**Rating Prediction of Google Play Store Apps with**

**Application of Data Mining Techniques**

**TABLE OF CONTENTS**

|  |  |  |
| --- | --- | --- |
| **CHAPTER** | **TITLE** | **PAGE NO** |
|  | **ABSTRACT** |  |
|  | **INTRODUCTION** |  |
|  | * 1. General Introduction |  |
|  | * 1. Project Objectives |  |
|  | * 1. Problem Statement |  |
|  | **SYSTEM** **PROPOSAL** |  |
|  | * 1. Existing System |  |
|  | * + 1. Advantages |  |
|  | * 1. Proposed System |  |
|  | 2.2.1 Disadvantages |  |
|  | * 1. Literature Survey |  |
|  | **SYSTEM DIAGRAMS** |  |
|  | * 1. Architecture Diagram |  |
|  | * 1. Flow Diagram |  |
|  | * 1. UML Diagrams |  |
|  | **IMPLEMENTATION** |  |
|  | * 1. Modules |  |
|  | * 1. Modules Description |  |
|  | **SYSTEM** **REQUIREMENTS** |  |
|  | * 1. Hardware Requirements |  |
|  | * 1. Software Requirements |  |
|  | * 1. Software Description |  |
|  | * 1. Testing of Products |  |
|  | **CONCLUSION** **AND** **FUTURE** **ENHANCEMENT** |  |
|  | * 1. Conclusion |  |
|  | * 1. Future Enhancement |  |
|  | **SAMPLE** **CODING** **AND** **SAMPLE** **SCREENSHOT** |  |
|  | **REFERENCES** |  |

**ABSTRACT**

Nowadays online reviews play a significant role in influencing the decision of consumers. Consumers show their experience and information about quality in their reviews. Online reviews typically consist of qualitative (text format) and quantitative (rating) formats. In the case of Google Play store fake numeric ratings can play a big role in the success of apps. People tend to believe that a high-star rating may be significantly attached with a good review. However, user star level rating information does not usually match with text format of review. Despite many efforts to resolve this issue, Google Play Store is still facing this problem. Here, proposes a novel Google App numeric reviews & ratings contradiction prediction framework using Machine Learning approaches. Star ratings are predicted from text format of reviews after training Machine Learning models. Experimental results demonstrate that based on actual user reviews the proposed framework significantly predicts unbiased star rating of app.

1. **INTRODUCTION**

**1.1** **GENERAL INTRODUCTION**

Machine learning approaches are essential for us to take care of numerous issues. In this paper, we present machine learning models and structures in detail. Machine learning has numerous applications in numerous perspectives and has incredible advancement potential.

It is predictable that machine learning could set up ideal speculations to clarify its exhibitions. In the meantime, its capacities of unsupervised learning will be improved since there is much information on the planet however it isn't relevant to add names to every one of them. It is additionally anticipated that neural system structures will turn out to be increasingly unpredictable with the goal that they can separate all the more semantically important highlights. In addition, profound learning will consolidate with support adapting better and we can utilize this points of interest to achieve more assignments.

In today’s scenario we can see that mobile apps playing an important role in any individual’s life. It has been seen that the development of the mobile application advertise has an incredible effect on advanced innovation. Having said that, with the consistently developing versatile application showcase there is additionally an eminent ascent of portable application designers inevitably bringing about high as can be income by the worldwide portable application industry.

With enormous challenge from everywhere throughout the globe, it is basic for a designer to realize that he is continuing in the right heading. To hold this income and their place in the market the application designers may need to figure out how to stick into their present position. The Google Play Store is observed to be the biggest application platform. It has been seen that in spite of the fact that it creates more than two fold the downloads than the Apple App Store yet makes just a large portion of the cash contrasted with the App Store. In this way, I scratched information from the Play Store to direct our examination on it.

With the fast development of advanced cells, portable applications (MobileApps) have turned out to be basic pieces of our lives. Be that as it may, it is troublesome for us to follow along the fact and to understand everything about the apps as new applications are entering market each day. It is accounted for that Android1market achieved a large portion of a million applications in September 2011. Starting at now, 0.675 million Android applications are accessible on Google Play App Store. Such a lot of applications are by all accounts an extraordinary open door for clients to purchase from a wide determination extend. We trust versatile application clients consider online application surveys as a noteworthy impact for paid applications. It is trying for a potential client to peruse all the literary remarks and rating to settle on a choice. Additionally, application engineers experience issues in discovering how to improve the application execution dependent on generally speaking evaluations alone and would profit by understanding the a huge number of printed remarks.

Machine learning is an application of AI (AI) that gives systems the power to automatically learn and improve from experience without being explicitly programmed. Machine learning focuses on the event of computer programs which will access data and use it learn for themselves.

It is defined as a learning in which we train a machine as per our dataset or input. From that point forward, the machine is furnished with another arrangement of examples (data) so supervised learning analyses the provided data (set of preparing models) and creates a right result from given input.

* 1. **PROJECT OBJECTIVES**
* The main objective is to predict the ratings of Google Play Store apps using machine learning Algorithms.
* To perform Data Analysis and prediction into the Google Play store application dataset that I have collected from repository.
* Using Machine Learning Algorithms, I have tried to discover the relationships among various attributes present in my dataset such as which application is free or paid, about the user reviews, rating of the application.
* To enhance the performance of overall prediction result.
  1. **PROBLEM STATEMENT**

We have taken the dataset and observed it nicely and as per our need we have taken various attributes to analyze and further display the result. By doing this, we can clearly and easily observe the dataset. Moreover, Firstly, I will analyze different attributes given in dataset. Secondly, I will do prediction of those different attributes like predict whether the user review is positive or negative.

1. **SYSTEM PROPOSAL**

**2.1 EXISTING SYSTEM**

In existing system, to find out names of those applications which come on the top of the entire free app with their respective rating. After analyzing all the attributes of given dataset using hive I have made the prediction on the user reviews of the particular application that which particular review is positive or negative using algorithms. Calculate the output compare it with actual output and then train the model and adjust the weights to minimize the error using backpropagation this process is performed several times. Finally by comparing the output after training process with the set range to find out which review is positive or negative.

**2.1.1 DISADVANTAGES**

* Data inconsistency Problems
* Less Efficiency and effective.
* Incorrect Prediction Results.
  1. **PROPOSED SYSTEM**

The proposed model is introduced to overcome all the disadvantages that arise in the existing system. This system will increase the accuracy of the classification results by classifying the data based on the googleplay apps. The dataset collected from the Google Play store is semi structured or unstructured and contains significant superfluous data (defined as not contributing significantly to the prediction process). Training a supervised machine learning algorithm requires textual documents to be represented in vectorial form. For this purpose, textual data must be converted into numbers without losing information. Apply Machine Learning algorithms to predict app’s rating with Regressors (KNN and Random Forest). Finally, the Performance of classifiers is evaluated on the accuracy, MAE, RMSE and R2-Score.

**2.2.1 ADVANTAGES**

* High performance.
* Provide accurate prediction results.
* It avoids data inconsistency.
  1. **LITERATURE SURVEY**

# Title: Correlation of benign incidental findings seen on whole-body pet-ct with knee mri: patterns of 18f-fdg avidity, intra-articular pathology, and bone marrow edema lesions

# Year: 2018

**Author**: **C. J. Burke, W. R. Walter, S. Gaddam, H. Pham, J. S. Babb, J. Sanger, and F. Ponzo**

**Methodology**

Objectives To correlate patterns of 18F-FDG uptake on whole-body PET-CT with MR findings and compare the degree of FDG activity between symptomatic and asymptomatic knees. Materials and methods Retrospective database query was performed using codes for knee MRI as well as whole-body PET-CT. Patients with malignant disease involving the knee or hardware were excluded. Patients who had both studies performed within 1 year between 2012 and 2017 were included for analysis. Knee joint osteoarthrosis, meniscal and ligamentous integrity, presence of joint effusion, and synovitis were assessed and recorded. Bone marrow edema lesions (BMELs) were identified, segmented, and analyzed using volumetric analysis. SUVmax was assessed over the suprapatellar joint space, intercondylar notch and Hoffa’s fat pad. Symptomatic and asymptomatic knees were compared in patients with unilateral symptoms. Results Twenty-two cases (20 patients) with mean age 63.3 years (range, 36–91 years) were included. Two patients had bilateral pain. The most FDG avid regions in both symptomatic and asymptomatic knees were the intercondylar notch (SUVmax = 1.84 vs. 1.51), followed by suprapatellar pouch (SUVmax = 1.74 vs. 1.29) and Hoffa’s fat pad (SUVmax = 1.01 vs. 0.87). SUVmax was significantly associated with cartilage loss (mean modified Outerbridge score) (r = 0.60, p = 0.003) and degree of synovitis (r = 0.48, p = 0023). Overall, mean SUVmax was significantly higher in the presence of a meniscal tear (1.83 ± 0.67 vs. 1.22 ± 0.40, p = 0.030). Nine patients had BMELs (volume: range = 0.6–27.8, mean = 7.79) however there was no significant association between BMEL volume and SUVmax. Conclusions Higher FDG activity correlates with intra-articular derangement and the intercondylar notch represents the most metabolically active region of the knee.

The primary purpose of this study therefore was to correlate the breadth of pathology revealed by both exams, particularly cartilage loss, meniscal and cruciate ligamentous pathology in conjunction with metabolic activity and, where present, bone marrow edema lesion (BMEL) volume. Furthermore, as the inclusion of both knees within the whole-body PET-CT scan field allows the opportunity to compare bilateral knees, i.e., symptomatic and asymptomatic sides, a secondary purpose was to compare the FDG avidity in the symptomatic knees with the asymptomatic contralateral knees as an internal control.

**Advantage**

Compare the degree of FDG activity between symptomatic and asymptomatic knees.

