In [ ]:

1. Problem Definition

For this dataset, the problem we're trying to solve, or better, the question we're trying to answer is,

How well can we predict the future sale price of a bulldozer, given its characteristics previous examples of how much similar bulldozers have been sold for?

2. Data

Looking at the dataset from Kaggle, you can you it's a time series problem. This means there's a time attribute to dataset.

In this case, it's historical sales data of bulldozers. Including things like, model type, size, sale date and more.

There are 3 datasets:

- Train.csv - Historical bulldozer sales examples up to 2011 (close to 400,000 examples with 50+ different attributes, including SalePrice which is the target variable).
- Valid.csv - Historical bulldozer sales examples from January 1 2012 to April 30 2012 (close to 12,000 examples with the same attributes as Train.csv).
- Test.csv - Historical bulldozer sales examples from May 1 2012 to November 2012 (close to 12,000 examples but missing the SalePrice attribute, as this is what we'll be trying to predict).

3. Evaluation

For this problem, Kaggle has set the evaluation metric to being root mean squared log error (RMSLE). As with many regression evaluations, the goal will be to get this value as low as possible.

To see how well our model is doing, we'll calculate the RMSLE and then compare our results to others on the Kaggle leaderboard.

4. Features

Features are different parts of the data. During this step, you'll want to start finding out what you can about the data.

One of the most common ways to do this, is to create a data dictionary.

For this dataset, Kaggle provide a data dictionary which contains information about what each attribute of the dataset means. You can download this file directly from the Kaggle competition page (account required) or view it on Google Sheets.

With all of this being known, let's get started!

First, we'll import the dataset and start exploring. Since we know the evaluation metric we're trying to minimise, our first goal will be building a baseline model and seeing how it stacks up against the competition

In [121…
```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
import sklearn
```

In [15]:
```
df= pd.read_csv("TrainAndValid.csv",low_memory=False)
df
```

Out[15]:

	SalesID	SalePrice	MachineID	ModelID	datasource	auctioneerID	YearMade	MachineHoursCurrentMeter	UsageBand	saledate	...
0	1139246	66000.0	999089	3157	121	3.0	2004	68.0	Low	11/16/2006 0:00	...
1	1139248	57000.0	117657	77	121	3.0	1996	4640.0	Low	3/26/2004 0:00	...
2	1139249	10000.0	434808	7009	121	3.0	2001	2838.0	High	2/26/2004 0:00	...
3	1139251	38500.0	1026470	332	121	3.0	2001	3486.0	High	5/19/2011 0:00	...
4	1139253	11000.0	1057373	17311	121	3.0	2007	722.0	Medium	7/23/2009 0:00	...
...	...	...	...	...	...	...	...	...	...	...	...
412693	6333344	10000.0	1919201	21435	149	2.0	2005	NaN	NaN	3/7/2012 0:00	...
412694	6333345	10500.0	1882122	21436	149	2.0	2005	NaN	NaN	1/28/2012 0:00	...
412695	6333347	12500.0	1944213	21435	149	2.0	2005	NaN	NaN	1/28/2012 0:00	...
412696	6333348	10000.0	1794518	21435	149	2.0	2006	NaN	NaN	3/7/2012 0:00	...
412697	6333349	13000.0	1944743	21436	149	2.0	2006	NaN	NaN	1/28/2012 0:00	...

412698 rows × 53 columns

In [3]:
```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 412698 entries, 0 to 412697
Data columns (total 53 columns):
 #   Column                    Non-Null Count   Dtype  
---  ------                    --------------   -----  
 0   SalesID                   412698 non-null  int64  
 1   SalePrice                 412698 non-null  float64
 2   MachineID                 412698 non-null  int64  
 3   ModelID                   412698 non-null  int64  
 4   datasource                412698 non-null  int64  
 5   auctioneerID              392562 non-null  float64
 6   YearMade                  412698 non-null  int64  
 7   MachineHoursCurrentMeter  147504 non-null  float64
 8   UsageBand                 73670 non-null   object 
 9   saledate                  412698 non-null  object 
 10  fiModelDesc               412698 non-null  object 
 11  fiBaseModel               412698 non-null  object 
 12  fiSecondaryDesc           271971 non-null  object 
 13  fiModelSeries             58667 non-null   object 
 14  fiModelDescriptor         74816 non-null   object 
 15  ProductSize               196093 non-null  object 
 16  fiProductClassDesc        412698 non-null  object 
 17  state                     412698 non-null  object 
 18  ProductGroup              412698 non-null  object 
 19  ProductGroupDesc          412698 non-null  object 
 20  Drive_System              107087 non-null  object 
 21  Enclosure                 412364 non-null  object 
 22  Forks                     197715 non-null  object 
 23  Pad_Type                  81096 non-null   object 
 24  Ride_Control              152728 non-null  object 
 25  Stick                     81096 non-null   object 
 26  Transmission              188007 non-null  object 
 27  Turbocharged              81096 non-null   object 
 28  Blade_Extension           25983 non-null   object 
 29  Blade_Width               25983 non-null   object 
 30  Enclosure_Type            25983 non-null   object 
 31  Engine_Horsepower         25983 non-null   object 
 32  Hydraulics                330133 non-null  object 
 33  Pushblock                 25983 non-null   object 
 34  Ripper                    106945 non-null  object 
 35  Scarifier                 25994 non-null   object 
 36  Tip_Control               25983 non-null   object 
 37  Tire_Size                 97638 non-null   object 
 38  Coupler                   220679 non-null  object 
 39  Coupler_System            44974 non-null   object 
 40  Grouser_Tracks            44875 non-null   object 
 41  Hydraulics_Flow           44875 non-null   object 
 42  Track_Type                102193 non-null  object 
 43  Undercarriage_Pad_Width   102916 non-null  object 
 44  Stick_Length              102261 non-null  object 
 45  Thumb                     102332 non-null  object 
 46  Pattern_Changer           102261 non-null  object 
 47  Grouser_Type              102193 non-null  object 
 48  Backhoe_Mounting          80712 non-null   object 
 49  Blade_Type                81875 non-null   object 
 50  Travel_Controls           81877 non-null   object 
 51  Differential_Type         71564 non-null   object 
 52  Steering_Controls         71522 non-null   object 
dtypes: float64(3), int64(5), object(45)
memory usage: 166.9+ MB
```

In [4]:
```
df.isnull().sum()
```

Out[4]:
```
SalesID                         0
SalePrice                       0
MachineID                       0
ModelID                         0
datasource                      0
auctioneerID                20136
YearMade                        0
MachineHoursCurrentMeter    265194
UsageBand                   339028
saledate                        0
fiModelDesc                     0
fiBaseModel                     0
fiSecondaryDesc             140727
fiModelSeries               354031
fiModelDescriptor           337882
ProductSize                 216605
fiProductClassDesc              0
state                           0
ProductGroup                    0
ProductGroupDesc                0
Drive_System                305611
Enclosure                      334
Forks                       214983
Pad_Type                    331602
Ride_Control                259970
Stick                       331602
Transmission                224691
Turbocharged                331602
Blade_Extension             386715
Blade_Width                 386715
Enclosure_Type              386715
Engine_Horsepower           386715
Hydraulics                   82565
Pushblock                   386715
Ripper                      305753
Scarifier                   386704
Tip_Control                 386715
Tire_Size                   315060
Coupler                     192019
Coupler_System              367724
Grouser_Tracks              367823
Hydraulics_Flow             367823
Track_Type                  310505
Undercarriage_Pad_Width     309782
Stick_Length                310437
Thumb                       310366
Pattern_Changer             310437
Grouser_Type                310505
Backhoe_Mounting            331986
Blade_Type                  330823
Travel_Controls             330821
Differential_Type           341134
Steering_Controls           341176
dtype: int64
```

In [5]:
```
df.columns
```

Out[5]:
```
Index(['SalesID', 'SalePrice', 'MachineID', 'ModelID', 'datasource',
       'auctioneerID', 'YearMade', 'MachineHoursCurrentMeter', 'UsageBand',
       'saledate', 'fiModelDesc', 'fiBaseModel', 'fiSecondaryDesc',
       'fiModelSeries', 'fiModelDescriptor', 'ProductSize',
       'fiProductClassDesc', 'state', 'ProductGroup', 'ProductGroupDesc',
       'Drive_System', 'Enclosure', 'Forks', 'Pad_Type', 'Ride_Control',
       'Stick', 'Transmission', 'Turbocharged', 'Blade_Extension',
       'Blade_Width', 'Enclosure_Type', 'Engine_Horsepower', 'Hydraulics',
       'Pushblock', 'Ripper', 'Scarifier', 'Tip_Control', 'Tire_Size',
       'Coupler', 'Coupler_System', 'Grouser_Tracks', 'Hydraulics_Flow',
       'Track_Type', 'Undercarriage_Pad_Width', 'Stick_Length', 'Thumb',
       'Pattern_Changer', 'Grouser_Type', 'Backhoe_Mounting', 'Blade_Type',
       'Travel_Controls', 'Differential_Type', 'Steering_Controls'],
      dtype='object')
```

In [6]:
```
df.saledate.dtype
```

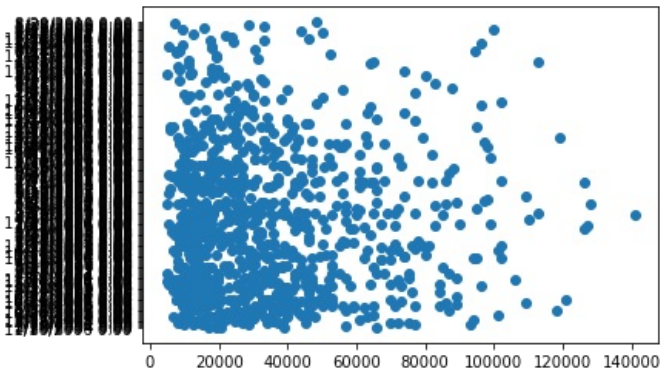
Out[6]:
```
dtype('O')
```

In [7]:
```
df.saledate
```

Out[7]:
```
0         11/16/2006 0:00
1          3/26/2004 0:00
2          2/26/2004 0:00
3          5/19/2011 0:00
4          7/23/2009 0:00
               ...
412693      3/7/2012 0:00
412694     1/28/2012 0:00
412695     1/28/2012 0:00
412696      3/7/2012 0:00
412697     1/28/2012 0:00
Name: saledate, Length: 412698, dtype: object
```

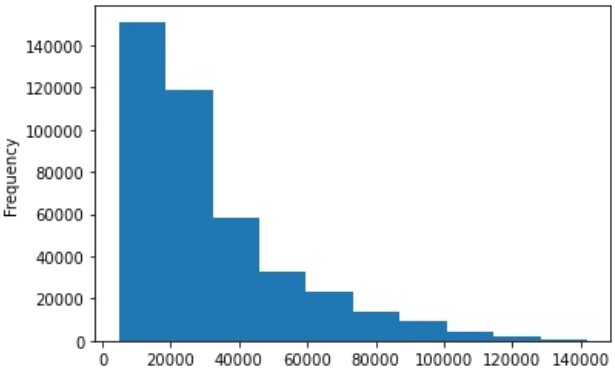
In [8]:
```
fig,ax= plt.subplots()
ax.scatter(df["SalePrice"][:1000],df["saledate"][:1000],cmap=("r","g"))
```

Out[8]:
```
<matplotlib.collections.PathCollection at 0x1b85aa7c5b0>
```



In [10]:
```
df.SalePrice.plot.hist()
```

Out[10]:
```
<AxesSubplot:ylabel='Frequency'>
```



In [ ]:
```

```

Dealing with Times Series

1. Parsing date

when we are dealing with times series data, we want to enrich the date & time series as much as possible

We can do this by telling Pandas which of the columns has date in it using the "parse_dates" parameter.

