# Predicting the Sale Price of Bulldozers using ML

## 1. Problem Defination

> Predict future price of Bulldozer , based on characteristics

## 2. Data

> Data downloaded from https://www.kaggle.com/c/bluebook-for-bulldozers

The data for this competition is split into three parts:

- Train.csv is the training set, which contains data through the end of 2011.
- Valid.csv is the validation set, which contains data from January 1, 2012 - April 30, 2012 You make predictions on this set throughout the majority of the competition. Your score on this set is used to create the public leaderboard.
- Test.csv is the test set, which won't be released until the last week of the competition. It contains data from May 1, 2012 - November 2012. Your score on the test set determines your final rank for the competition.

The key fields are in train.csv are:

- SalesID: the uniue identifier of the sale
- MachineID: the unique identifier of a machine. A machine can be sold multiple times
- saleprice: what the machine sold for at auction (only provided in train.csv)
- saledate: the date of the sale

## 3. Evaluation

The evaluation metric for this competition is the RMSLE (root mean squared log error) between the actual and predicted auction prices.

Goal : Minimize the Error RMSLE

## 4. Features

Check Data Dictionary https://www.kaggle.com/competitions/bluebook-for-bulldozers/data

In [1]:
```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import sklearn
```

In [2]:
```
#Import training and validation sets
df = pd.read_csv("TrainAndValid.csv",low_memory=False)
```