**Disadvantage**

Less Effective

# Title: Introduction to machine learning: k-nearest neighbors

# Year: 2016

**Author**: **Z. Zhang**

**Methodology**

Machine learning techniques have been widely used in many scientific fields, but its use in medical literature is limited partly because of technical difficulties. k-nearest neighbors (kNN) is a simple method of machine learning. The article introduces some basic ideas underlying the kNN algorithm, and then focuses on how to perform kNN modeling with R. The dataset should be prepared before running the knn() function in R. After prediction of outcome with kNN algorithm, the diagnostic performance of the model should be checked. Average accuracy is the mostly widely used statistic to reflect the kNN algorithm. Factors such as k value, distance calculation and choice of appropriate predictors all have significant impact on the model performance.

kNN classifier is to classify unlabeled observations by assigning them to the class of the most similar labeled examples. Characteristics of observations are collected for both training and test dataset. For example, fruit, vegetable and grain can be distinguished by their crunchiness and sweetness (Figure 1). For the purpose of displaying them on a two-dimension plot, only two characteristics are employed. In reality, there can be any number of predictors, and the example can be extended to incorporate any number of characteristics. In general, fruits are sweeter than vegetables. Grains are neither crunchy nor sweet. Our work is to determine which category the sweet potato belongs to. In this example we choose four nearest kinds of food, they are apple, green bean, lettuce, and corn. Because the vegetable wins the most votes, sweet potato is assigned to the class of vegetable. You can see that the key concept of kNN is easy to understand.

Another concept is the parameter k which decides how many neighbors will be chosen for kNN algorithm. The appropriate choice of k has significant impact on the diagnostic performance of kNN algorithm. A large k reduces the impact of variance caused by random error, but runs the risk of ignoring small but important pattern. The key to choose an appropriate k value is to strike a balance between overfitting and underfitting. Because kNN is a non-parametric algorithm, we will not obtain parameters for the model. The kNN() function returns a vector containing factor of classifications of test set.

**Advantage**

Very useful to predict labelled data with the help of producers based on the distance.

**Disadvantage**

It is having some scalability problem if it is unlabelled.

# Title: Risk analysis tool for granting credit by financial cooperatives for legal entities

# Year: 2020

**Author**: **Adailton Vieira, Marino Luiz EyerkauferI, Rodrigo Renge**

**Methodology**

This study aims to elaborate an analysis tool with a broad view of the risks involved in a financial cooperative in granting of credit. The risks in cooperative were analyzed to identify current tools of credit analysis; to verify the efficiency of these, the risk mutations and their impacts during different periods. This research was applied with managers by an interview and was carried out a documentary survey of loan portfolio, which revealed that the lack of an effective credit risk analysis interferes directly with the institution's economic result, which would represent an average of five million more between 2010 and 2016. The findings show that 26% of concessions in the analyzed period were not recovered. It is concluded that the analysis of credit in the cooperative investigated has limitations, mainly regarding the risks not identified before the granting of credit, therefore, a generic model of comprehensive risk assessment was developed, capable of remedying the frailties found.

**Advantage**

More Reliable.

**Disadvantage**

It is less in efficiency and not give perfect result.

# Title: Classifiers Comparison for Attack Detection in Computer Networks

# Year: 2017

**Author**: **K. R. Santos, I. R. R. Silva, and R. A. A. Fagundes**

**Methodology**

With the advancement of information technology and the consequent growth of information trafficked the computer network attacks and anomalies in this environment it has become increasingly common in recent decades, causing many researchers were concerned with the identification of these attacks. In the present work presents a comparative study of classifiers to detect attacks on computer networks, which uses the "dataset" 10% of KDDCUP'99. The classifiers Naive Bayes, Bayesian Networks and k-NN were used, which were evaluated according to the number of correctly classified instances. At the end of this work, we present the most efficient data mining classification technique, among the analysed to detect attacks on computer networks. We offer a review on attack detection methods involving strength of deep learning techniques. Specifically, we firstly summarize fundamental problems of network security and attack detection and introduce several successful related applications using deep learning structure. On the basis of categorization on deep learning methods, we pay special attention to attack detection methods built on different kinds of architectures, such as autoencoders, generative adversarial network, recurrent neural network, and convolutional neural network. Afterwards, we present some benchmark datasets with descriptions and compare the performance of representing approaches to show the current working state of attack detection methods with deep learning structures. Finally, we summarize this paper and discuss some ways to improve the performance of attack detection under thoughts of utilizing deep learning structures.

The continuous development and extensive usage of Internet benefit numerous network users from a quantity of aspects. Meanwhile, network security becomes much more important with wide usage of network. Network security is closely related to computers, networks, programs, various data, and so forth, where the purpose of defense is to prevent unauthorized access and modification. However, the growing number of internet-connected systems in finance, E-commerce, and military makes them become targets of network attacks, resulting in large quantity of risk and damage. Essentially, it is necessary to provide effective strategies to detect and defend attacks and maintain network security. Furthermore, different kinds of attacks are usually required to be processed in different ways. How to identify different kinds of network attacks thus becomes the main challenge in domain of network security to be solved, especially those attacks never seen before.

**Advantage**

High Efficiency.

**Disadvantage**

Low in accuracy performance.

# Title: Numeric rating of Apps on Google Play Store by sentiment analysis on user reviews

# Year: 2014

**Author**: **M. R. Islam**

**Methodology**

The sudden eruption of sentiment analysis and opinion mining has opened new possibilities to improve our information gathering interests. We are always keen to know what others say about the devices or applications we are going to use. Its observed that sometimes the numeric rating has vast difference than the reviews given by the users. To remove this ambiguity a unified rating system has been proposed here. The starred rating and a generated numeric polarity of the reviews are combined to generate the final rating. The proposition is based on sentiment analysis and an optimized probabilistic approach described by a group of researchers. The approach is proved for its efficiency in a diverse corpus of writings where the targets are of different categories.

Generally choose their required apps from the GooglePlay Store. Generally it is seen that people makes the decision for any app on the basis of the numeric rating of that. The rating is the average of all the ratings given by other users by stars. Moreover, the users have to include a comment as well. It is observed that ambiguity lies between the star rating and the comments of the users. It creates confusion to the new user who is going to download and use the app. Here another problem evolves, people always go for a summary rather than an elaborative statement. So this problem rises with ambiguity of reviews and the biasness of users to summarized option.

**Advantage**

Good in Performance.

**Disadvantage**

Not Reliable

# Title: The effect of data pre-processing on optimized training of artificial neural networks

# Year: 2013

**Author**: **N. Mohd Nawi, W. Atomi, and S. M. Rehman Gillani**

**Methodology**

Recently, the popularity of artificial neural networks (ANN) is increasing since its capacity to model very complex problems in the area of Machine Learning, Data Mining and Pattern Recognition. Improving training efficacy of ANN based algorithm is a dynamic area of research and several papers have been reviewed in the literature. The performance of Multi-layer Perceptrons (MLP) trained with Back Propagation Artificial Neural Network (BP-ANN) method is highly influenced by the size of the datasets and the data-preprocessing techniques used. This work analyses the benefits of using pre-processing datasets using different techniques in-order to improve the ANN convergence. Specifically Min-Max, Z-Score and Decimal Scaling Normalization preprocessing techniques were evaluated. The simulation results show that the computational efficiency of ANN training process is highly enhanced when coupled with different preprocessing techniques

Data pre-processing is an important step in the data mining process. Mo

stly, data gathering methods are lightly

controlled, resulting in outliers, impossible data combinations, and missing values, etc. Analyzing data that has not

been carefully separated for such problems can produce confusing results. Thus, the depiction and quality of data is

first and foremost before running any analysis [29]. The quality, reliability and availability are some of the factors

th

at may lead to a successful data interpretation by a neural network. If there is inappropriate information present or

noisy and unreliable data, then knowledge discovery becomes very difficult during the training process. Data

preparation and filtering steps can take considerable amount of processing time [30] but once pre-processing is done

th

e data become more reliable and robust results are achieved. This study will compare three methods for data

normalization including min-max normalization, z-score normalization and normalization by decimal scaling.

Data pre-processing is an important step in the data mining process. Mo

stly, data gathering methods are lightly

controlled, resulting in outliers, impossible data combinations, and missing values, etc. Analyzing data that has not

been carefully separated for such problems can produce confusing results. Thus, the depiction and quality of data is

first and foremost before running any analysis [29]. The quality, reliability and availability are some of the factors

th

at may lead to a successful data interpretation by a neural network. If there is inappropriate information present or

noisy and unreliable data, then knowledge discovery becomes very difficult during the training process. Data

preparation and filtering steps can take considerable amount of processing time [30] but once pre-processing is done

th

e data become more reliable and robust results are achieved. This study will compare three methods for data

normalization including min-max normalization, z-score normalization and normalization by decimal scaling.

Data pre-processing is an important step in the data mining process. Mostly, data gathering methods are lightly controlled, resulting in outliers, impossible data combinations, and missing values, etc. Analyzing data that has not been carefully separated for such problems can produce confusing results. Thus, the depiction and quality of data is first and foremost before running any analysis. The quality, reliability and availability are some of the factors that may lead to a successful data interpretation by a neural network. If there is inappropriate information present or noisy and unreliable data, then knowledge discovery becomes very difficult during the training process. Data preparation and filtering steps can take considerable amount of processing time but once pre-processing is done the data become more reliable and robust results are achieved. This study will compare three methods for data normalization including min-max normalization, z-score normalization and normalization by decimal scaling.

The main focus of this study is to improve the accuracy of ANN models by using three selected pre-processing techniques. Before discussing the simulation test results there are certain things that need to be explained such as tools and technologies, network topologies, parameters settings, data sets and training methodology used during the experimentation process. The simulations were carried out on three datasets taken from UCI Machine Learning Repository. For each dataset, we have estimated the overall accuracy by 5-fold cross-validation: each data set was divided into five equal parts, using four folds as the training set and the remaining block as independent test set. Those three datasets are the Wine recognition, Haberman’s survival and the Iris.

From these results, some initial remarks can be drawn. Firstly, for the majority of the datasets, there exists at least one pre-processing technique whose classification accuracy is better as compared to others. Also, we detect that after applying the pre-processing techniques with several MLP-ANN methods described previously, the PMMN1 technique was the best one preserving the overall accuracy as much with the GDG as well as GDGL models.

**Advantage**

Efficiency is good.

**Disadvantage**

Not give accurate prediction result.

# Title: Scikit-learn: Machine Learning in Python

# Year: 2011

**Author**: **F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay**

**Methodology**

Scikit-learn is a Python module integrating a wide range of state-of-the-art machine learning algorithms for medium-scale supervised and unsupervised problems. This package focuses on bringing machine learning to non-specialists using a general-purpose high-level language. Emphasis is put on ease of use, performance, documentation, and API consistency. It has minimal dependencies and is distributed under the simplified BSD license, encouraging its use in both academic and commercial settings. Source code, binaries, and documentation can be downloaded from http://scikit-learn.sourceforge.net.

Scikit-learn harnesses this rich environment to provide state-of-the-art implementations of many well known machine learning algorithms, while maintaining an easy-to-use interface tightly integrated with the Python language. This answers the growing need for statistical data analysis by non-specialists in the software and web industries, as well as in fields outside of computer-science, such as biology or physics. Scikit-learn differs from other machine learning toolboxes in Python for various reasons: i) it is distributed under the BSD license ii) it incorporates compiled code for efficiency, unlike MDP (Zito et al., 2008) and pybrain (Schaul et al., 2010), iii) it depends only on numpy and scipy to facilitate easy distribution, unlike pymvpa (Hanke et al., 2009) that has optional dependencies such as R and shogun, and iv) it focuses on imperative programming, unlike pybrain which uses a data-flow framework. While the package is mostly written in Python, it incorporates the C++ libraries LibSVM (Chang and Lin, 2001) and LibLinear (Fan et al., 2008) that provide reference implementations of SVMs and generalized linear models with compatible licenses. Binary packages are available on a rich set of platforms including Windows and any POSIX platforms.

Model selection. Scikit-learn can evaluate an estimator’s performance or select parameters using cross-validation, optionally distributing the computation to several cores. This is accomplished by wrapping an estimator in a GridSearchCV object, where the “CV” stands for “cross-validated”. During the call to fit, it selects the parameters on a specified parameter grid, maximizing a score (the score method of the underlying estimator). predict, score, or transform are then delegated to the tuned estimator. This object can therefore be used transparently as any other estimator. Cross validation can be made more efficient for certain estimators by exploiting specific properties, such as warm restarts or regularization paths (Friedman et al., 2010). This is supported through special objects, such as the LassoCV. Finally, a Pipeline object can combine several transformers and an estimator to create a combined estimator to, for example, apply dimension reduction before fitting. It behaves as a standard estimator, and GridSearchCV therefore tune the parameters of all steps.

**Advantage**

Text corpus for content mining gives good analysis result.

**Disadvantage**

Not effective.

1. **SYSTEM DIAGRAMS**

**3.1 ARCHITECTURE DIAGRAM**

**Dataset**

**Data preprocess**

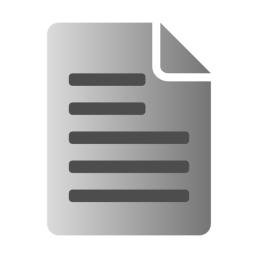
**Splitting Dataset into Train and Test**

**Feature Extraction**

**Classification**

**Result Generation**

**Prediction**



**Select and View Dataset**

**3.2 FLOW DIAGRAM**

**Start**

**Select Dataset**

**View Data**

**Data Preprocessing**

**Null Value Checking**

**Label Encoding**

**Data Visualization**

**Splitting Dataset into Training and Test Dataset**

**Classification using ML Algorithm**

**Prediction**

**Performance Evaluation**

**KNN**

**Random Forest**

**Accuracy**

**MAE**

**RMSE**

**R2-Score**

**3.3 UML DIAGRAMS**

**USE CASE DIAGRAM**

**ANALYST**

**CLASS DIAGRAM**

**DATASET**

Select dataset ()

Import dataset ()

View dataset ()

**FEATURE EXTRACTION**

Dataset Splitting Train and Test

Feature Extraction

**DATA PREPROCESSING**

Data Cleaning ()

Null value check ()

Label Encoding ()

**CLASSIFICATION**

Detection ()

Prediction ()

**RESULT GENERATION**

Result Generation ()

**SEQUENCE DIAGRAM**

DATA SELECTION & VIEW

PREPROCESSING

FEATURE EXTRACTION

CLASSIFICATION

Select path

Preview Data

Data Preprocess

Label encoding

Classification

RESULT GENERATION

Import Dataset

Splitting Dataset into Training and Test Data

Null value check

**ER DIAGRAM**

**DATA SELECTION & LOAD**

**DATA PREPROCESS**

**FEATURE EXTRACTION**

**CLASSIFICATION**

**RESULT GENERATION**

**FEATURES**

1. **IMPLEMENTATION**

**4.1 MODULES**

* Data Selection and Loading
* Data Preprocessing
* Exploratory Data Analysis
* Splitting Dataset into Train and Test Data
* Classification
* Result Generation

**4.2 MODULE DESCRIPTION**

**DATA SELECTION AND LOADING**

The data selection is the process of selecting the data for detecting the ratings from the googleplay app dataset. The dataset which contains the information about the each app (row) has values for catergory, rating, size, and more. This information is scraped from the Google Play Store. This app information would not be available without it. The Play Store apps data has enormous potential to drive app-making businesses to success. Actionable insights can be drawn for developers to work on and capture the Android market!

**DATA PREPROCESSING**

Data pre-processing is the process of removing the unwanted data from the dataset. Preprocessing is important into transitioning raw data into a more desirable format. Undergoing the preprocessing process can help with completeness and compellability. For instance, you'll see if certain values were recorded or not. Also, you'll see how trustable the info is. It could also help with finding how consistent the values are. We need preprocessing because most real-world data are dirty. Data can be noisy i.e. the data can contain outliers or simply errors generally. Data can also be incomplete i.e. there can be some missing values.

**EXPLORATORY DATA ANALYSIS**

**Exploratory data analysis** is an approach of [analyzing](https://en.wikipedia.org/wiki/Data_analysis) data sets to summarize their main characteristics, often using statistical graphics and other data visualization methods. A statistical model can be used or not, but primarily EDA is for seeing what the data can tell us beyond the formal modeling or hypothesis testing task.

The objectives of EDA are to:

* Suggest hypotheses about the causes of observed phenomena
* Assess assumptions on which statistical inference will be based
* Support the selection of appropriate statistical tools and techniques
* Provide a basis for further data collection through surveys or experiments

**SPLITTING DATASET INTO TRAIN AND TEST DATA**

Data splitting is the act of partitioning available data into two portions, usually for cross-validator purposes. One portion of the data is used to develop a predictive model. And the other to evaluate the model's performance. Separating data into training and testing sets is an important part of evaluating data mining models. Typically, when you separate a data set into a training set and testing set, most of the data is used for training, and a smaller portion of the data is used for testing.

The train-test split procedure is used to estimate the performance of machine learning algorithms when they are used to make predictions on data not used to train the model.

It is a fast and easy procedure to perform, the results of which allow you to compare the performance of machine learning algorithms for your predictive modeling problem. Although simple to use and interpret, there are times when the procedure should not be used, such as when you have a small dataset and situations where additional configuration is required, such as when it is used for classification and the dataset is not balanced.

**CLASSIFICATION**

Classification is a process related to categorization, the process in which ideas and objects are recognized, differentiated, and understood. In this project, the KNN and Random Forest classification algorithm is used for predicting the googleplay app ratings.

KNN is easiest supervised machine learning algorithm. It’s the foremost basic machine learning algorithm you'll find on scikit-learn. We will use KNN solve complicated problems. With the assistance of KNN we will do pattern recognition and data processing. KNN defines the similarity. From the given dataset KNN finds common groups between attributes. We split the info into training and test set. Then we will see what proportion similarity it becomes on the result.

Random forest regression is applied to all the variables the results of random forest determine the importance of all the variable and their influence on the rating. The results of random forest regression are evaluated using Mean Square Error. Random forest model is the first model that is applied to the dataset and the results of Random forest classification are computed for a number of variables to find the importance of these variables.

**RESULT GENERATION**

The Final Result will get generated based on the overall classification and prediction. The performance of this proposed approach is evaluated using some measures like,

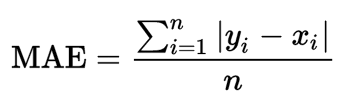
**Accuracy**

Accuracy of classifier refers to the ability of classifier. It predicts the class label correctly and the accuracy of the predictor refers to how well a given predictor can guess the value of predicted attribute for a new data.

AC= (TP+TN)/(TP+TN+FP+FN)

**MAE – Mean Absolute Error**

**Mean Absolute Error** (**MAE**) is a measure of [errors](https://en.wikipedia.org/wiki/Error_(statistics)) between paired observations expressing the same phenomenon. Examples of *Y* versus *X* include comparisons of predicted versus observed, subsequent time versus initial time, and one technique of measurement versus an alternative technique of measurement. MAE is calculated as:



**RMSE – Root Mean Square Error**

**Root Mean Square Error**(RMSE) is the standard deviation of the residuals (prediction errors). Residuals are a measure of how far from the regression line data points are; RMSE is a measure of how spread out these residuals are. In other words, it tells you how concentrated the data is around the line of best fit. Root mean square error is commonly used in climatology, forecasting, and regression analysis to verify experimental results.

 Where SDy is the standard deviation of Y.

1. **SYSTEM REQUIREMENTS**

**5.1 Hardware Requirements**

* System : Pentium IV 2.4 GHz
* Hard Disk : 1000 GB
* Monitor : 15 VGA color
* Mouse : Logitech.
* Keyboard : 110 keys enhanced
* Ram : 4GB

**5.2 Software Requirements**

* O/S : Windows 7
* Language : python.
* IDE : Anaconda - Spyder

**5.3 SOFTWARE DESCRIPTION**

**Python**

Python is a general-purpose interpreted, interactive, object-oriented, and high-level programming language. It was created by Guido van Rossum during 1985- 1990. Like Perl, Python source code is also available under the GNU General Public License (GPL). This tutorial gives enough understanding on Python programming language.

Python is a popular programming language. It was created in 1991 by Guido van Rossum.

It is used for:

* web development (server-side),
* software development,
* mathematics,
* System scripting.

Python can be used on a server to create web applications. Python can be used alongside software to create workflows. Python can connect to database systems. It can also read and modify files. Python can be used to handle big data and perform complex mathematics. Python can be used for rapid prototyping, or for production-ready software development.

Python works on different platforms (Windows, Mac, Linux, Raspberry Pi, etc). Python has a simple syntax similar to the English language. Python has syntax that allows developers to write programs with fewer lines than some other programming languages.

Python runs on an interpreter system, meaning that code can be executed as soon as it is written. This means that prototyping can be very quick. Python can be treated in a procedural way, an object-orientated way or a functional way.

Python is a high-level, interpreted, interactive and object-oriented scripting language. Python is designed to be highly readable. It uses English keywords frequently where as other languages use punctuation, and it has fewer syntactical constructions than other languages.

* **Python is Interpreted** − Python is processed at runtime by the interpreter. You do not need to compile your program before executing it. This is similar to PERL and PHP.
* **Python is Interactive** − you can actually sit at a Python prompt and interact with the interpreter directly to write your programs.
* **Python is Object-Oriented** − Python supports Object-Oriented style or technique of programming that encapsulates code within objects.
* **Python is a Beginner's Language** − Python is a great language for the beginner-level programmers and supports the development of a wide range of applications from simple text processing to WWW browsers to games.

## History of Python

Python was developed by Guido van Rossum in the late eighties and early nineties at the National Research Institute for Mathematics and Computer Science in the Netherlands.

Python is derived from many other languages, including ABC, Modula-3, C, C++, Algol-68, SmallTalk, and Unix shell and other scripting languages.

Python is copyrighted. Like Perl, Python source code is now available under the GNU General Public License (GPL).

Python is now maintained by a core development team at the institute, although Guido van Rossum still holds a vital role in directing its progress.

## Python Features

Python's features include −

* **Easy-to-learn** − Python has few keywords, simple structure, and a clearly defined syntax. This allows the student to pick up the language quickly.
* **Easy-to-read** − Python code is more clearly defined and visible to the eyes.
* **Easy-to-maintain** − Python's source code is fairly easy-to-maintain.
* **A broad standard library** − Python's bulk of the library is very portable and cross-platform compatible on UNIX, Windows, and Macintosh.
* **Interactive Mode** − Python has support for an interactive mode which allows interactive testing and debugging of snippets of code.
* **Portable** − Python can run on a wide variety of hardware platforms and has the same interface on all platforms.
* **Extendable** − you can add low-level modules to the Python interpreter. These modules enable programmers to add to or customize their tools to be more efficient.
* **Databases** − Python provides interfaces to all major commercial databases.
* **GUI Programming** − Python supports GUI applications that can be created and ported to many system calls, libraries and windows systems, such as Windows MFC, Macintosh, and the X Window system of Unix.
* **Scalable** − Python provides a better structure and support for large programs than shell scripting.

Apart from the above-mentioned features, Python has a big list of good features, few are listed below −

* It supports functional and structured programming methods as well as OOP.
* It can be used as a scripting language or can be compiled to byte-code for building large applications.
* It provides very high-level dynamic data types and supports dynamic type checking.
* It supports automatic garbage collection.
* It can be easily integrated with C, C++, COM, ActiveX, CORBA, and Java.

Python is available on a wide variety of platforms including Linux and Mac OS X.

### Python Syntax compared to other programming languages

* Python was designed to for readability, and has some similarities to the English language with influence from mathematics.
* Python uses new lines to complete a command, as opposed to other programming languages which often use semicolons or parentheses.
* Python relies on indentation, using whitespace, to define scope; such as the scope of loops, functions and classes. Other programming languages often use curly-brackets for this purpose.

**Anaconda**

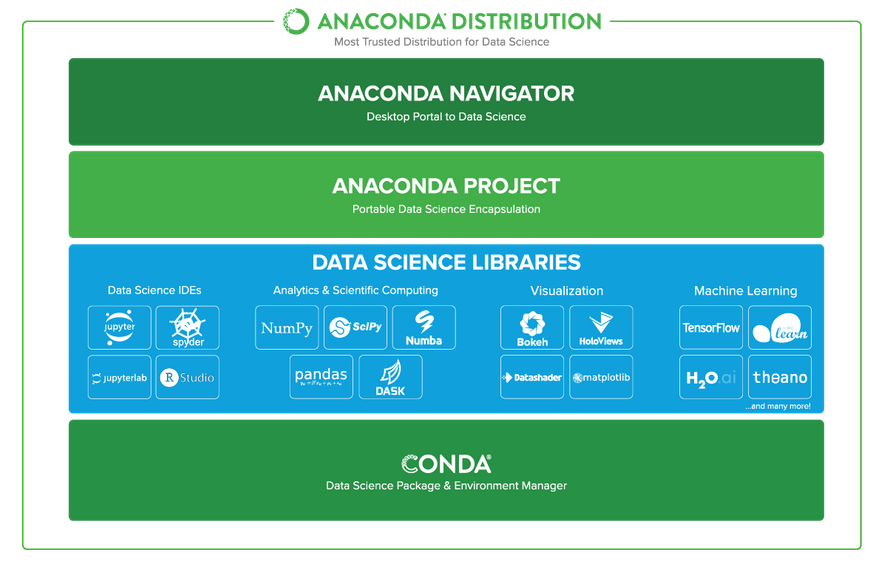
Anaconda is the most popular python data science platform.

## Anaconda Distribution

With over 6 million users, the open source Anaconda Distribution is the fastest and easiest way to do Python and R data science and machine learning on Linux, Windows, and Mac OS X. It's the industry standard for developing, testing, and training on a single machine.

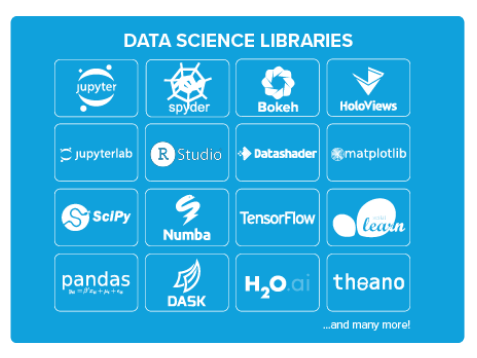
## Anaconda Enterprise

Anaconda Enterprise is an AI/ML enablement platform that empowers organizations to develop, govern, and automate AI/ML and data science from laptop through training to production. It lets organizations scale from individual data scientists to collaborative teams of thousands, and to go from a single server to thousands of nodes for model training and deployment.



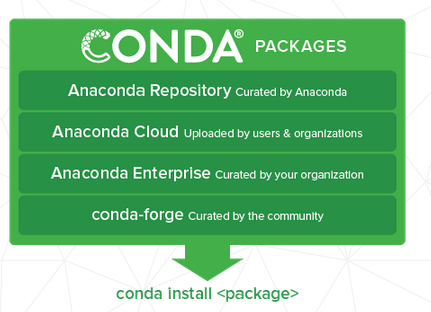
### Anaconda Data Science Libraries

* Over 1,400 Anaconda-curated and community data science packages
* Develop data science projects using your favourite IDEs, including Jupyter, JupyterLab, Spyder, and RStudio
* Analyse data with scalability and performance with Dask, numpy, pandas, and Numba
* Visualize your data with Matplotlib, Bokeh, Datashader, and Holoviews
* Create machine learning and deep learning models with Scikit-learn, Tensorflow, h20, and Theano



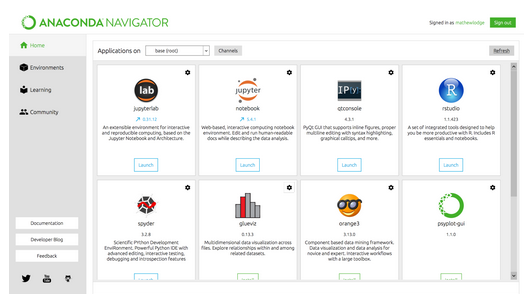
### Conda, the Data Science Package & Environment Manager

* Automatically manages all packages, including cross-language dependencies
* Works across all platforms: Linux, macOS, Windows
* Create virtual environments
* Download conda packages from Anaconda, Anaconda Enterprise, Conda Forge, and Anaconda Cloud



### Anaconda Navigator, the Desktop Portal to Data Science

* Install and launch applications and editors including Jupyter, RStudio, Visual Studio Code, and Spyder
* Manage your local environments and data science projects from a graphical interface
* Connect to Anaconda Cloud or Anaconda Enterprise
* Access the latest learning and community resources



**Spyder**

Spyder is an open source cross-platform integrated development environment (IDE) for scientific programming in the Python language. ... Initially created and developed by Pierre Raybaut in 2009, since 2012 Spyder has been maintained and continuously improved by a team of scientific Python developers and the community. Strongly recommend the free, open-source Spyder Integrated Development Environment (IDE) for scientific and engineering programming, due to its integrated editor, interpreter console, and debugging tools. Spyder is included in Anaconda and other distributions.

Spyder is a powerful scientific environment written in Python, for Python, and designed by and for scientists, engineers and data analysts. It offers a unique combination of the advanced editing, analysis, debugging, and profiling functionality of a comprehensive development tool with the data exploration, interactive execution, deep inspection, and beautiful visualization capabilities of a scientific package.

Beyond its many built-in features, its abilities can be extended even further via its plugin system and API. Furthermore, Spyder can also be used as a PyQt5 extension library, allowing developers to build upon its functionality and embed its components, such as the interactive console, in their own PyQt software.

### Editor

Work efficiently in a multi-language editor with a function/class browser, code analysis tools, automatic code completion, horizontal/vertical splitting, and go-to-definition.

### IPython Console

Harness the power of as many IPython consoles as you like within the flexibility of a full GUI interface; run your code by line, cell, or file; and render plots right inline.

### Variable Explorer

Interact with and modify variables on the fly: plot a histogram or time series, edit a date frame or Numpy array, sort a collection, dig into nested objects, and more!

### Profiler

Find and eliminate bottlenecks to unchain your code's performance.

### Debugger

Trace each step of your code's execution interactively.

### Help

Instantly view any object's docs, and render your own.

**FEASIBILITY STUDY**

The feasibility study is carried out to test whether the proposed system is worth being implemented. The proposed system will be selected if it is best enough in meeting the performance requirements.

The feasibility carried out mainly in three sections namely.

**•** Economic Feasibility

• Technical Feasibility

• Behavioural Feasibility

**Economic Feasibility**

Economic analysis is the most frequently used method for evaluating effectiveness of the proposed system. More commonly known as cost benefit analysis. This procedure determines the benefits and saving that are expected from the system of the proposed system. The hardware in system department if sufficient for system development.

**Technical Feasibility**

This study centre around the system’s department hardware, software and to what extend it can support the proposed system department is having the required hardware and software there is no question of increasing the cost of implementing the proposed system. The criteria, the proposed system is technically feasible and the proposed system can be developed with the existing facility.

**Behavioural Feasibility**

People are inherently resistant to change and need sufficient amount of training, which would result in lot of expenditure for the organization. The proposed system can generate reports with day-to-day information immediately at the user’s request, instead of getting a report, which doesn’t contain much detail.

**5.4 TESTING OF PRODUCT**

**Testing of Product**

System testing is the stage of implementation, which aimed at ensuring that system works accurately and efficiently before the live operation commence. Testing is the process of executing a program with the intent of finding an error. A good test case is one that has a high probability of finding an error. A successful test is one that answers a yet undiscovered error.

Testing is vital to the success of the system.  System testing makes a logical assumption that if all parts of the system are correct, the goal will be successfully achieved.  The candidate system is subject to variety of tests-on-line response, Volume Street, recovery and security and usability test.  A series of tests are performed before the system is ready for the user acceptance testing.  Any engineered product can be tested in one of the following ways.  Knowing the specified function that a product has been designed to from, test can be conducted to demonstrate each function is fully operational.  Knowing the internal working of a product, tests can be conducted to ensure that “al gears mesh”, that is the internal operation of the product performs according to the specification and all internal components have been adequately exercised.

**UNIT TESTING**

Unit testing is the testing of each module and the integration of the overall system is done.  Unit testing becomes verification efforts on the smallest unit of software design in the module.  This is also known as ‘module testing’.  The modules of the system are tested separately.  This testing is carried out during the programming itself.  In this testing step, each model is found to be working satisfactorily as regard to the expected output from the module.  There are some validation checks for the fields.  For example, the validation check is done for verifying the data given by the user where both format and validity of the data entered is included.  It is very easy to find error and debug the system.

**INTEGRATION TESTING**

Data can be lost across an interface, one module can have an adverse effect on the other sub function, when combined, may not produce the desired major function.  Integrated testing is systematic testing that can be done with sample data.  The need for the integrated test is to find the overall system performance. There are two types of integration testing. They are:

1. Top-down integration testing.
2. Bottom-up integration testing.

**WHITE BOX TESTING**

White Box testing is a test case design method that uses the control structure of the procedural design to drive cases.  Using the white box testing methods, we derived test cases that guarantee that all independent paths within a module have been exercised at least once.

**BLACK BOX TESTING**

* Black box testing is done to find incorrect or missing function
* Interface error
* Errors in external database access
* Performance errors
* Initialization and termination errors

In ‘functional testing’, is performed to validate an application conforms to its specifications of correctly performs all its required functions. So this testing is also called ‘black box testing’.  It tests the external behaviour of the system.  Here the engineered product can be tested knowing the specified function that a product has been designed to perform, tests can be conducted to demonstrate that each function is fully operational.

**VALIDATION TESTING**

After the culmination of black box testing, software is completed assembly as a package, interfacing errors have been uncovered and corrected and final series of software validation tests begin validation testing can be defined as many, but a single definition is that validation succeeds when the software functions in a manner that can be reasonably expected by the customer.

# USER ACCEPTANCE TESTING

User acceptance of the system is the key factor for the success of the system.  The system under consideration is tested for user acceptance by constantly keeping in touch with prospective system at the time of developing changes whenever required.

# OUTPUT TESTING

After performing the validation testing, the next step is output asking the user about the format required testing of the proposed system, since no system could be useful if it does not produce the required output in the specific format.  The output displayed or generated by the system under consideration.  Here the output format is considered in two ways.  One is screen and the other is printed format.  The output format on the screen is found to be correct as the format was designed in the system phase according to the user needs.  For the hard copy also output comes out as the specified requirements by the user. Hence the output testing does not result in any connection in the system.

**Agile Testing**

Agile Testing is a type of software testing that accommodates agile software development approach and practices. In an Agile development environment, testing is an integral part of software development and is done along with coding. Agile testing allows incremental and iterative coding and testing.

**API Testing**

API testing is a type of testing that is similar to unit testing. Each of the Software APIs are tested as per API specification. API testing is mostly done by testing team unless APIs to be tested or complex and needs extensive coding. API testing requires understanding both API functionality and possessing good coding skills.

**Automated testing**

This is a testing approach that makes use of testing tools and/or programming to run the test cases using software or custom developed test utilities. Most of the automated tools provided capture and playback facility, however there are tools that require writing extensive scripting or programming to automate test cases.

**End-to-end Testing**

End to end testing is performed by testing team, focus of end to end testing is to test end to end flows e.g. right from order creation till reporting or order creation till item return etc. and checking. End to end testing is usually focused mimicking real life scenarios and usage. End to end testing involves testing information flow across applications.

**Exploratory Testing**

Exploratory testing is an informal type of testing conducted to learn the software at the same time looking for errors or application behaviour that seems non-obvious. Exploratory testing is usually done by testers but can be done by other stake holders as well like Business Analysts, developers, end users etc. who are interested in learning functions of the software and at the same time looking for errors or behaviour is seems non-obvious.

**Performance Testing**

**It** is a type of software testing and part of performance engineering that is performed to check some of the quality attributes of software like Stability, reliability, availability. Performance testing is carried out by performance engineering team. Unlike Functional testing, Performance testing is done to check non-functional requirements. Performance testing checks how well software works in anticipated and peak workloads. There are different variations or sub types of performance like load testing, stress testing, volume testing, soak testing and configuration testing.

**Penetration Testing**

**It** is a type of security testing, also known as pen test in short. Penetration testing is done to tests how secure software and its environments (Hardware, Operating system and network) are when subject to attack by an external or internal intruder. Intruder can be a human/hacker or malicious programs. Pen test uses methods to forcibly intrude (by brute force attack) or by using a weakness (vulnerability) to gain access to a software or data or hardware with an intent to expose ways to steal, manipulate or corrupt data, software files or configuration. Penetration Testing is a way of ethical hacking, an experienced Penetration tester will use the same methods and tools that a hacker would use but the intention of Penetration tester is to identify vulnerability and get them fixed before a real hacker or malicious program exploits it.

**Security Testing**

**It** is a type of software testing carried out by specialized team of software testers. Objective of security testing is to secure the software is to external or internal threats from humans and malicious programs. Security testing basically checks, how good is software’s authorization mechanism, how strong is authentication, how software maintains confidentiality of the data, how does the software maintain integrity of the data, what is the availability of the software in an event of an attack on the software by hackers and malicious programs is for Security testing requires good knowledge of application, technology, networking, security testing tools. With increasing number of web applications necessarily of security testing has increased to a greater extent.

**Sanity Testing**

**It** is a type of testing that is carried out mostly by testers and in some projects by developers as well. Sanity testing is a quick evaluation of the software, environment, network, external systems are up & running, software environment as a whole is stable enough to proceed with extensive testing. Sanity tests are narrow and most of the time sanity tests are not documented.

**Scalability Testing**

**It** is a non-functional test intended to test one of the software quality attributes i.e. “Scalability”. Scalability test is not focused on just one or few functionality of the software instead performance of software as a whole. Scalability testing is usually done by performance engineering team. Objective of scalability testing is to test the ability of the software to scale up with increased users, increased transactions, increase in database size etc., It is not necessary that software’s performance increases with increase in hardware configuration, scalability tests helps to find out how much more workload the software can support with expanding user base, transactions, data storage etc.,

**Stability Testing**

**It** is a non-functional test intended to test one of the software quality attributes i.e. “Stability”. Stability testing focuses on testing how stable software is when it is subject to loads at acceptable levels, peak loads, loads generated in spikes, with more volumes of data to be processed. Scalability testing will involve performing different types of performance tests like load testing, stress testing, spike testing, soak testing, spike testing etc.…

**Static Testing** is a form of testing where in approaches like reviews, walkthroughs are employed to evaluate the correctness of the deliverable. In static testing software code is not executed instead it is reviewed for syntax, commenting, naming convention, size of the functions and methods etc. Static testing usually has check lists against which deliverables are evaluated. Static testing can be applied for requirements, designs, and test cases by using approaches like reviews or walkthroughs.

**Stress Testing** is a type of performance testing, in which software is subjected to peak loads and even to a break point to observe how the software would behave at breakpoint. Stress testing also tests the behaviour of the software with insufficient resources like CPU, Memory, Network bandwidth, Disk space etc. Stress testing enables to check some of the quality attributes like robustness and reliability.

1. **CONCLUSION** **AND** **FUTURE** **ENHANCEMENT**

**6.1 CONCLUSION**

From the analysis, these algorithms and process, we concluded that our hypothesis is true. Meaning you can predict the app ratings, however significant preprocessing must be done before you start the classification and regression processes. The Play Store apps data has enormous potential to drive app-making businesses to success. Actionable insights can be drawn for developers to work on and capture the Android market! This shows that given the Size, Type, Price, Content Rating, and Genre of an app, we can predict accuracy if an app will have more than 100,000 installs and be a hit on the Google Play Store. User reviews are limited to identifying polarity and subjectivity. However, the massive increase in review based data implies a requirement to focus also on performing predictions. This process is challenging yet fruitful, as user reviews are qualitative while ratings are essentially quantitative. The numeric scoring of apps within the Google App store could also be biased and overrated because higher ratings given by users potentially attract several new users disproportionately.

* 1. **FUTURE ENHANCEMENT**

In the future, ensemble classifiers were investigated to guage their performance on the reviews scraped from the Google App store. Also includes the implementation of the deep learning technique to predict numeric rating.

1. **SAMPLE** **CODING** **AND** **SAMPLE** **SCREENSHOT**

**CODING**

#Importing required libraries

import pandas as pd

import matplotlib.pyplot as plt

import numpy as np

import seaborn as sns

from itertools import cycle, islice

from sklearn import preprocessing

from sklearn.model\_selection import train\_test\_split

from sklearn.metrics import mean\_absolute\_error, mean\_squared\_error, r2\_score

df = pd.read\_csv('googleplaystore.csv')

print(df.info())

print(df['Category'].value\_counts())

#Data Visualization

'''total no of Apps per Category'''

plt.figure()

my\_colors = list(islice(cycle(['b', 'r', 'g', 'c', 'y', 'm','k']), None, len(df))) # a way to represent different colours

df.Category.value\_counts().plot(kind = 'bar', color = my\_colors, figsize = (10,8), title = 'Number of apps in each category');

plt.xlabel('Category')

plt.ylabel('Frequency');

# Piechart

plt.figure()

df['Category'].value\_counts().plot.pie(y = df['Category'], figsize = (15, 16), label = '', autopct = '%1.1f%%', title = 'Distribution of apps by category', );

'''Ratings'''

plt.figure()

# We clean the Rating column with the help of lambda function

df['Rating'] = df['Rating'].apply(lambda x: str(x).replace('nan', 'NaN') if 'nan' in str(x) else str(x)) # Replace nan with NaN

df['Rating'] = df['Rating'].apply(lambda x: float(x)) # Rewrite column in a float format

plt.figure()

df['Rating'] = df['Rating'].fillna(df['Rating'].median()) # Replace null numbers with median numbers

df.Rating.isnull().sum()

plt.figure()

df.Rating.value\_counts().plot(kind = 'bar', stacked = True, figsize = (12, 8), title = 'Distribution of Rating among apps available on Google Play Store'); # Historgram of frequencies

plt.xlabel('Rating')

plt.ylabel('Frequencies');

#Lets remove the comma & the plus sign

plt.figure()

df2=df.copy()

df2["Installs"]=df["Installs"].str.replace(",","").str[:-1]

df2["Installs"]=pd.to\_numeric(df2["Installs"],errors='coerce')

ginstalls=df2.groupby("Category").sum()["Installs"].sort\_values().head(10)

ginstalls.plot(kind="barh")

plt.gca().spines["right"].set\_visible(False)

plt.gca().spines["top"].set\_visible(False)

plt.title("Installs by category")

#Analysing the family category

plt.figure()

family\_category=df2[df2["Category"]=="FAMILY"]

fam=family\_category.groupby("Genres").size().sort\_values(ascending=False).head(5)

fam.plot(kind="bar",color="lightblue")

plt.title("Family Category (Number of apps)")

plt.ylabel("Number of Apps")

plt.figure()

genres=["Education","Simulation","Entertainment","Puzzle","Casual"]

fam2=family\_category[family\_category["Genres"].isin(genres)]

groupfam2=fam2.groupby("Genres").sum()["Installs"]

groupfam2.plot(kind="bar",color="pink")

#Ratings & Categories

plt.figure()

Catratings=df.groupby("Category").mean().drop("1.9").head(10)

print(Catratings)

plt.figure()

colors=["lightblue" if (x<max(Catratings["Rating"])) else "blue" for x in Catratings["Rating"]]

plt.bar(height=Catratings["Rating"],x=Catratings.index,color=colors)

plt.xticks(rotation='vertical')

plt.title("Top 10 Categories by Ratings")

plt.gca().spines["top"].set\_visible(False)

plt.gca().spines["right"].set\_visible(False)

plt.figure()

df['Type'].fillna(str(df['Type'].mode().values[0]), inplace = True)

df.Type.isnull().sum()

plt.figure()

df.Type.value\_counts().plot.pie(y = df.Type, figsize = (8, 10), autopct = '%1.1f%%', title = 'Ratio of Free and Paid apps in the market',label = '');

#Content Rating

plt.figure()

print(df['Content Rating'].unique())

df['Content Rating'].value\_counts().plot(kind = 'bar', title = 'Apps by Content Rating', color = my\_colors, figsize = (8,4))

plt.xlabel('Category')

plt.ylabel('Number of installs')

plt.yscale('log');

df.isnull().sum()

df.fillna(0)

#Heatmap

plt.figure(figsize=(12, 8))

corr = df.apply(lambda x: pd.factorize(x)[0]).corr()

ax = sns.heatmap(corr, xticklabels=corr.columns, yticklabels=corr.columns, annot=True, linewidths=.2, cmap='coolwarm', vmin=-1, vmax=1)

plt.show()

df.isnull().sum()

df.dropna(inplace = True)

df.info()

df.drop(labels = ['Current Ver','Android Ver','App'], axis = 1, inplace = True)

df.head()

category\_list = df['Category'].unique().tolist()

category\_list = ['cat\_' + word for word in category\_list]

df = pd.concat([df, pd.get\_dummies(df['Category'], prefix='cat')], axis=1)

df.head()

#Label Encoding

le = preprocessing.LabelEncoder()

df['Genres'] = le.fit\_transform(df['Genres'])

le = preprocessing.LabelEncoder()

df['Content Rating'] = le.fit\_transform(df['Content Rating'])

df['Price'] = df['Price'].apply(lambda x : x.strip('$'))

df['Installs'] = df['Installs'].apply(lambda x : x.strip('+').replace(',', ''))

df['Type'] = pd.get\_dummies(df['Type'])

df["Size"] = [str(round(float(i.replace("k", ""))/1024, 3)) if "k" in i else i for i in df.Size]

df['Size'] = df['Size'].apply(lambda x: x.strip('M'))

df[df['Size'] == 'Varies with device'] = 0

df['Size'] = df['Size'].astype(float)

df["Size"]

df['new'] = pd.to\_datetime(df['Last Updated'])

df['lastupdate'] = (df['new'] - df['new'].max()).dt.days

#Modelling

x = df.drop(labels=["Rating","Category", "Last Updated", "new"], axis = 1)

y = df['Rating']

#Split the data into train and test

x\_train, x\_test, y\_train, y\_test = train\_test\_split(x, y, test\_size = 0.30, random\_state = 42)

#Apply Algorithms

'''KNN Regression'''

print()

print('------KNN Regression------')

from sklearn.neighbors import KNeighborsRegressor

knn = KNeighborsRegressor(n\_neighbors=50)

accuracy = knn.score(x\_test,y\_test)

print('KNN Accuracy: ' + str(np.round(accuracy\*100, 2)) + '%')

knn\_pred = knn.predict(x\_test)

mae = mean\_absolute\_error(y\_test, knn\_pred)

print('KNN MAE:', mae)

rmse = np.sqrt(mean\_squared\_error(y\_test, knn\_pred))

print('KNN RMSE:', rmse)

R2\_score = r2\_score(y\_test, knn\_pred)

print('KNN R2\_Score:', R2\_score)

n\_neighbors = np.arange(1, 50, 1)

scores = []

for n in n\_neighbors:

knn.set\_params(n\_neighbors=n)

knn.fit(x\_train, y\_train)

scores.append(knn.score(x\_test, y\_test))

plt.figure(figsize=(7, 5))

plt.title("Effect of Estimators")

plt.xlabel("Number of Neighbors K")

plt.ylabel("Score")

plt.plot(n\_neighbors, scores)

'''Random Forest Regression'''

print()

print('------Random Forest Regression------')

from sklearn.ensemble import RandomForestRegressor

regressor = RandomForestRegressor(n\_estimators=20, random\_state=0)

accuracy = regressor.score(x\_test,y\_test)

print('RF Accuracy: ' + str(np.round(accuracy\*100, 2)) + '%')

rf\_pred = regressor.predict(x\_test)

mae = mean\_absolute\_error(y\_test, rf\_pred)

print('RF MAE:', mae)

rmse = np.sqrt(mean\_squared\_error(y\_test, rf\_pred))

print('RF RMSE:', rmse)

R2\_score = r2\_score(y\_test, rf\_pred)

print('RF R2\_Score:', R2\_score)

estimators = np.arange(10, 150, 10)

scores = []

for n in estimators:

regressor.set\_params(n\_estimators=n)

regressor.fit(x\_train, y\_train)

scores.append(regressor.score(x\_test, y\_test))

plt.figure(figsize=(7, 5))

plt.title("Effect of Estimators")

plt.xlabel("no. estimator")

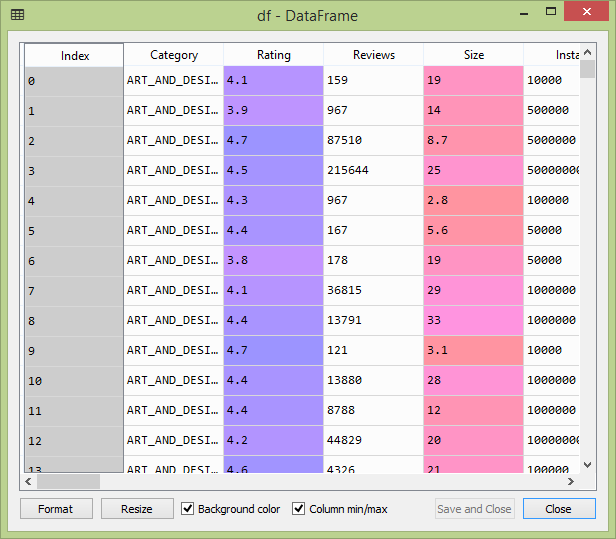
plt.ylabel("score")

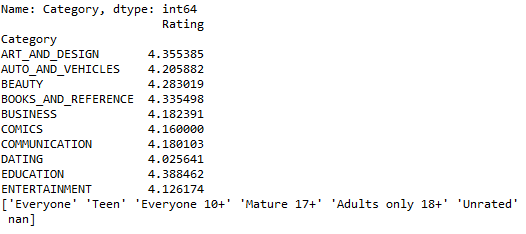
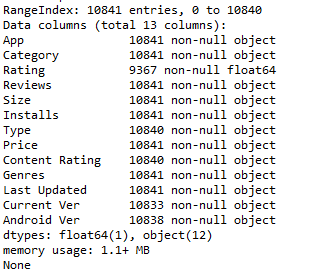
plt.plot(estimators, scores)

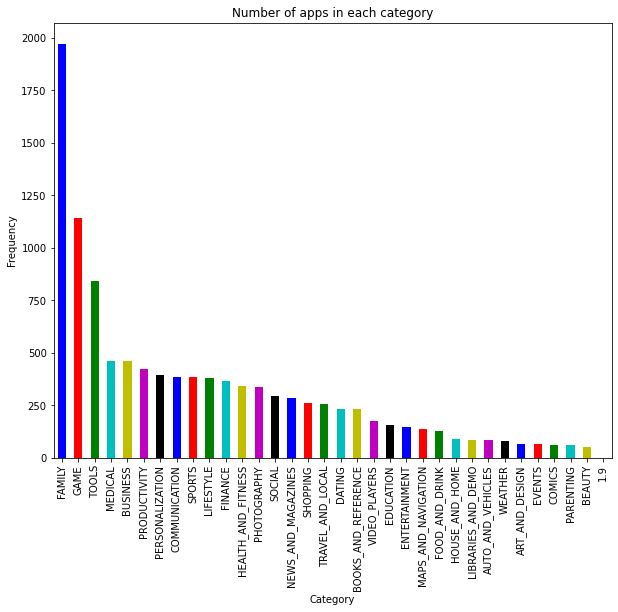
print("max accuracy is: ", max(scores))

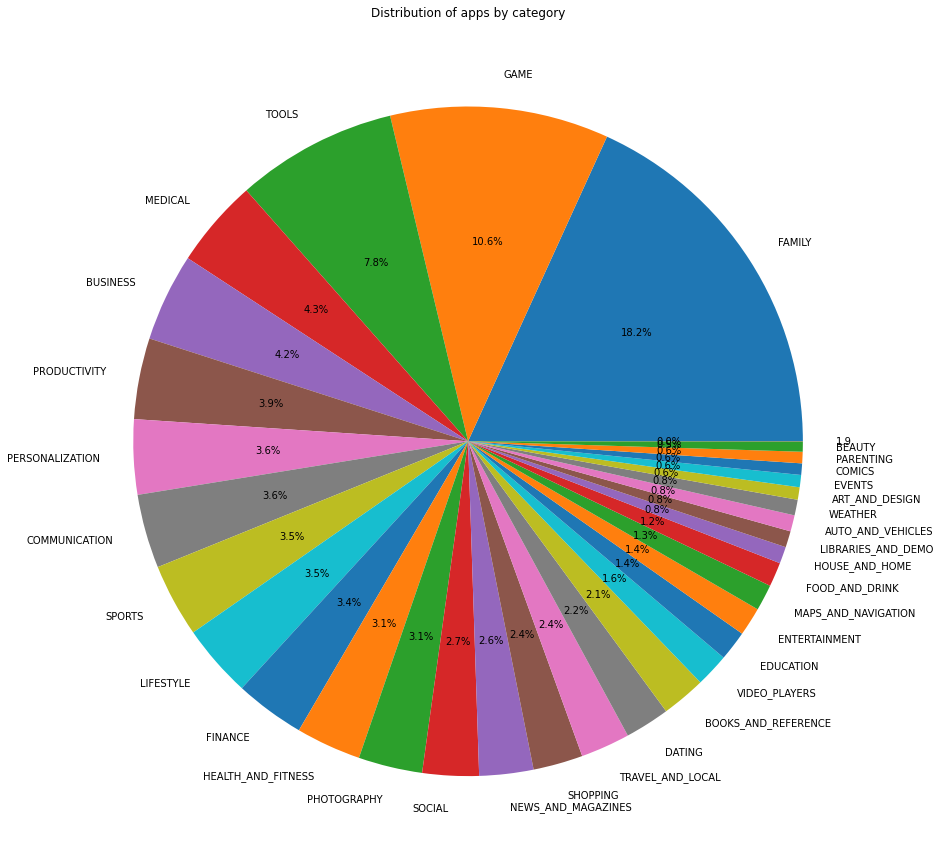
print("the number of estimators required to achieve this result: ", estimators[scores.index(max(scores))])

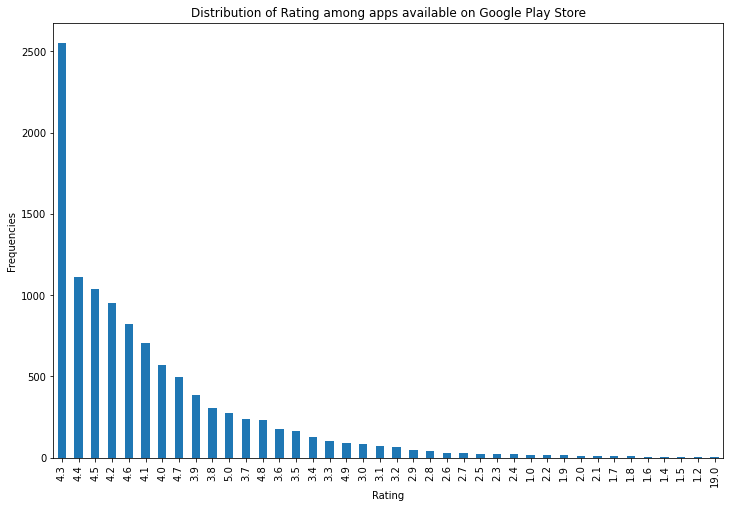
**SCREENSHOTS:**

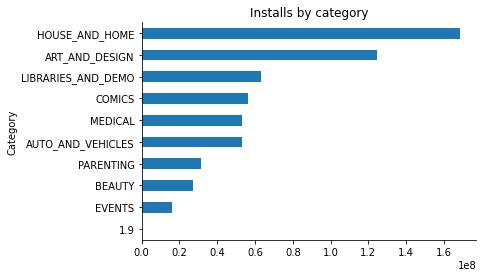


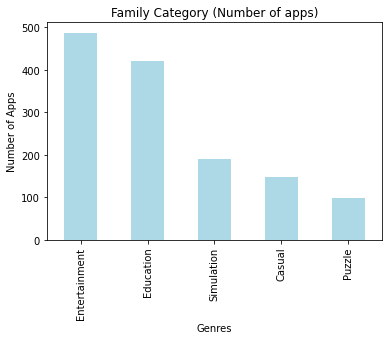


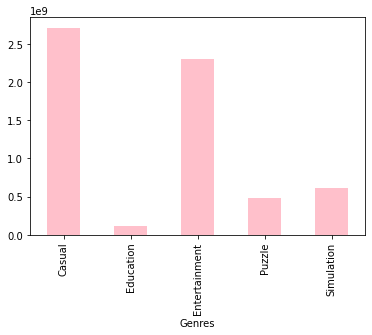


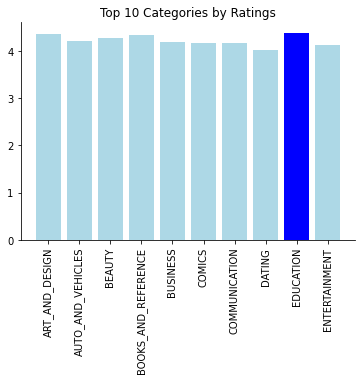


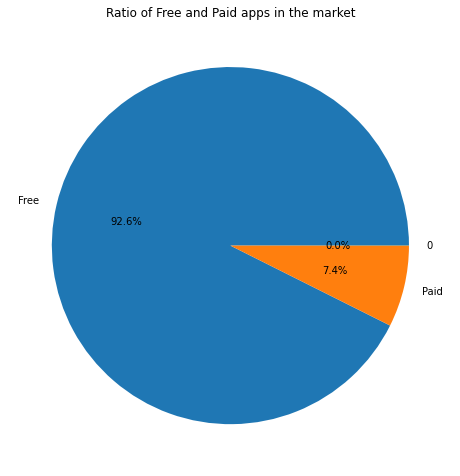


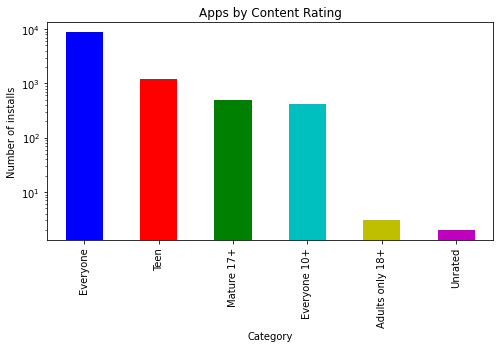


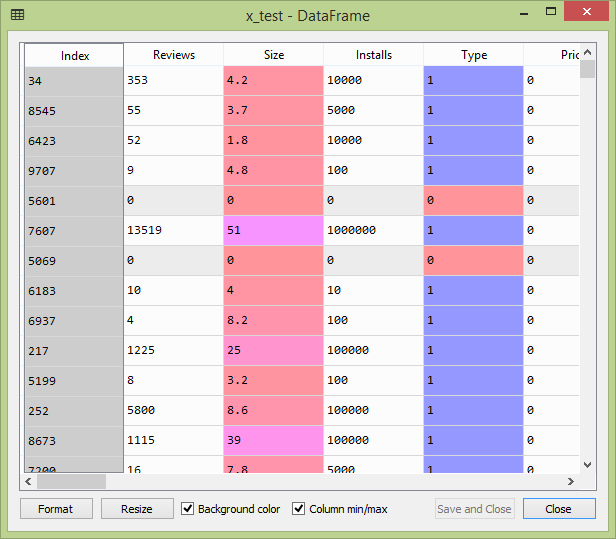
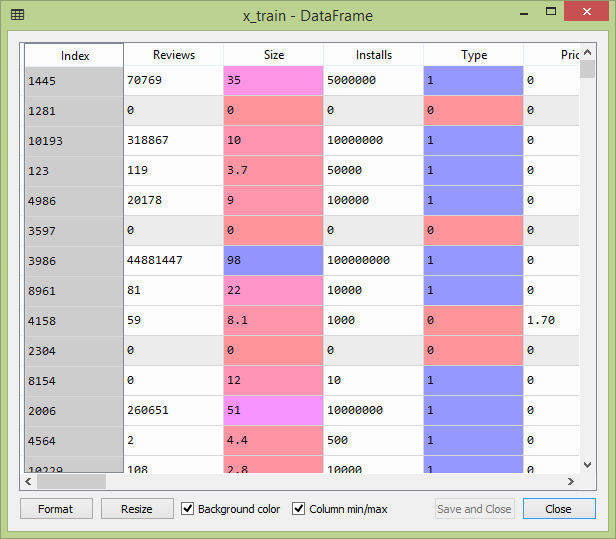
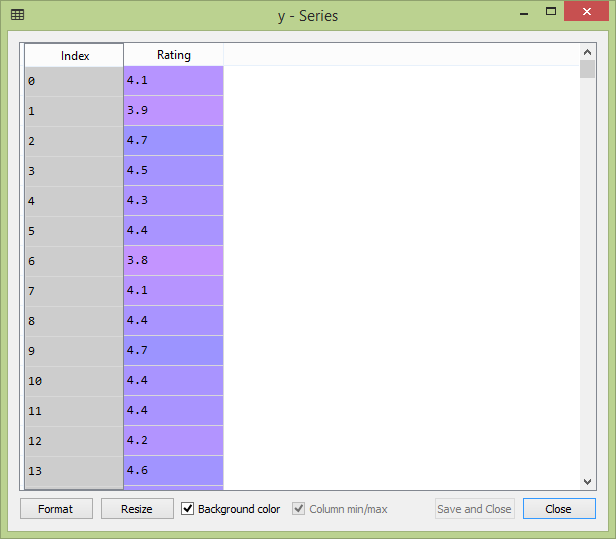
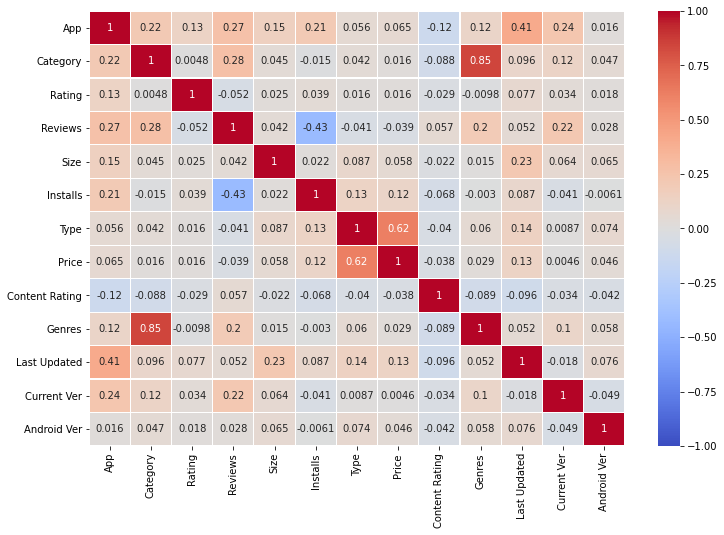


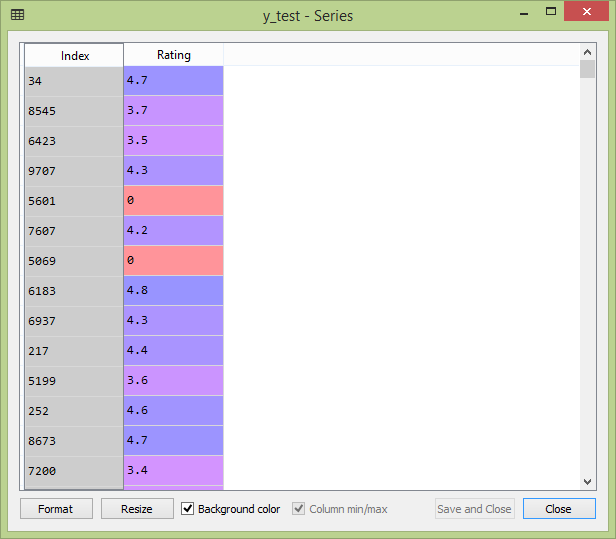
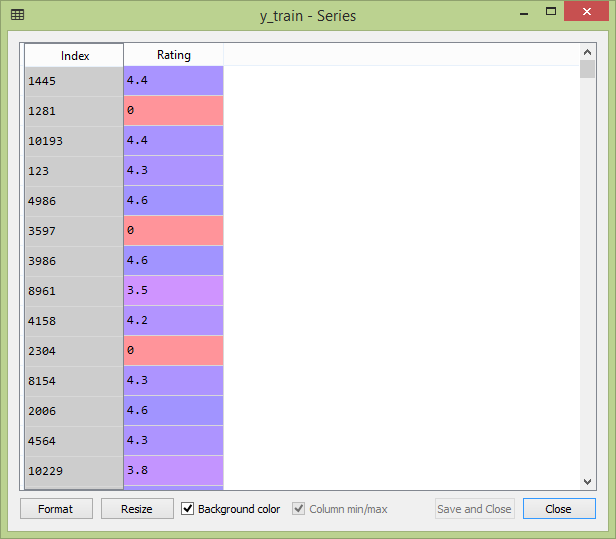


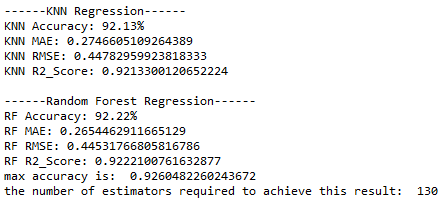


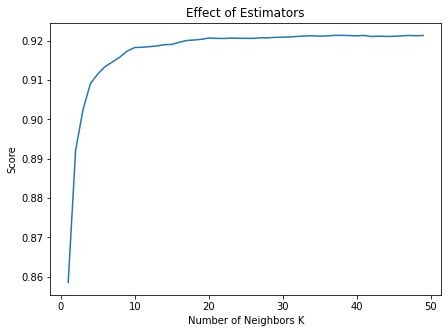


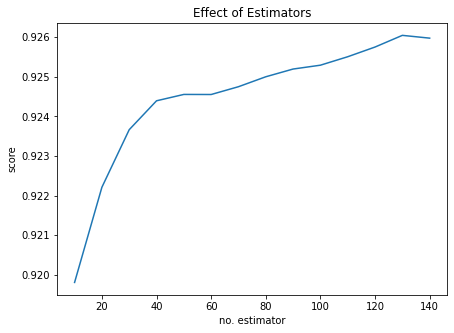












1. **REFERENCES**

[1] C. J. Burke, W. R. Walter, S. Gaddam, H. Pham, J. S. Babb, J. Sanger, and F. Ponzo, “Correlation of benign incidental findings seen on whole-body pet-ct with knee mri: patterns of 18f-fdg avidity, intra-articular pathology, and bone marrow edema lesions,” Skeletal Radiology, vol. 47, no. 12, pp. 1651–1660, Dec 2018.

[2] Z. Zhang, “Introduction to machine learning: k-nearest neighbors,” Annals of translational medicine, vol. 4, no. 11, 2016.

[3] Adailton Vieira , Marino Luiz EyerkauferI, Rodrigo Renge, “Risk analysis tool for granting credit by financial cooperatives for legal entities,” ˜ RGC, Santa Maria, v. 7, n. 13, Jan./Jun., 2020.

[4] K. R. Santos, I. R. R. Silva, and R. A. A. Fagundes, “Classifiers comparison for attack detection in computer networks,” IEEE Latin America Transactions, vol. 15, no. 1, pp. 87–96, 2017.

[5] M. R. Islam, “Numeric rating of apps on google play store by sentiment analysis on user reviews,” in 2014 International Conference on Electrical Engineering and Information & Communication Technology. IEEE, apr 2014.

[6] N. Mohd Nawi, W. Atomi, and S. M. Rehman Gillani, “The effect of data pre-processing on optimized training of artificial neural networks,” vol. 11, 06 2013.

[7] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay, “Scikit-learn: Machine learning in Python,” Journal of Machine Learning Research, vol. 12, pp. 2825–2830, 2011.

[8] T. Thiviya et al., Mobile apps'feature extraction based on user reviews using machine learning, 2019.

[9] H. S. Le, T. V. Le, and T. V. Pham, Aspect analysis for opinion mining of vietnamese text, in Proc. Int. Conf. Adv. Comput. Applicat. (Ho Chi Minh, Vietnam), Nov. 2015, pp. 118–123.

[10] H. Wang, L. Yue, and C. Zhai, Latent aspect rating analysis on review text data: a rating regression approach, in Proc. ACM SIGKDD Int. Conf. Knowledge Discovery Data Mining (Washington, D.C., USA), July 2010, pp. 783–792.