In [16]:
```
# Import the data again, but this time , parse date
df= pd.read_csv("TrainAndValid.csv", low_memory=False, parse_dates= ["saledate"])
df
```

Out[16]:

	SalesID	SalePrice	MachineID	ModelID	datasource	auctioneerID	YearMade	MachineHoursCurrentMeter	UsageBand	saledate	...
0	1139246	66000.0	999089	3157	121	3.0	2004	68.0	Low	2006-11-16	...
1	1139248	57000.0	117657	77	121	3.0	1996	4640.0	Low	2004-03-26	...
2	1139249	10000.0	434808	7009	121	3.0	2001	2838.0	High	2004-02-26	...
3	1139251	38500.0	1026470	332	121	3.0	2001	3486.0	High	2011-05-19	...
4	1139253	11000.0	1057373	17311	121	3.0	2007	722.0	Medium	2009-07-23	...
...	...	...	...	...	...	...	...	...	...	...	...
412693	6333344	10000.0	1919201	21435	149	2.0	2005	NaN	NaN	2012-03-07	...
412694	6333345	10500.0	1882122	21436	149	2.0	2005	NaN	NaN	2012-01-28	...
412695	6333347	12500.0	1944213	21435	149	2.0	2005	NaN	NaN	2012-01-28	...
412696	6333348	10000.0	1794518	21435	149	2.0	2006	NaN	NaN	2012-03-07	...
412697	6333349	13000.0	1944743	21436	149	2.0	2006	NaN	NaN	2012-01-28	...

412698 rows × 53 columns

In [12]:
```
df.saledate.dtype
```

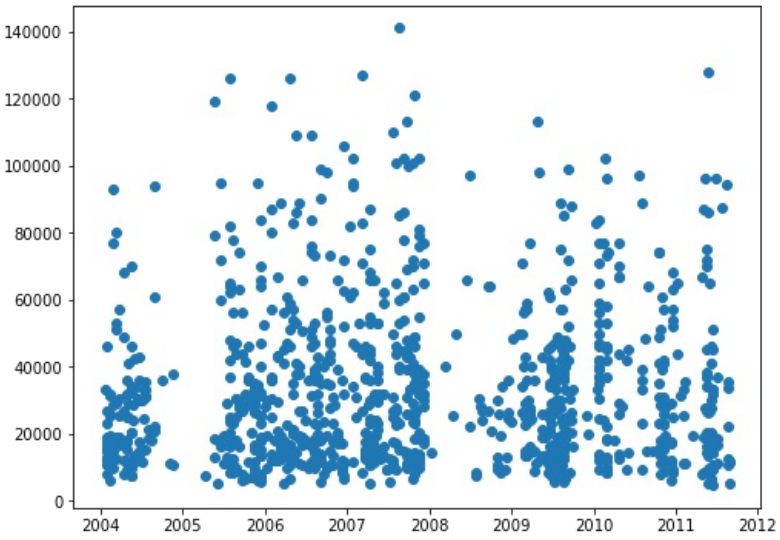
Out[12]:
```
dtype('<M8[ns]')
```

In [13]:
```
df.saledate[:1000]
```

Out[13]:
```
0      2006-11-16
1      2004-03-26
2      2004-02-26
3      2011-05-19
4      2009-07-23
          ...
995    2009-07-16
996    2007-06-14
997    2005-09-22
998    2005-07-28
999    2011-06-16
Name: saledate, Length: 1000, dtype: datetime64[ns]
```

In [14]:
```
fig,ax= plt.subplots(figsize=(8,6))
ax.scatter(df.saledate[:1000], df.SalePrice[:1000])
```

Out[14]:
```
<matplotlib.collections.PathCollection at 0x1b805dce790>
```



In [15]:
```
df.saledate.head(20)
```

Out[15]:
```
0    2006-11-16
1    2004-03-26
2    2004-02-26
3    2011-05-19
4    2009-07-23
5    2008-12-18
6    2004-08-26
7    2005-11-17
8    2009-08-27
9    2007-08-09
10   2008-08-21
11   2006-08-24
12   2005-10-20
13   2006-01-26
14   2006-01-03
15   2006-11-16
16   2007-06-14
17   2010-01-28
18   2006-03-09
19   2005-11-17
Name: saledate, dtype: datetime64[ns]
```

Sort DataFrame by saledate

When working with time-series data, it is good practice to sort it by date

In [16]:
```
df.sort_values(by=["saledate"], ascending=True, inplace=True)
df.saledate.head(20)
```

Out[16]:
```
205615   1989-01-17
274835   1989-01-31
141296   1989-01-31
212552   1989-01-31
62755    1989-01-31
54653    1989-01-31
81383    1989-01-31
204924   1989-01-31
135376   1989-01-31
113390   1989-01-31
113394   1989-01-31
116419   1989-01-31
32138    1989-01-31
127610   1989-01-31
76171    1989-01-31
127000   1989-01-31
128130   1989-01-31
127626   1989-01-31
55455    1989-01-31
55454    1989-01-31
Name: saledate, dtype: datetime64[ns]
```

Make A Copy Of The Original DataFrame

It is good practices to make a copy of the original dataframe, so that whatever we do on our copy won't affect the original dataframe

In [17]:
```
# Make a copy
df_tmp= df.copy()
```

In [18]:
```
df_tmp
```

Out[18]:

	SalesID	SalePrice	MachineID	ModelID	datasource	auctioneerID	YearMade	MachineHoursCurrentMeter	UsageBand	saledate	...
0	1139246	66000.0	999089	3157	121	3.0	2004	68.0	Low	2006-11-16	...
1	1139248	57000.0	117657	77	121	3.0	1996	4640.0	Low	2004-03-26	...
2	1139249	10000.0	434808	7009	121	3.0	2001	2838.0	High	2004-02-26	...
3	1139251	38500.0	1026470	332	121	3.0	2001	3486.0	High	2011-05-19	...
4	1139253	11000.0	1057373	17311	121	3.0	2007	722.0	Medium	2009-07-23	...
...	...	...	...	...	...	...	...	...	...	...	...
412693	6333344	10000.0	1919201	21435	149	2.0	2005	NaN	NaN	2012-03-07	...
412694	6333345	10500.0	1882122	21436	149	2.0	2005	NaN	NaN	2012-01-28	...
412695	6333347	12500.0	1944213	21435	149	2.0	2005	NaN	NaN	2012-01-28	...
412696	6333348	10000.0	1794518	21435	149	2.0	2006	NaN	NaN	2012-03-07	...
412697	6333349	13000.0	1944743	21436	149	2.0	2006	NaN	NaN	2012-01-28	...

412698 rows × 53 columns

In [9]:
```
df_tmp.head(20).T
```

Out[9]:

	0	1	2	3	4	5	6	7	8
SalesID	1139246	1139248	1139249	1139251	1139253	1139255	1139256	1139261	1139272
SalePrice	66000.0	57000.0	10000.0	38500.0	11000.0	26500.0	21000.0	27000.0	21500.0
MachineID	999089	117657	434808	1026470	1057373	1001274	772701	902002	1036251
ModelID	3157	77	7009	332	17311	4605	1937	3539	36003
datasource	121	121	121	121	121	121	121	121	121
auctioneerID	3.0	3.0	3.0	3.0	3.0	3.0	3.0	3.0	3.0
YearMade	2004	1996	2001	2001	2007	2004	1993	2001	2008
MachineHoursCurrentMeter	68.0	4640.0	2838.0	3486.0	722.0	508.0	11540.0	4883.0	302.0
UsageBand	Low	Low	High	High	Medium	Low	High	High	Low
saledate	2006-11-16 00:00:00	2004-03-26 00:00:00	2004-02-26 00:00:00	2011-05-19 00:00:00	2009-07-23 00:00:00	2008-12-18 00:00:00	2004-08-26 00:00:00	2005-11-17 00:00:00	2009-08-27 00:00:00
fiModelDesc	521D	950FII	226	PC120-6E	S175	310G	790ELC	416D	430HAG
fiBaseModel	521	950	226	PC120	S175	310	790	416	430
fiSecondaryDesc	D	F	NaN	NaN	NaN	G	E	D	HAG
fiModelSeries	NaN	II	NaN	-6E	NaN	NaN	NaN	NaN	NaN
fiModelDescriptor	NaN	NaN	NaN	NaN	NaN	NaN	LC	NaN	NaN
ProductSize	NaN	Medium	NaN	Small	NaN	NaN	Large / Medium	NaN	Mini
fiProductClassDesc	Wheel Loader - 110.0 to 120.0 Horsepower	Wheel Loader - 150.0 to 175.0 Horsepower	Skid Steer Loader - 1351.0 to 1601.0 Lb Operat...	Hydraulic Excavator, Track - 12.0 to 14.0 Metr...	Skid Steer Loader - 1601.0 to 1751.0 Lb Operat...	Backhoe Loader - 14.0 to 15.0 Ft Standard Digg...	Hydraulic Excavator, Track - 21.0 to 24.0 Metr...	Backhoe Loader - 14.0 to 15.0 Ft Standard Digg...	Hydraulic Excavator, Track - 3.0 to 4.0 Metric...
state	Alabama	North Carolina	New York	Texas	New York	Arizona	Florida	Illinois	Texas
ProductGroup	WL	WL	SSL	TEX	SSL	BL	TEX	BL	TEX
ProductGroupDesc	Wheel Loader	Wheel Loader	Skid Steer Loaders	Track Excavators	Skid Steer Loaders	Backhoe Loaders	Track Excavators	Backhoe Loaders	Track Excavators
Drive_System	NaN	NaN	NaN	NaN	NaN	Four Wheel Drive	NaN	Four Wheel Drive	NaN
Enclosure	EROPS w AC	EROPS w AC	OROPS	EROPS w AC	EROPS	OROPS	EROPS	OROPS	EROPS
Forks	None or Unspecified	None or Unspecified	None or Unspecified	NaN	None or Unspecified	None or Unspecified	NaN	None or Unspecified	NaN
Pad_Type	NaN	NaN	NaN	NaN	NaN	None or Unspecified	NaN	Reversible	NaN
	None or	None or							

Ride_Control	Unspecified	Unspecified	NaN	NaN	NaN	No	NaN	No	NaN
Stick	NaN	NaN	NaN	NaN	NaN	Extended	NaN	Standard	NaN
Transmission	NaN	NaN	NaN	NaN	NaN	Powershuttle	NaN	Standard	NaN
Turbocharged	NaN	NaN	NaN	NaN	NaN	None or Unspecified	NaN	Yes	NaN
Blade_Extension	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
Blade_Width	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
Enclosure_Type	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
Engine_Horsepower	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
Hydraulics	2 Valve	2 Valve	Auxiliary	2 Valve	Auxiliary	NaN	Standard	NaN	Auxiliary
Pushblock	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
Ripper	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
Scarifier	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
Tip_Control	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
Tire_Size	None or Unspecified	23.5	NaN	NaN	NaN	NaN	NaN	NaN	NaN
Coupler	None or Unspecified	None or Unspecified	None or Unspecified	None or Unspecified	None or Unspecified	NaN	None or Unspecified	NaN	Manual
Coupler_System	NaN	NaN	None or Unspecified	NaN	None or Unspecified	NaN	NaN	NaN	NaN
Grouser_Tracks	NaN	NaN	None or Unspecified	NaN	None or Unspecified	NaN	NaN	NaN	NaN
Hydraulics_Flow	NaN	NaN	Standard	NaN	Standard	NaN	NaN	NaN	NaN
Track_Type	NaN	NaN	NaN	NaN	NaN	NaN	Steel	NaN	Rubber
Undercarriage_Pad_Width	NaN	NaN	NaN	NaN	NaN	NaN	None or Unspecified	NaN	None or Unspecified
Stick_Length	NaN	NaN	NaN	NaN	NaN	NaN	None or Unspecified	NaN	None or Unspecified
Thumb	NaN	NaN	NaN	NaN	NaN	NaN	None or Unspecified	NaN	None or Unspecified
Pattern_Changer	NaN	NaN	NaN	NaN	NaN	NaN	None or Unspecified	NaN	None or Unspecified
Grouser_Type	NaN	NaN	NaN	NaN	NaN	NaN	Double	NaN	Double
Backhoe_Mounting	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
Blade_Type	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
Travel_Controls	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
Differential_Type	Standard	Standard	NaN	NaN	NaN	NaN	NaN	NaN	NaN
Steering_Controls	Conventional	Conventional	NaN	NaN	NaN	NaN	NaN	NaN	NaN C

Add date time parameters for "saledate" column

check out the documentation: https://pandas.pydata.org/pandas-docs/version/0.23/generated/pandas.DatetimeIndex.html

In [19]:
```
df_tmp["saleYear"]= df_tmp.saledate.dt.year
df_tmp["saleMonth"]= df_tmp.saledate.dt.month
df_tmp["saleDay"]= df_tmp.saledate.dt.day
df_tmp["saleDayOfWeek"]= df_tmp.saledate.dt.dayofweek
df_tmp["saleDayOfYear"]= df_tmp.saledate.dt.dayofyear
```