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

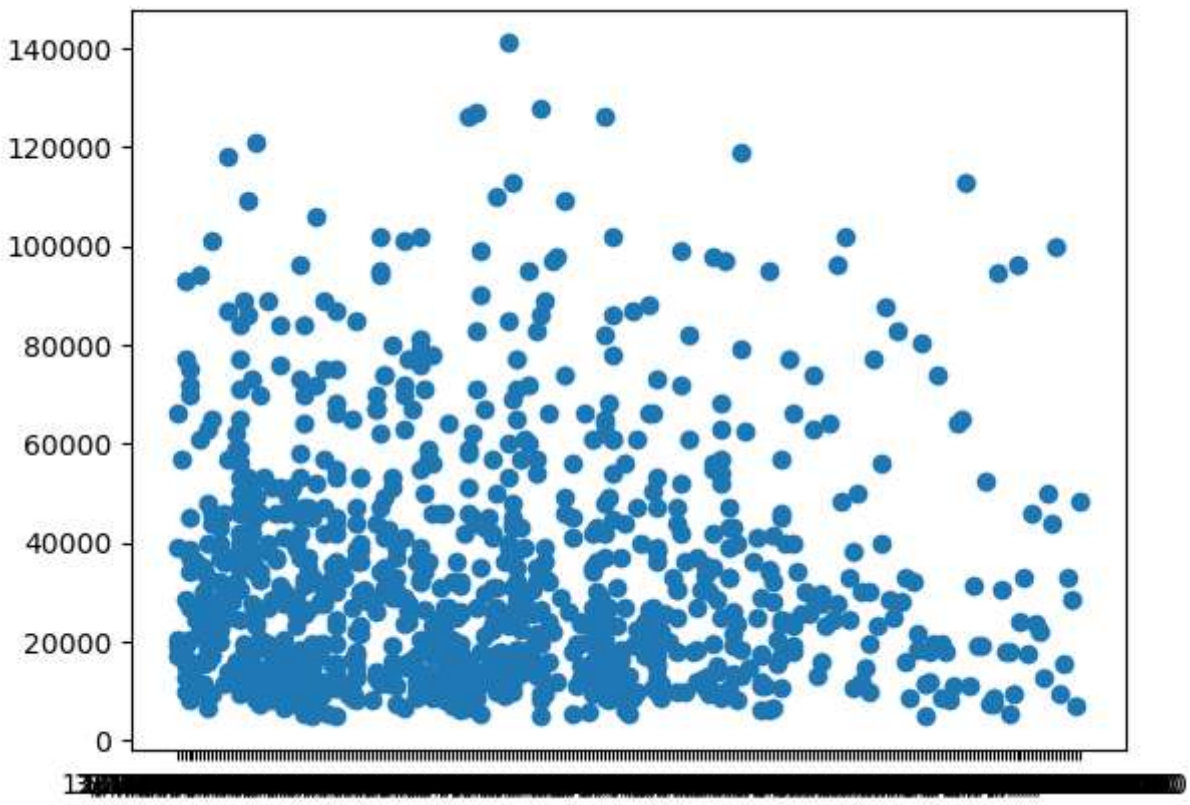
```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 412698 entries, 0 to 412697
Data columns (total 53 columns):
 #   Column                    Non-Null Count   Dtype  
---  ------                    --------------   -----  
 0   SalesID                   412698 non-null  int64  
 1   SalePrice                 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.isna().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]:
```
fig, ax = plt.subplots()
ax.scatter(df["saledate"][:1000],df["SalePrice"][:1000])
```

Out[5]:
```
<matplotlib.collections.PathCollection at 0x2ae9db00550>
```

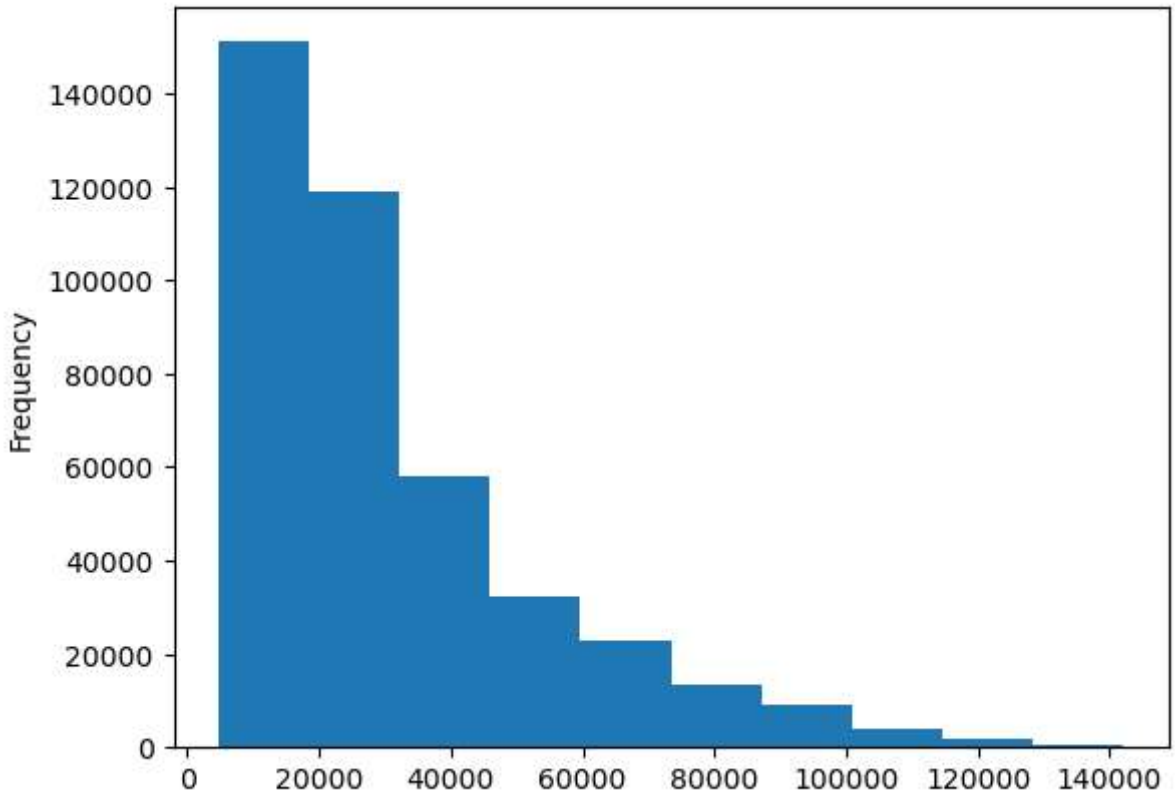



In [6]:
```
df.saledate[:1000]
```

Out[6]:
```
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
            ...
995     7/16/2009 0:00
996     6/14/2007 0:00
997     9/22/2005 0:00
998     7/28/2005 0:00
999     6/16/2011 0:00
Name: saledate, Length: 1000, dtype: object
```

In [7]:
```
df.SalePrice.plot.hist()
```

Out[7]:
```
<Axes: ylabel='Frequency'>
```




## Parsing Dates

Tell pandas which col has dates in it using parse dates

In [8]:
```
#Import data again but this time parse dates
df = pd.read_csv("TrainAndValid.csv",low_memory=False,parse_dates=["saledate"])
df.saledate.dtype
```

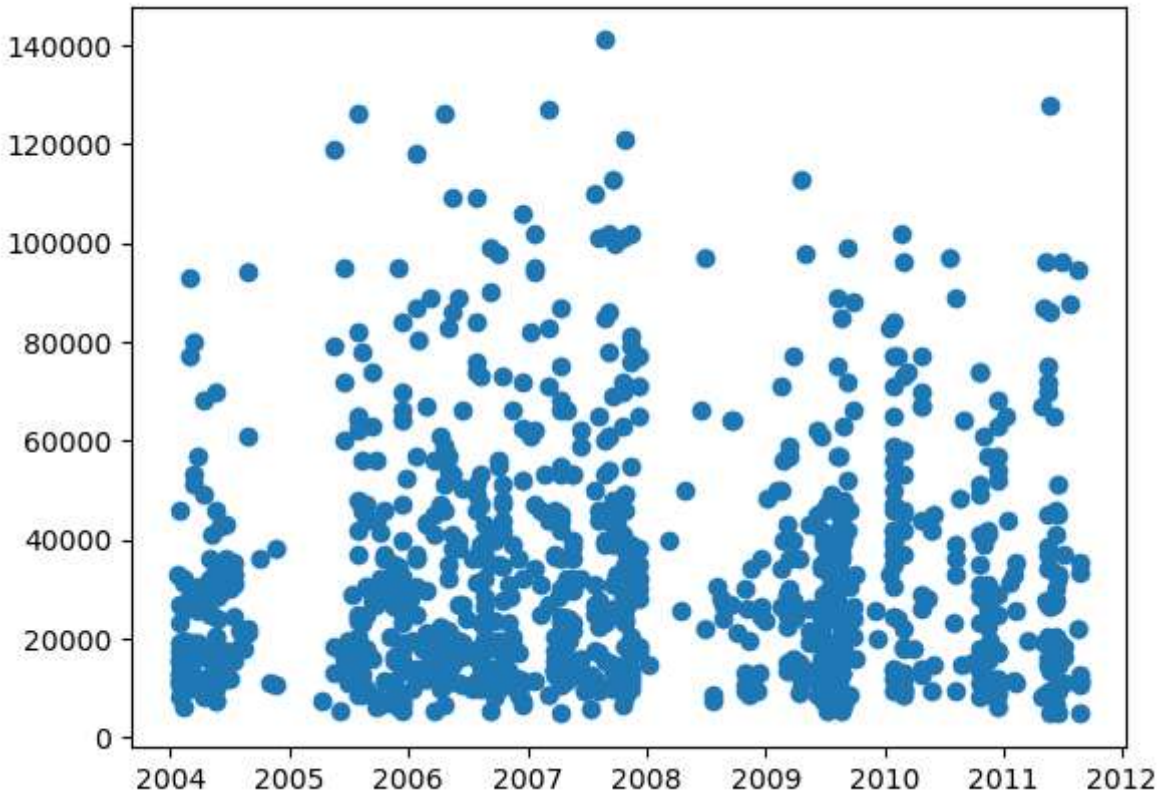
Out[8]:
```
dtype('<M8[ns]')
```

In [9]:
```
df.saledate[:1000]
```

Out[9]:
```
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 [10]:
```
fig, ax = plt.subplots()
ax.scatter(df["saledate"][:1000],df["SalePrice"][:1000])
```

