Table of Contents

1 Problem Statement – Understand Business need.

There is a need to accurately predict the price of a car to help prospective buyers and sellers of a vehicle. Prediction should consider factors such as make/model, number of miles on the vehicle, year of manufacture, etc., and assess which feature most impacts the vehicle price.

There are two broad categories of clients for this machine learning model - Dealers and Individuals. A better understanding of data has many benefits, such as

Dealers (Used car sellers):

1. Features impacting the car price would help dealers provide better customer service.
2. Allow dealers to mark a used car's price as attractive and competitive.
3. Make informed decisions on the dealer's inventory stock to maintain to improve earnings and profits.

Individuals

1. A better understanding of the fair price of a car
2. A better understanding of the prices impacting the price of a car.
3. With a sense of features affecting the price of a Used car, an individual can make an informed decision when buying a new car.
4. An individual can offer Online Pricing services via Websites and or APIs

2 The Data - Data Understanding

This model uses the dataset available on Kaggle and contains information on 426K cars with below attributes.

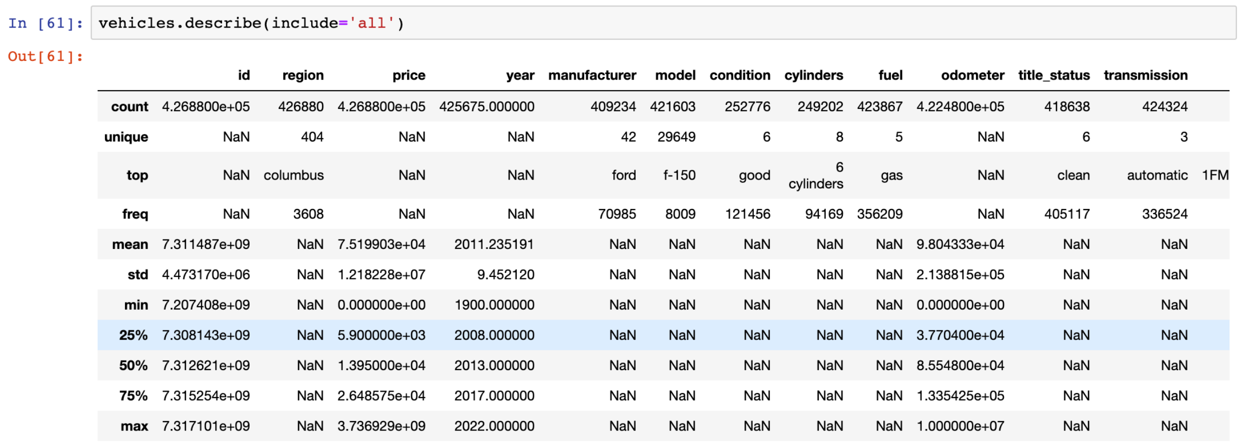
1. price
2. year
3. manufacturer
4. condition
5. cylinders
6. fuel
7. odometer
8. title_status
9. transmission
10. drive
11. size
12. type
13. paint_color
14. state
15. id
16. region
17. VIN
18. modem

2.1 Data Preparation

This section captures the data exploration, observations, and cleaning activities performed to make data useful for ML model-building activities.

2.2 Data Exploration

The data exploration and visualizations are applied to the data to understand the data better. This activity gave some ideas on dealing with "missing" values and data outliers.



```
In [61]: vehicles.describe(include='all')
```

Out[61]:

	id	region	price	year	manufacturer	model	condition	cylinders	fuel	odometer	title_status	transmission	
count	4.268800e+05	426880	4.268800e+05	425675.000000	409234	421603	252776	249202	423867	4.224800e+05	418638	424324	
unique	NaN	404	NaN	NaN	42	29649	6	8	5	NaN	6	3	
top	NaN	columbus	NaN	NaN	ford	f-150	good	6 cylinders	gas	NaN	clean	automatic	1FM
freq	NaN	3608	NaN	NaN	70985	8009	121456	94169	356209	NaN	405117	336524	
mean	7.311487e+09	NaN	7.519903e+04	2011.235191	NaN	NaN	NaN	NaN	NaN	9.804333e+04	NaN	NaN	
std	4.473170e+06	NaN	1.218228e+07	9.452120	NaN	NaN	NaN	NaN	NaN	2.138815e+05	NaN	NaN	
min	7.207408e+09	NaN	0.000000e+00	1900.000000	NaN	NaN	NaN	NaN	NaN	0.000000e+00	NaN	NaN	
25%	7.308143e+09	NaN	5.900000e+03	2008.000000	NaN	NaN	NaN	NaN	NaN	3.770400e+04	NaN	NaN	
50%	7.312621e+09	NaN	1.395000e+04	2013.000000	NaN	NaN	NaN	NaN	NaN	8.554800e+04	NaN	NaN	
75%	7.315254e+09	NaN	2.648575e+04	2017.000000	NaN	NaN	NaN	NaN	NaN	1.335425e+05	NaN	NaN	
max	7.317101e+09	NaN	3.736929e+09	2022.000000	NaN	NaN	NaN	NaN	NaN	1.000000e+07	NaN	NaN	

2.2.1 Observations

- Id, VIN, region, and model feature columns can be dropped.
- Cars with Price = zero can be dropped, thereby eliminating 7.7% of records
- Memory footprint can be reducing by changing data of categorical columns to type "Category".
    - manufacturer
    - condition
    - cylinders
    - fuel
    - title_status
    - transmission
    - drive
    - size
    - type
    - paint_color
- Outliers can be dropped.
    - Vehicles with price greater than $100,000 and less than $700 can be dropped thereby eliminating x records.
    - Vehicles with Odometer greater than 300,000 and less than 300 can be dropped thereby eliminating x records.
    - Null point percentage (Table -1)

Column	Null %
size	71.767476
cylinders	41.62247
condition	40.785232
drive	30.586347
paint_color	30.501078
type	21.752717
manufacturer	4.133714
title_status	1.930753
odometer	1.030735
fuel	0.705819
transmission	0.598763

year	0.282281
price	0
state	0

- Unique values:
  - Unique values by "Condition". Looking at the null point Percentage (Table 1) and below table, null values in the condition table can be filled with "new" or "like new" based on the "year" attribute.
    - If year >=2019 and condition is null, then condition is "new"
    - If year >= 2017 and condition is null, then condition is "like new"

good	111207
excellent	85541
like new	18120
fair	4903
new	573
salvage	437

3 Data Cleansing

3.1.1 Drop Invalid Data

- Records with Price equal to zero are dropped from the dataset. This activity eliminated 7.7% of records from the dataset.

3.1.2 Drop Features

Id, VIN, region, and model features are not needed and are dropped from the datasets.

3.1.3 Convert Datatypes

Datatypes of below features are updated to "category" type. This reduced the data size from *58.6+* MB to 18.8+MB.

- manufacturer
- condition
- cylinders
- fuel
- title_status
- transmission
- drive
- size
- type
- paint_color

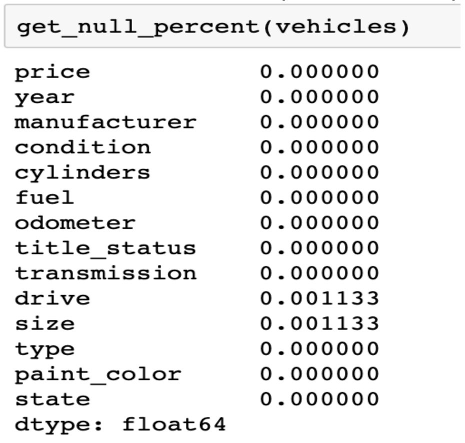
3.1.4 Drop Outliers

- Vehicles with price greater than $100,000 and less than $700 are dropped thereby eliminating 3% of records.
- Vehicles with Odometer greater than 300,000 and less than 300 are dropped thereby eliminating 2.87% of records.
- Vehicles with Year of manufacture greater than 2023 and less than 1984 are dropped thereby eliminating 2.3% of records.

3.1.5 Fill Missing Values

- Vehicles with missing “Condition” feature is updated based on the “year” attribute.
  - If year >=2019 and condition is null, then condition is “new”
  - If year >= 2017 and condition is null, then condition is “like new”
- Above step doesn’t cover all the missing “Condition” feature records. “Odometer” feature is used to fill in the missing “Condition” feature records.
- “ffill” method is used to fill in the missing records for below features.
  - manufacturer
  - cylinders
  - fuel
  - title_status
  - transmission
  - drive
  - size
  - type
  - paint_color

- Most of “null” records are updated after performing the above data cleansing



```
get_null_percent(vehicles)
```

```
price           0.000000
year            0.000000
manufacturer    0.000000
condition       0.000000
cylinders       0.000000
fuel            0.000000
odometer        0.000000
title_status    0.000000
transmission    0.000000
drive           0.001133
size            0.001133
type            0.000000
paint_color     0.000000
state           0.000000
dtype: float64
```

steps.

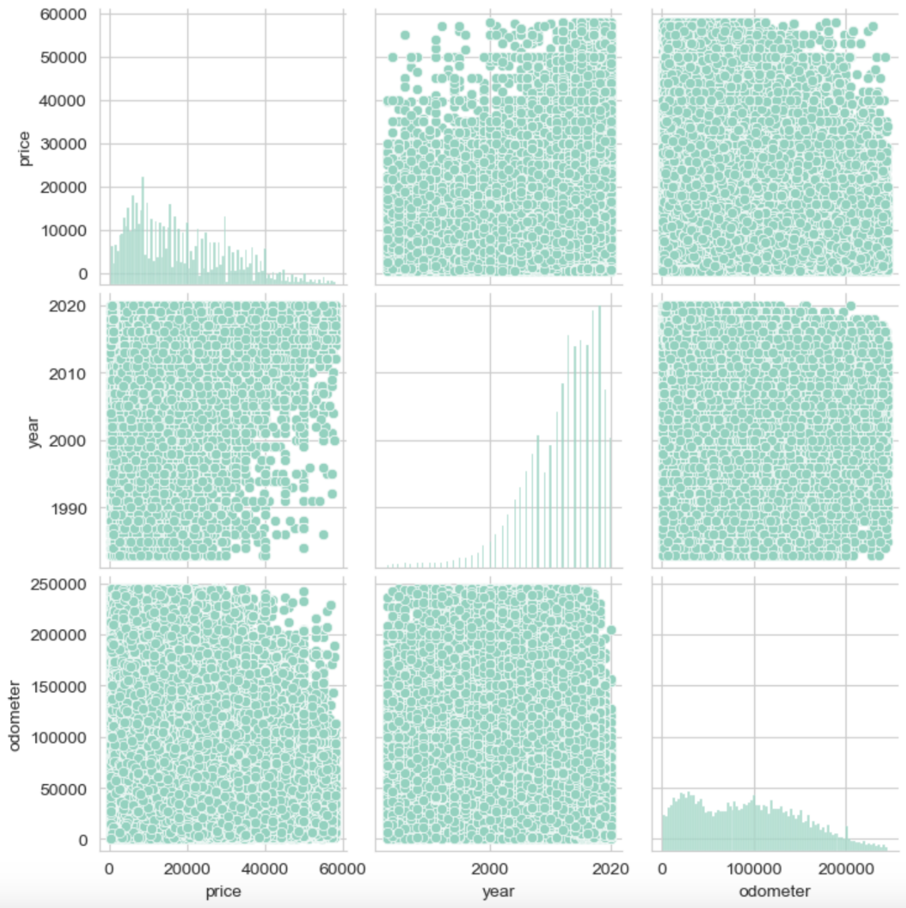
- “drive” and “size” features have 0.001 % of records (4 records). Records with “null” “drive” and “size” are dropped since the number of records are too small. Below is the output after dropping “drive” and “Size” features.

```
In [55]: get_null_percent(vehicles)

Out[55]: price           0.0
         year            0.0
         manufacturer    0.0
         condition       0.0
         cylinders       0.0
         fuel            0.0
         odometer        0.0
         title_status    0.0
         transmission    0.0
         drive           0.0
         size            0.0
         type            0.0
         paint_color     0.0
         state           0.0
         dtype: float64
```

