Vehicle Fuel Consumption Analysis using MapReduce

18MIS0232- A. Pooja

18MIS0257- A. Poojitha

VIT®
Vellore Institute of Technology
(Deemed to be University under section 3 of UGC Act, 1956)

SCHOOL OF INFORMATION TECHNOLOGY AND ENGINEERING

Project Report for J component of

Course Code: SWE2011,

Course Title: Big Data Analytics

4

Faculty: Ranichandra C ,

Slot: B1+TB1

Fall Semester 2021-2022

On

Vehicle fuel Consumption Analysis using MapReduce

By

18MIS0232 - A.Pooja

18MIS0257 - A.Poojitha

Big Data Analytics Review 2

1.Discovery Phase:

a) Problem statement/ Initial Hypothesis

Big data analytic methodology has been used to derive important predictions or estimations in various areas. The reduction of greenhouse gas emissions and energy consumption in the automotive sector has emerged as a global issue. Eco-driving methods that can be used to save energy by improving automobile driving behaviour represent major solutions. Accurate data is required for fuel economy calculations. In this study, we attempt to build Vehicle fuel economy calculation using data available from fueleconomy.gov data portal. We chose this dataset because it has data related to both electric and gasoline consumed car details which make us analyse the accurate data for transition into a complete eco driving environment. In this project we are going to implement Hadoop Map Reduce program in Java. Based on this we can come to a conclusion about fuel consumptions.

b) Literature Study/Introduction

Introduction:

With the continuous growth of car ownership around the world, the energy consumption of its private cars has increased. Based on factors like manufactures, car types, fuel type and many more of USA, the trend of future transportation energy consumption can be predicted.
This study proposes a vehicle fuel consumption prediction method based on data collected from website having dataset contains United State vehicle fuel economy data starting from 1984- 2021. By matching the type of vehicle it's manufacturer and the fuel consumption data, the fuel consumption can be analysed, and the fuel consumption prediction models were constructed using Hadoop MapReduce programs.

Literature Survey:

[1] "Vehicle Fuel Consumption Prediction Method Based on Driving Behavior Data Collected from Smartphones", Journal of Advanced Transportation, vol. 2020, Article ID 9263605, 11 pages, 2020. https://doi.org/10.1155/2020/9263605

In this paper, a study on driving behavior data and fuel consumption data of taxi drivers is collected from OBD and mobile phone terminals, were matched and the correlation between driving behavior and fuel consumption was analyzed, and relevant driving behavior indicators affecting fuel consumption were extracted through the filter-based feature selection method three fuel consumption prediction models based on a Back Propagation neural network, SVR, and a random forest were constructed.

[2] DTG Big Data Analysis for Fuel Consumption Estimation Wonhee Cho* and Eunmi ChoiJ Inf Process Syst, Vol.13, No.2, pp.285~304, April 2017 ISSN 1976-913X (Print) https://doi.org/10.3745/JIPS.04.0031 ISSN 2092-805X (Electronic) https://www.koreascience.or.kr/article/JAKO201717447649033.pdf

They have researched to estimate the fuel consumption mileage using the driving patterns extracted from the DTG big data of commercial vehicles. They worked on big data analysis for the fuel consumption estimation in this paper. They have derived an analytical fuel consumption formula to estimate the fuel mileage using DTG data variables through a linear regression using a regression analysis using the driving pattern and analyzed the data correlation of DTG data, which contains driving patterns, especially actual driving logging data of commercial vehicles.

a) S/w and H/w Requirements

Software Requirements:
Cloudera Virtual Machine
Hadoop

Hardware Requirements:
System Memory: 3GB
2 CPU cores
windows 10 : 64 bit (Virtual box is needed)

2 Data Preparation Phase:

a. Dataset identified, URL of csv or exl file.

https://www.fueleconomy.gov/feg/download.shtml
This dataset contains United State vehicle fuel economy data starting from 1984. Data from Fuel Economy.GOV.Here, the data set is in CSV file format containing details aboutus-vehicle-fuel-economy-data-1984-2021.
https://www.fueleconomy.gov/feg/ws/index.shtml#vehicle contains details about all the datatypes mentioned in the dataset.

b. mention cleaning/transformation/aggregation methods used if any

This deals with detecting and removing missing data and inconsistencies from data in order to improve the quality of data.

First the normal dataset:

```
In [23]: df=pd.read_csv("C:/Users/Pooja/OneDrive/Desktop/fuel.csv")
         df
```

Out[23]:

lodel	barrels08	barrelsA08	charge120	charge240	city08	city08U	cityA08	...	mfrCode	c240Dscr	charge240b	c240bDscr	createdOn	modifiedOn	startStop
ickup 2500 4WD	23.543571	0	0	0	13	0	0	...	NaN	NaN	0	NaN	01-01-2013	01-01-2013	NaN
ickup 2500 4WD	23.886563	0	0	0	14	0	0	...	NaN	NaN	0	NaN	01-01-2013	01-01-2013	NaN
S10 ickup 4WD	19.388824	0	0	0	15	0	0	...	NaN	NaN	0	NaN	01-01-2013	01-01-2013	NaN
Truck 4WD	21.974000	0	0	0	14	0	0	...	NaN	NaN	0	NaN	01-01-2013	01-01-2013	NaN
akota ickup 4WD	25.354615	0	0	0	11	0	0	...	NaN	NaN	0	NaN	01-01-2013	01-01-2013	NaN
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
racer	12.677308	0	0	0	22	0	0	...	NaN	NaN	0	NaN	01-01-2013	01-01-2013	NaN
D 2.5 Turbo	15.287400	0	0	0	23	0	0	...	NaN	NaN	0	NaN	01-01-2013	01-01-2013	NaN
iante	18.311667	0	0	0	16	0	0	...	NaN	NaN	0	NaN	01-01-2013	01-01-2013	NaN
alant	15.695714	0	0	0	19	0	0	...	NaN	NaN	0	NaN	01-01-2013	01-01-2013	NaN
F250 ickup 4WD	25.354615	0	0	0	12	0	0	...	NaN	NaN	0	NaN	01-01-2013	01-01-2013	NaN

