predicting the sale price of bulldozer using machine learning

in this nb we are going through an axample machine learning project with the goal of predicting the sell prize of bulldozers.

1. Problem definition

> Predict the future sale price of a bulldozer ,given its chracterstics and previous examples of how much similar bulldozer have been sold for?

2. Data

> The data is downloaded from thee Kaggle bluebook for bulldozzer competetion:https://www.kaggle.com/c/bluebook-for-bulldozers/data

There are 3 main datasets:

- 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.

3. Evaluation

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

for more on the evaluation of this project check: https://www.kaggle.com/c/bluebook-for-bulldozers/overview/evaluation

Note: The goal for the most regression evaluation metrics is to minimize the error.For example, our goal in this project is to to minimize the RMSLE.

4. Features

> kaggle provide a data dictionary deatailing all the features of teh dataset.you can veiw this data dictionary on google:https://www.kaggle.com/c/bluebook-for-bulldozers/data?

In [1]:

```
import numpy as np
import pandas as pd
import  matplotlib.pyplot as plt
import sklearn
```

In [2]:

```
# importing training and validation sets
df=pd.read_csv("Data/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
```

```
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]:

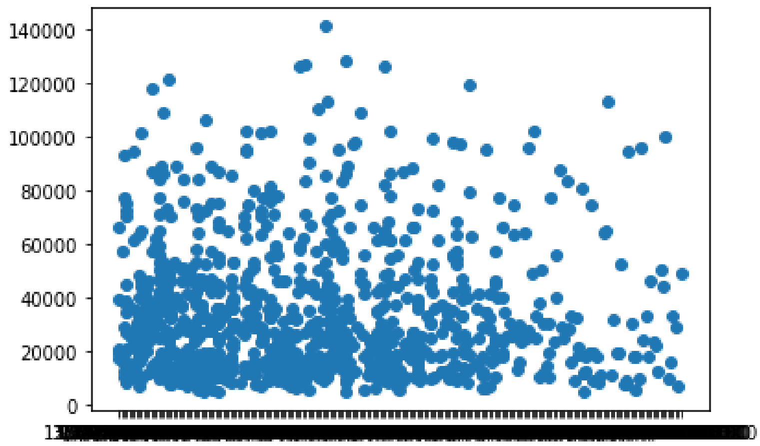
```
df.columns
```

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]:

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



In [7]:

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

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
            ...
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 [8]:

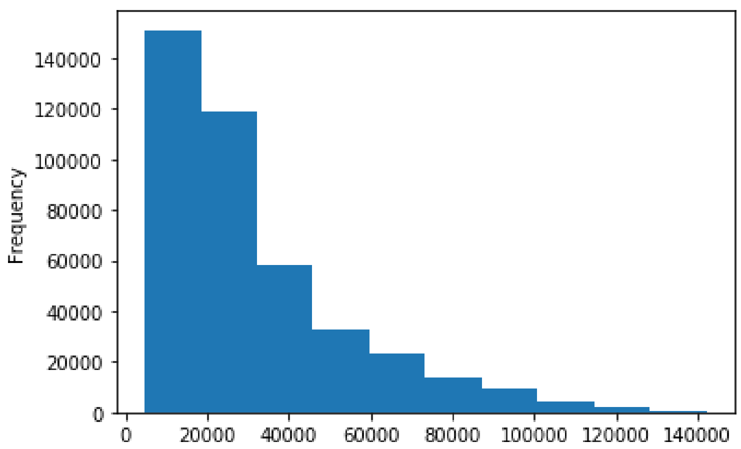
```
df.saledate.dtype
```

Out[8]:

```
dtype('O')
```

In [9]:

```
df.SalePrice.plot.hist();
```



Parsing Dates

when we work with time series data,we wnat to enroch the time and data component as much as possible.

we can do that by telling pandas which of our column has dates with it using the 'parse_dates' parameter

In [10]:

```
# import data agin but this time parse data
df=pd.read_csv("Data/TrainAndValid.csv",low_memory=False,parse_dates=["saledate"])
```

In [11]:

```
df.saledate.dtype
```

Out[11]:

```
dtype('<M8[ns]')
```

In [12]:

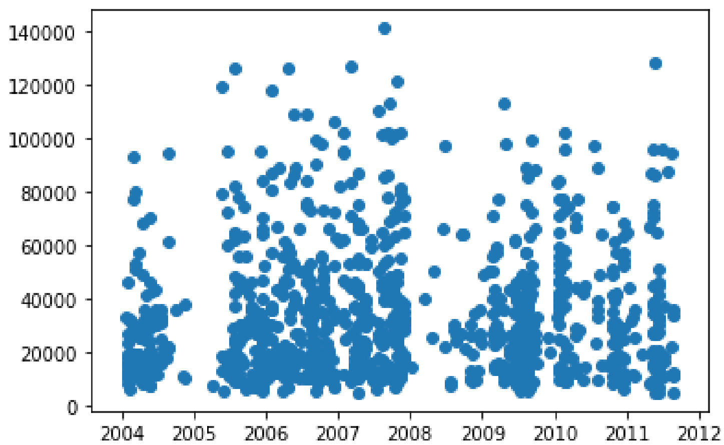
```
df.saledate[:1000]
```

Out[12]:

```
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 [13]:

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



In [14]:

```
df.head()
```

Out[14]:

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

	SalesID	SalePrice	MachineID	ModelID	datasource	auctioneerID	YearMade	MachineHoursCurrentMeter	UsageBand	sa
4	1139253	11000.0	1057373	17311	121	3.0	2007	722.0	Medium	

5 rows × 53 columns

In [15]:

```
df.head().T
```

Out[15]:

	0	1	2	3	4
SalesID	1139246	1139248	1139249	1139251	1139253
SalePrice	66000	57000	10000	38500	11000
MachineID	999089	117657	434808	1026470	1057373
ModelID	3157	77	7009	332	17311
datasource	121	121	121	121	121
auctioneerID	3	3	3	3	3
YearMade	2004	1996	2001	2001	2007
MachineHoursCurrentMeter	68	4640	2838	3486	722
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

	0	1	2	3	4
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 [16]:

```
df.saledate.head(20)
```

Out[16]:

```
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's good idea to sort it by date.

In [17]:

```
# sort dataframe in date order

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

Out[17]:

```
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 original dataframe

we make a copy of the original DataFrame so when we manupulate the copy,we've still got our original Data

In [18]:

```
# meke a copy

df_tmp=df.copy()
```

In [19]:

```
df_tmp
```

Out[19]:

	SalesID	SalePrice	MachineID	ModelID	datasource	auctioneerID	YearMade	MachineHoursCurrentMeter	UsageBar
205615	1646770	9500.0	1126363	8434	132	18.0	1974	NaN	Na
274835	1821514	14000.0	1194089	10150	132	99.0	1980	NaN	Na
141296	1505138	50000.0	1473654	4139	132	99.0	1978	NaN	Na
212552	1671174	16000.0	1327630	8591	132	99.0	1980	NaN	Na
62755	1329056	22000.0	1336053	4089	132	99.0	1984	NaN	Na
...	...	...	...	...	...	...	...	...	
410879	6302984	16000.0	1915521	5266	149	99.0	2001	NaN	Na
412476	6324811	6000.0	1919104	19330	149	99.0	2004	NaN	Na
411927	6313029	16000.0	1918416	17244	149	99.0	2004	NaN	Na
407124	6266251	55000.0	509560	3357	149	99.0	1993	NaN	Na

	SalesID	SalePrice	MachineID	ModelID	datasource	auctioneerID	YearMade	MachineHoursCurrentMeter	UsageBar
409203	6283635	34000.0	1869284	4701	149	99.0	1000	NaN	Na

412698 rows × 53 columns

Add datetime parameters for saledate column

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]:

```
df_tmp.head(10)
```

Out[21]:

	SalesID	SalePrice	MachineID	ModelID	datasource	auctioneerID	YearMade	MachineHoursCurrentMeter	UsageBar
205615	1646770	9500.0	1126363	8434	132	18.0	1974	NaN	Na
274835	1821514	14000.0	1194089	10150	132	99.0	1980	NaN	Na
141296	1505138	50000.0	1473654	4139	132	99.0	1978	NaN	Na
212552	1671174	16000.0	1327630	8591	132	99.0	1980	NaN	Na
62755	1329056	22000.0	1336053	4089	132	99.0	1984	NaN	Na
54653	1301884	23500.0	1182999	4123	132	99.0	1976	NaN	Na
81383	1379228	31000.0	1082797	7620	132	99.0	1986	NaN	Na
204924	1645390	11750.0	1527216	8202	132	99.0	1970	NaN	Na
135376	1493279	63000.0	1363756	2759	132	99.0	1987	NaN	Na
113390	1449549	13000.0	1289412	3356	132	99.0	1966	NaN	Na

10 rows × 58 columns

In [22]:

```
# now we have  enriched our df with date time features,we can remove saledate

df_tmp.drop("saledate",axis=1,inplace=True)
```

In [23]:

```
# check the values of diff. columns

df_tmp.state.value_counts()
```

Out[23]:

```
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
```

5.Modelling

we have done enough EDA(we could always do more) but let's start to do some model driven EDA

In [24]:

```
np.random.seed(42)

