Used Cars Analysis using Spark SQL in Zeppelin notebook

LONG T. NGUYEN

www.longnguyendata.com

Dataset

Found on Kaggle: https://www.kaggle.com/datasets/mirosval/personal-cars-classifieds

The used car sales data was collected from various websites in the Czech Republic and Germany for over a year. However, some of the websites were not structured; therefore, the data obtained is not perfect, and there are some missing values and incorrect entries (e.g., phone numbers scraped as mileage, etc.). This means that we need to clean and validate the data before we can use it effectively.

There are roughly 3.5 million rows and the following columns:

- maker - normalized all lowercase
- model - normalized all lowercase
- mileage - in KM
- manufacture_year
- engine_displacement - in ccm
- engine_power - in kW
- body_type - almost never present, but I scraped only personal cars, no motorcycles or utility vehicles
- color_slug - also almost never present
- stk_year - the year of the last emission control
- transmission - automatic or manual
- door_count
- seat_count
- fuel_type - gasoline, diesel, cng, lpg, electric
- date_created - when the ad was scraped
- date_last_seen - when the ad was last seen. Our policy was to remove all ads older than 60 days.
- price_eur - list price converted to EUR

Tasks:

1. Write a Spark SQL query to create a table called **used_cars** from data and let it infer the schema.

2. Print the inferred schema.

3. Write Spark SQL queries to see how many missing values you have in each attribute. In the Text section of your notebook, write how many missing values in each column we have. Especially, mention those columns with more than 50% missing values.

4. Write a Spark SQL query to create a new table called **clean_used_cars** from **used_cars** with the following conditions:

- Drop the columns with more than 50% missing values
- The manufacture year between 2000 and 2017 including 2000 and 2017
- Both maker and model exist in the row

- The price range is from 3000 to 2,000,000 (3000 ≤ price ≤ 2,000,000)

5. Write a Spark SQL query to find how many records remained in **clean_used_cars**.

6. Write a Spark SQL query to find the make and model for the cars with the top 10 highest average price.

7. Write a Spark SQL query to find the make and model for the cars with the top 10 lowest average price.

8/9/10. Write a Spark SQL query to recommend top five make and model for 3 segment customers:

- Economic (average price between €3,000 and €20,000)
- Intermediate (between €20,000 and €300,000)
- Luxury (between €300,000 and €2,000,000)

DOWNLOADING THE DATASET

Use the **Car Dataset** to write your code using Apache Spark SQL in Zeppelin notebook:

First off, download and unzip the dataset. The screenshots are below.

```
%sh
## Create a new directory called assignment
hadoop fs -ls /user/assignment
```

FINISHED

```
Found 1 items
-rw-r--r--   2 zeppelin hadoop  419466302 2022-10-25 03:35 /user/assignment/cars.csv
```

Took 2 sec. Last updated by anonymous at October 28 2022, 2:17:22 PM.

```
%sh
## Downloading the file to local
wget -q https://bit.ly/ClassifiedCars -O cars.zip
ls -lah cars.csv
```

FINISHED

```
-rw-r--r-- 1 zeppelin zeppelin 401M Apr 11  2020 cars.csv
```

Took 2 sec. Last updated by anonymous at October 28 2022, 2:17:24 PM.

```
%sh
## Unzip the file and put it in the "/user/assignment" directory
unzip cars.zip && rm cars.zip
hadoop fs -mkdir /user/assignment
hadoop fs -put cars.csv /user/assignment
hadoop fs -ls -h /user/assignment
```

FINISHED

```
Archivreplace __MeA:C O ScXa/r.s_.czairps
.csv? [y]es, [n]o, [A]ll, [N]one, [r]ename:  NULL
(EOF or read error, treating as "[N]one" ...)
mkdir: `/user/assignment': File exists
put: `/user/assignment/cars.csv': File exists
Found 1 items
-rw-r--r--   2 zeppelin hadoop    400.0 M 2022-10-25 03:35 /user/assignment/cars.csv
```

Took 7 sec. Last updated by anonymous at October 28 2022, 2:17:31 PM.

```
%sh
## Check the first 5 rows of the cars.csv file
hadoop fs -cat /user/assignment/cars.csv | head -n 5
```