Fig 2.b.1 The description of dataset

```
In [24]: print(df)
         print("================================================")
         print(df.info())
```

```
       Year   Manufacturer             Model  barrels08  barrelsA08  \
0      1995      Chevrolet    Pickup 2500 4WD  23.543571           0
1      1995      Chevrolet    Pickup 2500 4WD  23.886563           0
2      1995      Chevrolet     S10 Pickup 4WD  19.388824           0
3      1995         Nissan          Truck 4WD  21.974000           0
4      1995          Dodge  Dakota Pickup 4WD  25.354615           0
...     ...            ...                ...        ...         ...
44182  1992        Mercury            Tracer  12.677308           0
44183  1992  Mercedes-Benz     300D 2.5 Turbo  15.287400           0
44184  1992     Mitsubishi           Diamante  18.311667           0
44185  1992     Mitsubishi             Galant  15.695714           0
44186  1985           Ford    F250 Pickup 4WD  25.354615           0

       charge120  charge240  city08  city08U  cityA08  ...  mfrCode  c240Dscr  \
0              0          0      13        0        0  ...      NaN       NaN
1              0          0      14        0        0  ...      NaN       NaN
2              0          0      15        0        0  ...      NaN       NaN
3              0          0      14        0        0  ...      NaN       NaN
4              0          0      11        0        0  ...      NaN       NaN
...          ...        ...     ...      ...      ...  ...      ...       ...
44182          0          0      22        0        0  ...      NaN       NaN
44183          0          0      23        0        0  ...      NaN       NaN
44184          0          0      16        0        0  ...      NaN       NaN
44185          0          0      19        0        0  ...      NaN       NaN
44186          0          0      12        0        0  ...      NaN       NaN

       charge240b  c240bDscr   createdOn  modifiedOn  startStop  phevCity  \
0               0        NaN  01-01-2013  01-01-2013        NaN         0
1               0        NaN  01-01-2013  01-01-2013        NaN         0
2               0        NaN  01-01-2013  01-01-2013        NaN         0
3               0        NaN  01-01-2013  01-01-2013        NaN         0
4               0        NaN  01-01-2013  01-01-2013        NaN         0
...           ...        ...         ...         ...        ...       ...
44182           0        NaN  01-01-2013  01-01-2013        NaN         0
44183           0        NaN  01-01-2013  01-01-2013        NaN         0
44184           0        NaN  01-01-2013  01-01-2013        NaN         0
44185           0        NaN  01-01-2013  01-01-2013        NaN         0
44186           0        NaN  01-01-2013  01-01-2013        NaN         0

       phevHwy  phevComb
0            0         0
1            0         0
2            0         0
3            0         0
4            0         0
...        ...       ...
44182        0         0
44183        0         0
44184        0         0
44185        0         0
44186        0         0

[44187 rows x 83 columns]
```

```
==================================================
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 44187 entries, 0 to 44186
Data columns (total 83 columns):
 #   Column           Non-Null Count  Dtype
---  ------           --------------  -----
 0   Year             44187 non-null  int64
 1   Manufacturer     44187 non-null  object
 2   Model            44187 non-null  object
 3   barrels08        44187 non-null  float64
 4   barrelsA08       44187 non-null  int64
 5   charge120        44187 non-null  int64
 6   charge240        44187 non-null  int64
 7   city08           44187 non-null  int64
 8   city08U          44187 non-null  int64
 9   cityA08          44187 non-null  int64
 10  cityA08U         44187 non-null  int64
 11  cityCD           44187 non-null  int64
 12  cityE            44187 non-null  int64
 13  cityUF           44187 non-null  int64
 14  co2              44187 non-null  int64
 15  co2A             44187 non-null  int64
 16  co2TailpipeAGpm  44187 non-null  int64
 17  co2TailpipeGpm   44187 non-null  float64
 18  comb08           44187 non-null  int64
 19  comb08U          44187 non-null  int64
 20  combA08          44187 non-null  int64
 21  combA08U         44187 non-null  int64
 22  combE            44187 non-null  int64
 23  combinedCD       44187 non-null  int64
 24  combinedUF       44187 non-null  int64
 25  cylinders        43882 non-null  float64
 26  displ            43884 non-null  float64
 27  drive            43001 non-null  object
 28  engId            44187 non-null  int64
 29  eng_dscr         27549 non-null  object
 30  feScore          44187 non-null  int64
 31  fuelCost08       44187 non-null  int64
 32  fuelCostA08      44187 non-null  int64
 33  fuelType         44187 non-null  object
 34  fuelType1        44187 non-null  object
 35  ghgScore         44187 non-null  int64
 36  ghgScoreA        44187 non-null  int64
 37  highway08        44187 non-null  int64
 38  highway08U       44187 non-null  int64
 39  highwayA08       44187 non-null  int64
 40  highwayA08U      44187 non-null  int64
 41  VClass           44187 non-null  object
 42  highwayCD        44187 non-null  int64
 43  highwayE         44187 non-null  int64
 44  highwayUF        44187 non-null  int64
 45  hlv              44187 non-null  int64
 46  hpv              44187 non-null  int64
 47  id               44187 non-null  int64
 48  lv2              44187 non-null  int64
 49  lv4              44187 non-null  int64
 50  mpgData          44135 non-null  object
 51  phevBlended      44187 non-null  bool
 52  pv2              44187 non-null  int64
 53  pv4              44187 non-null  int64
 54  range            44187 non-null  int64
 55  rangeCity        44187 non-null  int64
 56  rangeCityA       44187 non-null  int64
 57  rangeHwy         44187 non-null  int64
 58  rangeHwyA        44187 non-null  int64
 59  trany            44176 non-null  object
 60  UCity            44187 non-null  float64
 61  UCityA           44187 non-null  int64
 62  UHighway         44187 non-null  float64
 63  UHighwayA        44187 non-null  int64
 64  youSaveSpend     44187 non-null  int64
 65  guzzler          2605 non-null   object
 66  trans_dscr       15044 non-null  object
 67  tCharger         8587 non-null   object
 68  sCharger         952 non-null    object
 69  atvType          4166 non-null   object
 70  fuelType2        1766 non-null   object
 71  rangeA           1761 non-null   object
 72  evMotor          1372 non-null   object
 73  mfrCode          13379 non-null  object
 74  c240Dscr         118 non-null    object
 75  charge240b       44187 non-null  int64
 76  c240bDscr        112 non-null    object
 77  createdOn        44187 non-null  object
 78  modifiedOn       44187 non-null  object
 79  startStop        12498 non-null  object
 80  phevCity         44187 non-null  int64
 81  phevHwy          44187 non-null  int64
 82  phevComb         44187 non-null  int64
dtypes: bool(1), float64(6), int64(53), object(23)
memory usage: 27.7+ MB
None
```

Fig 2.b.2 The datatypes of all columns in Dataset

```
df.describe()
```

	Year	barrels08	barrelsA08	charge120	charge240	city08	city08U	cityA08	cityA08U	cityCD	...	rangeH
count	44187.000000	44187.000000	44187.000000	44187.0	44187.000000	44187.000000	44187.000000	44187.000000	44187.000000	44187.000000	...	44187.0(
mean	2002.872519	17.085386	0.199833	0.0	0.071741	18.755018	7.065472	0.814946	0.667097	0.000249	...	0.1⁴
std	11.822723	4.699161	1.036795	0.0	0.773559	9.031354	12.806830	6.281619	6.137284	0.033973	...	2.37
min	1984.000000	0.060000	0.000000	0.0	0.000000	6.000000	0.000000	0.000000	0.000000	0.000000	...	0.0(
25%	1992.000000	14.330870	0.000000	0.0	0.000000	15.000000	0.000000	0.000000	0.000000	0.000000	...	0.0(
50%	2004.000000	16.480500	0.000000	0.0	0.000000	17.000000	0.000000	0.000000	0.000000	0.000000	...	0.0(
75%	2013.000000	19.388824	0.000000	0.0	0.000000	21.000000	16.000000	0.000000	0.000000	0.000000	...	0.0(
max	2022.000000	47.087143	18.000000	0.0	15.000000	150.000000	150.000000	145.000000	145.000000	5.000000	...	114.0(

Fig 2.b.3 Description of dataset

```
df.isnull()
```

	Year	Manufacturer	Model	barrels08	barrelsA08	charge120	charge240	city08	city08U	cityA08	...	mfrCode	c240Dscr	charge240b	c240bDscr	crea
0	False	False	False	False	False	False	False	False	False	False	...	True	True	False	True	
1	False	False	False	False	False	False	False	False	False	False	...	True	True	False	True	
2	False	False	False	False	False	False	False	False	False	False	...	True	True	False	True	
3	False	False	False	False	False	False	False	False	False	False	...	True	True	False	True	
4	False	False	False	False	False	False	False	False	False	False	...	True	True	False	True	
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	
44182	False	False	False	False	False	False	False	False	False	False	...	True	True	False	True	
44183	False	False	False	False	False	False	False	False	False	False	...	True	True	False	True	
44184	False	False	False	False	False	False	False	False	False	False	...	True	True	False	True	
44185	False	False	False	False	False	False	False	False	False	False	...	True	True	False	True	
44186	False	False	False	False	False	False	False	False	False	False	...	True	True	False	True	

44187 rows × 83 columns

Fig 2.b.4 Finding if dataset has missing values

```
# get the number of missing data points per column
missing_values_count = df.isnull().sum()

# look at the # of missing points in the first ten columns
missing_values_count[0:83]
```

```
Year                0
Manufacturer        0
Model               0
barrels08           0
barrelsA08          0
                   ...
modifiedOn          0
startStop       31689
phevCity            0
phevHwy             0
phevComb            0
Length: 83, dtype: int64
```

```
# how many total missing values do we have?
total_cells = np.product(df.shape)
total_missing = missing_values_count.sum()

# percent of data that is missing
(total_missing/total_cells) * 100
```

```
13.261791820687598
```

Fig 2.b.5 Missing data per column & total percentage of data missing

```
def list_and_visualize_missing_data(dataset):
    # Listing total null items and its percent with respect to all nulls
    total = dataset.isnull().sum().sort_values(ascending=False)
    percent = ((dataset.isnull().sum())/(dataset.isnull().count())).sort_values(ascending=False)
    missing_data = pd.concat([total, percent], axis=1, keys=['Total', 'Percent'])
    missing_data = missing_data[missing_data.Total > 0]

    missing_data.plot.bar(subplots=True, figsize=(16,9))

list_and_visualize_missing_data(df)
```

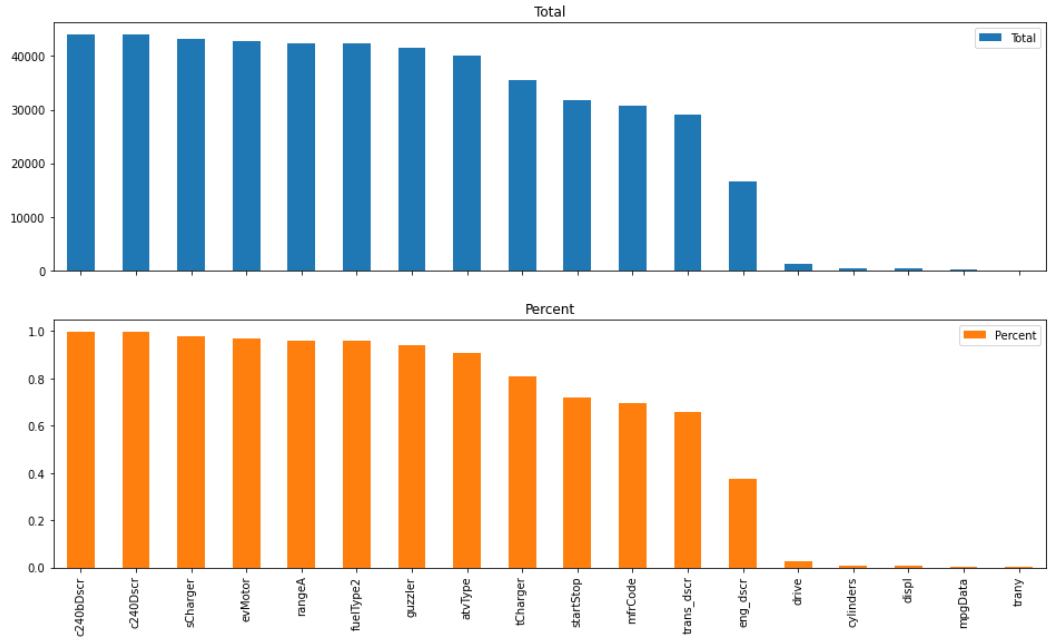


Fig 2.b.6 Visualizing the missing data

In [63]:
```
import seaborn as sns

import matplotlib.pyplot as plt
import matplotlib.mlab as mlab
import matplotlib
cols = df.columns[:30] # first 30 columns
colours = ['#000099', '#ffff00'] # specify the colours - yellow is missing. blue is not missing.
sns.heatmap(df[cols].isnull(), cmap=sns.color_palette(colours))
```

Out[63]: <AxesSubplot:>

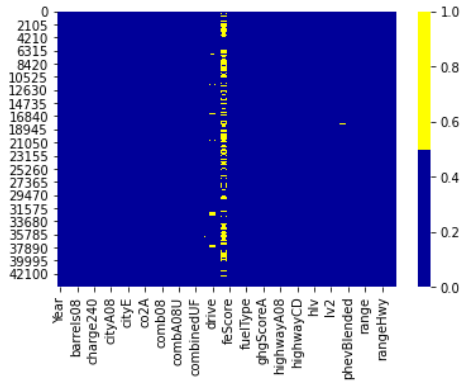


Fig 2.b.7 Heatmap visualization of missing data

```
# % of missing.
for col in df.columns:
    pct_missing = np.mean(df[col].isnull())
    print('{} - {}%'.format(col, round(pct_missing*100)))
```

```
Year - 0%
Manufacturer - 0%
Model - 0%
barrels08 - 0%
barrelsA08 - 0%
charge120 - 0%
charge240 - 0%
city08 - 0%
city08U - 0%
cityA08 - 0%
cityA08U - 0%
cityCD - 0%
cityE - 0%
cityUF - 0%
co2 - 0%
co2A - 0%
co2TailpipeAGpm - 0%
co2TailpipeGpm - 0%
comb08 - 0%
comb08U - 0%
combA08 - 0%
combA08U - 0%
combE - 0%
combinedCD - 0%
combinedUF - 0%
cylinders - 1%
displ - 1%
drive - 3%
engId - 0%
eng_dscr - 38%
feScore - 0%
fuelCost08 - 0%
fuelCostA08 - 0%
fuelType - 0%
fuelType1 - 0%
ghgScore - 0%
ghgScoreA - 0%
highway08 - 0%
highway08U - 0%
highwayA08 - 0%
highwayA08U - 0%
VClass - 0%
highwayCD - 0%
highwayE - 0%
highwayUF - 0%
hlv - 0%
hpv - 0%
id - 0%
lv2 - 0%
lv4 - 0%
mpgData - 0%
phevBlended - 0%
pv2 - 0%
pv4 - 0%
range - 0%
rangeCity - 0%
rangeCityA - 0%
rangeHwy - 0%
rangeHwyA - 0%
trany - 0%
UCity - 0%
UCityA - 0%
UHighway - 0%
UHighwayA - 0%
youSaveSpend - 0%
guzzler - 94%
trans_dscr - 66%
tCharger - 81%
sCharger - 98%
atvType - 91%
fuelType2 - 96%
rangeA - 96%
evMotor - 97%
mfrCode - 70%
c240Dscr - 100%
charge240b - 0%
c240bDscr - 100%
createdOn - 0%
modifiedOn - 0%
startStop - 72%
phevCity - 0%
phevHwy - 0%
phevComb - 0%
```

Fig 2.b.8 missing % per column

```
df.fillna(0)
```

	Year	Manufacturer	Model	barrels08	barrelsA08	charge120	charge240	city08	city08U	cityA08	...	mfrCode	c240Dscr	charge240b	c240bDscr	c
0	1995	Chevrolet	Pickup 2500 4WD	23.543571	0	0	0	13	0	0	...	0	0	0	0	
1	1995	Chevrolet	Pickup 2500 4WD	23.886563	0	0	0	14	0	0	...	0	0	0	0	
2	1995	Chevrolet	S10 Pickup 4WD	19.388824	0	0	0	15	0	0	...	0	0	0	0	
3	1995	Nissan	Truck 4WD	21.974000	0	0	0	14	0	0	...	0	0	0	0	
4	1995	Dodge	Dakota Pickup 4WD	25.354615	0	0	0	11	0	0	...	0	0	0	0	
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	
44182	1992	Mercury	Tracer	12.677308	0	0	0	22	0	0	...	0	0	0	0	
44183	1992	Mercedes-Benz	300D 2.5 Turbo	15.287400	0	0	0	23	0	0	...	0	0	0	0	
44184	1992	Mitsubishi	Diamante	18.311667	0	0	0	16	0	0	...	0	0	0	0	
44185	1992	Mitsubishi	Galant	15.695714	0	0	0	19	0	0	...	0	0	0	0	
44186	1985	Ford	F250 Pickup 4WD	25.354615	0	0	0	12	0	0	...	0	0	0	0	

44187 rows × 83 columns

Fig 2.b.9 Replacing the missing and NaN data with 0

```
# writing to Excel
datatoexcel = pd.ExcelWriter('C:/Users/Pooja/OneDrive/Desktop/fuelData2.xlsx')

# write DataFrame to excel
dg.to_excel(datatoexcel)

# save the excel
datatoexcel.save()
print('DataFrame is written to Excel File successfully.')
```

```
DataFrame is written to Excel File successfully.
```

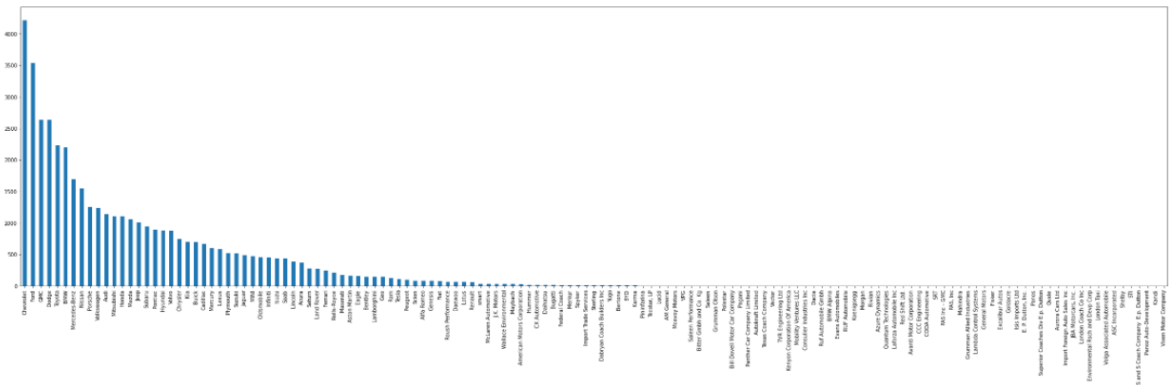
Fig 2.b.10 Downloading the modified dataset

For Understanding to implement the data visualization done is:

```
print("No of unique manufacturers for :",len(df["Manufacturer"].unique()))
plt.figure(figsize=(40,10))
df["Manufacturer"].value_counts().plot(kind="bar")
```

```
No of unique manufacturers for : 141

<AxesSubplot:>
```



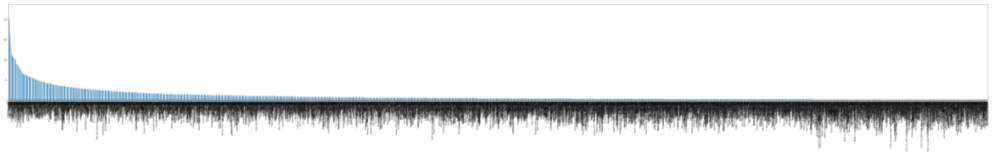
```
print(df["Model"].value_counts())
print("No of unique models:",len(df["Model"].unique()))
```

```
F150 Pickup 2WD                 227
F150 Pickup 4WD                 207
Camaro                          206
Mustang                         203
Jetta                           196
                               ...
Huracan Spyder RWD                1
E500 4matic (wagon)               1
ITS 230CE                         1
Transit Connect Electric Van      1
Silver Spur II Mpw Touring L      1
Name: Model, Length: 4521, dtype: int64
No of unique models: 4521
```

```
print("No of unique models for :",len(df["Model"].unique()))
plt.figure(figsize=(100,10))
df["Model"].value_counts().plot(kind="bar")
```

```
No of unique models for : 4521

<AxesSubplot:>
```



3 Model planning:

a. Elaborate write up of each Module with necessary diagrams

MapReduce:

MapReduce is a programming paradigm that runs in the Hadoop background to provide scalability and easy data processing solutions. The Map task takes one record and converts it to another record, dividing the individual elements into tuples (key-value pairs). The reducer task takes the mapper's output as input and combines these data tuples (key-value pairs) into a smaller set of tuples.

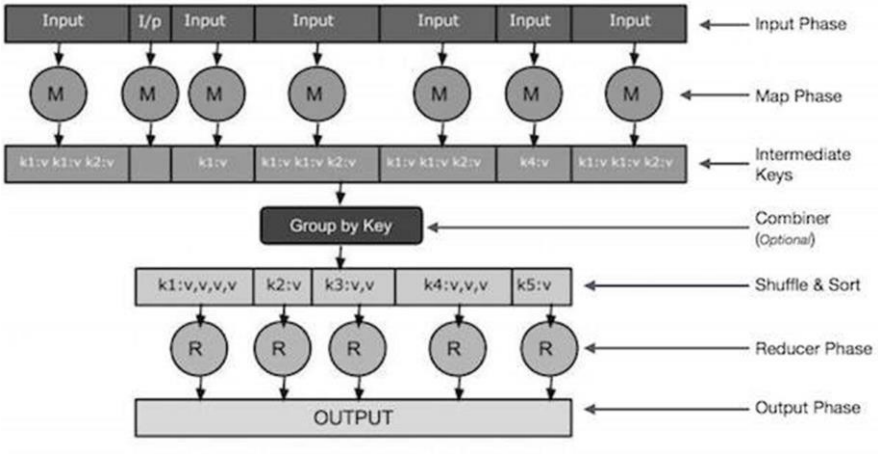


Fig 3.a.1 Working of MapReduce Framework

Modules in the Model:

Input Phase:

The input reader reads the incoming data and divides it into appropriate-sized data blocks. A Map function is assigned to each data block.

Map:

The map function processed the next key-value pairs and generated the corresponding output key-value pairs. The map input and output types may differ from one another.

Map is a user-defined function that takes a set of key-value pairs and processes each one to produce zero or more key-value pairs.

Intermediate Keys:

The key-value pairs produced by the mapper are referred to as intermediate keys.

Combiner:

A combiner is a type of local Reducer that groups similar data from the map phase into identifiable sets. It takes the intermediate keys from the mapper as input and applies a user-defined code to aggregate the values in a small scope of one mapper. It is not a part of the main MapReduce algorithm; it is optional.

Shuffling and Sorting

The data are shuffled between/within nodes so that it moves out from the map and get ready to process for reduce function. Sometimes, the shuffling of data can take much computation time.

The Reducer task starts with the Shuffle and Sort step. It downloads the grouped key-value pairs onto the local machine, where the Reducer is running. The individual key-value pairs are sorted by key into a larger data list. The data list groups the equivalent keys together so that their values can be iterated easily in the Reducer task.

The sorting operation is performed on input data for Reduce function. Here, the data is compared using comparison function and arranged in a sorted form.

Reduce function

The Reduce function is assigned to each unique key. These keys are already arranged in sorted order. The values associated with the keys can iterate the Reduce and generates the corresponding output.

Output writer

Once the data flow from all the above phases, Output writer executes. The role of Output writer is to write the Reduce output to the stable storage. we have an output formatter that translates the final key-value pairs from the Reducer function and writes them onto a file using a record writer.

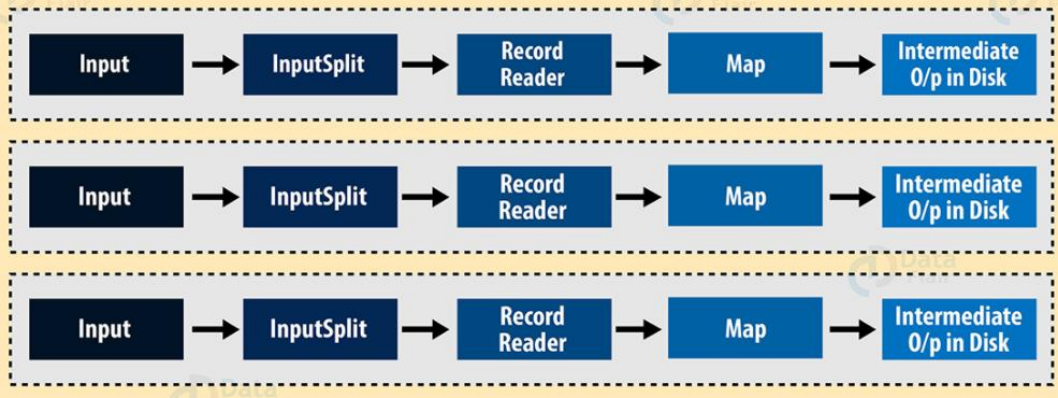
b. Explanation of the Model applied with necessary diagram

MapReduce is a processing technique and a program model for distributed computing based on java. The MapReduce algorithm contains two important tasks, namely Map and Reduce. Map takes a set of data and converts it into another set of data, where individual elements are broken down into tuples (key/value pairs). Secondly, reduce task, which takes the output from

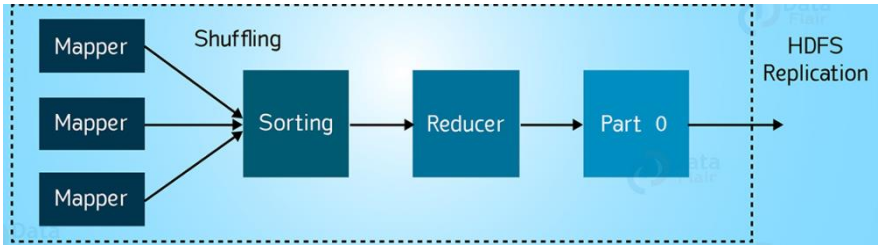
a map as an input and combines those data tuples into a smaller set of tuples. As the sequence of the name MapReduce implies, the reduce task is always performed after the map job.

Model Applied is Map Reduce job. Has three main parts Mapper, Reducer

Mapper:

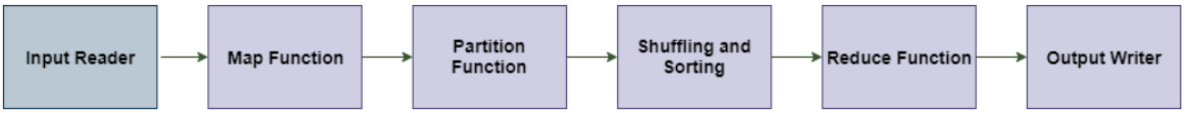


Reducer:



For each problem we write a Java program that has a mapper to map values and form key pair values, a reducer class where we reduce the values as per the question and a main class to run the program.

The entire MapReduce flow is as shown below:



We are planning to write a MapReduce program in java using Hadoop. We will try to implement MapReduce program to analyze these areas

1. To Display Top 5 Manufacturers and their corresponding count of models
   We have almost 141 unique manufacturers and 4521 unique models from this data we find the top 5 manufacturers and count of models they produce.

2. To find the average annual petroleum consumption in barrels for each manufacturer of vehicle.

Thus, we get to know which car takes less petroleum. Among all the vehicles we find annual petroleum consumption and print them in descending order. We take annual petroleum consumption in barrels and Manufacturer columns.

3. to display the average city MPG (Miles Per Gallon) for fuelType1(For single fuel vehicles).

   We find the average of city MPG using city MPG column of dataset.