(7082CEM)

Coursework
Demonstration of a Big Data Program

MODULE LEADER: Dr. Marwan Fuad

Student Name: Jyothi Yendamuri

SID: 11467683

Electronic Products and Pricing Data Using Pyspark

I can confirm that all work submitted is my own: Yes

7082CEM CW 2021

Table of Contents

List of Figures

1. Introduction

In the rapid advancement of technology, it can be seen that big data has become a focal point for most of the industries and due to availability of large amount of data brings lot of potential in the big data and analytics domain. Big data is defined as the process of examining large amount of data to uncover possible patterns, correlation, market trend and customer preferences that will help the organization to make informed decisions. Due to the large amount of data, managing it effectively is one of the serious issue. Big data should be managed in such a way that data quality, accessibility and security of data can be preserved to use it for analytical purposes(Sagiroglu, 2013). There are lot of tools available in the market to process the large amount of data but the one that is most preferred among the individuals is use of Spark. Spark is a distributed analytical engine, which is used to process large amount of data. Spark provides lot of functionality in managing big data such as faster processing of data, ease of use, fault tolerance, low latency due to high use of internal memory and it offer support for multiple programming language like python, Scala etc. The big reason for choosing spark tool than any other tool is that it internally used Resilient Distributed Dataset (RDD), which is a primary data structure of Spark, which can be used to split the nodes and quickly perform the operations and tasks(Zaharia M. X., 2016).

In this project, we are using the electronic products and pricing data and data has been taken from kaggle, one of the largest data repository. In this project, we are taking only a small extract of data due to computation power and time limitation. The dataset contain information of about 15000 electronic products with pricing information on 10 different attributes, which are amountMax, amountMin, availability, condition, currency, isSale, dataSeen, merchant and shipping. The dataset also contain information on brand, category, merchant name, source, and other information(Datafiniti, 2018). This dataset can be used to identify retail industry trends in pricing strategies. Once the dataset is, selected appropriate tools and frameworks need to be pick up for data analysis and visualization to make decisions that are more informed. So for this project PySpark is selected for doing data analysis and visualization. After the tool is selected we need to perform various tasks and operation on it such as data cleaning, data preprocessing like removing null values, converting categorical variables to numerical representation using one hot encoding, proper feature selection, dropping insignificant features, normalizing the data, splitting the data and then finally perform modelling. More information about the data analysis is discussed in detail in the discussion section.

2. Implementation

In this section we are going to discuss about the tools that is going to be used for this project i.e. Spark, its various components like Resilient Distributed Dataset (RDD), Data Frames in detail and also discuss on the installation of these tools and then finally implement the dataset using this tool.

2.1 Spark

Spark is an open source distributed computing framework act as a common platform for big data processing. The main purpose of using spark is to overcome the drawbacks of Map Reduce because of its fast processing and highly efficient memory management. It is an open source project from Apache. Due to growing community of spark developers, it has come into lime light. Almost all the big data companies start using it for their data processing and analysis. The growth in demand is due to many factors such as fast processing speed, support multiple languages, support for multiple libraries, real-time processing and compatibility. From the low level, side spark follows a master slave architecture where master (Driver) is a key node and it control the overall execution of a program where actually the execution happens in the slave (executor) node. For any spark program, there can be only one master node but executor node can be configured with any number based on the requirement. Therefore, master node is a single point of failure if it gets failed then whole program stops working. Apart from this there is also a cluster Manager, which manages the master and driver nodes regarding how much memory, is required in each executor node, what is wait and load time for each node, makes multiple copies of memory so if one fails to work then other can be used(Zaharia M. C., 2010).

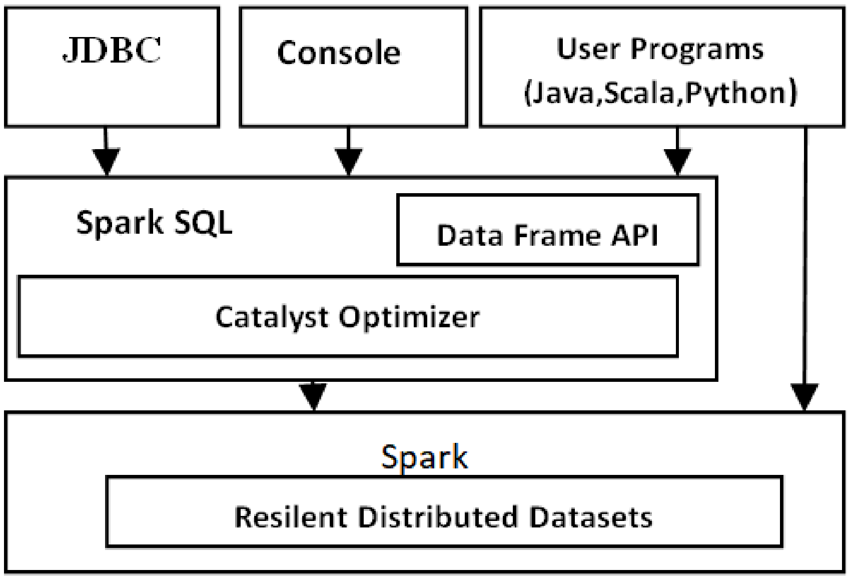


Figure 1. Interfaces in Spark(Armbrust, 2015)

2.2 Installation

We need to download some software's to start working on the project. So below are the steps discussed to download all the relevant software and tools. In this project, we are using Pyspark for big data analysis and visualization. Therefore, to download it we need to ensure that we have two additional software one is python (version greater than 2.6) and other is java (version greater than 7). In addition, we are installing all these software's in the window operating system. So all the commands discussed below are related to window operating system.

1. First, we will check whether java is already installed in our operating system or not. To check this open your command prompt (Win+R) and type the following command.

```
java --version
```

If the java is already installed it is going to display the following information.

```
java version "1.8.0_92"
```

If java is not installed then it will display the following error message.

'java' is not perceived as an internal or external command, operable program or batch file'.

Then in that case, we need to download java. To download java go to Oracle website and download the latest version and complete the installation process and configure the environment variables(L, 2019).

2. Next, we will check whether python is installed in our operating system or not. To check this open your command prompt (Win+R) and type the following command.

   python --version

   If the python is already installed it is going to display the following information.

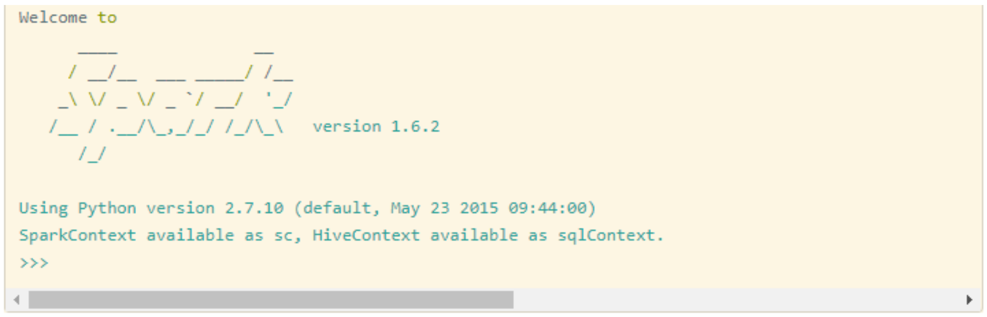
   Python 3.6.5:: Anaconda, Inc.

   If python is not installed then it will display the following error message.

   'python' is not perceived as an internal or external command, operable program or batch file'.

   Then in that case, we need to download python. To download python go to python official website and download the latest version and complete the installation process and configure the environment variables(L, 2019).

3. Now the pre-requisite software has been installed successfully to download the Apache spark. To download Apache spark go to spark official website and choose package type i.e. pre built for latest version of Hadoop 2.7 and later and then download it. After that create one folder with name Spark and unzip all the files that you downloaded to it and refer this folder as SPARK_HOME. After completing this steps to check if spark is successfully installed in the system or not open Anaconda prompt and type **bin\pyspark.** If the installation is successful, it will display the following information(L, 2019).

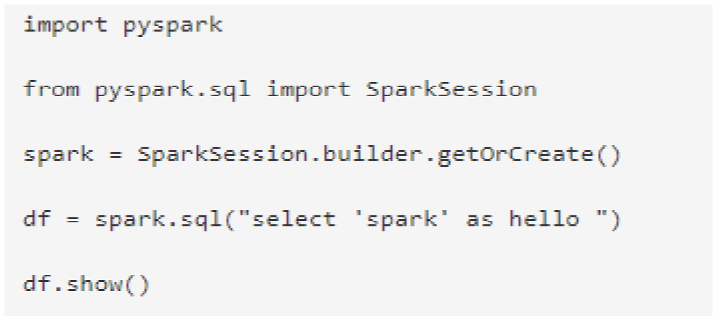


```
Welcome to
      ____              __
     / __/__  ___ _____/ /__
    _\ \/ _ \/ _ `/ __/  '_/
   /__ / .__/\_,_/_/ /_/\_\   version 1.6.2
      /_/

Using Python version 2.7.10 (default, May 23 2015 09:44:00)
SparkContext available as sc, HiveContext available as sqlContext.
>>>
```

Figure 2. Installing Spark

4. After that download the **winutils.exe** and configure the spark installation to find **winutils.exe.** This executable file need to be present inside the SPARK_HOME folder with another folder created inside it with the name Hadoop\bin (L, 2019).
5. After completing the above steps, we need to configure the environment variables SPARK_HOME and HADOOP_HOME in the system variables path with the respective folder location (L, 2019).

6. Now the time is to use spark in Jupyter Notebook. For that open Anaconda prompt and type the following command **"python -m pip install findspark".** This package is required to run spark in Jupyter notebook. Finally open Jupyter notebook and create a python file inside it with the following command.



```
import pyspark

from pyspark.sql import SparkSession

spark = SparkSession.builder.getOrCreate()

df = spark.sql("select 'spark' as hello ")

df.show()
```

Figure 3. Using Spark in Python Jupyter

If the above code executes successfully then pyspark is ready to start working on the project (L, 2019).

2.3 Pyspark

Pyspark is one of the programming language, which is supported by spark. It is almost similar to python. With the release of it enabled users to collaborate between python and spark. It is also called as python API for spark. Pyspark provides almost all the functionality like performing data analysis, creating machine-learning models and performing ETL on data. It provides support for rich set of libraries for performing different tasks such as GraphX, SparkSQL, Spark Streaming and Mlib Machine learning. It is closely integrated with python so we can also use vast variety of python inbuilt libraries. It is considered as one of the best language for creating analytical solution at scale. In addition, it has many embedded data visualization functions, which will help to detect patterns or trends from the dataset. Moreover, pyspark does not load the entire data at once, which helps in faster processing and managing the memory in an efficient manner(Meng, 2016).

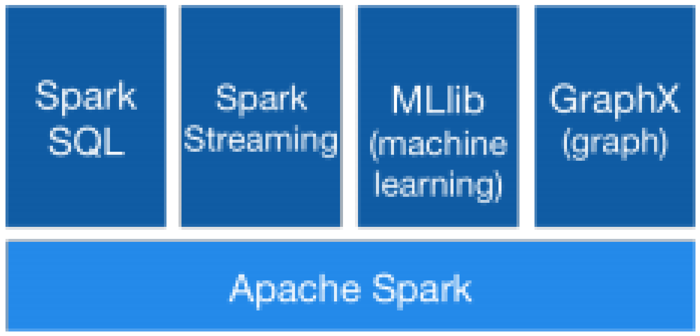


Figure 4. Apache Spark Ecosystem(Meng, 2016)

2.4 Dataset

In the project the dataset, which is chosen, belongs to the technological covering everything from home appliances to handheld devices. The dataset contain information about electronic products and pricing data. Due to time limitation and computation power, the data is extracted in small amount and is taken from kaggle, one of the largest data repository. The dataset contain information of about 15000 electronic products with pricing information on 10 different attributes, which are amountMax, amountMin, availability, condition, currency, isSale, dataSeen, merchant and shipping. The dataset also contain information on brand, category, merchant name, source, and other information. All this attributes can be used to identify retail industry trends in pricing strategies(Datafiniti, 2018). The main aim of this project is to analyze the dataset and find patterns, relationship among features to identify trends in pricing strategies. More information about the data attributes will be discussed in the implementation part.

The table below shows the chosen attributes from the dataset including it's description and datatypes

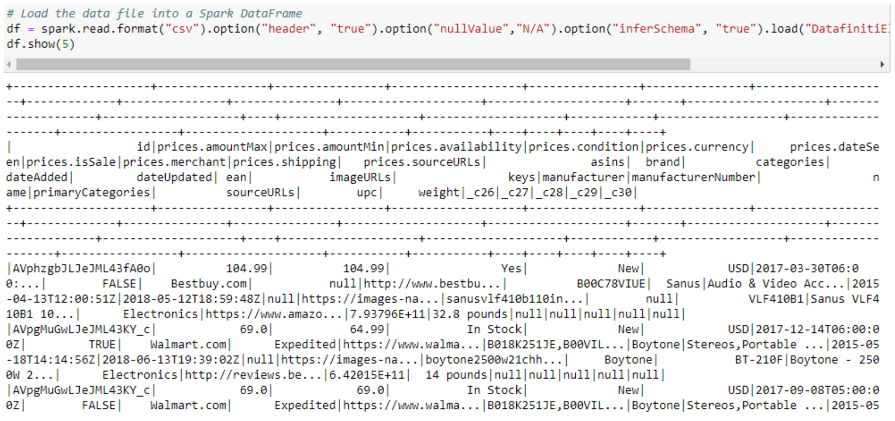
Attributes	**Description**	**Data Types**
Id	Universal number that gives a unique identity to product	Integer
Categories	States the Electronic Products types	String
Prices.amountMin	Minimum prices of products	Double
Prices.availability	All available prices of products	String
Prices.currency	Prices are shown in different currency	String
Prices.dateSeen	Date of the prices	Date
Prices.isSale	Checks product availibility in offer or not	Boolean
Prices.shipping	shipping address of the each product	String
Prices.sourceURLs	Perticular selected product prices urls	String
Asins	Unique identifier of 10 numbers for the product	Integer
Brand	It refers to the individual products of a company	String
Prices.amountMax	Shows the highest price of the products	Integer
DateAdded	Each product with particualar date	Integer
DateUpdated	Check whether products are updated or not	Date
EAN	Standard number to identify a specific retail product type	Integer
Keys	Each product has a particular keyvalue	String
ManufacturerNumber	Indicates the manufaturing date of the product	Integer
Name	Name of the specific product	String
PrimaryCategories	Products shows in primary categories list	String
SourceURLs	Product website urls	String
Upc	Universal product code assigned to products.	integer
Weight	checks each product weight	float

2.5 Implementation

Now we get an idea about the dataset, the next big thing is to start implementing it so that it can be used for any analytical purpose. In the dataset there are lot of missing values, duplicates and null values present in some of the columns so we need to first clean it up and then process it so below are all the details defined.

Loading dataset

The first thing that needs to be performed before starting any cleaning or processing the dataset is to load the dataset into the Jupyter notebook. Below command is loading the csv data locally using the read() function into a variable df.



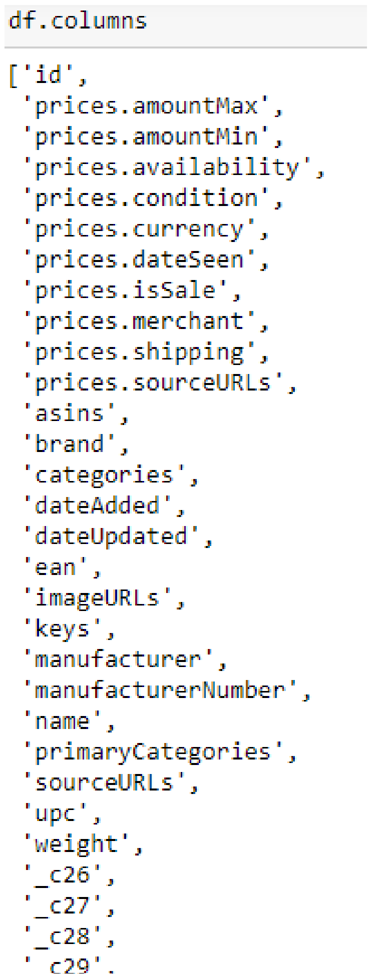
```
# Load the data file into a Spark DataFrame
df = spark.read.format("csv").option("header", "true").option("nullValue","N/A").option("inferSchema", "true").load("DatafinitiE
df.show(5)
```

```
+--------------------+----------------+----------------+--------------------+----------------+---------------+------------------
--+--------------+----------------+----------------+--------------------+--------------------+-------+---------------------+-------
-------------+---------------------+----+---------------------+--------------------+-------------+--------------------+------------
-------+----------------+--------------------+------------+------------+----+----+----+----+----+
|                  id|prices.amountMax|prices.amountMin|prices.availability|prices.condition|prices.currency|     prices.dateSe
en|prices.isSale|prices.merchant|prices.shipping|   prices.sourceURLs|               asins|  brand|           categories|
dateAdded|          dateUpdated| ean|           imageURLs|                keys|manufacturer|manufacturerNumber|                  n
ame|primaryCategories|           sourceURLs|         upc|      weight|_c26|_c27|_c28|_c29|_c30|
+--------------------+----------------+----------------+--------------------+----------------+---------------+------------------
--+--------------+----------------+----------------+--------------------+--------------------+-------+---------------------+-------
-------------+---------------------+----+---------------------+--------------------+-------------+--------------------+------------
-------+----------------+--------------------+------------+------------+----+----+----+----+----+
|AVphzgbJLJeJML43fA0o|          104.99|          104.99|                Yes|             New|            USD|2017-03-30T06:0
0:...|         FALSE|    Bestbuy.com|            null|http://www.bestbu...|          B00C78VIUE|  Sanus|Audio & Video Acc...|2015
-04-13T12:00:51Z|2018-05-12T18:59:48Z|null|https://images-na...|sanusvlf410b110in...|         null|          VLF410B1|Sanus VLF4
10B1 10...|       Electronics|https://www.amazo...|7.93796E+11|32.8 pounds|null|null|null|null|null|
|AVpgMuGwLJeJML43KY_c|            69.0|           64.99|           In Stock|             New|            USD|2017-12-14T06:00:0
0Z|        TRUE|    Walmart.com|       Expedited|https://www.walma...|B018K251JE,B00VIL...|Boytone|Stereos,Portable ...|2015-05
-18T14:14:56Z|2018-06-13T19:39:02Z|null|https://images-na...|boytone2500w21chh...|      Boytone|          BT-210F|Boytone - 250
0W 2...|       Electronics|http://reviews.be...|6.42015E+11|  14 pounds|null|null|null|null|null|
|AVpgMuGwLJeJML43KY_c|            69.0|            69.0|           In Stock|             New|            USD|2017-09-08T05:00:0
0Z|       FALSE|    Walmart.com|       Expedited|https://www.walma...|B018K251JE,B00VIL...|Boytone|Stereos,Portable ...|2015-05
```

Figure 5. Loading dataset

Printing column of dataset

The below command displays all the columns present in the dataset.



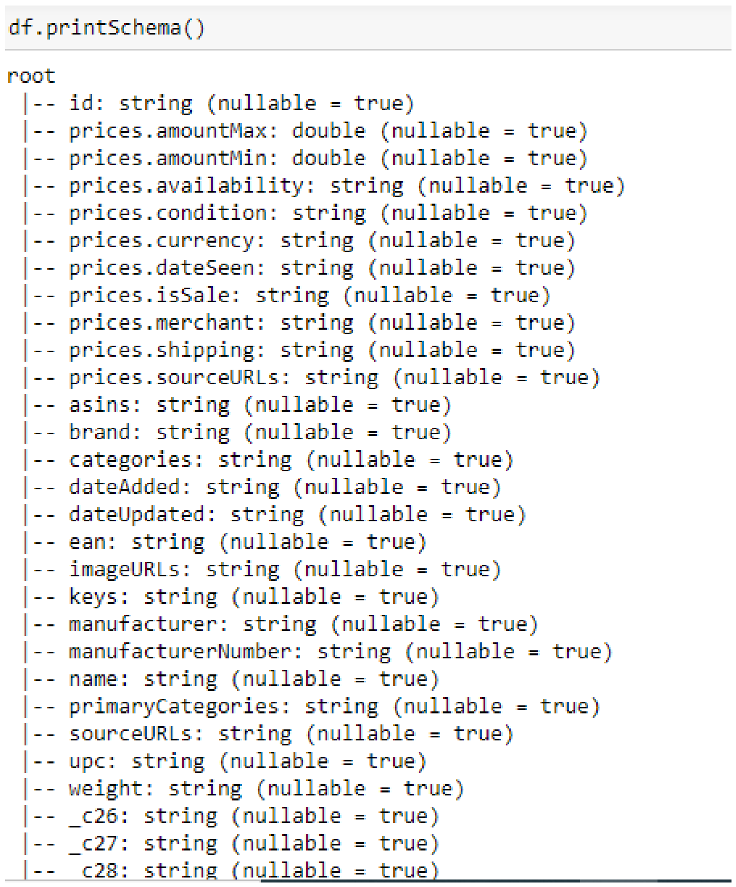
```
df.columns
```

```
['id',
 'prices.amountMax',
 'prices.amountMin',
 'prices.availability',
 'prices.condition',
 'prices.currency',
 'prices.dateSeen',
 'prices.isSale',
 'prices.merchant',
 'prices.shipping',
 'prices.sourceURLs',
 'asins',
 'brand',
 'categories',
 'dateAdded',
 'dateUpdated',
 'ean',
 'imageURLs',
 'keys',
 'manufacturer',
 'manufacturerNumber',
 'name',
 'primaryCategories',
 'sourceURLs',
 'upc',
 'weight',
 '_c26',
 '_c27',
 '_c28',
 '_c29',
```

Figure 6. Printing columns of dataset

Printing Schema of a dataset

Once the dataset is loaded, it is good to use a PrintSchema() function it is going to scan through each of the columns and display the column name along with their corresponding data type.

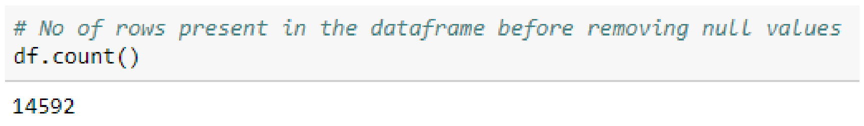


```
df.printSchema()

root
 |-- id: string (nullable = true)
 |-- prices.amountMax: double (nullable = true)
 |-- prices.amountMin: double (nullable = true)
 |-- prices.availability: string (nullable = true)
 |-- prices.condition: string (nullable = true)
 |-- prices.currency: string (nullable = true)
 |-- prices.dateSeen: string (nullable = true)
 |-- prices.isSale: string (nullable = true)
 |-- prices.merchant: string (nullable = true)
 |-- prices.shipping: string (nullable = true)
 |-- prices.sourceURLs: string (nullable = true)
 |-- asins: string (nullable = true)
 |-- brand: string (nullable = true)
 |-- categories: string (nullable = true)
 |-- dateAdded: string (nullable = true)
 |-- dateUpdated: string (nullable = true)
 |-- ean: string (nullable = true)
 |-- imageURLs: string (nullable = true)
 |-- keys: string (nullable = true)
 |-- manufacturer: string (nullable = true)
 |-- manufacturerNumber: string (nullable = true)
 |-- name: string (nullable = true)
 |-- primaryCategories: string (nullable = true)
 |-- sourceURLs: string (nullable = true)
 |-- upc: string (nullable = true)
 |-- weight: string (nullable = true)
 |-- _c26: string (nullable = true)
 |-- _c27: string (nullable = true)
 |-- _c28: string (nullable = true)
```

Figure 7. Printing Schema of dataset

Count number of records

The below count() function is used to print the number of records present in the dataset.

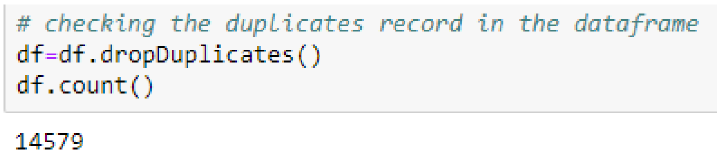


```
# No of rows present in the dataframe before removing null values
df.count()

14592
```

Figure 8. Counting number of records

Dropping the duplicates record

The below command dropDuplicates() function is used to drop the duplicates record present in the dataset. Here, it can be seen that there are 13 duplicates record present in the dataset. Removing them is one of the essential part of data cleaning process.

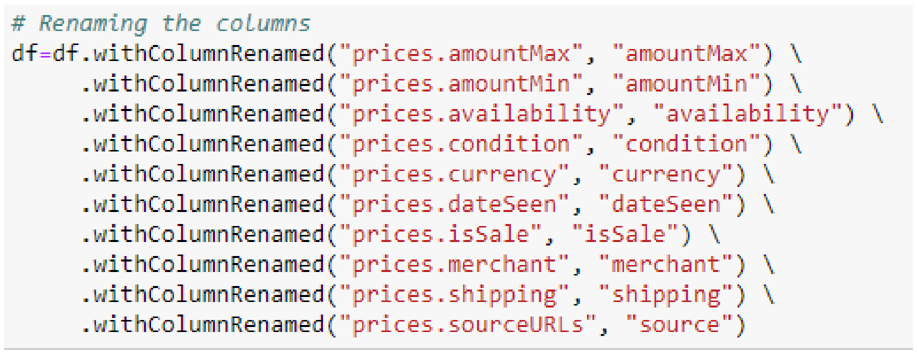


```
# checking the duplicates record in the dataframe
df=df.dropDuplicates()
df.count()

14579
```

Figure 9. Dropping duplicates records

Renaming Columns

We have seen that some of the columns has a '.' present. This can be an issue when working on some of the columns so we need to rename those columns which contains '.'To rename any column use withColumnRenamed() function.

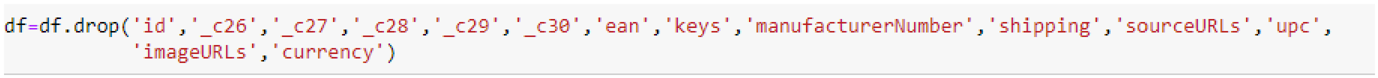


```
# Renaming the columns
df=df.withColumnRenamed("prices.amountMax", "amountMax") \
     .withColumnRenamed("prices.amountMin", "amountMin") \
     .withColumnRenamed("prices.availability", "availability") \
     .withColumnRenamed("prices.condition", "condition") \
     .withColumnRenamed("prices.currency", "currency") \
     .withColumnRenamed("prices.dateSeen", "dateSeen") \
     .withColumnRenamed("prices.isSale", "isSale") \
     .withColumnRenamed("prices.merchant", "merchant") \
     .withColumnRenamed("prices.shipping", "shipping") \
     .withColumnRenamed("prices.sourceURLs", "source")
```

Figure 10. Renaming columns

Dropping Columns

After exploring the dataset it is found that some of the columns are redundant or unnecessary columns. So use drop() function and passed the column name and it will remove that particular column.

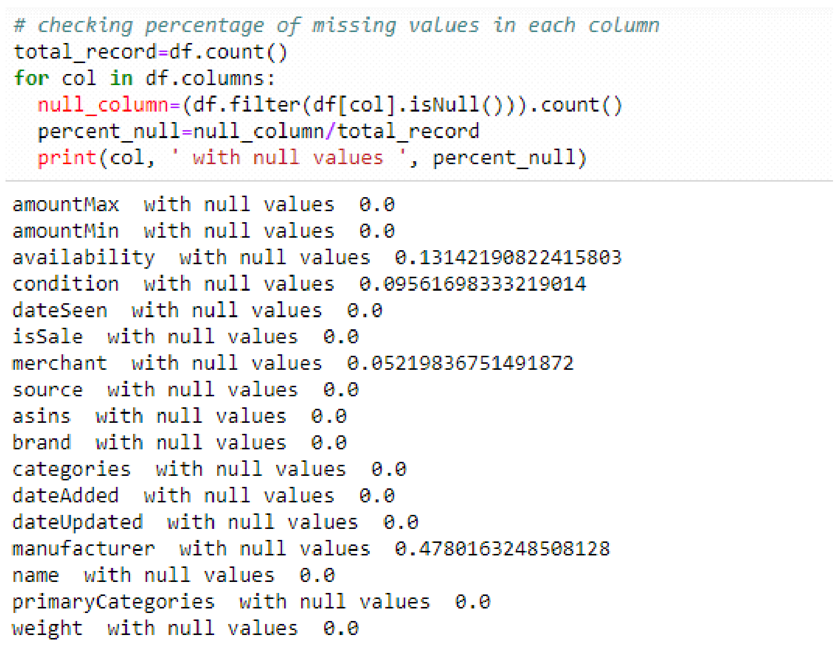


```
df=df.drop('id','_c26','_c27','_c28','_c29','_c30','ean','keys','manufacturerNumber','shipping','sourceURLs','upc',
           'imageURLs','currency')
```

Figure 11. Dropping columns

Handling missing values in the dataset

In dataset, there are some columns, which contain null values. Therefore, it is essential to process these null values. To process the null values we need to perform imputation. There are many ways through which we can impute the null values i.e mean, median, mode, filling by some value and soon.

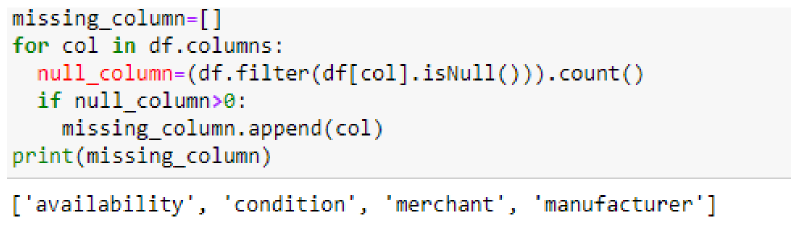


```
# checking percentage of missing values in each column
total_record=df.count()
for col in df.columns:
  null_column=(df.filter(df[col].isNull())).count()
  percent_null=null_column/total_record
  print(col, ' with null values ', percent_null)
```

```
amountMax  with null values  0.0
amountMin  with null values  0.0
availability  with null values  0.13142190822415803
condition  with null values  0.09561698333219014
dateSeen  with null values  0.0
isSale  with null values  0.0
merchant  with null values  0.05219836751491872
source  with null values  0.0
asins  with null values  0.0
brand  with null values  0.0
categories  with null values  0.0
dateAdded  with null values  0.0
dateUpdated  with null values  0.0
manufacturer  with null values  0.4780163248508128
name  with null values  0.0
primaryCategories  with null values  0.0
weight  with null values  0.0
```

Figure 12. Checking percentage of missing values

Now we are printing those columns which have null values greater than zero using the below snippet of code.

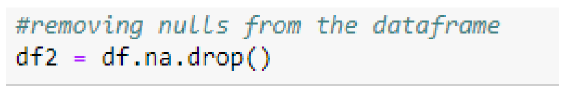


```
missing_column=[]
for col in df.columns:
  null_column=(df.filter(df[col].isNull())).count()
  if null_column>0:
    missing_column.append(col)
print(missing_column)
```

```
['availability', 'condition', 'merchant', 'manufacturer']
```

Figure 13. Fetching columns with null values

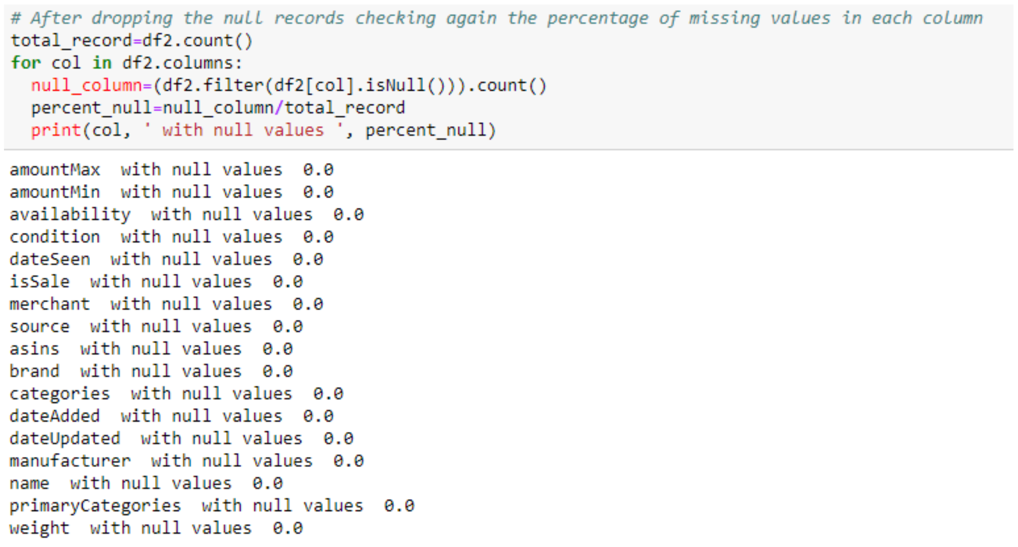
Here, instead of performing imputation we are going to remove all the null values from the dataset using the below snippet of code.



```
#removing nulls from the dataframe
df2 = df.na.drop()
```

Figure 14. Dropping null values

Now there are no null values present in any of the column.



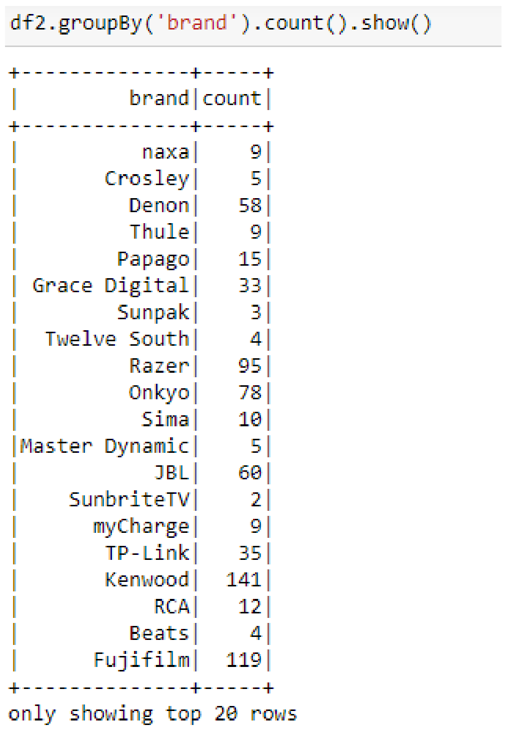
```
# After dropping the null records checking again the percentage of missing values in each column
total_record=df2.count()
for col in df2.columns:
  null_column=(df2.filter(df2[col].isNull())).count()
  percent_null=null_column/total_record
  print(col, ' with null values ', percent_null)
```

```
amountMax  with null values  0.0
amountMin  with null values  0.0
availability  with null values  0.0
condition  with null values  0.0
dateSeen  with null values  0.0
isSale  with null values  0.0
merchant  with null values  0.0
source  with null values  0.0
asins  with null values  0.0
brand  with null values  0.0
categories  with null values  0.0
dateAdded  with null values  0.0
dateUpdated  with null values  0.0
manufacturer  with null values  0.0
name  with null values  0.0
primaryCategories  with null values  0.0
weight  with null values  0.0
```

Figure 15.Verifying columns post dropping null values

GroupBY

GroupBy operation is used to find the count of each category present in the column. For example in the 'brand' column, you can see all the different values along with their count.



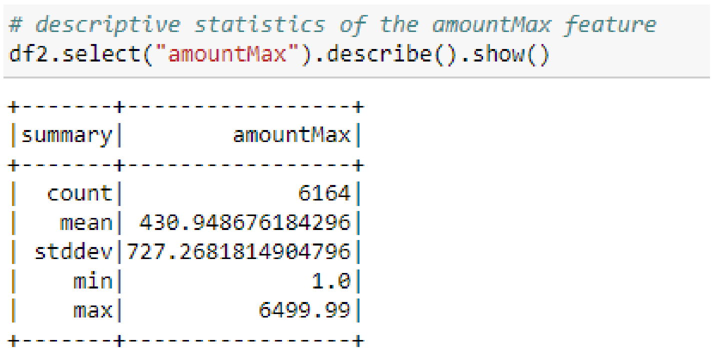
```
df2.groupBy('brand').count().show()
```

```
+--------------+-----+
|         brand|count|
+--------------+-----+
|          naxa|    9|
|       Crosley|    5|
|         Denon|   58|
|         Thule|    9|
|        Papago|   15|
| Grace Digital|   33|
|        Sunpak|    3|
|   Twelve South|    4|
|         Razer|   95|
|         Onkyo|   78|
|          Sima|   10|
|Master Dynamic|    5|
|           JBL|   60|
|     SunbriteTV|    2|
|       myCharge|    9|
|        TP-Link|   35|
|       Kenwood|  141|
|           RCA|   12|
|         Beats|    4|
|      Fujifilm|  119|
+--------------+-----+
only showing top 20 rows
```

Figure 16. GroupBy on brand column

Describe

The describe() function provides the summary about the column including minimum, maximum, count mean and stdev. Below is the code snippet of using describe on 'amountMax' column.



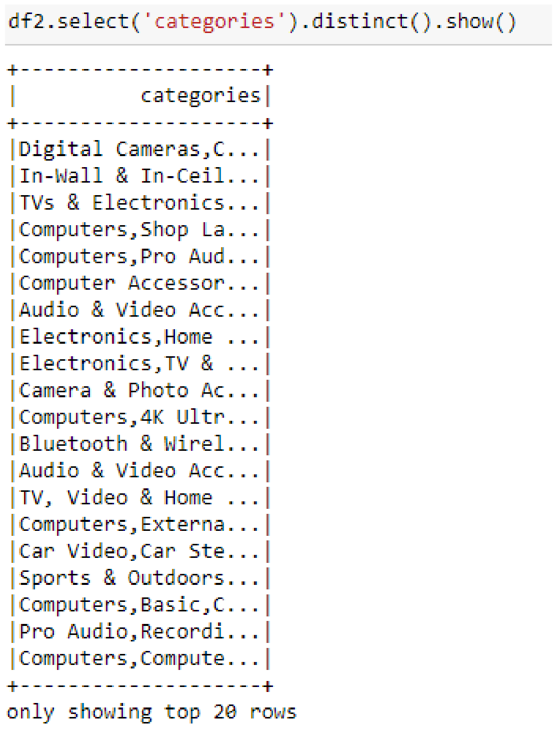
```
# descriptive statistics of the amountMax feature
df2.select("amountMax").describe().show()
```

```
+-------+-----------------+
|summary|        amountMax|
+-------+-----------------+
|  count|             6164|
|   mean| 430.948676184296|
| stddev|727.2681814904796|
|    min|              1.0|
|    max|          6499.99|
+-------+-----------------+
```

Figure 17. Descriptive statistics for amountMax feature

Distinct

The distinct command is used to get all the distinct values present in a particular column. Here, we are applying distinct() on categories column and it display the different category present in it.



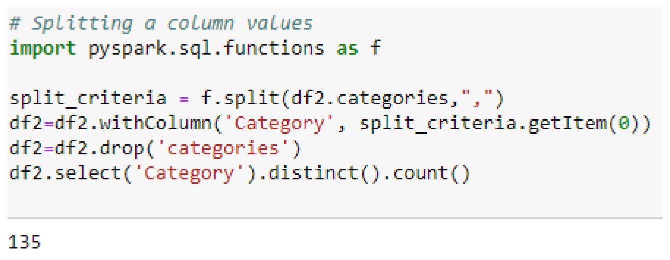
```
df2.select('categories').distinct().show()
```

```
+--------------------+
|          categories|
+--------------------+
|Digital Cameras,C...|
|In-Wall & In-Ceil...|
|TVs & Electronics...|
|Computers,Shop La...|
|Computers,Pro Aud...|
|Computer Accessor...|
|Audio & Video Acc...|
|Electronics,Home ...|
|Electronics,TV & ...|
|Camera & Photo Ac...|
|Computers,4K Ultr...|
|Bluetooth & Wirel...|
|Audio & Video Acc...|
|TV, Video & Home ...|
|Computers,Externa...|
|Car Video,Car Ste...|
|Sports & Outdoors...|
|Computers,Basic,C...|
|Pro Audio,Recordi...|
|Computers,Compute...|
+--------------------+
only showing top 20 rows
```

Figure 18. Checking distinct categories

Splitting column values

Here we need to perform below task splitting the category column.Then we can see that no of distinct categories has also drop a lot.



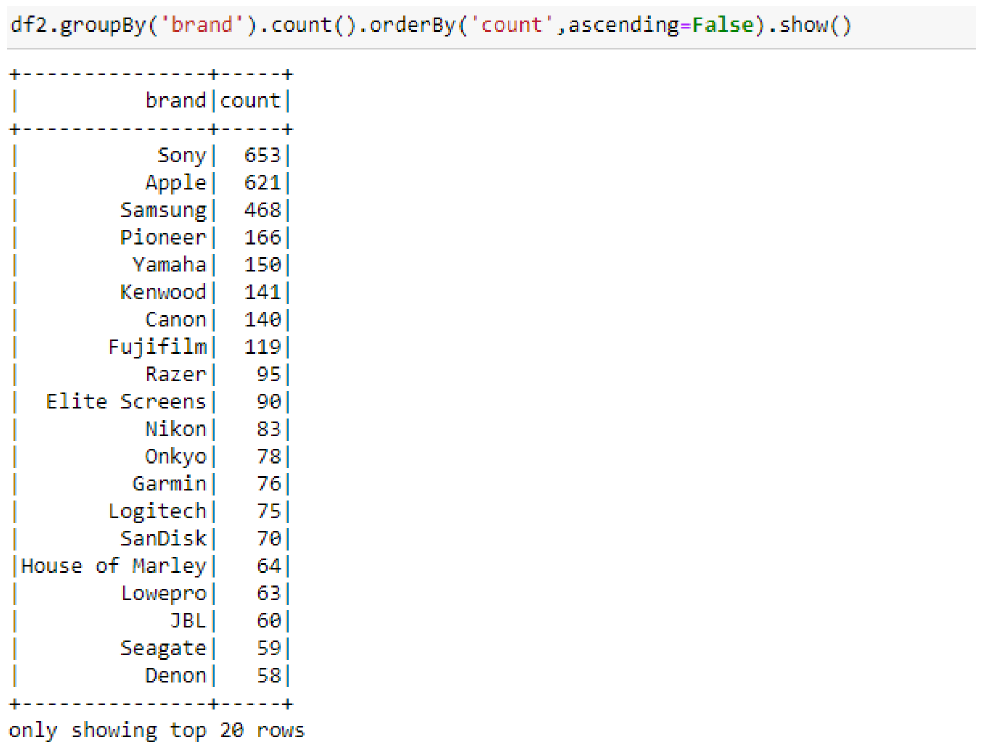
```
# Splitting a column values
import pyspark.sql.functions as f

split_criteria = f.split(df2.categories,",")
df2=df2.withColumn('Category', split_criteria.getItem(0))
df2=df2.drop('categories')
df2.select('Category').distinct().count()
```

```
135
```

Figure 19. Assigning correct categories

OrderBy

The orderBy() command is used to sort the particular column value count in a particular order either ascending or descending.



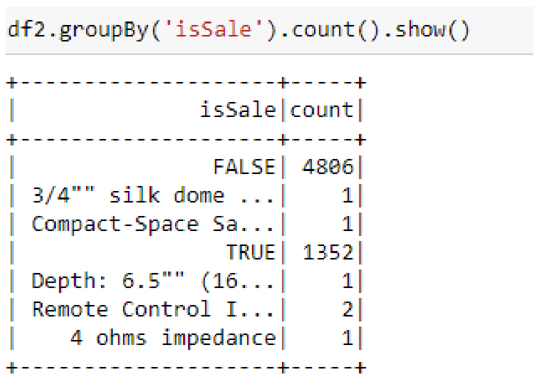
```
df2.groupBy('brand').count().orderBy('count',ascending=False).show()
```

```
+---------------+-----+
|          brand|count|
+---------------+-----+
|           Sony|  653|
|          Apple|  621|
|        Samsung|  468|
|        Pioneer|  166|
|         Yamaha|  150|
|        Kenwood|  141|
|          Canon|  140|
|       Fujifilm|  119|
|          Razer|   95|
|  Elite Screens|   90|
|          Nikon|   83|
|          Onkyo|   78|
|         Garmin|   76|
|       Logitech|   75|
|        SanDisk|   70|
|House of Marley|   64|
|        Lowepro|   63|
|            JBL|   60|
|        Seagate|   59|
|          Denon|   58|
+---------------+-----+
only showing top 20 rows
```

Figure 20. orderBy on brand column

Cleaning up the dependent column

'Sale' is one of the crucial column to work on, as it is a dependent feature for our machine-learning problem. Some of the values of 'sale' feature is invalid. Therefore, remove all these values and keep only those,valid, and has some significance.

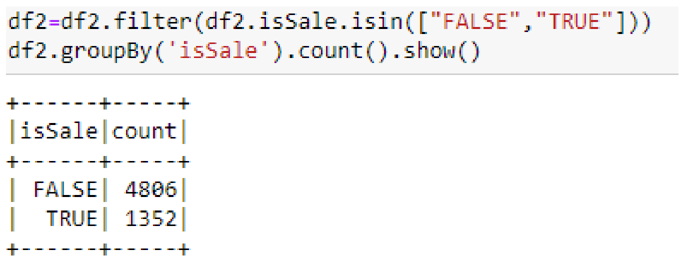


```
df2.groupBy('isSale').count().show()
```

```
+--------------------+-----+
|              isSale|count|
+--------------------+-----+
|               FALSE| 4806|
| 3/4"" silk dome ...|    1|
| Compact-Space Sa...|    1|
|                TRUE| 1352|
| Depth: 6.5"" (16...|    1|
| Remote Control I...|    2|
|    4 ohms impedance|    1|
+--------------------+-----+
```

Figure 21. Removing irrelevant values from sale column

The below code clean up the invalid values and contains only the actual values i.e. TRUE and FALSE. So now, the 'sale' feature is cleaned up and can be used for a machine-learning problem.



```
df2=df2.filter(df2.isSale.isin(["FALSE","TRUE"]))
df2.groupBy('isSale').count().show()
```

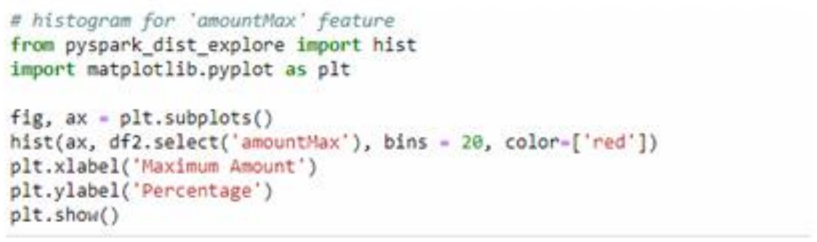
```
+------+-----+
|isSale|count|
+------+-----+
| FALSE| 4806|
|  TRUE| 1352|
+------+-----+
```

Figure 22. Verifying sale column

Exploratory Data Analysis

Histogram

Histogram is the univariate type of analysis. It provides information as what percent of value falls in the given range.Hereit can be seen that most of the data lies in 0 to 1500 range.



```
# histogram for 'amountMax' feature
from pyspark_dist_explore import hist
import matplotlib.pyplot as plt

fig, ax = plt.subplots()
hist(ax, df2.select('amountMax'), bins = 20, color=['red'])
plt.xlabel('Maximum Amount')
plt.ylabel('Percentage')
plt.show()
```

```python
# histogram for 'amountMax' feature
from pyspark_dist_explore import hist
import matplotlib.pyplot as plt

fig, ax = plt.subplots()
hist(ax, df2.select('amountMax'), bins = 20, color=['red'])
plt.xlabel('Maximum Amount')
plt.ylabel('Percentage')
plt.show()
```

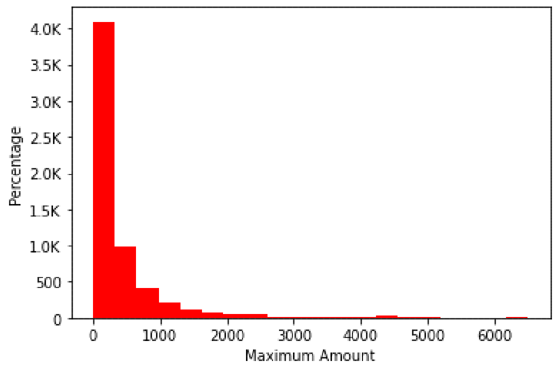


Figure 23. Histogram for amountMax feature

Boxplot and Violin plot

Boxplot provides the description of the data in terms of Quantiles like it tell about how much data lies in 25, 50 and 75 percentile and helps in the identification of outliers and skewness present in the data. Violin plot is almost similar to boxplot. It is a combination of boxplot and probability density function.

```python
import seaborn as sns

x=df2.select('amountMax').toPandas()
fig=plt.figure(figsize=(10,5))
ax=fig.add_subplot(1,2,1)
ax=sns.boxplot(data=x)

ax=fig.add_subplot(1,2,2)
ax=sns.violinplot(data=x)
```

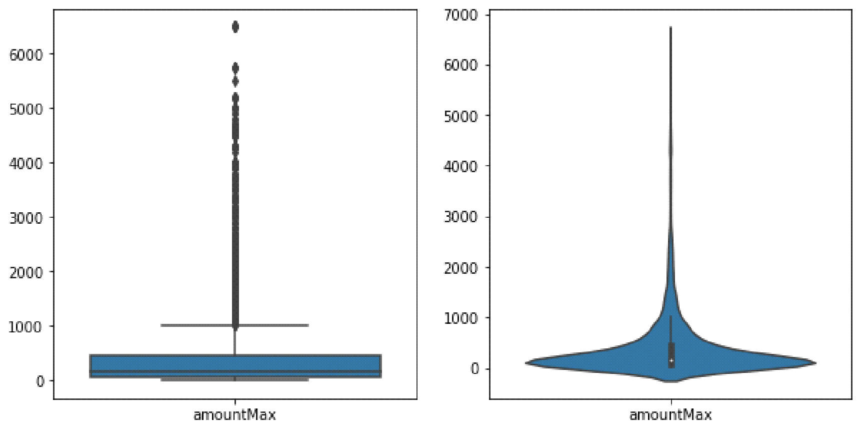
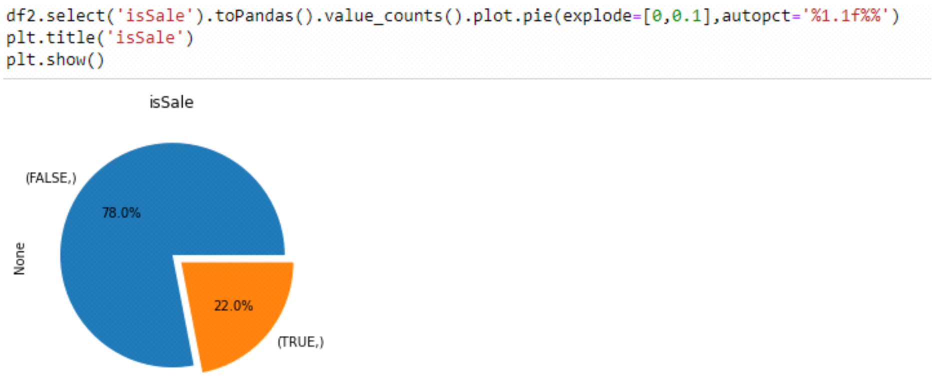


Figure 24. Box and Violin plot for amountMax feature

Pie chart

Pie chart displays the distribution of data into slices. Here it is seen that it is divided into 2one is 78% and other is 22%. By seeing this distribution, it is found that data is highly unbalanced.

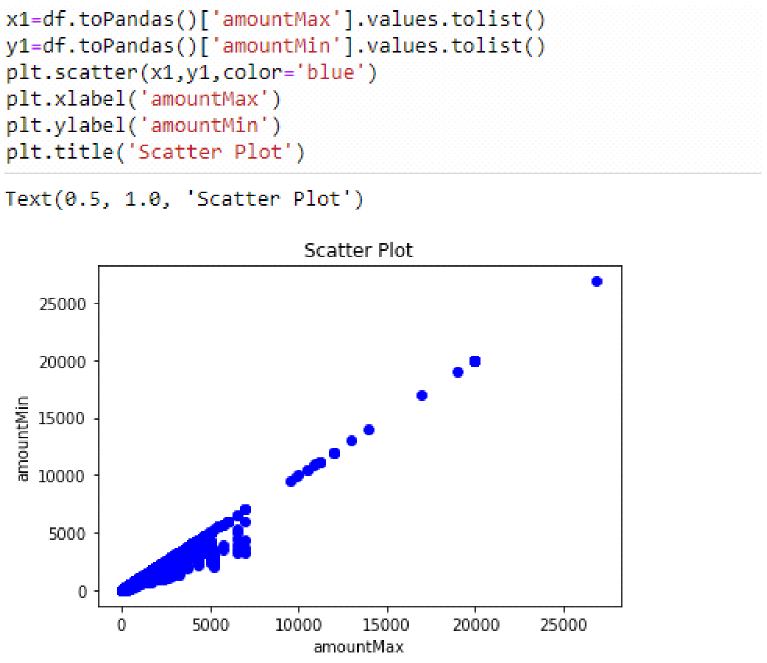


```
df2.select('isSale').toPandas().value_counts().plot.pie(explode=[0,0.1],autopct='%1.1f%%')
plt.title('isSale')
plt.show()
```

Figure 25. Pie chart for isSale feature

Scatter plot

Scatter plot is used to find relationship between two features. Here, it is seen that amountMax and amountMin feature are highly correlated with each other.



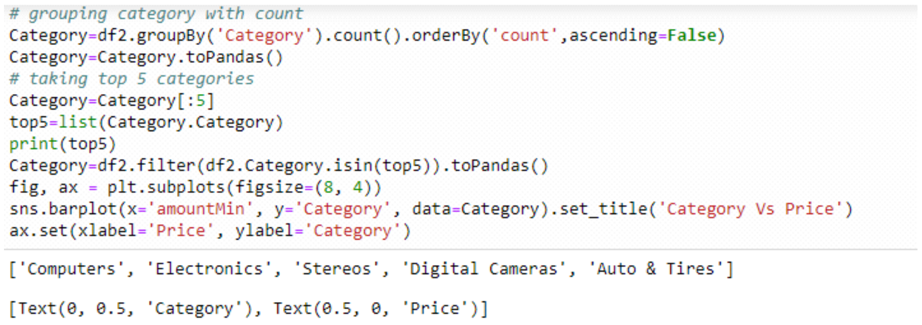
```
x1=df.toPandas()['amountMax'].values.tolist()
y1=df.toPandas()['amountMin'].values.tolist()
plt.scatter(x1,y1,color='blue')
plt.xlabel('amountMax')
plt.ylabel('amountMin')
plt.title('Scatter Plot')
```

```
Text(0.5, 1.0, 'Scatter Plot')
```

Figure 26. Scatterplot between amountMax and amountMin

Horizontal bar chart

The below bar chart is displaying the top five categories with their price. It is seen that Digital Camera is having the highest price and Auto & Tires category is having the lowest price.



```
# grouping category with count
Category=df2.groupBy('Category').count().orderBy('count',ascending=False)
Category=Category.toPandas()
# taking top 5 categories
Category=Category[:5]
top5=list(Category.Category)
print(top5)
Category=df2.filter(df2.Category.isin(top5)).toPandas()
fig, ax = plt.subplots(figsize=(8, 4))
sns.barplot(x='amountMin', y='Category', data=Category).set_title('Category Vs Price')
ax.set(xlabel='Price', ylabel='Category')
```

```
['Computers', 'Electronics', 'Stereos', 'Digital Cameras', 'Auto & Tires']

[Text(0, 0.5, 'Category'), Text(0.5, 0, 'Price')]
```

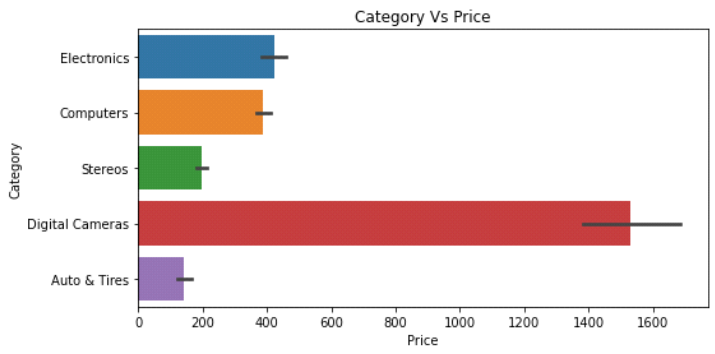
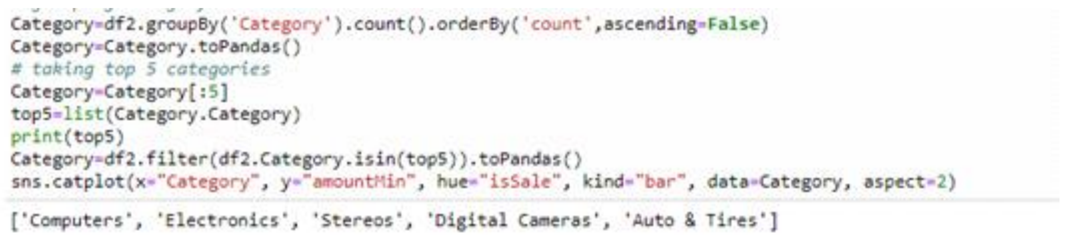


Figure 27. Horizontal Bar Chart for category feature

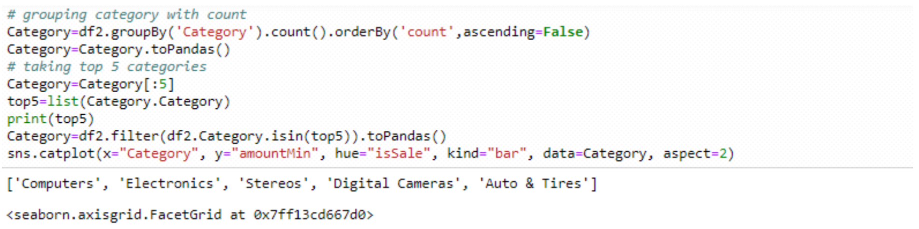
Vertical bar chart

The below chart displays the top 5 categories with their price and sale. It compare the sales of the different category with the price. The Sale column represents if there are any offers or not on the particular category.



```
Category=df2.groupBy('Category').count().orderBy('count',ascending=False)
Category=Category.toPandas()
# taking top 5 categories
Category=Category[:5]
top5=list(Category.Category)
print(top5)
Category=df2.filter(df2.Category.isin(top5)).toPandas()
sns.catplot(x="Category", y="amountMin", hue="isSale", kind="bar", data=Category, aspect=2)
```

```
['Computers', 'Electronics', 'Stereos', 'Digital Cameras', 'Auto & Tires']
```



```
# grouping category with count
Category=df2.groupBy('Category').count().orderBy('count',ascending=False)
Category=Category.toPandas()
# taking top 5 categories
Category=Category[:5]
top5=list(Category.Category)
print(top5)
Category=df2.filter(df2.Category.isin(top5)).toPandas()
sns.catplot(x="Category", y="amountMin", hue="isSale", kind="bar", data=Category, aspect=2)
```

```
['Computers', 'Electronics', 'Stereos', 'Digital Cameras', 'Auto & Tires']

<seaborn.axisgrid.FacetGrid at 0x7ff13cd667d0>
```

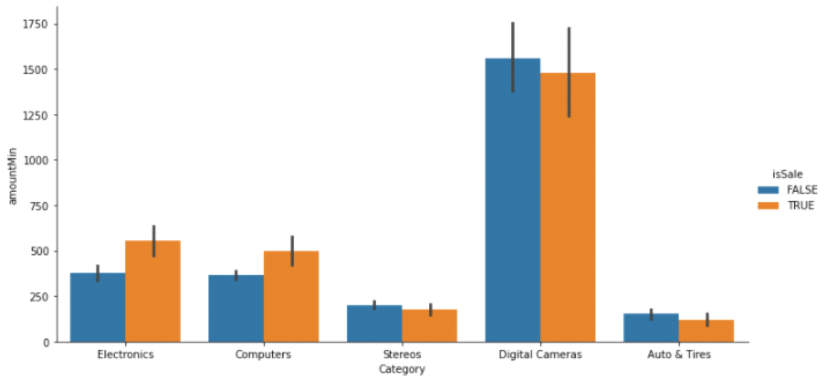
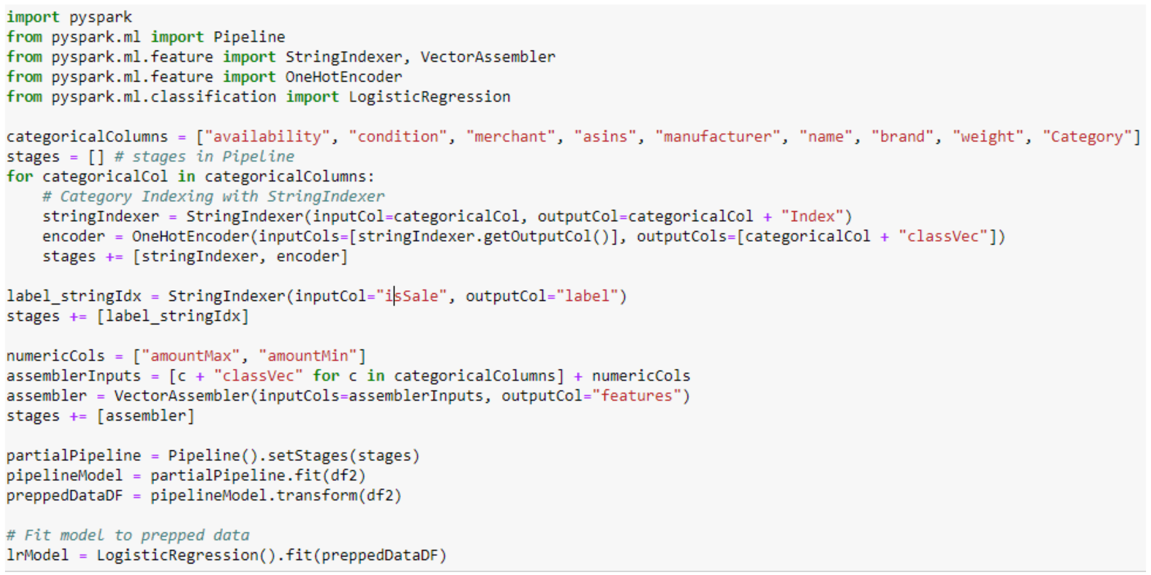


Figure 28. Vertical Bar Chart for category feature

Machine Learning

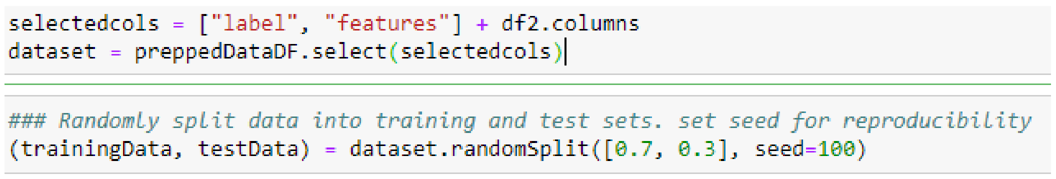
After doing the data pre-processing, analysis and visualizations now is the time to perform modelling on the data to make predictive analytics on it. From the analysis, it is found that 'Sale' feature is a dependent feature and it has only two classes of data. Therefore, it is a classification problem and here we are using Logistic Regression model. The data contains both numerical and categorical columns. In pyspark it is achieved through String Indexer. Once all the data is converted into numerical form then it is passed through Vector Assembler for transformation i.e. converting all the feature values into a single list.



```
import pyspark
from pyspark.ml import Pipeline
from pyspark.ml.feature import StringIndexer, VectorAssembler
from pyspark.ml.feature import OneHotEncoder
from pyspark.ml.classification import LogisticRegression

categoricalColumns = ["availability", "condition", "merchant", "asins", "manufacturer", "name", "brand", "weight", "Category"]
stages = [] # stages in Pipeline
for categoricalCol in categoricalColumns:
    # Category Indexing with StringIndexer
    stringIndexer = StringIndexer(inputCol=categoricalCol, outputCol=categoricalCol + "Index")
    encoder = OneHotEncoder(inputCols=[stringIndexer.getOutputCol()], outputCols=[categoricalCol + "classVec"])
    stages += [stringIndexer, encoder]

label_stringIdx = StringIndexer(inputCol="isSale", outputCol="label")
stages += [label_stringIdx]

numericCols = ["amountMax", "amountMin"]
assemblerInputs = [c + "classVec" for c in categoricalColumns] + numericCols
assembler = VectorAssembler(inputCols=assemblerInputs, outputCol="features")
stages += [assembler]

partialPipeline = Pipeline().setStages(stages)
pipelineModel = partialPipeline.fit(df2)
preppedDataDF = pipelineModel.transform(df2)

# Fit model to prepped data
lrModel = LogisticRegression().fit(preppedDataDF)
```

Figure 29. Transforming features

After getting the feature values in a list and target class the data, need to be splitted randomly into training and test dataset with 70% as training and 30% as testing.

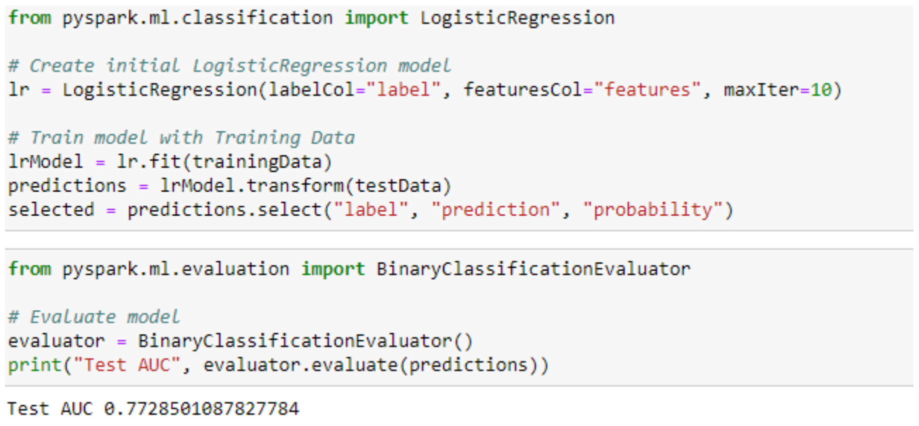


```
selectedcols = ["label", "features"] + df2.columns
dataset = preppedDataDF.select(selectedcols)
```

```
### Randomly split data into training and test sets. set seed for reproducibility
(trainingData, testData) = dataset.randomSplit([0.7, 0.3], seed=100)
```

Figure 30. Selecting features for modelling and splitting train and test data

Then at the end, the model is trained on training data and validation is made on testing data. Then finally, the model is evaluated through different metrics. As this is a classification problem, so here we have use AUC metric. We can see that AUC is 0.77, which is good.



```
from pyspark.ml.classification import LogisticRegression

# Create initial LogisticRegression model
lr = LogisticRegression(labelCol="label", featuresCol="features", maxIter=10)

# Train model with Training Data
lrModel = lr.fit(trainingData)
predictions = lrModel.transform(testData)
selected = predictions.select("label", "prediction", "probability")
```

```
from pyspark.ml.evaluation import BinaryClassificationEvaluator

# Evaluate model
evaluator = BinaryClassificationEvaluator()
print("Test AUC", evaluator.evaluate(predictions))
```

```
Test AUC 0.7728501087827784
```

Figure 31. Training model and predicting

3. Discussion

In this section, we are going to discuss the overall project from the initial stage to the final stage (Data extraction to Data Modelling). The data for this project is related to electronic product and pricing and is extracted from kaggle and we have data for 15000 products. The main aim of the project is to identify suitable pricing strategies. By analyzing the data, it is found that there exist some null values in the data. Therefore, we have remove those null values and there are some duplicate records present so we have remove them also as it does not have any significance to keep duplicate records. Apart from this we have rename some of the columns as there exist '.' between them and it cause issue later on. In addition, we have done some of the insignificant features. The category column contains a collection of category for a product so it needs to be cleaned up. From the visualizationpoint, we have drawn various plots to identify trends and pattern in the data. From the Histogram, it is found that Maximum amount for the electronic product lies in the range 0-1500. In addition, we have divided an amount data into percentiles using boxplot and violin plot and see the distribution of data in each percentile. From the scatter plot, we have found that amountMin and amountMax columns are related to each other. From the pie chart, we found the distribution of target class.From the bar chart we have plot the top 5 categories with their price and sale and it is seen that Digital Camera is the most costly and also has more sale offers. Therefore, it can be concluded that more sale is given to those product whose price is high.

From the modelling point, we have seen that most of the data is of categorical and numerical type so to perform modelling this categorical data needs to be converted into numerical type. In addition, the target class variable has only two classes therefore it is a classification problem and for classification we have used here Logistic Regression algorithm. The metric that is used here is AUC as data is highly unbalanced so it is better to use this instead of Accuracy and we are getting AUC as 0.77, which is good. By seeing,a metric value it can be said that model is performing really well on the given data.

4. Conclusion

From the above discussion, it can be concluded that there are some major challenges in starting of the project to end. One of the first challenge is to learn new language Pyspark and properly download the required software to use it. There are many issues, which comes in between like memory, data cleaning, data pre-processing, and what plot to use to get most insights. The visualizations is performed on the top values to get the relevant result. It is observe that major brands lead the way with prices, market share. To make future predictions Machine Learning was utilized and Logistic Regression model is giving a good result. So it can be said that lot has been gained by working on this project.

5. References

Armbrust, M. X. (2015). Spark sql: Relational data processing in spark. *In Proceedings of the 2015 ACM SIGMOD international conference on management of data* .
Datafiniti. (2018). *Electronics products and Pricing data*. Retrieved from kaggle: https://www.kaggle.com/datafiniti/electronic-products-prices
L, A. K. (2019). *Guide to install Spark and use PySpark from Jupyter in Windows*. Retrieved from Big Data Made Simple: https://bigdata-madesimple.com/guide-to-install-spark-and-use-pyspark-from-jupyter-in-windows/
Meng, X. B. (2016). Mllib: Machine learning in apache spark. . *The Journal of Machine Learning Research* .
Sagiroglu, S. &. (2013). Big data: A review. *In 2013 international conference on collaboration technologies and systems (CTS)* .
Zaharia, M. C. (2010). Spark: Cluster computing with working sets. *HotCloud* .
Zaharia, M. X. (2016). Apache spark: a unified engine for big data processing. . *Communications of the ACM* .

Appendix

Please see the link for to access the codes used for this study.

https://github.com/YendamuriJ/ElectronicProducts-And-Pricing-Using-Pyspark/blob/main/pyspark2_code.ipynb

And please see the below link for source code
https://drive.google.com/file/d/1BUP963v9M0R7_IruKV4zLisEfWENEWbW/view?usp=sharing