**Google Apps Installation prediction using Regression analysis for the year 2021 Using Pyspark and Tableau**

**Abstract:**

This paper analyses the number of Google Apps installed from the play store for the year 2021.

By using regression analysis in our Google Apps dataset, we will be able to identify how and what are all the independent variables impacting the dependent variable "Installs", as well as relate them to each other. Regression analysis is being performed with Pyspark. With the help of various techniques like Linear Regression, Decision Tree, GBT Regression and Random Forest Regression.

We will use Tableau to perform all the visualizations, and we can explore the dataset by visualizing, for example, the top ten categories with the highest number of installs, the highest paid app, and the most rated categories, among other aspects of the Google Apps data.

Table of Contents

[1. Introduction: 3](#_Toc138435241)

[2. Installation Process – Spark and Pyspark: 4](#_Toc138435242)

[3. Dataset Description: 4](#_Toc138435243)

[4. Data Loading: 5](#_Toc138435244)

[5. Data Engineering: 6](#_Toc138435245)

[5.1. Converting Boolean to Integer Datatype: 7](#_Toc138435246)

[5.2. Checking for Missing Values, and NaN Values: 8](#_Toc138435247)

[5.3. Dropping Duplicate Records: 8](#_Toc138435248)

[5.4. Handling NULL Values: 8](#_Toc138435249)

[6. Feature Engineering: 9](#_Toc138435250)

[6.1. String Indexing and One-Hot Encoding categorial features: 9](#_Toc138435251)

[6.2. Pipeline: 10](#_Toc138435252)

[6.3. Feature Transformation using Vector Assembler: 11](#_Toc138435253)

[6.4. Train and Test Dataset: 11](#_Toc138435254)

[7. Regression Analysis: 12](#_Toc138435255)

[7.1. Linear Regression (Multivariate): 12](#_Toc138435256)

[7.2. Decision Tree Model: 14](#_Toc138435257)

[7.3. Gradient Boost Tree Model (GBT): 15](#_Toc138435258)

[7.4. Random Forest Model: 16](#_Toc138435259)

[8. Discussion: 17](#_Toc138435260)

[9. Conclusion: 19](#_Toc138435261)

[10. Data Visualization Using Tableau: 20](#_Toc138435262)

[REFERENCES 28](#_Toc138435263)

[APPENDIX 30](#_Toc138435264)

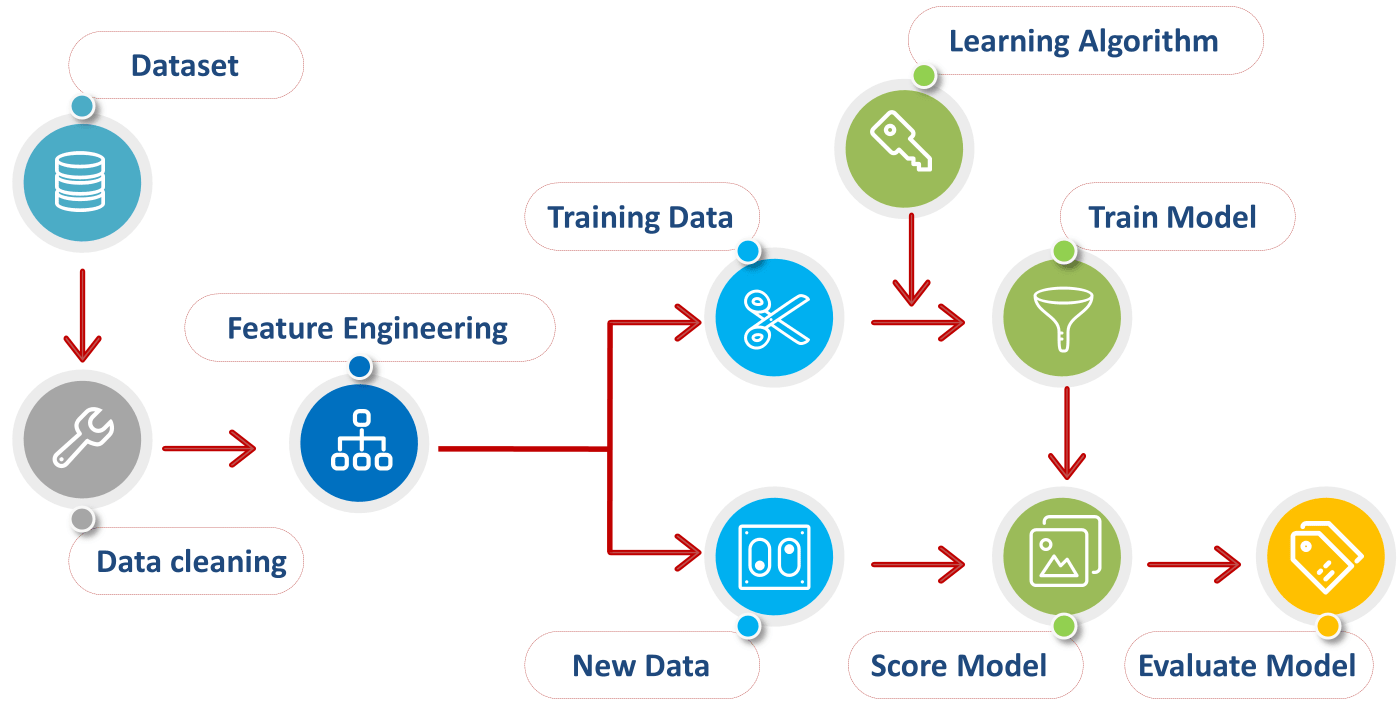
[Appendix A – Spark Installation: 30](#_Toc138435265)

# **Introduction:**

In this Big Data Management and Data Visualization Coursework, we are going to analyse the Google Apps Play store dataset using Pyspark and Tableau.

**Pyspark:**

* PySpark is a Python API to support Python with Apache Spark. It processes large amount of data much quicker, faster than any other conventional framework.
* With Pyspark we will be performing Regression Analysis technique to predict the future happenings between a target variable and one or more independent variables.
* With our dataset, we will be able to find what are all the features that has effect on the Installation Count of the Googles Apps and how these features are dependent and their effect in predicting the number of Installs.
* Figure-1 represents the steps to be followed for performing regression analysis in Pyspark.



**Figure 1 - Steps involved in Regression Analysis**

**Tableau:**

Using Tableau, we will be performing exploratory data analysis for our dataset. We can visualize various factors from our dataset as described below:

* Top 10 categories with Maximum Installations
* Maximum Apps installed based on Content Rating feature
* Maximum and Minimum Apps installed – Monthly report (2021)
* Count of Free and Paid Categories released each month
* Count of Categories supporting Ads
* Most Rated Apps by Categories
* Different Price Range based on Content Rating of the Apps
* Price Range for different categories of Apps

# **Installation Process – Spark and Pyspark:**

First step is to create a virtual machine, like Oracle VMBox and to install Ubuntu OS to work with the Pyspark. To have PySpark installed successfully, we need to check if we have installed **Java** and **Spark** in our Virtual Machine. After successfully installing java and spark in your Virtual Machine, we can use pyspark by opening “Jupyter Notebook” and run the command **“pip install pyspark”**.

# **Dataset Description:**

* This dataset contains Application data for 2.3 million+ applications with the following 24 attributes as described in Table -1.
* It is a huge dataset of size 676.46 MiB and with 2312944 unique values for the apps released from 28 Jan 2010 to 16 Jun 2021.
* Due to time and resource constraint, to process such a large dataset, I have filtered out few rows and taken the data only for the apps that has been released in the year 2021 to perform regression analysis.

Table 1 - Dataset Description

|  |  |  |
| --- | --- | --- |
| Column Names | Data Type | Description |
| App Name | String | Name of the App |
| App Id | String | Package Name |
| Category | String | App Category |
| Rating | Double | Average Rating |
| Rating Count | Integer | Number of Rating |
| Installs | String | Approximate Install Count |
| Minimum Installs | Integer | Approximate minimum app install count |
| Maximum Installs | Integer | Approximate maximum app install count |
| Free | Boolean | Whether app is Free or Paid |
| Price | Double | App Price |
| Currency | String | App Currency |
| Size | String | Size of application package |
| Minimum Android | String | Minimum android version supported |
| Developer Id | String | Developer Id in Google Playstore |
| Developer Website | String | Website of the developer |
| Developer Email | String | Email-id of developer |
| Released | String | App Launch date on Google Playstore |
| Last Updated | String | Last app update date |
| Content Rating | String | Maturity level of app |
| Privacy Policy | String | Privacy policy from developer |
| Ad Supported | Boolean | Ad support in app |
| In App Purchases | Boolean | In-App purchases in app |
| Editor’s Choice | Boolean | Whether rated as Editor’s Choice. |
| Scraped Time | String | Scraped date-time in GMT |

# **Data Loading:**

After successfully installing and importing Pyspark, the entry point into all functionality in Spark is the SparkSession class. It is the entry point to Spark to work with RDD, DataFrame, and Dataset. To create SparkSession in Python, we need to use the builder() method and calling getOrCreate() method.

To create a basic SparkSession, just use SparkSession.builder command and then we must upload the dataset as mentioned in Figure – 5.

 A screenshot of a computer

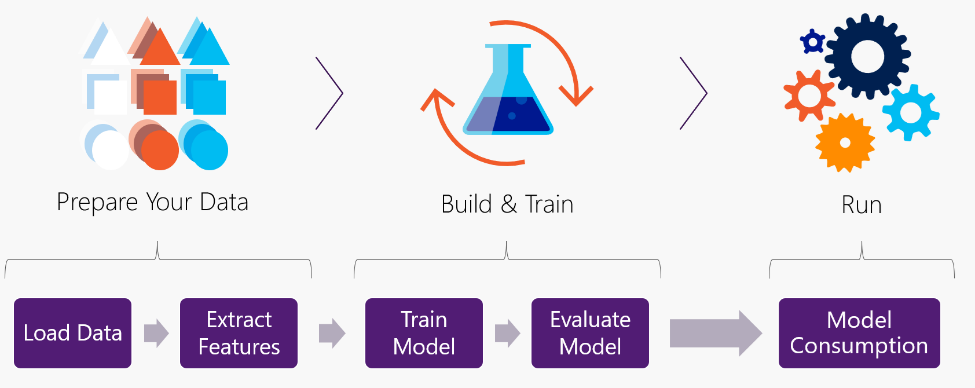
Description automatically generated with medium confidence

**Figure 5 - Initializing Spark Session and Dataset Loadin**

# **Data Engineering:**

Data engineering is often a precursory step to preparing your data for machine learning (ML) tasks. Any kind of processing done on the raw data to prepare it for another data processing procedure is called Data pre-processing. This is the preliminary step for any kind of data analysis procedure.

In Machine Learning, this means that we are cleaning and organizing the raw data to make it suitable for building and training Machine learning models.



**Figure 6 - Steps involved in ML Regression Process**

Figure – 7 represents the schema of the dataset we are using after dropping few unnecessary columns that does not have any impact or associate with our regression analysis.

Graphical user interface, text, application

Description automatically generated

**Figure 7 - Schema/Structure of the Dataset used**

## **5.1. Converting Boolean to Integer Datatype:**

As a next step, we will be converting the columns Free, Ad Supported, In App Purchases, and Editors Choice from Boolean datatype into Integer type as described in the Figure - 8.

Graphical user interface, text

Description automatically generated

**Figure 8 - Converting Boolean Columns to Integer Datatype**

## **5.2. Checking for Missing Values, and NaN Values:**

Noise Leads to Over-Fitting of the Model, so we need to pre-process the Data. These are missing values in the data, or these are data with duplicate/dummy/default/null values. When we don't provide the values for optional fields in the User-interface of any platform then in those such cases either the defaults values or null values will be stored in the backend, hence which becomes the noise is the data during Analysis.

Solution to use-case is the pre-processing techniques:

a) to either remove the null values observation or

b) replace them with mean or median value of these features

In our case, we will be removing the Noise data, as it may cause the model to perform poor due to over-fitting.

In Figure – 9, we are checking for any missing and NaN values in the dataset. From the output, we do not have any nan values and missing values in our dataset.

Table

Description automatically generated

**Figure 9 - Checking for Missing Values**

## **Dropping Duplicate Records:**

In Figure – 10 illustrates the command to drop any Duplicate values and Figure – 11 helps how to perform a Null Check on the dataset.

Graphical user interface, text, application, email

Description automatically generated

**Figure 10 - Dropping Duplicate Records from the dataset**

## **5.4. Handling NULL Values:**

Table

Description automatically generated

**Figure 11 - Performing Null Check in the dataset**

From the output, we see that there are few Null values in the Rating and Rating Count columns. Hence, we are removing them and once again performing the Null check as mentioned in Figure -12, to confirm that all the NULL values are removed successfully.

Table

Description automatically generated

**Figure 12 - After deleting Null Values, performing null check again**

* Before starting our analysis, we must check the data types of the columns using the dtypes command.
* To perform regression analysis, we must separate the categorical columns with the datatype String from the numerical columns and convert them into datatype integer using String Indexing.

# **Feature Engineering:**

## **6.1. String Indexing and One-Hot Encoding categorial features:**

For this, we need to use String Indexing from Pyspark Machine Learning feature to import the String Indexer. It encodes a string column of labels to a column of label indices as illustrated in Figure - 13. Graphical user interface, text, application, email

Description automatically generated

**Figure 13 - Separating Categorical and Numerical Columns and performing String Indexing and One hot Encoding**

## **6.2. Pipeline:**

Pipelines makes streamlining your feature transformations easier. The main concept behind pipelines is to combine complex algorithms and transformations to create a workflow. As our target column is with datatype “string”, we will be using only the following 2 steps as our stages before performing pipelining.

1. Using String Indexer
2. Using One-hot encoding

These 2 steps can be run as a sequence of pipeline stages to form a workflow as described in the below mentioned Figure -14.

Graphical user interface, text, application, email

Description automatically generated

**Figure 14 - String Indexing and One hot encoding as stages in the Pipeline**

As we have successfully transformed the dataset using the pipeline, now we will be dropping the categorical columns as shown in the Figure – 15 and then start with the Vector Assembling process.

Text

Description automatically generated

**Figure 15 - Dropping Categorical Columns after model transformation using Pipeline**

## **6.3. Feature Transformation using Vector Assembler:**

We now have a clean dataset after converting all the categorical columns. The next step is to select the independent variables and assemble them into features using Vector Assembler. Features is an aggregated set of independent columns used to determine the dependent variable "Number of Installs" from our dataset as shown in Figure – 16.

## **6.4. Train and Test Dataset:**

It is the set of data that is used to train and make the model learn the hidden features/patterns in the data. Each time the same training data is fed to the different regression models repeatedly, and the model continues to learn the features of the data.

The training set should have a diversified set of inputs so that the model is trained in all scenarios and can predict any unseen data sample that may appear in the future. The model is trained on the training set, and, simultaneously, the model evaluation is performed on the validation set after each time.

The test set is a separate set of data used to test the model after completing the training. It provides an unbiased final model performance metric in terms of accuracy, precision, etc. The result of the evaluation helps us to understand how well the model performs.

In our scenario, after performing all the required data pre-processing, converting the categorical attributes using string indexing we reproduced the final pipelined data set. Using Vector Assembling, we have combined all the independent variables into a single column named “features” and then transformed the pipelined model (after string indexing) to the Vector Assembler.

Then we finally, split the vector assembled dataset into train and test sets. Train set contains higher percentage of data as compared with the test set, because the bigger the data to train a model the better the results. Hence, we have split 70% of our vector assembled dataset as train dataset and 30% for test data as described in the below Figure – 16.

Graphical user interface, text, application, email

Description automatically generated

**Figure 16 - Features Selection and Transformation using Vector Assembler and Splitting the dataset into Train and Test set**

# **7.** **Regression Analysis:**

Regression analysis is a simple yet powerful technique. It can help us understand how close our calculations when compared to reality.

The aspects that we are considering evaluating the regression model are R-Square Value, Mean Square Error, Root Mean Square Error, and Mean Absolute Error.

In this Google Apps regression analysis, we are trying to predict the number of Apps Installed for the year 2021. The models, that we are using are Linear regression (multivariate), Decision Tree, Random Forest, and Gradient Boost Tree Regression.

## **7.1. Linear Regression (Multivariate):**

In simple linear regression, we will be analysing the correlation/relationship between one independent variable and one target variable. Whereas, in terms of multivariate linear regression, we will be evaluating how strong the relationship between many independent variables and one dependent variable.

Now that we have our train data set ready to implement the regression analysis. Let’s implement the first model multivariate Linear Regression as mentioned in the Figure – 17.

Text

Description automatically generated

**Figure 17 - Linear Regression analysis and Evaluation of Test Dataset**

## **7.2. Decision Tree Model:**

Graphical user interface, text, application

Description automatically generated with medium confidence

**Figure 18 - Decision Tree Model training on train and test dataset**

Graphical user interface, text, application

Description automatically generated

**Figure 19 - Decision Tree Model Evaluation**

**7.3. Gradient Boost Tree Model (GBT):**

Graphical user interface, text, application

Description automatically generated

**Figure 20 - GBT Model training on train and test dataset**

Graphical user interface, text, application, email

Description automatically generated

**Figure 21 - GBT Model Evaluation on test dataset**

## **7.4. Random Forest Model:**

Graphical user interface, text, application

Description automatically generated

**Figure 22 - Random Forest Tree Model training on train and test dataset**

Graphical user interface, application

Description automatically generated with medium confidence

**Figure 23 - Random Forest Model Evaluation on test dataset**

# **8. Discussion:**

Google Apps dataset is a vast dataset with information regarding the apps and their history. Hence, we are considering the Apps that are released in the year 2021 to conduct regression analysis. . Due to space, resource, and time constraints I have reduced the number of rows from the dataset by filtering.

There are many columns like Developer ID, Developer Mail, Developer Website were all not required for our analysis, as they do not have any impact on our target variable.

Also, few features like App Name, App ID are unique values respective to each app and this does not affect the count of installations made. When a particular app was released and when was it last updated, and its privacy policy features, minimum android version needed to download the app and by whom the app was developed are general information.

These variables are not essential to conduct regression analysis because they do not have high level of correlation with the count of apps installed.

From the dataset, we can see that the main features to predict the Installation count are only few columns like Maximum and minimum number of installs, Ratings, Rating Count and very few other attributes which influences the count of number of Installs of a particular App.

In the Linear Regression model, the R-Square value is 0.16 for the training dataset and 0.15 for the test set, which is very low. Hence, we alternatively used other models like Decision Tree, Random Forest and GBT Regression to further fine tune the results.

From the results, it is evident that there no huge differences in the R-Square value and the difference between the MAE, MSE and RMSE scores are also only few times higher than the GBT model.

Also, from the Table we can see that the GBT Regression model seems to have higher R-Square value when compared to the other models.

In addition to that, other evaluator metric’s like MSE, RMSE and MAE scores in the GBT model are lower than the Decision Tree and Random Forest model. We can conclude that the best performing model for our dataset to predict the Installation count of the Apps is Gradient Boost Tree Regression Model.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Model Name | R squared | MSE | RMSE | MAE |
| Decision Tree | 0.95 | 0.13 | 0.37 | 0.11 |
| Random Forest | 0.95 | 0.27 | 0.51 | 0.34 |
| GBT | 0.96 | 0.10 | 0.32 | 0.08 |

**Table 2 - Model Comparison using different metrics**

Chart, scatter chart

Description automatically generated

**Figure 24 - Regression Analysis using Tableau**

# **9. Conclusion:**

As a part of the big data management and data visualization project, I have performed regression analysis for the Google Apps dataset for the year 2021.

* For the regression analysis we are using Pyspark and Tableau for all the visualization. After successfully installing all the necessary software, we must load the dataset.
* Then by performing nan check, missing value and NULL value check we have dropped all the unnecessary data which might be a hinderance to in performing our analysis.
* To perform regression analysis, the main task is to check the data types of the columns. And convert all the categorical columns using String Indexing process.
* The next step after String Indexing is to One hot encode the string indexed columns.
* To do these steps for all the categorical columns in the dataset, we will be using the pipeline architecture.
* We can set each of these steps as individual stages before passing them into the pipeline.
* Then, we will be passing these stages into the pipeline to transform all the categorical values in the complete dataset in an orderly manner and provides the pipelined model as the output.
* Now that we have the final data ready with all the columns converted, we can use the Vector Assembler from the Machine learning library to aggregate and transform all the independent variables as one single vector in the feature column.
* After executing the vector assembler transformation, we will have the final dataset ready to split 70% of the data as training set and the remaining 30% data as the test set to evaluate.
* Finally, we can start importing all the necessary packages to perform regression and evaluation and start implementing various regression models and compare their results.
* After analysing the values of various metrics like R – Square value, MSE, MAE and RMSE from different models, we can conclude that Gradient Boost Tree Regression model is the best performing model for predicting the Installs attribute from our dataset.

# **10. Data Visualization Using Tableau:**

In this section, we will be performing exploratory analysis with our dataset, to have a detailed investigation of each attribute.

* Top 10 categories with Maximum Installations
* Maximum Apps installed based on Content Rating feature.
* Maximum and Minimum Apps installed – Monthly report (2021)
* Count of Free and Paid Categories released each month.
* Count of Categories supporting Ads
* Most Rated Apps by Categories
* Different Price Range based on Content Rating of the Apps
* Price Range for different categories of Apps

In the first horizontal bar chat, we have filtered the Top 10 categories of the Apps maximum Installed from the play store for the year 2021.

Chart

Description automatically generated with low confidence

**Figure 25 - Top 10 Categories of Google Apps with Maximum Installs**

From figure - 25, we can see that “Tools” is the category with Maximum number of Installs more than 800 million.

The second top category is “Casual” with more than 200 million apps installed.

Categories “Simulation” and “Puzzle” are the third and fourth with a light difference in the maximum installed around 100 million.

Diagram

Description automatically generated

**Figure 26 - Maximum Apps installed based on Content Rating feature**

Figure -26, is a Donut Chart created using tableau, to represent the maximum number of Apps installed based on the Content rating category.

It is clear that 79.55% is the maximum installed apps with “Everyone” as their content rating.

Only 0.02% of the apps were isntalled for the Adult only 18+ content.

The second content rating category is the “Teen” category with 12.18% of installations.

Chart, bar chart

Description automatically generated

**Figure 27 - Maximum and Minimum Apps installed – Monthly report (2021)**

Figure – 27 is a grouped bar chart presenting the Monthly report of the Maximum and minimum number of apps installed for the months from January to June 2021.

From the bar chart, we can see that beginning and mid of the firsts 6 months the apps installation rate is higher.

Then the rates gradually decrease from April to June (2021).

As a result, we can confirm that January is the highest month for both maximum and minimum number of apps installed.

Chart

Description automatically generated

**Figure 28 - Count of Categories supporting Ads**

Using Area graph in the figure – 28, we have visualized the count of categories released each month.

The colour represents whether the application is free or paid and the data is filtered for the top 10 categories.

Visualization makes it easy to understand that most of the apps released and installed are free.

Chart, bubble chart

Description automatically generated

**Figure 29 - Packed Bubble Chart**

In this packed bubbled chart, the colours represent whether which what are all the different categories that supports advertisement.

Green colour are the categories that supports advertisements and blue colour are the categories that do not support ads.

The size is based on the count of category.

Timeline

Description automatically generated

**Figure 30 - Horizontal Bar Chart - Most Rated Apps by Categories**

Figure – 30, is a horizontal bar chart that represents the count of ratings provided for the top 10 categories.

From the graph, we can see that “Entertainment” is the most rated category with almost near to 5000 ratings.

“Tools” is the second most rated category with 3306 ratings.

“Education” and “Personalization” are the third and fourth most rated apps.

Chart

Description automatically generated

**Figure 31 - Area graph of Price Range by Content Rating**

The figure -31 area graph represents the percentage of total price for each content rating categories. There are totally 5 content rating categories as follows:

1. Adults only 18+ - 0.00 %
2. Everyone - 62.30%
3. Everyone 10+ - 11.17%
4. Mature 17+ - 0.00%
5. Teen - 26.53%

We can see that the content rating category “Everyone” has the highest price range of 62.3% and the apps that are for Mature 17+ and Adults only 18+ are with zero percentage.

Chart, treemap chart

Description automatically generated

**Figure 32 - Tree map - Price Range for different categories**

Figure – 32 is a tree map visualization and the size of each blocks represents the sum of price of that category.

“Role Playing” and “Personalization” category apps are with the very high price range.

Categories “Tools” and “Puzzle” are almost with a similar price range.

When we visualize the categories according to their price range, we can quickly estimate the high and the low-price range categories.

# **REFERENCES**

MLlib: Main Guide - Spark 3.3.0 Documentation. <https://medium.com/fintechexplained/part-3-regression-analysis-bcfe15a12866>

Please wait... | Cloudflare. <https://netmiko.com/install-apache-spark-on-ubuntu/>

Drop rows in pyspark with condition - Data Science Made Simple. DataScience Made Simple

<https://www.datasciencemadesimple.com/drop-rows-in-pyspark-drop-rows-with-condition/>

Role of StringIndexer and Pipelines in PySpark ML Feature. Medium. <https://medium.com/@nutanbhogendrasharma/role-of-stringindexer-and-pipelines-in-pyspark-ml-feature-b79085bb8a6c>

Linear Regression and Decision Tree Implementation using Pyspark. Medium. <https://medium.com/analytics-vidhya/linear-regression-and-decision-tree-implementation-using-pyspark-bfcd93dee86>

Feature Selection Techniques in Machine Learning with Python. Medium. <https://towardsdatascience.com/feature-selection-techniques-in-machine-learning-with-python-f24e7da3f36e>

Complete Machine Learning Project with PySpark MLlib Tutorial ❌Logistic Regression with Spark MLlib. YouTube. (2020). <https://www.youtube.com/watch?v=1a7bB1ZcZ3k>

GeeksforGeeks. (2021). <https://www.geeksforgeeks.org/how-to-change-column-type-in-pyspark-dataframe/>

What are the Various Noise Types exists in Dataset And How to Handle Them ?? | Data Science and Machine Learning. Kaggle.com. <https://www.kaggle.com/general/141069>

12. Clustering — Learning Apache Spark with Python documentation. <https://runawayhorse001.github.io/LearningApacheSpark/clustering.html>

9. Regression — Learning Apache Spark with Python documentation. <https://runawayhorse001.github.io/LearningApacheSpark/regression.html#id9>

PySpark Tutorial 17: PySpark Correlation Analysis | PySpark with Python. YouTube. (2021). <https://www.youtube.com/watch?v=R4mkiDj2oys>

Z. Statology. What is Considered a Good RMSE Value?. (2021). <https://www.statology.org/what-is-a-good-rmse/>

data cleansing using machine learning. Faepa.br. <https://www.faepa.br/kgyo.aspx?cname=data+cleansing+using+machine+learning&cid=66>

K Means Clustering using PySpark on Big Data. Medium. <https://towardsdatascience.com/k-means-clustering-using-pyspark-on-big-data-6214beacdc8b>

PySpark Tutorial 33: PySpark Logistic Regression | PySpark with Python. YouTube. (2021). <https://www.youtube.com/watch?v=YpI4_RrargQ>

# **APPENDIX**

## **Appendix A – Spark Installation:**

In your Linux OS, open the terminal and perform the following steps in the same order:

1. The first step will be to create a new directory in the home directory where we will download and Install Apache Spark.

mkdir -p spark

1. Now, to be able to run and use Apache Spark we will need Java JDK installed in our Ubuntu system. To check whether you have java installed on your Ubuntu you can run the following command in the terminal.

java – version

1. If it says “Command “java” not found” this means that java is not installed. Using the next command, we can install java.

sudo apt-get update

sudo apt-get install default-jdk -y

1. Then we can verify the java version once again, to verify if java was successfully installed.
2. As a next step, you can download Apache Spark into the spark directory. After that from the terminal, we must perform the following steps to successfully install spark in Ubuntu.
3. The first step will be to untar the tar file that we downloaded. To do that, first, change your working directory to the “spark” directory.

cd spark

1. The following command is used to untar the tar file. **“tar -xvf filename”,** the file name will be the name of the downloaded tar file that you want to untar.

tar -xvf spark-3.2.0-bin-hadoop2.7.tgz

1. After untar the file, we must set up the environment variable path for Apache Spark. For that we must change the working directory to home directory using the following command.

cd ~

1. Now, after changing the directory, using the gedit editor edit the Bashrc file.

gedit ~/.bashrc

1. This command will open the bashrc file in the gedit text editor, scroll down to the bottom of the bashrc file, and in the end add these lines –

SPARK\_HOME=/home/username/spark/spark-3.2.0-bin-hadoop2.7

export PATH=$PATH:$SPARK\_HOME/bin:$SPARK\_HOME/sbin

1. Change “username” to your own username, the file name at the end will be the name of the tar file that you untar-ed. After that on the top right corner click on the save button and exit the editor.
2. Finish the installation by using the source command –

source ~/.bashrc

1. With these steps, the installation of Apache spark is now completed, close the current terminal window, and open a new one. To verify the installation of Apache Spark, type the below command in the terminal:

spark-shell

This command will help us to open the Spark shell meaning that the installation of Spark was successful. Once the Spark installation is successful, we can install PySpark in the Jupyter Notebook, using the below mentioned command:

**pip install pyspark**