# let's build a machine learning model
from sklearn.ensemble import RandomForestRegressor

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


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

```
---------------------------------------------------------------------------
ValueError                                Traceback (most recent call last)
<ipython-input-24-d9819086c9df> in <module>
      8 
      9 
---> 10 model.fit(df_tmp.drop("SalePrice",axis=1),df_tmp["SalePrice"])

~\anaconda3\lib\site-packages\sklearn\ensemble\_forest.py in fit(self, X, y, sample_weigh
t)
    293         """
    294         # Validate or convert input data
--> 295         X = check_array(X, accept_sparse="csc", dtype=DTYPE)
    296         y = check_array(y, accept_sparse='csc', ensure_2d=False, dtype=None)
    297         if sample_weight is not None:

~\anaconda3\lib\site-packages\sklearn\utils\validation.py in check_array(array, accept_sp
arse, accept_large_sparse, dtype, order, copy, force_all_finite, ensure_2d, allow_nd, ens
ure_min_samples, ensure_min_features, warn_on_dtype, estimator)
    529                     array = array.astype(dtype, casting="unsafe", copy=False)
    530                 else:
--> 531                     array = np.asarray(array, order=order, dtype=dtype)
    532             except ComplexWarning:
    533                 raise ValueError("Complex data not supported\n"

~\anaconda3\lib\site-packages\numpy\core\_asarray.py in asarray(a, dtype, order)
     83 
     84     """
---> 85     return array(a, dtype, copy=False, order=order)
     86 
     87 

ValueError: could not convert string to float: 'Low'
```

In [25]:

```
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
```

In [26]:

```
df_tmp["UsageBand"].dtype
```

Out[26]:

```
dtype('O')
```

In [27]:

```
df_tmp.isna().sum()
```

Out[27]:

```
SalesID                          0
SalePrice                        0
MachineID                        0
ModelID                          0
datasource                       0
auctioneerID                 20136
YearMade                         0
MachineHoursCurrentMeter    265194
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
dtype: int64
```

convert string to catrgry

one way we can convert all our data to numbers is we can convert them in category

we can check different datatypes compatible with pandas here: https://pandas.pydata.org/pandas-docs/version/0.25.3/reference/general_utility_functions.html#data-types-related-functionality

In [28]:

```
df_tmp.head().T
```

Out[28]:

	205615	274835	141296	212552	62755
SalesID	1646770	1821514	1505138	1671174	1329056
SalePrice	9500	14000	50000	16000	22000
MachineID	1126363	1194089	1473654	1327630	1336053
ModelID	8434	10150	4139	8591	4089
datasource	132	132	132	132	132
auctioneerID	18	99	99	99	99
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

	205615	274835	141296	212552	62755
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

	205615	274835	141296	212552	62755
saleDayOfWeek	1	1	1	1	1
saleDayOfYear	17	31	31	31	31

In [29]:

```
### now we use one of the API

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

Out[29]:

```
True
```

In [30]:

```
## find the columns whih 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 [31]:

```
# this will turn all the string column into categorical value
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 [32]:

```
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: 63.3 MB
```

In [33]:

In [33]:

```
df_tmp.state.cat.categories
```

Out[33]:

```
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 [34]:

```
df_tmp.state.cat.codes
```

Out[34]:

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

thank to pandas categories now have away to access all of the dta in the form of number but we still have bunch of missing data

In [35]:

```
# check missing data
df_tmp.isnull().sum()/len(df_tmp)
```

Out[35]:

```
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 [36]:

```
# export current tmp dataframe
df_tmp.to_csv("train_tmp.csv",index=False)
```

In [37]:

```
# import the preprocessed data
df_tmp=pd.read_csv("train_tmp.csv",low_memory=False)
```

In [38]:

```
df_tmp.head().T
```

Out[38]:

	0	1	2	3	4
SalesID	1646770	1821514	1505138	1671174	1329056
SalePrice	9500	14000	50000	16000	22000
MachineID	1126363	1194089	1473654	1327630	1336053
ModelID	8434	10150	4139	8591	4089
datasource	132	132	132	132	132
auctioneerID	18	99	99	99	99
YearMade	1974	1980	1978	1980	1984
MachineHoursCurrentMeter	NaN	NaN	NaN	NaN	NaN
UsageBand	NaN	NaN	NaN	NaN	NaN
fiModelDesc	TD20	A66	D7G	A62	D3B

	0	1	2	3	4
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

	0	1	2	3	4
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 [39]:

```
df_tmp.isna().sum()
```

Out[39]:

```
SalesID                          0
SalePrice                        0
MachineID                        0
ModelID                          0
datasource                       0
auctioneerID                 20136
YearMade                         0
MachineHoursCurrentMeter    265194
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
dtype: int64
```

fill the missing values

fill the numerical values first

In [40]:

```
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 [41]:

```
df_tmp.ModelID
```