Out[10]:
```
<matplotlib.collections.PathCollection at 0x2ae9dcb51e0>
```




In [11]:
```
df.head()
```

Out[11]:

| | SalesID | SalePrice | MachineID | ModelID | datasource | auctioneerID | YearMade | MachineHoursCurre |
|---|---|---|---|---|---|---|---|---|
| **0** | 1139246 | 66000.0 | 999089 | 3157 | 121 | 3.0 | 2004 | |
| **1** | 1139248 | 57000.0 | 117657 | 77 | 121 | 3.0 | 1996 | |
| **2** | 1139249 | 10000.0 | 434808 | 7009 | 121 | 3.0 | 2001 | |
| **3** | 1139251 | 38500.0 | 1026470 | 332 | 121 | 3.0 | 2001 | |
| **4** | 1139253 | 11000.0 | 1057373 | 17311 | 121 | 3.0 | 2007 | |

5 rows × 53 columns

In [12]:
```
df.head().T
```

Out[12]:

| | 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 |
| **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 |

| | 0 | 1 | 2 | 3 | 4 |
|---|---|---|---|---|---|
| **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 |

In [13]:
```
df.saledate.head(20)
```

Out[13]:
```
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

In [14]:
```
df.sort_values(by=["saledate"],inplace=True, ascending=True)
df.saledate.head(20)
```

Out[14]:
```
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]
```

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

Out[15]:

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

5 rows × 53 columns

## Make Copy of original data frame

In [16]:
```
df_tmp = df.copy()
```

## Add datetime parameters for saledate column

In [17]:
```
df_tmp[:1].saledate.dt.year
```

Out[17]:
```
205615    1989
Name: saledate, dtype: int64
```

In [18]:
```
df_tmp[:1].saledate.dt.day
```

Out[18]:
```
205615    17
Name: saledate, dtype: int64
```

In [19]:
```
df_tmp[:1].saledate
```

Out[19]:
```
205615   1989-01-17
Name: saledate, dtype: datetime64[ns]
```

In [20]:
```
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 [21]:
```
#Dropping Saledate as we enriched our existing features
df_tmp.drop("saledate", axis=1, inplace=True)
```

In [22]:
```
df_tmp.state.value_counts()
```

Out[22]:
```
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
```

# Modelling

model driven EDA

In [23]:
```
from sklearn.ensemble import RandomForestRegressor

#model = RandomForestRegressor(n_jobs=-1,
#                              random_state=42)
#model.fit(df_tmp.drop("SalePrice", axis=1),df_tmp["SalePrice"])
```

In [24]:
```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 412698 entries, 205615 to 409203
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  datetime64[ns]
 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: datetime64[ns](1), float64(3), int64(5), object(44)
memory usage: 170.0+ MB
```

# Convert String to categories

https://pandas.pydata.org/pandas-docs/version/1.4/reference/api/pandas.api.types.is_string_dtype.html

In [25]:
```
pd.api.types.is_string_dtype(df_tmp["UsageBand"])
```

Out[25]:
```
True
```

In [26]:
```
#find the cols which contain 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 [27]:
```
#random test code to explain
random_dict = {"key1" : "hello",
               "key2" : "world"}

for key, value in random_dict.items():
    print(f"this is a key: {key}",
          f"this is a value: {value}")
```

```
this is a key: key1 this is a value: hello
this is a key: key2 this is a value: world
```

In [28]:
```
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 [29]:
```
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   category
 9   fiModelDesc               412698 non-null  category
 10  fiBaseModel               412698 non-null  category
 11  fiSecondaryDesc           271971 non-null  category
 12  fiModelSeries             58667 non-null   category
 13  fiModelDescriptor         74816 non-null   category
 14  ProductSize               196093 non-null  category
 15  fiProductClassDesc        412698 non-null  category
 16  state                     412698 non-null  category
 17  ProductGroup              412698 non-null  category
 18  ProductGroupDesc          412698 non-null  category
 19  Drive_System              107087 non-null  category
 20  Enclosure                 412364 non-null  category
 21  Forks                     197715 non-null  category
 22  Pad_Type                  81096 non-null   category
 23  Ride_Control              152728 non-null  category
 24  Stick                     81096 non-null   category
 25  Transmission              188007 non-null  category
 26  Turbocharged              81096 non-null   category
 27  Blade_Extension           25983 non-null   category
 28  Blade_Width               25983 non-null   category
 29  Enclosure_Type            25983 non-null   category
 30  Engine_Horsepower         25983 non-null   category
 31  Hydraulics                330133 non-null  category
 32  Pushblock                 25983 non-null   category
 33  Ripper                    106945 non-null  category
 34  Scarifier                 25994 non-null   category
 35  Tip_Control               25983 non-null   category
 36  Tire_Size                 97638 non-null   category
 37  Coupler                   220679 non-null  category
 38  Coupler_System            44974 non-null   category
 39  Grouser_Tracks            44875 non-null   category
 40  Hydraulics_Flow           44875 non-null   category
 41  Track_Type                102193 non-null  category
 42  Undercarriage_Pad_Width   102916 non-null  category
 43  Stick_Length              102261 non-null  category
 44  Thumb                     102332 non-null  category
 45  Pattern_Changer           102261 non-null  category
 46  Grouser_Type              102193 non-null  category
 47  Backhoe_Mounting          80712 non-null   category
 48  Blade_Type                81875 non-null   category
 49  Travel_Controls           81877 non-null   category
 50  Differential_Type         71564 non-null   category
 51  Steering_Controls         71522 non-null   category
 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: category(44), float64(3), int64(10)
memory usage: 63.2 MB
```

In [30]:
```
df_tmp.state.cat.categories
```

Out[30]:
```
Index(['Alabama', 'Alaska', 'Arizona', 'Arkansas', 'California', 'Colorado',
       'Connecticut', 'Delaware', 'Florida', 'Georgia', 'Hawaii', 'Idaho',
       'Illinois', 'Indiana', 'Iowa', 'Kansas', 'Kentucky', 'Louisiana',
       'Maine', 'Maryland', 'Massachusetts', 'Michigan', 'Minnesota',
       'Mississippi', 'Missouri', 'Montana', 'Nebraska', 'Nevada',
       'New Hampshire', 'New Jersey', 'New Mexico', 'New York',
       'North Carolina', 'North Dakota', 'Ohio', 'Oklahoma', 'Oregon',
       'Pennsylvania', 'Puerto Rico', 'Rhode Island', 'South Carolina',
       'South Dakota', 'Tennessee', 'Texas', 'Unspecified', 'Utah', 'Vermont',
       'Virginia', 'Washington', 'Washington DC', 'West Virginia', 'Wisconsin',
       'Wyoming'],
      dtype='object')
```

In [31]:
```
df_tmp.state.cat.codes
```

Out[31]:
```
205615    43
274835     8
141296     8
212552     8
62755      8
          ..
410879     4
412476     4
411927     4
407124     4
409203     4
Length: 412698, dtype: int8
```

In [32]:
```
df_tmp.isnull().sum()/len(df_tmp)
```

Out[32]:

```
SalesID                     0.000000
SalePrice                   0.000000
MachineID                   0.000000
ModelID                     0.000000
datasource                  0.000000
auctioneerID                0.048791
YearMade                    0.000000
MachineHoursCurrentMeter    0.642586
UsageBand                   0.821492
fiModelDesc                 0.000000
fiBaseModel                 0.000000
fiSecondaryDesc             0.340993
fiModelSeries               0.857845
fiModelDescriptor           0.818715
ProductSize                 0.524851
fiProductClassDesc          0.000000
state                       0.000000
ProductGroup                0.000000
ProductGroupDesc            0.000000
Drive_System                0.740520
Enclosure                   0.000809
Forks                       0.520921
Pad_Type                    0.803498
Ride_Control                0.629928
Stick                       0.803498
Transmission                0.544444
Turbocharged                0.803498
Blade_Extension             0.937041
Blade_Width                 0.937041
Enclosure_Type              0.937041
Engine_Horsepower           0.937041
Hydraulics                  0.200062
Pushblock                   0.937041
Ripper                      0.740864
Scarifier                   0.937014
Tip_Control                 0.937041
Tire_Size                   0.763415
Coupler                     0.465277
Coupler_System              0.891024
Grouser_Tracks              0.891264
Hydraulics_Flow             0.891264
Track_Type                  0.752378
Undercarriage_Pad_Width     0.750626
Stick_Length                0.752213
Thumb                       0.752041
Pattern_Changer             0.752213
Grouser_Type                0.752378
Backhoe_Mounting            0.804428
Blade_Type                  0.801610
Travel_Controls             0.801606
Differential_Type           0.826595
Steering_Controls           0.826697
saleYear                    0.000000
saleMonth                   0.000000
saleDay                     0.000000
saleDayOfWeek               0.000000
saleDayOfYear               0.000000
dtype: float64
```

## Save preprocessed data

In [33]:
```
df_tmp.to_csv("train_tmp.csv", index=False)
```

In [34]:
```
df_tmp = pd.read_csv("train_tmp.csv",low_memory=False)
df_tmp.head().T
```

Out[34]:

| | 0 | 1 | 2 | 3 | 4 |
|---|---|---|---|---|---|
| **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 |

| | 0 | 1 | 2 | 3 | 4 |
|---|---|---|---|---|---|
| **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 |

# Fill Missing Values

## Fill Numerical Missing values first

In [35]:
```
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 [36]:
```
df_tmp.ModelID
```

Out[36]:
```
0          8434
1         10150
2          4139
3          8591
4          4089
          ...
412693     5266
412694    19330
412695    17244
412696     3357
412697     4701
Name: ModelID, Length: 412698, dtype: int64
```

In [37]:
```
#check for null columns
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 [38]:
```
# 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 column which tells us if the data was missing or not
            df_tmp[label+"_is_missing"] = pd.isnull(content)
            # Fill missing numeric values with median
            df_tmp[label] = content.fillna(content.median())
```

In [39]:
```
#demonstrate how median is more robust than mean
hundreds = np.full((1000,),100)
hundreds_billion = np.append(hundreds,1000000000)
np.mean(hundreds),np.mean(hundreds_billion),np.median(hundreds),np.median(hundreds_bil
```

Out[39]:
```
(100.0, 999100.8991008991, 100.0, 100.0)
```

In [40]:
```
# Check if there's any null numeric values
for label, content in df_tmp.items():
    if pd.api.types.is_numeric_dtype(content):
        if pd.isnull(content).sum():
            print(label)
```

## Fill Categorical Missing values

In [41]:
```
#check for columns which not numerical
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 [42]:
```
df.isna().sum()
```

Out[42]:
```
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 [43]:
```
df_tmp.isna().sum()
```

Out[43]:

```
SalesID                             0
SalePrice                           0
MachineID                           0
ModelID                             0
datasource                          0
auctioneerID                        0
YearMade                            0
MachineHoursCurrentMeter            0
UsageBand                      339028
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
saleYear                            0
saleMonth                           0
saleDay                             0
saleDayOfWeek                       0
saleDayOfYear                       0
auctioneerID_is_missing             0
MachineHoursCurrentMeter_is_missing 0
dtype: int64
```

In [44]:
```
#check for columns which not numerical
for label, content in df_tmp.items():
    if not pd.api.types.is_numeric_dtype(content):
        #Add binary column to indicate whether sample had missing value
        df_tmp[label+"_is_missing"] = pd.isnull(content)
        #Tur categories into numbers and add+1
        df_tmp[label] = pd.Categorical(content).codes + 1 #missing is assigned neg 1 s
```

In [45]:
```
pd.Categorical(df_tmp["state"]).codes
```

Out[45]:
```
array([43,  8,  8, ...,  4,  4,  4], dtype=int8)
```

In [46]:
```
pd.Categorical(df_tmp["UsageBand"]).codes+1
```

Out[46]:
```
array([1, 1, 1, ..., 1, 1, 1], dtype=int8)
```

In [47]:
```
df_tmp.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 412698 entries, 0 to 412697
Columns: 103 entries, SalesID to Steering_Controls_is_missing
dtypes: bool(46), float64(3), int16(4), int64(10), int8(40)
memory usage: 77.9 MB
```

In [48]:
```
df_tmp.head().T
```

Out[48]:

| | 0 | 1 | 2 | 3 | 4 |
|---|---|---|---|---|---|
| **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 |
| **...** | ... | ... | ... | ... | ... |
| **Backhoe_Mounting_is_missing** | False | True | False | True | False |
| **Blade_Type_is_missing** | False | True | False | True | False |
| **Travel_Controls_is_missing** | False | True | False | True | False |
| **Differential_Type_is_missing** | True | False | True | False | True |
| **Steering_Controls_is_missing** | True | False | True | False | True |

103 rows × 5 columns

In [49]:
```
df_tmp.isna().sum()[:10]
```

Out[49]:
```
SalesID                     0
SalePrice                   0
MachineID                   0
ModelID                     0
datasource                  0
auctioneerID                0
YearMade                    0
MachineHoursCurrentMeter    0
UsageBand                   0
fiModelDesc                 0
dtype: int64
```

# FIT the MODEL

In [50]:
```
%%time
# Instantiate model
model = RandomForestRegressor(n_jobs=-1,
                              random_state=42)

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

```
CPU times: total: 35min 18s
Wall time: 2min 25s
```

Out[50]:
```
RandomForestRegressor
RandomForestRegressor(n_jobs=-1, random_state=42)
```

In [51]:
```
#Score the model - Nonrelaiable - cause we are testing on train data
model.score(df_tmp.drop("SalePrice",axis=1),df_tmp["SalePrice"])
```

Out[51]:
```
0.9875468079970562
```

# Splitting data into train/validation set

In [52]:
```
# We'll be working on train and valid sets
```

In [53]:
```
df_tmp.saleYear
```

Out[53]:
```
0         1989
1         1989
2         1989
3         1989
4         1989
          ...
412693    2012
412694    2012
412695    2012
412696    2012
412697    2012
Name: saleYear, Length: 412698, dtype: int64
```

In [54]:
```
df_tmp.saleYear.value_counts()
```

Out[54]:
```
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 [55]:
```
# Split data into train and validation set
# Year 2012 - Valid set , before that is all 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[55]:
```
(11573, 401125)
```

In [56]:
```
# Split data into X & 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[56]:
```
((401125, 102), (401125,), (11573, 102), (11573,))
```

In [57]:
```
y_train
```

Out[57]:
```
0          9500.0
1         14000.0
2         50000.0
3         16000.0
4         22000.0
           ...
401120    29000.0
401121    11000.0
401122    11000.0
401123    18000.0
401124    13500.0
Name: SalePrice, Length: 401125, dtype: float64
```

## Building an evaluation function

In [58]:
```
# Create evaluation function (the competition uses RMSLE)
from sklearn.metrics import mean_squared_log_error, mean_absolute_error, r2_score

def rmsle(y_test, y_preds):
    """
    Caculates root mean squared log error between predictions and
    true labels.
    """
    return np.sqrt(mean_squared_log_error(y_test, y_preds))

# Create function to evaluate model on a few different levels
def show_scores(model):
    train_preds = model.predict(X_train)
    val_preds = model.predict(X_valid)
    scores = {"Training MAE": mean_absolute_error(y_train, train_preds),
              "Valid MAE": mean_absolute_error(y_valid, val_preds),
              "Training RMSLE": rmsle(y_train, train_preds),
              "Valid RMSLE": rmsle(y_valid, val_preds),
              "Training R^2": r2_score(y_train, train_preds),
              "Valid R^2": r2_score(y_valid, val_preds)}
    return scores
```

# Testing our model on subset (to tune the hyperparameters)

In [59]:
```
# # this takes lots time
#%%time
#model = RandomForestRegressor(n_jobs=-1,
#                              random_state=42)
#model.fit(X_train, y _train)
```

In [60]:
```
#model.fit(X_train[:10000], y _train[:10000])
```

In [61]:
```
# Change max_samples value
model = RandomForestRegressor(n_jobs=-1,
                              random_state=42,
                              max_samples=10000)
```

In [62]:
```
%%time
# Cutting down on the max number of samples each estimator can see improves training t
model.fit(X_train, y_train)
```

```
CPU times: total: 1min 18s
Wall time: 5.58 s
```

Out[62]:
```
RandomForestRegressor(max_samples=10000, n_jobs=-1, random_state=42)
```

In [63]:
```
show_scores(model)
```

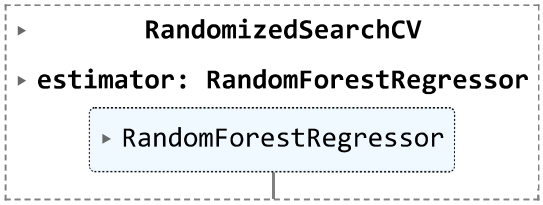
Out[63]:
```
{'Training MAE': 5561.2988092240585,
 'Valid MAE': 7177.26365505919,
 'Training RMSLE': 0.257745378256977,
 'Valid RMSLE': 0.29362638671089003,
 'Training R^2': 0.8606658995199189,
 'Valid R^2': 0.8320374995090507}
```

## Hyperparameter tuning with Randomized SeacrhCV

In [64]:
```
%%time
from sklearn.model_selection import RandomizedSearchCV

#Different RandomForestRegressor hyperparameters
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,
                              n_iter=2,
                              cv=5,
                              verbose=True)

rs_model.fit(X_train,y_train)
```

```
Fitting 5 folds for each of 2 candidates, totalling 10 fits
CPU times: total: 13.8 s
Wall time: 20.3 s
```

Out[64]:




In [65]:
```
# Find the best hyperparameters
rs_model.best_params_
```

Out[65]:
```
{'n_estimators': 40,
 'min_samples_split': 16,
 'min_samples_leaf': 13,
 'max_samples': 10000,
 'max_features': 0.5,
 'max_depth': 10}
```

In [66]:
```
# Evaluate the RansomizedSearch Model
show_scores(rs_model)
```

Out[66]:
```
{'Training MAE': 7144.863476107355,
 'Valid MAE': 8355.827830969576,
 'Training RMSLE': 0.3159932305736454,
 'Valid RMSLE': 0.3352320866505292,
 'Training R^2': 0.7811921842270204,
 'Valid R^2': 0.7782814104020885}
```

# Train a model with the best hyperparameters

Note: These were found after 100 iterations of RandomizedSearchCV

In [68]:
```
%%time

#Most Ideal hyperparameters
ideal_model = RandomForestRegressor(n_estimators=40,
                                    min_samples_leaf=1,
                                    min_samples_split=14,
                                    max_features=0.5,
                                    n_jobs=-1,
                                    max_samples=None,
                                    random_state=42)

#Fit the model
ideal_model.fit(X_train, y_train)
```

```
CPU times: total: 6min 15s
Wall time: 32.6 s
```

Out[68]:
```
RandomForestRegressor(max_features=0.5, min_samples_split=14, n_estimators=40,
                      n_jobs=-1)
```

In [69]:
```
show_scores(ideal_model)
```

Out[69]:
```
{'Training MAE': 2948.637328517366,
 'Valid MAE': 5956.756488137891,
 'Training RMSLE': 0.144272782095458,
 'Valid RMSLE': 0.24593747427122192,
 'Training R^2': 0.9590243396382552,
 'Valid R^2': 0.8807314699470798}
```

# Make predictions on test data

In [101…
```
#Import test data
df_test = pd.read_csv("Test.csv",
                      low_memory=False,
                      parse_dates=["saledate"])

df_test.head()
```

Out[101]:

| | SalesID | MachineID | ModelID | datasource | auctioneerID | YearMade | MachineHoursCurrentMeter | U |
|---|---|---|---|---|---|---|---|---|
| **0** | 1227829 | 1006309 | 3168 | 121 | 3 | 1999 | 3688.0 | |
| **1** | 1227844 | 1022817 | 7271 | 121 | 3 | 1000 | 28555.0 | |
| **2** | 1227847 | 1031560 | 22805 | 121 | 3 | 2004 | 6038.0 | |
| **3** | 1227848 | 56204 | 1269 | 121 | 3 | 2006 | 8940.0 | |
| **4** | 1227863 | 1053887 | 22312 | 121 | 3 | 2005 | 2286.0 | |

In [71]:
```
#test data has NA , Non numerical values and different column size

#MAke prediction on test dataset
#test_preds = ideal_model.predict(df_test)
```

# Preprocessing test data

In [102…
```
def preprocess_data(df):
    """
    Performs transformations on df and
    """
    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)

    # Fill numeric rows with the median
    for label, content in df.items():
        if pd.api.types.is_numeric_dtype(content):
            if pd.isnull(content).sum():
                # Add a binary column which tells us if the data was missing or not
                df[label+"_is_missing"] = pd.isnull(content)
                # Fill missing numeric values with median
                df[label] = content.fillna(content.median())

        # Fill categorical with num codes
        if not pd.api.types.is_numeric_dtype(content):
            df[label+"_is_missing"] = pd.isnull(content)
            #we add +1
            df[label] = pd.Categorical(content).codes+1

    return df
```

In [103…
```
#process test data
df_test = preprocess_data(df_test)
df_test.head()
```

Out[103]:

| | SalesID | MachineID | ModelID | datasource | auctioneerID | YearMade | MachineHoursCurrentMeter | U |
|---|---|---|---|---|---|---|---|---|
| **0** | 1227829 | 1006309 | 3168 | 121 | 3 | 1999 | 3688.0 | |
| **1** | 1227844 | 1022817 | 7271 | 121 | 3 | 1000 | 28555.0 | |
| **2** | 1227847 | 1031560 | 22805 | 121 | 3 | 2004 | 6038.0 | |
| **3** | 1227848 | 56204 | 1269 | 121 | 3 | 2006 | 8940.0 | |
| **4** | 1227863 | 1053887 | 22312 | 121 | 3 | 2005 | 2286.0 | |

In [104…
```
#Make predictions on updated test data
#-> No of cols in train data set is 102 and test is 101
#-> So this throws error

#Find how the columns differ using sets
set(X_train.columns) - set(df_test.columns)
```

Out[104]:
```
{'auctioneerID_is_missing'}
```

In [105…
```
#Manually adjust df_test to have autioneerigID_is_misisng column
df_test["auctioneerID_is_missing"] = False
df_test.head()
```

Out[105]:

| | SalesID | MachineID | ModelID | datasource | auctioneerID | YearMade | MachineHoursCurrentMeter | U |
|---|---|---|---|---|---|---|---|---|
| **0** | 1227829 | 1006309 | 3168 | 121 | 3 | 1999 | 3688.0 | |
| **1** | 1227844 | 1022817 | 7271 | 121 | 3 | 1000 | 28555.0 | |
| **2** | 1227847 | 1031560 | 22805 | 121 | 3 | 2004 | 6038.0 | |
| **3** | 1227848 | 56204 | 1269 | 121 | 3 | 2006 | 8940.0 | |
| **4** | 1227863 | 1053887 | 22312 | 121 | 3 | 2005 | 2286.0 | |

In [106…
```
pd.set_option('display.max_columns', None)
X_train.head()
X_train.columns.get_loc("auctioneerID_is_missing")
```

Out[106]:
```
56
```

In [112…
```
#i had to do this cause tool was trowing error for aution.. being differently indexed
cols = list(df_test)
cols.insert(56,cols.pop(cols.index('auctioneerID_is_missing')))
cols
df_test=df_test.reindex(columns=cols)
df_test.columns.get_loc("auctioneerID_is_missing")
```

Out[112]:
```
56
```

In [113…
```
test_preds = ideal_model.predict(df_test)
```

In [114…
```
test_preds
```

Out[114]:
```
array([16824.59949248, 16610.73260562, 45175.61153815, ...,
       12064.76647108, 17516.62443005, 28052.5037774 ])
```

In [115…
```
#Format predictions into same format Kaggle wants
df_preds = pd.DataFrame()
df_preds["SalesID"] = df_test["SalesID"]
df_preds["SalesPrice"] = test_preds
df_preds
```

Out[115]:

| | SalesID | SalesPrice |
|---|---|---|
| **0** | 1227829 | 16824.599492 |
| **1** | 1227844 | 16610.732606 |
| **2** | 1227847 | 45175.611538 |
| **3** | 1227848 | 65915.931927 |
| **4** | 1227863 | 59694.991367 |
| **...** | ... | ... |
| **12452** | 6643171 | 45148.542195 |
| **12453** | 6643173 | 12938.147555 |
| **12454** | 6643184 | 12064.766471 |
| **12455** | 6643186 | 17516.624430 |
| **12456** | 6643196 | 28052.503777 |

12457 rows × 2 columns

In [116…
```
#Export
df_preds.to_csv("test_predictions.csv",index=False)
```

# Feature Importance

Figure out which attributes of data were most important for predicting target value

In [117…
```
ideal_model.feature_importances_
```

Out[117]:
```
array([3.48294064e-02, 1.65875066e-02, 4.44953027e-02, 1.86248461e-03,
       3.33132871e-03, 2.13696536e-01, 3.22969003e-03, 1.01934242e-03,
       5.59895797e-02, 4.74758732e-02, 6.25292981e-02, 4.73232165e-03,
       1.21491430e-02, 1.67258430e-01, 4.50848994e-02, 5.91170875e-03,
       1.75817105e-03, 1.39665998e-03, 2.09363205e-03, 4.56627953e-02,
       4.83182556e-04, 4.76371677e-05, 8.92150296e-04, 2.02099288e-04,
       9.66878256e-04, 7.17457006e-05, 4.02293502e-03, 6.88849561e-03,
       1.68632640e-03, 1.69345122e-04, 2.95291088e-03, 2.59336978e-03,
       4.26038351e-03, 6.54300115e-04, 6.03640893e-04, 5.42771199e-03,
       7.90225974e-04, 1.07525807e-02, 3.55987873e-04, 3.24376780e-03,
       8.36021238e-04, 1.05913569e-03, 2.38276756e-03, 5.83427788e-04,
       6.23731420e-04, 4.22356000e-04, 3.52967948e-04, 2.22005777e-03,
       9.94055257e-04, 2.92248904e-04, 1.54258976e-04, 7.23999175e-02,
       3.78285685e-03, 5.63008120e-03, 2.93382699e-03, 9.90998692e-03,
       2.65041985e-04, 1.42810128e-03, 3.33876795e-04, 0.00000000e+00,
       0.00000000e+00, 1.62818773e-03, 1.29352306e-03, 7.29124785e-03,
       3.51688046e-02, 0.00000000e+00, 0.00000000e+00, 0.00000000e+00,
       0.00000000e+00, 1.11283471e-05, 6.19861847e-06, 1.02205749e-04,
       1.57002198e-03, 2.43064680e-04, 5.74935000e-05, 3.98377072e-04,
       5.95582973e-06, 1.04689150e-04, 4.69126752e-06, 4.69229830e-05,
       2.40583028e-03, 2.52427794e-03, 1.47265902e-04, 6.09288981e-04,
       9.65106771e-04, 2.96550585e-05, 3.05110172e-03, 3.50887859e-04,
       1.09892986e-02, 4.29621648e-03, 7.95605489e-04, 1.57901659e-04,
       5.87915686e-05, 7.51772948e-05, 2.47428946e-05, 1.04451917e-04,
       3.77341115e-05, 2.06262693e-04, 1.13799382e-04, 1.20592822e-04,
       6.70533362e-05, 1.71942836e-04])
```

In [118…
```
len(ideal_model.feature_importances_)
```

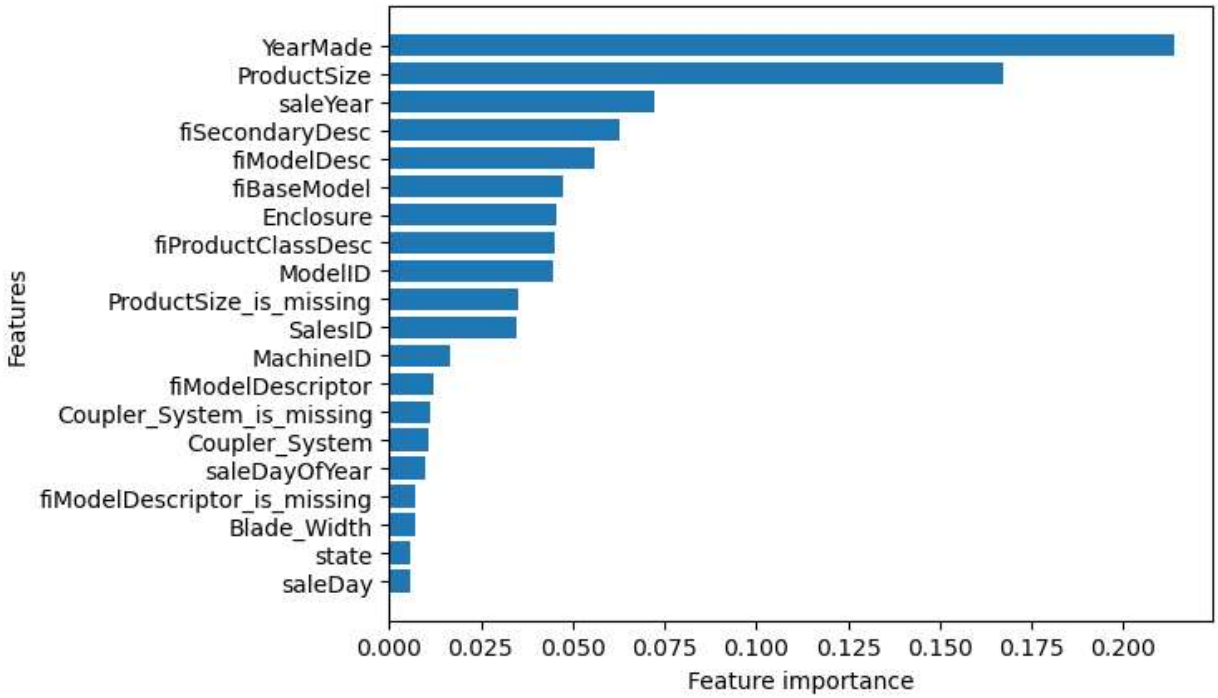
Out[118]:
```
102
```

In [120…
```
X_train.shape
```

Out[120]:
```
(401125, 102)
```

In [127…
```
#Helper function for plotting importance

def plot_features(columns, importances, n=20):
    df = (pd.DataFrame({"features":columns,
                        "feature_importances": importances})
          .sort_values("feature_importances",ascending=False)
          .reset_index(drop=True))

    #plot the dataframe
    fig, ax = plt.subplots()
    ax.barh(df["features"][:n],df["feature_importances"][:20])
    ax.set_ylabel("Features")
    ax.set_xlabel("Feature importance")
    ax.invert_yaxis()
```

In [128…
```
plot_features(X_train.columns, ideal_model.feature_importances_)
```




In [129…
```
df["ProductSize"].value_counts()
```

Out[129]:
```
Medium            64342
Large / Medium    51297
Small             27057
Mini              25721
Large             21396
Compact            6280
Name: ProductSize, dtype: int64
```

In [ ]: