR model including genres:



Integer encoding(mean) :4.192589655878121
Dummy encoding(mean) :4.188436289522658
Integer encoding(std) :0.07788200835381136
Dummy encoding(std) :0.08888018936518423
SVR including genres integers
MSE: 0.2325604031051762

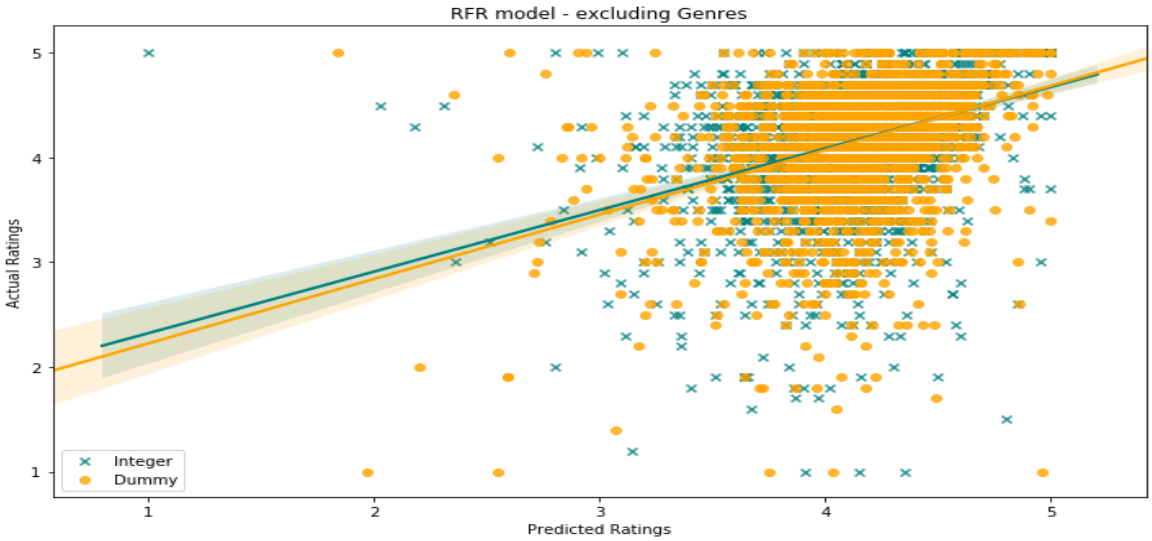
MAE: 0.34102230389444577
MSLE: 0.010907658380077267
SVR including genres dummy
MSE: 0.27217206366420454
MAE: 0.3550543119645206
MSLE: 0.013801833924004785

With the inclusion of the genre variable, the dummy encoding model now seems to be performing better, as we see the regression line comparing the actual vs the predicted results to be very similar to that of the integer encoded model.

Furthermore, the std of the dummy encoded model has fallen significantly and now has a higher mean compared to the integer encoded model.

Random Forest Regressor

This is one of the best model as not only is it fast, it also allows you to see what independent variables significantly affect the outcome of the model.

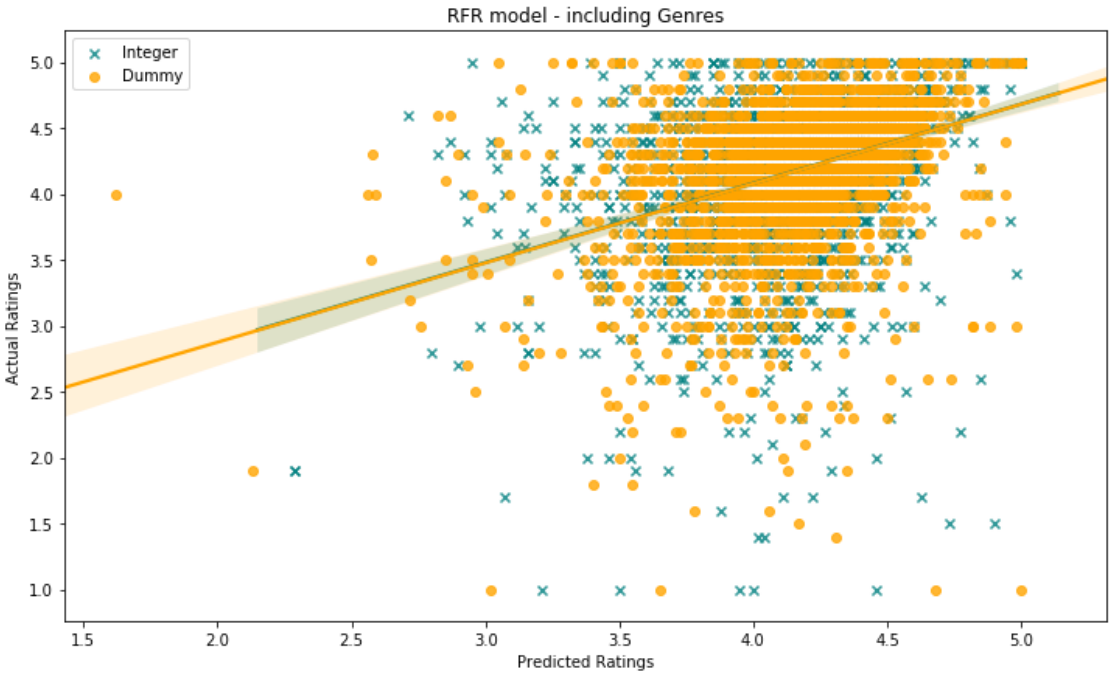


Integer encoding(mean) :4.1769935897435895
Dummy encoding(mean) :4.189666429249762
Integer encoding(std) :0.3284440265728504
Dummy encoding(std) :0.3181516319387557
RFR model excluding genres integer
MSE: 0.2571051166904084

MAE: 0.3228603988603989
MSLE: 0.012558849149305643
RFR model excluding genres dummy
MSE: 0.25499339276175215
MAE: 0.327880608974359
MSLE: 0.012402040821311905

At first glance, I would say that the RFR model produced the best predictive results, just looking at the scatter graph plotted. Overall both models, the integer and the dummy encoded models seem to perform relatively similar, although the dummy encoded model has a higher overall predicted mean

RFR model including genres:



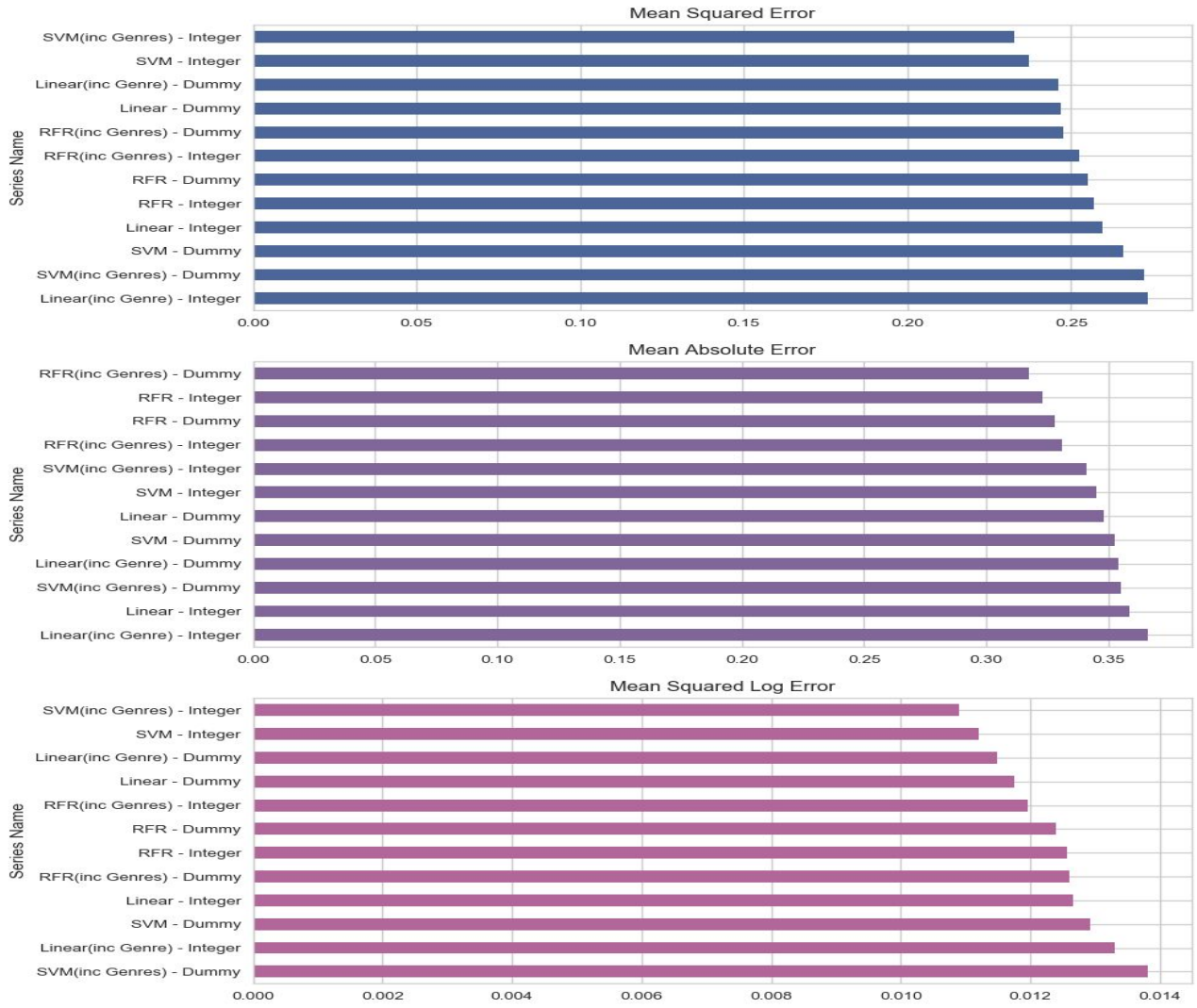
Integer encoding(mean) :4.1769935897435895
Dummy encoding(mean) :4.189666429249762
Integer encoding(std) :0.3284440265728504
Dummy encoding(std) :0.3181516319387557
RFR model including genres integer:

MSE: 0.2550664021842355
MAE: 0.3318710826210826
MSLE: 0.012831974717007583

RFR model including genres dummy:

MSE: 0.23398842948717952
MAE: 0.31800142450142455
MSLE: 0.011495910831403488

The final result is compiled below:



Discussion

(C) There are several questions that can be asked to ourselves when we start to ponder over how we should build an application like:
1. What should be the maximum size of an application that a user chooses to download to try out the application?
So you might question why size is such an important factor in determining the rating so some important statistics are:
-More installs directly correspond to higher rating(This can be verified by seeing that all immensely popular applications have good rating and it makes sense as if application get high ratings it will be recommended to other users and if they also like the application then they will rate it higher and installs will increase and thus installs and rating directly reinforce each other. This is how applications become a household name).
So coming back to the size, unless the application already has high ratings and installs a user is always sceptical of trying out a newly released application and the first thing he checks is that whats the size of the application.This is important because more size means that the developer can accommodate more features but then initial installs will be less. Less size is the direct opposite of it but this gives the user a motivation to download the application and test it out and again if he likes it the application will skyrocket faster in installs than a heavier app.
So thus we get some kind of ideal size in each category which a beginner developer can aim for.

2.Should an app be paid or free?
Well this depends on the application to most of the extent as if an application is very resource consuming or it is commercialised app then it makes sense to make the application paid.This also has a threshold,how much can a user afford to pay to try out the application. Again we reach the same deadlock as for the size of the application, as costly applications can probably provide much better functionality but the initial installs do not mount fast and so on, so we won't discuss this further.
The second method is for in-app advertisements, this also should be moderate as too much in-app advertisements drive away the user and results in negative reviews and ratings pretty fast, as the user can simply download an app which provides almost the same features but has less intrusive ads. Though our dataset doesnt allow any statistics for this so we wont discuss it further.

3.Does content rating matter?
As in our analysis we found that Most bulky apps (>50MB) belong to the **Game** and **Family** category. Despite this, these bulky apps are fairly highly rated indicating that they are bulky for a purpose. This indicates that when a user wants to download a game he or she is prepared to download an application of higher size than usual size because these categories are resource intensive and thus will tend to be of greater size and users will be install this game and unless game is way too large this wont be an issue.

Future work: The results obtained for predicting number of installs and ratings of android app gives a pretty good insight of the parameters on which they depend. However, this still doesnot give the complete picture since we have not included quality of apps in our analysis. Quality is a very important parameter which is also quite difficult to quantify. Apps with better quality are expected to fetch more number of installs and better user ratings. The question now arises is how to measure an app's quality. This can be done easily if we can somehow gather and process users' perception regarding the apps they use. That is, the reviews are as important as the number of reviews if we are to get deeper insights into this analysis. Processing user reviews can be done using sentiment analysis on the reviews available for these apps. Sentiment analysis refers to contextually mining the text so as to identify and extract subjective information in source material. Using this tool on the reviews of apps can help to understand the social sentiment of users. Integrating this information into our analysis and models given above can greatly increase our accuracy. The resulting model will then be in some sense complete.

References

[1] Shruthi Puranika , Pranav Deshpandea , K Chandrasekaran (2016). "A Novel Machine Learning Approach For Bug Prediction".6th International Conference On Advances In Computing & Communications, ICACC 2016, 6-8 September 2016, Cochin, India

[2] Chun-Hsin Wu, Jan-Ming Ho, D T Lee (2004). "Travel-time prediction with support vector regression".IEEE Transactions on Intelligent Transport Systems.

[3] Warren S. Sarle (1994)."Neural Networks and statistical models". 19th Annual SAS Users group International Conference.

[4] Nemeslaki, András & Pocsarovszky, Károly. (2011). Web crawler research methodology.Group

[5] Xu Chu, Ihab F. Ilyas, Sanjay Krishnan, Jiannan Wang (2016). "Data Cleaning: Overview and Emerging Challenges". 2016 ACM SIGMOD Conference.

[6] Diederik P. Kingma, Jimmy Lei Ba. (2015)."Adam: A Method For Stochastic Optimization". ICLR 2015 Conference.

[7] Kedar Potdar, Taher S. Pardawala, Chinmay D. Pai (2017). “A Comparative Study of Categorical Variable Encoding Techniques for Neural Network Classifiers”. International Journal of Computer Applications
[8] Handling Multicollinearity in Regression Analysis, April 2013, Retrieved from http://blog.minitab.com/blog/understanding-statistics/handlingmulticollinearity-in-regression-analysis.

Q How was Project Management done in your SE mini-project ?

A Software Project management is concerned with activities involved in ensuring that software is delivered on time and on schedule and in accordance with the requirements of the organisations developing and procuring the software.It is needed because software development is always subject to budget and schedule constraints that are set by the organisation developing the software.

The main objectives of software Project management are-

• To explain the main tasks undertaken by project managers

• To introduce software project management and to describe its distinctive characteristics

• To discuss project planning and the planning process

• To show how graphical schedule representations are used by project management

• To discuss the notion of risks and the risk management process

For our mini-project, the management activities followed by us were

- Project Planning and scheduling
  We proposed various different plans to manage our project. Since ours is a machine learning problem we decided to devote significant amount of time for data cleaning, preprocessing and graphical analysis. This Exploratory Data Analysis was done to strike outliers and various dependencies.
- Project monitoring and reviews
  We monitored our project
- Report writing and presentations