

Predicting the sale price of Bulldozers using machine learning

In this notebook we are going to predict sale price of bulldozers based on real dataset provided on Kaggle.

1. Problem Definition

How well can we predict the future sale price of a bulldozer based on previous sale prices and characteristics (features/data).

2. Data

This data is taken from the Kaggle Bluebook for Bulldozers competition: (<https://www.kaggle.com/c/bluebook-for-bulldozers/data>)

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
- Test.csv is the test set, which won't be released until the last week of the competition.

The key fields in train.csv are:

- SalesID: the unique 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.

4. Features

Link to data dictionary on Kaggle (<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 set  
df = pd.read_csv("data/TrainAndValid.csv", low_memory=False)  
df.head(), 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

```

```

Out[2]: (  SalesID  SalePrice  MachineID  ModelID  datasource  auctioneerID  YearMade
\
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

    MachineHoursCurrentMeter  UsageBand          saledate  ...  \
0                68.0          Low  11/16/2006 0:00  ...
1             4640.0          Low   3/26/2004 0:00  ...
2             2838.0         High   2/26/2004 0:00  ...
3             3486.0         High   5/19/2011 0:00  ...
4              722.0        Medium   7/23/2009 0:00  ...

    Undercarriage_Pad_Width  Stick_Length  Thumb  Pattern_Changer  Grouser_Type  \
0                NaN          NaN    NaN          NaN          NaN
1                NaN          NaN    NaN          NaN          NaN
2                NaN          NaN    NaN          NaN          NaN
3                NaN          NaN    NaN          NaN          NaN
4                NaN          NaN    NaN          NaN          NaN

    Backhoe_Mounting  Blade_Type  Travel_Controls  Differential_Type  \
0                NaN          NaN          NaN          Standard
1                NaN          NaN          NaN          Standard
2                NaN          NaN          NaN          NaN
3                NaN          NaN          NaN          NaN
4                NaN          NaN          NaN          NaN

    Steering_Controls
0    Conventional
1    Conventional
2                NaN
3                NaN
4                NaN

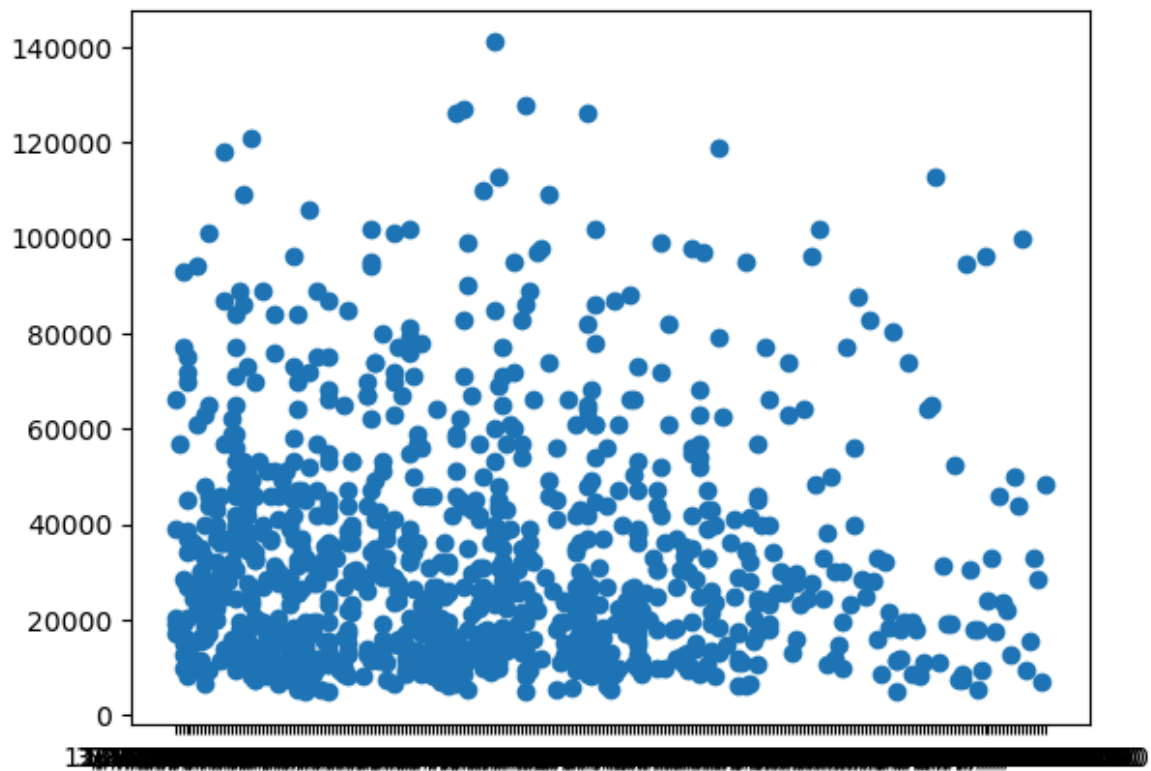
[5 rows x 53 columns],
None)

```

```

In [3]: fig, ax = plt.subplots()
ax.scatter(df["saledate"][:1000], df["SalePrice"][:1000]);

```

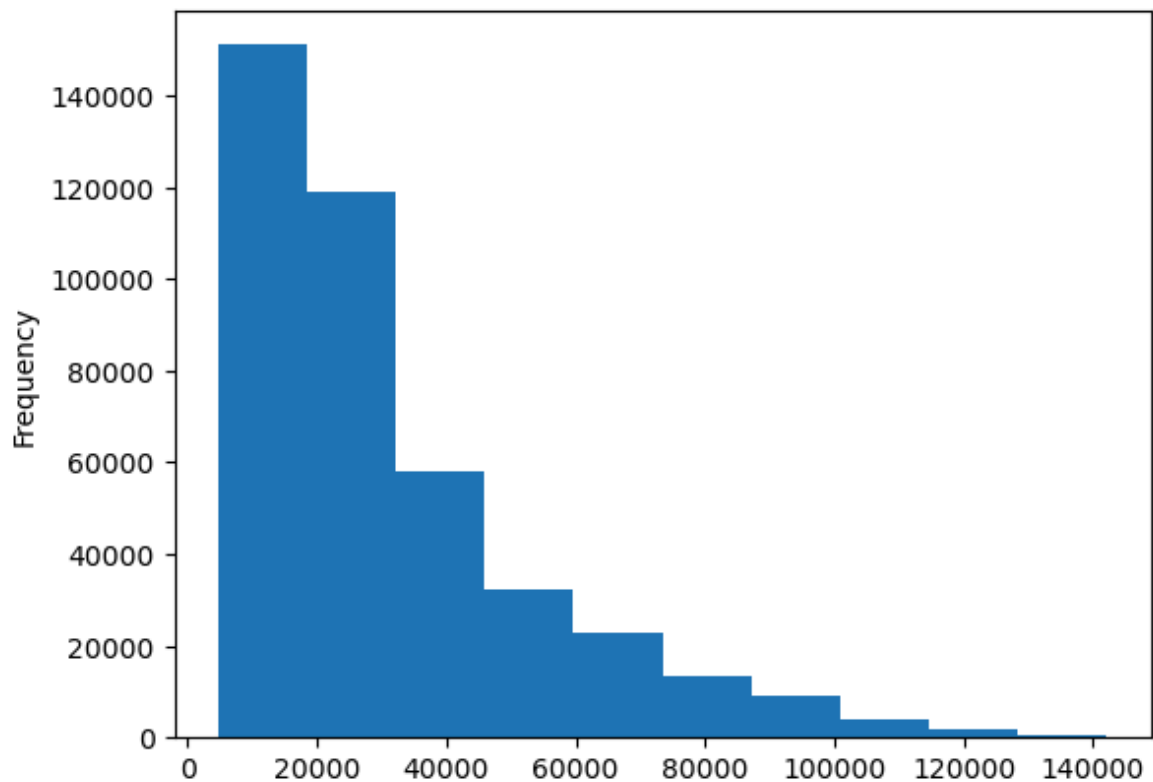


```
In [4]: df.saledate[:1000]
```

```
Out[4]: 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 [5]: df.SalePrice.plot.hist()
```

```
Out[5]: <Axes: ylabel='Frequency'>
```



Parsing dates

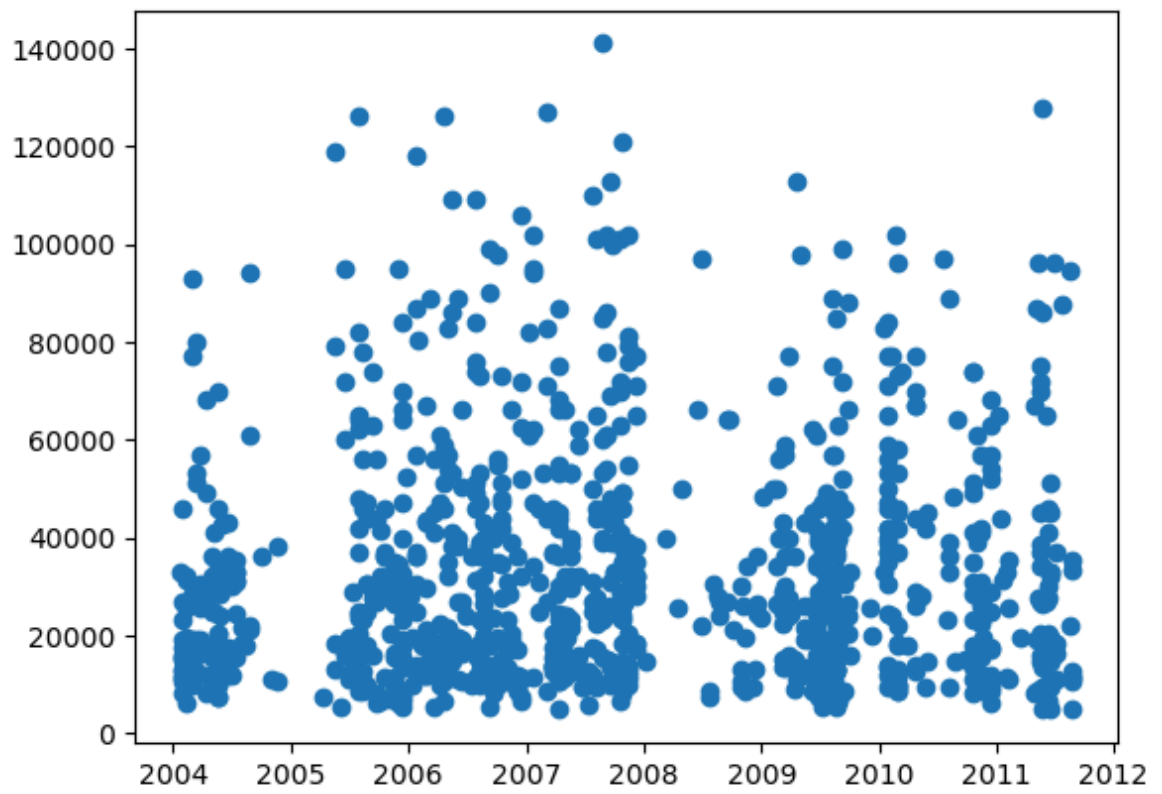
Enriching time and date component

```
In [6]: # Import data again but this time parse dates
df = pd.read_csv("data/TrainAndValid.csv",
                 low_memory=False,
                 parse_dates=["saledate"])
```

```
In [7]: df.saledate[:1000]
```

```
Out[7]: 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 [8]: fig, ax = plt.subplots()
ax.scatter(df["saledate"][:1000], df["SalePrice"][:1000]);
```



```
In [9]: df.head().T
```

Out[9]:

	0	1	2	3	
SalesID	1139246	1139248	1139249	1139251	1139252
SalePrice	66000.0	57000.0	10000.0	38500.0	11000.0
MachineID	999089	117657	434808	1026470	1057089
ModelID	3157	77	7009	332	1700
datasource	121	121	121	121	121
auctioneerID	3.0	3.0	3.0	3.0	3.0
YearMade	2004	1996	2001	2001	2001
MachineHoursCurrentMeter	68.0	4640.0	2838.0	3486.0	7200.0
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	2009-07-01 00:00:00
fiModelDesc	521D	950FII	226	PC120-6E	S
fiBaseModel	521	950	226	PC120	S
fiSecondaryDesc	D	F	NaN	NaN	N
fiModelSeries	NaN	II	NaN	-6E	N
fiModelDescriptor	NaN	NaN	NaN	NaN	N
ProductSize	NaN	Medium	NaN	Small	N
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 Load 1601.0 1751.0 Oper...
state	Alabama	North Carolina	New York	Texas	New York
ProductGroup	WL	WL	SSL	TEX	
ProductGroupDesc	Wheel Loader	Wheel Loader	Skid Steer Loaders	Track Excavators	Skid Steer Loaders
Drive_System	NaN	NaN	NaN	NaN	N
Enclosure	EROPS w AC	EROPS w AC	OROPS	EROPS w AC	ERC
Forks	None or Unspecified	None or Unspecified	None or Unspecified	NaN	None or Unspecified
Pad_Type	NaN	NaN	NaN	NaN	N
Ride_Control	None or Unspecified	None or Unspecified	NaN	NaN	N
Stick	NaN	NaN	NaN	NaN	N
Transmission	NaN	NaN	NaN	NaN	N

	0	1	2	3	
Turbocharged	NaN	NaN	NaN	NaN	NaN
Blade_Extension	NaN	NaN	NaN	NaN	NaN
Blade_Width	NaN	NaN	NaN	NaN	NaN
Enclosure_Type	NaN	NaN	NaN	NaN	NaN
Engine_Horsepower	NaN	NaN	NaN	NaN	NaN
Hydraulics	2 Valve	2 Valve	Auxiliary	2 Valve	Auxiliary
Pushblock	NaN	NaN	NaN	NaN	NaN
Ripper	NaN	NaN	NaN	NaN	NaN
Scarifier	NaN	NaN	NaN	NaN	NaN
Tip_Control	NaN	NaN	NaN	NaN	NaN
Tire_Size	None or Unspecified	23.5	NaN	NaN	NaN
Coupler	None or Unspecified	None or Unspecified	None or Unspecified	None or Unspecified	None or Unspecified
Coupler_System	NaN	NaN	None or Unspecified	NaN	None or Unspecified
Grouser_Tracks	NaN	NaN	None or Unspecified	NaN	None or Unspecified
Hydraulics_Flow	NaN	NaN	Standard	NaN	Standard
Track_Type	NaN	NaN	NaN	NaN	NaN
Undercarriage_Pad_Width	NaN	NaN	NaN	NaN	NaN
Stick_Length	NaN	NaN	NaN	NaN	NaN
Thumb	NaN	NaN	NaN	NaN	NaN
Pattern_Changer	NaN	NaN	NaN	NaN	NaN
Grouser_Type	NaN	NaN	NaN	NaN	NaN
Backhoe_Mounting	NaN	NaN	NaN	NaN	NaN
Blade_Type	NaN	NaN	NaN	NaN	NaN
Travel_Controls	NaN	NaN	NaN	NaN	NaN
Differential_Type	Standard	Standard	NaN	NaN	NaN
Steering_Controls	Conventional	Conventional	NaN	NaN	NaN

```
In [10]: df.saledate.head()
```

```
Out[10]: 0    2006-11-16
         1    2004-03-26
         2    2004-02-26
         3    2011-05-19
         4    2009-07-23
         Name: saledate, dtype: datetime64[ns]
```

Sort data by saledate

```
In [11]: # Sort dataframe in date order
         df.sort_values(by=["saledate"], inplace=True, ascending=True)
         df.saledate.head(20)
```

```
Out[11]: 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

```
In [12]: df_tmp = df.copy()
         df_tmp.head(10)
```

Out[12]:

	SalesID	SalePrice	MachineID	ModelID	datasource	auctioneerID	YearMade
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
54653	1301884	23500.0	1182999	4123	132	99.0	1976
81383	1379228	31000.0	1082797	7620	132	99.0	1986
204924	1645390	11750.0	1527216	8202	132	99.0	1970
135376	1493279	63000.0	1363756	2759	132	99.0	1987
113390	1449549	13000.0	1289412	3356	132	99.0	1966

10 rows × 53 columns



Add datetime parameters for `saledate` column

```
In [13]: df_tmp["saleYear"] = df_tmp.saledate.dt.year
df_tmp["saleMonth"] = df_tmp.saledate.dt.month
df_tmp["saleDate"] = df_tmp.saledate.dt.day
df_tmp["saleDayOfWeek"] = df_tmp.saledate.dt.dayofweek
df_tmp["saleDayOfYear"] = df_tmp.saledate.dt.dayofyear

In [77]: df_tmp.head().T
```

Out[77]:

	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 [15]: # Now we have enriched our data with dat time features, we cam remove saledate
df_tmp.drop("saledate", axis = 1, inplace = True)
```

```
In [16]: # check the values of other columns
df_tmp.state.value_counts()
```

```
Out[16]: state
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: count, dtype: int64
```

5. Modelling

Model driven EDA

Building our 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"]) This  
will show error as not all values are numerical and not all values are filled
```

```
In [17]: df_tmp.info(), df_tmp.isna().sum()
```

```
<class 'pandas.core.frame.DataFrame'>
```

```
Index: 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	int32
53	saleMonth	412698 non-null	int32
54	saleDate	412698 non-null	int32

```
55  saleDayOfWeek      412698 non-null  int32
56  saleDayOfYear      412698 non-null  int32
dtypes: float64(3), int32(5), int64(5), object(44)
memory usage: 174.7+ MB
```



```

Out[17]: (None,
          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
          saleDate              0
          saleDayOfWeek         0
          saleDayOfYear         0
          dtype: int64)

```

convert string to categories

One way of doing it is converting them into pandas categories.

```
In [18]: # Finding column containing strings
for label, content in df_tmp.items():
    if pd.api.types.is_object_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 [19]: # Changing all string values in category value
for label, content in df_tmp.items():
    if pd.api.types.is_object_dtype(content):
        df_tmp[label] = content.astype("category").cat.as_ordered()
```

```
In [20]: df_tmp.info()
```

```
<class 'pandas.core.frame.DataFrame'>
```

```
Index: 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	int32
53	saleMonth	412698 non-null	int32
54	saleDate	412698 non-null	int32

```
55  saleDayOfWeek      412698 non-null  int32
56  saleDayOfYear      412698 non-null  int32
dtypes: category(44), float64(3), int32(5), int64(5)
memory usage: 55.4 MB
```

```
In [21]: df_tmp.state.cat.categories
```

```
Out[21]: 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 [22]: df_tmp.state.cat.codes
```

```
Out[22]: 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 [23]: # Check missing data
df_tmp.isnull().sum()/len(df_tmp)
```

```

Out[23]: 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
         saleDate          0.000000
         saleDayOfWeek     0.000000
         saleDayOfYear     0.000000
         dtype: float64

```

Saving preprocessed data

```
In [24]: # Export current temp dataframe  
df_tmp.to_csv("data/train_tmp.csv",  
              index=False)
```

```
In [25]: # Import preprocessed data  
df_tmp = pd.read_csv("data/train_tmp.csv",  
                     low_memory=False)  
df_tmp.head().T
```

Out[25]:

	0	1	2	3	
SalesID	1646770	1821514	1505138	1671174	13290
SalePrice	9500.0	14000.0	50000.0	16000.0	2200
MachineID	1126363	1194089	1473654	1327630	13360
ModelID	8434	10150	4139	8591	40
datasource	132	132	132	132	
auctioneerID	18.0	99.0	99.0	99.0	9
YearMade	1974	1980	1978	1980	19
MachineHoursCurrentMeter	NaN	NaN	NaN	NaN	NaN
UsageBand	NaN	NaN	NaN	NaN	NaN
fiModelDesc	TD20	A66	D7G	A62	D
fiBaseModel	TD20	A66	D7	A62	
fiSecondaryDesc	NaN	NaN	G	NaN	
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 T Trac Doz 20.0 to 7 Hor...
state	Texas	Florida	Florida	Florida	Flor
ProductGroup	TTT	WL	TTT	WL	
ProductGroupDesc	Track Type Tractors	Wheel Loader	Track Type Tractors	Wheel Loader	Track T Tract
Drive_System	NaN	NaN	NaN	NaN	NaN
Enclosure	OROPS	OROPS	OROPS	EROPS	ORC
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	Stand
Turbocharged	NaN	NaN	NaN	NaN	NaN
Blade_Extension	NaN	NaN	NaN	NaN	NaN

	0	1	2	3	
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	NaN
Travel_Controls	None or Unspecified	NaN	None or Unspecified	NaN	Le
Differential_Type	NaN	Standard	NaN	Standard	NaN
Steering_Controls	NaN	Conventional	NaN	Conventional	NaN
saleYear	1989	1989	1989	1989	19
saleMonth	1	1	1	1	
saleDate	17	31	31	31	
saleDayOfWeek	1	1	1	1	
saleDayOfYear	17	31	31	31	

```
In [26]: df_tmp.isna().sum()
```

```

Out[26]: 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
         saleDate              0
         saleDayOfWeek         0
         saleDayOfYear         0
         dtype: int64

```

Filling the missing values

```
In [27]: # Filling numerical values
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
saleDate
saleDayOfWeek
saleDayOfYear

```
In [28]: # checking 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 [29]: # Filling the missing values
for label, content in df_tmp.items():
    if pd.api.types.is_numeric_dtype(content):
        if pd.isnull(content).sum():
            #checking for missing data
            df_tmp[label+"is_missing"]=pd.isnull(content)
            #filling the missing values
            df_tmp[label] = content.fillna(content.median())
```

```
In [30]: # check for remaining null values if any
for label, content in df_tmp.items():
    if pd.api.types.is_numeric_dtype(content):
        if pd.isnull(content).sum():
            print(label)
```

```
In [31]: df_tmp.isna().sum()
```

```

Out[31]: 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
         saleDate             0
         saleDayOfWeek        0
         saleDayOfYear        0
         auctioneerIDis_missing 0
         MachineHoursCurrentMeteris_missing 0
         dtype: int64

```

Filling and turning ctegorical variables into numbers

```
In [32]: # checking for non numeric columns
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 [33]: # Turning categorical variables into numbers and fill the missing values
for label, content in df_tmp.items():
    if not pd.api.types.is_numeric_dtype(content):
        df_tmp[label+"_is_missing"] = pd.isnull(content)
        df_tmp[label] = pd.Categorical(content).codes+1
```

In [34]: `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 [35]: `df_tmp.head().T`

Out[35]:

	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 [36]: `df_tmp.isna().sum()`

Out[36]:

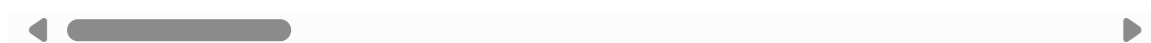
```
SalesID      0
SalePrice    0
MachineID    0
ModelID      0
datasource   0
...
Backhoe_Mounting_is_missing  0
Blade_Type_is_missing      0
Travel_Controls_is_missing  0
Differential_Type_is_missing 0
Steering_Controls_is_missing 0
Length: 103, dtype: int64
```

In [37]: `df_tmp.head()`

Out[37]:

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

5 rows × 103 columns



%%time

Intantiate the model

```
from sklearn.ensemble import RandomForestRegressor model =
RandomForestRegressor(n_jobs = -1, random_state=42)
```

Fit the model

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

Scoring the model

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

```
In [38]: # Splitting data into train and validation, based on year as mentioned in the da
df_val = df_tmp[df_tmp.saleYear == 2012]
df_train = df_tmp[df_tmp.saleYear != 2012]
len(df_val), len(df_train)
```

Out[38]: (11573, 401125)

```
In [39]: # 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[39]: ((401125, 102), (401125,), (11573, 102), (11573,))

Building an Evaluation Function

```
In [48]: # Create evaluation function (RMSLE)
from sklearn.metrics import mean_squared_log_error, mean_absolute_error, r2_score
```



```
def rmsle(y_test, y_preds):
    return np.sqrt(mean_squared_log_error(y_test, y_preds))

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_valid, val_preds),
              "Valid R^2": r2_score(y_valid, val_preds)}
    return scores
```

Testing our model on subset (Tuning Hyperparameters)

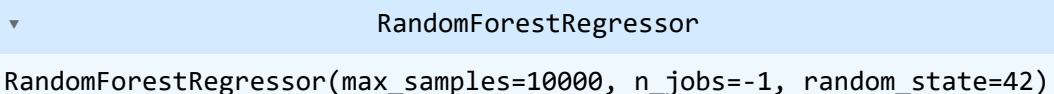
In [42]: `from sklearn.ensemble import RandomForestRegressor`

In [43]: `# change the max_sample value`
`model = RandomForestRegressor(n_jobs=-1,`
 `random_state=42,`
 `max_samples=10000)`

In [44]: `%%time`
`model.fit(x_train, y_train)`

CPU times: total: 1min 33s

Wall time: 25.2 s

Out[44]: 

In [49]: `show_scores(model)`

Out[49]: `{'Training MAE': 5561.2988092240585,`
`'Valid MAE': 7177.26365505919,`
`'Training RMSLE': 0.257745378256977,`
`'Valid RMSLE': 0.29362638671089003,`
`'Training R^2': 0.8320374995090507,`
`'Valid R^2': 0.8320374995090507}`

Hyperparameter tuning with RandomizedSearchCV

In [57]: `%%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)
rs_model.fit(x_train, y_train)
```

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

C:\Users\suraj\Desktop\projects\Bulldozer-Price-Predictor-Project\env\Lib\site-packages\sklearn\model_selection_validation.py:425: FitFailedWarning:

5 fits failed out of a total of 10.

The score on these train-test partitions for these parameters will be set to nan. If these failures are not expected, you can try to debug them by setting error_score='raise'.

Below are more details about the failures:

5 fits failed with the following error:

Traceback (most recent call last):

File "C:\Users\suraj\Desktop\projects\Bulldozer-Price-Predictor-Project\env\Lib\site-packages\sklearn\model_selection_validation.py", line 732, in _fit_and_score

estimator.fit(X_train, y_train, **fit_params)

File "C:\Users\suraj\Desktop\projects\Bulldozer-Price-Predictor-Project\env\Lib\site-packages\sklearn\base.py", line 1144, in wrapper

estimator._validate_params()

File "C:\Users\suraj\Desktop\projects\Bulldozer-Price-Predictor-Project\env\Lib\site-packages\sklearn\base.py", line 637, in _validate_params

validate_parameter_constraints(

File "C:\Users\suraj\Desktop\projects\Bulldozer-Price-Predictor-Project\env\Lib\site-packages\sklearn\utils_param_validation.py", line 95, in validate_parameter_constraints

raise InvalidParameterError(

sklearn.utils._param_validation.InvalidParameterError: The 'max_features' parameter of RandomForestRegressor must be an int in the range [1, inf), a float in the range (0.0, 1.0], a str among {'log2', 'sqrt'} or None. Got 'auto' instead.

warnings.warn(some_fits_failed_message, FitFailedWarning)

C:\Users\suraj\Desktop\projects\Bulldozer-Price-Predictor-Project\env\Lib\site-packages\sklearn\model_selection_search.py:976: UserWarning: One or more of the test scores are non-finite: [0.28841056 nan]

warnings.warn(

CPU times: total: 5 s

Wall time: 12 s

```
Out[57]: RandomizedSearchCV
          estimator: RandomForestRegressor
              RandomForestRegressor
```

```
In [58]: # Finding the best fit for the model
rs_model.best_params_
```

```
Out[58]: {'n_estimators': 40,
          'min_samples_split': 10,
          'min_samples_leaf': 19,
          'max_samples': 10000,
          'max_features': 1,
          'max_depth': 5}
```

```
In [59]: # Evaluate the RandomizedSearch model
show_scores(rs_model)
```

```
Out[59]: {'Training MAE': 14177.373457499434,
          'Valid MAE': 16224.907089308534,
          'Training RMSLE': 0.6040226316645464,
          'Valid RMSLE': 0.6109480028897742,
          'Training R^2': 0.2837478978453394,
          'Valid R^2': 0.2837478978453394}
```

Train a model with the best hyperparameters

Note: These were found after 100 iterations

```
In [61]: %%time

# Most ideal hyperparameter
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)

ideal_model.fit(x_train, y_train)
```

CPU times: total: 5min 47s

Wall time: 1min 34s

```
Out[61]: ▼ RandomForestRegressor

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

```
In [62]: show_scores(ideal_model)
```

```
Out[62]: {'Training MAE': 2953.8161137163484,
          'Valid MAE': 5951.247761444453,
          'Training RMSLE': 0.14469006962371858,
          'Valid RMSLE': 0.2452416398953833,
          'Training R^2': 0.8818019502450094,
          'Valid R^2': 0.8818019502450094}
```

Make predictions on test data

```
In [146... # Import test data
df_test = pd.read_csv("data/test.csv",
                      low_memory=False,
```

```

        parse_dates=["saledate"])
df_test.head()

```

Out[146...

	SalesID	MachineID	ModelID	datasource	auctioneerID	YearMade	MachineHoursC
0	1227829	1006309	3168	121	3	1999	
1	1227844	1022817	7271	121	3	1000	
2	1227847	1031560	22805	121	3	2004	
3	1227848	56204	1269	121	3	2006	
4	1227863	1053887	22312	121	3	2005	

5 rows × 52 columns



Preprocessing test data for testing on model

In [147...

```

def preprocess_data(df):
    df["saleYear"] = df.saledate.dt.year
    df["saleMonth"] = df.saledate.dt.month
    df["saleDate"] = df.saledate.dt.day
    df["saleDayOfWeek"] = df.saledate.dt.dayofweek
    df["saleDayOfYear"] = df.saledate.dt.dayofyear
    df.drop("saledate", axis=1, inplace=True)

    # Filling the missing values
    for label, content in df.items():
        if pd.api.types.is_numeric_dtype(content):
            if pd.isnull(content).sum():
                #checking for missing data
                df[label+"_is_missing"] = pd.isnull(content)
                #filling the missing values
                df[label] = content.fillna(content.median())

        if not pd.api.types.is_numeric_dtype(content):
            df[label+"_is_missing"] = pd.isnull(content)
            # We add +1 to the category code because pandas encodes missing cate
            df[label] = pd.Categorical(content).codes+1

    return df

```

In [148...

```

df_test = preprocess_data(df_test)
df_test.head()

```

Out[148...

	SalesID	MachineID	ModelID	datasource	auctioneerID	YearMade	MachineHoursC
0	1227829	1006309	3168	121	3	1999	
1	1227844	1022817	7271	121	3	1000	
2	1227847	1031560	22805	121	3	2004	
3	1227848	56204	1269	121	3	2006	
4	1227863	1053887	22312	121	3	2005	

5 rows × 101 columns



In [149...

```
x_train.head()
```

Out[149...

	SalesID	MachineID	ModelID	datasource	auctioneerID	YearMade	MachineHoursC
0	1646770	1126363	8434	132	18.0	1974	
1	1821514	1194089	10150	132	99.0	1980	
2	1505138	1473654	4139	132	99.0	1978	
3	1671174	1327630	8591	132	99.0	1980	
4	1329056	1336053	4089	132	99.0	1984	

5 rows × 102 columns



In [150...

```
column_number = x_train.columns.get_loc("auctioneerIDis_missing")
print("Column number of 'auctioneerIDis_missing' column:", column_number)
```

Column number of 'auctioneerIDis_missing' column: 56

In [151...

```
set(x_train.columns)-set(df_test.columns)
```

Out[151...

```
{'MachineHoursCurrentMeteris_missing', 'auctioneerIDis_missing'}
```

In [152...

```
# Manually adding missing column
df_test["auctioneerIDis_missing"] = False
```

In [153...

```
# Get the index of the "auctioneerIDis_missing" column
column_index = df_test.columns.get_loc("auctioneerIDis_missing")

# Reorder the columns to move "auctioneerIDis_missing" to the 56th position
columns_reordered = list(df_test.columns)
columns_reordered.insert(56, columns_reordered.pop(column_index))
df_test = df_test[columns_reordered]
df_test.rename(columns={'MachineHoursCurrentMeter_is_missing': 'MachineHoursCurr
df_test.head()
```

Out[153...

	SalesID	MachineID	ModelID	datasource	auctioneerID	YearMade	MachineHoursC
0	1227829	1006309	3168	121	3	1999	
1	1227844	1022817	7271	121	3	1000	
2	1227847	1031560	22805	121	3	2004	
3	1227848	56204	1269	121	3	2006	
4	1227863	1053887	22312	121	3	2005	

5 rows × 102 columns

In [154...

```
set(x_train.columns)-set(df_test.columns)
```

Out[154...

```
set()
```

In [157...

```
# Make prediction on test data
test_preds = ideal_model.predict(df_test)
```

In [158...

```
test_preds
```

Out[158...

```
array([17030.00927386, 14355.53565165, 46623.08774286, ...,
       11964.85073347, 16496.71079281, 27119.99044029])
```

In [161...

```
# Formatting prediction according to needed data
df_preds = pd.DataFrame()
df_preds["SalesID"] = df_test["SalesID"]
df_preds["SalesPrice"] = test_preds
df_preds
```

Out[161...

	SalesID	SalesPrice
0	1227829	17030.009274
1	1227844	14355.535652
2	1227847	46623.087743
3	1227848	71680.261335
4	1227863	61762.999424
...
12452	6643171	39966.363007
12453	6643173	12049.704433
12454	6643184	11964.850733
12455	6643186	16496.710793
12456	6643196	27119.990440

12457 rows × 2 columns

```
In [162... # Exportr Prediction Data
df_preds.to_csv("data/test_prediction.csv", index=False)
```

Feature Importance

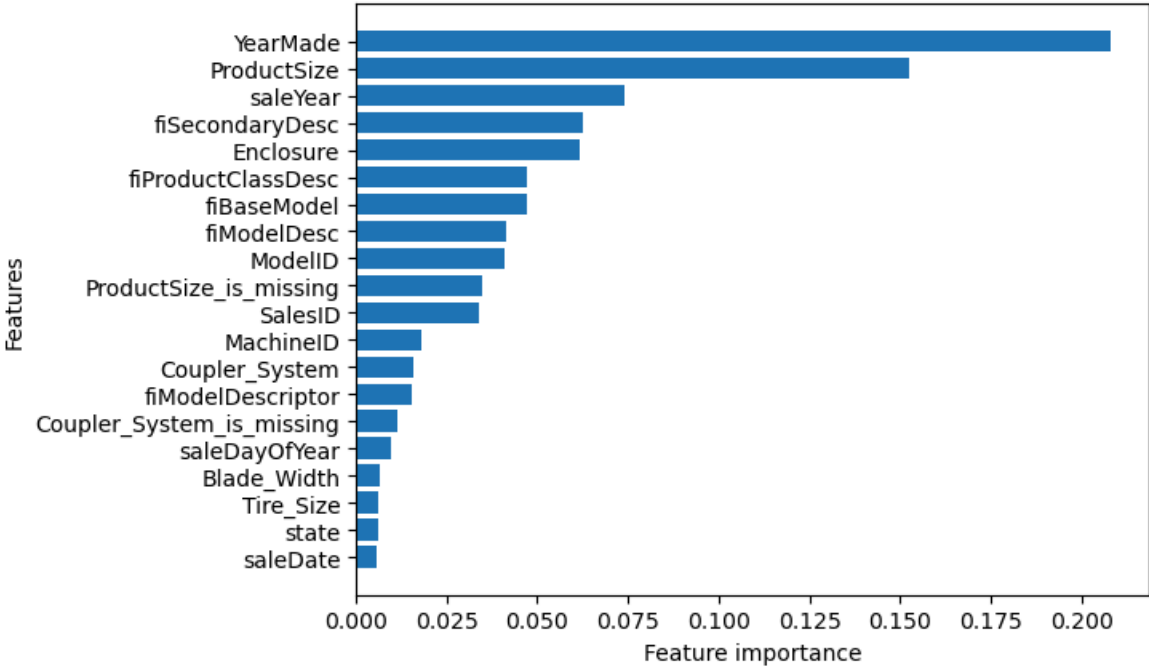
```
In [163... # Finding feature importance of our best model
ideal_model.feature_importances_
```

```
Out[163... array([3.39445533e-02, 1.81148281e-02, 4.09167072e-02, 1.70752171e-03,
       3.40797459e-03, 2.08200698e-01, 2.95067052e-03, 1.10113725e-03,
       4.16122668e-02, 4.71911805e-02, 6.23815431e-02, 4.67433955e-03,
       1.52524442e-02, 1.52517337e-01, 4.72224713e-02, 5.96817956e-03,
       1.29351899e-03, 2.78088439e-03, 2.37248769e-03, 6.17114453e-02,
       8.13525488e-04, 3.61873268e-05, 9.19098115e-04, 2.23170993e-04,
       1.28102678e-03, 2.06519636e-05, 2.01477316e-03, 6.63364759e-03,
       2.15274492e-03, 2.50178165e-03, 4.63902393e-03, 3.85873985e-03,
       2.76062667e-03, 1.00782454e-03, 2.47969268e-04, 6.04239818e-03,
       7.64997072e-04, 1.57100537e-02, 2.29716203e-03, 2.58372272e-03,
       8.07637426e-04, 9.18548690e-04, 1.35656446e-03, 5.81458569e-04,
       4.96716928e-04, 3.79552257e-04, 5.31712788e-04, 2.71823509e-03,
       8.34294376e-04, 3.12136841e-04, 2.14075157e-04, 7.42422919e-02,
       3.80158492e-03, 5.67641024e-03, 2.87154703e-03, 9.83349904e-03,
       2.65470837e-04, 1.57946459e-03, 3.10058108e-04, 0.00000000e+00,
       0.00000000e+00, 2.27421721e-03, 1.05632062e-03, 5.42819222e-03,
       3.48484864e-02, 0.00000000e+00, 0.00000000e+00, 0.00000000e+00,
       0.00000000e+00, 1.90858845e-05, 9.09490682e-06, 1.31265147e-04,
       5.29163902e-06, 1.11952381e-04, 4.78452431e-06, 3.43582863e-04,
       5.57068428e-06, 1.07167376e-03, 3.99179008e-03, 4.07753410e-03,
       1.05749617e-04, 2.76528927e-03, 2.59244312e-05, 3.51888176e-04,
       2.31519337e-03, 1.99211177e-03, 4.02034629e-03, 2.03778082e-04,
       1.13483313e-02, 9.02551628e-04, 1.58182497e-03, 4.63243398e-05,
       2.92071004e-04, 3.11923094e-05, 1.56873538e-04, 2.87205987e-05,
       3.80543083e-05, 2.55045807e-04, 1.66878572e-04, 2.10341792e-04,
       1.26024842e-04, 9.40663015e-05])
```

```
In [167... # 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))

    #Plotting 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 [168... plot_features(x_train.columns, ideal_model.feature_importances_)
```



```
In [ ]:
```