FINISHED

```
maker,model,mileage,manufacture_yeacat: Urn,aebnlgei nteo_ dwirsiptlea cteom eonutt,peuntg isnter_epaomw.er,body_type,color_s
lug,stk_year,transmission,door_count,seat_count,fuel_type,date_created,date_last_seen,price_eur
ford,galaxy,151000,2011,2000,103,,,None,man,5,7,diesel,2015-11-14 18:10:06.838319+00,2016-01-27 20:40:15.46361+00,10584.75
skoda,octavia,143476,2012,2000,81,,,None,man,5,5,diesel,2015-11-14 18:10:06.853411+00,2016-01-27 20:40:15.46361+00,8882.31
bmw,,97676,2010,1995,85,,,None,man,5,5,diesel,2015-11-14 18:10:06.861792+00,2016-01-27 20:40:15.46361+00,12065.06
skoda,fabia,111970,2004,1200,47,,,None,man,5,5,gasoline,2015-11-14 18:10:06.872313+00,2016-01-27 20:40:15.46361+00,2960.77
```

Took 2 sec. Last updated by anonymous at October 28 2022, 2:17:34 PM.

1. Write a Spark SQL query to create a table called used_cars from data and let it infer the schema.

Create a new table from the cars.csv file

FINISHED

Took 0 sec. Last updated by anonymous at October 28 2022, 2:17:35 PM.

```
%sql
CREATE TABLE IF NOT EXISTS usedcars
USING CSV
OPTIONS (path "/user/assignment/cars.csv", delimiter  "," , header "true", inferSchema "true")
```

FINISHED

Took 0 sec. Last updated by anonymous at October 28 2022, 2:17:35 PM.

View the first 5 rows in the new table

FINISHED

Took 0 sec. Last updated by anonymous at October 28 2022, 2:17:35 PM.

```
%sql
SELECT * FROM usedcars LIMIT 5
```

SPARK JOB FINISHED

settings

maker	model	mileage	manufacture_year	engine_displacement	engine_power	body_type	color_s
ford	galaxy	151000	2011	2000	103	null	null
skoda	octavia	143476	2012	2000	81	null	null
bmw	null	97676	2010	1995	85	null	null
skoda	fabia	111970	2004	1200	47	null	null
skoda	fabia	128886	2004	1200	47	null	null

Took 0 sec. Last updated by anonymous at October 28 2022, 2:17:35 PM.

2. Print the inferred schema to see if it makes sense.

```
%sql
DESC usedcars
```

FINISHED

settings

col_name	data_type	comment
maker	string	null
model	string	null
mileage	int	null
manufacture_year	int	null
engine_displacement	int	null
engine_power	int	null
body_type	string	null
color_slug	string	null
stk_year	string	null
transmission	string	null
door_count	string	null
seat_count	string	null
fuel_type	string	null
date_created	string	null
date_last_seen	string	null
price_eur	double	null

Took 0 sec. Last updated by anonymous at October 28 2022, 2:17:36 PM. (outdated)

Comment: The schema automatically inferred by Spark SQL has some columns' data types not making sense such as:

- door_count: the data type should be *integer* rather than *string*.
- seat_count: the data type should be *integer* rather than *string*.

3. Write Spark SQL queries to see how many missing values you have in each attribute. In the Text section of your notebook, write how many missing values in each column we have. Especially, mention those columns with more than 50% missing values:

```
%sql
SELECT
SUM(CASE WHEN maker IS NULL OR maker='' THEN 1 ELSE 0 END) AS maker,
SUM(CASE WHEN model IS NULL OR model='' THEN 1 ELSE 0 END) AS model,
SUM(CASE WHEN mileage IS NULL OR mileage='' THEN 1 ELSE 0 END) AS mileage,
SUM(CASE WHEN manufacture_year IS NULL OR manufacture_year='' THEN 1 ELSE 0 END) AS manufacture_year,
SUM(CASE WHEN engine_displacement IS NULL OR engine_displacement='' THEN 1 ELSE 0 END) AS engine_displacement,
SUM(CASE WHEN engine_power IS NULL OR engine_power='' THEN 1 ELSE 0 END) AS engine_power,
SUM(CASE WHEN body_type IS NULL OR body_type='' THEN 1 ELSE 0 END) AS body_type,
SUM(CASE WHEN color_slug IS NULL OR color_slug='' THEN 1 ELSE 0 END) AS color_slug,
SUM(CASE WHEN stk_year IS NULL OR stk_year='' THEN 1 ELSE 0 END) AS stk_year,
SUM(CASE WHEN transmission IS NULL OR transmission='' THEN 1 ELSE 0 END) AS transmission,
SUM(CASE WHEN door_count IS NULL OR door_count='' THEN 1 ELSE 0 END) AS door_count,
SUM(CASE WHEN seat_count IS NULL OR seat_count='' THEN 1 ELSE 0 END) AS seat_count,
SUM(CASE WHEN fuel_type IS NULL OR fuel_type='' THEN 1 ELSE 0 END) AS fuel_type,
SUM(CASE WHEN date_created IS NULL OR date_created='' THEN 1 ELSE 0 END) AS date_created,
SUM(CASE WHEN date_last_seen IS NULL OR date_last_seen='' THEN 1 ELSE 0 END) AS date_last_seen,
SUM(CASE WHEN price_eur IS NULL OR price_eur='' THEN 1 ELSE 0 END) AS price_eur
FROM usedcars
```

SPARK JOB FINISHED

settings

maker	model	mileage	manufacture_year	engine_displacement	engine_power	body_type	color_s
518915	1133361	362584	370578	743414	554877	1122914	3343411

Took 21 sec. Last updated by anonymous at October 28 2022, 2:17:58 PM.

Figure 1. Quantity of missing values in each attribute

```sql
%sql
SELECT
SUM(CASE WHEN maker IS NULL OR maker='' THEN 1 ELSE 0 END) / COUNT(*) * 100 AS maker,
SUM(CASE WHEN model IS NULL OR model='' THEN 1 ELSE 0 END) / COUNT(*) * 100 AS model,
SUM(CASE WHEN mileage IS NULL OR mileage='' THEN 1 ELSE 0 END) / COUNT(*) * 100 AS mileage,
SUM(CASE WHEN manufacture_year IS NULL OR manufacture_year='' THEN 1 ELSE 0 END) / COUNT(*) * 100 AS manufacture_year,
SUM(CASE WHEN engine_displacement IS NULL OR engine_displacement='' THEN 1 ELSE 0 END) / COUNT(*) * 100 AS engine_displacement,
SUM(CASE WHEN engine_power IS NULL OR engine_power='' THEN 1 ELSE 0 END) / COUNT(*) * 100 AS engine_power,
SUM(CASE WHEN body_type IS NULL OR body_type='' THEN 1 ELSE 0 END) / COUNT(*) * 100 AS body_type,
SUM(CASE WHEN color_slug IS NULL OR color_slug='' THEN 1 ELSE 0 END) / COUNT(*) * 100 AS color_slug,
SUM(CASE WHEN stk_year IS NULL OR stk_year='' THEN 1 ELSE 0 END) / COUNT(*) * 100 AS stk_year,
SUM(CASE WHEN transmission IS NULL OR transmission='' THEN 1 ELSE 0 END) / COUNT(*) * 100 AS transmission,
SUM(CASE WHEN door_count IS NULL OR door_count='' THEN 1 ELSE 0 END) / COUNT(*) * 100 AS door_count,
SUM(CASE WHEN seat_count IS NULL OR seat_count='' THEN 1 ELSE 0 END) / COUNT(*) * 100 AS seat_count,
SUM(CASE WHEN fuel_type IS NULL OR fuel_type='' THEN 1 ELSE 0 END) / COUNT(*) * 100 AS fuel_type,
SUM(CASE WHEN date_created IS NULL OR date_created='' THEN 1 ELSE 0 END) / COUNT(*) * 100 AS date_created,
SUM(CASE WHEN date_last_seen IS NULL OR date_last_seen='' THEN 1 ELSE 0 END) / COUNT(*) * 100 AS date_last_seen,
SUM(CASE WHEN price_eur IS NULL OR price_eur='' THEN 1 ELSE 0 END) / COUNT(*) * 100 AS price_eur
FROM usedcars
```

SPARK JOB FINISHED

settings

maker	model	mileage	manufacture_year	engine_displacement	engine_power	body_type	color_s
14.605343447853478	31.899495399829775	10.205262612752582	10.430261149164403	20.924075800357567	15.617527256515217	31.605454905722404	94.10340

Took 13 sec. Last updated by anonymous at October 28 2022, 2:18:11 PM.

Figure 2. The percentage of missing values in each attribute

Results:

	# of missing values	% of missing values
maker	518,915	14.605%
model	1,133,361	31.899%
mileage	362,585	10.205%
manufacture_year	370,579	10.430%
engine_displacement	743,415	20.924%
engine_power	554,878	15.618%
body_type	1,122,914	31.605%
color_slug	3,343,411	94.103%
stk_year	1,708,156	48.078%
transmission	741,630	20.874%
door_count	1,090,067	30.681%
seat_count	1,287,100	36.227%
fuel_type	1,847,606	52.003%
date_created	0	0.000%
date_last_seen	0	0.000%
price_eur	1	0.000%

Comment: There are 2 columns that have more than 50% missing values: color_slug and fuel_type.

4. Write a Spark SQL query to create a new table called clean_used_cars from used_cars with the following conditions.

- **Drop the columns with more than 50% missing values**
- **The manufacture year between 2000 and 2017 including 2000 and 2017**
- **Both maker and model exist in the row**
- **The price range is from 3000 to 2,000,000 (3000 ≤ price ≤ 2,000,000)**

```
%sql
CREATE TABLE IF NOT EXISTS clean_used_cars as
SELECT maker, model, mileage, manufacture_year,
engine_displacement, engine_power, body_type,
stk_year, transmission, door_count, seat_count,
date_created, date_last_seen, price_eur
FROM usedcars
WHERE manufacture_year>=2000 AND manufacture_year<=2017
AND maker <> ('')
AND model <>('')
AND price_eur>=3000 AND price_eur<=2000000
```

FINISHED

Took 1 sec. Last updated by anonymous at October 28 2022, 2:18:12 PM.

Figure 3. Create a new table

```
%sql
SELECT * FROM clean_used_cars LIMIT 5
```

SPARK JOB FINISHED

settings

maker	model	mileage	manufacture_year	engine_displacement	engine_power	body_type	stk_yea
ford	galaxy	151000	2011	2000	103	null	None
skoda	octavia	143476	2012	2000	81	null	None
skoda	octavia	105389	2003	1900	81	null	None
nissan	x-trail	149465	2005	2500	121	null	None
skoda	superb	269398	2005	1900	96	null	None

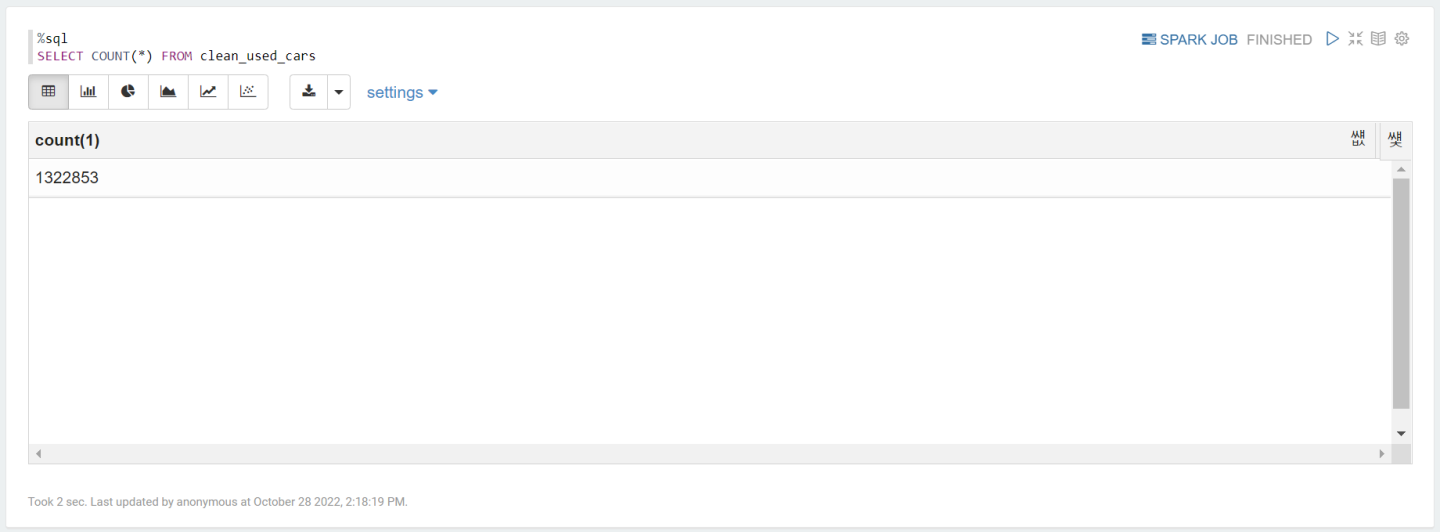
Took 0 sec. Last updated by anonymous at October 28 2022, 2:37:31 PM.

Figure 4. Verify the new table with the first 5 rows

Comments:

- In the new table, 2 attributes having more than 50% missing value (color_slug and fuel_type) are not included in the SELECT statement.
- The conditions (manufacture year, maker/model co-existing, price range) are included in the WHERE clause.

5. Write Spark SQL to find how many records remained clean_used_cars



Comments: In the new table clean_used_cars, there are 1,322,853 records remained.

6. Write a Spark SQL query to find the make and model for the cars with the top 10 highest average price.

```
%sql
SELECT maker, model, round(avg(price_eur),2)
FROM clean_used_cars
GROUP BY maker, model
ORDER BY avg(price_eur) DESC LIMIT 10
```

SPARK JOB FINISHED

settings

maker	model	round(avg(price_eur), 2)
lamborghini	aventador	365961.0
porsche	carrera-gt	302045.22
bmw	z8	245118.6
tesla	roadster	192880.28
tesla	model-x	176418.32
bentley	brooklands	138501.3
rolls-royce	wraith	137663.47
bentley	continental-gtc	129138.88

Took 4 sec. Last updated by anonymous at October 28 2022, 2:18:29 PM.

Comments:

In the ORDER BY clause, I used DESC (descending) to sort the results of the **highest** average price.

7. Write a Spark SQL query to find the make and model for the cars with the top 10 lowest average price.

```
%sql
SELECT maker, model, round(avg(price_eur),2)
FROM clean_used_cars
GROUP BY maker, model
ORDER BY avg(price_eur) ASC LIMIT 10
```

SPARK JOB FINISHED

settings

maker	model	round(avg(price_eur), 2)
skoda	galaxy	3071.8
rover	streetwise	3187.85
kia	retona	3200.25
bmw	transit	3290.16
chevrolet	alero	3305.37
hyundai	santamo	3391.01
opel	kadett	3405.48
fiat	128	3460.4

Took 2 sec. Last updated by anonymous at October 28 2022, 2:18:31 PM.

Comments:

The syntax is similar to Task #6, but in this task, I used ASC (ascending) in the ORDER BY clause to sort the results of the **lowest** average price.

8. Write a Spark SQL query to recommend top five make and model for Economic segment customers (Top five manufacturers in the 3000 to 20,000 price range; 3000 ≤ price < 20,000) - based on the top average price.

```
%sql
SELECT maker, model, round(avg(price_eur),2) average
FROM clean_used_cars
GROUP BY maker, model
HAVING avg(price_eur) >= 3000 AND avg(price_eur) < 20000
ORDER BY average DESC LIMIT 5
```

SPARK JOB FINISHED

settings

maker	model	average
volvo	v40	19915.03
peugeot	rcz	19755.5
skoda	130	19722.78
lexus	gs	19689.2
volkswagen	beetle	19458.79

Took 2 sec. Last updated by anonymous at October 28 2022, 2:54:57 PM.

9. Write a Spark SQL query to recommend top five make and model for Intermediate segment customers

```
%sql
SELECT maker, model, round(avg(price_eur),2) average
FROM clean_used_cars
GROUP BY maker, model
HAVING avg(price_eur) >= 20000 AND avg(price_eur) < 300000
ORDER BY average DESC LIMIT 5
```

SPARK JOB FINISHED

settings

maker	model	average
bmw	z8	245118.6
tesla	roadster	192880.28
tesla	model-x	176418.32
bentley	brooklands	138501.3
rolls-royce	wraith	137663.47

Took 2 sec. Last updated by anonymous at October 28 2022, 2:55:00 PM.

10. Write a Spark SQL query to recommend the top five make and model for the Luxury segment customers

```
%sql
SELECT maker, model, round(avg(price_eur),2) average
FROM clean_used_cars
GROUP BY maker, model
HAVING avg(price_eur) >= 300000 AND avg(price_eur) < 2000000
ORDER BY average DESC LIMIT 5
```

SPARK JOB FINISHED

settings

maker	model	average
lamborghini	aventador	365961.0
porsche	carrera-gt	302045.22

Took 2 sec. Last updated by anonymous at October 28 2022, 2:55:02 PM.

Comment: There are only 2 results for the Luxury segment customers which are:

- Lamborghini Aventador
- Porsche Carrera-GT