Used Cars Selling Price Report

To begin with, we searched the internet for a data set to work on and we found the following data set from a website called kaggle (a brief part from the data set):

year	make	model	trim	body	transmissi	vin	state	condition	odometer	color	interior	seller	mmr	sellingpric	saledate
2015	Kia	Sorento	LX	SUV	automatic	5xyktca69	ca	5	16639	white	black	kia motors	20500	21500	Tue Dec 16 2014 12:30:00 GMT-0800 (PST)
2015	Kia	Sorento	LX	SUV	automatic	5xyktca69	ca	5	9393	white	beige	kia motors	20800	21500	Tue Dec 16 2014 12:30:00 GMT-0800 (PST)
2014	BMW	3 Series	328i SULE	Sedan	automatic	wba3c1c5	ca	4.5	1331	gray	black	financial s	31900	30000	Thu Jan 15 2015 04:30:00 GMT-0800 (PST)
2015	Volvo	S60	T5	Sedan	automatic	yv1612tb4	ca	4.1	14282	white	black	volvo na r	27500	27750	Thu Jan 29 2015 04:30:00 GMT-0800 (PST)
2014	BMW	6 Series G	650i	Sedan	automatic	wba6b2c5	ca	4.3	2641	gray	black	financial s	66000	67000	Thu Dec 18 2014 12:30:00 GMT-0800 (PST)
2015	Nissan	Altima	2.5 S	Sedan	automatic	1n4al3ap1	ca	1	5554	gray	black	enterprise	15350	10900	Tue Dec 30 2014 12:00:00 GMT-0800 (PST)
2014	BMW	M5	Base	Sedan	automatic	wbsfv9c51	ca	3.4	14943	black	black	the hertz c	69000	65000	Wed Dec 17 2014 12:30:00 GMT-0800 (PST)
2014	Chevrolet	Cruze	1LT	Sedan	automatic	1g1pc5sb2	ca	2	28617	black	black	enterprise	11900	9800	Tue Dec 16 2014 13:00:00 GMT-0800 (PST)
2014	Audi	A4	2.0T Prem	Sedan	automatic	wauffafl3	ca	4.2	9557	white	black	audi missi	32100	32250	Thu Dec 18 2014 12:00:00 GMT-0800 (PST)
2014	Chevrolet	Camaro	LT	Convertib	automatic	2g1fb3d37	ca	3	4809	red	black	d/m auto	26300	17500	Tue Jan 20 2015 04:00:00 GMT-0800 (PST)
2014	Audi	A6	3.0T Prest	Sedan	automatic	wauhgafc	ca	4.8	14414	black	black	desert aut	47300	49750	Tue Dec 16 2014 12:30:00 GMT-0800 (PST)
2015	Kia	Optima	LX	Sedan	automatic	5xxgm4a7	ca	4.8	2034	red	tan	kia motors	15150	17700	Tue Dec 16 2014 12:00:00 GMT-0800 (PST)
2015	Ford	Fusion	SE	Sedan	automatic	3fa6p0hd	ca	2	5559	white	beige	enterprise	15350	12000	Tue Jan 13 2015 12:00:00 GMT-0800 (PST)
2015	Kia	Sorento	LX	SUV	automatic	5xyktca66	ca	5	14634	silver	black	kia motors	20600	21500	Tue Dec 16 2014 12:30:00 GMT-0800 (PST)
2014	Chevrolet	Cruze	2LT	Sedan	automatic	1g1pe5sb	ca		15686	blue	black	avis rac/sa	13900	10600	Tue Dec 16 2014 12:00:00 GMT-0800 (PST)
2015	Nissan	Altima	2.5 S	Sedan	automatic	1n4al3ap5	ca	2	11398	black	black	enterprise	14750	14100	Tue Dec 23 2014 12:00:00 GMT-0800 (PST)
2015	Hyundai	Sonata	SE	Sedan		5npe24af4	ca		8311	red	â€"	avis tra	15200	4200	Tue Dec 16 2014 13:00:00 GMT-0800 (PST)
2014	Audi	Q5	2.0T Prem	SUV	automatic	wa1lfafpx	ca	4.9	7983	white	black	audi north	37100	40000	Thu Dec 18 2014 12:30:00 GMT-0800 (PST)
2014	Chevrolet	Camaro	LS	Coupe	automatic	2g1fa1e39	ca	1.7	13441	black	black	wells farg	17750	17000	Tue Dec 30 2014 15:00:00 GMT-0800 (PST)
2014	BMW	6 Series	650i	Convertib	automatic	wbayp9c5	ca	3.4	8819	black	black	the hertz c	68000	67200	Wed Dec 17 2014 12:30:00 GMT-0800 (PST)

Through working on this current data set, there was a lot going on and had to be done.

Firstly, the data set had multiple misplaced data, empty records, pointless data, etc.

The data set was about 55k rows and 16 columns.

At first, the data looked ugly and hideous. After cleaning the data, we were able to put the data set to work as soon as possible since it was usable once again.

Data Processing using Python:

To begin with, these are the libraries we needed to import so that we can start working on our data set.

```
import numpy as np
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt
%matplotlib inline
from  datasist.structdata import detect_outliers
from sklearn.datasets import fetch_openml
from sklearn.linear_model import LinearRegression
from sklearn.model_selection import train_test_split
from sklearn.feature_extraction.text import CountVectorizer
from sklearn.naive_bayes import BernoulliNB
from sklearn.impute import SimpleImputer
```

We started by

importing the libraries Numpy and Pandas in order to manipulate the records inside the data set, Seaborn and Matplotlib.pyplot for visualization and graphing, and finally, Sklearn for machine learning.

We then loaded the data set and set the low_memory to false since we are dealing with a massive data set.

Since we're dealing with a massive data set, we needed a general idea about the datatypes used in the dataset, therefore, calling the function .info() was a good example of displaying what we needed.

After getting a glimpse of what we're dealing with, we found out that the "condition" and "mmr" columns are in the wrong datatypes.

So, we used the pandas command "To_Numeric" to change their datatype from object to float64 and int64.

```
car.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 558837 entries, 0 to 558836
Data columns (total 17 columns):
 #   Column        Non-Null Count   Dtype
---  ------        --------------   -----
 0   year          558837 non-null  int64
 1   make          548536 non-null  object
 2   model         548438 non-null  object
 3   trim          548186 non-null  object
 4   body          545642 non-null  object
 5   transmission  493484 non-null  object
 6   vin           558833 non-null  object
 7   state         558837 non-null  object
 8   condition     547043 non-null  object
 9   odometer      558743 non-null  float64
 10  color         558088 non-null  object
 11  interior      558088 non-null  object
 12  seller        558837 non-null  object
 13  mmr           558837 non-null  object
 14  sellingprice  558837 non-null  int64
 15  saledate      558837 non-null  object
 16  Unnamed: 16   26 non-null      object
dtypes: float64(1), int64(2), object(14)
memory usage: 72.5+ MB
```

Now that the dataset had the correct datatypes, we needed to check how many null values are there in each column. So, we used the ".isnull().sum()" and this was the outcome of the command:

We were overwhelmed by the number of empty cells. Work had to be done.

```
car.isnull().sum()
```

```
year              0
make          10301
model         10399
trim          10651
body          13195
transmission  65353
vin               4
state             0
condition     11794
odometer         94
color           749
interior        749
seller            0
mmr               0
sellingprice      0
saledate          0
Unnamed: 16  558811
dtype: int64
```

Data Cleaning

Initially, we had to get rid of "unnamed: 16" since it only held null cells only, "saledate" since we had no use for the date of sold cars, and "vin" since it held unique ids for every car in the data set, so we had no use for it.

Then we proceeded to use ".head()" command to check how the data set looked like so far and the following was the result:

	year	make	model	trim	body	transmission	state	condition	odometer	color	interior	seller	mmr	sellingprice
0	2015	Kia	Sorento	LX	SUV	automatic	ca	5	16639.0	white	black	kia motors america, inc	20500	21500
1	2015	Kia	Sorento	LX	SUV	automatic	ca	5	9393.0	white	beige	kia motors america, inc	20800	21500
2	2014	BMW	3 Series	328i SULEV	Sedan	automatic	ca	4.5	1331.0	gray	black	financial services remarketing (lease)	31900	30000
3	2015	Volvo	S60	T5	Sedan	automatic	ca	4.1	14282.0	white	black	volvo na rep/world omni	27500	27750
4	2014	BMW	6 Series Gran Coupe	650i	Sedan	automatic	ca	4.3	2641.0	gray	black	financial services remarketing (lease)	66000	67000

We fixed a part of the null values and unused data in the data set; however, we still have a vast number of null values to deal with.

We created a variable to store the mode of the “transmission” column so we can fill in the null values of that column with the mode of the "transmission” column.

Then we proceeded to do the same process with the “condition” and “mmr” columns except we used the median of the column to fill in the null values.

The outcome of the previous processes:

```
year              0
make          10301
model         10399
trim          10651
body          13195
transmission      0
state             0
condition     11794
odometer         94
color           749
interior        749
seller            0
mmr               0
sellingprice      0
dtype: int64
```

The rest of the null values barely represented 3% of the data set and we couldn’t deal with them, so we deleted the records with null values by using “.dropna” command.

The result was delightful.

```
year            0
make            0
model           0
trim            0
body            0
transmission    0
state           0
condition       0
odometer        0
color           0
interior        0
seller          0
mmr             0
sellingprice    0
dtype: int64
```

After dealing with the null values, we noticed that there were some duplicated data since there were upper- and lower-case letters.

So first, we had to display the number of unique values in the data set, and to do that, we used the ".nunique" command:

```
year              26
make              53
model            776
trim            1525
body              87
transmission       4
state             64
condition         41
odometer      169292
color             46
interior          17
seller         13995
mmr             1101
sellingprice    1869
dtype: int64
```

In order to deal with the duplicate data, we defined a function called "lower" that takes the argument "c" (which is a column) and returns the whole column in lower-case letters. Then, we proceeded to apply this function to all columns with the data type "object".

We also noticed that the "transmission" column contained 3 unique values after applying the lower function when it should only contain 2 which are "automatic" and "manual".

So, we replaced the extra value which was "sedan" with the median of the "transmission" column. The following was the result of the lower function:

```
year              26
make              53
model            772
trim            1506
body              46
transmission       2
state             64
condition         41
odometer      169292
color             46
interior          17
seller         13995
mmr             1101
sellingprice    1869
dtype: int64
```

We also noticed that the “color” column had plenty of unique values. So, while checking the values of color, we noticed some numbers were there and strange symbols.

In order to deal with the numbers first, we used the “.apply(lambda x: str(x).isdigit())]”. The “~” symbol is to remove the numbers if the outcome of the command is true.

Then we replaced the strange symbols with the median of the column by storing the median of the column in a variable then using the “.replace” command.

We also did the same process to the interior column since it had the same strange symbols. We stored the median of the column in a variable then replaced the symbols with the variable’s content.

<u>The final result of the cleaning:</u>

```
year              26
make              53
model            772
trim            1505
body              45
transmission       2
state             38
condition         41
odometer      169283
color             19
interior          16
seller         13993
mmr             1101
sellingprice    1869
dtype: int64
```