In [20]:
```
df_tmp.head().T
```

Out[20]:

	0	1	2	3	4
SalesID	1139246	1139248	1139249	1139251	1139253
SalePrice	66000.0	57000.0	10000.0	38500.0	11000.0
MachineID	999089	117657	434808	1026470	1057373
ModelID	3157	77	7009	332	17311
datasource	121	121	121	121	121
auctioneerID	3.0	3.0	3.0	3.0	3.0
YearMade	2004	1996	2001	2001	2007
MachineHoursCurrentMeter	68.0	4640.0	2838.0	3486.0	722.0
UsageBand	Low	Low	High	High	Medium

UsageBand	Low	Low	High	High	Medium
saledate	2006-11-16 00:00:00	2004-03-26 00:00:00	2004-02-26 00:00:00	2011-05-19 00:00:00	2009-07-23 00:00:00
fiModelDesc	521D	950FII	226	PC120-6E	S175
fiBaseModel	521	950	226	PC120	S175
fiSecondaryDesc	D	F	NaN	NaN	NaN
fiModelSeries	NaN	II	NaN	-6E	NaN
fiModelDescriptor	NaN	NaN	NaN	NaN	NaN
ProductSize	NaN	Medium	NaN	Small	NaN
fiProductClassDesc	Wheel Loader - 110.0 to 120.0 Horsepower	Wheel Loader - 150.0 to 175.0 Horsepower	Skid Steer Loader - 1351.0 to 1601.0 Lb Operat...	Hydraulic Excavator, Track - 12.0 to 14.0 Metr...	Skid Steer Loader - 1601.0 to 1751.0 Lb Operat...
state	Alabama	North Carolina	New York	Texas	New York
ProductGroup	WL	WL	SSL	TEX	SSL
ProductGroupDesc	Wheel Loader	Wheel Loader	Skid Steer Loaders	Track Excavators	Skid Steer Loaders
Drive_System	NaN	NaN	NaN	NaN	NaN
Enclosure	EROPS w AC	EROPS w AC	OROPS	EROPS w AC	EROPS
Forks	None or Unspecified	None or Unspecified	None or Unspecified	NaN	None or Unspecified
Pad_Type	NaN	NaN	NaN	NaN	NaN
Ride_Control	None or Unspecified	None or Unspecified	NaN	NaN	NaN
Stick	NaN	NaN	NaN	NaN	NaN
Transmission	NaN	NaN	NaN	NaN	NaN
Turbocharged	NaN	NaN	NaN	NaN	NaN
Blade_Extension	NaN	NaN	NaN	NaN	NaN
Blade_Width	NaN	NaN	NaN	NaN	NaN
Enclosure_Type	NaN	NaN	NaN	NaN	NaN
Engine_Horsepower	NaN	NaN	NaN	NaN	NaN
Hydraulics	2 Valve	2 Valve	Auxiliary	2 Valve	Auxiliary
Pushblock	NaN	NaN	NaN	NaN	NaN
Ripper	NaN	NaN	NaN	NaN	NaN
Scarifier	NaN	NaN	NaN	NaN	NaN
Tip_Control	NaN	NaN	NaN	NaN	NaN
Tire_Size	None or Unspecified	23.5	NaN	NaN	NaN
Coupler	None or Unspecified	None or Unspecified	None or Unspecified	None or Unspecified	None or Unspecified
Coupler_System	NaN	NaN	None or Unspecified	NaN	None or Unspecified
Grouser_Tracks	NaN	NaN	None or Unspecified	NaN	None or Unspecified
Hydraulics_Flow	NaN	NaN	Standard	NaN	Standard
Track_Type	NaN	NaN	NaN	NaN	NaN
Undercarriage_Pad_Width	NaN	NaN	NaN	NaN	NaN
Stick_Length	NaN	NaN	NaN	NaN	NaN
Thumb	NaN	NaN	NaN	NaN	NaN
Pattern_Changer	NaN	NaN	NaN	NaN	NaN
Grouser_Type	NaN	NaN	NaN	NaN	NaN
Backhoe_Mounting	NaN	NaN	NaN	NaN	NaN
Blade_Type	NaN	NaN	NaN	NaN	NaN
Travel_Controls	NaN	NaN	NaN	NaN	NaN
Differential_Type	Standard	Standard	NaN	NaN	NaN
Steering_Controls	Conventional	Conventional	NaN	NaN	NaN
saleYear	2006	2004	2004	2011	2009
saleMonth	11	3	2	5	7
saleDay	16	26	26	19	23
saleDayOfWeek	3	4	3	3	3
saleDayOfYear	320	86	57	139	204

In [21]:
```
# Now we've enriched our DataFrame with date time features, we can now remove "saledate"
df_tmp.drop("saledate",axis=1, inplace=True)
```

In [31]:
```
df_tmp.state.value_counts()
```

Out[31]:

```
Florida           67320
Texas             53110
California        29761
Washington        16222
Georgia           14633
Maryland          13322
Mississippi       13240
Ohio              12369
Illinois          11540
Colorado          11529
New Jersey        11156
North Carolina    10636
Tennessee         10298
Alabama           10292
Pennsylvania      10234
South Carolina     9951
Arizona            9364
New York           8639
Connecticut        8276
Minnesota          7885
Missouri           7178
Nevada             6932
Louisiana          6627
Kentucky           5351
Maine              5096
Indiana            4124
Arkansas           3933
New Mexico         3631
Utah               3046
Unspecified        2801
Wisconsin          2745
New Hampshire      2738
Virginia           2353
Idaho              2025
Oregon             1911
Michigan           1831
Wyoming            1672
Montana            1336
Iowa               1336
Oklahoma           1326
Nebraska            866
West Virginia       840
Kansas              667
Delaware            510
North Dakota        480
Alaska              430
Massachusetts       347
Vermont             300
South Dakota        244
Hawaii              118
Rhode Island         83
Puerto Rico          42
Washington DC         2
Name: state, dtype: int64
```

In [33]:

```
df_tmp.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 412698 entries, 205615 to 409203
Data columns (total 57 columns):
 #   Column                    Non-Null Count   Dtype
---  ------                    --------------   -----
 0   SalesID                   412698 non-null  int64
 1   SalePrice                 412698 non-null  float64
 2   MachineID                 412698 non-null  int64
 3   ModelID                   412698 non-null  int64
 4   datasource                412698 non-null  int64
 5   auctioneerID              392562 non-null  float64
 6   YearMade                  412698 non-null  int64
 7   MachineHoursCurrentMeter  147504 non-null  float64
 8   UsageBand                 73670 non-null   object
 9   fiModelDesc               412698 non-null  object
 10  fiBaseModel               412698 non-null  object
 11  fiSecondaryDesc           271971 non-null  object
 12  fiModelSeries             58667 non-null   object
 13  fiModelDescriptor         74816 non-null   object
 14  ProductSize               196093 non-null  object
 15  fiProductClassDesc        412698 non-null  object
 16  state                     412698 non-null  object
 17  ProductGroup              412698 non-null  object
 18  ProductGroupDesc          412698 non-null  object
 19  Drive_System              107087 non-null  object
 20  Enclosure                 412364 non-null  object
 21  Forks                     197715 non-null  object
 22  Pad_Type                  81096 non-null   object
 23  Ride_Control              152728 non-null  object
 24  Stick                     81096 non-null   object
 25  Transmission              188007 non-null  object
 26  Turbocharged              81096 non-null   object
 27  Blade_Extension           25983 non-null   object
 28  Blade_Width               25983 non-null   object
 29  Enclosure_Type            25983 non-null   object
 30  Engine_Horsepower         25983 non-null   object
 31  Hydraulics                330133 non-null  object
 32  Pushblock                 25983 non-null   object
 33  Ripper                    106945 non-null  object
 34  Scarifier                 25994 non-null   object
 35  Tip_Control               25983 non-null   object
 36  Tire_Size                 97638 non-null   object
 37  Coupler                   220679 non-null  object
 38  Coupler_System            44974 non-null   object
 39  Grouser_Tracks            44875 non-null   object
 40  Hydraulics_Flow           44875 non-null   object
 41  Track_Type                102193 non-null  object
 42  Undercarriage_Pad_Width   102916 non-null  object
 43  Stick_Length              102261 non-null  object
 44  Thumb                     102332 non-null  object
 45  Pattern_Changer           102261 non-null  object
 46  Grouser_Type              102193 non-null  object
 47  Backhoe_Mounting          80712 non-null   object
 48  Blade_Type                81875 non-null   object
 49  Travel_Controls           81877 non-null   object
 50  Differential_Type         71564 non-null   object
 51  Steering_Controls         71522 non-null   object
 52  saleYear                  412698 non-null  int64
 53  saleMonth                 412698 non-null  int64
 54  saleDay                   412698 non-null  int64
 55  saleDayOfWeek             412698 non-null  int64
 56  saleDayOfYear             412698 non-null  int64
dtypes: float64(3), int64(10), object(44)
memory usage: 182.6+ MB
```

Covert String to Categories

One way we can turn all of our data into numbers is by converting them into pandas categories.

We can check the different dtype compatible with pandas here:

https://pandas.pydata.org/pandas-docs/version/0.25/reference/general_utility_functions.html#data-types-related-functionality

In [34]:
```
df_tmp.head().T
```

Out[34]:

	205615	274835	141296	212552	62755
SalesID	1646770	1821514	1505138	1671174	1329056
SalePrice	9500.0	14000.0	50000.0	16000.0	22000.0
MachineID	1126363	1194089	1473654	1327630	1336053
ModelID	8434	10150	4139	8591	4089
datasource	132	132	132	132	132
auctioneerID	18.0	99.0	99.0	99.0	99.0
YearMade	1974	1980	1978	1980	1984

MachineHoursCurrentMeter	NaN	NaN	NaN	NaN	NaN
UsageBand	NaN	NaN	NaN	NaN	NaN
fiModelDesc	TD20	A66	D7G	A62	D3B
fiBaseModel	TD20	A66	D7	A62	D3
fiSecondaryDesc	NaN	NaN	G	NaN	B
fiModelSeries	NaN	NaN	NaN	NaN	NaN
fiModelDescriptor	NaN	NaN	NaN	NaN	NaN
ProductSize	Medium	NaN	Large	NaN	NaN
fiProductClassDesc	Track Type Tractor, Dozer - 105.0 to 130.0 Hor...	Wheel Loader - 120.0 to 135.0 Horsepower	Track Type Tractor, Dozer - 190.0 to 260.0 Hor...	Wheel Loader - Unidentified	Track Type Tractor, Dozer - 20.0 to 75.0 Horse...
state	Texas	Florida	Florida	Florida	Florida
ProductGroup	TTT	WL	TTT	WL	TTT
ProductGroupDesc	Track Type Tractors	Wheel Loader	Track Type Tractors	Wheel Loader	Track Type Tractors
Drive_System	NaN	NaN	NaN	NaN	NaN
Enclosure	OROPS	OROPS	OROPS	EROPS	OROPS
Forks	NaN	None or Unspecified	NaN	None or Unspecified	NaN
Pad_Type	NaN	NaN	NaN	NaN	NaN
Ride_Control	NaN	None or Unspecified	NaN	None or Unspecified	NaN
Stick	NaN	NaN	NaN	NaN	NaN
Transmission	Direct Drive	NaN	Standard	NaN	Standard
Turbocharged	NaN	NaN	NaN	NaN	NaN
Blade_Extension	NaN	NaN	NaN	NaN	NaN
Blade_Width	NaN	NaN	NaN	NaN	NaN
Enclosure_Type	NaN	NaN	NaN	NaN	NaN
Engine_Horsepower	NaN	NaN	NaN	NaN	NaN
Hydraulics	2 Valve	2 Valve	2 Valve	2 Valve	2 Valve
Pushblock	NaN	NaN	NaN	NaN	NaN
Ripper	None or Unspecified	NaN	None or Unspecified	NaN	None or Unspecified
Scarifier	NaN	NaN	NaN	NaN	NaN
Tip_Control	NaN	NaN	NaN	NaN	NaN
Tire_Size	NaN	None or Unspecified	NaN	None or Unspecified	NaN
Coupler	NaN	None or Unspecified	NaN	None or Unspecified	NaN
Coupler_System	NaN	NaN	NaN	NaN	NaN
Grouser_Tracks	NaN	NaN	NaN	NaN	NaN
Hydraulics_Flow	NaN	NaN	NaN	NaN	NaN
Track_Type	NaN	NaN	NaN	NaN	NaN
Undercarriage_Pad_Width	NaN	NaN	NaN	NaN	NaN
Stick_Length	NaN	NaN	NaN	NaN	NaN
Thumb	NaN	NaN	NaN	NaN	NaN
Pattern_Changer	NaN	NaN	NaN	NaN	NaN
Grouser_Type	NaN	NaN	NaN	NaN	NaN
Backhoe_Mounting	None or Unspecified	NaN	None or Unspecified	NaN	None or Unspecified
Blade_Type	Straight	NaN	Straight	NaN	PAT
Travel_Controls	None or Unspecified	NaN	None or Unspecified	NaN	Lever
Differential_Type	NaN	Standard	NaN	Standard	NaN
Steering_Controls	NaN	Conventional	NaN	Conventional	NaN
saleYear	1989	1989	1989	1989	1989
saleMonth	1	1	1	1	1
saleDay	17	31	31	31	31
saleDayOfWeek	1	1	1	1	1
saleDayOfYear	17	31	31	31	31

