# int skills Chris C

August 27, 2018

## 1 int skills makeup work

Christopher Csiszar

#### 1.0.1 hyperparameter tuning

I haven't worked with linear regressors in a while, so I'll be practicing on those + a simple RF.

Note the linear regressors below have a built in cross validation function, however I will be returning to basics and tuning the models in a simple way.

```
In [77]: import pandas as pd
         import numpy as np
         import datetime
         from datetime import datetime
         import dateutil.parser
         from sklearn.model_selection import train_test_split

         from sklearn import datasets, linear_model
         from sklearn.metrics import mean_squared_error, r2_score

         from sklearn.linear_model import Ridge
         from sklearn.ensemble import RandomForestRegressor
```

```
In [35]: ! pwd
```

```
/Users/chrispaul/Desktop/classes/intskills/kechup
```

```
In [36]: train = pd.read_csv('Train.csv')
```

```
/Users/chrispaul/anaconda2/envs/nlp/lib/python3.6/site-packages/IPython/core/interactiveshell.p
  interactivity=interactivity, compiler=compiler, result=result)
```

```
In [37]: train = train[0:23900]
```

```
In [38]: train.tail()
```

```
Out[38]:          SalesID  SalePrice  under_20k  MachineID  ModelID  datasource  \
        23895  1222558.0    81000.0        0.0  1019939.0  14287.0       121.0
        23896  1222569.0    23000.0        0.0  1057622.0   3350.0       121.0
        23897  1222570.0    17000.0        1.0  1001012.0   3414.0       121.0
        23898  1222571.0    16000.0        1.0  1023061.0   1528.0       121.0
        23899  1222575.0    48000.0        0.0  1061091.0  28920.0       121.0

               auctioneerID  YearMade  MachineHoursCurrentMeter       saledate  \
        23895           3.0    2006.0                    3395.0  12/22/11 0:00
        23896           3.0    1000.0                       0.0  12/22/11 0:00
        23897           3.0    1000.0                   13507.0  12/22/11 0:00
        23898           3.0    1000.0                    5417.0  12/22/11 0:00
        23899           3.0    2005.0                       0.0  12/22/11 0:00

                  state ProductGroup   Enclosure
        23895  Virginia          TEX  EROPS w AC
        23896  Virginia           MG       EROPS
        23897  Virginia          TEX       EROPS
        23898  Virginia          TTT       OROPS
        23899  Virginia           WL  EROPS w AC
```

```
In [39]: train.describe(include='all')
```

```
Out[39]:             SalesID      SalePrice     under_20k     MachineID       ModelID  \
        count   2.390000e+04   23900.000000  23900.000000  2.390000e+04  23900.000000
        unique           NaN            NaN           NaN           NaN           NaN
        top              NaN            NaN           NaN           NaN           NaN
        freq             NaN            NaN           NaN           NaN           NaN
        mean    1.179777e+06   33888.044477      0.375941  8.563724e+05   8326.160795
        std     2.392394e+04   24591.558640      0.484375  3.120474e+05   7562.924092
        min     1.139246e+06    4750.000000      0.000000  3.230000e+02     28.000000
        25%     1.159416e+06   15000.000000      0.000000  7.718310e+05   3357.000000
        50%     1.178196e+06   26500.000000      0.000000  1.018198e+06   4666.000000
        75%     1.200834e+06   45625.000000      1.000000  1.043985e+06  13395.000000
        max     1.222575e+06  142000.000000      1.000000  1.069977e+06  37198.000000

                datasource  auctioneerID      YearMade  MachineHoursCurrentMeter  \
        count      23900.0       23900.0  23900.000000              23900.000000
        unique         NaN           NaN           NaN                       NaN
        top            NaN           NaN           NaN                       NaN
        freq           NaN           NaN           NaN                       NaN
        mean         121.0           3.0   1821.906444               4649.534142
        std            0.0           0.0    384.023768               6066.597357
        min          121.0           3.0   1000.000000                  0.000000
        25%          121.0           3.0   1996.000000                930.000000
        50%          121.0           3.0   2001.000000               2638.000000
        75%          121.0           3.0   2004.000000               6470.000000
        max          121.0           3.0   2010.000000             220893.000000
```

```
               saledate  state ProductGroup Enclosure
count             23900  23900        23900     23884
unique              823     52            6         3
top        1/29/09 0:00  Texas          TEX     OROPS
freq                200   2823         6579      9399
mean                NaN    NaN          NaN       NaN
std                 NaN    NaN          NaN       NaN
min                 NaN    NaN          NaN       NaN
25%                 NaN    NaN          NaN       NaN
50%                 NaN    NaN          NaN       NaN
75%                 NaN    NaN          NaN       NaN
max                 NaN    NaN          NaN       NaN
```

```
In [40]: train['datesale'] = train.apply(lambda row: dateutil.parser.parse(row['saledate']).yea
```

```
In [41]: train
```

```
Out[41]:         SalesID  SalePrice  under_20k  MachineID  ModelID  datasource  \
         0      1139246.0    66000.0        0.0   999089.0   3157.0       121.0
         1      1139248.0    57000.0        0.0   117657.0     77.0       121.0
         2      1139249.0    10000.0        1.0   434808.0   7009.0       121.0
         3      1139251.0    38500.0        0.0  1026470.0    332.0       121.0
         4      1139253.0    11000.0        1.0  1057373.0  17311.0       121.0
         5      1139255.0    26500.0        0.0  1001274.0   4605.0       121.0
         6      1139256.0    21000.0        0.0   772701.0   1937.0       121.0
         7      1139261.0    27000.0        0.0   902002.0   3539.0       121.0
         8      1139272.0    21500.0        0.0  1036251.0  36003.0       121.0
         9      1139275.0    65000.0        0.0  1016474.0   3883.0       121.0
         10     1139278.0    24000.0        0.0  1024998.0   4605.0       121.0
         11     1139282.0    22500.0        0.0   319906.0   5255.0       121.0
         12     1139283.0    36000.0        0.0  1052214.0   2232.0       121.0
         13     1139284.0    30500.0        0.0  1068082.0   3542.0       121.0
         14     1139290.0    28000.0        0.0  1058450.0   5162.0       121.0
         15     1139291.0    19000.0        1.0  1004810.0   4604.0       121.0
         16     1139292.0    13500.0        1.0  1026973.0   9510.0       121.0
         17     1139299.0     9500.0        1.0  1002713.0  21442.0       121.0
         18     1139301.0    12500.0        1.0   125790.0   7040.0       121.0
         19     1139304.0    11500.0        1.0  1011914.0   3177.0       121.0
         20     1139311.0    41000.0        0.0  1014135.0   8867.0       121.0
         21     1139333.0    34500.0        0.0   999192.0   3350.0       121.0
         22     1139344.0    26000.0        0.0  1044500.0   7040.0       121.0
         23     1139346.0    73000.0        0.0   821452.0     85.0       121.0
         24     1139348.0    33000.0        0.0   294562.0   3542.0       121.0
         25     1139351.0    12500.0        1.0   833838.0   7009.0       121.0
         26     1139354.0    15500.0        1.0   565440.0   7040.0       121.0
         27     1139356.0    53000.0        0.0  1004127.0  25458.0       121.0
         28     1139357.0    46000.0        0.0    44800.0  19167.0       121.0
```

```
29     1139358.0   89000.0        0.0  1018076.0   1333.0        121.0
...          ...       ...        ...        ...      ...          ...
23870  1222464.0   27000.0        0.0  1027624.0    328.0        121.0
23871  1222466.0   79000.0        0.0  1008813.0  16506.0        121.0
23872  1222468.0   85000.0        0.0   520588.0  23926.0        121.0
23873  1222471.0   27000.0        0.0  1050702.0  22155.0        121.0
23874  1222474.0    7000.0        1.0  1022467.0  18263.0        121.0
23875  1222505.0   70000.0        0.0   198296.0   1263.0        121.0
23876  1222507.0   36000.0        0.0   213148.0   3542.0        121.0
23877  1222509.0   12500.0        1.0  1063187.0   4107.0        121.0
23878  1222510.0   26000.0        0.0  1065790.0  23737.0        121.0
23879  1222511.0   20000.0        0.0  1046116.0  28587.0        121.0
23880  1222512.0   11500.0        1.0  1033783.0   5436.0        121.0
23881  1222514.0    8500.0        1.0  1032478.0   3170.0        121.0
23882  1222516.0   10000.0        1.0  1057872.0   3170.0        121.0
23883  1222531.0    5500.0        1.0  1039959.0  17592.0        121.0
23884  1222534.0   12000.0        1.0  1008217.0   1958.0        121.0
23885  1222537.0   18000.0        1.0  1025370.0   3883.0        121.0
23886  1222538.0   23500.0        0.0  1046410.0   3883.0        121.0
23887  1222540.0   24000.0        0.0  1054036.0   3893.0        121.0
23888  1222541.0   13000.0        1.0  1055823.0   3369.0        121.0
23889  1222542.0   63000.0        0.0   753302.0   3886.0        121.0
23890  1222543.0   50000.0        0.0  1049123.0   3886.0        121.0
23891  1222544.0   12000.0        1.0  1002257.0   3883.0        121.0
23892  1222551.0   29500.0        0.0   705473.0   3539.0        121.0
23893  1222552.0   21000.0        0.0  1040718.0  18110.0        121.0
23894  1222553.0   29250.0        0.0  1022899.0  22854.0        121.0
23895  1222558.0   81000.0        0.0  1019939.0  14287.0        121.0
23896  1222569.0   23000.0        0.0  1057622.0   3350.0        121.0
23897  1222570.0   17000.0        1.0  1001012.0   3414.0        121.0
23898  1222571.0   16000.0        1.0  1023061.0   1528.0        121.0
23899  1222575.0   48000.0        0.0  1061091.0  28920.0        121.0

       auctioneerID  YearMade  MachineHoursCurrentMeter        saledate  \
0               3.0    2004.0                      68.0  11/16/06 0:00
1               3.0    1996.0                    4640.0   3/26/04 0:00
2               3.0    2001.0                    2838.0   2/26/04 0:00
3               3.0    2001.0                    3486.0   5/19/11 0:00
4               3.0    2007.0                     722.0   7/23/09 0:00
5               3.0    2004.0                     508.0  12/18/08 0:00
6               3.0    1993.0                   11540.0   8/26/04 0:00
7               3.0    2001.0                    4883.0  11/17/05 0:00
8               3.0    2008.0                     302.0   8/27/09 0:00
9               3.0    1000.0                   20700.0    8/9/07 0:00
10              3.0    2004.0                    1414.0   8/21/08 0:00
11              3.0    1998.0                    2764.0   8/24/06 0:00
12              3.0    1998.0                       0.0  10/20/05 0:00
13              3.0    2001.0                    1921.0   1/26/06 0:00
```

| | | | | |
|---|---|---|---|---|
| 14 | 3.0 | 2004.0 | 320.0 | 1/3/06 0:00 |
| 15 | 3.0 | 1999.0 | 2450.0 | 11/16/06 0:00 |
| 16 | 3.0 | 1999.0 | 1972.0 | 6/14/07 0:00 |
| 17 | 3.0 | 2003.0 | 0.0 | 1/28/10 0:00 |
| 18 | 3.0 | 2001.0 | 994.0 | 3/9/06 0:00 |
| 19 | 3.0 | 1991.0 | 8005.0 | 11/17/05 0:00 |
| 20 | 3.0 | 2000.0 | 3259.0 | 5/18/06 0:00 |
| 21 | 3.0 | 1000.0 | 16328.0 | 10/19/06 0:00 |
| 22 | 3.0 | 2005.0 | 109.0 | 10/25/07 0:00 |
| 23 | 3.0 | 1996.0 | 17033.0 | 10/19/06 0:00 |
| 24 | 3.0 | 2001.0 | 1877.0 | 5/20/04 0:00 |
| 25 | 3.0 | 2003.0 | 1028.0 | 3/9/06 0:00 |
| 26 | 3.0 | 2003.0 | 356.0 | 3/9/06 0:00 |
| 27 | 3.0 | 2000.0 | 0.0 | 2/22/07 0:00 |
| 28 | 3.0 | 2004.0 | 904.0 | 8/9/07 0:00 |
| 29 | 3.0 | 1998.0 | 10466.0 | 6/1/06 0:00 |
| ... | ... | ... | ... | ... |
| 23870 | 3.0 | 1000.0 | 9139.0 | 12/14/11 0:00 |
| 23871 | 3.0 | 2008.0 | 5234.0 | 12/15/11 0:00 |
| 23872 | 3.0 | 2000.0 | 11657.0 | 12/22/11 0:00 |
| 23873 | 3.0 | 2005.0 | 0.0 | 12/22/11 0:00 |
| 23874 | 3.0 | 2005.0 | 1032.0 | 12/22/11 0:00 |
| 23875 | 3.0 | 2001.0 | 11589.0 | 12/22/11 0:00 |
| 23876 | 3.0 | 2005.0 | 8563.0 | 12/22/11 0:00 |
| 23877 | 3.0 | 1000.0 | 9229.0 | 12/22/11 0:00 |
| 23878 | 3.0 | 1000.0 | 15060.0 | 12/22/11 0:00 |
| 23879 | 3.0 | 1000.0 | 17674.0 | 12/22/11 0:00 |
| 23880 | 3.0 | 1997.0 | 1533.0 | 12/22/11 0:00 |
| 23881 | 3.0 | 1000.0 | 881.0 | 12/22/11 0:00 |
| 23882 | 3.0 | 1000.0 | 2030.0 | 12/22/11 0:00 |
| 23883 | 3.0 | 1999.0 | 3333.0 | 12/22/11 0:00 |
| 23884 | 3.0 | 1000.0 | 5994.0 | 12/22/11 0:00 |
| 23885 | 3.0 | 1000.0 | 29598.0 | 12/22/11 0:00 |
| 23886 | 3.0 | 1000.0 | 10178.0 | 12/22/11 0:00 |
| 23887 | 3.0 | 1000.0 | 4607.0 | 12/22/11 0:00 |
| 23888 | 3.0 | 1000.0 | 0.0 | 12/22/11 0:00 |
| 23889 | 3.0 | 1997.0 | 23386.0 | 12/22/11 0:00 |
| 23890 | 3.0 | 1000.0 | 34057.0 | 12/22/11 0:00 |
| 23891 | 3.0 | 1000.0 | 24074.0 | 12/22/11 0:00 |
| 23892 | 3.0 | 2002.0 | 3138.0 | 12/21/11 0:00 |
| 23893 | 3.0 | 2007.0 | 1290.0 | 12/16/11 0:00 |
| 23894 | 3.0 | 2004.0 | 2299.0 | 12/16/11 0:00 |
| 23895 | 3.0 | 2006.0 | 3395.0 | 12/22/11 0:00 |
| 23896 | 3.0 | 1000.0 | 0.0 | 12/22/11 0:00 |
| 23897 | 3.0 | 1000.0 | 13507.0 | 12/22/11 0:00 |
| 23898 | 3.0 | 1000.0 | 5417.0 | 12/22/11 0:00 |
| 23899 | 3.0 | 2005.0 | 0.0 | 12/22/11 0:00 |

| | state | ProductGroup | Enclosure | datesale |
|---|---|---|---|---|
| 0 | Alabama | WL | EROPS w AC | 2006 |
| 1 | North Carolina | WL | EROPS w AC | 2004 |
| 2 | New York | SSL | OROPS | 2004 |
| 3 | Texas | TEX | EROPS w AC | 2011 |
| 4 | New York | SSL | EROPS | 2009 |
| 5 | Arizona | BL | OROPS | 2008 |
| 6 | Florida | TEX | EROPS | 2004 |
| 7 | Illinois | BL | OROPS | 2005 |
| 8 | Texas | TEX | EROPS | 2009 |
| 9 | Florida | WL | EROPS w AC | 2007 |
| 10 | Oregon | BL | OROPS | 2008 |
| 11 | Ohio | TTT | EROPS w AC | 2006 |
| 12 | Ohio | TEX | EROPS | 2005 |
| 13 | Texas | BL | OROPS | 2006 |
| 14 | North Carolina | BL | OROPS | 2006 |
| 15 | Arkansas | BL | OROPS | 2006 |
| 16 | Florida | TEX | EROPS | 2007 |
| 17 | Wisconsin | TEX | EROPS | 2010 |
| 18 | North Carolina | TEX | EROPS | 2006 |
| 19 | Illinois | BL | EROPS | 2005 |
| 20 | Kansas | TEX | EROPS w AC | 2006 |
| 21 | Nevada | MG | EROPS | 2006 |
| 22 | Iowa | TEX | EROPS w AC | 2007 |
| 23 | Maine | WL | EROPS w AC | 2006 |
| 24 | Texas | BL | OROPS | 2004 |
| 25 | Massachusetts | SSL | EROPS | 2006 |
| 26 | California | TEX | EROPS | 2006 |
| 27 | Texas | TEX | EROPS w AC | 2007 |
| 28 | Louisiana | MG | OROPS | 2007 |
| 29 | Minnesota | TEX | EROPS w AC | 2006 |
| ... | ... | ... | ... | ... |
| 23870 | Washington | TEX | EROPS | 2011 |
| 23871 | Arkansas | TEX | EROPS w AC | 2011 |
| 23872 | Florida | MG | EROPS w AC | 2011 |
| 23873 | Florida | TTT | OROPS | 2011 |
| 23874 | Maryland | SSL | OROPS | 2011 |
| 23875 | New Mexico | TEX | EROPS w AC | 2011 |
| 23876 | New Mexico | BL | EROPS w AC | 2011 |
| 23877 | Ohio | TTT | OROPS | 2011 |
| 23878 | Ohio | TEX | EROPS | 2011 |
| 23879 | Ohio | TEX | EROPS | 2011 |
| 23880 | Ohio | TEX | EROPS | 2011 |
| 23881 | Ohio | BL | EROPS | 2011 |
| 23882 | Ohio | BL | EROPS | 2011 |
| 23883 | North Carolina | SSL | OROPS | 2011 |
| 23884 | California | TEX | EROPS | 2011 |
| 23885 | California | WL | EROPS | 2011 |

```
23886      California          WL       EROPS      2011
23887      California          WL       EROPS      2011
23888      California          MG       EROPS      2011
23889      California          WL  EROPS w AC      2011
23890      California          WL  EROPS w AC      2011
23891      California          WL  EROPS w AC      2011
23892      California          BL  EROPS w AC      2011
23893         Florida          BL       OROPS      2011
23894         Florida          WL       OROPS      2011
23895        Virginia         TEX  EROPS w AC      2011
23896        Virginia          MG       EROPS      2011
23897        Virginia         TEX       EROPS      2011
23898        Virginia         TTT       OROPS      2011
23899        Virginia          WL  EROPS w AC      2011

[23900 rows x 14 columns]
```

```
In [42]: train['yearsold'] = train.datesale - train.YearMade
```

```
In [43]: train.head()
```

```
Out[43]:      SalesID  SalePrice  under_20k  MachineID  ModelID  datasource  \
         0  1139246.0    66000.0        0.0   999089.0   3157.0       121.0
         1  1139248.0    57000.0        0.0   117657.0     77.0       121.0
         2  1139249.0    10000.0        1.0   434808.0   7009.0       121.0
         3  1139251.0    38500.0        0.0  1026470.0    332.0       121.0
         4  1139253.0    11000.0        1.0  1057373.0  17311.0       121.0

            auctioneerID  YearMade  MachineHoursCurrentMeter       saledate  \
         0           3.0    2004.0                      68.0  11/16/06 0:00
         1           3.0    1996.0                    4640.0   3/26/04 0:00
         2           3.0    2001.0                    2838.0   2/26/04 0:00
         3           3.0    2001.0                    3486.0   5/19/11 0:00
         4           3.0    2007.0                     722.0   7/23/09 0:00

                     state ProductGroup   Enclosure  datesale  yearsold
         0         Alabama           WL  EROPS w AC      2006       2.0
         1  North Carolina           WL  EROPS w AC      2004       8.0
         2        New York          SSL       OROPS      2004       3.0
         3           Texas          TEX  EROPS w AC      2011      10.0
         4        New York          SSL       EROPS      2009       2.0
```

```
In [44]: train2 = train.drop(['saledate', "auctioneerID"], axis=1)
```

```
In [45]: train2.head()
```

```
Out[45]:      SalesID  SalePrice  under_20k  MachineID  ModelID  datasource  YearMade  \
         0  1139246.0    66000.0        0.0   999089.0   3157.0       121.0    2004.0
         1  1139248.0    57000.0        0.0   117657.0     77.0       121.0    1996.0
```

```
2  1139249.0    10000.0        1.0   434808.0   7009.0        121.0    2001.0
3  1139251.0    38500.0        0.0  1026470.0    332.0        121.0    2001.0
4  1139253.0    11000.0        1.0  1057373.0  17311.0        121.0    2007.0

   MachineHoursCurrentMeter           state ProductGroup   Enclosure  \
0                      68.0         Alabama           WL  EROPS w AC
1                    4640.0  North Carolina           WL  EROPS w AC
2                    2838.0        New York          SSL       OROPS
3                    3486.0           Texas          TEX  EROPS w AC
4                     722.0        New York          SSL       EROPS

   datesale  yearsold
0      2006       2.0
1      2004       8.0
2      2004       3.0
3      2011      10.0
4      2009       2.0
```

```
In [46]: def proc_col(col, train_col=None):
             """Encodes a pandas column with continous ids.
             """
             if train_col is not None:
                 uniq = train_col.unique()
             else:
                 uniq = col.unique()
             name2idx = {o:i for i,o in enumerate(uniq)}
             return name2idx, np.array([name2idx.get(x, -1) for x in col]), len(uniq)
```

```
In [47]: def encode_data(df, train=None):
             """ Encodes rating data with continous user and movie ids.
             If train is provided, encodes df with the same encoding as train.
             """
             df = df.copy()
             for col_name in ["state", "ProductGroup", "Enclosure"]:
                 train_col = None
                 if train is not None:
                     train_col = train[col_name]
                 _,col,_ = proc_col(df[col_name], train_col)
                 df[col_name] = col
                 df = df[df[col_name] >= 0]
             return df
```

```
In [48]: data = encode_data(train2)
```

```
In [49]: data.head()
```

```
Out[49]:      SalesID  SalePrice  under_20k  MachineID  ModelID  datasource  YearMade  \
         0  1139246.0    66000.0        0.0   999089.0   3157.0       121.0    2004.0
         1  1139248.0    57000.0        0.0   117657.0     77.0       121.0    1996.0
```

```
2  1139249.0    10000.0         1.0   434808.0   7009.0        121.0     2001.0
3  1139251.0    38500.0         0.0  1026470.0    332.0        121.0     2001.0
4  1139253.0    11000.0         1.0  1057373.0  17311.0        121.0     2007.0

   MachineHoursCurrentMeter  state  ProductGroup  Enclosure  datesale  \
0                      68.0      0             0          0      2006
1                    4640.0      1             0          0      2004
2                    2838.0      2             1          1      2004
3                    3486.0      3             2          0      2011
4                     722.0      2             1          2      2009

   yearsold
0       2.0
1       8.0
2       3.0
3      10.0
4       2.0
```

In [50]:
```python
data.describe(include='all')
```

Out[50]:
```
            SalesID      SalePrice     under_20k     MachineID       ModelID  \
count  2.390000e+04   23900.000000  23900.000000  2.390000e+04  23900.000000
mean   1.179777e+06   33888.044477      0.375941  8.563724e+05   8326.160795
std    2.392394e+04   24591.558640      0.484375  3.120474e+05   7562.924092
min    1.139246e+06    4750.000000      0.000000  3.230000e+02     28.000000
25%    1.159416e+06   15000.000000      0.000000  7.718310e+05   3357.000000
50%    1.178196e+06   26500.000000      0.000000  1.018198e+06   4666.000000
75%    1.200834e+06   45625.000000      1.000000  1.043985e+06  13395.000000
max    1.222575e+06  142000.000000      1.000000  1.069977e+06  37198.000000

       datasource      YearMade  MachineHoursCurrentMeter         state  \
count     23900.0  23900.000000              23900.000000  23900.000000
mean        121.0   1821.906444               4649.534142     16.290084
std           0.0    384.023768               6066.597357     13.107064
min         121.0   1000.000000                  0.000000      0.000000
25%         121.0   1996.000000                930.000000      5.000000
50%         121.0   2001.000000               2638.000000     16.000000
75%         121.0   2004.000000               6470.000000     26.000000
max         121.0   2010.000000             220893.000000     51.000000

       ProductGroup     Enclosure      datesale      yearsold
count  23900.000000  23900.000000  23900.000000  23900.000000
mean       2.184100      0.858536   2008.424477    186.518033
std        1.487749      0.767305      2.121588    384.468595
min        0.000000      0.000000   2004.000000      0.000000
25%        1.000000      0.000000   2007.000000      4.000000
50%        2.000000      1.000000   2009.000000      7.000000
75%        3.000000      1.000000   2010.000000     13.000000
max        5.000000      3.000000   2011.000000   1011.000000
```

## 2 Regression

```
In [56]: y = data.SalePrice
```

```
In [57]: x = data.drop(['SalePrice', "under_20k"], axis=1)
```

```
In [58]: X_train, X_test, y_train, y_test = train_test_split(x, y, test_size=0.2, random_state=
```

```
In [59]: len(X_train)
```

```
Out[59]: 19120
```

```
In [60]: x
```

```
Out[60]:          SalesID  MachineID  ModelID  datasource  YearMade  \
         0      1139246.0   999089.0   3157.0       121.0    2004.0
         1      1139248.0   117657.0     77.0       121.0    1996.0
         2      1139249.0   434808.0   7009.0       121.0    2001.0
         3      1139251.0  1026470.0    332.0       121.0    2001.0
         4      1139253.0  1057373.0  17311.0       121.0    2007.0
         5      1139255.0  1001274.0   4605.0       121.0    2004.0
         6      1139256.0   772701.0   1937.0       121.0    1993.0
         7      1139261.0   902002.0   3539.0       121.0    2001.0
         8      1139272.0  1036251.0  36003.0       121.0    2008.0
         9      1139275.0  1016474.0   3883.0       121.0    1000.0
         10     1139278.0  1024998.0   4605.0       121.0    2004.0
         11     1139282.0   319906.0   5255.0       121.0    1998.0
         12     1139283.0  1052214.0   2232.0       121.0    1998.0
         13     1139284.0  1068082.0   3542.0       121.0    2001.0
         14     1139290.0  1058450.0   5162.0       121.0    2004.0
         15     1139291.0  1004810.0   4604.0       121.0    1999.0
         16     1139292.0  1026973.0   9510.0       121.0    1999.0
         17     1139299.0  1002713.0  21442.0       121.0    2003.0
         18     1139301.0   125790.0   7040.0       121.0    2001.0
         19     1139304.0  1011914.0   3177.0       121.0    1991.0
         20     1139311.0  1014135.0   8867.0       121.0    2000.0
         21     1139333.0   999192.0   3350.0       121.0    1000.0
         22     1139344.0  1044500.0   7040.0       121.0    2005.0
         23     1139346.0   821452.0     85.0       121.0    1996.0
         24     1139348.0   294562.0   3542.0       121.0    2001.0
         25     1139351.0   833838.0   7009.0       121.0    2003.0
         26     1139354.0   565440.0   7040.0       121.0    2003.0
         27     1139356.0  1004127.0  25458.0       121.0    2000.0
         28     1139357.0    44800.0  19167.0       121.0    2004.0
         29     1139358.0  1018076.0   1333.0       121.0    1998.0
         ...          ...        ...      ...         ...       ...
         23870  1222464.0  1027624.0    328.0       121.0    1000.0
         23871  1222466.0  1008813.0  16506.0       121.0    2008.0
         23872  1222468.0   520588.0  23926.0       121.0    2000.0
```

```
23873  1222471.0  1050702.0  22155.0       121.0    2005.0
23874  1222474.0  1022467.0  18263.0       121.0    2005.0
23875  1222505.0   198296.0   1263.0       121.0    2001.0
23876  1222507.0   213148.0   3542.0       121.0    2005.0
23877  1222509.0  1063187.0   4107.0       121.0    1000.0
23878  1222510.0  1065790.0  23737.0       121.0    1000.0
23879  1222511.0  1046116.0  28587.0       121.0    1000.0
23880  1222512.0  1033783.0   5436.0       121.0    1997.0
23881  1222514.0  1032478.0   3170.0       121.0    1000.0
23882  1222516.0  1057872.0   3170.0       121.0    1000.0
23883  1222531.0  1039959.0  17592.0       121.0    1999.0
23884  1222534.0  1008217.0   1958.0       121.0    1000.0
23885  1222537.0  1025370.0   3883.0       121.0    1000.0
23886  1222538.0  1046410.0   3883.0       121.0    1000.0
23887  1222540.0  1054036.0   3893.0       121.0    1000.0
23888  1222541.0  1055823.0   3369.0       121.0    1000.0
23889  1222542.0   753302.0   3886.0       121.0    1997.0
23890  1222543.0  1049123.0   3886.0       121.0    1000.0
23891  1222544.0  1002257.0   3883.0       121.0    1000.0
23892  1222551.0   705473.0   3539.0       121.0    2002.0
23893  1222552.0  1040718.0  18110.0       121.0    2007.0
23894  1222553.0  1022899.0  22854.0       121.0    2004.0
23895  1222558.0  1019939.0  14287.0       121.0    2006.0
23896  1222569.0  1057622.0   3350.0       121.0    1000.0
23897  1222570.0  1001012.0   3414.0       121.0    1000.0
23898  1222571.0  1023061.0   1528.0       121.0    1000.0
23899  1222575.0  1061091.0  28920.0       121.0    2005.0

       MachineHoursCurrentMeter  state  ProductGroup  Enclosure  datesale  \
0                          68.0      0             0          0      2006
1                        4640.0      1             0          0      2004
2                        2838.0      2             1          1      2004
3                        3486.0      3             2          0      2011
4                         722.0      2             1          2      2009
5                         508.0      4             3          1      2008
6                       11540.0      5             2          2      2004
7                        4883.0      6             3          1      2005
8                         302.0      3             2          2      2009
9                       20700.0      5             0          0      2007
10                       1414.0      7             3          1      2008
11                       2764.0      8             4          0      2006
12                          0.0      8             2          2      2005
13                       1921.0      3             3          1      2006
14                        320.0      1             3          1      2006
15                       2450.0      9             3          1      2006
16                       1972.0      5             2          2      2007
17                          0.0     10             2          2      2010
18                        994.0      1             2          2      2006
```

| | | | | | |
|---|---|---|---|---|---|
| 19 | 8005.0 | 6 | 3 | 2 | 2005 |
| 20 | 3259.0 | 11 | 2 | 0 | 2006 |
| 21 | 16328.0 | 12 | 5 | 2 | 2006 |
| 22 | 109.0 | 13 | 2 | 0 | 2007 |
| 23 | 17033.0 | 14 | 0 | 0 | 2006 |
| 24 | 1877.0 | 3 | 3 | 1 | 2004 |
| 25 | 1028.0 | 15 | 1 | 2 | 2006 |
| 26 | 356.0 | 16 | 2 | 2 | 2006 |
| 27 | 0.0 | 3 | 2 | 0 | 2007 |
| 28 | 904.0 | 17 | 5 | 1 | 2007 |
| 29 | 10466.0 | 18 | 2 | 0 | 2006 |
| ... | ... | ... | ... | ... | ... |
| 23870 | 9139.0 | 28 | 2 | 2 | 2011 |
| 23871 | 5234.0 | 9 | 2 | 0 | 2011 |
| 23872 | 11657.0 | 5 | 5 | 0 | 2011 |
| 23873 | 0.0 | 5 | 4 | 1 | 2011 |
| 23874 | 1032.0 | 41 | 1 | 1 | 2011 |
| 23875 | 11589.0 | 44 | 2 | 0 | 2011 |
| 23876 | 8563.0 | 44 | 3 | 0 | 2011 |
| 23877 | 9229.0 | 8 | 4 | 1 | 2011 |
| 23878 | 15060.0 | 8 | 2 | 2 | 2011 |
| 23879 | 17674.0 | 8 | 2 | 2 | 2011 |
| 23880 | 1533.0 | 8 | 2 | 2 | 2011 |
| 23881 | 881.0 | 8 | 3 | 2 | 2011 |
| 23882 | 2030.0 | 8 | 3 | 2 | 2011 |
| 23883 | 3333.0 | 1 | 1 | 1 | 2011 |
| 23884 | 5994.0 | 16 | 2 | 2 | 2011 |
| 23885 | 29598.0 | 16 | 0 | 2 | 2011 |
| 23886 | 10178.0 | 16 | 0 | 2 | 2011 |
| 23887 | 4607.0 | 16 | 0 | 2 | 2011 |
| 23888 | 0.0 | 16 | 5 | 2 | 2011 |
| 23889 | 23386.0 | 16 | 0 | 0 | 2011 |
| 23890 | 34057.0 | 16 | 0 | 0 | 2011 |
| 23891 | 24074.0 | 16 | 0 | 0 | 2011 |
| 23892 | 3138.0 | 16 | 3 | 0 | 2011 |
| 23893 | 1290.0 | 5 | 3 | 1 | 2011 |
| 23894 | 2299.0 | 5 | 0 | 1 | 2011 |
| 23895 | 3395.0 | 29 | 2 | 0 | 2011 |
| 23896 | 0.0 | 29 | 5 | 2 | 2011 |
| 23897 | 13507.0 | 29 | 2 | 2 | 2011 |
| 23898 | 5417.0 | 29 | 4 | 1 | 2011 |
| 23899 | 0.0 | 29 | 0 | 0 | 2011 |

| | yearsold |
|---|---|
| 0 | 2.0 |
| 1 | 8.0 |
| 2 | 3.0 |
| 3 | 10.0 |

```
4           2.0
5           4.0
6          11.0
7           4.0
8           1.0
9        1007.0
10          4.0
11          8.0
12          7.0
13          5.0
14          2.0
15          7.0
16          8.0
17          7.0
18          5.0
19         14.0
20          6.0
21       1006.0
22          2.0
23         10.0
24          3.0
25          3.0
26          3.0
27          7.0
28          3.0
29          8.0
...         ...
23870    1011.0
23871       3.0
23872      11.0
23873       6.0
23874       6.0
23875      10.0
23876       6.0
23877    1011.0
23878    1011.0
23879    1011.0
23880      14.0
23881    1011.0
23882    1011.0
23883      12.0
23884    1011.0
23885    1011.0
23886    1011.0
23887    1011.0
23888    1011.0
23889      14.0
23890    1011.0
```

```
23891    1011.0
23892       9.0
23893       4.0
23894       7.0
23895       5.0
23896    1011.0
23897    1011.0
23898    1011.0
23899       6.0

[23900 rows x 11 columns]
```

## 2.1 linear

```
In [63]: regr = linear_model.LinearRegression()
```

```
In [64]: # Train the model using the training sets
         regr.fit(X_train, y_train)

         # Make predictions using the testing set
         y_pred = regr.predict(X_test)

         # The coefficients
         print('Coefficients: \n', regr.coef_)
         # The mean squared error
         print("Mean squared error: %.2f"
               % mean_squared_error(y_test, y_pred))
         # Explained variance score: 1 is perfect prediction
         print('Variance score: %.2f' % r2_score(y_test, y_pred))
```

```
Coefficients: 
 [ 1.95134325e-03 -1.08772671e-02  1.28392054e-01  9.16600129e-13
 -1.61230572e+02  8.00475075e-01 -6.07592080e+01  3.27000956e+03
 -1.36857816e+04 -3.25624571e+02 -1.64394000e+02]
Mean squared error: 418257601.00
Variance score: 0.30
```

```
In [65]: y_pred
```

```
Out[65]: array([15308.76228324, 20361.80805096, 21858.59306422, ...,
                33147.60545152, 53148.95056587, 32171.53320709])
```

## 2.2 Ridge

```
In [71]: for a in [0, 0.2, 0.4, 0.6, 0.8, 1]:

             regr = Ridge(alpha=a)
```

```
        # Train the model using the training sets
        regr.fit(X_train, y_train)

        # Make predictions using the testing set
        y_pred = regr.predict(X_test)

        print('alpha = ', str(a))
        # The coefficients
        print('Coefficients: \n', regr.coef_)
        # The mean squared error
        print("Mean squared error: %.2f"
              % mean_squared_error(y_test, y_pred))
        # Explained variance score: 1 is perfect prediction
        print('Variance score: %.2f' % r2_score(y_test, y_pred))
        print('-------------------------')
```

```
alpha =  0
Coefficients:
 [ 6.93354496e-02 -8.75989383e-03 -1.01218031e+00 -4.81631818e+16
 -3.46264392e+14 -2.03285641e+00 -6.11630211e+01  3.26988809e+03
 -1.36891390e+04  3.46264392e+14 -3.46264392e+14]
Mean squared error: 763940040.24
Variance score: -0.28
-------------------------
alpha =  0.2
Coefficients:
 [ 1.95109156e-03 -1.08773243e-02  1.28392382e-01  0.00000000e+00
 -1.61225312e+02  8.00482577e-01 -6.07593820e+01  3.26999244e+03
 -1.36855062e+04 -3.25611526e+02 -1.64388872e+02]
Mean squared error: 418257809.51
Variance score: 0.30
-------------------------
alpha =  0.4
Coefficients:
 [ 1.95083988e-03 -1.08773815e-02  1.28392709e-01  0.00000000e+00
 -1.61218254e+02  8.00490078e-01 -6.07595560e+01  3.26997531e+03
 -1.36852309e+04 -3.25600280e+02 -1.64381944e+02]
Mean squared error: 418258018.10
Variance score: 0.30
-------------------------
alpha =  0.6
Coefficients:
 [ 1.95058822e-03 -1.08774387e-02  1.28393036e-01  0.00000000e+00
 -1.61212335e+02  8.00497578e-01 -6.07597300e+01  3.26995819e+03
 -1.36849556e+04 -3.25587895e+02 -1.64376157e+02]
Mean squared error: 418258226.76
Variance score: 0.30
-------------------------
```

```
alpha =  0.8
Coefficients:
 [ 1.95033657e-03 -1.08774959e-02  1.28393364e-01  0.00000000e+00
 -1.61206109e+02  8.00505079e-01 -6.07599039e+01  3.26994106e+03
 -1.36846803e+04 -3.25575819e+02 -1.64370063e+02]
Mean squared error: 418258435.50
Variance score: 0.30
-------------------------
alpha =  1
Coefficients:
 [ 1.95008493e-03 -1.08775531e-02  1.28393691e-01  0.00000000e+00
 -1.61199937e+02  8.00512579e-01 -6.07600779e+01  3.26992394e+03
 -1.36844050e+04 -3.25563689e+02 -1.64364022e+02]
Mean squared error: 418258644.31
Variance score: 0.30
-------------------------
```

```
/Users/chrispaul/anaconda2/envs/nlp/lib/python3.6/site-packages/sklearn/linear_model/ridge.py:1
Ill-conditioned matrix detected. Result is not guaranteed to be accurate.
Reciprocal condition number1.061562e-16
  overwrite_a=True).T
```

```
In [90]: for a in [0.18, 0.19, 0.2, 0.21, 0.22, 0.23]:

             regr = Ridge(alpha=a)

             # Train the model using the training sets
             regr.fit(X_train, y_train)

             # Make predictions using the testing set
             y_pred = regr.predict(X_test)

             print('alpha = ', str(a))
             # The coefficients
             print('Coefficients: \n', regr.coef_)
             # The mean squared error
             print("Mean squared error: %.2f"
                   % mean_squared_error(y_test, y_pred))
             # Explained variance score: 1 is perfect prediction
             print('Variance score: %.2f' % r2_score(y_test, y_pred))
             print('-------------------------')
```

```
alpha =  0.18
Coefficients:
 [ 1.95111673e-03 -1.08773186e-02  1.28392349e-01  0.00000000e+00
 -1.61225920e+02  8.00481827e-01 -6.07593646e+01  3.26999415e+03
```

```
 -1.36855337e+04 -3.25612749e+02 -1.64389466e+02]
Mean squared error: 418257788.66
Variance score: 0.30
-------------------------
alpha =  0.19
Coefficients:
 [ 1.95110414e-03 -1.08773214e-02  1.28392365e-01  0.00000000e+00
 -1.61225339e+02  8.00482202e-01 -6.07593733e+01  3.26999329e+03
 -1.36855200e+04 -3.25612414e+02 -1.64388892e+02]
Mean squared error: 418257799.08
Variance score: 0.30
-------------------------
alpha =  0.2
Coefficients:
 [ 1.95109156e-03 -1.08773243e-02  1.28392382e-01  0.00000000e+00
 -1.61225312e+02  8.00482577e-01 -6.07593820e+01  3.26999244e+03
 -1.36855062e+04 -3.25611526e+02 -1.64388872e+02]
Mean squared error: 418257809.51
Variance score: 0.30
-------------------------
alpha =  0.21
Coefficients:
 [ 1.95107898e-03 -1.08773271e-02  1.28392398e-01  0.00000000e+00
 -1.61224751e+02  8.00482952e-01 -6.07593907e+01  3.26999158e+03
 -1.36854924e+04 -3.25611172e+02 -1.64388317e+02]
Mean squared error: 418257819.94
Variance score: 0.30
-------------------------
alpha =  0.22
Coefficients:
 [ 1.95106639e-03 -1.08773300e-02  1.28392414e-01  0.00000000e+00
 -1.61224559e+02  8.00483327e-01 -6.07593994e+01  3.26999072e+03
 -1.36854787e+04 -3.25610448e+02 -1.64388132e+02]
Mean squared error: 418257830.37
Variance score: 0.30
-------------------------
alpha =  0.23
Coefficients:
 [ 1.95105381e-03 -1.08773329e-02  1.28392431e-01  0.00000000e+00
 -1.61223914e+02  8.00483702e-01 -6.07594081e+01  3.26998987e+03
 -1.36854649e+04 -3.25610179e+02 -1.64387493e+02]
Mean squared error: 418257840.79
Variance score: 0.30
-------------------------


/Users/chrispaul/anaconda2/envs/nlp/lib/python3.6/site-packages/sklearn/linear_model/ridge.py:
Ill-conditioned matrix detected. Result is not guaranteed to be accurate.
```

```
Reciprocal condition number9.554057e-17
  overwrite_a=True).T
/Users/chrispaul/anaconda2/envs/nlp/lib/python3.6/site-packages/sklearn/linear_model/ridge.py:
Ill-conditioned matrix detected. Result is not guaranteed to be accurate.
Reciprocal condition number1.008485e-16
  overwrite_a=True).T
/Users/chrispaul/anaconda2/envs/nlp/lib/python3.6/site-packages/sklearn/linear_model/ridge.py:
Ill-conditioned matrix detected. Result is not guaranteed to be accurate.
Reciprocal condition number1.061562e-16
  overwrite_a=True).T
```

```
In [92]: for a in [0.1, 0.11, 0.12, 0.13, 0.14, 0.15, 0.16, 0.7]:

             regr = Ridge(alpha=a)

             # Train the model using the training sets
             regr.fit(X_train, y_train)

             # Make predictions using the testing set
             y_pred = regr.predict(X_test)

             print('alpha = ', str(a))
             # The coefficients
             print('Coefficients: \n', regr.coef_)
             # The mean squared error
             print("Mean squared error: %.2f"
                   % mean_squared_error(y_test, y_pred))
             # Explained variance score: 1 is perfect prediction
             print('Variance score: %.2f' % r2_score(y_test, y_pred))
             print('-------------------------')
```

```
alpha =  0.1
Coefficients:
 [ 1.95121740e-03 -1.08772957e-02  1.28392218e-01  0.00000000e+00
 -1.61228856e+02  8.00478826e-01 -6.07592950e+01  3.27000100e+03
 -1.36856439e+04 -3.25617135e+02 -1.64392349e+02]
Mean squared error: 418257705.25
Variance score: 0.30
-------------------------
alpha =  0.11
Coefficients:
 [ 1.95120482e-03 -1.08772985e-02  1.28392234e-01  0.00000000e+00
 -1.61228053e+02  8.00479201e-01 -6.07593037e+01  3.27000014e+03
 -1.36856301e+04 -3.25617023e+02 -1.64391553e+02]
Mean squared error: 418257715.67
Variance score: 0.30
-------------------------
```

```
alpha =  0.12
Coefficients:
 [ 1.95119223e-03 -1.08773014e-02  1.28392251e-01  0.00000000e+00
 -1.61227695e+02  8.00479576e-01 -6.07593124e+01  3.26999929e+03
 -1.36856163e+04 -3.25616465e+02 -1.64391202e+02]
Mean squared error: 418257726.10
Variance score: 0.30
-------------------------
alpha =  0.13
Coefficients:
 [ 1.95117965e-03 -1.08773043e-02  1.28392267e-01  0.00000000e+00
 -1.61227165e+02  8.00479951e-01 -6.07593211e+01  3.26999843e+03
 -1.36856026e+04 -3.25616079e+02 -1.64390679e+02]
Mean squared error: 418257736.52
Variance score: 0.30
-------------------------
alpha =  0.14
Coefficients:
 [ 1.95116707e-03 -1.08773071e-02  1.28392283e-01  0.00000000e+00
 -1.61227241e+02  8.00480326e-01 -6.07593298e+01  3.26999757e+03
 -1.36855888e+04 -3.25615089e+02 -1.64390761e+02]
Mean squared error: 418257746.95
Variance score: 0.30
-------------------------
alpha =  0.15
Coefficients:
 [ 1.95115448e-03 -1.08773100e-02  1.28392300e-01  0.00000000e+00
 -1.61226739e+02  8.00480701e-01 -6.07593385e+01  3.26999672e+03
 -1.36855750e+04 -3.25614675e+02 -1.64390265e+02]
Mean squared error: 418257757.38
Variance score: 0.30
-------------------------
alpha =  0.16
Coefficients:
 [ 1.95114190e-03 -1.08773128e-02  1.28392316e-01  0.00000000e+00
 -1.61226448e+02  8.00481076e-01 -6.07593472e+01  3.26999586e+03
 -1.36855613e+04 -3.25614051e+02 -1.64389981e+02]
Mean squared error: 418257767.80
Variance score: 0.30
-------------------------
alpha =  0.7
Coefficients:
 [ 1.95046240e-03 -1.08774673e-02  1.28393200e-01  0.00000000e+00
 -1.61209216e+02  8.00501329e-01 -6.07598169e+01  3.26994963e+03
 -1.36848179e+04 -3.25581864e+02 -1.64373104e+02]
Mean squared error: 418258331.12
Variance score: 0.30
-------------------------
```

```
/Users/chrispaul/anaconda2/envs/nlp/lib/python3.6/site-packages/sklearn/linear_model/ridge.py:
Ill-conditioned matrix detected. Result is not guaranteed to be accurate.
Reciprocal condition number5.307793e-17
  overwrite_a=True).T
/Users/chrispaul/anaconda2/envs/nlp/lib/python3.6/site-packages/sklearn/linear_model/ridge.py:
Ill-conditioned matrix detected. Result is not guaranteed to be accurate.
Reciprocal condition number5.838591e-17
  overwrite_a=True).T
/Users/chrispaul/anaconda2/envs/nlp/lib/python3.6/site-packages/sklearn/linear_model/ridge.py:
Ill-conditioned matrix detected. Result is not guaranteed to be accurate.
Reciprocal condition number6.369374e-17
  overwrite_a=True).T
/Users/chrispaul/anaconda2/envs/nlp/lib/python3.6/site-packages/sklearn/linear_model/ridge.py:
Ill-conditioned matrix detected. Result is not guaranteed to be accurate.
Reciprocal condition number6.900165e-17
  overwrite_a=True).T
/Users/chrispaul/anaconda2/envs/nlp/lib/python3.6/site-packages/sklearn/linear_model/ridge.py:
Ill-conditioned matrix detected. Result is not guaranteed to be accurate.
Reciprocal condition number7.430929e-17
  overwrite_a=True).T
/Users/chrispaul/anaconda2/envs/nlp/lib/python3.6/site-packages/sklearn/linear_model/ridge.py:
Ill-conditioned matrix detected. Result is not guaranteed to be accurate.
Reciprocal condition number7.961719e-17
  overwrite_a=True).T
/Users/chrispaul/anaconda2/envs/nlp/lib/python3.6/site-packages/sklearn/linear_model/ridge.py:
Ill-conditioned matrix detected. Result is not guaranteed to be accurate.
Reciprocal condition number8.492500e-17
  overwrite_a=True).T
```

### 2.3 Lasso

```
In [72]: for a in [0, 0.2, 0.4, 0.6, 0.8, 1]:

             regr = linear_model.Lasso(alpha=a)

             # Train the model using the training sets
             regr.fit(X_train, y_train)

             # Make predictions using the testing set
             y_pred = regr.predict(X_test)

             print('alpha = ', str(a))
             # The coefficients
             print('Coefficients: \n', regr.coef_)
             # The mean squared error
```

```
    print("Mean squared error: %.2f"
          % mean_squared_error(y_test, y_pred))
    # Explained variance score: 1 is perfect prediction
    print('Variance score: %.2f' % r2_score(y_test, y_pred))
    print('-------------------------')
```

```
/Users/chrispaul/anaconda2/envs/nlp/lib/python3.6/site-packages/ipykernel_launcher.py:6: UserWa

/Users/chrispaul/anaconda2/envs/nlp/lib/python3.6/site-packages/sklearn/linear_model/coordinate
  positive)
/Users/chrispaul/anaconda2/envs/nlp/lib/python3.6/site-packages/sklearn/linear_model/coordinate
  ConvergenceWarning)


alpha =  0
Coefficients:
 [ 1.95134325e-03 -1.08772671e-02  1.28392054e-01  0.00000000e+00
  5.27928021e+00  8.00475075e-01 -6.07592080e+01  3.27000956e+03
 -1.36857816e+04 -4.92134423e+02  2.11585219e+00]
Mean squared error: 418257601.00
Variance score: 0.30
-------------------------
alpha =  0.2
Coefficients:
 [ 1.94732410e-03 -1.08774212e-02  1.28389354e-01  0.00000000e+00
  5.27514744e+00  8.00482359e-01 -6.07582856e+01  3.26991438e+03
 -1.36853732e+04 -4.92030133e+02  2.11148863e+00]
Mean squared error: 418257744.51
Variance score: 0.30
-------------------------
alpha =  0.4
Coefficients:
 [ 1.94330495e-03 -1.08775753e-02  1.28386655e-01  0.00000000e+00
  5.27101468e+00  8.00489643e-01 -6.07573633e+01  3.26981920e+03
 -1.36849648e+04 -4.91925842e+02  2.10712506e+00]
Mean squared error: 418257888.28
Variance score: 0.30
-------------------------
alpha =  0.6
Coefficients:
 [ 1.93928581e-03 -1.08777293e-02  1.28383955e-01  0.00000000e+00
  5.26688192e+00  8.00496926e-01 -6.07564409e+01  3.26972403e+03
 -1.36845564e+04 -4.91821552e+02  2.10276150e+00]
Mean squared error: 418258032.28
Variance score: 0.30
-------------------------
alpha =  0.8
Coefficients:
```

```
 [ 1.93526666e-03 -1.08778834e-02  1.28381255e-01  0.00000000e+00
  5.26274916e+00  8.00504210e-01 -6.07555186e+01  3.26962885e+03
 -1.36841480e+04 -4.91717262e+02  2.09839794e+00]
Mean squared error: 418258176.54
Variance score: 0.30
-------------------------
alpha =  1
Coefficients:
 [ 1.93124751e-03 -1.08780375e-02  1.28378555e-01  0.00000000e+00
  5.25861640e+00  8.00511494e-01 -6.07545962e+01  3.26953367e+03
 -1.36837396e+04 -4.91612971e+02  2.09403437e+00]
Mean squared error: 418258321.05
Variance score: 0.30
-------------------------
```

### 2.4 Elastic Net

```
In [73]: for a in [0, 0.2, 0.4, 0.6, 0.8, 1]:

             regr = linear_model.ElasticNet(alpha=a)

             # Train the model using the training sets
             regr.fit(X_train, y_train)

             # Make predictions using the testing set
             y_pred = regr.predict(X_test)

             print('alpha = ', str(a))
             # The coefficients
             print('Coefficients: \n', regr.coef_)
             # The mean squared error
             print("Mean squared error: %.2f"
                   % mean_squared_error(y_test, y_pred))
             # Explained variance score: 1 is perfect prediction
             print('Variance score: %.2f' % r2_score(y_test, y_pred))
             print('-------------------------')
```

```
/Users/chrispaul/anaconda2/envs/nlp/lib/python3.6/site-packages/ipykernel_launcher.py:6: UserWa

/Users/chrispaul/anaconda2/envs/nlp/lib/python3.6/site-packages/sklearn/linear_model/coordinate
  positive)
/Users/chrispaul/anaconda2/envs/nlp/lib/python3.6/site-packages/sklearn/linear_model/coordinate
  ConvergenceWarning)


alpha =  0
Coefficients:
```

```
 [ 1.95134325e-03 -1.08772671e-02  1.28392054e-01  0.00000000e+00
  5.27928021e+00  8.00475075e-01 -6.07592080e+01  3.27000956e+03
 -1.36857816e+04 -4.92134423e+02  2.11585219e+00]
Mean squared error: 418257601.00
Variance score: 0.30
-------------------------
alpha =  0.2
Coefficients:
 [-1.81447077e-04 -1.13441265e-02  1.30691043e-01  0.00000000e+00
  4.96660340e+00  8.60001910e-01 -6.21619769e+01  3.11522203e+03
 -1.14770290e+04 -3.42362096e+02  7.54373158e-01]
Mean squared error: 422599616.41
Variance score: 0.29
-------------------------
alpha =  0.4
Coefficients:
 [-1.47706023e-03 -1.16885398e-02  1.32082847e-01  0.00000000e+00
  4.82008485e+00  9.02091622e-01 -6.31952279e+01  2.97598486e+03
 -9.88272856e+03 -2.38534440e+02 -1.35216045e-01]
Mean squared error: 429090896.23
Variance score: 0.28
-------------------------
alpha =  0.6
Coefficients:
 [-2.26474672e-03 -1.19545218e-02  1.32930982e-01  0.00000000e+00
  4.77287391e+00  9.33229136e-01 -6.39909455e+01  2.84957205e+03
 -8.67774452e+03 -1.63433171e+02 -7.33239085e-01]
Mean squared error: 435900056.01
Variance score: 0.27
-------------------------
alpha =  0.8
Coefficients:
 [-2.72767242e-03 -1.21670491e-02  1.33436124e-01  0.00000000e+00
  4.78413259e+00  9.57062363e-01 -6.46241212e+01  2.73402719e+03
 -7.73496058e+03 -1.07378280e+02 -1.14440228e+00]
Mean squared error: 442398051.36
Variance score: 0.26
-------------------------
alpha =  1
Coefficients:
 [-2.97504431e-03 -1.23413818e-02  1.33716788e-01  0.00000000e+00
  4.83343402e+00  9.75796959e-01 -6.51406469e+01  2.62785703e+03
 -6.97717129e+03 -6.45283502e+01 -1.42778011e+00]
Mean squared error: 448388144.43
Variance score: 0.25
-------------------------
```

```
In [74]: for a in [0, 0.2, 0.4, 0.6, 0.8, 1]:

             regr = linear_model.ElasticNet(alpha=0, l1_ratio=a)

             # Train the model using the training sets
             regr.fit(X_train, y_train)

             # Make predictions using the testing set
             y_pred = regr.predict(X_test)

             print('alpha = ', str(a))
             # The coefficients
             print('Coefficients: \n', regr.coef_)
             # The mean squared error
             print("Mean squared error: %.2f"
                   % mean_squared_error(y_test, y_pred))
             # Explained variance score: 1 is perfect prediction
             print('Variance score: %.2f' % r2_score(y_test, y_pred))
             print('-------------------------')
```

```
/Users/chrispaul/anaconda2/envs/nlp/lib/python3.6/site-packages/ipykernel_launcher.py:6: UserWa

/Users/chrispaul/anaconda2/envs/nlp/lib/python3.6/site-packages/sklearn/linear_model/coordinate
  positive)
/Users/chrispaul/anaconda2/envs/nlp/lib/python3.6/site-packages/sklearn/linear_model/coordinate
  ConvergenceWarning)


alpha =  0
Coefficients:
 [ 1.95134325e-03 -1.08772671e-02  1.28392054e-01  0.00000000e+00
  5.27928021e+00  8.00475075e-01 -6.07592080e+01  3.27000956e+03
 -1.36857816e+04 -4.92134423e+02  2.11585219e+00]
Mean squared error: 418257601.00
Variance score: 0.30
-------------------------
alpha =  0.2
Coefficients:
 [ 1.95134325e-03 -1.08772671e-02  1.28392054e-01  0.00000000e+00
  5.27928021e+00  8.00475075e-01 -6.07592080e+01  3.27000956e+03
 -1.36857816e+04 -4.92134423e+02  2.11585219e+00]
Mean squared error: 418257601.00
Variance score: 0.30
-------------------------
alpha =  0.4
Coefficients:
 [ 1.95134325e-03 -1.08772671e-02  1.28392054e-01  0.00000000e+00
  5.27928021e+00  8.00475075e-01 -6.07592080e+01  3.27000956e+03
```

```
 -1.36857816e+04 -4.92134423e+02  2.11585219e+00]
Mean squared error: 418257601.00
Variance score: 0.30
-------------------------
alpha =  0.6
Coefficients:
 [ 1.95134325e-03 -1.08772671e-02  1.28392054e-01  0.00000000e+00
  5.27928021e+00  8.00475075e-01 -6.07592080e+01  3.27000956e+03
 -1.36857816e+04 -4.92134423e+02  2.11585219e+00]
Mean squared error: 418257601.00
Variance score: 0.30
-------------------------
alpha =  0.8
Coefficients:
 [ 1.95134325e-03 -1.08772671e-02  1.28392054e-01  0.00000000e+00
  5.27928021e+00  8.00475075e-01 -6.07592080e+01  3.27000956e+03
 -1.36857816e+04 -4.92134423e+02  2.11585219e+00]
Mean squared error: 418257601.00
Variance score: 0.30
-------------------------
alpha =  1
Coefficients:
 [ 1.95134325e-03 -1.08772671e-02  1.28392054e-01  0.00000000e+00
  5.27928021e+00  8.00475075e-01 -6.07592080e+01  3.27000956e+03
 -1.36857816e+04 -4.92134423e+02  2.11585219e+00]
Mean squared error: 418257601.00
Variance score: 0.30
-------------------------
```

```
In [76]: for a in [3, 10, 18, 50, 100, 1000]:

             regr = linear_model.ElasticNet(alpha=0, max_iter=a)

             # Train the model using the training sets
             regr.fit(X_train, y_train)

             # Make predictions using the testing set
             y_pred = regr.predict(X_test)

             print('alpha = ', str(a))
             # The coefficients
             print('Coefficients: \n', regr.coef_)
             # The mean squared error
             print("Mean squared error: %.2f"
                   % mean_squared_error(y_test, y_pred))
             # Explained variance score: 1 is perfect prediction
             print('Variance score: %.2f' % r2_score(y_test, y_pred))
```

```
            print('-------------------------')
```

```
/Users/chrispaul/anaconda2/envs/nlp/lib/python3.6/site-packages/ipykernel_launcher.py:6: UserWa

/Users/chrispaul/anaconda2/envs/nlp/lib/python3.6/site-packages/sklearn/linear_model/coordinate
  positive)
/Users/chrispaul/anaconda2/envs/nlp/lib/python3.6/site-packages/sklearn/linear_model/coordinate
  ConvergenceWarning)


alpha =  3
Coefficients:
 [-9.31445297e-04 -1.09747225e-02  1.31362963e-01  0.00000000e+00
  5.23783572e+00  8.07259837e-01 -6.06477363e+01  3.27237151e+03
 -1.36469352e+04 -4.65112075e+02  2.04766015e+00]
Mean squared error: 418427184.07
Variance score: 0.30
-------------------------
alpha =  10
Coefficients:
 [ 1.94288381e-03 -1.08773894e-02  1.28388620e-01  0.00000000e+00
  5.27922274e+00  8.00477128e-01 -6.07590793e+01  3.27000972e+03
 -1.36857166e+04 -4.92061726e+02  2.11573827e+00]
Mean squared error: 418257650.54
Variance score: 0.30
-------------------------
alpha =  18
Coefficients:
 [ 1.95133561e-03 -1.08772672e-02  1.28392051e-01  0.00000000e+00
  5.27928015e+00  8.00475077e-01 -6.07592078e+01  3.27000956e+03
 -1.36857815e+04 -4.92134357e+02  2.11585209e+00]
Mean squared error: 418257601.04
Variance score: 0.30
-------------------------
alpha =  50
Coefficients:
 [ 1.95134325e-03 -1.08772671e-02  1.28392054e-01  0.00000000e+00
  5.27928021e+00  8.00475075e-01 -6.07592080e+01  3.27000956e+03
 -1.36857816e+04 -4.92134423e+02  2.11585219e+00]
Mean squared error: 418257601.00
Variance score: 0.30
-------------------------
alpha =  100
Coefficients:
 [ 1.95134325e-03 -1.08772671e-02  1.28392054e-01  0.00000000e+00
  5.27928021e+00  8.00475075e-01 -6.07592080e+01  3.27000956e+03
 -1.36857816e+04 -4.92134423e+02  2.11585219e+00]
Mean squared error: 418257601.00
```

```
Variance score: 0.30
-------------------------
alpha =  1000
Coefficients:
 [ 1.95134325e-03 -1.08772671e-02  1.28392054e-01  0.00000000e+00
  5.27928021e+00  8.00475075e-01 -6.07592080e+01  3.27000956e+03
 -1.36857816e+04 -4.92134423e+02  2.11585219e+00]
Mean squared error: 418257601.00
Variance score: 0.30
-------------------------
```

## 2.5 RF

```
In [81]: parameters = []
         mse = []

         for n in [2, 5, 10, 20, 50, 100]:
             for md in [3,5,10,20,50,100,1000]:
                 for mss in [2, 3, 5, 8, 10, 20]:

                     regr = RandomForestRegressor(max_depth=md, n_estimators=n, min_samples_spl

                     regr.fit(X_train, y_train)

                     y_pred = regr.predict(X_test)

                     mse_ = np.sqrt(np.mean((y_pred - y_test)**2))

                     parameters.append("n_estimators = " + str(n)
                                       + ", max_depth = " + str(md)
                                       + ", min_samples_split = " + str(mss))
                     mse.append(mse_)

             print("a sixth")
```

```
a sixth
a sixth
a sixth
a sixth
a sixth
a sixth
```

```
In [85]: ind = np.argmin(mse)

In [86]: ind

Out[86]: 236
```

In [87]: mse[236]

Out[87]: 9758.55780823173

In [89]: parameters[236]

Out[89]: 'n_estimators = 100, max_depth = 50, min_samples_split = 5'