Kaggle Used Car Dataset

Group 19: J.Mo Yang, Alex Pearce, Raleigh Christian, and Jing Wang

Introduction

We will be using two different data sets provided on Kaggle to examine the trends in the private party car sales industry. We will be using PySpark to run batch analysis of the data and compare the used car prices of two periods. This provides a broader context to the current rise in the value of used cars, given that Q1 was the impetus to the 3.5% increase in the used car market. However, there are significant gaps in this analysis within the sphere of private party, used car sales. Globally, this market segment encompasses approximately $40 million in sales, $17 million of which originated in the United States.

Nevertheless, there were also limitations of the data, which we will discuss in detail in the *Data* section of this report.

Business Problem

In light of the recent surge in used car prices due to supply-chain issues, we wanted to examine the price trend in the private (secondary) market for used cars. Also, we wanted to examine if any features of the used car has any impact on the price changes of the used cars (models, paint color, etc.)

Set Up Process

```
#Install Pyspark
!pip install pyspark
#Install findspark
!pip install findspark
```

```
Requirement already satisfied: pyspark in /usr/local/lib/python3.7/dist-packages (3.2.1)
Requirement already satisfied: py4j==0.10.9.3 in /usr/local/lib/python3.7/dist-packages (from pyspark) (0.10.9.3)
Requirement already satisfied: findspark in /usr/local/lib/python3.7/dist-packages (2.0.1)
```

```
#Import Packages
import pyspark.sql
from pyspark.sql import SQLContext
from pyspark.sql.types import *
from pyspark.sql import Row
from pyspark import SparkContext
from pyspark.sql.functions import col
from pyspark.sql.functions import *
from pyspark.ml.stat import *
from pyspark.mllib.stat import *
from pyspark.sql.types import StringType,BooleanType,DateType, IntegerType


#Attach Google Drive to import files and NOT worry about reuploading everytime
from google.colab import drive
drive.mount('/content/drive')
```

```
Drive already mounted at /content/drive; to attempt to forcibly remount, call drive.mount("/content/drive", force_remount=True).
```

```
#Import findspark
import findspark
findspark.init()


sc =SparkContext()


sqlContext = SQLContext(sc)

sc.version
```

```
/usr/local/lib/python3.7/dist-packages/pyspark/sql/context.py:79: FutureWarning: Deprecated in 3.0.0. Use SparkSession.builder.ge
```

```
#SparkContext version
sc.version

#Python version of SparkContext
sc.pythonVer
```

```
#master is the URL of the cluster or "local" string to run in local mode
sc.master
```

```
     'local[*]'
```

```
import pyspark
from pyspark.sql import SparkSession
spark = SparkSession.builder.master("local[1]").appName('Spark').getOrCreate()
```

Data

The *recent.csv* data set holds 1.45GB of information and *dec_2020.csv* data set holds 1.44GB of informations, both including the same 25 columns of information. List below:

- Post ID
- Price
- Year
- Manufacturer
- Model
- Condition
- Cylinders
- Obdometer
- Posting_Date
- Etc.

Initial data cleansing process was executed on excel due to the following reasons. Nevertheless, **minimum** cleansing process was carried out to show our understanding of the cleansing process and our PySpark skills. However, the *recent.csv* data set was precleaned on excel due to an unknown error.

- File downloaded from Kaggle wasn't completely comma separated.
    - Certain colums was bleeding into the next column when loaded.
- Certain rows held irrelevant/misleading information(Ads)

- Easier to sort the ID column information(misleading information)

Limitation of the Data

- Holds about 1 month worth of data each. Therefore, data inbetween the two time periods are missing.
- 1 Month worth of data is not enough to show trends for the whole year.

Pivot

- Due to such limitation within the two datasets, we have decided to "draw the larger picture". We decided to examine the Kaggle price trends of the two time periods separately. Then we will be tying our findings to the used car price trends provided by FRED.

```
#Import Data
dec_2020 = spark.read.csv("/content/drive/Shareddrives/Group19/dec_2020.csv", sep= ",", header=True, inferSchema=True)
dec_2020
```

```
DataFrame[_c0: int, id: bigint, url: string, region: string, region_url: string, price: bigint, year: int, manufacturer: string,
```

```
#Examine Data
dec_2020.show(3)
```

```
+----+----------+--------------------+-----------------+--------------------+-----+----+------------+--------------+---------+--
| _c0|        id|                 url|           region|          region_url|price|year|manufacturer|         model|condition|
+----+----------+--------------------+-----------------+--------------------+-----+----+------------+--------------+---------+--
|5594|7240652732|https://anchorage...|anchorage / mat-su|https://anchorage...|24988|2017|       mazda|          cx-5|     null|4
|5595|7240652585|https://anchorage...|anchorage / mat-su|https://anchorage...|31988|2016|        ford|         f-150|     null|8
|5596|7240652466|https://anchorage...|anchorage / mat-su|https://anchorage...|31950|2020|        jeep|grand cherokee|     null|6
+----+----------+--------------------+-----------------+--------------------+-----+----+------------+--------------+---------+--
only showing top 3 rows
```

```
#Import Data
recent =  spark.read.csv("/content/drive/Shareddrives/Group19/vehicles_recent_clean.csv", sep=",", header=True, inferSchema=True)
recent
```

```
DataFrame[id: bigint, region: string, price: bigint, year: int, manufacturer: string, model: string, condition: string, cylinders
```

```
#Examine Data
recent.show()
```

```
+----------+--------------------+-----+----+------------+-----+---------+---------+----+--------+------------+------------+----+-
|        id|              region|price|year|manufacturer|model|condition|cylinders|fuel|odometer|title_status|transmission| VIN|d
+----------+--------------------+-----+----+------------+-----+---------+---------+----+--------+------------+------------+----+-
|7207408119|          des moines|11700|null|        null| null|     null|     null|null|    null|        null|        null|null|
|7208549803|          bellingham|11999|null|        null| null|     null|     null|null|    null|        null|        null|null|
|7209027818|             el paso|    0|null|        null| null|     null|     null|null|    null|        null|        null|null|
|7209054699|           cleveland|17989|null|        null| null|     null|     null|null|    null|        null|        null|null|
|7209064557|     medford-ashland| 5000|null|        null| null|     null|     null|null|    null|        null|        null|null|
|7210384030|          greensboro| 4900|null|        null| null|     null|     null|null|    null|        null|        null|null|
|7212512589|           cleveland|18589|null|        null| null|     null|     null|null|    null|        null|        null|null|
|7212631321|skagit / island /...|21850|null|        null| null|     null|     null|null|    null|        null|        null|null|
|7213839225|          bellingham|26850|null|        null| null|     null|     null|null|    null|        null|        null|null|
|7213843538|skagit / island /...|24999|null|        null| null|     null|     null|null|    null|        null|        null|null|
|7215547569|fort collins / no...|15463|null|        null| null|     null|     null|null|    null|        null|        null|null|
|7215617048|          bellingham|21850|null|        null| null|     null|     null|null|    null|        null|        null|null|
|7216549243|fort collins / no...|13468|null|        null| null|     null|     null|null|    null|        null|        null|null|
|7216603380|           cleveland|15789|null|        null| null|     null|     null|null|    null|        null|        null|null|
|7216610120|           cleveland|16589|null|        null| null|     null|     null|null|    null|        null|        null|null|
|7216610223|           cleveland|13489|null|        null| null|     null|     null|null|    null|        null|        null|null|
|7216672204|          bellingham|24999|null|        null| null|     null|     null|null|    null|        null|        null|null|
|7217147606|             el paso|    0|null|        null| null|     null|     null|null|    null|        null|        null|null|
|7217189206|fort collins / no...|21928|null|        null| null|     null|     null|null|    null|        null|        null|null|
|7217788283|             el paso|    0|null|        null| null|     null|     null|null|    null|        null|        null|null|
+----------+--------------------+-----+----+------------+-----+---------+---------+----+--------+------------+------------+----+-
only showing top 20 rows
```

Data Cleansing

As we can see, there are large about of empty values in the data set. For more uniform analysis, we decided to drop the rows with NULL values.

```
#Drop Null Values then check
```

```
dec_2020=dec_2020.na.drop("any")
dec_2020.show(5)
```

```
+----+----------+--------------------+-----------------+--------------------+-----+----+------------+------------------+--------
| _c0|        id|                 url|           region|          region_url|price|year|manufacturer|             model|conditio
+----+----------+--------------------+-----------------+--------------------+-----+----+------------+------------------+--------
|5629|7240606876|https://anchorage...|anchorage / mat-su|https://anchorage...| 8950|2011|      nissan|          rouge sv|excelle
|5634|7240596660|https://anchorage...|anchorage / mat-su|https://anchorage...| 6950|2008|      nissan|            xterra|excelle
|5635|7240592595|https://anchorage...|anchorage / mat-su|https://anchorage...| 6950|2008|   chevrolet|         impala ls|excelle
|5646|7240580085|https://anchorage...|anchorage / mat-su|https://anchorage...| 7950|2009|      toyota|          camry le|excelle
|5685|7240271466|https://anchorage...|anchorage / mat-su|https://anchorage...|13950|2007|        jeep|wrangler unlimited|excelle
+----+----------+--------------------+-----------------+--------------------+-----+----+------------+------------------+--------
only showing top 5 rows
```

```
#Drop Null values then check the data
recent = recent.na.drop(how="any")
recent.show()
```

```
+----------+--------------------+-----+----+------------+------------+---------+-----------+------+--------+------------+--------
|        id|              region|price|year|manufacturer|       model|condition|  cylinders|  fuel|odometer|title_status|transmis
+----------+--------------------+-----+----+------------+------------+---------+-----------+------+--------+------------+--------
|7301592358|worcester / centr...| 8995|2013|        ford|explorer 4x4|     good|6 cylinders|   gas|  150000|       clean|   autom
|7301592395|western massachus...| 8995|2013|        ford|explorer 4x4|     good|6 cylinders|   gas|  150000|       clean|   autom
|7301592425|        rhode island| 8995|2013|        ford|explorer 4x4|     good|6 cylinders|   gas|  150000|       clean|   autom
|7301592468|            hartford| 8995|2013|        ford|explorer 4x4|     good|6 cylinders|   gas|  150000|       clean|   autom
|7301592549|             vermont| 8995|2013|        ford|explorer 4x4|     good|6 cylinders|   gas|  150000|       clean|   autom
|7301592579|              albany| 8995|2013|        ford|explorer 4x4|     good|6 cylinders|   gas|  150000|       clean|   autom
|7301593233|worcester / centr...| 8995|2013|        ford|explorer 4x4|     good|6 cylinders|   gas|  150000|       clean|   autom
|7301593262|western massachus...| 8995|2013|        ford|explorer 4x4|     good|6 cylinders|   gas|  150000|       clean|   autom
|7301593289|        rhode island| 8995|2013|        ford|explorer 4x4|     good|6 cylinders|   gas|  150000|       clean|   autom
|7301593334|            hartford| 8995|2013|        ford|explorer 4x4|     good|6 cylinders|   gas|  150000|       clean|   autom
|7301593436|             vermont| 8995|2013|        ford|explorer 4x4|     good|6 cylinders|   gas|  150000|       clean|   autom
|7301593465|              albany| 8995|2013|        ford|explorer 4x4|     good|6 cylinders|   gas|  150000|       clean|   autom
|7301594506|worcester / centr...| 6995|2006|        ford|    e350 van|     good|8 cylinders|   gas|  108000|       clean|   autom
|7301594568|western massachus...| 6995|2006|        ford|    e350 van|     good|8 cylinders|   gas|  108000|       clean|   autom
|7301594592|        rhode island| 6995|2006|        ford|    e350 van|     good|8 cylinders|   gas|  108000|       clean|   autom
|7301594622|            hartford| 6995|2006|        ford|    e350 van|     good|8 cylinders|   gas|  108000|       clean|   autom
|7301594739|             vermont| 6995|2006|        ford|    e350 van|     good|8 cylinders|   gas|  108000|       clean|   autom
|7301594765|              albany| 6995|2006|        ford|    e350 van|     good|8 cylinders|   gas|  108000|       clean|   autom
|7301603102|        cedar rapids|17995|2013|    cadillac|     ats awd|excellent|4 cylinders|   gas|   33444|       clean|   autom
```

```
|7301603652|             holland|17950|2014|        ram|       1500|excellent|6 cylinders|diesel|  171000|       clean|   autom
+----------+--------------------+-----+----+-----------+-----------+---------+-----------+------+--------+------------+-------
only showing top 20 rows
```

We have columns that we do not need for our analysis and certain columns are "bleeding" into the next column, which is confusing. We will be dropping the columns that do not provide any information. As noted above, the *recent* dataset was precleaned on excel.

```
#Drop unwanted columns
dec_2020 = dec_2020.drop("_c0","url", "region_url", "image_url", "lat", "long")
#Check
dec_2020.show(5)
```

```
+----------+------------------+-----+----+------------+------------------+---------+-----------+----+--------+------------+------
|        id|            region|price|year|manufacturer|             model|condition|  cylinders|fuel|odometer|title_status|transm
+----------+------------------+-----+----+------------+------------------+---------+-----------+----+--------+------------+------
|7240606876|anchorage / mat-su| 8950|2011|      nissan|          rouge sv|excellent|6 cylinders| gas|  133180|       clean|   aut
|7240596660|anchorage / mat-su| 6950|2008|      nissan|            xterra|excellent|6 cylinders| gas|  171126|       clean|   aut
|7240592595|anchorage / mat-su| 6950|2008|   chevrolet|         impala ls|excellent|6 cylinders| gas|   84471|       clean|   aut
|7240580085|anchorage / mat-su| 7950|2009|      toyota|          camry le|excellent|4 cylinders| gas|  127338|       clean|   aut
|7240271466|anchorage / mat-su|13950|2007|        jeep|wrangler unlimited|excellent|8 cylinders| gas|  145664|       clean|
+----------+------------------+-----+----+------------+------------------+---------+-----------+----+--------+------------+------
only showing top 5 rows
```

```
#Drop Rows with ads again.
dec_2020=dec_2020.filter((dec_2020.model != 'Wanted!'))
```

```
#Check the Schema/Data Type
recent.printSchema()
dec_2020.printSchema()
```

```
root
 |-- id: long (nullable = true)
```

```
 |-- region: string (nullable = true)
 |-- price: long (nullable = true)
 |-- year: integer (nullable = true)
 |-- manufacturer: string (nullable = true)
 |-- model: string (nullable = true)
 |-- condition: string (nullable = true)
 |-- cylinders: string (nullable = true)
 |-- fuel: string (nullable = true)
 |-- odometer: integer (nullable = true)
 |-- title_status: string (nullable = true)
 |-- transmission: string (nullable = true)
 |-- VIN: string (nullable = true)
 |-- drive: string (nullable = true)
 |-- size: string (nullable = true)
 |-- type: string (nullable = true)
 |-- paint_color: string (nullable = true)
 |-- state: string (nullable = true)
 |-- posting_date: string (nullable = true)

root
 |-- id: long (nullable = true)
 |-- region: string (nullable = true)
 |-- price: long (nullable = true)
 |-- year: integer (nullable = true)
 |-- manufacturer: string (nullable = true)
 |-- model: string (nullable = true)
 |-- condition: string (nullable = true)
 |-- cylinders: string (nullable = true)
 |-- fuel: string (nullable = true)
 |-- odometer: integer (nullable = true)
 |-- title_status: string (nullable = true)
 |-- transmission: string (nullable = true)
 |-- VIN: string (nullable = true)
 |-- drive: string (nullable = true)
 |-- size: string (nullable = true)
 |-- type: string (nullable = true)
 |-- paint_color: string (nullable = true)
 |-- state: string (nullable = true)
 |-- posting_date: string (nullable = true)
```

We can see that all of our data is in *string*. In this case, we cannot run any numerical analysis(i.e., average, median, etc.). Now we are going to change the data type of each column.

```
#Check the data type of each column
dec_2020 = dec_2020.withColumn("price",col("price").cast(IntegerType())).withColumn("odometer",col("odometer").cast(IntegerType())).wi
#Check Schema
dec_2020.printSchema()
```

```
root
 |-- id: long (nullable = true)
 |-- region: string (nullable = true)
 |-- price: integer (nullable = true)
 |-- year: integer (nullable = true)
 |-- manufacturer: string (nullable = true)
 |-- model: string (nullable = true)
 |-- condition: string (nullable = true)
 |-- cylinders: string (nullable = true)
 |-- fuel: string (nullable = true)
 |-- odometer: integer (nullable = true)
 |-- title_status: string (nullable = true)
 |-- transmission: string (nullable = true)
 |-- VIN: string (nullable = true)
 |-- drive: string (nullable = true)
 |-- size: string (nullable = true)
 |-- type: string (nullable = true)
 |-- paint_color: string (nullable = true)
 |-- state: string (nullable = true)
 |-- posting_date: date (nullable = true)
```

```
#Change the data type of each column
recent = recent.withColumn("price",col("price").cast(IntegerType())).withColumn("odometer",col("odometer").cast(IntegerType())).withCo
#Check Schema
recent.printSchema()
```

```
root
 |-- id: long (nullable = true)
 |-- region: string (nullable = true)
 |-- price: integer (nullable = true)
 |-- year: integer (nullable = true)
 |-- manufacturer: string (nullable = true)
 |-- model: string (nullable = true)
 |-- condition: string (nullable = true)
 |-- cylinders: string (nullable = true)
```

```
 |-- fuel: string (nullable = true)
 |-- odometer: integer (nullable = true)
 |-- title_status: string (nullable = true)
 |-- transmission: string (nullable = true)
 |-- VIN: string (nullable = true)
 |-- drive: string (nullable = true)
 |-- size: string (nullable = true)
 |-- type: string (nullable = true)
 |-- paint_color: string (nullable = true)
 |-- state: string (nullable = true)
 |-- posting_date: date (nullable = true)
```

```
#Make sure manufacturer name is uniform for both data
dec_2020.select("manufacturer").distinct().show(50)
```

```
+---------------+
|   manufacturer|
+---------------+
|         jaguar|
|          buick|
|     land rover|
|     mitsubishi|
|        pontiac|
|          lexus|
|         toyota|
|       chrysler|
|           tesla|
|        lincoln|
|            audi|
|         datsun|
|             bmw|
|            jeep|
|           dodge|
|           rover|
|         hyundai|
|             ford|
|     alfa-romeo|
|       cadillac|
|              ram|
|           mazda|
|        ferrari|
|             kia|
```

```
|  mercedes-benz|
|         porsche|
|   aston-martin|
|          saturn|
|       chevrolet|
|           honda|
|            mini|
|            fiat|
|     volkswagen|
|         mercury|
|harley-davidson|
|           acura|
|             gmc|
|        infiniti|
|          nissan|
|          subaru|
|           volvo|
+---------------+
```

```
recent.select("manufacturer").distinct().show(50)
```

```
+---------------+
|   manufacturer|
+---------------+
|         jaguar|
|          buick|
|     mitsubishi|
|        pontiac|
|          lexus|
|         toyota|
|       chrysler|
|          tesla|
|        lincoln|
|           audi|
|         datsun|
|            bmw|
|           jeep|
|          dodge|
|          rover|
|        hyundai|
|           ford|
|     alfa-romeo|
```

```
|       cadillac|
|            ram|
|          mazda|
|        ferrari|
|            kia|
|  mercedes-benz|
|        porsche|
|   aston-martin|
|         saturn|
|      chevrolet|
|          honda|
|           mini|
|           fiat|
|     volkswagen|
|        mercury|
|harley-davidson|
|          acura|
|            gmc|
|       infiniti|
|         nissan|
|         subaru|
|          volvo|
|     land rover|
+---------------+
```

Both datasets have both 'land rover' and 'rover', make it uniform.

```
#Change 'land rover' to 'rover'
recent = recent.withColumn('manufacturer', regexp_replace('manufacturer', 'land rover', 'rover'))
dec_2020 = dec_2020.withColumn('manufacturer', regexp_replace('manufacturer', 'land rover', 'rover'))
```

```
#This should return a empty table if "land rover" was replaced to "rover"
recent.createOrReplaceTempView("table1")
check = spark.sql("SELECT manufacturer FROM table1 WHERE manufacturer=='land rover'")
check.head(10)
```

```
[]
```

```
#Check if they changed
```

```
recent.filter(recent["manufacturer"]=="rover").show(5) #This should return 5 rows of information
```

```
+----------+--------------------+-----+----+------------+--------------+---------+-----------+----+--------+------------+------
|        id|              region|price|year|manufacturer|         model|condition|  cylinders|fuel|odometer|title_status|transmi
+----------+--------------------+-----+----+------------+--------------+---------+-----------+----+--------+------------+------
|7301671555|          des moines|    0|2016|       rover|discovery sport|excellent|4 cylinders| gas|  127072|       clean|   auto
|7302511753|western massachus...| 8995|2008|       rover|           lr3|excellent|8 cylinders| gas|  110047|       clean|   auto
|7302540721|         chattanooga|19900|2013|       rover|         sport|excellent|8 cylinders| gas|   93000|       clean|   auto
|7302580404|       daytona beach|19990|2011|       rover|         sport|excellent|8 cylinders| gas|   70117|       clean|   auto
|7302879408|              albany|23900|2016|       rover|discovery sport|excellent|4 cylinders| gas|   60412|       clean|   auto
+----------+--------------------+-----+----+------------+--------------+---------+-----------+----+--------+------------+------
only showing top 5 rows
```

```
#This should return a empty table if "land rover" was replaced to "rover"
dec_2020.createOrReplaceTempView("table2")
check = spark.sql("SELECT manufacturer FROM table2 WHERE manufacturer=='land rover'")
check.head(10)
```

```
[]
```

```
#Check if they changed
dec_2020.filter(dec_2020["manufacturer"]=="rover").show(5) #This should return 5 rows of information
```

```
+----------+--------------------+-----+----+------------+-------------+---------+-----------+----+--------+------------+-------
|        id|              region|price|year|manufacturer|        model|condition|  cylinders|fuel|odometer|title_status|transmis
+----------+--------------------+-----+----+------------+-------------+---------+-----------+----+--------+------------+-------
|7237931612|  anchorage / mat-su|54995|2016|       rover|        sport|excellent|8 cylinders| gas|   53739|       clean|   autom
|7233878213|  anchorage / mat-su|54995|2018|       rover|    discovery|excellent|6 cylinders| gas|   10304|       clean|   autom
|7230571058|  anchorage / mat-su|35999|2019|       rover|discovry sport|excellent|4 cylinders| gas|   14561|       clean|   autom
|7233551048|          birmingham| 7900|2008|       rover|          lr2|     good|6 cylinders| gas|  150260|       clean|   autom
|7233711871|huntsville / decatur| 7900|2008|       rover|          lr2|     good|6 cylinders| gas|  150260|       clean|   autom
+----------+--------------------+-----+----+------------+-------------+---------+-----------+----+--------+------------+-------
only showing top 5 rows
```

Ready for Analysis!

Analysis

First we wanted to find the average prices and odometer for each columns in the data set to see if there are any patterns/correlation between the two or anything interesting to note.

```
#Dec 2020 Data Set AVG Price Group By Condition of the Car
avg_dc=dec_2020.groupBy('condition').mean('price')
avg_dc.show()
```

```
+---------+------------------+
|condition|        avg(price)|
+---------+------------------+
|      new|20272.333333333332|
|excellent|13734.339634253562|
|  salvage|           3418.25|
| like new| 18459.75160187954|
|     good|14931.495519203414|
|     fair|3652.9226973684213|
+---------+------------------+
```

```
#Recent Data Set AVG Price Group By Condition of the Car
avg_rc=recent.groupBy('condition').mean('price')
avg_rc.show()
```

```
+---------+------------------+
|condition|        avg(price)|
+---------+------------------+
|      new|25033.325358851675|
|excellent|15225.610100204398|
|  salvage| 4723.611111111111|
| like new|21411.805590851334|
|     good| 15290.82154483575|
|     fair| 4367.993902439024|
+---------+------------------+
```

```
#Get the Price difference
new= 25033.32 - 20272.333
excellent = 15225.61 - 13734.339
salvage = 4723.611 - 3418.25
like_new = 21411.80 - 18459.751
good =  15290.82 -14931.49
fair= 4367.99 - 3652.92
print("new:", new , "excellent:", excellent, "salvage:", salvage, "like new:", like_new, "good:", good, "fair:", fair)
```

```
new: 4760.987000000001 excellent: 1491.2710000000006 salvage: 1305.3609999999999 like new: 2952.048999999999 good: 359.3299999999
```

```
#By Posting Date
recent.groupBy('posting_date').mean('price').orderBy("posting_date").show(5)
```

```
+------------+------------------+
|posting_date|        avg(price)|
+------------+------------------+
|  2021-04-04|12647.244755244756|
|  2021-04-05|17672.275229357798|
|  2021-04-06|14531.807926829268|
|  2021-04-07|16631.076051779935|
|  2021-04-08|16426.103641456582|
+------------+------------------+
only showing top 5 rows
```

```
#By Posting Date
dec_2020.groupBy('posting_date').mean('price').orderBy("posting_date").show(5)
```

```
+------------+------------------+
|posting_date|        avg(price)|
+------------+------------------+
|  2020-11-03|16284.463562753037|
|  2020-11-04|15487.001605136436|
|  2020-11-05|           11987.0|
|  2020-11-06|12763.579809004093|
|  2020-11-07|14070.902173913044|
+------------+------------------+
```

```
only showing top 5 rows
```

```
#Average Odometer Group By Manufacturer
recent.groupBy("manufacturer").avg("odometer").orderBy("avg(odometer)").show()
```

```
+------------+------------------+
| manufacturer|     avg(odometer)|
+------------+------------------+
| aston-martin|17006.333333333332|
|      ferrari|           19116.4|
|   alfa-romeo|36305.666666666664|
|        tesla| 48765.78947368421|
|         fiat| 69444.24489795919|
|      porsche| 76482.48550724638|
|        rover| 78467.21428571429|
|mercedes-benz| 88440.90632318502|
|          bmw|  90452.1410118407|
|       jaguar| 93156.06896551725|
|     cadillac|  95189.7225433526|
|         mini| 95326.71428571429|
|         audi| 95848.01777777777|
|       nissan| 96061.19835680751|
|          kia| 96286.76850393701|
|   mitsubishi| 96837.57014925373|
|   volkswagen| 97482.15595075239|
|        dodge| 98525.26789366054|
|      hyundai| 98916.40891218872|
|     infiniti| 99866.19291338582|
+------------+------------------+
only showing top 20 rows
```

```
dec_2020.groupBy("manufacturer").avg("odometer").orderBy("avg(odometer)").show()
```

```
+------------+------------------+
| manufacturer|     avg(odometer)|
+------------+------------------+
|      ferrari| 7024.666666666667|
| aston-martin|           13485.0|
|        tesla|29739.647058823528|
|   alfa-romeo|          60413.25|
|         fiat|  66235.3294117647|
```

```
|        rover| 78874.97692307692|
|   mitsubishi| 82132.13356164383|
|         mini| 85106.27325581395|
|         audi|  86350.1351888668|
|   volkswagen|  89105.8901734104|
|          kia|  89799.0935064935|
|mercedes-benz| 92399.90145228215|
|       jaguar| 93128.27586206897|
|       nissan| 95056.06196581197|
|         jeep| 95278.66482910694|
|     cadillac|  96885.9331210191|
|          bmw| 97343.16365568544|
|      hyundai|  97851.8673870334|
|        dodge|  99128.1124260355|
|      porsche| 99408.54651162791|
+-------------+------------------+
only showing top 20 rows
```

Odometer for **Ferrari** and **Aston Martin** is extremely low compared to other cars (needs further analysis)

```
#Check instances of Ferrai within the dataset.
dec_2020.groupBy("manufacturer").avg("price").orderBy("avg(price)",ascending=False).show()
```

```
+---------------+------------------+
|   manufacturer|        avg(price)|
+---------------+------------------+
|        ferrari|194966.66666666666|
|          tesla| 45471.94117647059|
|   aston-martin|           42897.5|
|        porsche|28314.593023255813|
|          rover| 22898.56923076923|
|            ram| 22194.50550086856|
|     alfa-romeo|           20349.0|
|         toyota|18347.597323247017|
|           audi|18063.634194831015|
|            gmc| 17877.58894230769|
|           ford|17160.142726231385|
|  mercedes-benz|16509.593360995852|
|      chevrolet| 15509.28659611993|
|         datsun|           15249.5|
|harley-davidson|15136.285714285714|
```

```
|           jeep|14970.409592061742|
|       cadillac|14359.579617834395|
|            bmw|13578.988310308183|
|         jaguar|13424.770114942528|
|     mitsubishi| 13264.09589041096|
+---------------+------------------+
only showing top 20 rows
```

```
recent.groupBy("manufacturer").avg("price").orderBy("avg(price)",ascending=False).show()
```

```
+---------------+------------------+
|   manufacturer|        avg(price)|
+---------------+------------------+
|        ferrari|           86246.5|
|   aston-martin|           57280.0|
|          tesla| 41423.57894736842|
|        porsche|30935.036231884056|
|            ram|24584.624381188118|
|     alfa-romeo|23893.133333333335|
|          rover|22401.553571428572|
|            gmc| 21485.66018158236|
|           ford|20526.193874797624|
|harley-davidson|19023.428571428572|
|     mitsubishi| 17773.36119402985|
|      chevrolet| 17583.79213483146|
|         datsun|15499.666666666666|
|           jeep|15237.987545787546|
|  mercedes-benz|15159.329039812646|
|           audi|14323.104444444445|
|       cadillac|14045.394990366089|
|       infiniti|13398.625984251968|
|          lexus|         13377.485|
|         toyota|13344.286209286209|
+---------------+------------------+
only showing top 20 rows
```

As we can see from above, we have some interesting patterns within our data. We see that Ferrari and Aston Martin has the **lowest** odometer values, but **highest** average price values. We want to visualize and see the average prices, depicted below.

```
recent_avgprice= recent.groupBy("manufacturer").avg("price")
recent_avgprice = recent_avgprice.toPandas()
recent_avgprice.head(3)
```

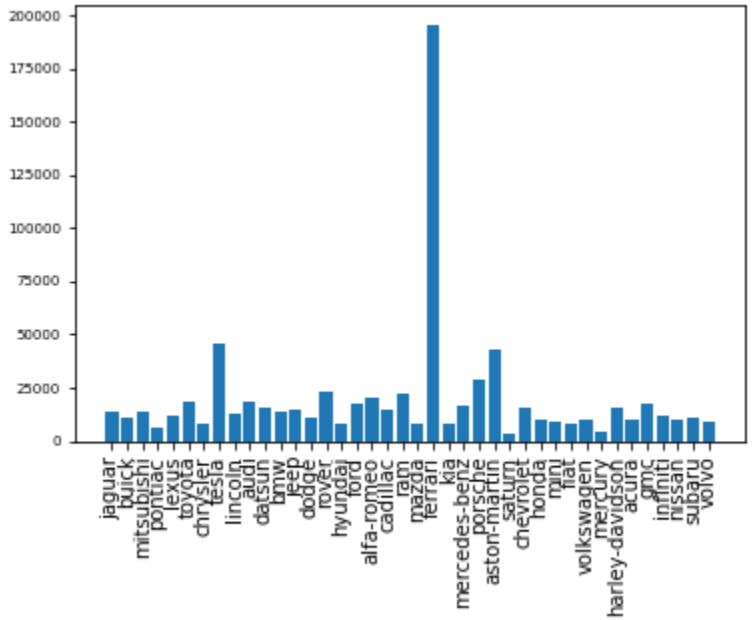
	manufacturer	avg(price)
0	jaguar	11667.609195
1	buick	10557.770619
2	mitsubishi	17773.361194

```
import matplotlib.pyplot as plt
from matplotlib.pyplot import figure

x = recent_avgprice["manufacturer"]
y = recent_avgprice["avg(price)"]

plt.bar(x, y)
plt.grid(visible=None)
plt.xticks(rotation=90)
plt.tick_params(axis='y', labelsize=7)
plt.show()
```

```
dec_avg= dec_2020.groupBy("manufacturer").avg("price")
dec_avg = dec_avg.toPandas()
dec_avg.head(3)
```

	manufacturer	avg(price)
0	jaguar	13424.770115
1	buick	10920.477860
2	mitsubishi	13264.095890

```
x = dec_avg["manufacturer"]
y = dec_avg["avg(price)"]

plt.bar(x,y)
plt.grid(visible=None)
plt.xticks(rotation=90)
plt.tick_params(axis='y', labelsize=7)
plt.show()
```



As we can see from above, Ferrari and Aston Martin has incredibly high used car values, enough to possible skew the average prices of the data. Therefore, we decided to use "median" as our indicator for trends.

```
#Average Price By State
st_avg= recent.groupBy("state").avg("price")
st_avg = st_avg.toPandas()
st_avg.head(3)
```

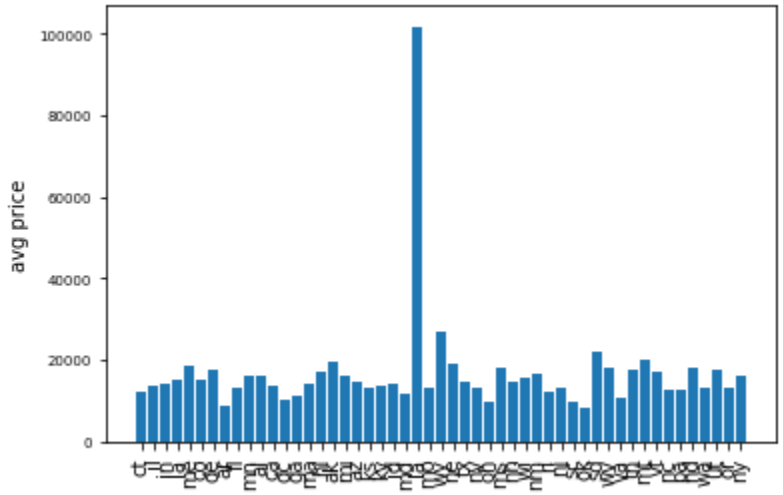
	state	avg(price)
0	ct	10874.137466
1	wy	22470.202247
2	ne	18847.037313

```
x = st_avg["state"]
y = st_avg["avg(price)"]
fig, ax = plt.subplots()
plt.bar(x, y)
plt.grid(visible=None)
plt.xticks(rotation=90)
ax.set_xticks(st_avg["state"])
plt.tick_params(axis='y', labelsize=7)
plt.ylabel("avg price")
plt.show()
```

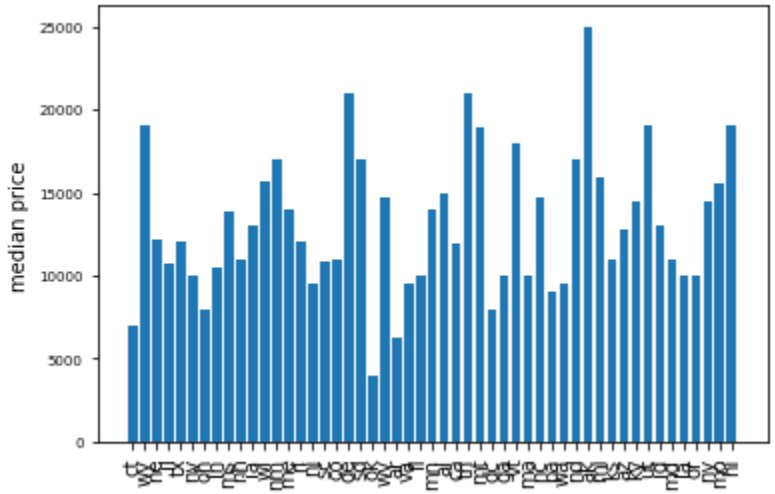

```
dec_avg= dec_2020.groupBy("state").avg("price")
dec_avg = dec_avg.toPandas()
dec_avg.head(3)
```

	state	avg(price)
0	ct	12177.321649
1	il	13499.401515
2	in	14168.734072

```
x = dec_avg["state"]
y = dec_avg["avg(price)"]
fig, ax = plt.subplots()
plt.bar(x, y)
plt.grid(visible=None)
plt.xticks(rotation=90)
ax.set_xticks(dec_avg["state"])
plt.tick_params(axis='y', labelsize=7)
plt.ylabel("avg price")
plt.show()
```

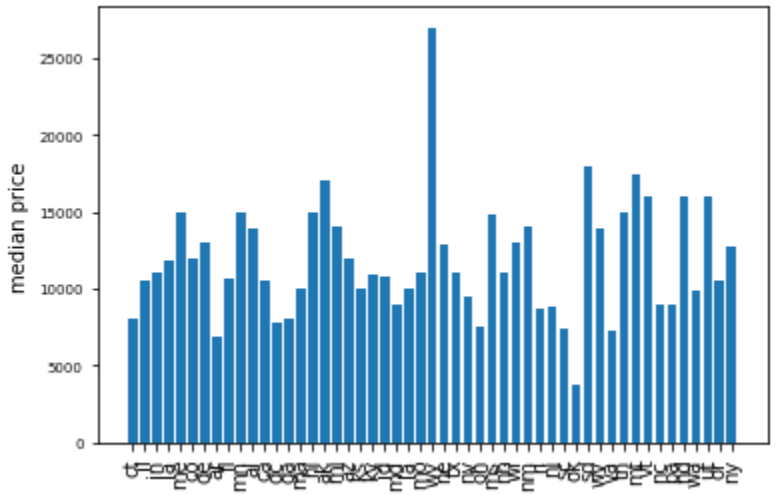


```
# Median Price across each state RECENT Data
st_median_recent=recent.groupBy("state").agg(percentile_approx("price", 0.5).alias("median price"))
st_median_recent = st_median_recent.toPandas()


x = st_median_recent["state"]
y = st_median_recent["median price"]
fig, ax = plt.subplots()
plt.bar(x, y)
plt.grid(visible=None)
plt.xticks(rotation=90)
ax.set_xticks(st_median_recent["state"])
plt.tick_params(axis='y', labelsize=7)
plt.ylabel("median price")
plt.show()
```




```
st_median_dec=dec_2020.groupBy("state").agg(percentile_approx("price", 0.5).alias("median price"))
st_median_dec = st_median_dec.toPandas()


x = st_median_dec["state"]
y = st_median_dec["median price"]
fig, ax = plt.subplots()
plt.bar(x, y)
plt.grid(visible=None)
plt.xticks(rotation=90)
```

```
ax.set_xticks(st_median_dec["state"])
plt.tick_params(axis='y', labelsize=7)
plt.ylabel("median price")
plt.show()
```



Just like how we have imagined, the average prices of individual states are extremely skewed for the **dec_2020** data set. On the other hand, median showed more of a uniformed distribution of the data.

Median Analysis

Median Odometer Mileage Across Manufacturers

```
# Median odometer mileage across each manufacturer
recent.groupBy("manufacturer").agg(percentile_approx("odometer", 0.5).alias("median mileage")).show(5)
```

```
+------------+--------------+
|manufacturer|median mileage|
+------------+--------------+
|      jaguar|         90818|
|       buick|         98859|
|  mitsubishi|         96626|
|     pontiac|        116500|
```

```
|       lexus|        114000|
+------------+--------------+
only showing top 5 rows
```

```
# Median odometer mileage across each manufacturer
dec_2020.groupBy("manufacturer").agg(percentile_approx("odometer", 0.5).alias("median mileage")).show(5)
```

```
+------------+--------------+
|manufacturer|median mileage|
+------------+--------------+
|      jaguar|         82659|
|       buick|         96770|
|  mitsubishi|         73000|
|     pontiac|        122280|
|       lexus|        113000|
+------------+--------------+
only showing top 5 rows
```

Median Price Grouped By Manufacturers and Odometer

```
recent.groupBy("manufacturer","odometer").agg(percentile_approx("price", 0.5).alias("median")).show()
```

```
+------------+--------+------+
|manufacturer|odometer|median|
+------------+--------+------+
|       acura|       1|  2995|
|       acura|      10| 39995|
|       acura|     115|     0|
|       acura|     140|     0|
|       acura|     170|     0|
|       acura|     180|     0|
|       acura|    1555| 42800|
|       acura|   11009| 34998|
|       acura|   11500| 36500|
|       acura|   17501| 30995|
|       acura|   18075|     1|
|       acura|   18280| 46567|
|       acura|   19969| 33990|
|       acura|   21347|     1|
|       acura|   21350| 23495|
```

```
|       acura|   23800| 17300|
|       acura|   35000| 23000|
|       acura|   35161| 22795|
|       acura|   35400| 18750|
|       acura|   36016| 27500|
+------------+--------+------+
only showing top 20 rows
```

```
dec_2020.groupBy("manufacturer","odometer").agg(percentile_approx("price", 0.5).alias("median")).show()
```

```
+------------+--------+------+
|manufacturer|odometer|median|
+------------+--------+------+
|       acura|       0|  6900|
|       acura|      15|     0|
|       acura|      30|     0|
|       acura|      50|     0|
|       acura|      72|     0|
|       acura|      90|     0|
|       acura|     101|     0|
|       acura|     115|     0|
|       acura|     200|  1800|
|       acura|    6730| 41950|
|       acura|    9290| 36995|
|       acura|   13173|     1|
|       acura|   18500|  3500|
|       acura|   19966| 25900|
|       acura|   22267| 33990|
|       acura|   22616| 21998|
|       acura|   23170| 28000|
|       acura|   26163| 21999|
|       acura|   28700| 12500|
|       acura|   29885| 39800|
+------------+--------+------+
only showing top 20 rows
```

Median Price Across Each Condition

```
# Median Price across each condition
```

```
recent.groupBy("condition").agg(percentile_approx("price", 0.5).alias("median price")).show()
dec_2020.groupBy("condition").agg(percentile_approx("price", 0.5).alias("median price")).show()
```

```
+---------+------------+
|condition|median price|
+---------+------------+
|      new|       15500|
|excellent|       11995|
|  salvage|        2700|
| like new|       17999|
|     good|       11500|
|     fair|        3250|
+---------+------------+
```

```
+---------+------------+
|condition|median price|
+---------+------------+
|      new|       12977|
|excellent|       10995|
|  salvage|        2500|
| like new|       15500|
|     good|        9974|
|     fair|        2500|
+---------+------------+
```

Median Price Across Each Vehicle Type

```
# Median Price across each vehicle type
recent.groupBy("type").agg(percentile_approx("price", 0.5).alias("median price")).show()
dec_2020.groupBy("type").agg(percentile_approx("price", 0.5).alias("median price")).show()
```

```
+-----------+------------+
|       type|median price|
+-----------+------------+
|        van|       16661|
|   mini-van|        7950|
|    offroad|       23388|
|      wagon|        6900|
|      coupe|       10950|
|        bus|       11997|
|        SUV|       11995|
```

```
|      other|       11995|
|convertible|       12950|
|      sedan|        7995|
|  hatchback|        7500|
|      truck|       24900|
|     pickup|       18999|
+-----------+------------+

+-----------+------------+
|       type|median price|
+-----------+------------+
|        van|       13995|
|   mini-van|        7797|
|    offroad|       16995|
|      wagon|        7000|
|      coupe|       10500|
|        bus|        9998|
|        SUV|       10999|
|      other|        7050|
|convertible|       11900|
|      sedan|        7600|
|  hatchback|        6950|
|      truck|       19995|
|     pickup|       17799|
+-----------+------------+
```

Median Price across each drive configuration

```
# Median Price across each drive configuration
recent.groupBy("drive").agg(percentile_approx("price", 0.5).alias("median price")).show(50)
dec_2020.groupBy("drive").agg(percentile_approx("price", 0.5).alias("median price")).show(50)
```

```
+-----+------------+
|drive|median price|
+-----+------------+
|  fwd|        7995|
|  rwd|       14999|
|  4wd|       16324|
+-----+------------+

+-----+------------+
```

```
|drive|median price|
+-----+------------+
|  fwd|        7495|
|  rwd|       12900|
|  4wd|       15000|
+-----+------------+
```

```
# Median Price across each transmission configuration
recent.groupBy("transmission").agg(percentile_approx("price", 0.5).alias("median price")).show(50)
dec_2020.groupBy("transmission").agg(percentile_approx("price", 0.5).alias("median price")).show(50)
```

```
+------------+------------+
|transmission|median price|
+------------+------------+
|   automatic|       12500|
|       other|           1|
|      manual|        9995|
+------------+------------+

+------------+------------+
|transmission|median price|
+------------+------------+
|   automatic|       10995|
|       other|           1|
|      manual|        9300|
+------------+------------+
```

As we see from our median analysis, we see an overall increase from Dec.2020 to May 2021 in prices across the used car market. This closely mimics the trend that we know from general media sources.

We also conducted some miscellaneous analysis below

```
# Correlation between feature columns (model 'year') and 'price' of listing
recent.stat.corr('year', 'price')
```

```
0.25405858410835164
```

```
dec_2020.stat.corr('year', 'price')
```

```
0.03682365887583517
```

```
pdate_df=dec_2020.groupBy('posting_date').avg('price')
pddf=pdate_df.toPandas()
pddf.head(3)
```

	posting_date	avg(price)
0	2020-11-29	12534.479084
1	2020-11-06	12763.579809
2	2020-11-27	13989.789968

```
recent_df=recent.groupBy('posting_date').avg('price')
recent_df=recent_df.toPandas()
recent_df.head(3)
```

	posting_date	avg(price)
0	2021-04-24	15139.975723
1	2021-04-25	14377.895706
2	2021-04-21	14599.761950

Extraction

We decided to extract various analysis we have conducted to compare the price difference from 2020 to 2021. We have visualized the extracted files in Tableau.

*Note: Not all analysis were extracted, and not all visuals we have created were made using the extracted files. We have extracted a couple files to show our PySpark skills. (It is much easier to handle simple math and visualize on Tableau)

```
with open('/content/drive/Shareddrives/Group19/Output_data_BD/dec_date.csv', 'w', encoding = 'utf-8-sig') as f:
```

```
  pddf.to_csv(f)


with open('/content/drive/Shareddrives/Group19/Output_data_BD/recent_date.csv', 'w', encoding = 'utf-8-sig') as f:
  recent_df.to_csv(f)


with open('/content/drive/Shareddrives/Group19/Output_data_BD/recent_manu.csv', 'w', encoding = 'utf-8-sig') as f:
  recent_avgprice.to_csv(f)


recent_date=recent.groupBy('posting_date').avg('price')
recent_date=recent_date.toPandas()
recent_date.head(3)
```

	posting_date	avg(price)
0	2021-04-24	15139.975723
1	2021-04-25	14377.895706
2	2021-04-21	14599.761950

```
with open('/content/drive/Shareddrives/Group19/Output_data_BD/recent_date.xlsx', 'w') as f:
  recent_date.to_csv(f)


dec_avg=dec_2020.groupBy('posting_date').avg('price')
dec_avg=dec_avg.toPandas()
dec_avg.head(3)
```

	posting_date	avg(price)
0	2020-11-29	12534.479084
1	2020-11-06	12763.579809
2	2020-11-27	13989.789968

```
with open('/content/drive/Shareddrives/Group19/Output_data_BD/dec_avg.csv', 'w', encoding = 'utf-8-sig') as f:
  dec_avg.to_csv(f)
```

Conclusion

There were various limitation to the data set that we have used this for analysis.

1. Missing data points between the two time lines
2. Great number of rows with advertisements providing zero/misleading information
3. Most data were inputed by Craiglist user, therefore, lacks data uniformity
4. Price data was skewed by high end *super* cars.

Nevertheless, overall, we can see that our findings somewhat mimics the used car price trends with the same time period (Nov.2020~May.2021). FRED.

- *If you are having trouble running the juypter notebook, please refer to our PDF file that has output of our code or I am more than happy to give you access to our google colab worksheet. There also could be random errors while loading the data set. CSV used for analysis/visualization on Tableau is attached to the Tableau file as a twbx file*

✓ 0s completed at 11:36 PM