○

4 Exploratory Data Analysis

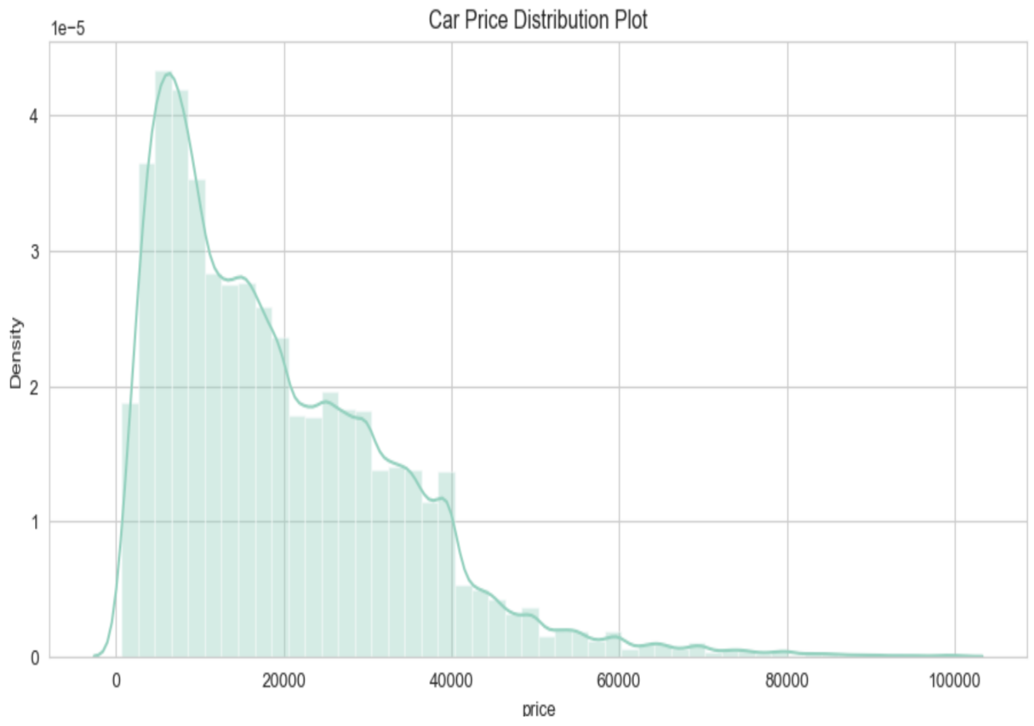
```
In [57]: sns.pairplot(vehicles)

Out[57]: <seaborn.axisgrid.PairGrid at 0x7fc7225e8760>
```



4.1.1 Car Price Distribution Plot

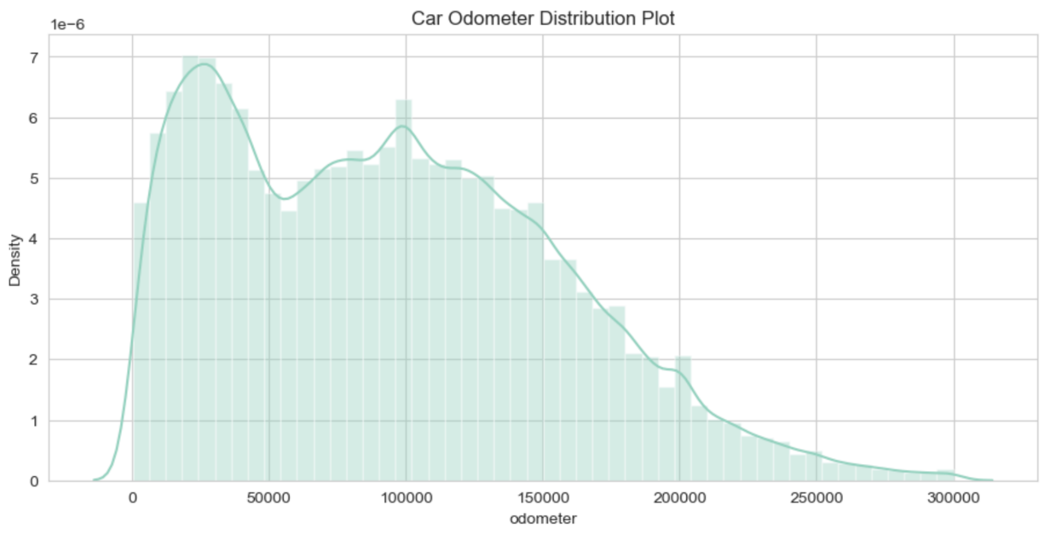
By looking at the below picture, it can be found that most of the car prices are less than 20K. There are good chunk of cars whose price range between 20K to 40K

```
Out[38]: []
```



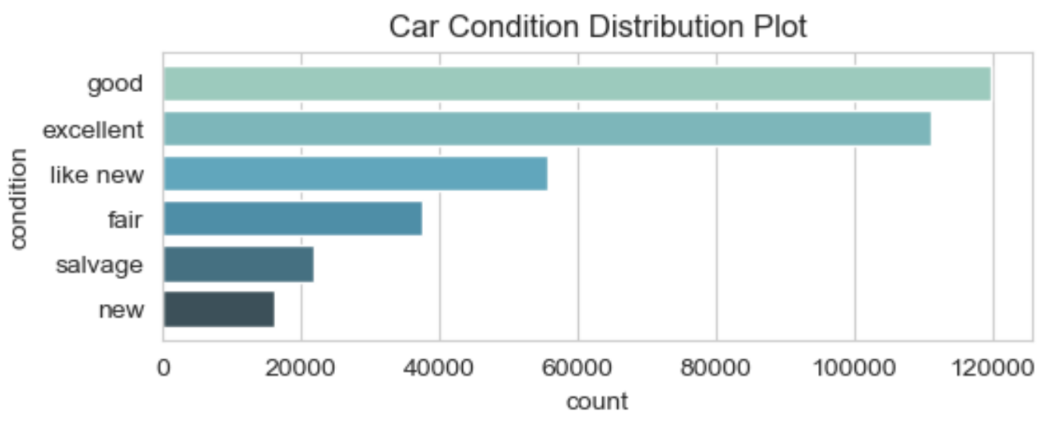
4.1.2 Car Odometer Distribution Plot

```
Out[39]: []
```

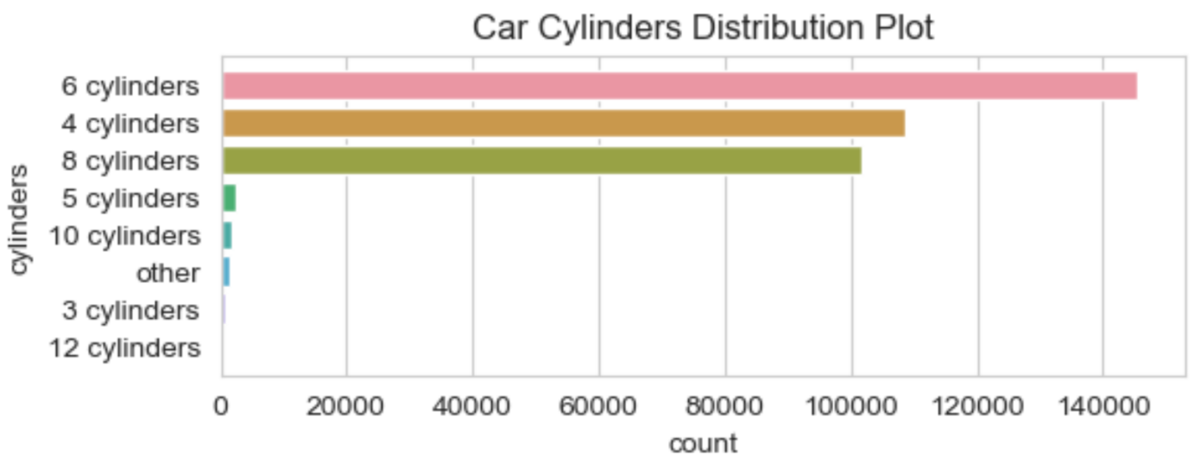


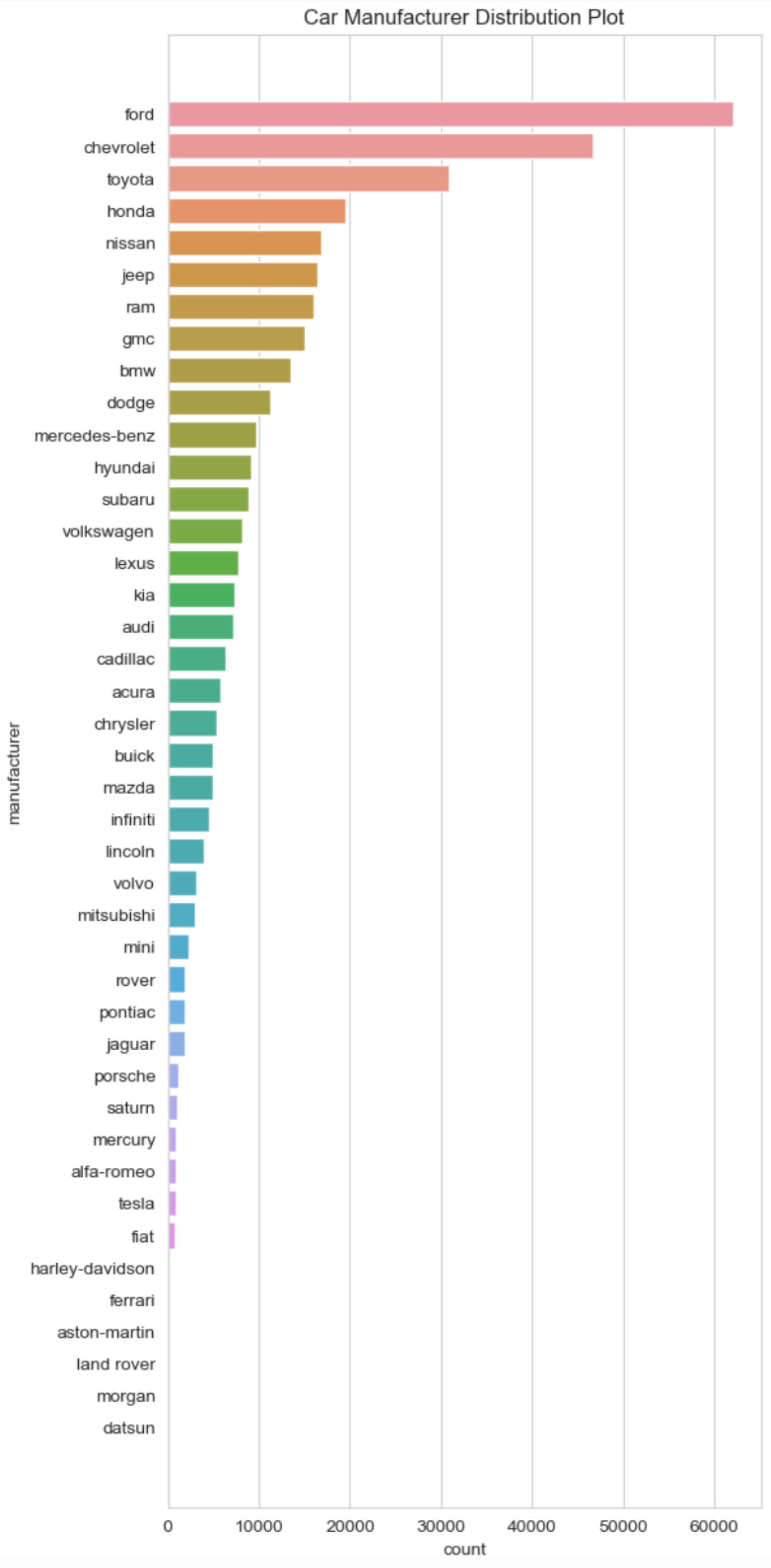
4.1.3 Car Condition Distribution Plot

```
Out[69]: []
```



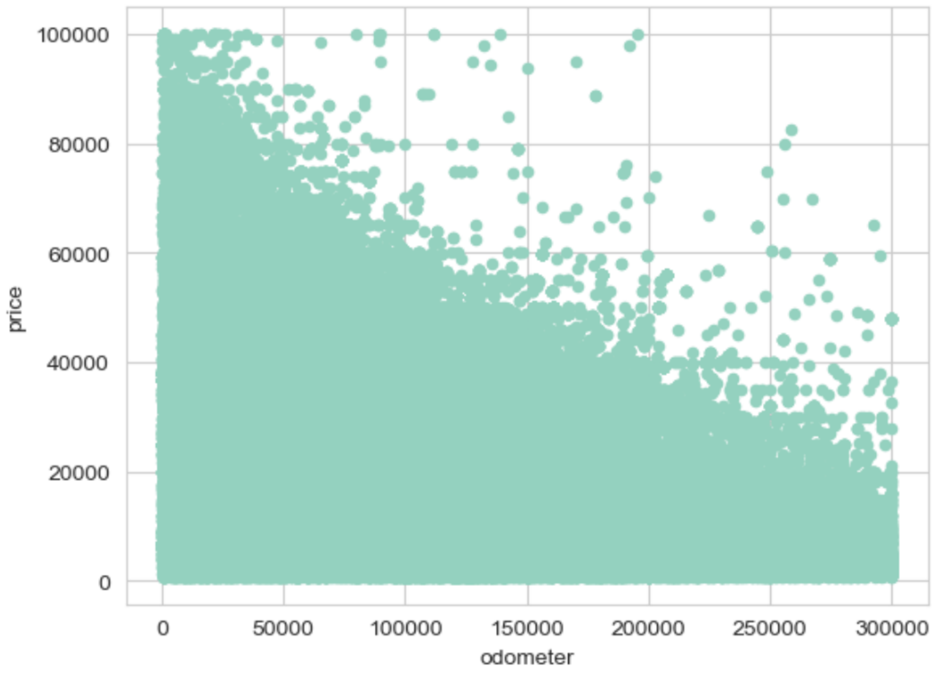
4.1.4 Car Cylinder Distribution Plot




Car Manufacturer Distribution Plot
manufacturer
ford
chevrolet
toyota
honda
nissan
jeep
ram
gmc
bmw
dodge
mercedes-benz
hyundai
subaru
volkswagen
lexus
kia
audi
cadillac
acura
chrysler
buick
mazda
infiniti
lincoln
volvo
mitsubishi
mini
rover
pontiac
jaguar
porsche
saturn
mercury
alfa-romeo
tesla
fiat
harley-davidson
ferrari
aston-martin
land rover
morgan
datsun
0
10000
20000
30000
40000
50000
60000
count

4.1.5 Price Trend vs Number of miles on the Vehicle

```
In [43]: vehicles.plot.scatter(x='odometer', y='price')
Out[43]: <AxesSubplot:xlabel='odometer', ylabel='price'>
```



Inference: Price continues to drop as the number of miles on the vehicles increases.

5 Machine Learning Models

5.1.1 Preprocessing the data

In the dataset, there are 13 predictors. 3 are numeric and rest are all of type “Category”. To apply machine learning models, the non-numeric columns need to be converted to numeric values.

