4/21/2023

MIS 6346.503

Used Cars Project Report

Big Data Final Project

Mahathi Rallapalli - MXR210124
Shivani Kothapally - SXK220116
Sai kruthik reddy paduru - sxp210343
Sindhuja Sai batchu - Sxb220095

Prof. Antonio Paes

Introduction:

According to recent survey, nearly 70 percent of Americans would be likely to consider a used vehicle for their next auto purchase. There is substantial reasoning for it-

- Cost: The lower price of used automobiles as opposed to new cars is one of the most significant factors influencing their choice for them. Once it is driven off the lot, a new car can lose a lot of value, making it a considerably more expensive buy than a used car.
- Depreciation: In the first few years of ownership, new cars also experience rapid depreciation, which lowers their resale value. Buyers can escape the first depreciating hit and even make long-term financial savings by choosing a used automobile.
- Better value: Compared to new cars, used cars frequently represent a better value. Customers may be able to buy a used automobile with more amenities, greater quality, or higher efficiency for the same price as a new car.
- Availability: Thanks to improvements in manufacturing and technology, automobiles are more durable and dependable than before. It is now simpler for customers to select a used car that suits their needs because there are more used cars accessible than ever before.
- Environmental Concerns: Due to their negative effects on the environment, some people are also preferring old cars. While buying a used automobile can help lessen the demand for new cars and the resulting environmental impact, producing a new car can often need many resources and energy.

Problem Statement:

Data analysis and Business answers to a big dataset of used cars in US and Canada.

The used cars dataset contains a massive amount of data that can help businesses in the automotive industry to answer key business questions. Some of the possible business questions that can be addressed using this dataset are:

- Which features have the highest correlation with the price of a car?
- What could be the predicted price and range of popular cars?
- How do different brands and models of used cars compare in terms of price, mileage, and other specifications?

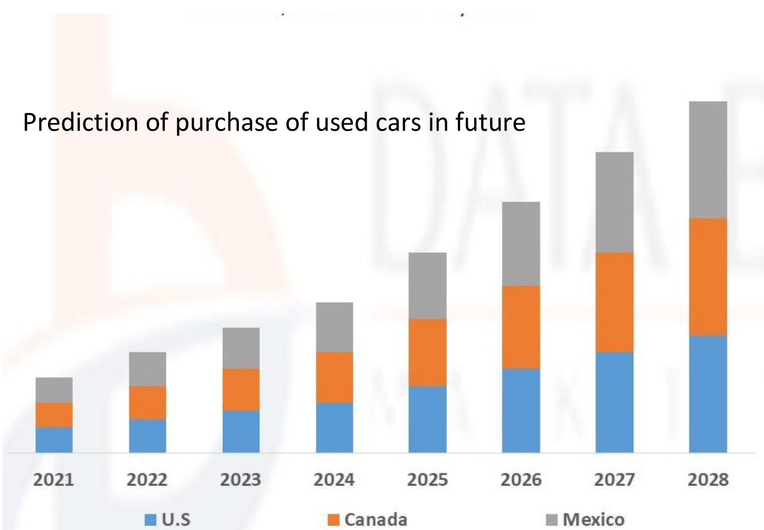
There are several reasons why conducting data analysis on the used cars in the US and Canada dataset could be beneficial:

Market insights: By analyzing the dataset, researchers can gain insights into the used car market in the US and Canada. This can include identifying trends in prices, sales volume, and popular models, as well as understanding how various factors such as age, mileage, and condition affect pricing.

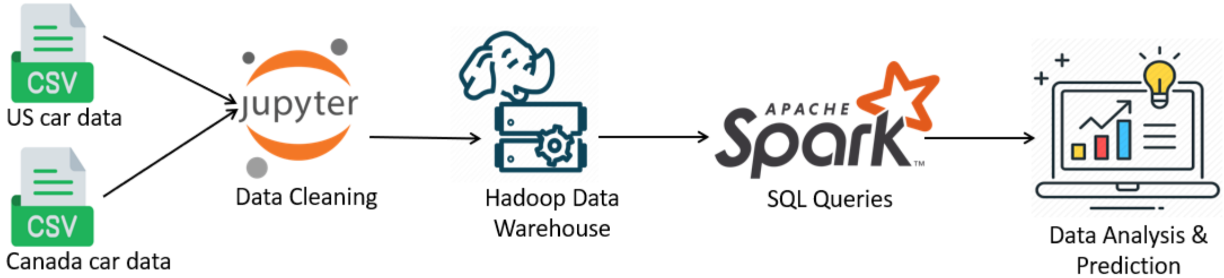
Customer behavior: Analysis of the dataset can also provide insights into customer behavior when purchasing used cars. This can include identifying the most important factors that buyers consider when making a purchase, such as price, mileage, and vehicle features.

Business strategy: The data analysis can be used to help businesses develop strategies for pricing and marketing their used cars. This includes identifying which models and features are in demand, what pricing strategies are most effective, and how to target specific customer segments.

Predictive modeling: The data can also be used to develop predictive models that can forecast future trends in the used car market. This can help businesses and investors make informed decisions about buying and selling used cars. Future prediction is that there would be a huge market of used cars in the US, Canada, Mexico.



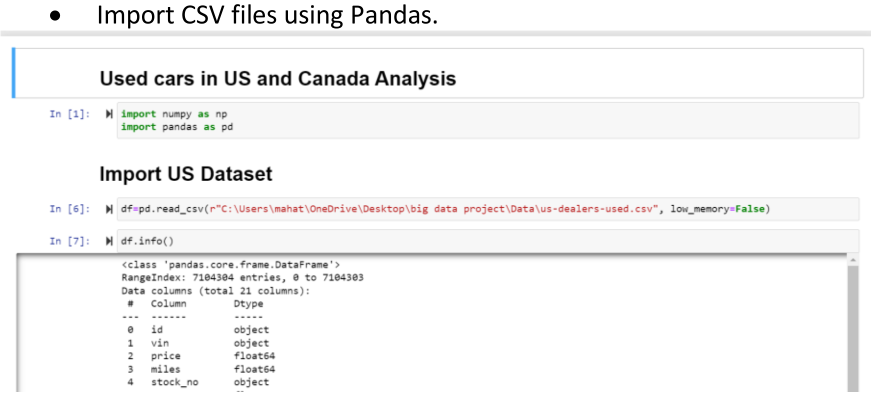
Project Architecture



We are going to use big data technologies such as Jupyter Notebook to include datasets of American cars and Canadian cars in the HDFS to do analysis of data and predict the value of used cars in the market. There is a plethora of factors that play a major role in deciding the resale value of a used car. In this project we will predict which factors affect the cost of cars, what is the most profitable car and queries as such.

DATA CLEANING:

- Import CSV files using Pandas.



Used cars in US and Canada Analysis

```
In [1]: import numpy as np
        import pandas as pd
```

Import US Dataset

```
In [6]: df=pd.read_csv(r"C:\Users\mahat\OneDrive\Desktop\big data project\Data\us-dealers-used.csv", low_memory=False)

In [7]: df.info()

<class 'pandas.core.frame.DataFrame'>
RangeIndex: 7104304 entries, 0 to 7104303
Data columns (total 21 columns):
 #   Column        Dtype
---  ------        -----
 0   id            object
 1   vin           object
 2   price         float64
 3   miles         float64
 4   stock_no      object
```

Count and drop all null values.

Count all null values in each row

```
In [8]:  num = df.isnull().sum()

         print(num)
```

```
id                   0
vin                  0
price           656779
miles            67290
stock_no        172288
year               160
make                 0
model             8040
trim             15169
body_type        27978
vehicle_type     38366
drivetrain       15921
transmission     13600
fuel_type        45956
engine_size     103266
engine_block    107110
seller_name      11111
street           31145
city             11404
state            11414
zip              11647
```

Replace the null values in prices with its mean value. And drop null values in other columns.

Data Cleaning

```
In [38]:  # Copy "price" column into an other variable
          price_column = df["price"].copy()

          # Drop all rows with missing values in all columns except "price"
          df = df.dropna(subset=df.columns.difference(["price"]))
          df["price"] = price_column

          # Calculate the mean value of non-missing values in "price"
          price_mean = df["price"].mean(skipna=True)

          # Replace missing values in "price" with the mean of non-missing values
          df["price"] = df["price"].fillna(value=price_mean)

          # Count the resulting DataFrame null values
          num = df.isnull().sum()

          print(num)
```

```
id              0
vin             0
price           0
miles           0
stock_no        0
year            0
make            0
model           0
trim            0
body_type       0
vehicle_type    0
drivetrain      0
transmission    0
fuel_type       0
engine_size     0
engine_block    0
seller_name     0
street          0
city            0
state           0
zip             0
dtype: int64
```

After data cleaning is completed for both data sets download the cleaned CSV file

Download cleaned datasets in CSV format

```
In [34]:  df.to_csv("us_cars.csv")
          df1.to_csv("canada_cars.csv")
```

Data Loading into Spark

mahathi@mahathi-VirtualBox: /usr/share/spark

```
>>> df = spark.read.csv("file:///home/mahathi/Downloads/us_cars.csv",inferSchema='true',header='true')
>>> df.count()
6750068
>>> df.show(5)
23/04/17 23:08:45 WARN CSVHeaderChecker: CSV header does not conform to the schema.
 Header: , id, vin, price, miles, stock_no, year, make, model, trim, body_type, vehicle_type, drivetrain, transmission, fuel_type, engine_size, engine
_block, seller_name, street, city, state, zip
 Schema: _c0, id, vin, price, miles, stock_no, year, make, model, trim, body_type, vehicle_type, drivetrain, transmission, fuel_type, engine_size, eng
ine_block, seller_name, street, city, state, zip
Expected: _c0 but found:
CSV file: file:///home/mahathi/Downloads/us_cars.csv
+---+-------------+-----------------+-----------------+--------+---------+-----+----------+-------------+--------+---------+------------+----------
+-----------+-----------------+----------+-----------+-----------------+------------------+-------------+----+-----+
|_c0|           id|              vin|            price|   miles| stock_no| year|      make|        model|    trim|body_type|vehicle_type|drivetrain
|transmission|         fuel_type|engine_size|engine_block|        seller_name|            street|         city|state|  zip|
+---+-------------+-----------------+-----------------+--------+---------+-----+----------+-------------+--------+---------+------------+----------
+-----------+-----------------+----------+-----------+-----------------+------------------+-------------+----+-----+
|  0|38b2f52e-8f5d|1GCWGFCF3F1284719|          20998.0|115879.0|W1T503168C|2015.0| Chevrolet|Express Cargo|Work Van|Cargo Van|       Truck|       RWD
|  Automatic|      E85 / Unleaded|        4.8|           V|nissan ellicott city|8569 Baltimore Na...|Ellicott City|   MD|21043|
|  1|97ba4955-ccf0|WBY7Z8C59JVB87514|          27921.0|  7339.0|    P33243|2018.0|       BMW|           i3|       s|Hatchback|         Car|       RWD
|  Automatic|Electric / Premiu...|        0.6|           I|hendrick honda po...|5381 N Federal Hi...|Pompano Beach|   FL|33064|
|  2|be1da9fd-0f34|ML32F4FJ2JHF10325|          11055.0| 39798.0|   WM2091A|2018.0|Mitsubishi|    Mirage G4|      SE|    Sedan|         Car|       FWD
|  Automatic|           Unleaded|        1.2|           I|  russ darrow toyota|2700 West Washing...|    West Bend|   WI|53095|
|  3|84327e45-6cb6|1GCPTEE15K1291189|          52997.0| 28568.0|  9U2Y425A|2019.0| Chevrolet|     Colorado|     ZR2|   Pickup|       Truck|       4WD
|  Automatic|             Diesel|        2.8|           I|           young kia|308 North Main St...|       Layton|   UT|84041|
|  4|cde691c3-91dd|1G2AL18F087312093|27889.284844765512|188485.0|   T36625A|2008.0|   Pontiac|           G5|    Base|    Coupe|         Car|       FWD
|  Automatic|           Unleaded|        2.2|           I|       pappas toyota|    10011 Spencer Rd| Saint Peters|   MO|63376|
+---+-------------+-----------------+-----------------+--------+---------+-----+----------+-------------+--------+---------+------------+----------
+-----------+-----------------+----------+-----------+-----------------+------------------+-------------+----+-----+
only showing top 5 rows
```

mahathi@mahathi-VirtualBox: /usr/share/spark

```
>>> df1 = spark.read.load("file:///home/mahathi/Downloads/canada_cars.csv",format='csv',inferSchema='true',header='true')
>>> df1.show(5)
23/04/17 23:12:48 WARN CSVHeaderChecker: CSV header does not conform to the schema.
 Header: , id, vin, price, miles, stock_no, year, make, model, trim, body_type, vehicle_type, drivetrain, transmission, fuel_type, engine_size, engine
_block, seller_name, street, city, state, zip
 Schema: _c0, id, vin, price, miles, stock_no, year, make, model, trim, body_type, vehicle_type, drivetrain, transmission, fuel_type, engine_size, eng
ine_block, seller_name, street, city, state, zip
Expected: _c0 but found:
CSV file: file:///home/mahathi/Downloads/canada_cars.csv
+---+-------------+-----------------+--------+-------+---------+------+-----+-----+----+---------+------------+----------+------------+--------------
-----+-----------+------------+--------------------+--------------------+--------------------+-----+-------+
|_c0|           id|              vin|   price|  miles| stock_no|  year| make|model|trim|body_type|vehicle_type|drivetrain|transmission|          fuel
_type|engine_size|engine_block|         seller_name|              street|                city|state|    zip|
+---+-------------+-----------------+--------+-------+---------+------+-----+-----+----+---------+------------+----------+------------+--------------
-----+-----------+------------+--------------------+--------------------+--------------------+-----+-------+
|  0|b39ea795-eca9|19UNC1B01HY800062|179999.0| 9966.0|  V-P4139|2017.0|Acura|  NSX|Base|    Coupe|         Car|       4WD|   Automatic|Electric / Prem
iu...|        3.5|           V|     edmundston honda|    475 Rue Victoria|          Edmundston|   NB|E3V 2K7|
|  1|026cb5b1-6e3e|19UNC1B02HY800023|179995.0| 5988.0|PPAP70374|2017.0|Acura|  NSX|Base|    Coupe|         Car|       4WD|   Automatic|Electric / Prem
iu...|        3.5|           V|garage daniel les...|2795 Route-du-prs...|Notre-dame-des-pins|   QC|G0M 1K0|
|  2|5cd5d5b2-5cc2|19UNC1B02HY800071|168528.0|24242.0|   B21085|2017.0|Acura|  NSX|Base|    Coupe|         Car|       4WD|   Automatic|Electric / Prem
iu...|        3.5|           V|       lougheed acura|1388 Lougheed Hig...|           Coquitlam|   BC|V3K 6S4|
|  3|b32473ed-5922|19UNC1B02LY800001|220000.0| 6637.0|   AP5333|2020.0|Acura|  NSX|Base|    Coupe|         Car|       4WD|   Automatic|Electric / Prem
iu...|        3.5|           V|     drive autogroup|1305 Parkway Suit...|           Pickering|   ON|L1V 3P2|
|  4|ac40c9fc-0676|19UNC1B02LY800001|220000.0| 6637.0|   AP5333|2020.0|Acura|  NSX|Base|    Coupe|         Car|       4WD|   Automatic|Electric / Prem
iu...|        3.5|           V|      acura pickering|   575 Kingston Road|           Pickering|   ON|L1V 3N7|
+---+-------------+-----------------+--------+-------+---------+------+-----+-----+----+---------+------------+----------+------------+--------------
-----+-----------+------------+--------------------+--------------------+--------------------+-----+-------+
only showing top 5 rows
```

Merge both datasets, using UNION

```
>>> df = df.union(df1)
>>> df.count()
7011263
>>> df.show(5)
23/04/17 23:18:24 WARN CSVHeaderChecker: CSV header does not conform to the schema.
 Header: , id, vin, price, miles, stock_no, year, make, model, trim, body_type, vehicle_type, drivetrain, transmission, fuel_type, engine_size, engine
_block, seller_name, street, city, state, zip
 Schema: _c0, id, vin, price, miles, stock_no, year, make, model, trim, body_type, vehicle_type, drivetrain, transmission, fuel_type, engine_size, eng
ine_block, seller_name, street, city, state, zip
Expected: _c0 but found:
CSV file: file:///home/mahathi/Downloads/us_cars.csv
+---+-------------+-----------------+-----------------+--------+---------+-----+----------+-------------+--------+---------+------------+----------
+-----------+-----------------+----------+-----------+-----------------+------------------+-------------+----+-----+
|_c0|           id|              vin|            price|   miles| stock_no| year|      make|        model|    trim|body_type|vehicle_type|drivetrain
|transmission|         fuel_type|engine_size|engine_block|        seller_name|            street|         city|state|  zip|
+---+-------------+-----------------+-----------------+--------+---------+-----+----------+-------------+--------+---------+------------+----------
+-----------+-----------------+----------+-----------+-----------------+------------------+-------------+----+-----+
|  0|38b2f52e-8f5d|1GCWGFCF3F1284719|          20998.0|115879.0|W1T503168C|2015.0| Chevrolet|Express Cargo|Work Van|Cargo Van|       Truck|       RWD
|  Automatic|      E85 / Unleaded|        4.8|           V|nissan ellicott city|8569 Baltimore Na...|Ellicott City|   MD|21043|
|  1|97ba4955-ccf0|WBY7Z8C59JVB87514|          27921.0|  7339.0|    P33243|2018.0|       BMW|           i3|       s|Hatchback|         Car|       RWD
|  Automatic|Electric / Premiu...|        0.6|           I|hendrick honda po...|5381 N Federal Hi...|Pompano Beach|   FL|33064|
|  2|be1da9fd-0f34|ML32F4FJ2JHF10325|          11055.0| 39798.0|   WM2091A|2018.0|Mitsubishi|    Mirage G4|      SE|    Sedan|         Car|       FWD
|  Automatic|           Unleaded|        1.2|           I|  russ darrow toyota|2700 West Washing...|    West Bend|   WI|53095|
|  3|84327e45-6cb6|1GCPTEE15K1291189|          52997.0| 28568.0|  9U2Y425A|2019.0| Chevrolet|     Colorado|     ZR2|   Pickup|       Truck|       4WD
|  Automatic|             Diesel|        2.8|           I|           young kia|308 North Main St...|       Layton|   UT|84041|
|  4|cde691c3-91dd|1G2AL18F087312093|27889.284844765512|188485.0|   T36625A|2008.0|   Pontiac|           G5|    Base|    Coupe|         Car|       FWD
|  Automatic|           Unleaded|        2.2|           I|       pappas toyota|    10011 Spencer Rd| Saint Peters|   MO|63376|
+---+-------------+-----------------+-----------------+--------+---------+-----+----------+-------------+--------+---------+------------+----------
+-----------+-----------------+----------+-----------+-----------------+------------------+-------------+----+-----+
only showing top 5 rows
```

Clean the data by using only relevant columns

```
mahathi@mahathi-VirtualBox: /usr/share/spark
>>> apt_column = ['price', 'miles', 'year', 'make', 'model', 'trim', 'body_type', 'vehicle_type', 'seller_name', 'city', 'state']
>>> df = df.select(*apt_column)
>>> df.printSchema()
root
 |-- price: double (nullable = true)
 |-- miles: double (nullable = true)
 |-- year: double (nullable = true)
 |-- make: string (nullable = true)
 |-- model: string (nullable = true)
 |-- trim: string (nullable = true)
 |-- body_type: string (nullable = true)
 |-- vehicle_type: string (nullable = true)
 |-- seller_name: string (nullable = true)
 |-- city: string (nullable = true)
 |-- state: string (nullable = true)
```

Business Questions

QUERY 1:

- Top 5 most expensive cars and their Mileage.
- Top 5 cars with the maximum Mileage and their respective prices.

```
mahathi@mahathi-VirtualBox: /usr/share/spark
>>> df.groupBy("seller_name","vehicle_type","make","state","miles").mean("price").sort("avg(price)", ascending=False).show(5)
+--------------------+------------+-------+-----+------+----------+
|         seller_name|vehicle_type|   make|state| miles|avg(price)|
+--------------------+------------+-------+-----+------+----------+
|   mclaren charlotte|         Car|Porsche|   NC|5984.0| 1499996.0|
|ferrari of newpor...|         Car|Ferrari|   CA| 859.0| 1495000.0|
|             karbuds|         Car|Ferrari|   FL| 697.0| 1479900.0|
|cauley ferrari of...|         Car|Ferrari|   MI| 697.0| 1479900.0|
|      cauley ferrari|         Car|Ferrari|   MI| 697.0| 1479900.0|
+--------------------+------------+-------+-----+------+----------+
only showing top 5 rows

>>> df.groupBy("seller_name","vehicle_type","make","state","price").mean("miles").sort("avg(miles)", ascending=False).show(5)
+--------------------+------------+---------+-----+------------------+----------+
|         seller_name|vehicle_type|     make|state|             price|avg(miles)|
+--------------------+------------+---------+-----+------------------+----------+
|   3 sons auto sales|       Truck|     Ford|   IL|           12990.0| 3000000.0|
|   3 sons auto sales|       Truck|    Acura|   IL|            7990.0| 3000000.0|
|cleveland motor c...|       Truck|Chevrolet|   OH|27889.284844765512| 2975291.0|
| lexus of clearwater|       Truck|     Ford|   FL|           45989.0| 2575500.0|
|  lexus of tampa bay|       Truck|     Ford|   FL|           45989.0| 2575500.0|
+--------------------+------------+---------+-----+------------------+----------+
only showing top 5 rows
```

By observing the results, we can conclude that the most expensive cars drive very less miles, and their market share is dominated by Ferrari cars. If the expectation of the vehicle is to give more miles and costs less than Trucks are preferable and Ford is making most of them.

QUERY 2:

- Market Share of all companies

```
mahathi@mahathi-VirtualBox: /usr/share/spark

>>> from pyspark.sql.functions import count, sum, col
>>> total_cars = df.count()
>>> car_counts = df.groupBy("make").agg(count("*").alias("car_count"))
>>> market_share = car_counts.withColumn("market_share", col("car_count") / total_cars * 100).orderBy(col("market_share").desc())
>>> market_share.show()
+-------------+---------+------------------+
|         make|car_count|      market_share|
+-------------+---------+------------------+
|         Ford|   917695|13.088868581880325|
|    Chevrolet|   791692|11.291717341083912|
|       Toyota|   639819| 9.125588356905167|
|        Honda|   514314| 7.335539973325776|
|       Nissan|   448512| 6.397021478155933|
|         Jeep|   427022|6.0905146476462235|
|      Hyundai|   278517|3.9724226576581136|
|          BMW|   266249|3.7974470505528033|
|          GMC|   254194|3.6255094124981477|
|          Kia|   231725|3.3050393345678235|
|Mercedes-Benz|   219127|3.1253570148488223|
|          RAM|   211126| 3.011240628115077|
|        Dodge|   203757|2.9061383091748234|
|       Subaru|   186700|2.6628583181090195|
|        Lexus|   177118|2.5261924991260494|
|   Volkswagen|   163148|2.3269416651464936|
|         Audi|   125041|1.7834304603892337|
|        Mazda|   117525|1.6762315149210634|
|     Cadillac|   100415|  1.43219559728397|
|        Acura|    98537|1.4054101236824235|
+-------------+---------+------------------+
only showing top 20 rows
```

We can observe that Ford, Chevrolet, and Toyota take over 30% of the market share.

QUERY 3

- Cars with the maximum market share over the time

```
mahathi@mahathi-VirtualBox: /usr/share/spark

>>> from pyspark.sql.functions import rank
>>> from pyspark.sql.window import Window
>>> market_share_df = df.groupBy("make", "year").agg(sum("price").alias("total_sales"))
>>> total_sales_df = market_share_df.groupBy("year").agg(sum("total_sales").alias("total_sales_year"))
>>> market_share_df = market_share_df.join(total_sales_df, "year")
>>> market_share_df = market_share_df.withColumn("market_share", market_share_df["total_sales"]/market_share_df["total_sales_year"])
>>> market_share_df = market_share_df.drop("total_sales_year")
>>> window = Window.partitionBy("year").orderBy(col("market_share").desc())
>>> highest_market_share_df = market_share_df.select("*", rank().over(window).alias("rank")).filter("rank = 1").orderBy(col("year").desc())
>>> highest_market_share_df.show()
+------+---------+-----------+-------------------+----+
|  year|     make|total_sales|       market_share|rank|
+------+---------+-----------+-------------------+----+
|2022.0|    Acura|   741339.0| 0.5762690388869801|   1|
|2021.0|      BMW| 6.0415764E7|0.18083605850610515|   1|
|2020.0|     Ford|1.28449534E8|0.11803152348112898|   1|
|2019.0|     Ford|1.75805011E8|0.12680506700451305|   1|
|2018.0|     Ford|1.52509477E8|0.10917571936817914|   1|
|2017.0|     Ford|1.16771678E8|0.09709864842935285|   1|
|2016.0|     Ford| 5.9067879E7|0.10168036494560344|   1|
|2015.0|     Ford| 3.8696919E7| 0.1161150294535269|   1|
|2014.0|     Ford| 2.9019577E7|0.13448475682502942|   1|
|2013.0|     Ford| 2.2517639E7|0.15397808781053388|   1|
|2012.0|     Ford|  9583555.0|0.10927574921132005|   1|
|2011.0|     Ford|  8059224.0|0.13604589043527415|   1|
|2010.0|     Ford|  5789767.0| 0.1468610075334358|   1|
|2009.0|   Toyota|  2782296.0|0.10447235284257729|   1|
|2008.0|     Ford|  3085392.0|0.13136248506074627|   1|
|2007.0|     Ford|  1893850.0|0.11788251299113409|   1|
|2006.0|Chevrolet|  1282363.0|0.12225757451096274|   1|
|2005.0|     Ford|   571554.0| 0.1011453026029315|   1|
|2004.0|     Ford|   691316.0| 0.1403028954332292|   1|
|2003.0|  Ferrari|   855314.0| 0.2430659282448614|   1|
+------+---------+-----------+-------------------+----+
only showing top 20 rows
```

Market share of each car over the time

The car industry is highly competitive, with many different makes of cars vying for market share.

Ford appears to be a consistently popular make, with sales in the top rank for many years. This could be due to factors such as brand loyalty, affordability, or a wide range of available models.

Other makes such as Acura, BMW, Chevrolet, Ferrari, and Toyota have also had strong sales in certain years.

The data covers a period of 20 years, from 2003 to 2022. Looking at trends over time could provide insights into changes in consumer preferences, economic conditions, or other factors that affect car sales.

The data can also be used to compare the performance of different car makes and identify patterns in sales. For example, some makes may have experienced a decline in sales over time, while others may have seen a steady increase.

QUERY 4

- Correlation between “price” and other factors. Which factors affect the price of cars.

```
mahathi@mahathi-VirtualBox: /usr/share/spark
>>> from pyspark.sql.functions import corr
>>> df.select(corr("price", "miles"),
...           corr("price", "year"),
...           corr("price", "make"),
...           corr("price", "model"),
...           corr("price", "trim"),
...           corr("price", "body_type"),
...           corr("price", "vehicle_type"),
...           corr("price", "seller_name"),
...           corr("price", "city"),
...           corr("price", "state")).show()
23/04/18 02:22:40 WARN package: Truncated the string representation of a plan since it was too large. This behavior can be adjusted by setting 'spark.sql.debug.maxToStringFields'.
+-------------------+------------------+-----------------+------------------+------------------+--------------------+-----------------------+----------------------+------------------
+-----------------+
|  corr(price, miles)| corr(price, year)|corr(price, make)|corr(price, model)| corr(price, trim)|corr(price, body_type)|corr(price, vehicle_type)|corr(price, seller_name)|corr(price, city)
|corr(price, state)|
+-------------------+------------------+-----------------+------------------+------------------+--------------------+-----------------------+----------------------+------------------
+-----------------+
|-0.39453844575817054|0.4204254553788824|             null|0.5259314566474526|0.3964831637921107|                null|                   null|                  null|             null
|             null|
+-------------------+------------------+-----------------+------------------+------------------+--------------------+-----------------------+----------------------+------------------
+-----------------+
```

This table shows the correlation coefficients between the price column and each of the other columns in the dataset. There is a moderate negative correlation between price and miles (-0.39) and a moderate positive correlation between price and year (0.42). The correlation coefficients indicate that as the mileage of a used car increases, its resale value tends to decrease. There is also a moderate positive correlation between price and model (0.53) and trim (0.40). However, there is no correlation between price and make, body_type, vehicle_type, seller_name, city, or state it is to the contrary belief to see that the make of the model plays no role in the resale value of the car.

QUERY 5

- Average price and mileage of cars in descending order.

```
mahathi@mahathi-VirtualBox: /usr/share/spark

>>> from pyspark.sql.functions import avg
>>> df.groupBy('body_type', 'vehicle_type').agg(avg('price').alias('avg_price'), avg('miles').alias('avg_mileage')).orderBy(col("avg_price").de
sc()).show()
+-------------+------------+------------------+------------------+
|    body_type|vehicle_type|         avg_price|       avg_mileage|
+-------------+------------+------------------+------------------+
|        Targa|         Car| 90393.29837207461|16112.233151803948|
|  Convertible|         Car|42803.842576089446|46131.584302221585|
|     Roadster|         Car| 40319.49759893186| 42645.98096324239|
|  Chassis Cab|       Truck|  38839.0586236172| 80770.40039361469|
|       Pickup|       Truck| 37762.54774964416| 60611.45363272988|
|        Coupe|         Car| 37144.91974477872|45036.341911445015|
|      Cutaway|       Truck| 34179.59007675749| 62138.87579393084|
|        Combi|       Truck| 32862.31548023485| 37447.54384133612|
|          SUV|       Truck|28664.655819943084|52916.044624374386|
|        Sedan|       Truck|27889.284844765512|          112444.0|
|    Cargo Van|       Truck|27884.268035456862|55824.589241850204|
|Passenger Van|       Truck| 27218.77891922609| 65716.39552919708|
|        Wagon|         Car|23935.940556910828|  57583.2343285676|
|    Crossover|       Truck|23907.845813081614|44822.950457599836|
|      Minivan|       Truck| 22000.65040204023| 78364.44849522546|
|        Sedan|         Car|  21486.6736575073|   56733.799486144|
|    Crossover|         Car| 21441.95918465987|27674.301040163995|
|    Hatchback|         Car|  19364.1764448648|  53997.2282544099|
|      Car Van|       Truck|18621.382981840703| 89043.42561205273|
|          SUV|         Car| 17808.70500012859| 63104.34043701106|
+-------------+------------+------------------+------------------+
only showing top 20 rows
```

The output shows the average price and mileage for each body type and vehicle type. The most expensive car is “Targa" is $90,393.30 and the average mileage is 16,112.23. The table provides insights into how prices and mileage vary by body type and vehicle type, which can be useful for understanding market trends and making informed purchasing decisions.

PREDICTION:

```
mahathi@mahathi-VirtualBox: /usr/share/spark
>>> from pyspark.ml.regression import RandomForestRegressor
>>> from pyspark.ml.feature import VectorAssembler
>>> from pyspark.ml.linalg import Vectors
>>> from pyspark.ml.evaluation import RegressionEvaluator
>>> assembler = VectorAssembler(inputCols=["miles", "year"], outputCol="features")
>>> data = assembler.transform(df)
>>> (trainingData, testData) = data.randomSplit([0.7, 0.3])
>>> rf = RandomForestRegressor(labelCol="price", featuresCol="features", numTrees=10, maxDepth=5, seed=42)
>>> model = rf.fit(trainingData)
23/04/20 17:37:56 WARN MemoryStore: Not enough space to cache rdd_140_2 in memory! (computed 63.2 MiB so far)
23/04/20 17:37:56 WARN BlockManager: Persisting block rdd_140_2 to disk instead.
23/04/20 17:38:01 WARN MemoryStore: Not enough space to cache rdd_140_3 in memory! (computed 3.4 MiB so far)
23/04/20 17:38:01 WARN BlockManager: Persisting block rdd_140_3 to disk instead.
23/04/20 17:38:09 WARN MemoryStore: Not enough space to cache rdd_140_4 in memory! (computed 17.6 MiB so far)
23/04/20 17:38:09 WARN BlockManager: Persisting block rdd_140_4 to disk instead.
23/04/20 17:38:16 WARN MemoryStore: Not enough space to cache rdd_140_5 in memory! (computed 17.6 MiB so far)
23/04/20 17:38:16 WARN BlockManager: Persisting block rdd_140_5 to disk instead.
23/04/20 17:38:23 WARN MemoryStore: Not enough space to cache rdd_140_6 in memory! (computed 17.6 MiB so far)
23/04/20 17:38:23 WARN BlockManager: Persisting block rdd_140_6 to disk instead.
23/04/20 17:38:30 WARN MemoryStore: Not enough space to cache rdd_140_7 in memory! (computed 17.6 MiB so far)
23/04/20 17:38:30 WARN BlockManager: Persisting block rdd_140_7 to disk instead.
23/04/20 17:38:38 WARN MemoryStore: Not enough space to cache rdd_140_8 in memory! (computed 5.2 MiB so far)
23/04/20 17:38:38 WARN BlockManager: Persisting block rdd_140_8 to disk instead.
23/04/20 17:38:45 WARN MemoryStore: Not enough space to cache rdd_140_9 in memory! (computed 17.6 MiB so far)
23/04/20 17:38:45 WARN BlockManager: Persisting block rdd_140_9 to disk instead.
23/04/20 17:38:49 WARN MemoryStore: Not enough space to cache rdd_140_1 in memory! (computed 63.2 MiB so far)
23/04/20 17:38:51 WARN MemoryStore: Not enough space to cache rdd_140_2 in memory! (computed 63.2 MiB so far)
23/04/20 17:38:52 WARN MemoryStore: Not enough space to cache rdd_140_3 in memory! (computed 63.2 MiB so far)
23/04/20 17:38:54 WARN MemoryStore: Not enough space to cache rdd_140_4 in memory! (computed 63.2 MiB so far)
23/04/20 17:38:55 WARN MemoryStore: Not enough space to cache rdd_140_5 in memory! (computed 63.2 MiB so far)
23/04/20 17:38:58 WARN MemoryStore: Not enough space to cache rdd_140_1 in memory! (computed 63.2 MiB so far)
23/04/20 17:39:00 WARN MemoryStore: Not enough space to cache rdd_140_2 in memory! (computed 63.2 MiB so far)
23/04/20 17:39:01 WARN MemoryStore: Not enough space to cache rdd_140_3 in memory! (computed 63.2 MiB so far)
23/04/20 17:39:02 WARN MemoryStore: Not enough space to cache rdd_140_4 in memory! (computed 63.2 MiB so far)
23/04/20 17:39:04 WARN MemoryStore: Not enough space to cache rdd_140_5 in memory! (computed 63.2 MiB so far)
23/04/20 17:39:08 WARN MemoryStore: Not enough space to cache rdd_140_1 in memory! (computed 63.2 MiB so far)
23/04/20 17:39:09 WARN MemoryStore: Not enough space to cache rdd_140_2 in memory! (computed 63.2 MiB so far)
23/04/20 17:39:10 WARN MemoryStore: Not enough space to cache rdd_140_3 in memory! (computed 63.2 MiB so far)
23/04/20 17:39:12 WARN MemoryStore: Not enough space to cache rdd_140_4 in memory! (computed 63.2 MiB so far)
23/04/20 17:39:13 WARN MemoryStore: Not enough space to cache rdd_140_5 in memory! (computed 63.2 MiB so far)
23/04/20 17:39:18 WARN MemoryStore: Not enough space to cache rdd_140_1 in memory! (computed 63.2 MiB so far)
23/04/20 17:39:19 WARN MemoryStore: Not enough space to cache rdd_140_2 in memory! (computed 63.2 MiB so far)
23/04/20 17:39:22 WARN MemoryStore: Not enough space to cache rdd_140_3 in memory! (computed 63.2 MiB so far)
23/04/20 17:39:23 WARN MemoryStore: Not enough space to cache rdd_140_4 in memory! (computed 63.2 MiB so far)
23/04/20 17:39:25 WARN MemoryStore: Not enough space to cache rdd_140_5 in memory! (computed 63.2 MiB so far)
>>> predictions = model.transform(testData)
>>> evaluator = RegressionEvaluator(labelCol="price", predictionCol="prediction", metricName="rmse")
>>> rmse = evaluator.evaluate(predictions)
>>> new_data = [(10000, 2025, "Ford", "Mustang", "GT", "Coupe", "New", "Dealer", "New York", "NY"),
...            (50000, 2038, "Toyota", "Camry", "SE", "Sedan", "Used", "Private Seller", "Los Angeles", "CA")]
>>> new_data_df = spark.createDataFrame(new_data, ["miles", "year", "make", "model", "trim", "body_type", "vehicle_type", "seller_name", "city", "state"])
>>> assembler = VectorAssembler(inputCols=["miles", "year"], outputCol="features")
>>> new_data_features = assembler.transform(new_data_df)
>>> predictions = model.transform(new_data_features)
>>> predictions.select("make", "model", "year", "miles", "prediction").show()
```

Predicted prices of cars in future:

make	model	year	miles	prediction
Ford	Mustang	2025	10000	41127.86025576979
Toyota	Camry	2038	50000	36882.95919682251

CONCLUSION:

Based on this study, we can draw several conclusions regarding the efficiency, market share, and pricing of different types of vehicles. Firstly, it appears that the most expensive cars tend to offer lower mileage, while trucks are both the most efficient and least expensive option, while also providing maximum mileage. Ford, Chevrolet, and Toyota are the dominant players in the market, with Honda and Nissan following closely behind in terms of sales numbers and potential for future growth. It is worth noting that Ford has consistently ranked as the top truck maker for the past two decades.

Furthermore, our correlation study has revealed that mileage has a negative impact on the price of used cars, while the year, model, and trim of the vehicle have a positive effect on the price. This suggests that dealers should prioritize acquiring recently bought and lightly used cars in order to maximize their profits.

Finally, our prediction model has a root mean square error of 4574.7. While this is not ideal, we believe that with additional data and further training, we can reduce the RSME value and achieve more accurate predictions.

USED CARS

BIG DATA FINAL PROJECT

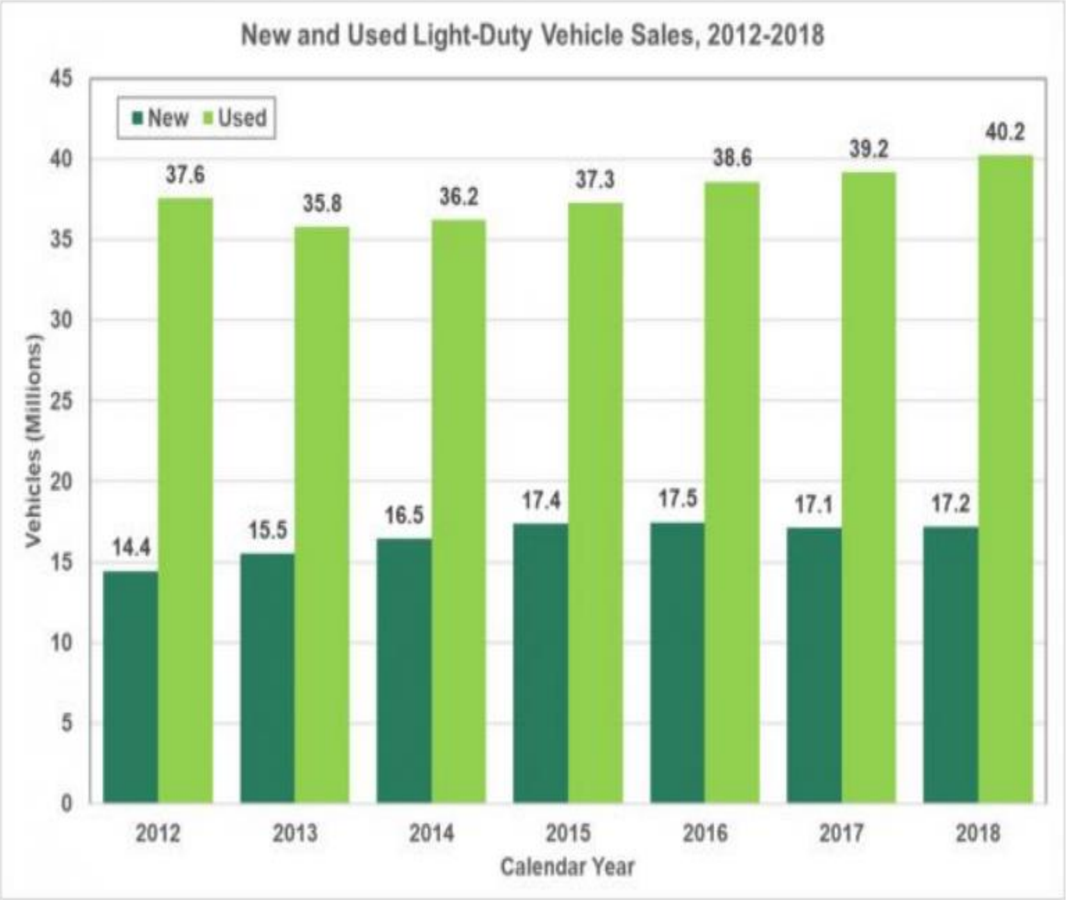
FRAMEWORK USED

- Jupyter Notebook
- Hadoop
- Spark

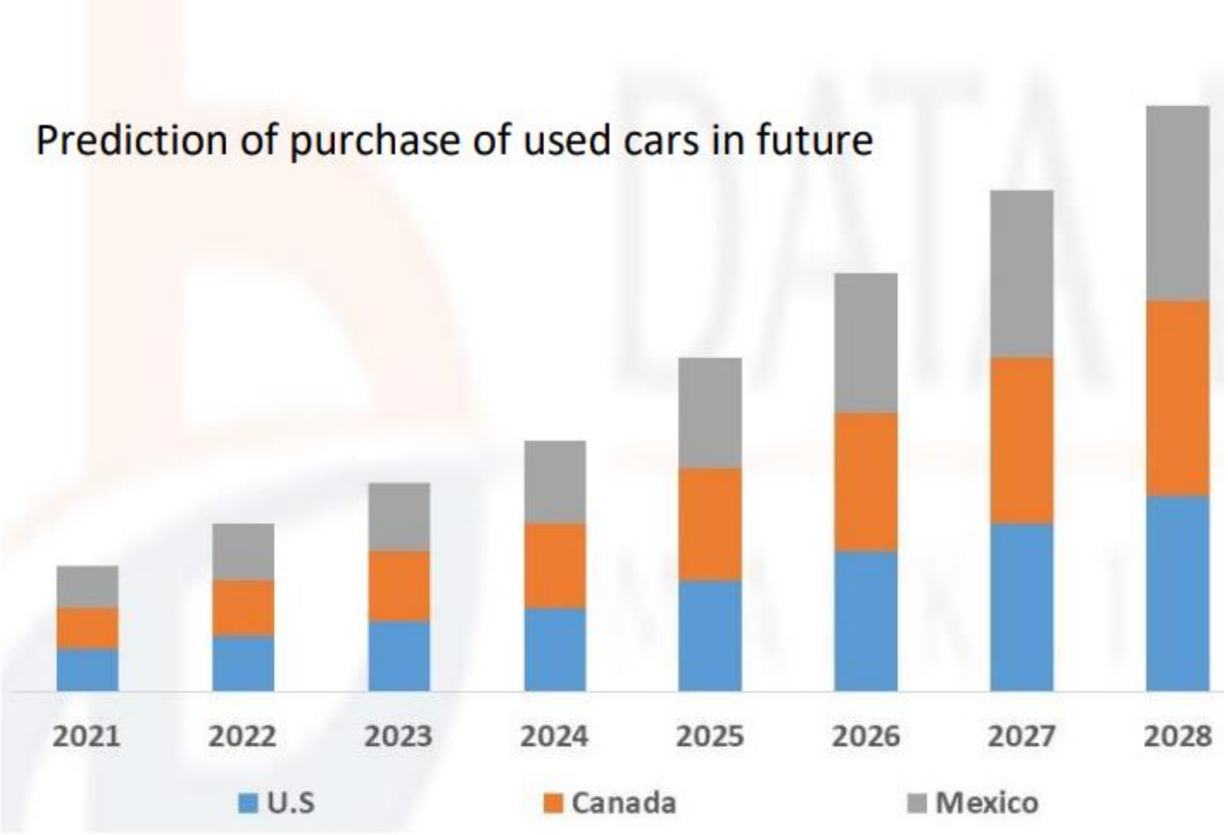
OBJECTIVE

In this project, we are leveraging these technology components to:

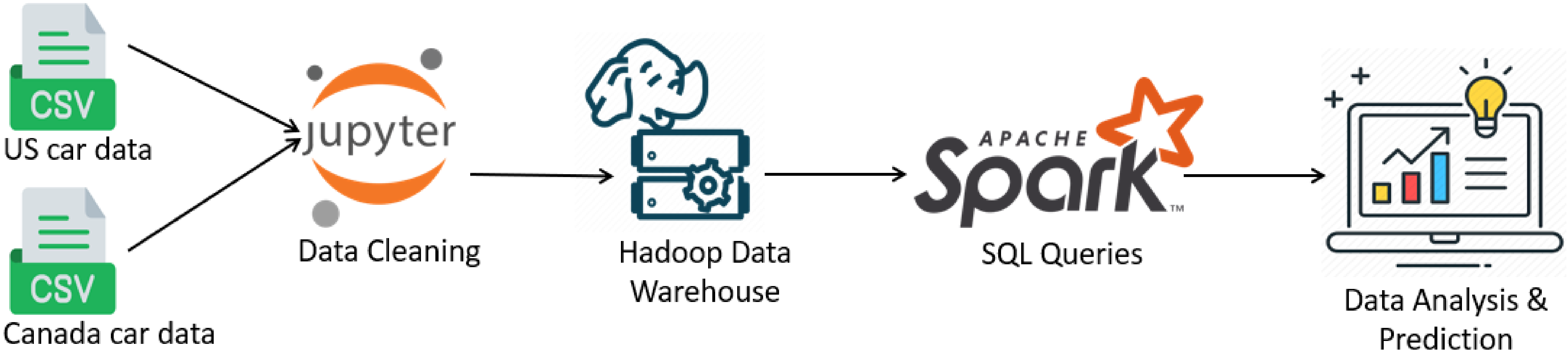
- Perform analysis on used car market data with multi-million records
  - By loading the unstructured data to data warehouse
  - Utilize spark analytics engine to interface with HDFS and process data
- Train ML models and predict car prices using generated model
- Provide Business recommendations to the sellers to make their sale profitable



New and Used Light-Duty Vehicle Sales, 2012-2018
New
Used
Vehicles (Millions)
Calendar Year
0
5
10
15
20
25
30
35
40
45
2012
2013
2014
2015
2016
2017
2018
14.4
37.6
15.5
35.8
16.5
36.2
17.4
37.3
17.5
38.6
17.1
39.2
17.2
40.2



Prediction of purchase of used cars in future
2021
2022
2023
2024
2025
2026
2027
2028
U.S
Canada
Mexico



CSV
US car data
CSV
Canada car data
jupyter
Data Cleaning
Hadoop Data Warehouse
APACHE Spark
SQL Queries
Data Analysis & Prediction

DATASET SUMMARY

Data Field	Data Description
id	This is a GUID and unique in the feed
vin	17 char long VIN of the car
price	The car price as listed on the website
miles	The car miles/odometer as listed on the website
stock_no	Stock number of the car listed on the website
year	Model Year of the car (VIN Decoded)
make	Make of the car (VIN Decoded)
model	Model of the car (VIN Decoded)
trim	Trim of the car (VIN Decoded)
vehicle_type	Vehicle type of the car (VIN Decoded)
body_type	Body type of the car (VIN Decoded)
drivetrain	Drivetrain of the car (VIN Decoded)
fuel_type	Fuel type of the car (VIN Decoded)
engine_block	Engine block of the car (VIN Decoded)
engine_size	Engine size of the car (VIN Decoded)
transmission	Transmission of the car (VIN Decoded)
seller_name	Dealer Name
city	Dealer Location
state	Dealer Location
zip	Dealer Location

There are two data sets in source zipped file:

- US used car dataset –
  - 7,104,304 records
  - 1.28 GB size
- Canada used car dataset –
  - 393,603 records
  - 70 MB size

Link to dataset zipped file:

https://www.kaggle.com/datasets/rupeshraundal/marketcheck-automotive-data-us-canada

DATA CLEANING

Count all null values in each row

In [8]:
```
num = df.isnull().sum()

print(num)
```

```
id                   0
vin                  0
price           656779
miles            67290
stock_no        172288
year               160
make                 0
model             8040
trim             15169
body_type        27978
vehicle_type     38366
drivetrain       15921
transmission     13600
fuel_type        45956
engine_size     103266
engine_block    107110
seller_name      11111
street           31145
city             11404
state            11414
zip              11647
```

```
id              0
vin             0
price           0
miles           0
stock_no        0
year            0
make            0
model           0
trim            0
body_type       0
vehicle_type    0
drivetrain      0
transmission    0
fuel_type       0
engine_size     0
engine_block    0
seller_name     0
street          0
city            0
state           0
zip             0
dtype: int64
```

REPLACE NULL VALUES IN PRICE WITH MEAN VALUE

In [14]:

```
n=len(df)
print(n)
```

```
7104304
```

Data Cleaning

In [38]:

```
# Copy "price" column into an other variable
price_column = df["price"].copy()

# Drop all rows with missing values in all columns except "price"
df = df.dropna(subset=df.columns.difference(["price"]))
df["price"] = price_column

# Calculate the mean value of non-missing values in "price"
price_mean = df["price"].mean(skipna=True)

# Replace missing values in "price" with the mean of non-missing values
df["price"] = df["price"].fillna(value=price_mean)

# Count the resulting DataFrame null values
num = df.isnull().sum()

print(num)
```

SPARK - LOADING DATA INTO DATAFRAMES

mahathi@mahathi-VirtualBox: /usr/share/spark

```
>>> df = spark.read.csv("file:///home/mahathi/Downloads/us_cars.csv",inferSchema='true',header='true')
>>> df.count()
6750068
>>> df.show(5)
23/04/17 23:08:45 WARN CSVHeaderChecker: CSV header does not conform to the schema.
 Header: , id, vin, price, miles, stock_no, year, make, model, trim, body_type, vehicle_type, drivetrain, transmission, fuel_type, engine_size, engine
_block, seller_name, street, city, state, zip
 Schema: _c0, id, vin, price, miles, stock_no, year, make, model, trim, body_type, vehicle_type, drivetrain, transmission, fuel_type, engine_size, eng
ine_block, seller_name, street, city, state, zip
Expected: _c0 but found:
CSV file: file:///home/mahathi/Downloads/us_cars.csv
+---+------------+-----------------+-----------------+--------+---------+------+-----------+-------------+--------+---------+------------+----------
+-----------+------------------+-----------+------------+-------------------+-------------------+------------+----+-----+
|_c0|          id|              vin|            price|   miles| stock_no|  year|       make|        model|    trim|body_type|vehicle_type|drivetrain
|transmission|         fuel_type|engine_size|engine_block|        seller_name|             street|        city|state|  zip|
+---+------------+-----------------+-----------------+--------+---------+------+-----------+-------------+--------+---------+------------+----------
+-----------+------------------+-----------+------------+-------------------+-------------------+------------+----+-----+
|  0|38b2f52e-8f5d|1GCWGFCF3F1284719|          20998.0|115879.0|W1T503168C|2015.0| Chevrolet|Express Cargo|Work Van|Cargo Van|       Truck|       RWD
|   Automatic|     E85 / Unleaded|        4.8|           V|nissan ellicott city|8569 Baltimore Na...|Ellicott City|   MD|21043|
|  1|97ba4955-ccf0|WBY7Z8C59JVB87514|          27921.0|  7339.0|    P33243|2018.0|        BMW|           i3|       s|Hatchback|         Car|       RWD
|   Automatic|Electric / Premiu...|        0.6|           I|hendrick honda po...|5381 N Federal Hi...|Pompano Beach|   FL|33064|
|  2|be1da9fd-0f34|ML32F4FJ2JHF10325|          11055.0| 39798.0|   WM2091A|2018.0|Mitsubishi|    Mirage G4|      SE|    Sedan|         Car|       FWD
|   Automatic|           Unleaded|        1.2|           I|  russ darrow toyota|2700 West Washing...|   West Bend|   WI|53095|
|  3|84327e45-6cb6|1GCPTEE15K1291189|          52997.0| 28568.0|  9U2Y425A|2019.0| Chevrolet|     Colorado|     ZR2|   Pickup|       Truck|       4WD
|   Automatic|             Diesel|        2.8|           I|           young kia|308 North Main St...|      Layton|   UT|84041|
|  4|cde691c3-91dd|1G2AL18F087312093|27889.284844765512|188485.0|   T36625A|2008.0|   Pontiac|           G5|    Base|    Coupe|         Car|       FWD
|   Automatic|           Unleaded|        2.2|           I|        pappas toyota|    10011 Spencer Rd| Saint Peters|   MO|63376|
+---+------------+-----------------+-----------------+--------+---------+------+-----------+-------------+--------+---------+------------+----------
+-----------+------------------+-----------+------------+-------------------+-------------------+------------+----+-----+
only showing top 5 rows
```

mahathi@mahathi-VirtualBox: /usr/share/spark

```
>>> df1 = spark.read.load("file:///home/mahathi/Downloads/canada_cars.csv",format='csv',inferSchema='true',header='true')
>>> df1.show(5)
23/04/17 23:12:48 WARN CSVHeaderChecker: CSV header does not conform to the schema.
 Header: , id, vin, price, miles, stock_no, year, make, model, trim, body_type, vehicle_type, drivetrain, transmission, fuel_type, engine_size, engine
_block, seller_name, street, city, state, zip
 Schema: _c0, id, vin, price, miles, stock_no, year, make, model, trim, body_type, vehicle_type, drivetrain, transmission, fuel_type, engine_size, eng
ine_block, seller_name, street, city, state, zip
Expected: _c0 but found:
CSV file: file:///home/mahathi/Downloads/canada_cars.csv
+---+-------------+-----------------+--------+-------+---------+------+-----+-----+----+---------+------------+----------+------------+------------
----+-----------+------------+-------------------+-------------------+-------------------+-----+-------+
|_c0|           id|              vin|   price|  miles| stock_no|  year| make|model|trim|body_type|vehicle_type|drivetrain|transmission|        fuel
_type|engine_size|engine_block|        seller_name|             street|               city|state|    zip|
+---+-------------+-----------------+--------+-------+---------+------+-----+-----+----+---------+------------+----------+------------+------------
----+-----------+------------+-------------------+-------------------+-------------------+-----+-------+
|  0|b39ea795-eca9|19UNC1B01HY800062|179999.0| 9966.0|  V-P4139|2017.0|Acura|  NSX|Base|    Coupe|         Car|       4WD|   Automatic|Electric / Prem
iu...|        3.5|           V|    edmundston honda|    475 Rue Victoria|          Edmundston|   NB|E3V 2K7|
|  1|026cb5b1-6e3e|19UNC1B02HY800023|179995.0| 5988.0|PPAP70374|2017.0|Acura|  NSX|Base|    Coupe|         Car|       4WD|   Automatic|Electric / Prem
iu...|        3.5|           V|garage daniel les...|2795 Route-du-prs...|Notre-dame-des-pins|   QC|G0M 1K0|
|  2|5cd5d5b2-5cc2|19UNC1B02HY800071|168528.0|24242.0|   B21085|2017.0|Acura|  NSX|Base|    Coupe|         Car|       4WD|   Automatic|Electric / Prem
iu...|        3.5|           V|      lougheed acura|1388 Lougheed Hig...|           Coquitlam|   BC|V3K 6S4|
|  3|b32473ed-5922|19UNC1B02LY800001|220000.0| 6637.0|   AP5333|2020.0|Acura|  NSX|Base|    Coupe|         Car|       4WD|   Automatic|Electric / Prem
iu...|        3.5|           V|     drive autogroup|1305 Parkway Suit...|            Pickering|   ON|L1V 3P2|
|  4|ac40c9fc-0676|19UNC1B02LY800001|220000.0| 6637.0|   AP5333|2020.0|Acura|  NSX|Base|    Coupe|         Car|       4WD|   Automatic|Electric / Prem
iu...|        3.5|           V|      acura pickering|    575 Kingston Road|            Pickering|   ON|L1V 3N7|
+---+-------------+-----------------+--------+-------+---------+------+-----+-----+----+---------+------------+----------+------------+------------
----+-----------+------------+-------------------+-------------------+-------------------+-----+-------+
only showing top 5 rows
```

MERGE DATASETS

```
>>> df = df.union(df1)
>>> df.count()
7011263
>>> df.show(5)
23/04/17 23:18:24 WARN CSVHeaderChecker: CSV header does not conform to the schema.
 Header: , id, vin, price, miles, stock_no, year, make, model, trim, body_type, vehicle_type, drivetrain, transmission, fuel_type, engine_size, engine
_block, seller_name, street, city, state, zip
 Schema: _c0, id, vin, price, miles, stock_no, year, make, model, trim, body_type, vehicle_type, drivetrain, transmission, fuel_type, engine_size, eng
ine_block, seller_name, street, city, state, zip
Expected: _c0 but found:
CSV file: file:///home/mahathi/Downloads/us_cars.csv
+---+-------------+-----------------+------------------+--------+----------+------+----------+-------------+--------+---------+------------+----------
+------------+--------------------+-----------+------------+--------------------+--------------------+-------------+-----+-----+
|_c0|           id|              vin|             price|   miles|  stock_no|  year|      make|        model|    trim|body_type|vehicle_type|drivetrain
|transmission|           fuel_type|engine_size|engine_block|         seller_name|              street|         city|state|  zip|
+---+-------------+-----------------+------------------+--------+----------+------+----------+-------------+--------+---------+------------+----------
+------------+--------------------+-----------+------------+--------------------+--------------------+-------------+-----+-----+
|  0|38b2f52e-8f5d|1GCWGFCF3F1284719|           20998.0|115879.0|W1T503168C|2015.0| Chevrolet|Express Cargo|Work Van|Cargo Van|       Truck|       RWD
|   Automatic|      E85 / Unleaded|        4.8|           V|nissan ellicott city|8569 Baltimore Na...|Ellicott City|   MD|21043|
|  1|97ba4955-ccf0|WBY7Z8C59JVB87514|           27921.0|  7339.0|    P33243|2018.0|       BMW|           i3|       s|Hatchback|         Car|       RWD
|   Automatic|Electric / Premiu...|        0.6|           I|hendrick honda po...|5381 N Federal Hi...|Pompano Beach|   FL|33064|
|  2|be1da9fd-0f34|ML32F4FJ2JHF10325|           11055.0| 39798.0|   WM2091A|2018.0|Mitsubishi|    Mirage G4|      SE|    Sedan|         Car|       FWD
|   Automatic|            Unleaded|        1.2|           I|  russ darrow toyota|2700 West Washing...|    West Bend|   WI|53095|
|  3|84327e45-6cb6|1GCPTEE15K1291189|           52997.0| 28568.0|  9U2Y425A|2019.0| Chevrolet|     Colorado|     ZR2|   Pickup|       Truck|       4WD
|   Automatic|              Diesel|        2.8|           I|           young kia|308 North Main St...|       Layton|   UT|84041|
|  4|cde691c3-91dd|1G2AL18F087312093|27889.284844765512|188485.0|   T36625A|2008.0|   Pontiac|           G5|    Base|    Coupe|         Car|       FWD
|   Automatic|            Unleaded|        2.2|           I|       pappas toyota|    10011 Spencer Rd| Saint Peters|   MO|63376|
+---+-------------+-----------------+------------------+--------+----------+------+----------+-------------+--------+---------+------------+----------
+------------+--------------------+-----------+------------+--------------------+--------------------+-------------+-----+-----+
only showing top 5 rows
```

RELEVANT COLUMNS

mahathi@mahathi-VirtualBox: /usr/share/spark

```
>>> apt_column = ['price', 'miles', 'year', 'make', 'model', 'trim', 'body_type', 'vehicle_type', 'seller_name', 'city', 'state']
>>> df = df.select(*apt_column)
>>> df.printSchema()
root
 |-- price: double (nullable = true)
 |-- miles: double (nullable = true)
 |-- year: double (nullable = true)
 |-- make: string (nullable = true)
 |-- model: string (nullable = true)
 |-- trim: string (nullable = true)
 |-- body_type: string (nullable = true)
 |-- vehicle_type: string (nullable = true)
 |-- seller_name: string (nullable = true)
 |-- city: string (nullable = true)
 |-- state: string (nullable = true)
```

PRICES VS MILEAGE

mahathi@mahathi-VirtualBox: /usr/share/spark

```
>>> df.groupBy("seller_name","vehicle_type","make","state","miles").mean("price").sort("avg(price)", ascending=False).show(5)
+--------------------+------------+-------+-----+------+----------+
|         seller_name|vehicle_type|   make|state| miles|avg(price)|
+--------------------+------------+-------+-----+------+----------+
|   mclaren charlotte|         Car|Porsche|   NC|5984.0| 1499996.0|
|ferrari of newpor...|         Car|Ferrari|   CA| 859.0| 1495000.0|
|             karbuds|         Car|Ferrari|   FL| 697.0| 1479900.0|
|cauley ferrari of...|         Car|Ferrari|   MI| 697.0| 1479900.0|
|      cauley ferrari|         Car|Ferrari|   MI| 697.0| 1479900.0|
+--------------------+------------+-------+-----+------+----------+
only showing top 5 rows

>>> df.groupBy("seller_name","vehicle_type","make","state","price").mean("miles").sort("avg(miles)", ascending=False).show(5)
+--------------------+------------+---------+-----+------------------+----------+
|         seller_name|vehicle_type|     make|state|             price|avg(miles)|
+--------------------+------------+---------+-----+------------------+----------+
|   3 sons auto sales|       Truck|     Ford|   IL|           12990.0| 3000000.0|
|   3 sons auto sales|       Truck|    Acura|   IL|            7990.0| 3000000.0|
|cleveland motor c...|       Truck|Chevrolet|   OH|27889.284844765512| 2975291.0|
| lexus of clearwater|       Truck|     Ford|   FL|           45989.0| 2575500.0|
|  lexus of tampa bay|       Truck|     Ford|   FL|           45989.0| 2575500.0|
+--------------------+------------+---------+-----+------------------+----------+
only showing top 5 rows
```

MARKET SHARE OF CARS

mahathi@mahathi-VirtualBox: /usr/share/spark

```
>>> from pyspark.sql.functions import count, sum, col
>>> total_cars = df.count()
>>> car_counts = df.groupBy("make").agg(count("*").alias("car_count"))
>>> market_share = car_counts.withColumn("market_share", col("car_count") / total_cars * 100).orderBy(col("market_share").desc())
>>> market_share.show()
+-------------+---------+------------------+
|         make|car_count|      market_share|
+-------------+---------+------------------+
|         Ford|   917695|13.088868581880325|
|    Chevrolet|   791692|11.291717341083912|
|       Toyota|   639819| 9.125588356905167|
|        Honda|   514314| 7.335539973325776|
|       Nissan|   448512| 6.397021478155933|
|         Jeep|   427022|6.0905146476462235|
|      Hyundai|   278517|3.9724226576581136|
|          BMW|   266249|3.7974470505528033|
|          GMC|   254194|3.6255094124981477|
|          Kia|   231725|3.3050393345678235|
|Mercedes-Benz|   219127|3.1253570148488223|
|          RAM|   211126| 3.011240628115077|
|        Dodge|   203757|2.9061383091748234|
|       Subaru|   186700|2.6628583181090195|
|        Lexus|   177118|2.5261924991260494|
|   Volkswagen|   163148|2.3269416651464936|
|         Audi|   125041|1.7834304603892337|
|        Mazda|   117525|1.6762315149210634|
|     Cadillac|   100415|  1.43219559728397|
|        Acura|    98537|1.4054101236824235|
+-------------+---------+------------------+
only showing top 20 rows
```

MARKET SHARE OVER A PERIOD OF TIME

mahathi@mahathi-VirtualBox: /usr/share/spark

```
>>> from pyspark.sql.functions import rank
>>> from pyspark.sql.window import Window
>>> market_share_df = df.groupBy("make", "year").agg(sum("price").alias("total_sales"))
>>> total_sales_df = market_share_df.groupBy("year").agg(sum("total_sales").alias("total_sales_year"))
>>> market_share_df = market_share_df.join(total_sales_df, "year")
>>> market_share_df = market_share_df.withColumn("market_share", market_share_df["total_sales"]/market_share_df["total_sales_year"])
>>> market_share_df = market_share_df.drop("total_sales_year")
>>> window = Window.partitionBy("year").orderBy(col("market_share").desc())
>>> highest_market_share_df = market_share_df.select("*", rank().over(window).alias("rank")).filter("rank = 1").orderBy(col("year").desc())
>>> highest_market_share_df.show()
+------+---------+------------+-------------------+----+
|  year|     make| total_sales|       market_share|rank|
+------+---------+------------+-------------------+----+
|2022.0|    Acura|    741339.0| 0.5762690388869801|   1|
|2021.0|      BMW| 6.0415764E7|0.18083605850610515|   1|
|2020.0|     Ford|1.28449534E8|0.11803152348112898|   1|
|2019.0|     Ford|1.75805011E8|0.12680506700451305|   1|
|2018.0|     Ford|1.52509477E8|0.10917571936817914|   1|
|2017.0|     Ford|1.16771678E8|0.09709864842935285|   1|
|2016.0|     Ford| 5.9067879E7|0.10168036494560344|   1|
|2015.0|     Ford| 3.8696919E7| 0.1161150294535269|   1|
|2014.0|     Ford| 2.9019577E7|0.13448475682502942|   1|
|2013.0|     Ford| 2.2517639E7|0.15397808781053388|   1|
|2012.0|     Ford|   9583555.0|0.10927574921132005|   1|
|2011.0|     Ford|   8059224.0|0.13604589043527415|   1|
|2010.0|     Ford|   5789767.0| 0.1468610075334358|   1|
|2009.0|   Toyota|   2782296.0|0.10447235284257729|   1|
|2008.0|     Ford|   3085392.0|0.13136248506074627|   1|
|2007.0|     Ford|   1893850.0|0.11788251299113409|   1|
|2006.0|Chevrolet|   1282363.0|0.12225757451096274|   1|
|2005.0|     Ford|    571554.0| 0.1011453026029315|   1|
|2004.0|     Ford|    691316.0| 0.1403028954332292|   1|
|2003.0|  Ferrari|    855314.0| 0.2430659282448614|   1|
+------+---------+------------+-------------------+----+
only showing top 20 rows
```

CORRELATION BETWEEN PRICE AND OTHER FACTORS

mahathi@mahathi-VirtualBox: /usr/share/spark

```
>>> from pyspark.sql.functions import corr
>>> df.select(corr("price", "miles"),
...           corr("price", "year"),
...           corr("price", "make"),
...           corr("price", "model"),
...           corr("price", "trim"),
...           corr("price", "body_type"),
...           corr("price", "vehicle_type"),
...           corr("price", "seller_name"),
...           corr("price", "city"),
...           corr("price", "state")).show()
23/04/18 02:22:40 WARN package: Truncated the string representation of a plan since it was too large. This behavior can be adjusted by setting 'spark.sql.debug.maxToStringFields'.
+-------------------+------------------+------------------+-------------------+-------------------+-----------------------+--------------------------+-------------------------+-----------------
------------------+
|  corr(price, miles)| corr(price, year)|corr(price, make)|corr(price, model)| corr(price, trim)|corr(price, body_type)|corr(price, vehicle_type)|corr(price, seller_name)|corr(price, city)
|corr(price, state)|
+-------------------+------------------+------------------+-------------------+-------------------+-----------------------+--------------------------+-------------------------+-----------------
------------------+
|-0.39453844575817054|0.4204254553788824|             null|0.5259314566474526|0.3964831637921107|                   null|                      null|                     null|            null
|              null|
+-------------------+------------------+------------------+-------------------+-------------------+-----------------------+--------------------------+-------------------------+-----------------
------------------+
```

RANDOM FOREST REGRESSION MODEL

```
mahathi@mahathi-VirtualBox: /usr/share/spark
>>> from pyspark.ml.regression import RandomForestRegressor
>>> from pyspark.ml.feature import VectorAssembler
>>> from pyspark.ml.linalg import Vectors
>>> from pyspark.ml.evaluation import RegressionEvaluator
>>> assembler = VectorAssembler(inputCols=["miles", "year"], outputCol="features")
>>> data = assembler.transform(df)
>>> (trainingData, testData) = data.randomSplit([0.7, 0.3])
>>> rf = RandomForestRegressor(labelCol="price", featuresCol="features", numTrees=10, maxDepth=5, seed=42)
>>> model = rf.fit(trainingData)
23/04/20 17:37:56 WARN MemoryStore: Not enough space to cache rdd_140_2 in memory! (computed 63.2 MiB so far)
23/04/20 17:37:56 WARN BlockManager: Persisting block rdd_140_2 to disk instead.
23/04/20 17:38:01 WARN MemoryStore: Not enough space to cache rdd_140_3 in memory! (computed 3.4 MiB so far)
23/04/20 17:38:01 WARN BlockManager: Persisting block rdd_140_3 to disk instead.
23/04/20 17:38:09 WARN MemoryStore: Not enough space to cache rdd_140_4 in memory! (computed 17.6 MiB so far)
23/04/20 17:38:09 WARN BlockManager: Persisting block rdd_140_4 to disk instead.
23/04/20 17:38:16 WARN MemoryStore: Not enough space to cache rdd_140_5 in memory! (computed 17.6 MiB so far)
23/04/20 17:38:16 WARN BlockManager: Persisting block rdd_140_5 to disk instead.
23/04/20 17:38:23 WARN MemoryStore: Not enough space to cache rdd_140_6 in memory! (computed 17.6 MiB so far)
23/04/20 17:38:23 WARN BlockManager: Persisting block rdd_140_6 to disk instead.
23/04/20 17:38:30 WARN MemoryStore: Not enough space to cache rdd_140_7 in memory! (computed 17.6 MiB so far)
23/04/20 17:38:30 WARN BlockManager: Persisting block rdd_140_7 to disk instead.
23/04/20 17:38:38 WARN MemoryStore: Not enough space to cache rdd_140_8 in memory! (computed 5.2 MiB so far)
23/04/20 17:38:38 WARN BlockManager: Persisting block rdd_140_8 to disk instead.
23/04/20 17:38:45 WARN MemoryStore: Not enough space to cache rdd_140_9 in memory! (computed 17.6 MiB so far)
23/04/20 17:38:45 WARN BlockManager: Persisting block rdd_140_9 to disk instead.
23/04/20 17:38:49 WARN MemoryStore: Not enough space to cache rdd_140_1 in memory! (computed 63.2 MiB so far)
23/04/20 17:38:51 WARN MemoryStore: Not enough space to cache rdd_140_2 in memory! (computed 63.2 MiB so far)
23/04/20 17:38:52 WARN MemoryStore: Not enough space to cache rdd_140_3 in memory! (computed 63.2 MiB so far)
23/04/20 17:38:54 WARN MemoryStore: Not enough space to cache rdd_140_4 in memory! (computed 63.2 MiB so far)
23/04/20 17:38:55 WARN MemoryStore: Not enough space to cache rdd_140_5 in memory! (computed 63.2 MiB so far)
23/04/20 17:38:58 WARN MemoryStore: Not enough space to cache rdd_140_1 in memory! (computed 63.2 MiB so far)
23/04/20 17:39:00 WARN MemoryStore: Not enough space to cache rdd_140_2 in memory! (computed 63.2 MiB so far)
23/04/20 17:39:01 WARN MemoryStore: Not enough space to cache rdd_140_3 in memory! (computed 63.2 MiB so far)
23/04/20 17:39:02 WARN MemoryStore: Not enough space to cache rdd_140_4 in memory! (computed 63.2 MiB so far)
23/04/20 17:39:04 WARN MemoryStore: Not enough space to cache rdd_140_5 in memory! (computed 63.2 MiB so far)
23/04/20 17:39:08 WARN MemoryStore: Not enough space to cache rdd_140_1 in memory! (computed 63.2 MiB so far)
23/04/20 17:39:09 WARN MemoryStore: Not enough space to cache rdd_140_2 in memory! (computed 63.2 MiB so far)
23/04/20 17:39:10 WARN MemoryStore: Not enough space to cache rdd_140_3 in memory! (computed 63.2 MiB so far)
23/04/20 17:39:12 WARN MemoryStore: Not enough space to cache rdd_140_4 in memory! (computed 63.2 MiB so far)
23/04/20 17:39:13 WARN MemoryStore: Not enough space to cache rdd_140_5 in memory! (computed 63.2 MiB so far)
23/04/20 17:39:18 WARN MemoryStore: Not enough space to cache rdd_140_1 in memory! (computed 63.2 MiB so far)
23/04/20 17:39:19 WARN MemoryStore: Not enough space to cache rdd_140_2 in memory! (computed 63.2 MiB so far)
23/04/20 17:39:22 WARN MemoryStore: Not enough space to cache rdd_140_3 in memory! (computed 63.2 MiB so far)
23/04/20 17:39:23 WARN MemoryStore: Not enough space to cache rdd_140_4 in memory! (computed 63.2 MiB so far)
23/04/20 17:39:25 WARN MemoryStore: Not enough space to cache rdd_140_5 in memory! (computed 63.2 MiB so far)
>>> predictions = model.transform(testData)
>>> evaluator = RegressionEvaluator(labelCol="price", predictionCol="prediction", metricName="rmse")
>>> rmse = evaluator.evaluate(predictions)
>>> new_data = [(10000, 2025, "Ford", "Mustang", "GT", "Coupe", "New", "Dealer", "New York", "NY"),
...             (50000, 2038, "Toyota", "Camry", "SE", "Sedan", "Used", "Private Seller", "Los Angeles", "CA")]
>>> new_data_df = spark.createDataFrame(new_data, ["miles", "year", "make", "model", "trim", "body_type", "vehicle_type", "seller_name", "city", "state"])
>>> assembler = VectorAssembler(inputCols=["miles", "year"], outputCol="features")
>>> new_data_features = assembler.transform(new_data_df)
>>> predictions = model.transform(new_data_features)
>>> predictions.select("make", "model", "year", "miles", "prediction").show()
```

PREDICTION

make	model	year	miles	prediction
Ford	Mustang	2025	10000	41127.86025576979
Toyota	Camry	2038	50000	36882.95919682251

- Prediction of new data

Ford Mustang and Toyota Camry prices in the years 2025 and 2038 respectively

BUSINESS RECOMMENDATION

- Car manufacturers should focus on producing more fuel-efficient vehicles, particularly trucks, as they are the most efficient and provide the maximum mileage while also being the least expensive option. This can help to attract more consumers who are looking for cost-effective and efficient vehicles.
- Dealers should prioritize acquiring recently bought and lightly used cars, as these tend to have a positive effect on the price, whereas mileage has a negative impact on the price. This can help dealers to maximize their profits and increase their sales.
- Companies that want to enter the market should focus on differentiating themselves by offering unique features or targeting specific niches, as the dominant players in the market are already well-established.

THANK YOU