Data Wrangling in Pyspark

Ramcharan Kakarla
Feb 4 · 5 min read

Data Science specialists spend majority of their time in data preparation. It is estimated to account for 70 to 80% of total time taken for model development. Often times new features designed via feature engineering aid the model performances. Spark gained a lot of momentum with the advent of big data. With limited capacity of traditional systems, the push for distributed computing is more than ever. When I started my journey with pyspark two years ago there were not many web resources with exception of offical documentation. The intent of this article is to help the data aspirants who are trying to migrate from other languages to pyspark.

Below collection is stack of most commonly used functions that are useful for data manipulations. The data used for this exercise is available on Kaggle (https://www.kaggle.com/neuromusic/avocado-prices)

Reading Data

All the following operations were performed on spark version 2.4. We primarily are going to see the operations performed on data frames

```
# Read from filepath on distributed system
file_location = "/FileStore/tables/avocado.csv"
file_type = "csv"
infer_schema = "false"
first_row_is_header = "true"
delimiter = ","

df = spark.read.format(file_type) \
  .option("inferSchema", infer_schema) \
  .option("header", first_row_is_header) \
  .option("sep", delimiter) \
  .load(file_location)


# Read from existing table on hive
 df=spark.sql('select * from tablename')
```

Metadata of the data frame

```
df.printSchema()
```

```
root
 |-- _c0: string (nullable = true)
 |-- Date: string (nullable = true)
 |-- AveragePrice: string (nullable = true)
 |-- Total Volume: string (nullable = true)
 |-- 4046: string (nullable = true)
 |-- 4225: string (nullable = true)
 |-- 4770: string (nullable = true)
 |-- Total Bags: string (nullable = true)
 |-- Small Bags: string (nullable = true)
 |-- Large Bags: string (nullable = true)
```

```
|-- year: string (nullable = true)
|-- region: string (nullable = true)
```

Glimpse of the data

```
df.show()
```

```
+---+----------+------------+------------+-------+---------+------+----------+----------+----------+-----------+------------+----+------+
|_c0|      Date|AveragePrice|Total Volume|   4046|     4225|  4770|Total Bags|Small Bags|Large Bags|XLarge Bags|        type|year|region|
+---+----------+------------+------------+-------+---------+------+----------+----------+----------+-----------+------------+----+------+
|  0|2015-12-27|        1.33|    64236.62|1036.74| 54454.85| 48.16|   8696.87|   8603.62|     93.25|        0.0|conventional|2015|Albany|
|  1|2015-12-20|        1.35|    54876.98| 674.28| 44638.81| 58.33|   9505.56|   9408.07|     97.49|        0.0|conventional|2015|Albany|
|  2|2015-12-13|        0.93|   118220.22|  794.7|109149.67| 130.5|   8145.35|   8042.21|    103.14|        0.0|conventional|2015|Albany|
|  3|2015-12-06|        1.08|    78992.15| 1132.0| 71976.41| 72.58|   5811.16|    5677.4|    133.76|        0.0|conventional|2015|Albany|
|  4|2015-11-29|        1.28|     51039.6| 941.48| 43838.39| 75.78|   6183.95|   5986.26|    197.69|        0.0|conventional|2015|Albany|
|  5|2015-11-22|        1.26|    55979.78|1184.27| 48067.99| 43.61|   6683.91|   6556.47|    127.44|        0.0|conventional|2015|Albany|
|  6|2015-11-15|        0.99|    83453.76|1368.92| 73672.72| 93.26|   8318.86|   8196.81|    122.05|        0.0|conventional|2015|Albany|
|  7|2015-11-08|        0.98|   109428.33| 703.75|101815.36|  80.0|   6829.22|   6266.85|    562.37|        0.0|conventional|2015|Albany|
|  8|2015-11-01|        1.02|    99811.42|1022.15| 87315.57| 85.34|  11388.36|  11104.53|    283.83|        0.0|conventional|2015|Albany|
|  9|2015-10-25|        1.07|    74338.76|  842.4| 64757.44| 113.0|   8625.92|   8061.47|    564.45|        0.0|conventional|2015|Albany|
| 10|2015-10-18|        1.12|    84843.44| 924.86| 75595.85|117.07|   8205.66|   7877.86|     327.8|        0.0|conventional|2015|Albany|
| 11|2015-10-11|        1.28|    64489.17|1582.03| 52677.92|105.32|   10123.9|   9866.27|    257.63|        0.0|conventional|2015|Albany|
| 12|2015-10-04|        1.31|     61007.1|2268.32| 49880.67|101.36|   8756.75|   8379.98|    376.77|        0.0|conventional|2015|Albany|
| 13|2015-09-27|        0.99|   106803.39|1204.88| 99409.21|154.84|   6034.46|   5888.87|    145.59|        0.0|conventional|2015|Albany|
| 14|2015-09-20|        1.33|    69759.01|1028.03| 59313.12| 150.5|   9267.36|    8489.1|    778.26|        0.0|conventional|2015|Albany|
| 15|2015-09-13|        1.28|    76111.27| 985.73| 65696.86| 142.0|   9286.68|   8665.19|    621.49|        0.0|conventional|2015|Albany|
| 16|2015-09-06|        1.11|    99172.96| 879.45| 90062.62|240.79|    7990.1|   7762.87|    227.23|        0.0|conventional|2015|Albany|
| 17|2015-08-30|        1.07|   105693.84| 689.01| 94362.67|335.43|  10306.73|  10218.93|      87.8|        0.0|conventional|2015|Albany|
| 18|2015-08-23|        1.34|    79992.09| 733.16| 67933.79|444.78|  10880.36|  10745.79|    134.57|        0.0|conventional|2015|Albany|
| 19|2015-08-16|        1.33|    80043.78| 539.65| 68666.01| 394.9|  10443.22|  10297.68|    145.54|        0.0|conventional|2015|Albany|
+---+----------+------------+------------+-------+---------+------+----------+----------+----------+-----------+------------+----+------+
only showing top 20 rows
```

Count the number of records

```
count_rows=df.count()
print ('Total data count is '+str(count_rows))
```

Subset Data

```
# Method 1 - using lists
columns_to_subset=['Total Volume','AveragePrice']
df1=df.select(*[columns_to_subset])

# Method 2 - Column names
df1=df.select('Total Volume','AveragePrice')

# Method 3 - Indexes
df1=df.select(df[2],df[3])
```

Count Missing Values

```
df.where(df['Total Volume'].isNull()).count()
```

One way Frequency

```
df.groupby(df['type']).count().show()
```

```
+------------+-----+
|        type|count|
+------------+-----+
|     organic| 9123|
|conventional| 9126|
+------------+-----+
```

Crosstab

```
df.where(df['region']=='Albany').crosstab('type','region').show()
```

```
+------------+------+
| type_region|Albany|
+------------+------+
|conventional|   169|
|     organic|   169|
+------------+------+
```

```
# This will give a quick glimpse to data, if columns are not
mentioned all the numeric columns stats are produced
columns_to_analyze=['Total Volume','AveragePrice']
df.select(*[columns_to_analyze]).describe().show()
```

```
+-------+------------------+------------------+
|summary|      Total Volume|      AveragePrice|
+-------+------------------+------------------+
|  count|             18249|             18249|
|   mean| 850644.0130089332|1.405978409775882 5|
| stddev|3453545.3553994684|0.4026765554955525|
|    min|           1000.86|              0.44|
|    max|          99982.71|              3.25|
+-------+------------------+------------------+
```

Casting a variable

```
df = df.withColumn('Total Volume',df['Total Volume'].cast("float"))
```

Median Value Calculation

```
#Three parameters have to be passed through approxQuantile function
#1. col – the name of the numerical column
#2. probabilities – a list of quantile probabilities Each number
must belong to [0, 1]. For example 0 is the minimum, 0.5 is the
median, 1 is the maximum.
#3. relativeError – The relative target precision to achieve (>= 0).
If set to zero, the exact quantiles are computed, which could be
very expensive. Note that values greater than 1 are accepted but
give the same result as 1.

median=df.approxQuantile('Total Volume',[0.5],0.1)
print ('The median of Total Volume is '+str(median))
```

```
from pyspark.sql.functions import col, countDistinct
column_name='region'
count_distinct=df.agg(countDistinct(col(column_name).alias("distinct
_counts"))).head()[0]print ('The number of distinct values of
'+column_name+ ' is ' +str(count_distinct))
```

Distinct Levels

```
column_name='type'
df.select(column_name).distinct().show()
```

```
+------------+
|        type|
+------------+
|     organic|
|conventional|
+------------+
```

Filter Data

```
# Lets filter the organic type avocados within Albany region
filtered_count=df.filter((df['type']=='organic') &
(df.region=='Albany')).count()
print ('subset data count is '+str(filtered_count))
```

Rename Columns

```
df=df.withColumnRenamed("Total Volume","Total_Volume")

#Muliple columns
df=df.withColumnRenamed("Total
```

```
#Verify output
df.printSchema()
```

Create new columns

```
#using withColumn
df1=df.select('Total_Volume','Total_Bags').withColumn('avg_volume_ba
g',df['Total_Volume']/df['Total_Bags'])

# Alternative method
df = df.withColumn('Total_Bags',df['Total_Bags'].cast("float"))
df1=df.select('_c0','Total_Volume','Total_Bags',
(df['Total_Volume']/df['Total_Bags']).alias('avg_volume_bag'))
df1.show()
```

```
+---+------------+----------+------------------+
|_c0|Total_Volume|Total_Bags|    avg_volume_bag|
+---+------------+----------+------------------+
|  0|    64236.62|   8696.87| 7.386176892167227|
|  1|    54876.98|   9505.56| 5.773145711499223|
|  2|   118220.22|   8145.35|14.513829035294233|
|  3|    78992.15|   5811.16|13.593180417260161|
|  4|     51039.6|   6183.95| 8.253559610035113|
|  5|    55979.78|   6683.91|  8.37530426671794|
|  6|    83453.76|   8318.86| 10.03187387282264|
|  7|   109428.33|   6829.22| 16.02354656643675|
|  8|    99811.42|  11388.36| 8.764336462299045|
|  9|    74338.76|   8625.92| 8.618067230601085|
| 10|    84843.44|   8205.66|10.339623611560047|
| 11|    64489.17|   10123.9| 6.369992728762788|
| 12|     61007.1|   8756.75| 6.966865739286836|
| 13|   106803.39|   6034.46|17.698914454046236|
| 14|    69759.01|   9267.36| 7.527386997608057|
| 15|    76111.27|   9286.68|  8.195746596057021|
```

```
| 17|   105693.84|  10306.73|10.254837270700312|
| 18|    79992.09|  10880.36|  7.35197099777238|
| 19|    80043.78|  10443.22| 7.664665050224629|
+---+------------+----------+------------------+
only showing top 20 rows
```