5.1.2 Training datasets

Input dataset is split into 70-30. 70% of data for training and remaining 30% for testing. This resulted in 289,700 and 72,426 as training and testing datasets respectively.

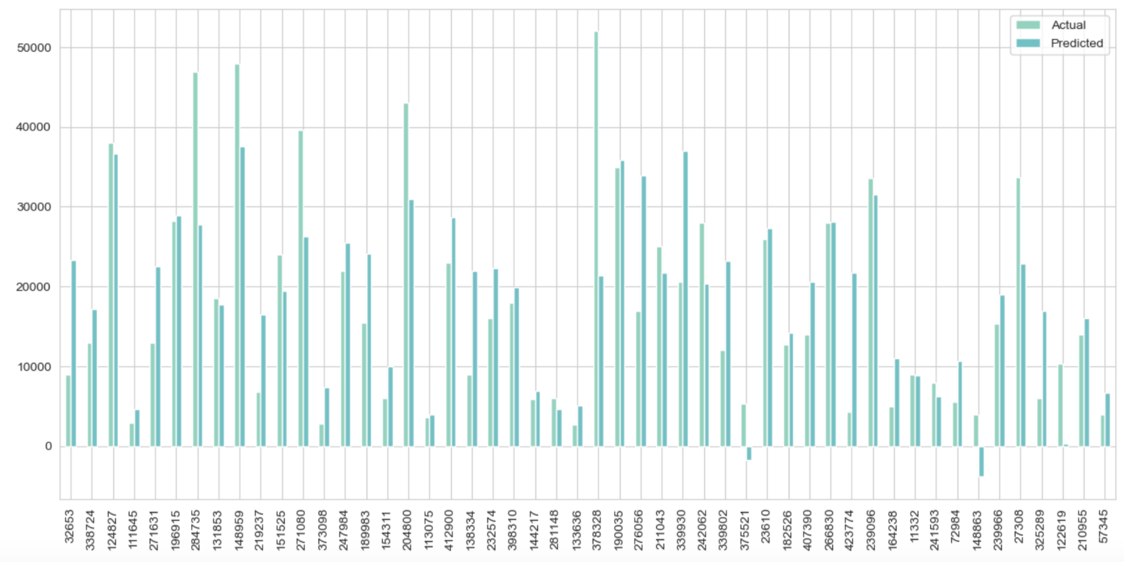
5.1.3 Linear Regression

Results from Linear regression

Linear Regression Evaluation	**Observations**
Root Mean Squared Error	9908.094105971484
Mean Squared Error	98170328.81
Mean Absolute Error	7237.44

Observation: It can be noticed that there are significant differences between actual and predicted values indicating that Linear Regression model is NOT the right model for this use case. Hence, there is a need to explore other models

```
In [66]: df.head(50).plot(kind="bar", figsize=(15,7))
         plt.show()
```



5.1.4 Random Forest

Random Forest may be of help here as

1. Random Forest can handle large datasets with high dimensionality without overfitting, making it a good choice when you have a lot of data.
2. Provides an estimate of the relative importance of each feature in the dataset, allowing you to identify the most important variables.
3. It can capture non-linear relationships between the features and the target variable, making it suitable for complex problems.

Observations when 20 decision trees are used.

Random Forest Evaluation	**Observations**
Mean Absolute Error	3073.74
Mean Squared Error	31221881.16
Root Mean Squared Error	5587.65

Observations when 100 decision trees are used.

Random Forest Evaluation	**Observations**
Mean Absolute Error	2976.18
Mean Squared Error	29825503.52
Root Mean Squared Error	5461.27

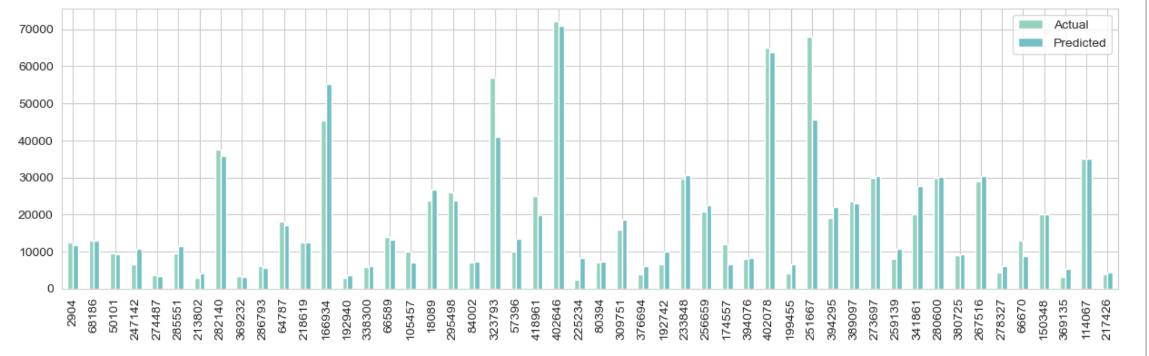
Observations when 200 decision trees are used.

Random Forest Evaluation	**Observations**
Mean Absolute Error	2961.59
Mean Squared Error	29634253.91
Root Mean Squared Error	5443.74

It can be observed from above tables that the increase in the number of decision trees decreased the MAE.

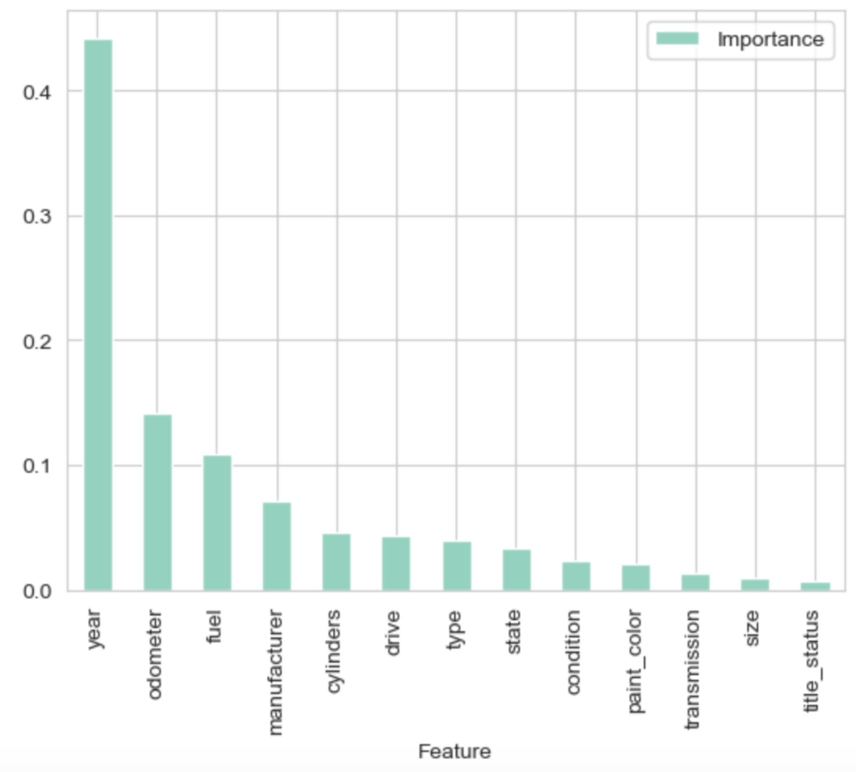
Below graph indicates the predicted and actual values are close when compared to Linear regression predictions.

```
In [66]: df.head(50).plot(kind="bar", figsize=(15,4))
         plt.show()
```



Below plot indicates that Year is the most important feature followed by odometer. Title_status is the least important feature.

```
In [79]: random_forest_importance_df.plot(kind="bar", x="Feature", y="Importance")

Out[79]: <AxesSubplot:xlabel='Feature'>
```



5.1.5 Ridge Regression

Ridge regression is a type of linear regression that adds a penalty term to the cost function to reduce the impact of high-valued coefficients on the model. The Ridge regression cost function is defined as:

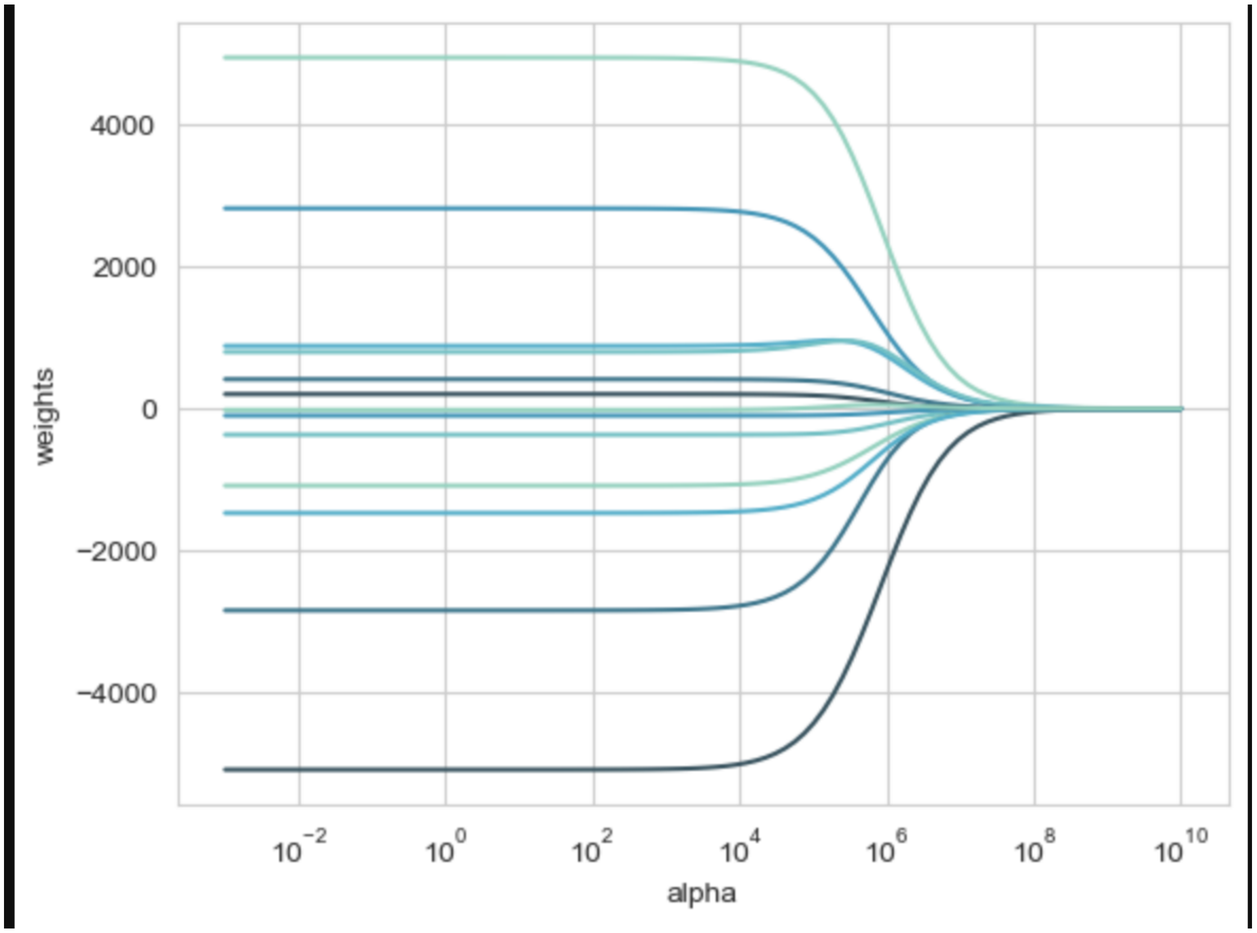
J(w) = RSS(w) + alpha * ||w||^2

where:

- J(w) is the Ridge regression cost function.
- RSS(w) is the residual sum of squares of the linear regression model.
- alpha is the hyperparameter that controls the strength of the penalty term.
- w is the vector of model coefficients
- ||w||^2 is the squared L2-norm of the coefficient vector, which is the sum of the squared values of the coefficients.

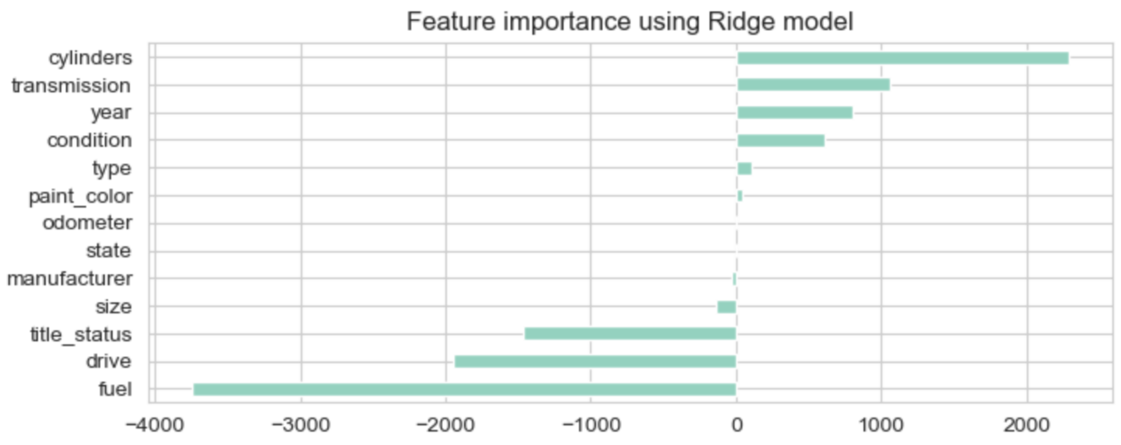
The objective of Ridge regression is to find the values of w that minimize the cost function J(w).

Below figure shows how coefficients are decreasing with the increase in alpha value.



Cross validation was applied to get the best alpha value. Best alpha value is 0.0005.

Ridge Regression Evaluation	Observations
Mean Absolute Error	7236.96
Mean Squared Error	98178770.7
Root Mean Squared Error	9908.52

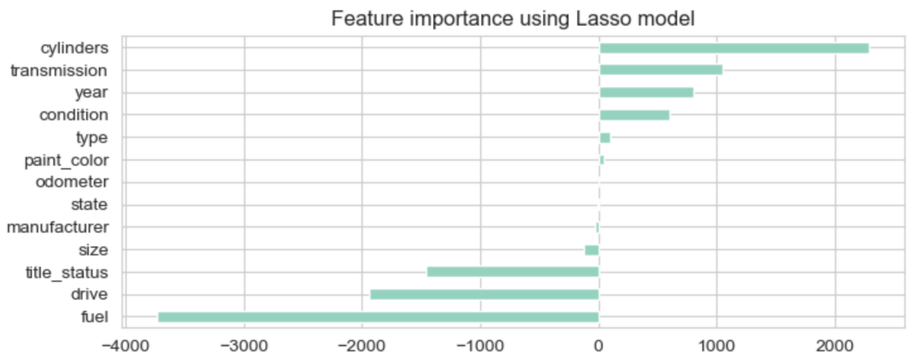


5.1.6 Lasso Regression

Cross validation is applied to find the best value for Lambda.

Lasso Model Evaluation	Observations
Mean Absolute Error	7237.72
Mean Squared Error	98174684.53
Root Mean Squared Error	9908.31

```
Out[122]: <AxesSubplot:title={'center':'Feature importance using Lasso model'}>
```



6 Conclusion

By looking at the results from four models, it can be concluded that the Random Forest model is the best based on Mean Absolute Error, Mean Squared Error and Root Mean Squared Error.

Year, Odometer, Fuel, Manufacturer, Cylinders, Type, Drive, and state are the important features.

	Lasso Model	Ridge Regression	Linear Regression	Random Forest
Mean Absolute Error	7237.72	7236.96	7237.44	2961.59
Mean Squared Error	98174684.53	98178770.7	98170328.81	29634253.91
Root Mean Squared Error	9908.31	9908.52	9908.094105971484	5443.74

6.1 Next steps and recommendations

Use a large amount of data would have been helpful to make the model prediction even more robust. Additional features such as how often the car was serviced, previous accident info, claim info, number of ownership transfers etc., would be helpful for better price predictions.