Out[41]:

```
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 [42]:

```
# check for which numeric columns have 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 [43]:

```
## fill numeric rows with 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 is 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 [44]:

```
# demonstrate how mwdian 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_bi
llion)
```

Out[44]:

```
(100.0, 999100.8991008991, 100.0, 100.0)
```

In [45]:

```
# chk if therer is 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)
```

In [46]:

```
# chk to see how many examples are missing
df_tmp.auctioneerID_is_missing.value_counts()
```

Out[46]:

```
False    392562
True      20136
Name: auctioneerID_is_missing, dtype: int64
```

In [47]:

```
df_tmp.isna().sum()
```

Out[47]:

```
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
```

fill and turning the categorical values into numeric

In [48]:

```python
# chk for column which are not numeric
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 [49]:

```
# turn categorical  values into numbers and fill missing
for label, content in df_tmp.items():
    if not pd.api.types.is_numeric_dtype(content):
        #Add a binary column to indicate that the sample had a missing value
        df_tmp[label+"_is_missing"]=pd.isnull(content)
        # turn the categories into numbers and add +1
        df_tmp[label] = pd.Categorical(content).codes + 1
```

In [50]:

```
pd.Categorical(df_tmp["state"]).codes+1
```

Out[50]:

```
array([44,  9,  9, ...,  5,  5,  5], dtype=int8)
```

In [51]:

```
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 [52]:

```
df_tmp.head().T
```

Out[52]:

	0	1	2	3	4
SalesID	1646770	1821514	1505138	1671174	1329056
SalePrice	9500	14000	50000	16000	22000
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 [53]:

```
df_tmp.isna().sum()
```

Out[53]:

```
SalesID                        0
SalePrice                      0
MachineID                      0
ModelID                        0
datasource                     0
                              ..
Backhoe_Mounting_is_missing    0
Blade_Type_is_missing          0
Travel_Controls_is_missing     0
Differential_Type_is_missing   0
Steering_Controls_is_missing   0
Length: 103, dtype: int64
```

now that all of our data is numeric and our dataset has no misising value we can make a machine learning model

In [54]:

```
df_tmp.head()
```

Out[54]:

	SalesID	SalePrice	MachineID	ModelID	datasource	auctioneerID	YearMade	MachineHoursCurrentMeter	UsageBand	fiM
0	1646770	9500.0	1126363	8434	132	18.0	1974	0.0	0	
1	1821514	14000.0	1194089	10150	132	99.0	1980	0.0	0	
2	1505138	50000.0	1473654	4139	132	99.0	1978	0.0	0	
3	1671174	16000.0	1327630	8591	132	99.0	1980	0.0	0	
4	1329056	22000.0	1336053	4089	132	99.0	1984	0.0	0	

5 rows × 103 columns

In [55]:

```
%%time
# istantiate model
model = RandomForestRegressor(n_jobs=-1,
                              random_state=42)

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

Wall time: 6min 16s

Out[55]:

```
RandomForestRegressor(bootstrap=True, ccp_alpha=0.0, criterion='mse',
                      max_depth=None, max_features='auto', max_leaf_nodes=None,
                      max_samples=None, min_impurity_decrease=0.0,
                      min_impurity_split=None, min_samples_leaf=1,
                      min_samples_split=2, min_weight_fraction_leaf=0.0,
                      n_estimators=100, n_jobs=-1, oob_score=False,
                      random_state=42, verbose=0, warm_start=False)
```

In [56]:

```
# score the model

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

Out[56]:

```
0.9875468079970563
```

Question why the baove metric is not holding water? (why it is not reliable)

Splitting data into train and validation set

In [57]:

```
df_tmp.saleYear
```

Out[57]:

```
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 [58]:

```
df_tmp.saleYear.value_counts()
```

Out[58]:

```
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 [59]:

```
# splitting data into training and validation
df_val = df_tmp[df_tmp.saleYear == 2012]
df_train= df_tmp[df_tmp.saleYear != 2012]
len(df_val),len(df_tmp)
```

Out[59]:

```
(11573, 412698)
```

In [60]:

```
# 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[60]:

```
((401125, 102), (401125,), (11573, 102), (11573,))
```

In [61]:

```
y_train
```

Out[61]:

```
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 [62]:

```
# 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 a subset (to tune the hyperparameters)

In [63]:

```
# this takes far too long... in an experiments
# %%time
# model=RandomForestRegressor(n_jobs=-1,
  #                             random_state=42)

# model.fit(x_train,y_train)
```

In [64]:

```
len(x_train)
```

Out[64]:

```
401125
```

In [65]:

```
# change max samples values
model=RandomForestRegressor(n_jobs=-1,
                            random_state=42,
                            max_samples=10000)
```

In [66]:

```
%%time
# cutting down the max number of samples each estimator can see improves training time
model.fit(x_train,y_train)
```

```
Wall time: 16.5 s
```

Out[66]:

```
RandomForestRegressor(bootstrap=True, ccp_alpha=0.0, criterion='mse',
                      max_depth=None, max_features='auto', max_leaf_nodes=None,
                      max_samples=10000, min_impurity_decrease=0.0,
                      min_impurity_split=None, min_samples_leaf=1,
                      min_samples_split=2, min_weight_fraction_leaf=0.0,
                      n_estimators=100, n_jobs=-1, oob_score=False,
                      random_state=42, verbose=0, warm_start=False)
```

In [67]:

```
(x_train.shape[0]*100)/1000000
```

Out[67]:

```
40.1125
```

In [68]:

```
show_scores(model)
```

Out[68]:

```
{'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 tunning with RandomizedSearchCV

In [69]:

```
%%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]
         }

#instantiate RandomizedSearchCV Model
rs_model=RandomizedSearchCV(RandomForestRegressor(n_jobs=-1,
                                                  random_state=42),
                            param_distributions=rf_grid,
                            n_iter=2,
                            cv=5,
                            verbose=True)
# fit the RandomizedSearchCV
rs_model.fit(x_train,y_train)
```

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

```
[Parallel(n_jobs=1)]: Using backend SequentialBackend with 1 concurrent workers.
```

```
[Parallel(n_jobs=1)]: Using backend SequentialBackend with 1 concurrent workers.
[Parallel(n_jobs=1)]: Done  10 out of  10 | elapsed:  1.7min finished
```

```
Wall time: 1min 46s
```

Out[69]:

```
RandomizedSearchCV(cv=5, error_score=nan,
                   estimator=RandomForestRegressor(bootstrap=True,
                                                   ccp_alpha=0.0,
                                                   criterion='mse',
                                                   max_depth=None,
                                                   max_features='auto',
                                                   max_leaf_nodes=None,
                                                   max_samples=None,
                                                   min_impurity_decrease=0.0,
                                                   min_impurity_split=None,
                                                   min_samples_leaf=1,
                                                   min_samples_split=2,
                                                   min_weight_fraction_leaf=0.0,
                                                   n_estimators=100, n_jobs=-1,
                                                   oob_score=False,...
                   param_distributions={'max_depth': [None, 3, 5, 10],
                                        'max_features': [0.5, 1, 'sqrt',
                                                         'auto'],
                                        'max_samples': [10000],
                                        'min_samples_leaf': array([ 1,  3,  5,  7,  9, 1
1, 13, 15, 17, 19]),
                                        'min_samples_split': array([ 2,  4,  6,  8, 10,
12, 14, 16, 18]),
                                        'n_estimators': array([10, 20, 30, 40, 50, 60, 7
0, 80, 90])},
                   pre_dispatch='2*n_jobs', random_state=None, refit=True,
                   return_train_score=False, scoring=None, verbose=True)
```

In [70]:

```
#find the best model hyperparameters
rs_model.best_params_
```

Out[70]:

```
{'n_estimators': 60,
 'min_samples_split': 12,
 'min_samples_leaf': 1,
 'max_samples': 10000,
 'max_features': 1,
 'max_depth': None}
```

In [71]:

```
# evaluate the RandomizedSearchModel
show_scores(rs_model)
```

Out[71]:

```
{'Training MAE': 8891.655193695899,
 'Valid MAE': 11313.549827603978,
 'Training RMSLE': 0.39237058829152377,
 'Valid RMSLE': 0.4484052255971633,
 'Training R^2': 0.6825547447927643,
 'Valid R^2': 0.6361784117343763}
```

train a model with best hyperparameter

Note- these were found after 100 interation of 'RandomizedSearchCV'

In [72]:

```
%%time

ideal_model=RandomForestRegressor(n_estimators=40,
```

```
                                   min_samples_leaf=1,
                                   min_samples_split=14,
                                   max_features=0.5,
                                   n_jobs=-1,
                                   max_samples=None)

# fit the ideal model
ideal_model.fit(x_train,y_train)
```

```
Wall time: 1min 12s
```

Out[72]:

```
RandomForestRegressor(bootstrap=True, ccp_alpha=0.0, criterion='mse',
                      max_depth=None, max_features=0.5, max_leaf_nodes=None,
                      max_samples=None, min_impurity_decrease=0.0,
                      min_impurity_split=None, min_samples_leaf=1,
                      min_samples_split=14, min_weight_fraction_leaf=0.0,
                      n_estimators=40, n_jobs=-1, oob_score=False,
                      random_state=None, verbose=0, warm_start=False)
```

In [73]:

```
# score on ideal_model(trained on all data)
show_scores(ideal_model)
```

Out[73]:

```
{'Training MAE': 2953.763471164893,
 'Valid MAE': 5945.451639564246,
 'Training RMSLE': 0.14471388482968298,
 'Valid RMSLE': 0.2454815012608934,
 'Training R^2': 0.9588318691876017,
 'Valid R^2': 0.8823200000439747}
```

In [74]:

```
# score on rs_model(trained on -10000 data)
show_scores(rs_model)
```

Out[74]:

```
{'Training MAE': 8891.655193695899,
 'Valid MAE': 11313.549827603978,
 'Training RMSLE': 0.39237058829152377,
 'Valid RMSLE': 0.4484052255971633,
 'Training R^2': 0.6825547447927643,
 'Valid R^2': 0.6361784117343763}
```

Make prediction on test data

In [82]:

```
# import the test data set
df_test=pd.read_csv("data/Test.csv",
                    low_memory=False,
                    parse_dates=["saledate"])

df_test.head()
```

Out[82]:

	SalesID	MachineID	ModelID	datasource	auctioneerID	YearMade	MachineHoursCurrentMeter	UsageBand	saledate	fiM
0	1227829	1006309	3168	121	3	1999	3688.0	Low	2012-05-03	
1	1227844	1022817	7271	121	3	1000	28555.0	High	2012-05-10	
2	1227847	1031560	22805	121	3	2004	6038.0	Medium	2012-05-10	E

	SalesID	MachineID	ModelID	datasource	auctioneerID	YearMade	MachineHoursCurrentMeter	UsageBand	saledate	fiM
3	1227848	56204	1269	121	3	2006	8940.0	High	2012-05-10	
4	1227863	1053887	22312	121	3	2005	2286.0	Low	2012-05-10	

5 rows × 52 columns

In [83]:

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

```
---------------------------------------------------------------------------
ValueError                                Traceback (most recent call last)
<ipython-input-83-2fbd4c21f375> in <module>
      1 # make prediction on the test dataset
----> 2 test_preds=ideal_model.predict(df_test)

~\anaconda3\lib\site-packages\sklearn\ensemble\_forest.py in predict(self, X)
    764         check_is_fitted(self)
    765         # Check data
--> 766         X = self._validate_X_predict(X)
    767 
    768         # Assign chunk of trees to jobs

~\anaconda3\lib\site-packages\sklearn\ensemble\_forest.py in _validate_X_predict(self, X)
    410         check_is_fitted(self)
    411 
--> 412         return self.estimators_[0]._validate_X_predict(X, check_input=True)
    413 
    414     @property

~\anaconda3\lib\site-packages\sklearn\tree\_classes.py in _validate_X_predict(self, X, ch
eck_input)
    378         """Validate X whenever one tries to predict, apply, predict_proba"""
    379         if check_input:
--> 380             X = check_array(X, dtype=DTYPE, accept_sparse="csr")
    381             if issparse(X) and (X.indices.dtype != np.intc or
    382                                 X.indptr.dtype != np.intc):

~\anaconda3\lib\site-packages\sklearn\utils\validation.py in check_array(array, accept_sp
arse, accept_large_sparse, dtype, order, copy, force_all_finite, ensure_2d, allow_nd, ens
ure_min_samples, ensure_min_features, warn_on_dtype, estimator)
    529                     array = array.astype(dtype, casting="unsafe", copy=False)
    530                 else:
--> 531                     array = np.asarray(array, order=order, dtype=dtype)
    532             except ComplexWarning:
    533                 raise ValueError("Complex data not supported\n"

~\anaconda3\lib\site-packages\numpy\core\_asarray.py in asarray(a, dtype, order)
     83 
     84     """
---> 85     return array(a, dtype, copy=False, order=order)
     86 
     87 

ValueError: could not convert string to float: 'Low'
```

preprocessing the data (getting the test dataset in the same formet as our training dataset)

In [84]:

```
def preprocess_data(df):
    """
    performs the transformation on df and return transformed df
    """
    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 the numeric rowa with mean
    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 is missing or not
                df[label+"_is_missing"]=pd.isnull(content)
                # fill missing numeric values with median
                df[label] = content.fillna(content.median())



              # filled categorical missing data and turn categories into number
        if not pd.api.types.is_numeric_dtype(content):
            df[label+"is_missing"]=pd.isnull(content)
            # we add +1 to category code because pandas encode missing categories as -1
            df[label]=pd.Categorical(content).codes+1


    return df
```

In [85]:

```
# process the test data
df_test=preprocess_data(df_test)
df_test.head()
```

Out[85]:

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

5 rows × 101 columns

In [86]:

```
# make predictions on updated test data
test_preds=ideal_model.predict(df_test)
```

```
---------------------------------------------------------------------------
ValueError                                Traceback (most recent call last)
<ipython-input-86-2955bc2680bc> in <module>
      1 # make predictions on updated test data
----> 2 test_preds=ideal_model.predict(df_test)

~\anaconda3\lib\site-packages\sklearn\ensemble\_forest.py in predict(self, X)
    764         check_is_fitted(self)
    765         # Check data
--> 766         X = self._validate_X_predict(X)
    767
    768         # Assign chunk of trees to jobs

~\anaconda3\lib\site-packages\sklearn\ensemble\_forest.py in _validate_X_predict(self, X)
    410         check_is_fitted(self)
    411
--> 412         return self.estimators_[0]._validate_X_predict(X, check_input=True)
    413
    414     @property
```

```
~\anaconda3\lib\site-packages\sklearn\tree\_classes.py in _validate_X_predict(self, X, ch
eck_input)
    389                             "match the input. Model n_features is %s and "
    390                             "input n_features is %s "
--> 391                             % (self.n_features_, n_features))
    392
    393         return X

ValueError: Number of features of the model must match the input. Model n_features is 102
and input n_features is 101
```

In [87]:

```
# we can find how the columns differ using python sets
set(x_train.columns)-set(df_test.columns)
```

Out[87]:

```
{'Backhoe_Mounting_is_missing',
 'Blade_Extension_is_missing',
 'Blade_Type_is_missing',
 'Blade_Width_is_missing',
 'Coupler_System_is_missing',
 'Coupler_is_missing',
 'Differential_Type_is_missing',
 'Drive_System_is_missing',
 'Enclosure_Type_is_missing',
 'Enclosure_is_missing',
 'Engine_Horsepower_is_missing',
 'Forks_is_missing',
 'Grouser_Tracks_is_missing',
 'Grouser_Type_is_missing',
 'Hydraulics_Flow_is_missing',
 'Hydraulics_is_missing',
 'Pad_Type_is_missing',
 'Pattern_Changer_is_missing',
 'ProductGroupDesc_is_missing',
 'ProductGroup_is_missing',
 'ProductSize_is_missing',
 'Pushblock_is_missing',
 'Ride_Control_is_missing',
 'Ripper_is_missing',
 'Scarifier_is_missing',
 'Steering_Controls_is_missing',
 'Stick_Length_is_missing',
 'Stick_is_missing',
 'Thumb_is_missing',
 'Tip_Control_is_missing',
 'Tire_Size_is_missing',
 'Track_Type_is_missing',
 'Transmission_is_missing',
 'Travel_Controls_is_missing',
 'Turbocharged_is_missing',
 'Undercarriage_Pad_Width_is_missing',
 'UsageBand_is_missing',
 'auctioneerID_is_missing',
 'fiBaseModel_is_missing',
 'fiModelDesc_is_missing',
 'fiModelDescriptor_is_missing',
 'fiModelSeries_is_missing',
 'fiProductClassDesc_is_missing',
 'fiSecondaryDesc_is_missing',
 'state_is_missing'}
```

In [88]:

```
df_test["auctioneerID_is_missing"]=False
df_test.head()
```

Out[88]:

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

5 rows × 102 columns

finally now our test df has same data as out training df ,now we can make predictions

In [89]:

```
# make predictions on the test data
test_preds=ideal_model.predict(df_test)
```

In [91]:

```
len(test_preds)
```

Out[91]:

```
12457
```

In [92]:

```
test_preds
```

Out[92]:

```
array([18731.50308829, 20415.43511302, 45397.50954757, ...,
       14117.521919  , 21509.87245758, 30646.67265808])
```

we have made predictions but they are not in the same format kaggle is asking for:

In [95]:

```
# format predictions in the same format kaggle is after
df_preds=pd.DataFrame()
df_preds["SalesID"]=df_test["SalesID"]
df_preds["SalesPrice"]=test_preds
df_preds
```

Out[95]:

	SalesID	SalesPrice
0	1227829	18731.503088
1	1227844	20415.435113
2	1227847	45397.509548
3	1227848	66403.957209
4	1227863	41892.349708
...	...	...
12452	6643171	45347.236038
12453	6643173	16734.561534
12454	6643184	14117.521919
12455	6643186	21509.872458
12456	6643196	30646.672658

12457 rows × 2 columns

In [96]:

```
# export prediction data
df_preds.to_csv("data/test_predictions.csv",index=False)
```

FEATURE IMPORTANCE

Feature important seeks to figure out which different attributes of the data were most important when it comes to predicting the target variable (SalePrice)

In [97]:

```
# find feature importance of our best model
ideal_model.feature_importances_
```

Out[97]:

```
array([3.44154778e-02, 1.73038512e-02, 4.04645176e-02, 1.65524910e-03,
       3.40539364e-03, 2.08462191e-01, 2.96450095e-03, 1.06518775e-03,
       4.21484864e-02, 4.70675936e-02, 6.27868928e-02, 4.60644701e-03,
       1.50777011e-02, 1.53516777e-01, 4.66607663e-02, 5.96073254e-03,
       1.49277381e-03, 2.68712781e-03, 2.65716594e-03, 6.03683392e-02,
       8.06790194e-04, 4.15096203e-05, 9.64365879e-04, 2.15775816e-04,
       1.24210513e-03, 1.99480760e-05, 2.01622346e-03, 8.53212792e-03,
       2.15240085e-03, 2.50318286e-03, 4.90089633e-03, 4.23147924e-03,
       2.50846669e-03, 5.69595314e-04, 2.47850601e-04, 6.05925913e-03,
       7.35734603e-04, 1.51805099e-02, 2.29726889e-03, 3.33395889e-03,
       8.25263294e-04, 9.16968827e-04, 1.31821444e-03, 5.77257731e-04,
       4.81436263e-04, 3.67240219e-04, 4.85423546e-04, 2.68320817e-03,
       8.60413658e-04, 3.15534492e-04, 2.12199965e-04, 7.43751875e-02,
       3.78715292e-03, 5.66570602e-03, 2.87503357e-03, 9.83162371e-03,
       2.71271525e-04, 1.40023472e-03, 3.08858351e-04, 0.00000000e+00,
       0.00000000e+00, 2.15792025e-03, 1.02494912e-03, 5.62569056e-03,
       3.45137496e-02, 0.00000000e+00, 0.00000000e+00, 0.00000000e+00,
       0.00000000e+00, 1.49311863e-05, 1.07073398e-05, 1.33507997e-04,
       5.68473516e-06, 1.18596778e-04, 4.31521031e-06, 3.66910279e-04,
       5.56935228e-06, 3.03221651e-03, 3.99213276e-03, 4.07730622e-03,
       6.87293635e-05, 2.42923644e-03, 9.36111150e-05, 3.60394161e-04,
       3.77210900e-04, 1.99734751e-03, 3.64003585e-03, 1.74859311e-04,
       1.02357715e-02, 9.02538684e-04, 2.39236553e-03, 4.44649882e-05,
       3.04361208e-04, 2.16091613e-05, 5.54166923e-05, 3.22492933e-05,
       4.32064962e-05, 2.72691097e-04, 1.90223202e-04, 1.85599967e-04,
       1.23332928e-04, 8.57053886e-05])
```

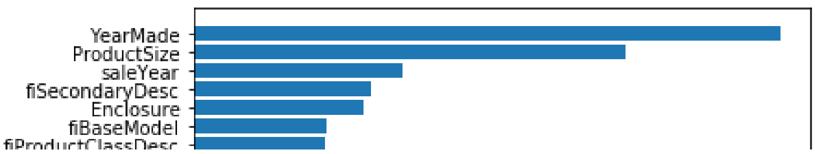
In [107]:

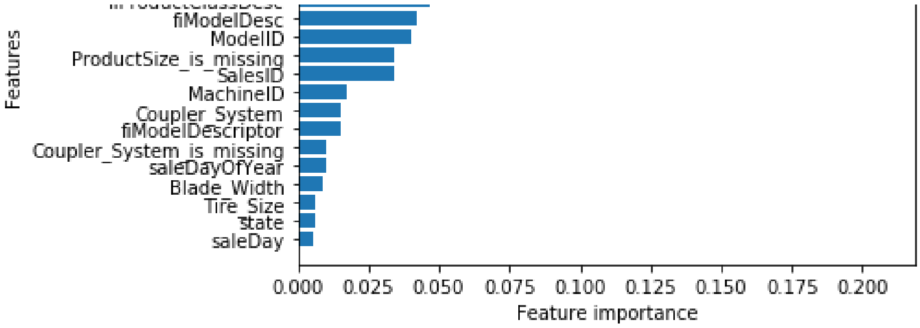
```
# Helper function for plotting feature importance
def plot_features(columns, importances, n=20):
    df = (pd.DataFrame({"features": columns,
                        "feature_importances": importances})
          .sort_values("feature_importances", ascending=False)
          .reset_index(drop=True))

    # 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 [108]:

```
plot_features(x_train.columns,ideal_model.feature_importances_)
```





In [109]:

```
df["ProductSize"].value_counts()
```

Out[109]:

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

In [110]:

```
df["Enclosure"].value_counts()
```

Out[110]:

```
OROPS                      177971
EROPS                      141769
EROPS w AC                  92601
EROPS AC                       18
NO ROPS                         3
None or Unspecified             2
Name: Enclosure, dtype: int64
```

In [ ]: