car_price

December 12, 2019

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

```
In [2]: from sklearn.model_selection import KFold
        from sklearn.preprocessing import MinMaxScaler
        from sklearn.neighbors import KNeighborsRegressor
        from sklearn.linear_model import LinearRegression
        from sklearn.metrics import r2_score, accuracy_score, f1_score, roc_auc_score
        from sklearn.ensemble import RandomForestRegressor, BaggingRegressor, RandomForestClas
```

```
In [3]: %matplotlib inline
```

```
In [4]: df = pd.read_csv('train_data.csv')
```

```
In [5]: dft = pd.read_csv('test_data.csv')
```

0.1 Imputation

- Rotary Engines and Electric Engines lack cylinders so they are filled with **0 in Engine Cylinders**.
- **Engine Horsepower** is missing in some cases of electric engines. So the **mean value** is filled. However this **can be improved** since this is an important predictor of MSRP.
- **Number of Doors** is mostly 4 so missing values were filled with **4**.

```
In [6]: df.isna().sum()
```

```
Out[6]: Make                  0
        Model                 0
        Year                  0
        Engine Fuel Type      3
        Engine HP            65
        Engine Cylinders     23
        Transmission Type     0
        Driven_Wheels         0
        Number of Doors       4
```

```
Market Category      3376
Vehicle Size            0
Vehicle Style           0
highway MPG             0
city mpg                0
Popularity              0
MSRP                    0
dtype: int64
```

In [7]:
```
df[df['Number of Doors'].isnull()]
```

Out[7]:
```
        Make    Model   Year Engine Fuel Type  Engine HP  Engine Cylinders  \
2456   Tesla  Model S  2016          electric        NaN               0.0
4074   Tesla  Model S  2016          electric        NaN               0.0
4668   Tesla  Model S  2016          electric        NaN               0.0
10196  Tesla  Model S  2016          electric        NaN               0.0

      Transmission Type     Driven_Wheels  Number of Doors  \
2456       DIRECT_DRIVE   all wheel drive              NaN
4074       DIRECT_DRIVE  rear wheel drive              NaN
4668       DIRECT_DRIVE   all wheel drive              NaN
10196      DIRECT_DRIVE   all wheel drive              NaN

               Market Category Vehicle Size Vehicle Style  highway MPG  \
2456        Exotic,Performance        Large         Sedan          105
4074        Exotic,Performance        Large         Sedan          100
4668   Exotic,High-Performance        Large         Sedan          105
10196       Exotic,Performance        Large         Sedan          107

       city mpg  Popularity    MSRP
2456        102        1391   79500
4074         97        1391   74500
4668         92        1391  134500
10196       101        1391   71000
```

In [8]:
```
df[df['Model']=='Model S']['Number of Doors']
```

Out[8]:
```
278      4.0
1526     4.0
1626     4.0
2222     4.0
2456     NaN
3547     4.0
3882     4.0
4074     NaN
4087     4.0
4668     NaN
6529     4.0
7552     4.0
```

```
8626     4.0
9040     4.0
10048    4.0
10196    NaN
10308    4.0
Name: Number of Doors, dtype: float64
```

In [9]:
```python
df.loc[df['Number of Doors'].isnull(), 'Number of Doors'] = 4
```

In [10]:
```python
df[df['Engine HP'].isnull()]
```

Out[10]:
```
                Make        Model  Year                    Engine Fuel Type  \
278            Tesla      Model S  2015                            electric
343              Kia      Soul EV  2015                            electric
744             Ford       Escape  2017                    regular unleaded
820        Chevrolet       Impala  2016  flex-fuel (unleaded/natural gas)
955          Lincoln          MKZ  2017                    regular unleaded
1033            Ford     Freestar  2005                    regular unleaded
1110            Ford        Focus  2017                            electric
1519         Lincoln          MKZ  2017                    regular unleaded
1526           Tesla      Model S  2015                            electric
1595       Chevrolet       Impala  2015  flex-fuel (unleaded/natural gas)
1626           Tesla      Model S  2016                            electric
1880            Ford     Freestar  2005                    regular unleaded
2173           Honda       Fit EV  2013                            electric
2222           Tesla      Model S  2014                            electric
2456           Tesla      Model S  2016                            electric
2625          Nissan         Leaf  2015                            electric
2671             Kia      Soul EV  2016                            electric
2987          Nissan         Leaf  2014                            electric
3043            Ford       Escape  2017                    regular unleaded
3113       Chevrolet       Impala  2017  flex-fuel (unleaded/natural gas)
3154       Chevrolet       Impala  2016  flex-fuel (unleaded/natural gas)
3225          Nissan         Leaf  2014                            electric
3358           Honda       Fit EV  2014                            electric
3547           Tesla      Model S  2014                            electric
3882           Tesla      Model S  2016                            electric
4033         Lincoln  Continental  2017    premium unleaded (recommended)
4074           Tesla      Model S  2016                            electric
4087           Tesla      Model S  2014                            electric
4296             Kia      Soul EV  2015                            electric
4325         Lincoln          MKZ  2017                    regular unleaded
...              ...          ...   ...                                 ...
5089            Ford     Freestar  2005                    regular unleaded
5108         Lincoln  Continental  2017    premium unleaded (recommended)
5559            Ford     Freestar  2005                    regular unleaded
6033       Chevrolet       Impala  2015  flex-fuel (unleaded/natural gas)
6074            Ford       Escape  2017                    regular unleaded
```

6529	Tesla	Model S	2015	electric
6703	FIAT	500e	2016	electric
6769	Nissan	Leaf	2016	electric
6806	Mercedes-Benz	M-Class	2015	diesel
6827	Lincoln	Continental	2017	premium unleaded (recommended)
6943	FIAT	500e	2015	electric
7147	Nissan	Leaf	2015	electric
7254	Toyota	RAV4 EV	2013	electric
7441	Ford	Escape	2017	regular unleaded
7463	Ford	Freestar	2005	regular unleaded
7502	Nissan	Leaf	2015	electric
7552	Tesla	Model S	2016	electric
7966	Nissan	Leaf	2016	electric
7967	Nissan	Leaf	2015	electric
8312	Ford	Focus	2016	electric
8626	Tesla	Model S	2015	electric
8639	Nissan	Leaf	2016	electric
8908	FIAT	500e	2017	electric
9040	Tesla	Model S	2015	electric
9601	Nissan	Leaf	2014	electric
10048	Tesla	Model S	2016	electric
10135	Lincoln	MKZ	2017	regular unleaded
10196	Tesla	Model S	2016	electric
10308	Tesla	Model S	2014	electric
10338	Kia	Soul EV	2016	electric

	Engine HP	Engine Cylinders	Transmission Type	Driven_Wheels \
278	NaN	0.0	DIRECT_DRIVE	rear wheel drive
343	NaN	0.0	DIRECT_DRIVE	front wheel drive
744	NaN	4.0	AUTOMATIC	all wheel drive
820	NaN	6.0	AUTOMATIC	front wheel drive
955	NaN	4.0	AUTOMATIC	front wheel drive
1033	NaN	6.0	AUTOMATIC	front wheel drive
1110	NaN	0.0	DIRECT_DRIVE	front wheel drive
1519	NaN	4.0	AUTOMATIC	front wheel drive
1526	NaN	0.0	DIRECT_DRIVE	all wheel drive
1595	NaN	6.0	AUTOMATIC	front wheel drive
1626	NaN	0.0	DIRECT_DRIVE	all wheel drive
1880	NaN	6.0	AUTOMATIC	front wheel drive
2173	NaN	0.0	DIRECT_DRIVE	front wheel drive
2222	NaN	0.0	DIRECT_DRIVE	rear wheel drive
2456	NaN	0.0	DIRECT_DRIVE	all wheel drive
2625	NaN	0.0	DIRECT_DRIVE	front wheel drive
2671	NaN	0.0	DIRECT_DRIVE	front wheel drive
2987	NaN	0.0	DIRECT_DRIVE	front wheel drive
3043	NaN	4.0	AUTOMATIC	front wheel drive
3113	NaN	6.0	AUTOMATIC	front wheel drive
3154	NaN	6.0	AUTOMATIC	front wheel drive

```
3225          NaN                 0.0       DIRECT_DRIVE  front wheel drive
3358          NaN                 0.0       DIRECT_DRIVE  front wheel drive
3547          NaN                 0.0       DIRECT_DRIVE   rear wheel drive
3882          NaN                 0.0       DIRECT_DRIVE    all wheel drive
4033          NaN                 6.0          AUTOMATIC    all wheel drive
4074          NaN                 0.0       DIRECT_DRIVE   rear wheel drive
4087          NaN                 0.0       DIRECT_DRIVE    all wheel drive
4296          NaN                 0.0       DIRECT_DRIVE  front wheel drive
4325          NaN                 4.0          AUTOMATIC  front wheel drive
...           ...                 ...                ...                ...
5089          NaN                 6.0          AUTOMATIC  front wheel drive
5108          NaN                 6.0          AUTOMATIC    all wheel drive
5559          NaN                 6.0          AUTOMATIC  front wheel drive
6033          NaN                 6.0          AUTOMATIC  front wheel drive
6074          NaN                 4.0          AUTOMATIC  front wheel drive
6529          NaN                 0.0       DIRECT_DRIVE    all wheel drive
6703          NaN                 0.0       DIRECT_DRIVE  front wheel drive
6769          NaN                 0.0       DIRECT_DRIVE  front wheel drive
6806          NaN                 4.0          AUTOMATIC    all wheel drive
6827          NaN                 6.0          AUTOMATIC  front wheel drive
6943          NaN                 0.0       DIRECT_DRIVE  front wheel drive
7147          NaN                 0.0       DIRECT_DRIVE  front wheel drive
7254          NaN                 0.0       DIRECT_DRIVE  front wheel drive
7441          NaN                 4.0          AUTOMATIC    all wheel drive
7463          NaN                 6.0          AUTOMATIC  front wheel drive
7502          NaN                 0.0       DIRECT_DRIVE  front wheel drive
7552          NaN                 0.0       DIRECT_DRIVE   rear wheel drive
7966          NaN                 0.0       DIRECT_DRIVE  front wheel drive
7967          NaN                 0.0       DIRECT_DRIVE  front wheel drive
8312          NaN                 0.0       DIRECT_DRIVE  front wheel drive
8626          NaN                 0.0       DIRECT_DRIVE    all wheel drive
8639          NaN                 0.0       DIRECT_DRIVE  front wheel drive
8908          NaN                 0.0       DIRECT_DRIVE  front wheel drive
9040          NaN                 0.0       DIRECT_DRIVE   rear wheel drive
9601          NaN                 0.0       DIRECT_DRIVE  front wheel drive
10048         NaN                 0.0       DIRECT_DRIVE    all wheel drive
10135         NaN                 4.0          AUTOMATIC  front wheel drive
10196         NaN                 0.0       DIRECT_DRIVE    all wheel drive
10308         NaN                 0.0       DIRECT_DRIVE   rear wheel drive
10338         NaN                 0.0       DIRECT_DRIVE  front wheel drive

       Number of Doors          Market Category Vehicle Size  \
278                4.0       Exotic,Performance        Large
343                4.0                      NaN      Compact
744                4.0                Crossover      Compact
820                4.0    Flex Fuel,Performance        Large
955                4.0           Luxury,Hybrid      Midsize
1033               4.0                      NaN      Midsize
```

1110	4.0	Hatchback	Compact
1519	4.0	Luxury,Hybrid	Midsize
1526	4.0	Exotic,High-Performance	Large
1595	4.0	Flex Fuel,Performance	Large
1626	4.0	Exotic,High-Performance	Large
1880	4.0	NaN	Midsize
2173	4.0	Hatchback	Compact
2222	4.0	Exotic,Performance	Large
2456	4.0	Exotic,Performance	Large
2625	4.0	Hatchback	Compact
2671	4.0	NaN	Compact
2987	4.0	Hatchback	Compact
3043	4.0	Crossover	Compact
3113	4.0	Flex Fuel,Performance	Large
3154	4.0	Flex Fuel,Performance	Large
3225	4.0	Hatchback	Compact
3358	4.0	Hatchback	Compact
3547	4.0	Exotic,High-Performance	Large
3882	4.0	Exotic,Performance	Large
4033	4.0	Luxury	Large
4074	4.0	Exotic,Performance	Large
4087	4.0	Exotic,High-Performance	Large
4296	4.0	NaN	Compact
4325	4.0	Luxury,Hybrid	Midsize
...	...	...	...
5089	4.0	NaN	Midsize
5108	4.0	Luxury	Large
5559	4.0	NaN	Midsize
6033	4.0	Flex Fuel,Performance	Large
6074	4.0	Crossover	Compact
6529	4.0	Exotic,High-Performance	Large
6703	2.0	Hatchback	Compact
6769	4.0	Hatchback	Compact
6806	4.0	Crossover,Luxury,Diesel	Midsize
6827	4.0	Luxury	Large
6943	2.0	Hatchback	Compact
7147	4.0	Hatchback	Compact
7254	4.0	Crossover	Midsize
7441	4.0	Crossover	Compact
7463	4.0	NaN	Midsize
7502	4.0	Hatchback	Compact
7552	4.0	Exotic,Performance	Large
7966	4.0	Hatchback	Compact
7967	4.0	Hatchback	Compact
8312	4.0	Hatchback	Compact
8626	4.0	Exotic,Performance	Large
8639	4.0	Hatchback	Compact
8908	2.0	Hatchback	Compact

9040	4.0	Exotic,Performance	Large
9601	4.0	Hatchback	Compact
10048	4.0	Exotic,High-Performance	Large
10135	4.0	Luxury,Hybrid	Midsize
10196	4.0	Exotic,Performance	Large
10308	4.0	Exotic,High-Performance	Large
10338	4.0	NaN	Compact

	Vehicle Style	highway MPG	city mpg	Popularity	MSRP
278	Sedan	90	88	1391	80000
343	Wagon	92	120	1720	33700
744	4dr SUV	28	22	5657	26850
820	Sedan	25	17	1385	37570
955	Sedan	38	41	61	39510
1033	Passenger Minivan	22	16	5657	26530
1110	4dr Hatchback	99	110	5657	29120
1519	Sedan	38	41	61	36760
1526	Sedan	106	95	1391	85000
1595	Sedan	25	17	1385	40660
1626	Sedan	100	91	1391	112000
1880	Passenger Minivan	21	16	5657	29030
2173	4dr Hatchback	105	132	2202	36625
2222	Sedan	97	94	1391	69900
2456	Sedan	105	102	1391	79500
2625	4dr Hatchback	101	126	2009	29010
2671	Wagon	92	120	1720	31950
2987	4dr Hatchback	101	126	2009	32000
3043	4dr SUV	30	23	5657	29100
3113	Sedan	25	17	1385	40915
3154	Sedan	25	17	1385	40810
3225	4dr Hatchback	101	126	2009	28980
3358	4dr Hatchback	105	132	2202	36625
3547	Sedan	90	88	1391	79900
3882	Sedan	102	101	1391	75000
4033	Sedan	25	17	61	64915
4074	Sedan	100	97	1391	74500
4087	Sedan	94	86	1391	104500
4296	Wagon	92	120	1720	35700
4325	Sedan	38	41	61	35010
...	...	...	...	...	...
5089	Passenger Minivan	22	16	5657	23930
5108	Sedan	25	17	61	55915
5559	Cargo Minivan	22	16	5657	21630
6033	Sedan	25	17	1385	37535
6074	4dr SUV	30	23	5657	25100
6529	Sedan	98	89	1391	105000
6703	2dr Hatchback	103	121	819	31800
6769	4dr Hatchback	101	126	2009	29010

```
6806              4dr SUV          29        22          617    49800
6827                Sedan          27        18           61    62915
6943        2dr Hatchback         108       122          819    31800
7147        4dr Hatchback         101       126         2009    32100
7254              4dr SUV          74        78         2031    49800
7441              4dr SUV          28        22         5657    30850
7463    Passenger Minivan          22        16         5657    28030
7502        4dr Hatchback         101       126         2009    32000
7552                Sedan          90        88         1391    70000
7966        4dr Hatchback         101       124         2009    34200
7967        4dr Hatchback         101       126         2009    35120
8312        4dr Hatchback          99       110         5657    29170
8626                Sedan         102       101         1391    75000
8639        4dr Hatchback         101       124         2009    36790
8908        2dr Hatchback         103       121          819    31800
9040                Sedan          97        94         1391    69900
9601        4dr Hatchback         101       126         2009    35020
10048               Sedan         107       101         1391    89500
10135               Sedan          38        41           61    47670
10196               Sedan         107       101         1391    71000
10308               Sedan          90        88         1391    93400
10338               Wagon          92       120         1720    33950

[65 rows x 16 columns]
```

In [11]:
```
ENGINE_HP_MEAN = df['Engine HP'].mean()
df.loc[df['Engine HP'].isnull(), 'Engine HP'] = ENGINE_HP_MEAN
```

In [12]:
```
df[df['Engine Cylinders'].isnull()]
```

Out[12]:
```
            Make   Model  Year             Engine Fuel Type  Engine HP  \
813        Mazda    RX-8  2011  premium unleaded (required)      232.0
925       Toyota RAV4 EV  2012                     electric      154.0
1131       Mazda    RX-8  2010  premium unleaded (required)      232.0
1785       Mazda    RX-8  2010  premium unleaded (required)      212.0
1938       Mazda    RX-8  2011  premium unleaded (required)      212.0
2181   Chevrolet Bolt EV  2017                     electric      200.0
2395       Mazda    RX-8  2010  premium unleaded (required)      232.0
3012       Mazda    RX-8  2009  premium unleaded (required)      212.0
4760       Mazda    RX-8  2009  premium unleaded (required)      212.0
5052       Mazda    RX-8  2009  premium unleaded (required)      232.0
5308       Mazda    RX-7  1993             regular unleaded      255.0
5443  Mitsubishi  i-MiEV  2017                     electric       66.0
5906       Mazda    RX-7  1994             regular unleaded      255.0
5945       Mazda    RX-8  2010  premium unleaded (required)      232.0
6659  Mitsubishi  i-MiEV  2016                     electric       66.0
7563       Mazda    RX-8  2011  premium unleaded (required)      212.0
7917       Mazda    RX-8  2009  premium unleaded (required)      232.0
```

```
8528   Volkswagen   e-Golf  2015                          electric      115.0
9194   Volkswagen   e-Golf  2015                          electric      115.0
9253        Mazda     RX-8  2011  premium unleaded (required)      232.0
9456        Mazda     RX-8  2011  premium unleaded (required)      232.0
10261       Mazda     RX-8  2009  premium unleaded (required)      232.0
10577       Mazda     RX-8  2010  premium unleaded (required)      212.0

       Engine Cylinders Transmission Type      Driven_Wheels  Number of Doors  \
813                 NaN            MANUAL   rear wheel drive              4.0
925                 NaN      DIRECT_DRIVE  front wheel drive              4.0
1131                NaN            MANUAL   rear wheel drive              4.0
1785                NaN         AUTOMATIC   rear wheel drive              4.0
1938                NaN         AUTOMATIC   rear wheel drive              4.0
2181                NaN      DIRECT_DRIVE  front wheel drive              4.0
2395                NaN            MANUAL   rear wheel drive              4.0
3012                NaN         AUTOMATIC   rear wheel drive              4.0
4760                NaN         AUTOMATIC   rear wheel drive              4.0
5052                NaN            MANUAL   rear wheel drive              4.0
5308                NaN            MANUAL   rear wheel drive              2.0
5443                NaN      DIRECT_DRIVE   rear wheel drive              4.0
5906                NaN            MANUAL   rear wheel drive              2.0
5945                NaN            MANUAL   rear wheel drive              4.0
6659                NaN      DIRECT_DRIVE   rear wheel drive              4.0
7563                NaN         AUTOMATIC   rear wheel drive              4.0
7917                NaN            MANUAL   rear wheel drive              4.0
8528                NaN      DIRECT_DRIVE  front wheel drive              4.0
9194                NaN      DIRECT_DRIVE  front wheel drive              4.0
9253                NaN            MANUAL   rear wheel drive              4.0
9456                NaN            MANUAL   rear wheel drive              4.0
10261               NaN            MANUAL   rear wheel drive              4.0
10577               NaN         AUTOMATIC   rear wheel drive              4.0

                  Market Category Vehicle Size  Vehicle Style  highway MPG  \
813                   Performance      Compact          Coupe           22
925                     Crossover      Midsize        4dr SUV           74
1131                  Performance      Compact          Coupe           22
1785                  Performance      Compact          Coupe           23
1938                  Performance      Compact          Coupe           23
2181                    Hatchback      Compact  4dr Hatchback          110
2395                  Performance      Compact          Coupe           22
3012                  Performance      Compact          Coupe           23
4760                  Performance      Compact          Coupe           23
5052                  Performance      Compact          Coupe           22
5308   Factory Tuner,Performance      Compact          Coupe           23
5443                    Hatchback      Compact  4dr Hatchback          102
5906   Factory Tuner,Performance      Compact          Coupe           23
5945                  Performance      Compact          Coupe           22
6659                    Hatchback      Compact  4dr Hatchback           99
```

```
7563                Performance    Compact          Coupe          23
7917                Performance    Compact          Coupe          22
8528                  Hatchback    Compact  4dr Hatchback         105
9194                  Hatchback    Compact  4dr Hatchback         105
9253                Performance    Compact          Coupe          22
9456                Performance    Compact          Coupe          22
10261               Performance    Compact          Coupe          22
10577               Performance    Compact          Coupe          23

       city mpg  Popularity   MSRP
813          16         586  26795
925          78        2031  49800
1131         16         586  26645
1785         16         586  26645
1938         16         586  26795
2181        128        1385  40905
2395         16         586  32140
3012         16         586  28560
4760         16         586  31700
5052         16         586  31930
5308         15         586   7523
5443        121         436  22995
5906         15         586   8147
5945         16         586  32110
6659        126         436  22995
7563         16         586  32960
7917         16         586  31000
8528        126         873  35445
9194        126         873  33450
9253         16         586  32260
9456         16         586  32290
10261        16         586  27860
10577        16         586  32810
```

In [13]:
```
df.loc[df['Model']=='RX-8', 'Engine Cylinders'] = 0
```

In [14]:
```
df.loc[df['Model']=='RX-7', 'Engine Cylinders'] = 0
```

In [15]:
```
df.loc[df['Engine Cylinders'].isnull(), 'Engine Cylinders'] = 0
```

In [16]:
```
df.isna().sum()
```

Out[16]:
```
Make                 0
Model                0
Year                 0
Engine Fuel Type     3
Engine HP            0
Engine Cylinders     0
Transmission Type    0
```

```
Driven_Wheels           0
Number of Doors         0
Market Category      3376
Vehicle Size            0
Vehicle Style           0
highway MPG             0
city mpg                0
Popularity              0
MSRP                    0
dtype: int64
```

0.2 Exploration

- All columns in the train_data and test_data were compared, and it was found that **some columns in test_data contain values that are absent in train_data** e.g.
  - Make
  - Model
  - Engine Fuel Type
  - Market Category
- These were therefore removed from the training data
- The distribution of MSRP values was studies because **simple regression was performing very poorly**.
- It was found that the **MSRP values** could be **divided into four categories**.

```
In [17]: df.head().T
```

```
Out[17]:                                   0                  1                  2  \
         Make                            GMC              Scion            Hyundai
         Model                       Terrain                 xD           Veloster
         Year                           2017               2013               2016
         Engine Fuel Type   regular unleaded   regular unleaded   regular unleaded
         Engine HP                       182                128                132
         Engine Cylinders                  4                  4                  4
         Transmission Type         AUTOMATIC          AUTOMATIC   AUTOMATED_MANUAL
         Driven_Wheels     front wheel drive  front wheel drive  front wheel drive
         Number of Doors                   4                  4                  3
         Market Category           Crossover          Hatchback          Hatchback
         Vehicle Size                Compact            Compact            Compact
         Vehicle Style               4dr SUV      4dr Hatchback      2dr Hatchback
         highway MPG                      31                 33                 36
         city mpg                         21                 27                 28
         Popularity                      549                105               1439
         MSRP                          27300              16545              19100

                                             3                  4
         Make                              GMC              Mazda
         Model             Sierra 1500 Classic           B-Series
```

```
Year                              2007                2001
Engine Fuel Type      regular unleaded    regular unleaded
Engine HP                          285                 150
Engine Cylinders                     8                   6
Transmission Type            AUTOMATIC              MANUAL
Driven_Wheels         four wheel drive    rear wheel drive
Number of Doors                      2                   2
Market Category              Flex Fuel                 NaN
Vehicle Size                     Large             Compact
Vehicle Style       Regular Cab Pickup  Extended Cab Pickup
highway MPG                         18                  21
city mpg                            14                  15
Popularity                         549                 586
MSRP                             26295               17545
```

In [18]:
```
df.columns
```

Out[18]:
```
Index(['Make', 'Model', 'Year', 'Engine Fuel Type', 'Engine HP',
       'Engine Cylinders', 'Transmission Type', 'Driven_Wheels',
       'Number of Doors', 'Market Category', 'Vehicle Size', 'Vehicle Style',
       'highway MPG', 'city mpg', 'Popularity', 'MSRP'],
      dtype='object')
```

In [19]:
```
df['Vehicle Size'].unique()
```

Out[19]:
```
array(['Compact', 'Large', 'Midsize'], dtype=object)
```

In [20]:
```
df['Driven_Wheels'].unique()
```

Out[20]:
```
array(['front wheel drive', 'four wheel drive', 'rear wheel drive',
       'all wheel drive'], dtype=object)
```

In [21]:
```
df['Vehicle Style'].unique()
```

Out[21]:
```
array(['4dr SUV', '4dr Hatchback', '2dr Hatchback', 'Regular Cab Pickup',
       'Extended Cab Pickup', 'Sedan', 'Coupe', 'Convertible',
       'Crew Cab Pickup', '2dr SUV', 'Passenger Van', 'Wagon',
       'Cargo Minivan', 'Cargo Van', 'Passenger Minivan', 'Convertible SUV'], dtype=ol
```

In [22]:
```
df[df['Vehicle Size'] == 'Compact']['Vehicle Style'].unique()
```

Out[22]:
```
array(['4dr SUV', '4dr Hatchback', '2dr Hatchback', 'Extended Cab Pickup',
       'Regular Cab Pickup', 'Coupe', 'Convertible', '2dr SUV',
       'Crew Cab Pickup', 'Sedan', 'Wagon', 'Passenger Minivan',
       'Passenger Van', 'Cargo Minivan', 'Convertible SUV', 'Cargo Van'], dtype=objec
```

In [23]:
```
df[df['Vehicle Size'] == 'Midsize']['Vehicle Style'].unique()
```

Out[23]:
```
array(['4dr SUV', 'Convertible', 'Sedan', 'Passenger Van', 'Cargo Minivan',
       'Coupe', 'Cargo Van', 'Passenger Minivan', 'Wagon', '4dr Hatchback',
       '2dr SUV', '2dr Hatchback', 'Convertible SUV'], dtype=object)
```

```
In [24]: df[df['Vehicle Size'] == 'Large']['Vehicle Style'].unique()
```

```
Out[24]: array(['Regular Cab Pickup', 'Sedan', 'Crew Cab Pickup', 'Wagon',
                'Extended Cab Pickup', '4dr SUV', 'Passenger Minivan', 'Cargo Van',
                'Passenger Van', '4dr Hatchback', 'Cargo Minivan', 'Coupe',
                'Convertible'], dtype=object)
```

```
In [25]: df[df['Model'] == 'Sierra 1500 Classic'].sort_values('MSRP', axis=0)
```

```
Out[25]:       Make                Model  Year          Engine Fuel Type  Engine HP  \
         5749   GMC  Sierra 1500 Classic  2007          regular unleaded      195.0
         826    GMC  Sierra 1500 Classic  2007          regular unleaded      195.0
         2246   GMC  Sierra 1500 Classic  2007          regular unleaded      195.0
         5792   GMC  Sierra 1500 Classic  2007          regular unleaded      195.0
         6372   GMC  Sierra 1500 Classic  2007          regular unleaded      195.0
         546    GMC  Sierra 1500 Classic  2007          regular unleaded      285.0
         477    GMC  Sierra 1500 Classic  2007  flex-fuel (unleaded/E85)      295.0
         9119   GMC  Sierra 1500 Classic  2007          regular unleaded      285.0
         9531   GMC  Sierra 1500 Classic  2007          regular unleaded      285.0
         4270   GMC  Sierra 1500 Classic  2007          regular unleaded      285.0
         3      GMC  Sierra 1500 Classic  2007          regular unleaded      285.0
         4118   GMC  Sierra 1500 Classic  2007          regular unleaded      285.0
         9397   GMC  Sierra 1500 Classic  2007          regular unleaded      285.0
         6990   GMC  Sierra 1500 Classic  2007  flex-fuel (unleaded/E85)      295.0
         9950   GMC  Sierra 1500 Classic  2007          regular unleaded      295.0
         1968   GMC  Sierra 1500 Classic  2007          regular unleaded      285.0
         8121   GMC  Sierra 1500 Classic  2007          regular unleaded      285.0
         3340   GMC  Sierra 1500 Classic  2007  flex-fuel (unleaded/E85)      295.0
         3809   GMC  Sierra 1500 Classic  2007          regular unleaded      285.0
         1525   GMC  Sierra 1500 Classic  2007  flex-fuel (unleaded/E85)      295.0
         6080   GMC  Sierra 1500 Classic  2007          regular unleaded      285.0
         8440   GMC  Sierra 1500 Classic  2007          regular unleaded      295.0
         3197   GMC  Sierra 1500 Classic  2007  flex-fuel (unleaded/E85)      295.0
         551    GMC  Sierra 1500 Classic  2007  flex-fuel (unleaded/E85)      295.0
         6091   GMC  Sierra 1500 Classic  2007  flex-fuel (unleaded/E85)      295.0
         7675   GMC  Sierra 1500 Classic  2007          regular unleaded      285.0
         2953   GMC  Sierra 1500 Classic  2007          regular unleaded      285.0
         5362   GMC  Sierra 1500 Classic  2007  flex-fuel (unleaded/E85)      295.0
         1909   GMC  Sierra 1500 Classic  2007  flex-fuel (unleaded/E85)      295.0
         3806   GMC  Sierra 1500 Classic  2007  flex-fuel (unleaded/E85)      295.0
         7056   GMC  Sierra 1500 Classic  2007  flex-fuel (unleaded/E85)      295.0
         8388   GMC  Sierra 1500 Classic  2007          regular unleaded      295.0
         9192   GMC  Sierra 1500 Classic  2007  flex-fuel (unleaded/E85)      295.0
         8693   GMC  Sierra 1500 Classic  2007          regular unleaded      310.0
         6771   GMC  Sierra 1500 Classic  2007  flex-fuel (unleaded/E85)      295.0
         7991   GMC  Sierra 1500 Classic  2007  flex-fuel (unleaded/E85)      295.0
         10087  GMC  Sierra 1500 Classic  2007          regular unleaded      295.0
         4658   GMC  Sierra 1500 Classic  2007  flex-fuel (unleaded/E85)      295.0
```

419	GMC	Sierra 1500 Classic	2007	regular unleaded	310.0
6304	GMC	Sierra 1500 Classic	2007	flex-fuel (unleaded/E85)	295.0
4637	GMC	Sierra 1500 Classic	2007	regular unleaded	345.0

	Engine Cylinders	Transmission Type	Driven_Wheels	Number of Doors \
5749	6.0	MANUAL	rear wheel drive	2.0
826	6.0	MANUAL	rear wheel drive	2.0
2246	6.0	AUTOMATIC	four wheel drive	2.0
5792	6.0	MANUAL	rear wheel drive	2.0
6372	6.0	AUTOMATIC	rear wheel drive	4.0
546	8.0	AUTOMATIC	rear wheel drive	2.0
477	8.0	AUTOMATIC	rear wheel drive	4.0
9119	8.0	AUTOMATIC	rear wheel drive	4.0
9531	8.0	AUTOMATIC	four wheel drive	4.0
4270	8.0	AUTOMATIC	four wheel drive	2.0
3	8.0	AUTOMATIC	four wheel drive	2.0
4118	8.0	AUTOMATIC	rear wheel drive	4.0
9397	8.0	AUTOMATIC	rear wheel drive	4.0
6990	8.0	AUTOMATIC	four wheel drive	4.0
9950	8.0	AUTOMATIC	rear wheel drive	4.0
1968	8.0	AUTOMATIC	four wheel drive	2.0
8121	8.0	AUTOMATIC	four wheel drive	4.0
3340	8.0	AUTOMATIC	rear wheel drive	4.0
3809	8.0	AUTOMATIC	four wheel drive	2.0
1525	8.0	AUTOMATIC	four wheel drive	4.0
6080	8.0	AUTOMATIC	four wheel drive	4.0
8440	8.0	AUTOMATIC	rear wheel drive	4.0
3197	8.0	AUTOMATIC	rear wheel drive	4.0
551	8.0	AUTOMATIC	rear wheel drive	4.0
6091	8.0	AUTOMATIC	rear wheel drive	4.0
7675	8.0	AUTOMATIC	four wheel drive	4.0
2953	8.0	AUTOMATIC	four wheel drive	4.0
5362	8.0	AUTOMATIC	four wheel drive	4.0
1909	8.0	AUTOMATIC	rear wheel drive	4.0
3806	8.0	AUTOMATIC	rear wheel drive	4.0
7056	8.0	AUTOMATIC	rear wheel drive	4.0
8388	8.0	AUTOMATIC	four wheel drive	4.0
9192	8.0	AUTOMATIC	four wheel drive	4.0
8693	8.0	AUTOMATIC	four wheel drive	4.0
6771	8.0	AUTOMATIC	four wheel drive	4.0
7991	8.0	AUTOMATIC	rear wheel drive	4.0
10087	8.0	AUTOMATIC	four wheel drive	4.0
4658	8.0	AUTOMATIC	four wheel drive	4.0
419	8.0	AUTOMATIC	four wheel drive	4.0
6304	8.0	AUTOMATIC	four wheel drive	4.0
4637	8.0	AUTOMATIC	all wheel drive	4.0

	Market Category	Vehicle Size	Vehicle Style	highway MPG \

5749	Flex Fuel	Large	Regular Cab Pickup	21
826	Flex Fuel	Large	Regular Cab Pickup	21
2246	Flex Fuel	Large	Regular Cab Pickup	17
5792	Flex Fuel	Large	Regular Cab Pickup	21
6372	Flex Fuel	Large	Extended Cab Pickup	20
546	Flex Fuel	Large	Regular Cab Pickup	19
477	Flex Fuel	Large	Extended Cab Pickup	19
9119	Flex Fuel	Large	Extended Cab Pickup	19
9531	Flex Fuel	Large	Extended Cab Pickup	18
4270	Flex Fuel	Large	Regular Cab Pickup	18
3	Flex Fuel	Large	Regular Cab Pickup	18
4118	Flex Fuel	Large	Extended Cab Pickup	19
9397	NaN	Large	Crew Cab Pickup	19
6990	Flex Fuel	Large	Extended Cab Pickup	17
9950	NaN	Large	Extended Cab Pickup	19
1968	Flex Fuel	Large	Regular Cab Pickup	18
8121	Flex Fuel	Large	Extended Cab Pickup	18
3340	Flex Fuel	Large	Extended Cab Pickup	19
3809	Flex Fuel	Large	Regular Cab Pickup	18
1525	Flex Fuel	Large	Extended Cab Pickup	17
6080	NaN	Large	Crew Cab Pickup	18
8440	NaN	Large	Extended Cab Pickup	19
3197	Flex Fuel	Large	Crew Cab Pickup	19
551	Flex Fuel	Large	Extended Cab Pickup	19
6091	Flex Fuel	Large	Extended Cab Pickup	19
7675	Flex Fuel	Large	Extended Cab Pickup	18
2953	NaN	Large	Crew Cab Pickup	18
5362	Flex Fuel	Large	Extended Cab Pickup	17
1909	Flex Fuel	Large	Extended Cab Pickup	19
3806	Flex Fuel	Large	Crew Cab Pickup	19
7056	Flex Fuel	Large	Extended Cab Pickup	19
8388	NaN	Large	Extended Cab Pickup	17
9192	Flex Fuel	Large	Crew Cab Pickup	17
8693	Flex Fuel	Large	Extended Cab Pickup	17
6771	Flex Fuel	Large	Extended Cab Pickup	17
7991	Flex Fuel	Large	Crew Cab Pickup	19
10087	NaN	Large	Extended Cab Pickup	17
4658	Flex Fuel	Large	Crew Cab Pickup	17
419	Flex Fuel	Large	Extended Cab Pickup	17
6304	Flex Fuel	Large	Crew Cab Pickup	17
4637	NaN	Large	Crew Cab Pickup	16

	city mpg	Popularity	MSRP
5749	15	549	15840
826	15	549	16115
2246	14	549	20715
5792	15	549	20750
6372	14	549	21465

546	15	549	24140
477	14	549	24515
9119	15	549	25215
9531	14	549	25845
4270	14	549	25850
3	14	549	26295
4118	15	549	26555
9397	15	549	26820
6990	13	549	27060
9950	14	549	27430
1968	14	549	27560
8121	14	549	27580
3340	14	549	27730
3809	14	549	27850
1525	13	549	28790
6080	14	549	28845
8440	14	549	28915
3197	14	549	29015
551	14	549	29065
6091	14	549	29355
7675	14	549	29380
2953	14	549	30340
5362	13	549	30550
1909	14	549	30825
3806	14	549	30930
7056	14	549	31115
8388	13	549	31735
9192	13	549	31965
8693	13	549	32025
6771	13	549	32170
7991	14	549	32690
10087	13	549	33495
4658	13	549	33755
419	13	549	33785
6304	13	549	35520
4637	13	549	39125

```
In [26]: set(df['Vehicle Style'].unique()) == set(dft['Vehicle Style'].unique())
```

```
Out[26]: True
```

```
In [27]: type(df['MSRP'][0])
```

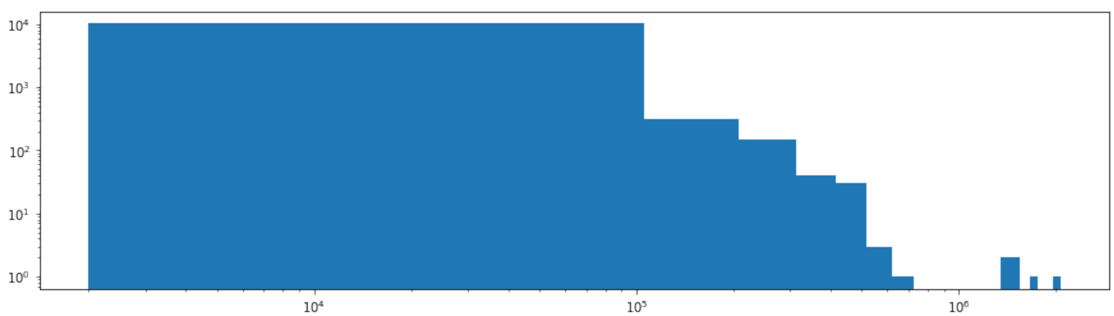
```
Out[27]: numpy.int64
```

```
In [28]: for col in df.columns:
             print(col, 'contains no oov values', set(df[col].unique()) == set(dft[col].unique
```

```
Make contains no oov values False <class 'str'>
Model contains no oov values False <class 'str'>
```

```
Year contains no oov values True <class 'numpy.int64'>
Engine Fuel Type contains no oov values False <class 'str'>
Engine HP contains no oov values False <class 'numpy.float64'>
Engine Cylinders contains no oov values False <class 'numpy.float64'>
Transmission Type contains no oov values True <class 'str'>
Driven_Wheels contains no oov values True <class 'str'>
Number of Doors contains no oov values False <class 'numpy.float64'>
Market Category contains no oov values False <class 'str'>
Vehicle Size contains no oov values True <class 'str'>
Vehicle Style contains no oov values True <class 'str'>
highway MPG contains no oov values False <class 'numpy.int64'>
city mpg contains no oov values False <class 'numpy.int64'>
Popularity contains no oov values False <class 'numpy.int64'>
MSRP contains no oov values False <class 'numpy.int64'>
```

In [29]:
```
plt.figure(figsize=(15,4))
ax = plt.gca()
ax.hist(df['MSRP'], bins=20, log=True)
ax.set_xscale('log')
# ax.set_xticks([0,10000,50000,100000,200000,500000, 1000000, 1500000, 2000000])
# ax.xticks = (range(8), [10**i for i in range(8)])
# ax.xlim=(0, 50000)
```



- The figure above shows the distribution of MSRP values on log-log scale.
- On the x-axis we can the values lie between **four categories: 0-e4, e4-e5, e5-e6, e6-inf**

In [30]:
```
df['MSRP'].describe()
```

Out[30]:
```
count    1.072200e+04
mean     4.073484e+04
std      6.155895e+04
min      2.000000e+03
25%      2.107625e+04
50%      2.999500e+04
75%      4.230000e+04
```

```
max      2.065902e+06
Name: MSRP, dtype: float64
```

df['Model'].unique()

0.3 MSRP_cat training

- It was found that the **MSRP values** could be divided into four categories **ordinary/deluxe/luxury/super-luxury**
- **Adding that** as a feature **improved the performance of regression model significantly**. It will be shown later.
- Since this **new categorical feature** will be absent in test_data, therefore, **a classifier was trained to predict that**. And fortunately it performed extremely well too.

```
In [31]: def get_msrp_cat(x):
             if x < 10**4:
                 return 'ordinary'
             elif x < 10**5:
                 return 'deluxe'
             elif x < 10**6:
                 return 'luxury'
             else:
                 return 'super-luxury'
```

```
In [32]: df.loc[:, 'MSRP_cat'] = df['MSRP'].apply(get_msrp_cat)
```

Number of Driven Wheels and Vehicle Size are not categorical features Altough I haven't plotted the graph between those and price, which might say against that. Will try that later.

```
In [33]: driven_wheels_map = {'front wheel drive':1,
                              'four wheel drive':2,
                              'rear wheel drive':0,
                              'all wheel drive':2}

         vehicle_size_map = {'Compact':0,
                             'Large':2,
                             'Midsize':1}
```

```
In [34]: df.loc[:, 'Driven_Wheels'] = df['Driven_Wheels'].apply(lambda x: driven_wheels_map[x]
```

```
In [35]: df.loc[:, 'Vehicle Size'] = df['Vehicle Size'].apply(lambda x: vehicle_size_map[x])
```

```
In [36]: X_ = pd.get_dummies(df.drop(['Make', 'Model', 'Engine Fuel Type', 'Market Category',
         y_ = df['MSRP_cat']
```

```
In [37]: y_true = []
         y_pred = []
         X, y = np.array(X_), np.array(y_)
```

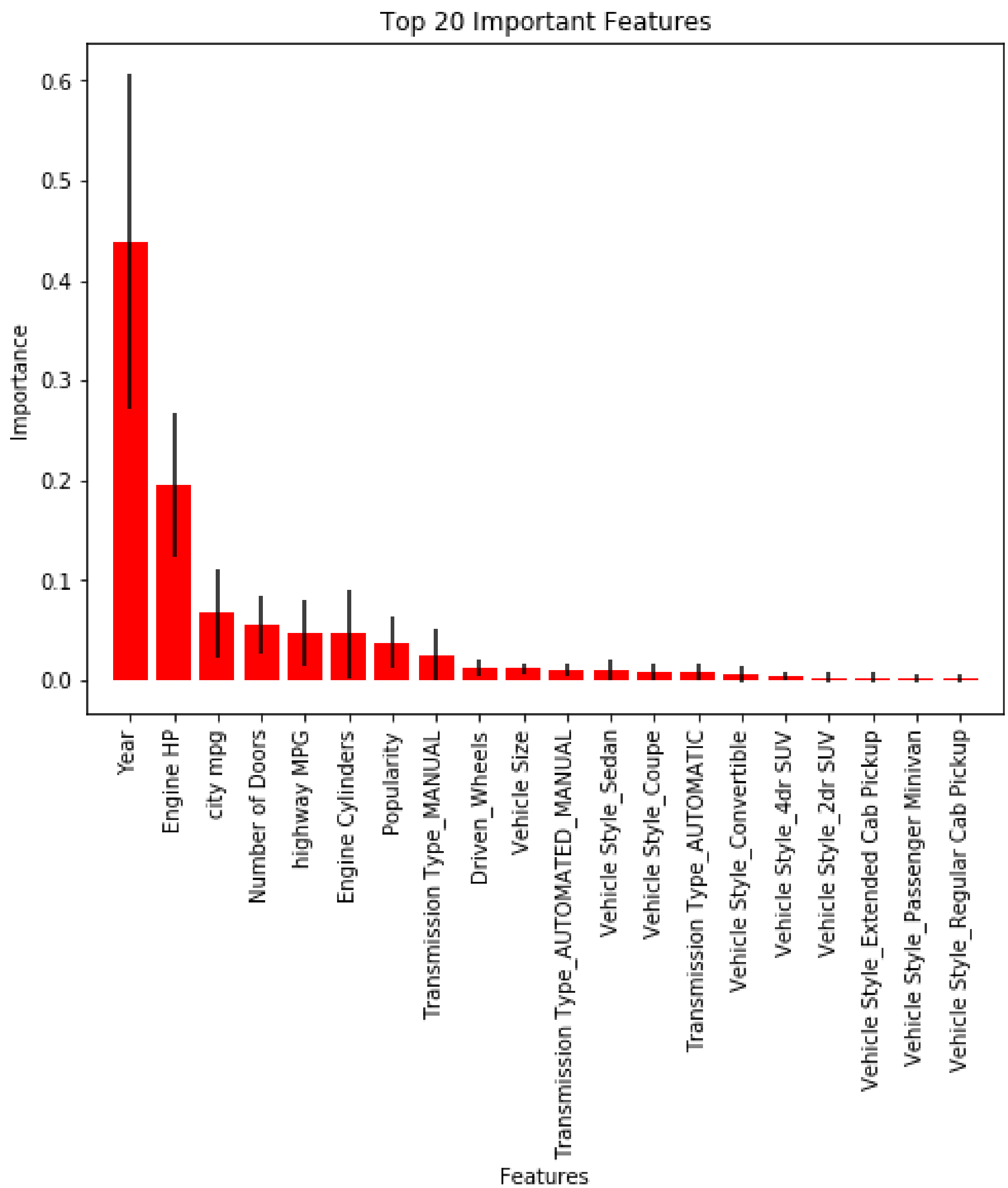
```
    x_scaler = MinMaxScaler()
    y_scaler = MinMaxScaler()
    # x_scaler.fit(X)
    # y_scaler.fit(y.reshape(-1, 1))

    kf = KFold(n_splits=10, shuffle=True, random_state=2019)
    for i, (train_idx, test_idx) in tqdm(enumerate(kf.split(X))):
        X_train, X_test = X[train_idx], X[test_idx]
        y_train, y_test = y[train_idx], y[test_idx]

    #     X_train = x_scaler.transform(X_train)
    #     y_train = y_scaler.transform(y_train.reshape(-1, 1)).reshape(-1)

        model = RandomForestClassifier()
        model.fit(X_train, y_train)

        y_true.append(y_test)
        y_pred.append(model.predict(X_test))
    #     y_pred.append(y_scaler.inverse_transform(model.predict(x_scaler.transform(X_tes

        print('Train acc:', accuracy_score(y_train, model.predict(X_train)), 'F1:', f1_sco
        print('Test acc:', accuracy_score(y_test, model.predict(X_test)), 'F1:', f1_score
    #     print('Test acc:', r2_score(y_test, y_scaler.inverse_transform(model.predict(x_

    y_true = np.concatenate(y_true, axis=0)
    y_pred = np.concatenate(y_pred, axis=0)

    print('\n', 'OOB acc:', accuracy_score(y_true, y_pred))
    print('OOB F1:', f1_score(y_true, y_pred, average='micro'))
```

```
2it [00:00,  7.51it/s]

Train acc: 0.998859985491 F1: 0.998859985491 Test acc: 0.994408201305 F1: 0.994408201305
Train acc: 0.998859985491 F1: 0.998859985491 Test acc: 0.994408201305 F1: 0.994408201305


4it [00:00,  7.48it/s]

Train acc: 0.998756476684 F1: 0.998756476684 Test acc: 0.991604477612 F1: 0.991604477612
Train acc: 0.998445595855 F1: 0.998445595855 Test acc: 0.992537313433 F1: 0.992537313433


6it [00:00,  7.45it/s]

Train acc: 0.998341968912 F1: 0.998341968912 Test acc: 0.996268656716 F1: 0.996268656716
Train acc: 0.998860103627 F1: 0.998860103627 Test acc: 0.993470149254 F1: 0.993470149254


8it [00:01,  7.36it/s]
```

```
Train acc: 0.998860103627 F1: 0.998860103627 Test acc: 0.996268656716 F1: 0.996268656716
Train acc: 0.99896373057 F1: 0.99896373057 Test acc: 0.98973880597 F1: 0.98973880597


10it [00:01,  7.30it/s]

Train acc: 0.998860103627 F1: 0.998860103627 Test acc: 0.995335820896 F1: 0.995335820896
Train acc: 0.998549222798 F1: 0.998549222798 Test acc: 0.995335820896 F1: 0.995335820896

 OOB acc: 0.993937698191
OOB F1: 0.993937698191
```

In [38]:
```python
importances = model.feature_importances_
std = np.std([tree.feature_importances_ for tree in model.estimators_],axis=0)
indices = np.argsort(importances)[::-1]

# Get the feature names
features = X_.columns.values

# Want the top 20 features, so limit the indices and labels
topLimit = 20 # limit to show up to, ex. top 10
indices = indices[0: topLimit] # indices for features
topLabels = features[indices[0: topLimit]] # actual feature labels, we want to print

# Plot the feature importances of the forest (top 20)
figsize = (8,6)
plt.figure(figsize=figsize)
plt.title("Top 20 Important Features")
ax = plt.bar(range(topLimit), importances[indices], color="r", yerr=std[indices], ali
plt.xticks(rotation=90)
plt.xticks(range(topLimit), topLabels)
plt.xlim([-1, topLimit])
plt.xlabel('Features')
plt.ylabel('Importance')
```

Out[38]:
```
Text(0,0.5,'Importance')
```



0.4 Training

```
In [39]: X_ = pd.get_dummies(df.drop(['Make', 'Model', 'Engine Fuel Type', 'Market Category',
         y_ = df['MSRP']
```

```
In [40]: X_.shape
```

```
Out[40]: (10722, 34)
```

In [41]: y_.shape

Out[41]: (10722,)

```
In [42]: y_true = []
         y_pred = []
         X, y = np.array(X_), np.array(y_)

         x_scaler = MinMaxScaler()
         y_scaler = MinMaxScaler()
         # x_scaler.fit(X)
         # y_scaler.fit(y.reshape(-1, 1))

         kf = KFold(n_splits=10, shuffle=True, random_state=2019)
         for i, (train_idx, test_idx) in tqdm(enumerate(kf.split(X))):
             X_train, X_test = X[train_idx], X[test_idx]
             y_train, y_test = y[train_idx], y[test_idx]

         #     X_train = x_scaler.transform(X_train)
         #     y_train = y_scaler.transform(y_train.reshape(-1, 1)).reshape(-1)

             model = RandomForestRegressor()
             model.fit(X_train, y_train)

             y_true.append(y_test)
             y_pred.append(model.predict(X_test))
         #     y_pred.append(y_scaler.inverse_transform(model.predict(x_scaler.transform(X_tes

             print('Train acc:', r2_score(y_train, model.predict(X_train)), end=' ')
             print('Test acc:', r2_score(y_test, model.predict(X_test)))
         #     print('Test acc:', r2_score(y_test, y_scaler.inverse_transform(model.predict(x_

         y_true = np.concatenate(y_true, axis=0)
         y_pred = np.concatenate(y_pred, axis=0)

         print('\n', 'OOB r2:', r2_score(y_true, y_pred))
```

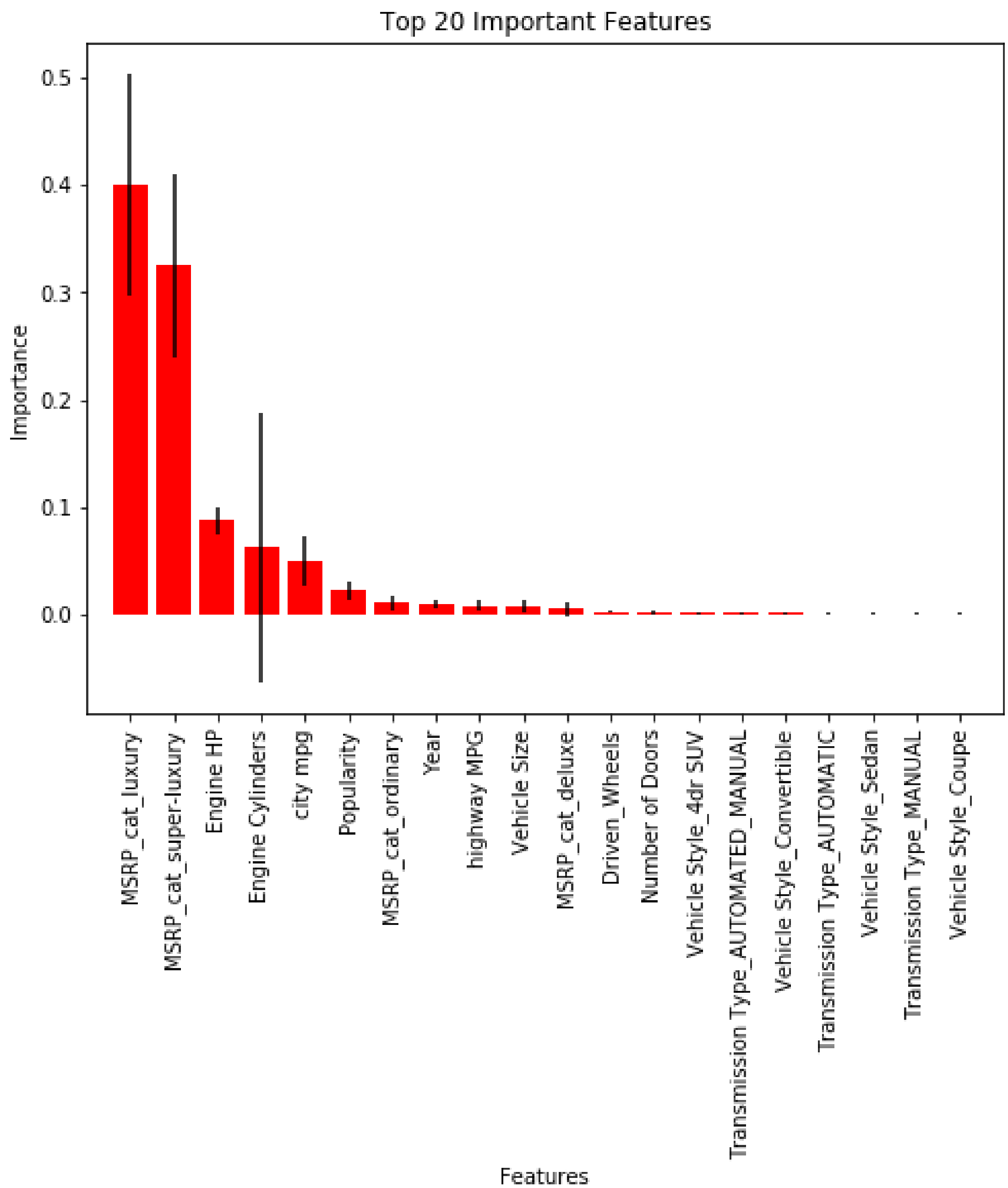
```
1it [00:00,  2.78it/s]

Train acc: 0.990947545823 Test acc: 0.956913594055


2it [00:00,  2.82it/s]

Train acc: 0.989949742526 Test acc: 0.962979146158


3it [00:01,  2.84it/s]
```

```
Train acc: 0.988808075696 Test acc: 0.987849059391


4it [00:01,  2.85it/s]

Train acc: 0.993422275089 Test acc: 0.94166839009


5it [00:01,  2.86it/s]

Train acc: 0.991176010771 Test acc: 0.96334071097


6it [00:02,  2.86it/s]

Train acc: 0.995706999421 Test acc: 0.950227038018


7it [00:02,  2.86it/s]

Train acc: 0.989547461885 Test acc: 0.972394200862


8it [00:02,  2.87it/s]

Train acc: 0.985733747051 Test acc: 0.98134719338


9it [00:03,  2.87it/s]

Train acc: 0.991747764998 Test acc: 0.983213821224


10it [00:03,  2.87it/s]

Train acc: 0.990614416014 Test acc: 0.991336886202

 OOB r2: 0.968773863192
```

In [43]:
```
importances = model.feature_importances_
std = np.std([tree.feature_importances_ for tree in model.estimators_],axis=0)
indices = np.argsort(importances)[::-1]

# Get the feature names
features = X_.columns.values

# Want the top 20 features, so limit the indices and labels
```

```
topLimit = 20 # limit to show up to, ex. top 10
indices = indices[0: topLimit] # indices for features
topLabels = features[indices[0: topLimit]] # actual feature labels, we want to print

# Plot the feature importances of the forest (top 20)
figsize = (8,6)
plt.figure(figsize=figsize)
plt.title("Top 20 Important Features")
ax = plt.bar(range(topLimit), importances[indices], color="r", yerr=std[indices], alig
plt.xticks(rotation=90)
plt.xticks(range(topLimit), topLabels)
plt.xlim([-1, topLimit])
plt.xlabel('Features')
plt.ylabel('Importance')
```

Out[43]: Text(0,0.5,'Importance')

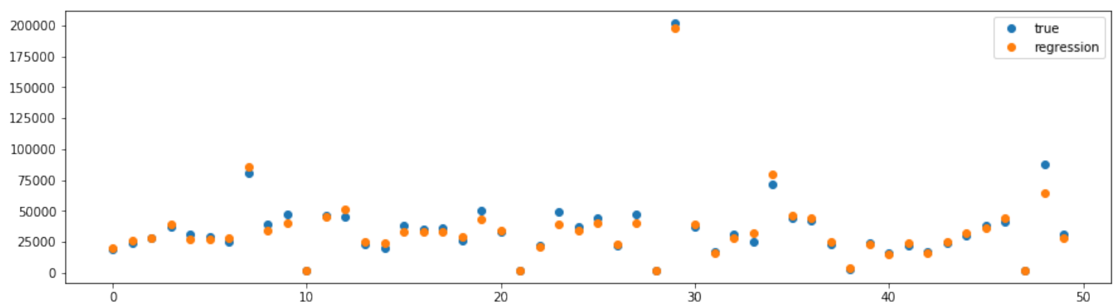


- Now we can **remove unimportant features** like Transmission Type, Vehicle Style, Driven Wheels, Number of Doors
- Now it is evident from the above plot that **deducing MSRP categories first was critical for predicting exact MSRP values**.

```
In [44]: y_true[:10]
```

```
Out[44]: array([19100, 24599, 27880, 36850, 31180, 29470, 25045, 81013, 39280, 47095])
```

```
In [45]: y_pred[:10]

Out[45]: array([ 19650.        ,  26606.73989899,  27748.25      ,  38772.        ,
                 27411.35833333,  26644.825     ,  28209.10833333,  85841.4       ,
                 34352.11287879,  39989.16666667])

In [46]: plt.figure(figsize=(15,4))
         plt.plot(y_true[:50], 'o', label='true')
         plt.plot(y_pred[:50], 'o', label='regression')

         plt.legend()

Out[46]: <matplotlib.legend.Legend at 0x7fcb63d534a8>
```



1 Final Feature Selection and Model Training

- Unimportant features will be removed and bigger models will be trained to obtain maximum performance.

1.1 MSRP_cat training

Number of Driven Wheels and Vehicle Size are not categorical features Altough I haven't plotted the graph between those and price, which might say against that. Will try that later.

```
In [47]: X_ = pd.get_dummies(df.drop(['Make', 'Model', 'Engine Fuel Type', 'Market Category','I
         y_ = df['MSRP_cat']

In [48]: y_true = []
         y_pred = []
         X, y = np.array(X_), np.array(y_)

         kf = KFold(n_splits=10, shuffle=True, random_state=2019)
         for i, (train_idx, test_idx) in tqdm(enumerate(kf.split(X))):
             X_train, X_test = X[train_idx], X[test_idx]
             y_train, y_test = y[train_idx], y[test_idx]
```

```
            clf = RandomForestClassifier()
            clf.fit(X_train, y_train)

            y_true.append(y_test)
            y_pred.append(clf.predict(X_test))

            print('Train acc:', accuracy_score(y_train, clf.predict(X_train)), 'F1:', f1_score
            print('Test acc:', accuracy_score(y_test, clf.predict(X_test)), 'F1:', f1_score(y_

        y_true = np.concatenate(y_true, axis=0)
        y_pred = np.concatenate(y_pred, axis=0)

        print('\n', 'OOB acc:', accuracy_score(y_true, y_pred))
        print('OOB F1:', f1_score(y_true, y_pred, average='micro'))
```

```
2it [00:00,  7.11it/s]

Train acc: 0.998963623173 F1: 0.998963623173 Test acc: 0.994408201305 F1: 0.994408201305
Train acc: 0.999067260856 F1: 0.999067260856 Test acc: 0.994408201305 F1: 0.994408201305


4it [00:00,  7.33it/s]

Train acc: 0.99896373057 F1: 0.99896373057 Test acc: 0.992537313433 F1: 0.992537313433
Train acc: 0.998756476684 F1: 0.998756476684 Test acc: 0.995335820896 F1: 0.995335820896


6it [00:00,  7.34it/s]

Train acc: 0.998549222798 F1: 0.998549222798 Test acc: 0.993470149254 F1: 0.993470149254
Train acc: 0.998860103627 F1: 0.998860103627 Test acc: 0.992537313433 F1: 0.992537313433


8it [00:01,  7.38it/s]

Train acc: 0.99896373057 F1: 0.99896373057 Test acc: 0.993470149254 F1: 0.993470149254
Train acc: 0.999067357513 F1: 0.999067357513 Test acc: 0.988805970149 F1: 0.988805970149


10it [00:01,  7.42it/s]

Train acc: 0.998756476684 F1: 0.998756476684 Test acc: 0.995335820896 F1: 0.995335820896
Train acc: 0.998860103627 F1: 0.998860103627 Test acc: 0.994402985075 F1: 0.994402985075

 OOB acc: 0.993471367282
OOB F1: 0.993471367282
```

1.2 Training

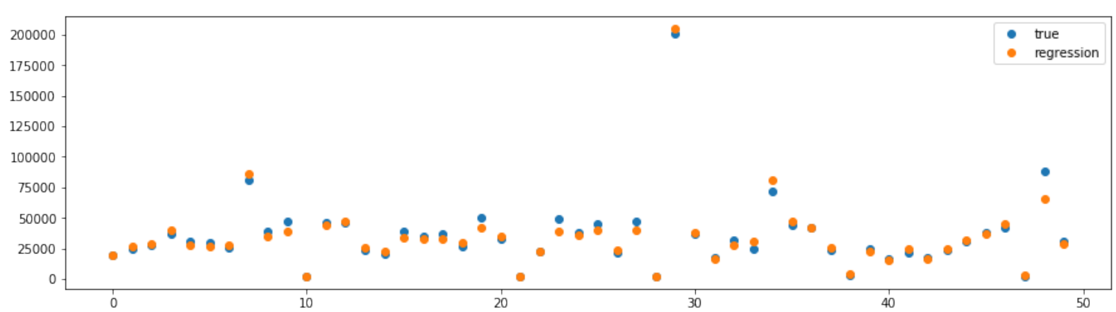
```
In [49]: X_ = pd.get_dummies(df.drop(['Make', 'Model', 'Engine Fuel Type', 'Market Category',
         y_ = df['MSRP']

In [50]: y_true = []
         y_pred = []
         X, y = np.array(X_), np.array(y_)

         kf = KFold(n_splits=10, shuffle=True, random_state=2019)
         for i, (train_idx, test_idx) in tqdm(enumerate(kf.split(X))):
             X_train, X_test = X[train_idx], X[test_idx]
             y_train, y_test = y[train_idx], y[test_idx]

             model = RandomForestRegressor(n_estimators=50, max_features=6)
             model.fit(X_train, y_train)

             y_true.append(y_test)
             y_pred.append(model.predict(X_test))

             print('Train acc:', r2_score(y_train, model.predict(X_train)), end=' ')
             print('Test acc:', r2_score(y_test, model.predict(X_test)))

         y_true = np.concatenate(y_true, axis=0)
         y_pred = np.concatenate(y_pred, axis=0)

         print('\n', 'OOB r2:', r2_score(y_true, y_pred))
```

```
1it [00:00,  1.33it/s]

Train acc: 0.991396208597 Test acc: 0.970642747167


2it [00:01,  1.38it/s]

Train acc: 0.991244022577 Test acc: 0.971206194871


3it [00:02,  1.38it/s]

Train acc: 0.989823553108 Test acc: 0.986251016968


4it [00:02,  1.39it/s]

Train acc: 0.993191226895 Test acc: 0.957448803314


5it [00:03,  1.39it/s]
```

```
Train acc: 0.990877791868 Test acc: 0.968551008058


6it [00:04,  1.38it/s]

Train acc: 0.994052160286 Test acc: 0.936948329456


7it [00:05,  1.38it/s]

Train acc: 0.989706594561 Test acc: 0.986295515136


8it [00:05,  1.38it/s]

Train acc: 0.990503570166 Test acc: 0.988657054439


9it [00:06,  1.38it/s]

Train acc: 0.991153688946 Test acc: 0.981890308648


10it [00:07,  1.38it/s]

Train acc: 0.989950754841 Test acc: 0.98952001157

 OOB r2: 0.971590735221
```

In [51]:
```python
plt.figure(figsize=(15,4))
plt.plot(y_true[:50], 'o', label='true')
plt.plot(y_pred[:50], 'o', label='regression')

plt.legend()
```

Out[51]: <matplotlib.legend.Legend at 0x7fcb63ca2668>



1.3 Submission

```
In [52]: dft_orig = pd.read_csv('test_data.csv')
         dft = dft_orig.copy()
         dft.loc[:, 'Driven_Wheels'] = dft['Driven_Wheels'].apply(lambda x: driven_wheels_map[x
         dft.loc[:, 'Vehicle Size'] = dft['Vehicle Size'].apply(lambda x: vehicle_size_map[x])

         dft.loc[dft['Number of Doors'].isnull(), 'Number of Doors'] = 4
         dft.loc[dft['Engine HP'].isnull(), 'Engine HP'] = ENGINE_HP_MEAN
         dft.loc[dft['Engine Cylinders'].isnull(), 'Engine Cylinders'] = 0

         print(dft.isnull().sum())

         X_ = pd.get_dummies(dft.drop(['Make', 'Model', 'Engine Fuel Type', 'Market Category',
         dft.loc[:, 'MSRP_cat'] = clf.predict(np.array(X_))

         X_ = pd.get_dummies(dft.drop(['Make', 'Model', 'Engine Fuel Type', 'Market Category',
         X_.loc[:, 'MSRP_cat_super-luxury'] = 0

         dft_orig['MSRP'] = model.predict(X_).astype('int').reshape(-1)
```

```
Make                    0
Model                   0
Year                    0
Engine Fuel Type        0
Engine HP               0
Engine Cylinders        0
Transmission Type       0
Driven_Wheels           0
Number of Doors         0
Market Category       366
Vehicle Size            0
Vehicle Style           0
highway MPG             0
city mpg                0
Popularity              0
MSRP                 1192
dtype: int64
```

```
In [53]: dft_orig.to_csv('submission.csv', index=False, float_format='%d')
```

```
In [ ]:
```