In [22]:
```
# To check if What kind of datatype you are working with
```

In [22]:
```
# To check if what kind of datatype you are working with
pd.api.types.is_string_dtype(df_tmp["UsageBand"])
```

Out[22]:
```
True
```

In [23]:
```
# Find the columns which contains strings
for label,content in df_tmp.items():
    if pd.api.types.is_string_dtype(content):
        print(label)
```

```
UsageBand
fiModelDesc
fiBaseModel
fiSecondaryDesc
fiModelSeries
fiModelDescriptor
ProductSize
fiProductClassDesc
state
ProductGroup
ProductGroupDesc
Drive_System
Enclosure
Forks
Pad_Type
Ride_Control
Stick
Transmission
Turbocharged
Blade_Extension
Blade_Width
Enclosure_Type
Engine_Horsepower
Hydraulics
Pushblock
Ripper
Scarifier
Tip_Control
Tire_Size
Coupler
Coupler_System
Grouser_Tracks
Hydraulics_Flow
Track_Type
Undercarriage_Pad_Width
Stick_Length
Thumb
Pattern_Changer
Grouser_Type
Backhoe_Mounting
Blade_Type
Travel_Controls
Differential_Type
Steering_Controls
```

In [24]:
```
#This will turn all string values into category values
for label,content in df_tmp.items():
    if pd.api.types.is_string_dtype(content):
        df_tmp[label]= content.astype("category").cat.as_ordered()
```

In [12]:
```
df_tmp.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 412698 entries, 0 to 412697
Data columns (total 58 columns):
 #   Column                    Non-Null Count   Dtype
---  ------                    --------------   -----
 0   SalesID                   412698 non-null  int64
 1   SalePrice                 412698 non-null  float64
 2   MachineID                 412698 non-null  int64
 3   ModelID                   412698 non-null  int64
 4   datasource                412698 non-null  int64
 5   auctioneerID              392562 non-null  float64
 6   YearMade                  412698 non-null  int64
 7   MachineHoursCurrentMeter  147504 non-null  float64
 8   UsageBand                 73670 non-null   category
 9   saledate                  412698 non-null  datetime64[ns]
 10  fiModelDesc               412698 non-null  category
 11  fiBaseModel               412698 non-null  category
 12  fiSecondaryDesc           271971 non-null  category
 13  fiModelSeries             58667 non-null   category
 14  fiModelDescriptor         74816 non-null   category
 15  ProductSize               196093 non-null  category
 16  fiProductClassDesc        412698 non-null  category
 17  state                     412698 non-null  category
 18  ProductGroup              412698 non-null  category
 19  ProductGroupDesc          412698 non-null  category
 20  Drive_System              107087 non-null  category
 21  Enclosure                 412364 non-null  category
 22  Forks                     197715 non-null  category
 23  Pad_Type                  81096 non-null   category
 24  Ride_Control              152728 non-null  category
 25  Stick                     81096 non-null   category
 26  Transmission              188007 non-null  category
 27  Turbocharged              81096 non-null   category
 28  Blade_Extension           25983 non-null   category
 29  Blade_Width               25983 non-null   category
 30  Enclosure_Type            25983 non-null   category
 31  Engine_Horsepower         25983 non-null   category
 32  Hydraulics                330133 non-null  category
 33  Pushblock                 25983 non-null   category
 34  Ripper                    106945 non-null  category
 35  Scarifier                 25994 non-null   category
 36  Tip_Control               25983 non-null   category
 37  Tire_Size                 97638 non-null   category
 38  Coupler                   220679 non-null  category
 39  Coupler_System            44974 non-null   category
 40  Grouser_Tracks            44875 non-null   category
 41  Hydraulics_Flow           44875 non-null   category
 42  Track_Type                102193 non-null  category
 43  Undercarriage_Pad_Width   102916 non-null  category
 44  Stick_Length              102261 non-null  category
 45  Thumb                     102332 non-null  category
 46  Pattern_Changer           102261 non-null  category
 47  Grouser_Type              102193 non-null  category
 48  Backhoe_Mounting          80712 non-null   category
 49  Blade_Type                81875 non-null   category
 50  Travel_Controls           81877 non-null   category
 51  Differential_Type         71564 non-null   category
 52  Steering_Controls         71522 non-null   category
 53  saleYear                  412698 non-null  int64
 54  saleMonth                 412698 non-null  int64
 55  saleDay                   412698 non-null  int64
 56  saleDayOfWeek             412698 non-null  int64
 57  saleDayOfYear             412698 non-null  int64
dtypes: category(44), datetime64[ns](1), float64(3), int64(10)
memory usage: 63.2 MB
```

Fill Missing Values

Fill Numeric Missing Values First

In [25]:
```
for label,content in df_tmp.items():
    if pd.api.types.is_numeric_dtype(content):
        print(label)
```

```
SalesID
SalePrice
MachineID
ModelID
datasource
auctioneerID
YearMade
MachineHoursCurrentMeter
saleYear
saleMonth
saleDay
saleDayOfWeek
saleDayOfYear
```

In [26]:
```
# Check for whic numeric column has null values

for label,content in df_tmp.items():
    if pd.api.types.is_numeric_dtype(content):
        if pd.isnull(content).sum():
            print(label)
```

```
auctioneerID
MachineHoursCurrentMeter
```

In [30]:
```
#Fill numeric rows with the median

for label,content in df_tmp.items():
    if pd.api.types.is_numeric_dtype(content):
        if pd.isnull(content).sum():
            #Add a binary content which tell us if the column was missing
            df_tmp[label+ "Missing_Data"]= pd.isnull(content)
            #Fill missing numeric values with Median
            df_tmp[label]= content.fillna(content.median())
```

In [34]:
```
#Check if there are any null numeric value
for label,content in df_tmp.items():
    if pd.api.types.is_numeric_dtype(content):
        if pd.isnull(content).sum():
            print(label)
```

Filling and Turning Categorical Variables into numbers

- WE CAN TURN CATEGORICAL VARIABLES INTO NUMBERS USING (.CODE)

In [41]:
```
for label, content in df_tmp.items():
    if not pd.api.types.is_numeric_dtype(content):
        print(label)
```

```
UsageBand
fiModelDesc
fiBaseModel
fiSecondaryDesc
fiModelSeries
fiModelDescriptor
ProductSize
fiProductClassDesc
state
ProductGroup
ProductGroupDesc
Drive_System
Enclosure
Forks
Pad_Type
Ride_Control
Stick
Transmission
Turbocharged
Blade_Extension
Blade_Width
Enclosure_Type
Engine_Horsepower
Hydraulics
Pushblock
Ripper
Scarifier
Tip_Control
Tire_Size
Coupler
Coupler_System
Grouser_Tracks
Hydraulics_Flow
Track_Type
Undercarriage_Pad_Width
Stick_Length
Thumb
Pattern_Changer
Grouser_Type
Backhoe_Mounting
Blade_Type
Travel_Controls
Differential_Type
Steering_Controls
```

In [62]:
```
#Check for non numeric data
for label,content in df_tmp.items():
    if not pd.api.types.is_numeric_dtype(content):
        if pd.isnull(content).sum():
            print(label)
```

```
UsageBand
fiSecondaryDesc
fiModelSeries
fiModelDescriptor
ProductSize
Drive_System
Enclosure
Forks
Pad_Type
Ride_Control
Stick
Transmission
Turbocharged
Blade_Extension
Blade_Width
Enclosure_Type
Engine_Horsepower
Hydraulics
Pushblock
Ripper
Scarifier
Tip_Control
Tire_Size
Coupler
Coupler_System
Grouser_Tracks
Hydraulics_Flow
Track_Type
Undercarriage_Pad_Width
Stick_Length
Thumb
Pattern_Changer
Grouser_Type
Backhoe_Mounting
Blade_Type
Travel_Controls
Differential_Type
Steering_Controls
```

In [77]:
```
#Turn categorical data into numeric and fill missing values
for label,content in df_tmp.items():
    if not pd.api.types.is_numeric_dtype(content):
        #Addd binary column to indicate whether sample had missing values
        df_tmp[label+ "Is_missing"]= pd.isnull(content)
        # Tunr Categorical data into numbers and add +1
        df_tmp[label]= pd.Categorical(content).codes+1
```

In [78]:
```
df_tmp.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 412698 entries, 0 to 412697
Columns: 149 entries, SalesID to Steering_ControlsIs_missing
dtypes: bool(92), float64(3), int16(4), int64(10), int8(40)
memory usage: 96.0 MB
```

In [76]:
```
df_tmp.head().T
```

Out[76]:

	0	1	2	3	4
SalesID	1139246	1139248	1139249	1139251	1139253
SalePrice	66000.0	57000.0	10000.0	38500.0	11000.0
MachineID	999089	117657	434808	1026470	1057373
ModelID	3157	77	7009	332	17311
datasource	121	121	121	121	121
...	...	...	...	...	...
Backhoe_Mountingls_missing	True	True	True	True	True
Blade_Typels_missing	True	True	True	True	True
Travel_Controlsls_missing	True	True	True	True	True
Differential_Typels_missing	False	False	True	True	True
Steering_Controlsls_missing	False	False	True	True	True

149 rows × 5 columns

In [82]:
```
df_tmp.isna().sum()
```

Out[82]:
```
SalesID                       0
SalePrice                     0
MachineID                     0
ModelID                       0
datasource                    0
                             ..
Backhoe_MountingIs_missing    0
Blade_TypeIs_missing          0
Travel_ControlsIs_missing     0
Differential_TypeIs_missing   0
Steering_ControlsIs_missing   0
Length: 149, dtype: int64
```

In [ ]:
```
```

In [85]:
```
%%time

from sklearn.ensemble import RandomForestRegressor

#instantiate model
model= RandomForestRegressor(n_jobs=-1, random_state=42)

#Fit model
model.fit(df_tmp.drop("SalePrice",axis=1),df_tmp["SalePrice"] )
```

```
CPU times: total: 30min 1s
Wall time: 8min 47s
```

Out[85]:
```
RandomForestRegressor
RandomForestRegressor(n_jobs=-1, random_state=42)
```

In [86]:
```
#Score the model
model.score(df_tmp.drop("SalePrice",axis=1),df_tmp["SalePrice"] )
```

Out[86]:
```
0.987550144377029
```

Question: Why doesn't the above metric hold water(why is it not reliable)

Splitting Data into Training and Validation Set

Kaggle gave the validation set already, but we're going to create our own. Our validation set should be data from 2012 while our train set contains data from every other year except 2012

In [90]:
```
df_tmp.saleYear.value_counts()
```

Out[90]:
```
2009    43849
2008    39767
2011    35197
2010    33390
2007    32208
2006    21685
2005    20463
2004    19879
2001    17594
2000    17415
2002    17246
2003    15254
1998    13046
1999    12793
2012    11573
1997     9785
1996     8829
1995     8530
1994     7929
1993     6303
1992     5519
1991     5109
1989     4806
1990     4529
Name: saleYear, dtype: int64
```

In [88]:
```
#Split into validation and train set
df_val= df_tmp[df_tmp.saleYear==2012]
df_train= df_tmp[df_tmp.saleYear != 2012]

len(df_val), len(df_train)
```

