Project Report: App Rating prediction

Course Title: Big Data and IoT Lab
Course Code: CSE 413

Submitted to:

Md. Zahim Hassan
Lecturer
Department of Computer Science & Engineering
Daffodil International University

Submitted by:

NAME	ID
Hasibul Hasan Chowdhury Mahi	201-15-14184
Md Siam Khan Chowdhury	201-15-13992
Tasnova Rebonya	201-15-14170

Date of Submission: 21th November,2023

Abstract:

To effectively develop new software, adherence to established implementation standards is crucial. The challenges in creating applications for marketplaces, especially concerning client acceptance and sales, highlight the significance of analyzing key metrics available in app stores. These metrics include download counts, user comments, and app ratings, providing valuable insights into patterns of app success.

In response to these challenges, the intention is to build two inference engines employing Support Vector Machine (SVM) and Random Forest algorithms. The focus is on researching attributes that exhibit strong correlations with app ratings. To achieve this, a comprehensive analysis will be conducted using a dataset sourced from the Google Play Store. The goal is to calculate and assess regression metrics, allowing for a deeper understanding of the factors influencing app ratings and aiding in the formulation of effective problem-solving strategies.

Introduction:

In this project we discussion shifts to the impact of smartphones on personal and professional life, with a focus on the abundance of apps available on the Google Play store. The significance of user reviews and numerical ratings in influencing app adoption is emphasized, citing statistics on app downloads and their projected increase. In there points out the impact of ratings on product pricing and the mutual benefits of shared ratings, problem reports, and reviews for both users and developers.

Finally, it addresses the challenges in addressing issues related to user reviews' polarity, particularly the limitations of solutions that only consider extremes in polarity without accounting for numerical inconsistencies between ratings and reviews.

Data Analysis:

Mobile apps are vital for consumers and drive innovation in advertising. Intense global competition among mobile app designers underscores the need for effective market strategies. Despite dominating downloads, the Google Play Store generates less revenue than the Apple App Store, prompting focused investigation.

Mobile technology integration has made apps integral to daily life, with the Google Play App Store offering 0.675 million Android applications. The belief is that online reviews influence paid apps, posing challenges for users navigating comments and ratings. Application engineers face difficulties improving program performance based on general evaluations.

Automated intelligence, particularly machine learning in supervised and unsupervised forms, is briefly introduced. Supervised learning involves training computers with labeled datasets, while unsupervised learning enables machines to find patterns without explicit instruction.

In there semi-supervised machine learning, combining supervised and unsupervised methods using labeled datasets for training, offering a balanced learning approach.

Related Works:

Predictive Modeling for App Ratings:

Explore research papers or articles that discuss predictive modeling for app ratings. Look for studies that employ machine learning techniques, especially regression models, to predict user ratings based on diverse features.

Feature Engineering in Mobile App Analysis:

Investigate works that focus on feature engineering for mobile app analysis. Effective feature selection and engineering can significantly impact the performance of predictive models.

Sentiment Analysis in User Reviews:

Explore studies that incorporate sentiment analysis of user reviews. Understanding user sentiment from reviews can provide valuable insights into user satisfaction and influence app ratings.

Handling Missing Data in Predictive Modeling:

Given that the code includes handling missing data through imputation, look for related works on strategies for dealing with missing data in predictive modeling, especially in the context of mobile app data.

Categorical Data Encoding in Machine Learning:

Since the code includes encoding categorical variables using StringIndexer, explore works that discuss different techniques for handling categorical data in machine learning models.

Code Brief Explanation:

Importing Libraries and Data:

Installing PySpark:

there we uses the !pip command to install the PySpark library, which is a Spark API for Python. PySpark allows you to interact with Apache Spark, a distributed data processing engine.

Mounting Google Drive:

This snippet mounts your Google Drive to the Colab environment. This is useful for accessing files stored in your Google Drive.

Importing necessary PySpark modules:

Here, the code imports various PySpark modules needed for building and evaluating a linear regression model.

Creating a SparkSession:

There initializes a SparkSession named "playstore." A SparkSession is the entry point for interacting with Spark functionality.

Reading Data from CSV:

It reads a CSV file ('Playstore_final.csv') into a PySpark DataFrame named data. The option('header','true') indicates that the first row of the CSV file contains the header.

Displaying Data:

displays the contents of the data DataFrame.

Dropping Unnecessary Columns:

It creates a new DataFrame data_drop by removing specific columns from the original DataFrame data. The dropped columns seem to be related to app identifiers, developer information, version details, and other metadata.

Displaying Modified Data:

In this command displays the contents of the modified DataFrame (data_drop), which no longer contains the dropped columns.

Dataset Preprocessing:

This code segment focuses on handling missing values, cleaning specific columns, and ensuring data consistency in a PySpark DataFrame.

Counting Null Values:

This loop iterates through each column in the DataFrame data_drop and counts the number of null values in each column, printing the results. This provides an overview of missing data in the dataset.

Dropping Rows with Null Values:

This line creates a new DataFrame data_null_removed by removing rows with any null values from the original DataFrame data_drop.

Verifying Null Value Counts after Dropping Rows:

Similar to the first loop, this loop verifies the null value counts in each column after removing rows with null values. It prints the updated counts.

Showing the Modified DataFrame:

This command displays the contents of the DataFrame after dropping rows with null values.

Cleaning 'Installs' Column:

This segment removes commas and plus signs from the 'Installs' column, assuming it contains numeric values with commas.

Cleaning 'Size' Column:

This part removes the 'M' character from the 'Size' column, assuming it contains sizes with 'M' as a unit.

Removing Rows with "N/A" Values:

This loop iterates through all columns and removes rows where the values are "N/A" or " N/A".

Counting "N/A" Values in Each Column:

This part calculates and displays the counts of "N/A" values in each column of the final DataFrame (data_remove_nan).

Label Encoding Categorical Columns:

This code performs encoding of categorical columns, conversion of specified columns to DoubleType, and imputation of missing values with mean values. It aims to prepare the data for further analysis or model training.

This code segment focuses on further preprocessing of the PySpark DataFrame. Let's break down each part:

StringIndexer for Categorical Columns:

This section encodes categorical columns using the StringIndexer transformer, which assigns numerical indices to categorical values. The resulting DataFrame is displayed.

Null Value Count after Encoding:

This loop calculates and prints the null value counts for each column in the DataFrame after encoding.

Converting Columns to DoubleType:

This part converts specific columns to DoubleType, assuming they represent numeric values.

Null Value Count after Conversion:

This loop calculates and prints the null value counts for each column in the DataFrame after converting columns to DoubleType.

Imputing Missing Values with Mean:

This part uses the Imputer transformer to replace missing values in specified columns with the mean of each column.

Null Value Count after Imputation:

This loop calculates and prints the null value counts for each column in the DataFrame after imputing missing values.

Applying Linear Regression:

This code prepares the data, splits it into training and testing sets, initializes and trains a linear regression model, and finally prints the coefficients and intercept of the trained model.

Define Independent Features and Target Column:

This section defines the independent feature columns and the target column for the linear regression model.

Drop Rows with Null Values in Specific Columns:

This line drops rows with null values in the specified independent feature columns and the target column.

Create a Vector Assembler:

This part uses VectorAssembler to combine the independent feature columns into a single feature vector named "Independent_Features."

Select Feature Vector and Target Column:

It selects the feature vector and the target column for further processing.

Split the Data into Training and Testing Sets:

This line randomly splits the data into training (80%) and testing (20%) sets.

Initialize Linear Regression Model

Fit the Model to Training Data:

It fits the linear regression model to the training data.

Access Model Coefficients and Intercept

Print Coefficients and Intercept:

This line prints the coefficients and intercept of the trained linear regression model.

Make Predictions on Test Data:

This line applies the trained linear regression model (regression_model) to the test data (test_data) and generates predictions.

Show Predictions:

This command selects and displays relevant columns from the prediction results, including the original independent features, the actual target values (Rating), and the predicted values.

The output will be a table showing the original features, the actual target values, and the predicted values for each corresponding row in the test data.

This code evaluates the performance of the trained linear regression model by making predictions on the test data and displaying the results.

```
# Show the predictions
predict_result.select("Independent_Features", target_col, "prediction").show()
```

```
+--------------------+---------+------------------+
|Independent_Features|   Rating|        prediction|
+--------------------+---------+------------------+
|(14,[0,1,2,3,4,5,...|      4.5|  4.15922960340566|
|(14,[0,1,2,3,4,5,...|     3.36| 4.171236221633009|
|(14,[0,1,2,3,4,5,...|      3.7| 4.161988279658477|
|(14,[0,1,2,3,4,5,...|4.3366337| 4.173640820026883|
|(14,[0,1,2,3,4,5,...|4.2954545| 4.172188730156745|
|(14,[0,1,2,3,4,5,...|4.3989525| 4.172761751674257|
|(14,[0,1,2,3,4,5,...|      4.8|4.0542581764049865|
|(14,[0,1,2,4,5],[...|      4.2| 4.182561399552659|
|(14,[0,1,2,4,5,6]...|      4.2| 4.172668619300984|
|(14,[0,1,2,4,5,6]...|      4.2| 4.175583486421332|
|(14,[0,1,2,4,5,6]...|      4.0|  4.19843866852586|
|(14,[0,1,2,4,5,6]...|      4.6| 4.173443081382994|
|(14,[0,1,2,4,5,6]...|      4.4| 4.175837632488977|
|(14,[0,1,2,4,5,6]...|      4.8|  4.14125830054908|
|(14,[0,1,2,4,5,6]...|      4.6| 4.175583448538538|
|(14,[0,1,2,4,5,6]...|      5.0| 4.172767040536329|
|(14,[0,1,2,4,5,6]...|      3.8| 4.198438755290491|
|(14,[0,1,2,4,5,6]...|      5.0| 4.198487922525849|
|(14,[0,1,2,4,5,6]...|      4.4|4.1815049741421095|
|(14,[0,1,2,4,5,6]...|      4.4|4.1986923810493675|
+--------------------+---------+------------------+
only showing top 20 rows
```

Analysis of Results

Only polarity and subjectiveness may be obtained from user reviews. Predictions are also important because of the enormous

expansion in review-based data. This is a challenging but rewarding process, as user reviews are qualitative and ratings are mainly quantitative. Additionally, Google's numerical rating system may be distorted and amplified by the fact that higher ratings supplied by consumers may bring in disproportionately more new users. So this study investigated if ensemble classifiers might be used to predict numerical ratings for Google Play store apps based on user reviews. The Google App store assessments were used to test many ensemble classifiers. To predict numerical ratings in the future, deep learning technology will be applied Future Works:

Conclusion:

We came to the conclusion that our hypothesis is correct after running through all of these algorithms and processes. As a result, it is possible to predict app ratings, but a large amount of preprocessing is required before the classification and regression processes can be started. The data collected from Google Play Store apps has huge potential to help app development companies succeed. Developers can use the information to their advantage to work on and conquer the Android market! In order to accurately estimate whether an app will have more than 100,000 downloads and be a success on the Google Play Store, we need to know the app's Size, Type, Price, Content Rating, and Genre.

References:

[1] Statista, Numerous accessible applications within the Google Play store from December 2009 to March 2019,

https://www.statista.com/statistics/266210/numberof-available-applications-in-the-google-play-store/, Online: accessed twentytwo May 2019.

[2] Statista, various mobile app downloads worldwide in 2017, 2018, and 2020 (in billions), https://www.statista.com/statistics/

271644/worldwide-free-and-paid-mobile-app-store downloads/, Online: accessed twenty-two May 2019.

[3] Online shopping, pew internet, and American life project, Washington, DC, 2018, http://www.pewinternet.org/Repor

ts/2008/online shopping/01-Summary-of-Findings.aspx Online: accessed eight August. 2014.

[4] D. Pagano and W. Maalej, User feedback within the AppStore: an empirical study, in Proc. IEEE Int. Requirements Eng. Conf.

(Rio de Janeiro, Brazil), July 2013, pp. 125–134.

[5] T. Chumwatana, Using sentiment analysis technique for analyzing Thai client satisfaction from social media, 2015.