Create multiple columns

```
# Import Necessary data types
from pyspark.sql.functions import udf,split
from pyspark.sql.types import StructType, StructField, IntegerType,
StringType, ArrayType

# Create a function for all the data maipulations
def new_cols(Total_Volume,AveragePrice):
 if Total_Volume<44245: Volume_Category='Small'
 elif Total_Volume<850644: Volume_Category='Medium'
 else: Volume_Category='Large'
 if AveragePrice<1.25: Price_Category='Low'
 elif AveragePrice<1.4: Price_Category='Mid'
 else: Price_Category='High'
 return Volume_Category,Price_Category

# Apply the user defined function on the dataframe

udfB=udf(new_cols,StructType([StructField("Volume_Category",
StringType(), True),StructField("Price_Category", StringType(),
True)]))
df2=df.select('_c0','Total_Volume','AveragePrice').withColumn("newca
t",udfB("Total_Volume","AveragePrice"))

# Unbundle the struct type columns into individual columns and drop
the struct type
df3 =
df2.select('_c0','Total_Volume','AveragePrice','newcat').withColumn(
'Volume_Category',
df2.newcat.getItem('Volume_Category')).withColumn('Price_Category',
df2.newcat.getItem('Price_Category')).drop('newcat')
df3.show()
```

```
+---+------------+-----------+--------------+--------------+
|_c0|Total_Volume|AveragePrice|Volume_Category|Price_Category|
+---+------------+-----------+--------------+--------------+
|  0|    64236.62|       1.33|        Medium|          High|
```

```
|  3|    78992.15|        1.08|         Medium|          High|
|  4|     51039.6|        1.28|         Medium|          High|
|  5|    55979.78|        1.26|         Medium|          High|
|  6|    83453.76|        0.99|         Medium|          High|
|  7|   109428.33|        0.98|         Medium|          High|
|  8|    99811.42|        1.02|         Medium|          High|
|  9|    74338.76|        1.07|         Medium|          High|
| 10|    84843.44|        1.12|         Medium|          High|
| 11|    64489.17|        1.28|         Medium|          High|
| 12|     61007.1|        1.31|         Medium|          High|
| 13|   106803.39|        0.99|         Medium|          High|
| 14|    69759.01|        1.33|         Medium|          High|
| 15|    76111.27|        1.28|         Medium|          High|
| 16|    99172.96|        1.11|         Medium|          High|
| 17|   105693.84|        1.07|         Medium|          High|
| 18|    79992.09|        1.34|         Medium|          High|
| 19|    80043.78|        1.33|         Medium|          High|
+---+------------+------------+---------------+--------------+
only showing top 20 rows
```

String Operations — Concatenation

```
from pyspark.sql.functions import concat
df3=df3.withColumn('VolumePrice_Category',concat(df3.Volume_Category
,df3.Price_Category))
df3.show()
```

```
+---+------------+------------+---------------+--------------+--------------------+
|_c0|Total_Volume|AveragePrice|Volume_Category|Price_Category|VolumePrice_Category|
+---+------------+------------+---------------+--------------+--------------------+
|  0|    64236.62|        1.33|         Medium|          High|          MediumHigh|
|  1|    54876.98|        1.35|         Medium|          High|          MediumHigh|
|  2|   118220.22|        0.93|         Medium|          High|          MediumHigh|
|  3|    78992.15|        1.08|         Medium|          High|          MediumHigh|
|  4|     51039.6|        1.28|         Medium|          High|          MediumHigh|
|  5|    55979.78|        1.26|         Medium|          High|          MediumHigh|
|  6|    83453.76|        0.99|         Medium|          High|          MediumHigh|
|  7|   109428.33|        0.98|         Medium|          High|          MediumHigh|
|  8|    99811.42|        1.02|         Medium|          High|          MediumHigh|
|  9|    74338.76|        1.07|         Medium|          High|          MediumHigh|
| 10|    84843.44|        1.12|         Medium|          High|          MediumHigh|
| 11|    64489.17|        1.28|         Medium|          High|          MediumHigh|
```

```
| 14|    69759.01|        1.33|         Medium|          High|          MediumHigh|
| 15|    76111.27|        1.28|         Medium|          High|          MediumHigh|
| 16|    99172.96|        1.11|         Medium|          High|          MediumHigh|
| 17|   105693.84|        1.07|         Medium|          High|          MediumHigh|
| 18|    79992.09|        1.34|         Medium|          High|          MediumHigh|
| 19|    80043.78|        1.33|         Medium|          High|          MediumHigh|
+---+------------+------------+---------------+--------------+--------------------+
only showing top 20 rows
```

String Operations — ChangeCases

```
from pyspark.sql.functions import concat,trim,upper
df3=df3.withColumn('Price_Category',trim(upper(df3.Price_Category)))
df3.show()
```

```
+---+------------+------------+---------------+--------------+--------------------+
|_c0|Total_Volume|AveragePrice|Volume_Category|Price_Category|VolumePrice_Category|
+---+------------+------------+---------------+--------------+--------------------+
|  0|    64236.62|        1.33|         Medium|          HIGH|          MediumHigh|
|  1|    54876.98|        1.35|         Medium|          HIGH|          MediumHigh|
|  2|   118220.22|        0.93|         Medium|          HIGH|          MediumHigh|
|  3|    78992.15|        1.08|         Medium|          HIGH|          MediumHigh|
|  4|     51039.6|        1.28|         Medium|          HIGH|          MediumHigh|
|  5|    55979.78|        1.26|         Medium|          HIGH|          MediumHigh|
|  6|    83453.76|        0.99|         Medium|          HIGH|          MediumHigh|
|  7|   109428.33|        0.98|         Medium|          HIGH|          MediumHigh|
|  8|    99811.42|        1.02|         Medium|          HIGH|          MediumHigh|
|  9|    74338.76|        1.07|         Medium|          HIGH|          MediumHigh|
| 10|    84843.44|        1.12|         Medium|          HIGH|          MediumHigh|
| 11|    64489.17|        1.28|         Medium|          HIGH|          MediumHigh|
| 12|     61007.1|        1.31|         Medium|          HIGH|          MediumHigh|
| 13|   106803.39|        0.99|         Medium|          HIGH|          MediumHigh|
| 14|    69759.01|        1.33|         Medium|          HIGH|          MediumHigh|
| 15|    76111.27|        1.28|         Medium|          HIGH|          MediumHigh|
| 16|    99172.96|        1.11|         Medium|          HIGH|          MediumHigh|
| 17|   105693.84|        1.07|         Medium|          HIGH|          MediumHigh|
| 18|    79992.09|        1.34|         Medium|          HIGH|          MediumHigh|
| 19|    80043.78|        1.33|         Medium|          HIGH|          MediumHigh|
+---+------------+------------+---------------+--------------+--------------------+
only showing top 20 rows
```

Update a column value

```
from pyspark.sql.functions import *
df4 =
```

```
+---+------------+------------+---------------+-------------+--------------------+
|_c0|Total_Volume|AveragePrice|Volume_Category|Price_Category|VolumePrice_Category|
+---+------------+------------+---------------+-------------+--------------------+
|  0|    64236.62|        1.33|            Mid|         HIGH|          MediumHigh|
|  1|    54876.98|        1.35|            Mid|         HIGH|          MediumHigh|
|  2|   118220.22|        0.93|            Mid|         HIGH|          MediumHigh|
|  3|    78992.15|        1.08|            Mid|         HIGH|          MediumHigh|
|  4|     51039.6|        1.28|            Mid|         HIGH|          MediumHigh|
|  5|    55979.78|        1.26|            Mid|         HIGH|          MediumHigh|
|  6|    83453.76|        0.99|            Mid|         HIGH|          MediumHigh|
|  7|   109428.33|        0.98|            Mid|         HIGH|          MediumHigh|
|  8|    99811.42|        1.02|            Mid|         HIGH|          MediumHigh|
|  9|    74338.76|        1.07|            Mid|         HIGH|          MediumHigh|
| 10|    84843.44|        1.12|            Mid|         HIGH|          MediumHigh|
| 11|    64489.17|        1.28|            Mid|         HIGH|          MediumHigh|
| 12|     61007.1|        1.31|            Mid|         HIGH|          MediumHigh|
| 13|   106803.39|        0.99|            Mid|         HIGH|          MediumHigh|
| 14|    69759.01|        1.33|            Mid|         HIGH|          MediumHigh|
| 15|    76111.27|        1.28|            Mid|         HIGH|          MediumHigh|
| 16|    99172.96|        1.11|            Mid|         HIGH|          MediumHigh|
| 17|   105693.84|        1.07|            Mid|         HIGH|          MediumHigh|
| 18|    79992.09|        1.34|            Mid|         HIGH|          MediumHigh|
| 19|    80043.78|        1.33|            Mid|         HIGH|          MediumHigh|
+---+------------+------------+---------------+-------------+--------------------+
only showing top 20 rows
```

Drop a column

```
df4=df4.drop('VolumePrice_Category')

#Verify output
df4.columns
```

Sorting

```
df4=df4.sort(col("Total_Volume").desc())
df4.show()
```

```
+---+------------+------------+---------------+-------------+
|_c0|Total_Volume|AveragePrice|Volume_Category|Price_Category|
```

```
| 47| 6.1034456E7|         0.77|          Large|           HIGH|
| 46| 5.2288696E7|         0.76|          Large|           HIGH|
| 34|  4.729392E7|         1.09|          Large|           HIGH|
| 33| 4.6324528E7|         0.82|          Large|           HIGH|
| 47|  4.465546E7|         0.89|          Large|           HIGH|
|  0| 4.3409836E7|         1.03|          Large|           HIGH|
|  6| 4.3167808E7|         0.97|          Large|           HIGH|
|  9|  4.293982E7|         1.08|          Large|           HIGH|
| 34| 4.2867608E7|         0.84|          Large|           HIGH|
| 49| 4.2140392E7|         0.94|          Large|           HIGH|
|  1| 4.1386316E7|         1.05|          Large|           HIGH|
| 34| 4.1291704E7|         0.96|          Large|           HIGH|
| 46| 4.1077472E7|         0.87|          Large|           HIGH|
|  3| 4.0741216E7|         1.07|          Large|           HIGH|
|  2| 4.0449604E7|         1.09|          Large|           HIGH|
| 29|  4.023126E7|         0.97|          Large|           HIGH|
|  8|  4.017164E7|         1.09|          Large|           HIGH|
|  4| 4.0021528E7|         1.06|          Large|           HIGH|
| 30| 4.0019076E7|         0.94|          Large|           HIGH|
+---+------------+-------------+---------------+---------------+
only showing top 20 rows
```

Save as hive table

```
df3.write.saveAsTable('avocado')
```

Save as text file

```
df3.write.format("csv").option("delimiter",
",").save('avocado_textfile')
```

Convert to Pandas

```
import pandas as pd
pandas_df=df3.toPandas()
```

	_c0	Total_Volume	AveragePrice	Volume_Category	Price_Category	\
0	0	6.423662e+04	1.33	Medium	HIGH	
1	1	5.487698e+04	1.35	Medium	HIGH	
2	2	1.182202e+05	0.93	Medium	HIGH	
3	3	7.899215e+04	1.08	Medium	HIGH	
4	4	5.103960e+04	1.28	Medium	HIGH	
5	5	5.597978e+04	1.26	Medium	HIGH	
6	6	8.345376e+04	0.99	Medium	HIGH	
7	7	1.094283e+05	0.98	Medium	HIGH	
8	8	9.981142e+04	1.02	Medium	HIGH	
9	9	7.433876e+04	1.07	Medium	HIGH	
10	10	8.484344e+04	1.12	Medium	HIGH	
11	11	6.448917e+04	1.28	Medium	HIGH	
12	12	6.100710e+04	1.31	Medium	HIGH	
13	13	1.068034e+05	0.99	Medium	HIGH	
14	14	6.975901e+04	1.33	Medium	HIGH	
15	15	7.611127e+04	1.28	Medium	HIGH	
16	16	9.917296e+04	1.11	Medium	HIGH	

Add a montonically increasing id

```
# The generated ID is guaranteed to be monotonically increasing and
unique, but not consecutive.
df5 = df4.withColumn(“new_id”, monotonically_increasing_id())
```

Joins

```
# The join will include both keys from the tables. Common key can be
explicitly dropped using a drop statement or subset of columns
needed after join can be selected
# inner, outer, left_outer, right_outer, leftsemi joins are
available

joined_df = df3.join(df1, df1[‘_c0’] == df3[‘_c0’], ‘inner’)
joined_df.show()

#Note :since join key is not unique, there will be multiple records
on each join key if you use this data
```

. . .