Out[88]:
```
(11573, 401125)
```

In [92]:
```
#Split data into X and Y
x_train,y_train= df_train.drop("SalePrice", axis=1), df_train.SalePrice
x_valid,y_valid= df_val.drop("SalePrice", axis=1), df_val.SalePrice

x_train.shape,y_train.shape,x_valid.shape,y_valid.shape
```

Out[92]:
```
((401125, 148), (401125,), (11573, 148), (11573,))
```

Building an Evaluation Function

In [101…
```
#Create an evaluation function
from sklearn.metrics import mean_squared_log_error, mean_absolute_error, r2_score

def rmsle(y_test, y_preds):
    """
    Calculate the mean squared log error between prediction and true labels
    """
    return np.sqrt(mean_squared_log_error(y_test,y_preds))

#Create function to evaluate metrics on a few more level
def show_scores(model):
    train_preds= model.predict(x_train)
    val_preds= model.predict(x_valid)

    score= {"Training MAE":mean_absolute_error(y_train,train_preds),
            "Valid MAE":mean_absolute_error(y_valid,val_preds),
            "Training RMSLE": mean_squared_log_error(y_train, train_preds),
            "Valid RMSLE": mean_squared_log_error(y_valid, val_preds),
            "Training r2": r2_score(y_train,train_preds),
            "Valid r2": r2_score(y_valid, val_preds)}
    return(score)
```

Testing our model on a subset (to tune hyperparameter)

In [96]:
```
# Our model is going to take too long to run because of the large amount of data, so we'll reduce the data
model= RandomForestRegressor(n_jobs=-1,
                             random_state=42,
                             max_samples=10000)
```

In [97]:
```
%%time
#Cutting down the maximum number of samples can help fasten the run time
model.fit(x_train,y_train)
```

```
CPU times: total: 1min 27s
Wall time: 32.7 s
```

Out[97]:
```
RandomForestRegressor(max_samples=10000, n_jobs=-1, random_state=42)
```

In [102…
```
show_scores(model)
```

Out[102]:
```
{'Training MAE': 5551.885231860392,
 'Valid MAE': 7143.059586105591,
 'Training RMSLE': 0.06628094748199449,
 'Valid RMSLE': 0.08546347222620242,
 'Training r2': 0.8609502198295116,
 'Valid r2': 0.8332715810983576}
```

Randomized SearchCV

In [138…
```
%%time
from sklearn.model_selection import RandomizedSearchCV

#Different RandomForest Hyperparameter
rf_grid= {"n_estimators":np.arange(10,100,10),
          "max_depth":[None,3,5,10],
          "min_samples_split":np.arange(2,20,2),
          "min_samples_leaf":np.arange(1,20,2),
          "max_features":[0.5,1,"sqrt","auto"],
          "max_samples":[10000]}

#Instantiate RAndpmizedSearchCV model
rs_model= RandomizedSearchCV(RandomForestRegressor(n_jobs=-1,
                                                   random_state=42),
                             param_distributions=rf_grid,
                             n_iter=2,
                             cv=5,
                             verbose=True)
#Fit RandomizedSearchCV
rs_model.fit(x_train,y_train)
```

```
Fitting 5 folds for each of 2 candidates, totalling 10 fits
```

```
C:\Users\Aboya\anaconda3\lib\site-packages\sklearn\ensemble\_forest.py:416: FutureWarning: `max_features='auto'
` has been deprecated in 1.1 and will be removed in 1.3. To keep the past behaviour, explicitly set `max_featur
es=1.0` or remove this parameter as it is also the default value for RandomForestRegressors and ExtraTreesRegre
ssors.
  warn(
C:\Users\Aboya\anaconda3\lib\site-packages\sklearn\ensemble\_forest.py:416: FutureWarning: `max_features='auto'
` has been deprecated in 1.1 and will be removed in 1.3. To keep the past behaviour, explicitly set `max_featur
es=1.0` or remove this parameter as it is also the default value for RandomForestRegressors and ExtraTreesRegre
ssors.
  warn(
C:\Users\Aboya\anaconda3\lib\site-packages\sklearn\ensemble\_forest.py:416: FutureWarning: `max_features='auto'
` has been deprecated in 1.1 and will be removed in 1.3. To keep the past behaviour, explicitly set `max_featur
es=1.0` or remove this parameter as it is also the default value for RandomForestRegressors and ExtraTreesRegre
ssors.
  warn(
C:\Users\Aboya\anaconda3\lib\site-packages\sklearn\ensemble\_forest.py:416: FutureWarning: `max_features='auto'
` has been deprecated in 1.1 and will be removed in 1.3. To keep the past behaviour, explicitly set `max_featur
es=1.0` or remove this parameter as it is also the default value for RandomForestRegressors and ExtraTreesRegre
ssors.
  warn(
C:\Users\Aboya\anaconda3\lib\site-packages\sklearn\ensemble\_forest.py:416: FutureWarning: `max_features='auto'
` has been deprecated in 1.1 and will be removed in 1.3. To keep the past behaviour, explicitly set `max_featur
es=1.0` or remove this parameter as it is also the default value for RandomForestRegressors and ExtraTreesRegre
ssors.
  warn(
C:\Users\Aboya\anaconda3\lib\site-packages\sklearn\ensemble\_forest.py:416: FutureWarning: `max_features='auto'
` has been deprecated in 1.1 and will be removed in 1.3. To keep the past behaviour, explicitly set `max_featur
es=1.0` or remove this parameter as it is also the default value for RandomForestRegressors and ExtraTreesRegre
ssors.
  warn(
CPU times: total: 1min 8s
Wall time: 3min 13s
```

Out[138]:

```
▸ RandomizedSearchCV
▸ estimator: RandomForestRegressor
  ▸ RandomForestRegressor
```

In [139…

```
rs_model.best_params_
```

Out[139]:

```
{'n_estimators': 20,
 'min_samples_split': 2,
 'min_samples_leaf': 13,
 'max_samples': 10000,
 'max_features': 'auto',
 'max_depth': 10}
```

In [140…

```
show_scores(rs_model)
```

Out[140]:

```
{'Training MAE': 6749.7833169545265,
 'Valid MAE': 8265.474471853617,
 'Training RMSLE': 0.09067018890096433,
 'Valid RMSLE': 0.1083052133574266,
 'Training r2': 0.7999779623787006,
 'Valid r2': 0.7713028711277398}
```

Train a model with the best hyperparameters

Note: These parameters were gotten after 100 iterations of the RandomizedSearchCV

In [142…

```
%%time
#Most ideal hyperparameter
ideal_model= RandomForestRegressor(n_jobs=-1,
                                   n_estimators=40,
                                   min_samples_split=14,
                                   min_samples_leaf=1,
                                   max_features=0.5,
                                   max_samples=None,
                                   random_state=42)


#fit Model
ideal_model.fit(x_train,y_train)
```

```
CPU times: total: 5min 5s
Wall time: 1min 33s
```

Out[142]:

```
▾ RandomForestRegressor
RandomForestRegressor(max_features=0.5, min_samples_split=14, n_estimators=40,
                      n_jobs=-1)
```

In [144…

```
show_scores(ideal_model)
```

Out[144]:
```
{'Training MAE': 2953.1450802995564,
 'Valid MAE': 5969.8586758528445,
 'Training RMSLE': 0.020924803520354633,
 'Valid RMSLE': 0.06066059633685338,
 'Training r2': 0.9589479331157784,
 'Valid r2': 0.8812601397663162}
```

Make predictions on the Test dataset

In [154…
```
df=pd.read_csv("Test.csv", parse_dates=["saledate"])
df.head()
```

Out[154]:

	SalesID	MachineID	ModelID	datasource	auctioneerID	YearMade	MachineHoursCurrentMeter	UsageBand	saledate	fiModelDesc	...	Ur
0	1227829	1006309	3168	121	3	1999	3688.0	Low	2012-05-03	580G	...	
1	1227844	1022817	7271	121	3	1000	28555.0	High	2012-05-10	936	...	
2	1227847	1031560	22805	121	3	2004	6038.0	Medium	2012-05-10	EC210BLC	...	
3	1227848	56204	1269	121	3	2006	8940.0	High	2012-05-10	330CL	...	
4	1227863	1053887	22312	121	3	2005	2286.0	Low	2012-05-10	650K	...	

5 rows × 52 columns

In [155…
```
df1= df.copy()
```

In [202…
```
#Make prediction on the Test dataset
test_preds= ideal_model.predict(df1)
```

```
C:\Users\Aboya\anaconda3\lib\site-packages\sklearn\base.py:493: FutureWarning: The feature names should match t
hose that were passed during fit. Starting version 1.2, an error will be raised.
Feature names unseen at fit time:
- Backhoe_Mountingmiss
- Blade_Extensionmiss
- Blade_Typemiss
- Blade_Widthmiss
- Coupler_Systemmiss
- ...
Feature names seen at fit time, yet now missing:
- Backhoe_MountingIs_missing
- Backhoe_Mountingis_missing
- Blade_ExtensionIs_missing
- Blade_Extensionis_missing
- Blade_TypeIs_missing
- ...

  warnings.warn(message, FutureWarning)
```

```
---------------------------------------------------------------------------
ValueError                                Traceback (most recent call last)
Input In [202], in <cell line: 2>()
      1 #Make prediction on the Test dataset
----> 2 test_preds= ideal_model.predict(df1)

File ~\anaconda3\lib\site-packages\sklearn\ensemble\_forest.py:991, in ForestRegressor.predict(self, X)
    989 check_is_fitted(self)
    990 # Check data
--> 991 X = self._validate_X_predict(X)
    993 # Assign chunk of trees to jobs
    994 n_jobs, _, _ = _partition_estimators(self.n_estimators, self.n_jobs)

File ~\anaconda3\lib\site-packages\sklearn\ensemble\_forest.py:605, in BaseForest._validate_X_predict(self, X)
    602 """
    603 Validate X whenever one tries to predict, apply, predict_proba."""
    604 check_is_fitted(self)
--> 605 X = self._validate_data(X, dtype=DTYPE, accept_sparse="csr", reset=False)
    606 if issparse(X) and (X.indices.dtype != np.intc or X.indptr.dtype != np.intc):
    607     raise ValueError("No support for np.int64 index based sparse matrices")

File ~\anaconda3\lib\site-packages\sklearn\base.py:600, in BaseEstimator._validate_data(self, X, y, reset, vali
date_separately, **check_params)
    597     out = X, y
    599 if not no_val_X and check_params.get("ensure_2d", True):
--> 600     self._check_n_features(X, reset=reset)
    602 return out

File ~\anaconda3\lib\site-packages\sklearn\base.py:400, in BaseEstimator._check_n_features(self, X, reset)
    397     return
    399 if n_features != self.n_features_in_:
--> 400     raise ValueError(
    401         f"X has {n_features} features, but {self.__class__.__name__} "
    402         f"is expecting {self.n_features_in_} features as input."
    403     )

ValueError: X has 101 features, but RandomForestRegressor is expecting 148 features as input.
```

Preprocessing our data(making our test dataset in the same format as our training dataset)

In [156…
```
df1.head()
```

Out[156]:

	SalesID	MachineID	ModelID	datasource	auctioneerID	YearMade	MachineHoursCurrentMeter	UsageBand	saledate	fiModelDesc	...	Ur
0	1227829	1006309	3168	121	3	1999	3688.0	Low	2012-05-03	580G	...	
1	1227844	1022817	7271	121	3	1000	28555.0	High	2012-05-10	936	...	
2	1227847	1031560	22805	121	3	2004	6038.0	Medium	2012-05-10	EC210BLC	...	
3	1227848	56204	1269	121	3	2006	8940.0	High	2012-05-10	330CL	...	
4	1227863	1053887	22312	121	3	2005	2286.0	Low	2012-05-10	650K	...	

5 rows × 52 columns

In [157…
```
df1["saleYear"]= df1.saledate.dt.year
df1["saleMonth"]= df1.saledate.dt.month
df1["saleDay"]= df1.saledate.dt.day
df1["saleDayOfWeek"]= df1.saledate.dt.dayofweek
df1["saleDayOfYear"]= df1.saledate.dt.dayofyear
```

In [178…
```
df1.drop("saledate",axis=1,inplace=True)
```

In [179…
```
df1.head().T
```

Out[179]:

	0	1	2	3	4
SalesID	1227829	1227844	1227847	1227848	1227863
MachineID	1006309	1022817	1031560	56204	1053887
ModelID	3168	7271	22805	1269	22312
datasource	121	121	121	121	121
auctioneerID	3	3	3	3	3
YearMade	1999	1000	2004	2006	2005
MachineHoursCurrentMeter	3688.0	28555.0	6038.0	8940.0	2286.0
UsageBand	Low	High	Medium	High	Low
fiModelDesc	580G	936	EC210BLC	330CL	650K

fiBaseModel	580	936	EC210	330	650
fiSecondaryDesc	G	NaN	B	C	K
fiModelSeries	NaN	NaN	NaN	NaN	NaN
fiModelDescriptor	NaN	NaN	LC	L	NaN
ProductSize	NaN	Medium	Large / Medium	Large / Medium	NaN
fiProductClassDesc	Backhoe Loader - 14.0 to 15.0 Ft Standard Digg...	Wheel Loader - 135.0 to 150.0 Horsepower	Hydraulic Excavator, Track - 21.0 to 24.0 Metr...	Hydraulic Excavator, Track - 33.0 to 40.0 Metr...	Track Type Tractor, Dozer - 20.0 to 75.0 Horse...
state	Wyoming	Virginia	New Jersey	New Jersey	Florida
ProductGroup	BL	WL	TEX	TEX	TTT
ProductGroupDesc	Backhoe Loaders	Wheel Loader	Track Excavators	Track Excavators	Track Type Tractors
Drive_System	Two Wheel Drive	NaN	NaN	NaN	NaN
Enclosure	OROPS	EROPS	EROPS w AC	EROPS w AC	OROPS
Forks	Yes	Yes	NaN	NaN	NaN
Pad_Type	None or Unspecified	NaN	NaN	NaN	NaN
Ride_Control	No	None or Unspecified	NaN	NaN	NaN
Stick	Standard	NaN	NaN	NaN	NaN
Transmission	Standard	NaN	NaN	NaN	Hydrostatic
Turbocharged	None or Unspecified	NaN	NaN	NaN	NaN
Blade_Extension	NaN	NaN	NaN	NaN	NaN
Blade_Width	NaN	NaN	NaN	NaN	NaN
Enclosure_Type	NaN	NaN	NaN	NaN	NaN
Engine_Horsepower	NaN	NaN	NaN	NaN	NaN
Hydraulics	NaN	2 Valve	Auxiliary	Standard	2 Valve
Pushblock	NaN	NaN	NaN	NaN	NaN
Ripper	NaN	NaN	NaN	NaN	None or Unspecified
Scarifier	NaN	NaN	NaN	NaN	NaN
Tip_Control	NaN	NaN	NaN	NaN	NaN
Tire_Size	NaN	20.5	NaN	NaN	NaN
Coupler	NaN	None or Unspecified	None or Unspecified	None or Unspecified	NaN
Coupler_System	NaN	NaN	NaN	NaN	NaN
Grouser_Tracks	NaN	NaN	NaN	NaN	NaN
Hydraulics_Flow	NaN	NaN	NaN	NaN	NaN
Track_Type	NaN	NaN	Steel	Steel	NaN
Undercarriage_Pad_Width	NaN	NaN	None or Unspecified	None or Unspecified	NaN
Stick_Length	NaN	NaN	9' 6"	None or Unspecified	NaN
Thumb	NaN	NaN	Manual	Manual	NaN
Pattern_Changer	NaN	NaN	None or Unspecified	Yes	NaN
Grouser_Type	NaN	NaN	Double	Triple	NaN
Backhoe_Mounting	NaN	NaN	NaN	NaN	None or Unspecified
Blade_Type	NaN	NaN	NaN	NaN	PAT
Travel_Controls	NaN	NaN	NaN	NaN	None or Unspecified
Differential_Type	NaN	Standard	NaN	NaN	NaN
Steering_Controls	NaN	Conventional	NaN	NaN	NaN
saleYear	2012	2012	2012	2012	2012
saleMonth	5	5	5	5	5
saleDay	3	10	10	10	10
saleDayOfWeek	3	3	3	3	3
saleDayOfYear	124	131	131	131	131

In [180…

```
df1.head(5)
```

Out[180]:

	SalesID	MachineID	ModelID	datasource	auctioneerID	YearMade	MachineHoursCurrentMeter	UsageBand	fiModelDesc	fiBaseModel	...
0	1227829	1006309	3168	121	3	1999	3688.0	Low	580G	580	...
1	1227844	1022817	7271	121	3	1000	28555.0	High	936	936	...
2	1227847	1031560	22805	121	3	2004	6038.0	Medium	EC210BLC	EC210	...
3	1227848	56204	1269	121	3	2006	8940.0	High	330CL	330	...
4	1227863	1053887	22312	121	3	2005	2286.0	Low	650K	650	...

5 rows × 56 columns

In [181…
```
pd.api.types.is_categorical_dtype(df1.YearMade)
```

Out[181]:
```
False
```

In [182…
```
df1.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 12457 entries, 0 to 12456
Data columns (total 56 columns):
 #   Column                    Non-Null Count  Dtype  
---  ------                    --------------  -----  
 0   SalesID                   12457 non-null  int64  
 1   MachineID                 12457 non-null  int64  
 2   ModelID                   12457 non-null  int64  
 3   datasource                12457 non-null  int64  
 4   auctioneerID              12457 non-null  int64  
 5   YearMade                  12457 non-null  int64  
 6   MachineHoursCurrentMeter  2129 non-null   float64
 7   UsageBand                 1834 non-null   object 
 8   fiModelDesc               12457 non-null  object 
 9   fiBaseModel               12457 non-null  object 
 10  fiSecondaryDesc           8482 non-null   object 
 11  fiModelSeries             2006 non-null   object 
 12  fiModelDescriptor         3024 non-null   object 
 13  ProductSize               6048 non-null   object 
 14  fiProductClassDesc        12457 non-null  object 
 15  state                     12457 non-null  object 
 16  ProductGroup              12457 non-null  object 
 17  ProductGroupDesc          12457 non-null  object 
 18  Drive_System              2759 non-null   object 
 19  Enclosure                 12455 non-null  object 
 20  Forks                     6308 non-null   object 
 21  Pad_Type                  2108 non-null   object 
 22  Ride_Control              4241 non-null   object 
 23  Stick                     2108 non-null   object 
 24  Transmission              4818 non-null   object 
 25  Turbocharged              2108 non-null   object 
 26  Blade_Extension           651 non-null    object 
 27  Blade_Width               651 non-null    object 
 28  Enclosure_Type            651 non-null    object 
 29  Engine_Horsepower         651 non-null    object 
 30  Hydraulics                10315 non-null  object 
 31  Pushblock                 651 non-null    object 
 32  Ripper                    2704 non-null   object 
 33  Scarifier                 651 non-null    object 
 34  Tip_Control               651 non-null    object 
 35  Tire_Size                 2778 non-null   object 
 36  Coupler                   7601 non-null   object 
 37  Coupler_System            2066 non-null   object 
 38  Grouser_Tracks            2066 non-null   object 
 39  Hydraulics_Flow           2066 non-null   object 
 40  Track_Type                3394 non-null   object 
 41  Undercarriage_Pad_Width   3398 non-null   object 
 42  Stick_Length              3394 non-null   object 
 43  Thumb                     3395 non-null   object 
 44  Pattern_Changer           3394 non-null   object 
 45  Grouser_Type              3394 non-null   object 
 46  Backhoe_Mounting          2051 non-null   object 
 47  Blade_Type                2058 non-null   object 
 48  Travel_Controls           2058 non-null   object 
 49  Differential_Type         2129 non-null   object 
 50  Steering_Controls         2129 non-null   object 
 51  saleYear                  12457 non-null  int64  
 52  saleMonth                 12457 non-null  int64  
 53  saleDay                   12457 non-null  int64  
 54  saleDayOfWeek             12457 non-null  int64  
 55  saleDayOfYear             12457 non-null  int64  
dtypes: float64(1), int64(11), object(44)
memory usage: 5.3+ MB
```

In [183…
```
for label,content in df1.items():
    if pd.api.types.is_string_dtype(content):
        print(label)
```

```
UsageBand
fiModelDesc
fiBaseModel
fiSecondaryDesc
fiModelSeries
fiModelDescriptor
ProductSize
fiProductClassDesc
state
ProductGroup
ProductGroupDesc
Drive_System
Enclosure
Forks
Pad_Type
Ride_Control
Stick
Transmission
Turbocharged
Blade_Extension
Blade_Width
Enclosure_Type
Engine_Horsepower
Hydraulics
Pushblock
Ripper
Scarifier
Tip_Control
Tire_Size
Coupler
Coupler_System
Grouser_Tracks
Hydraulics_Flow
Track_Type
Undercarriage_Pad_Width
Stick_Length
Thumb
Pattern_Changer
Grouser_Type
Backhoe_Mounting
Blade_Type
Travel_Controls
Differential_Type
Steering_Controls
```

In [190…

```
#Convert String data types to categorical dtype
for label,content in df1.items():
    if pd.api.types.is_string_dtype(content):
        df1[label]= content.astype("category").cat.as_ordered()
```

In [191…

```
df1.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 12457 entries, 0 to 12456
Data columns (total 56 columns):
 #   Column                    Non-Null Count  Dtype
---  ------                    --------------  -----
 0   SalesID                   12457 non-null  int64
 1   MachineID                 12457 non-null  int64
 2   ModelID                   12457 non-null  int64
 3   datasource                12457 non-null  int64
 4   auctioneerID              12457 non-null  int64
 5   YearMade                  12457 non-null  int64
 6   MachineHoursCurrentMeter  2129 non-null   float64
 7   UsageBand                 1834 non-null   category
 8   fiModelDesc               12457 non-null  category
 9   fiBaseModel               12457 non-null  category
 10  fiSecondaryDesc           8482 non-null   category
 11  fiModelSeries             2006 non-null   category
 12  fiModelDescriptor         3024 non-null   category
 13  ProductSize               6048 non-null   category
 14  fiProductClassDesc        12457 non-null  category
 15  state                     12457 non-null  category
 16  ProductGroup              12457 non-null  category
 17  ProductGroupDesc          12457 non-null  category
 18  Drive_System              2759 non-null   category
 19  Enclosure                 12455 non-null  category
 20  Forks                     6308 non-null   category
 21  Pad_Type                  2108 non-null   category
 22  Ride_Control              4241 non-null   category
 23  Stick                     2108 non-null   category
 24  Transmission              4818 non-null   category
 25  Turbocharged              2108 non-null   category
 26  Blade_Extension           651 non-null    category
 27  Blade_Width               651 non-null    category
 28  Enclosure_Type            651 non-null    category
 29  Engine_Horsepower         651 non-null    category
 30  Hydraulics                10315 non-null  category
 31  Pushblock                 651 non-null    category
 32  Ripper                    2704 non-null   category
 33  Scarifier                 651 non-null    category
 34  Tip_Control               651 non-null    category
 35  Tire_Size                 2778 non-null   category
 36  Coupler                   7601 non-null   category
 37  Coupler_System            2066 non-null   category
 38  Grouser_Tracks            2066 non-null   category
 39  Hydraulics_Flow           2066 non-null   category
 40  Track_Type                3394 non-null   category
 41  Undercarriage_Pad_Width   3398 non-null   category
 42  Stick_Length              3394 non-null   category
 43  Thumb                     3395 non-null   category
 44  Pattern_Changer           3394 non-null   category
 45  Grouser_Type              3394 non-null   category
 46  Backhoe_Mounting          2051 non-null   category
 47  Blade_Type                2058 non-null   category
 48  Travel_Controls           2058 non-null   category
 49  Differential_Type         2129 non-null   category
 50  Steering_Controls         2129 non-null   category
 51  saleYear                  12457 non-null  int64
 52  saleMonth                 12457 non-null  int64
 53  saleDay                   12457 non-null  int64
 54  saleDayOfWeek             12457 non-null  int64
 55  saleDayOfYear             12457 non-null  int64
dtypes: category(44), float64(1), int64(11)
memory usage: 1.8 MB
```

Fill missing data

Fill numeric missing data first

In [195…
```
for label,content in df1.items():
    if pd.api.types.is_numeric_dtype(content):
        if pd.isnull(content).sum():
            print(label)
```

```
MachineHoursCurrentMeter
```

In [196…
```
for label,content in df1.items():
    if pd.api.types.is_numeric_dtype(content):
        if pd.isnull(content).sum():
            df1[label + "Is_missing"]= pd.isnull(content)
            df1[label]= content.fillna(content.median())
```

In [197…
```
for label,content in df1.items():
    if pd.api.types.is_numeric_dtype(content):
        if pd.isnull(content).sum():
            print(label)
```

Convert Numerical data and fill missing data

In [198…
```
#Check for categorical data
for label,content in df1.items():
    if not pd.api.types.is_numeric_dtype(content):
        print(label)
```

```
UsageBand
fiModelDesc
fiBaseModel
fiSecondaryDesc
fiModelSeries
fiModelDescriptor
ProductSize
fiProductClassDesc
state
ProductGroup
ProductGroupDesc
Drive_System
Enclosure
Forks
Pad_Type
Ride_Control
Stick
Transmission
Turbocharged
Blade_Extension
Blade_Width
Enclosure_Type
Engine_Horsepower
Hydraulics
Pushblock
Ripper
Scarifier
Tip_Control
Tire_Size
Coupler
Coupler_System
Grouser_Tracks
Hydraulics_Flow
Track_Type
Undercarriage_Pad_Width
Stick_Length
Thumb
Pattern_Changer
Grouser_Type
Backhoe_Mounting
Blade_Type
Travel_Controls
Differential_Type
Steering_Controls
```

In [200…
```
for label, content in df1.items():
    if not pd.api.types.is_numeric_dtype(content):
        df1[label + "miss"]= pd.isnull(content)
        df1[label]= pd.Categorical(content).codes+1
```

In [201…
```
df1.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 12457 entries, 0 to 12456
Columns: 101 entries, SalesID to Steering_Controlsmiss
dtypes: bool(45), float64(1), int16(2), int64(11), int8(42)
memory usage: 2.2 MB
```

In [274…
```
df1
```

Out[274]:

	SalesID	MachineID	ModelID	datasource	auctioneerID	YearMade	MachineHoursCurrentMeter	UsageBand	fiModelDesc	fiBaseModel
0	1227829	1006309	3168	121	3	1999	3688.0	2	499	180
1	1227844	1022817	7271	121	3	1000	28555.0	1	831	292
2	1227847	1031560	22805	121	3	2004	6038.0	3	1177	404
3	1227848	56204	1269	121	3	2006	8940.0	1	287	113
4	1227863	1053887	22312	121	3	2005	2286.0	2	566	196
...	...	...	...	...	...	...	...	...	...	...
12452	6643171	2558317	21450	149	2	2008	3525.0	0	713	235
12453	6643173	2558332	21434	149	2	2005	3525.0	0	186	80
12454	6643184	2558342	21437	149	2	1000	3525.0	0	325	123
12455	6643186	2558343	21437	149	2	2006	3525.0	0	325	123
12456	6643196	2558346	21446	149	2	2008	3525.0	0	483	171

12457 rows × 101 columns

Writing a simple function for all the code above

In [271…

```
data=pd.read_csv("Test.csv", parse_dates=["saledate"])
data.head()
```

Out[271]:

	SalesID	MachineID	ModelID	datasource	auctioneerID	YearMade	MachineHoursCurrentMeter	UsageBand	saledate	fiModelDesc	...	Ur
0	1227829	1006309	3168	121	3	1999	3688.0	Low	2012-05-03	580G	...	
1	1227844	1022817	7271	121	3	1000	28555.0	High	2012-05-10	936	...	
2	1227847	1031560	22805	121	3	2004	6038.0	Medium	2012-05-10	EC210BLC	...	
3	1227848	56204	1269	121	3	2006	8940.0	High	2012-05-10	330CL	...	
4	1227863	1053887	22312	121	3	2005	2286.0	Low	2012-05-10	650K	...	

5 rows × 52 columns

In [272…

```
def preprocessing(data):
    """
    performs transformation of data and returns transformed data
    """
    data["saleYear"]= data.saledate.dt.year
    data["saleMonth"]= data.saledate.dt.month
    data["saleDay"]= data.saledate.dt.day
    data["saleDayOfWeek"]= data.saledate.dt.dayofweek
    data["saleDayOfYear"]= data.saledate.dt.dayofyear
    data.drop("saledate",axis=1,inplace=True)

    #turn string data into category
    for label,content in data.items():
        if pd.api.types.is_string_dtype(content):
            data[label]= content.astype("category").cat.as_ordered()

    #Fill missing numeric data with Median
    for label,content in data.items():
        if pd.api.types.is_numeric_dtype(content):
            if pd.isnull(content).sum():
                data[label + "Is_missing"]= pd.isnull(content)
                data[label]= content.fillna(content.median())

    #Fill categorical missing data into it into numbers
    for label, content in data.items():
        if not pd.api.types.is_numeric_dtype(content):
            data[label + "miss"]= pd.isnull(content)
            data[label]= pd.Categorical(content).codes+1

    return(data)
```

In [273…

```
df_test=preprocessing(data)
df_test
```

Out[273]:

	SalesID	MachineID	ModelID	datasource	auctioneerID	YearMade	MachineHoursCurrentMeter	UsageBand	fiModelDesc	fiBaseModel
0	1227829	1006309	3168	121	3	1999	3688.0	2	499	180
1	1227844	1022817	7271	121	3	1000	28555.0	1	831	292
2	1227847	1031560	22805	121	3	2004	6038.0	3	1177	404
3	1227848	56204	1269	121	3	2006	8940.0	1	287	113
4	1227863	1053887	22312	121	3	2005	2286.0	2	566	196
...	...	...	...	...	...	...	...	...	...	...
12452	6643171	2558317	21450	149	2	2008	3525.0	0	713	235
12453	6643173	2558332	21434	149	2	2005	3525.0	0	186	80
12454	6643184	2558342	21437	149	2	1000	3525.0	0	325	123
12455	6643186	2558343	21437	149	2	2006	3525.0	0	325	123
12456	6643196	2558346	21446	149	2	2008	3525.0	0	483	171

12457 rows × 101 columns

In [275…
```
test_preds= ideal_model.predict(df_test)
```

```
C:\Users\Aboya\anaconda3\lib\site-packages\sklearn\base.py:493: FutureWarning: The feature names should match t
hose that were passed during fit. Starting version 1.2, an error will be raised.
Feature names unseen at fit time:
- Backhoe_Mountingmiss
- Blade_Extensionmiss
- Blade_Typemiss
- Blade_Widthmiss
- Coupler_Systemmiss
- ...
Feature names seen at fit time, yet now missing:
- Backhoe_MountingIs_missing
- Backhoe_Mountingis_missing
- Blade_ExtensionIs_missing
- Blade_Extensionis_missing
- Blade_TypeIs_missing
- ...

  warnings.warn(message, FutureWarning)
---------------------------------------------------------------------------
ValueError                                Traceback (most recent call last)
Input In [275], in <cell line: 1>()
----> 1 test_preds= ideal_model.predict(df_test)

File ~\anaconda3\lib\site-packages\sklearn\ensemble\_forest.py:991, in ForestRegressor.predict(self, X)
    989 check_is_fitted(self)
    990 # Check data
--> 991 X = self._validate_X_predict(X)
    993 # Assign chunk of trees to jobs
    994 n_jobs, _, _ = _partition_estimators(self.n_estimators, self.n_jobs)

File ~\anaconda3\lib\site-packages\sklearn\ensemble\_forest.py:605, in BaseForest._validate_X_predict(self, X)
    602 """
    603 Validate X whenever one tries to predict, apply, predict_proba."""
    604 check_is_fitted(self)
--> 605 X = self._validate_data(X, dtype=DTYPE, accept_sparse="csr", reset=False)
    606 if issparse(X) and (X.indices.dtype != np.intc or X.indptr.dtype != np.intc):
    607     raise ValueError("No support for np.int64 index based sparse matrices")

File ~\anaconda3\lib\site-packages\sklearn\base.py:600, in BaseEstimator._validate_data(self, X, y, reset, vali
date_separately, **check_params)
    597     out = X, y
    599 if not no_val_X and check_params.get("ensure_2d", True):
--> 600     self._check_n_features(X, reset=reset)
    602 return out

File ~\anaconda3\lib\site-packages\sklearn\base.py:400, in BaseEstimator._check_n_features(self, X, reset)
    397     return
    399 if n_features != self.n_features_in_:
--> 400     raise ValueError(
    401         f"X has {n_features} features, but {self.__class__.__name__} "
    402         f"is expecting {self.n_features_in_} features as input."
    403     )

ValueError: X has 101 features, but RandomForestRegressor is expecting 148 features as input.
```

In [276…
```
x_train.head()
```

Out[276]:

	SalesID	MachineID	ModelID	datasource	auctioneerID	YearMade	MachineHoursCurrentMeter	UsageBand	fiModelDesc	fiBaseModel	...
0	1139246	999089	3157	121	3.0	2004	68.0	2	963	298	...
1	1139248	117657	77	121	3.0	1996	4640.0	2	1745	529	...
2	1139249	434808	7009	121	3.0	2001	2838.0	1	336	111	...
3	1139251	1026470	332	121	3.0	2001	3486.0	1	3716	1381	...
4	1139253	1057373	17311	121	3.0	2007	722.0	3	4261	1538	...

5 rows × 148 columns

In [ ]:

In [277…

```
# Import the data again, but this time , parse date
df= pd.read_csv("TrainAndValid.csv", low_memory=False, parse_dates= ["saledate"])
df
```

Out[277]:

	SalesID	SalePrice	MachineID	ModelID	datasource	auctioneerID	YearMade	MachineHoursCurrentMeter	UsageBand	saledate	...
0	1139246	66000.0	999089	3157	121	3.0	2004	68.0	Low	2006-11-16	...
1	1139248	57000.0	117657	77	121	3.0	1996	4640.0	Low	2004-03-26	...
2	1139249	10000.0	434808	7009	121	3.0	2001	2838.0	High	2004-02-26	...
3	1139251	38500.0	1026470	332	121	3.0	2001	3486.0	High	2011-05-19	...
4	1139253	11000.0	1057373	17311	121	3.0	2007	722.0	Medium	2009-07-23	...
...	...	...	...	...	...	...	...	...	...	...	...
412693	6333344	10000.0	1919201	21435	149	2.0	2005	NaN	NaN	2012-03-07	...
412694	6333345	10500.0	1882122	21436	149	2.0	2005	NaN	NaN	2012-01-28	...
412695	6333347	12500.0	1944213	21435	149	2.0	2005	NaN	NaN	2012-01-28	...
412696	6333348	10000.0	1794518	21435	149	2.0	2006	NaN	NaN	2012-03-07	...
412697	6333349	13000.0	1944743	21436	149	2.0	2006	NaN	NaN	2012-01-28	...

412698 rows × 53 columns

In [278…

```
def preprocessing(df):
    """
    performs transformation of data and returns transformed data
    """
    df["saleYear"]= df.saledate.dt.year
    df["saleMonth"]= df.saledate.dt.month
    df["saleDay"]= df.saledate.dt.day
    df["saleDayOfWeek"]= df.saledate.dt.dayofweek
    df["saleDayOfYear"]= df.saledate.dt.dayofyear
    df.drop("saledate",axis=1,inplace=True)

    #turn string data into category
    for label,content in df.items():
        if pd.api.types.is_string_dtype(content):
            df[label]= content.astype("category").cat.as_ordered()

    #Fill missing numeric data with Median
    for label,content in df.items():
        if pd.api.types.is_numeric_dtype(content):
            if pd.isnull(content).sum():
                df[label + "Is_missing"]= pd.isnull(content)
                df[label]= content.fillna(content.median())

    #Fill categorical missing data into it into numbers
    for label, content in df.items():
        if not pd.api.types.is_numeric_dtype(content):
            df[label + "miss"]= pd.isnull(content)
            df[label]= pd.Categorical(content).codes+1

    return(df)
```

In [279…

```
dff= preprocessing(df)
dff
```

Out[279]:

	SalesID	SalePrice	MachineID	ModelID	datasource	auctioneerID	YearMade	MachineHoursCurrentMeter	UsageBand	fiModelDesc
0	1139246	66000.0	999089	3157	121	3.0	2004	68.0	2	963
1	1139248	57000.0	117657	77	121	3.0	1996	4640.0	2	1745
2	1139249	10000.0	434808	7009	121	3.0	2001	2838.0	1	336
3	1139251	38500.0	1026470	332	121	3.0	2001	3486.0	1	3716
4	1139253	11000.0	1057373	17311	121	3.0	2007	722.0	3	4261
...	...	...	...	...	...	...	...	...	...	...
412693	6333344	10000.0	1919201	21435	149	2.0	2005	0.0	0	490
412694	6333345	10500.0	1882122	21436	149	2.0	2005	0.0	0	491
412695	6333347	12500.0	1944213	21435	149	2.0	2005	0.0	0	490
412696	6333348	10000.0	1794518	21435	149	2.0	2006	0.0	0	490
412697	6333349	13000.0	1944743	21436	149	2.0	2006	0.0	0	491

412698 rows × 103 columns

In [280…

```
%%time

from sklearn.ensemble import RandomForestRegressor

#instantiate model
model= RandomForestRegressor(n_jobs=-1, random_state=42)

#Fit model
model.fit(dff.drop("SalePrice",axis=1),dff["SalePrice"] )
```

```
CPU times: total: 26min
Wall time: 7min 50s
```

Out[280]:

```
RandomForestRegressor(n_jobs=-1, random_state=42)
```

In [ ]:

```
#Score the model
model.score(dff.drop("SalePrice",axis=1),df_tmp["SalePrice"] )
```

Question: Why doesn't the above metric hold water(why is it not reliable)

Splitting Data into Training and Validation Set

Kaggle gave the validation set already, but we're going to create our own. Our validation set should be data from 2012 while our train set contains data from every other year except 2012

In [282…

```
dff.saleYear.value_counts()
```

Out[282]:

```
2009    43849
2008    39767
2011    35197
2010    33390
2007    32208
2006    21685
2005    20463
2004    19879
2001    17594
2000    17415
2002    17246
2003    15254
1998    13046
1999    12793
2012    11573
1997     9785
1996     8829
1995     8530
1994     7929
1993     6303
1992     5519
1991     5109
1989     4806
1990     4529
Name: saleYear, dtype: int64
```

In [287…

```
#Split into validation and training dataset

dff_train= dff[dff.saleYear!= 2012]
dff_val= dff[dff.saleYear==2012]

len(dff_train), len(dff_val)
```

Out[287]: (401125, 11573)

In [293…
```
#Split data into X and Y
x_train,y_train= dff_train.drop("SalePrice", axis=1), dff_train.SalePrice
x_valid,y_valid= dff_val.drop("SalePrice", axis=1), dff_val.SalePrice

x_train.shape,y_train.shape,x_valid.shape,y_valid.shape
```

Out[293]: ((401125, 102), (401125,), (11573, 102), (11573,))

Building An Evaluation

In [294…
```
#Create an evaluation function
from sklearn.metrics import mean_squared_log_error, mean_absolute_error, r2_score

def rmsle(y_test, y_preds):
    """
    Calculate the mean squared log error between prediction and true labels
    """
    return np.sqrt(mean_squared_log_error(y_test,y_preds))

#Create function to evaluate metrics on a few more level
def show_scores(model):
    train_preds= model.predict(x_train)
    val_preds= model.predict(x_valid)

    score= {"Training MAE":mean_absolute_error(y_train,train_preds),
            "Valid MAE":mean_absolute_error(y_valid,val_preds),
            "Training RMSLE": mean_squared_log_error(y_train, train_preds),
            "Valid RMSLE": mean_squared_log_error(y_valid, val_preds),
            "Training r2": r2_score(y_train,train_preds),
            "Valid r2": r2_score(y_valid, val_preds)}
    return(score)
```

In [295…
```
show_scores(model)
```

Out[295]:
```
{'Training MAE': 1576.5832503434087,
 'Valid MAE': 1861.035806273222,
 'Training RMSLE': 0.007069451838302368,
 'Valid RMSLE': 0.007582917695929614,
 'Training r2': 0.987556297245471,
 'Valid r2': 0.9870036782759568}
```

Hyperparameter Tuning

In [298…
```
from sklearn.model_selection import RandomizedSearchCV

#Different RandomForest Hyperparameter
rf_grid= {"n_estimators":np.arange(10,100,10),
          "max_depth":[None,3,5,10],
          "min_samples_split":np.arange(2,20,2),
          "min_samples_leaf":np.arange(1,20,2),
          "max_features":[0.5,1,"sqrt","auto"],
          "max_samples":[10000]}

rs_model= RandomizedSearchCV(RandomForestRegressor(n_jobs=-1,
                                                   random_state=42),
                             param_distributions= rf_grid,
                             verbose= True,
                             cv=5,
                             n_iter=2)

rs_model.fit(x_train,y_train)
```

```
Fitting 5 folds for each of 2 candidates, totalling 10 fits
```

```
C:\Users\Aboya\anaconda3\lib\site-packages\sklearn\ensemble\_forest.py:416: FutureWarning: `max_features='auto'
` has been deprecated in 1.1 and will be removed in 1.3. To keep the past behaviour, explicitly set `max_featur
es=1.0` or remove this parameter as it is also the default value for RandomForestRegressors and ExtraTreesRegre
ssors.
  warn(
C:\Users\Aboya\anaconda3\lib\site-packages\sklearn\ensemble\_forest.py:416: FutureWarning: `max_features='auto'
` has been deprecated in 1.1 and will be removed in 1.3. To keep the past behaviour, explicitly set `max_featur
es=1.0` or remove this parameter as it is also the default value for RandomForestRegressors and ExtraTreesRegre
ssors.
  warn(
C:\Users\Aboya\anaconda3\lib\site-packages\sklearn\ensemble\_forest.py:416: FutureWarning: `max_features='auto'
` has been deprecated in 1.1 and will be removed in 1.3. To keep the past behaviour, explicitly set `max_featur
es=1.0` or remove this parameter as it is also the default value for RandomForestRegressors and ExtraTreesRegre
ssors.
  warn(
C:\Users\Aboya\anaconda3\lib\site-packages\sklearn\ensemble\_forest.py:416: FutureWarning: `max_features='auto'
` has been deprecated in 1.1 and will be removed in 1.3. To keep the past behaviour, explicitly set `max_featur
es=1.0` or remove this parameter as it is also the default value for RandomForestRegressors and ExtraTreesRegre
ssors.
  warn(
C:\Users\Aboya\anaconda3\lib\site-packages\sklearn\ensemble\_forest.py:416: FutureWarning: `max_features='auto'
` has been deprecated in 1.1 and will be removed in 1.3. To keep the past behaviour, explicitly set `max_featur
es=1.0` or remove this parameter as it is also the default value for RandomForestRegressors and ExtraTreesRegre
ssors.
  warn(
C:\Users\Aboya\anaconda3\lib\site-packages\sklearn\ensemble\_forest.py:416: FutureWarning: `max_features='auto'
` has been deprecated in 1.1 and will be removed in 1.3. To keep the past behaviour, explicitly set `max_featur
es=1.0` or remove this parameter as it is also the default value for RandomForestRegressors and ExtraTreesRegre
ssors.
  warn(
C:\Users\Aboya\anaconda3\lib\site-packages\sklearn\ensemble\_forest.py:416: FutureWarning: `max_features='auto'
` has been deprecated in 1.1 and will be removed in 1.3. To keep the past behaviour, explicitly set `max_featur
es=1.0` or remove this parameter as it is also the default value for RandomForestRegressors and ExtraTreesRegre
ssors.
  warn(
C:\Users\Aboya\anaconda3\lib\site-packages\sklearn\ensemble\_forest.py:416: FutureWarning: `max_features='auto'
` has been deprecated in 1.1 and will be removed in 1.3. To keep the past behaviour, explicitly set `max_featur
es=1.0` or remove this parameter as it is also the default value for RandomForestRegressors and ExtraTreesRegre
ssors.
  warn(
C:\Users\Aboya\anaconda3\lib\site-packages\sklearn\ensemble\_forest.py:416: FutureWarning: `max_features='auto'
` has been deprecated in 1.1 and will be removed in 1.3. To keep the past behaviour, explicitly set `max_featur
es=1.0` or remove this parameter as it is also the default value for RandomForestRegressors and ExtraTreesRegre
ssors.
  warn(
C:\Users\Aboya\anaconda3\lib\site-packages\sklearn\ensemble\_forest.py:416: FutureWarning: `max_features='auto'
` has been deprecated in 1.1 and will be removed in 1.3. To keep the past behaviour, explicitly set `max_featur
es=1.0` or remove this parameter as it is also the default value for RandomForestRegressors and ExtraTreesRegre
ssors.
  warn(
C:\Users\Aboya\anaconda3\lib\site-packages\sklearn\ensemble\_forest.py:416: FutureWarning: `max_features='auto'
` has been deprecated in 1.1 and will be removed in 1.3. To keep the past behaviour, explicitly set `max_featur
es=1.0` or remove this parameter as it is also the default value for RandomForestRegressors and ExtraTreesRegre
ssors.
  warn(
```

Out[298]:

```
▸ RandomizedSearchCV
▸ estimator: RandomForestRegressor
  ▸ RandomForestRegressor
```

In [300…

```python
rs_model.best_params_
```

Out[300]:

```
{'n_estimators': 80,
 'min_samples_split': 6,
 'min_samples_leaf': 3,
 'max_samples': 10000,
 'max_features': 'auto',
 'max_depth': 5}
```

In [301…

```python
%%time
#Most ideal hyperparameter
ideal_model= RandomForestRegressor(n_jobs=-1,
                                   n_estimators=40,
                                   min_samples_split=14,
                                   min_samples_leaf=1,
                                   max_features=0.5,
                                   max_samples=None,
                                   random_state=42)


#fit Model
ideal_model.fit(x_train,y_train)
```

```
CPU times: total: 4min 40s
Wall time: 1min 15s
```

Out[301]:

```
RandomForestRegressor(max_features=0.5, min_samples_split=14, n_estimators=40,
                      n_jobs=-1, random_state=42)
```

In [302…
```
show_scores(ideal_model)
```

Out[302]:
```
{'Training MAE': 2951.123355082265,
 'Valid MAE': 5966.934781951541,
 'Training RMSLE': 0.020859282950977772,
 'Valid RMSLE': 0.060639861245551006,
 'Training r2': 0.9589090179371786,
 'Valid r2': 0.8819835895768973}
```

In [311…
```
test_preds= ideal_model.predict(df_test)
```

```
C:\Users\Aboya\anaconda3\lib\site-packages\sklearn\base.py:493: FutureWarning: The feature names should match t
hose that were passed during fit. Starting version 1.2, an error will be raised.
Feature names must be in the same order as they were in fit.

  warnings.warn(message, FutureWarning)
```

In [304…
```
#Check which column our df_test is miss
set(dff)-set(df_test)
```

Out[304]:
```
{'SalePrice', 'auctioneerIDIs_missing'}
```

In [305…
```
#Add auctioneerIDIs_missing to df_test
df_test["auctioneerIDIs_missing"]= True
```

In [306…
```
df_test.head()
```

Out[306]:

	SalesID	MachineID	ModelID	datasource	auctioneerID	YearMade	MachineHoursCurrentMeter	UsageBand	fiModelDesc	fiBaseModel	...
0	1227829	1006309	3168	121	3	1999	3688.0	2	499	180	...
1	1227844	1022817	7271	121	3	1000	28555.0	1	831	292	...
2	1227847	1031560	22805	121	3	2004	6038.0	3	1177	404	...
3	1227848	56204	1269	121	3	2006	8940.0	1	287	113	...
4	1227863	1053887	22312	121	3	2005	2286.0	2	566	196	...

5 rows × 102 columns

Now our test DataFrame has the same features as our traing DataFrame, so we can now make predictions

In [312…
```
test_preds= ideal_model.predict(df_test)
```

```
C:\Users\Aboya\anaconda3\lib\site-packages\sklearn\base.py:493: FutureWarning: The feature names should match t
hose that were passed during fit. Starting version 1.2, an error will be raised.
Feature names must be in the same order as they were in fit.

  warnings.warn(message, FutureWarning)
```

In [316…
```
test_preds
```

Out[316]:
```
array([20962.48769196, 19827.3550391 , 50104.98947283, ...,
       17066.32563906, 22990.1563963 , 31353.97590649])
```

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]: