ЛР №1. Анализ данных, Линейная регрессия (Ру 2.7)

Оглавление

---

1. Знакомство с инструментами

1.1. Горячие клавиши

Изменить масштаб страницы:

- Ctrl + "+ / -"
- Ctrl + "колесо мышки вверх/вниз"

Строки Jupyter Notebook:

- "Enter" - редактировать выделенную
- Shift + "Enter" - выполнить выделенную, выделить следующую
- Alt + "Enter" - выполнить выделенную, добавить новую ниже
- Ctrl + "Enter" - выполнить выделенную

В режиме редактирования строки:

- Shift + TAB - Вызов встроенной справки
- TAB - автодополнение строки

1.2. Библиотеки

Перед началом работы убедитесь, что необходимые для работы пакеты установлены в системе. Для установки отсутствующих используйте команду:

- pip install numpy scipy pandas sklearn seaborn matplotlib
- pip install pandas pandoc pydot pydot-ng pydotplus
- pip install jupyter jupyter_nbextensions_configurator jupyter_contrib_nbextensions

- pandas - обработка (загрузка, сохранение, анализа) данных
- seaborn - визуализация данных (на базе matplotlib)
- sklearn - (классификация, регрессия, кластеризация...)
- numpy - обработка многомерных массивов, линейная алгебра, преобразование Фурье, случайные числа
- scipy - пакет для выполнения научных и инженерных расчётов.

In [248]:
```
### Example.Python2.7

import pandas as pd
import pylab as pl
import seaborn as sns
from scipy import stats

from sklearn import linear_model
from sklearn.model_selection import train_test_split
from sklearn.model_selection import cross_val_score

from sklearn.metrics import mean_squared_error, r2_score, mean_absolute_error

pd.set_option('precision', 3)
pd.set_option('display.float_format', lambda x: '%.3f' % x)
sns.set(font_scale=1) # размер шрифта

%pylab inline
```

```
Populating the interactive namespace from numpy and matplotlib
```

1.3. Загрузка данных

Описание входных данных

Данные "Melbourne Housing Snapshot" содержат информацию о продажах недвижимости в г. Мельбурн (Австралия)

- Suburb: окрестности, пригород
- Rooms: кол-во комнат
- Price: цена
- Method:
  - S - property sold; SP - property sold prior;
  - PI - property passed in; PN - sold prior not disclosed;
  - SN - sold not disclosed; NB - no bid;
  - VB - vendor bid; W - withdrawn prior to auction;
  - SA - sold after auction; SS - sold after auction price not disclosed.
  - N/A - price or highest bid not available.
- Type:
  - br - bedroom(s);
  - h - house,cottage,villa, semi,terrace;
  - u - unit, duplex;
  - t - townhouse;
  - dev site - development site;
  - o res - other residential.
- SellerG: агент по недвижимости
- Date: дата продажи
- Distance: Расстояние до центрального делового района CBD (central business district)
- Regionname: Район
- Propertycount: Number of properties that exist in the suburb.
- Bedroom2 : кол-во спален
- Bathroom: кол-во ванных
- Car: кол-во парковочных мест
- Landsize: Land Size
- BuildingArea: Building Size
- CouncilArea: Управляющая компания

Для загрузки табличных данных из файла применятся функция **read_csv()** с указанием параметров (путь, кодировка, разделитель, обработка колонок с заголовками или датой...)

Загрузка входных данных

In [249]:
```
data = pd.read_csv('./melbourne-housing-snapshot/melb_data.csv', # путь к файлу, используй автодотолнение
                   sep=',', # разделитель данных в файле
                   header=0 # номер строки с заголовками. header='None', если заголовки отсутствуют
                  )
```

Вывести первые 4 строки в таблице

In [250]:
```
data['Price'].map('{:.2f}'.format)[:10]
```

Out[250]:
```
0    1480000.00
1    1035000.00
2    1465000.00
3     850000.00
4    1600000.00
5     941000.00
6    1876000.00
7    1636000.00
8     300000.00
9    1097000.00
Name: Price, dtype: object
```

In [251]:
```
data.head(4)
```

Out[251]:

	Suburb	Address	Rooms	Type	Price	Method	SellerG	Date	Distance	Postcode	...	Bathroom	Car	Landsize	Buildin
0	Abbotsford	85 Turner St	2	h	1480000.000	S	Biggin	3/12/2016	2.500	3067.000	...	1.000	1.000	202.000	
1	Abbotsford	25 Bloomburg St	2	h	1035000.000	S	Biggin	4/02/2016	2.500	3067.000	...	1.000	0.000	156.000	7
2	Abbotsford	5 Charles St	3	h	1465000.000	SP	Biggin	4/03/2017	2.500	3067.000	...	2.000	0.000	134.000	15
3	Abbotsford	40 Federation La	3	h	850000.000	PI	Biggin	4/03/2017	2.500	3067.000	...	2.000	1.000	94.000	

4 rows × 21 columns

In [252]:
```
type(data)
```

Out[252]:
```
pandas.core.frame.DataFrame
```

Переменная "data" является экземпляром класса "DataFrame" - основной структуры данных в библиотеке "pandas". "DataFrame" - двумерный массив с изменяемым размером. Каждый столбец является экземпляром класса "Series". "Series"- одномерный индексированный массив определенного типа. Строки соответствуют объектам. Столбецы - соответствуют признакам объектов.

DataFrame

Series	Series	Series	Series
int	string	float	bool
int	string	float	bool
int	string	float	bool

In [253]:
```
data [:5]
```

Out[253]:

	Suburb	Address	Rooms	Type	Price	Method	SellerG	Date	Distance	Postcode	...	Bathroom	Car	Landsize	Buildin
0	Abbotsford	85 Turner St	2	h	1480000.000	S	Biggin	3/12/2016	2.500	3067.000	...	1.000	1.000	202.000	
1	Abbotsford	25 Bloomburg St	2	h	1035000.000	S	Biggin	4/02/2016	2.500	3067.000	...	1.000	0.000	156.000	7
2	Abbotsford	5 Charles St	3	h	1465000.000	SP	Biggin	4/03/2017	2.500	3067.000	...	2.000	0.000	134.000	15
3	Abbotsford	40 Federation La	3	h	850000.000	PI	Biggin	4/03/2017	2.500	3067.000	...	2.000	1.000	94.000	
4	Abbotsford	55a Park St	4	h	1600000.000	VB	Nelson	4/06/2016	2.500	3067.000	...	1.000	2.000	120.000	14

5 rows × 21 columns

1.4. Выбор строк и столбцов

Получим список всех столбцов

In [254]:
```
data.columns
```

Out[254]:
```
Index([u'Suburb', u'Address', u'Rooms', u'Type', u'Price', u'Method',
       u'SellerG', u'Date', u'Distance', u'Postcode', u'Bedroom2', u'Bathroom',
       u'Car', u'Landsize', u'BuildingArea', u'YearBuilt', u'CouncilArea',
       u'Lattitude', u'Longtitude', u'Regionname', u'Propertycount'],
      dtype='object')
```

В случае, если названия столбцов отсутствуют, либо требуется их изменить, то полю data.columns следует присвоить новый список(list) из строк.

In [255]:
```
data.columns = ([u'Suburb', u'Address', u'Rooms', u'Type', u'Price', u'Method',
       u'SellerG', u'Date', u'Distance', u'Postcode', u'Bedroom2', u'Bathroom',
       u'Car', u'Landsize', u'BuildingArea', u'YearBuilt', u'CouncilArea',
       u'Lattitude', u'Longtitude', u'Regionname', u'Propertycount'])
```

Объект DataFrame допускает частичный выбор строк и колонок.

In [256]:
```
# data['столбец'][строка]
data['Address'][100]
```

Out[256]:
```
'86 Mills St'
```

In [257]:
```
# data.Имя_столбца [начальный инд. строки : конечный инд. строки]

data.Lattitude [:5]
```

Out[257]:
```
0   -37.800
1   -37.808
2   -37.809
3   -37.797
4   -37.807
Name: Lattitude, dtype: float64
```

In [258]:
```
# Выберем для строк с 10 по 15 значения столбцов с именами 'Price', 'Rooms', 'Address'
# data[ инд./'название' столбца1, ...][начальный инд. строки : конечный инд. строки]

data[ ['Price','Rooms','Address'] ][10:20]
```

Out[258]:

	Price	Rooms	Address
10	700000.000	2	411/8 Grosvenor St
11	1350000.000	3	40 Nicholson St
12	750000.000	2	123/56 Nicholson St
13	1172500.000	2	45 William St
14	441000.000	1	7/20 Abbotsford St
15	1310000.000	2	16 William St
16	1200000.000	3	42 Henry St
17	1176500.000	3	78 Yarra St
18	955000.000	3	196 Nicholson St
19	890000.000	2	42 Valiant St

Для получения отдельного столбца (тип данных "Series") необходимо запросить единственный столбец по его имени или индексу

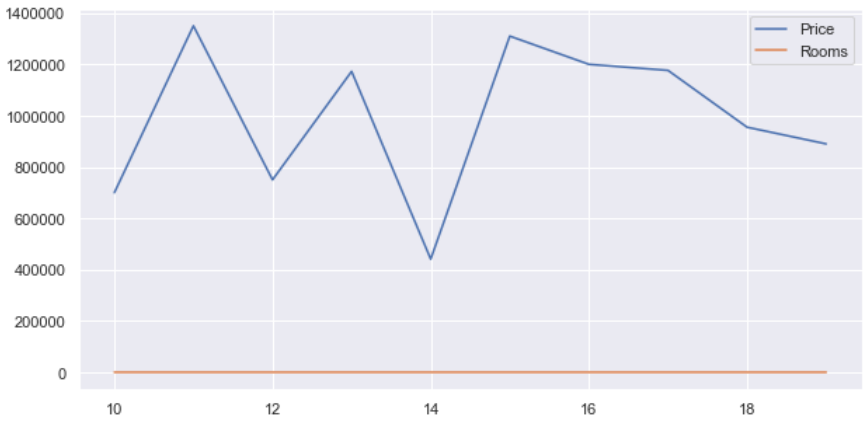
In [259]:
```
print  "{:27} is {}".format( "data[ \'Rooms\' ]", type(data[['Rooms']]) )
print  "{:27} is {}".format( "data[ [\'Rooms\'] ]", type(data['Rooms']) )
print  "{:27} is {}".format( "data[ [\'Rooms\', \'Rooms\'] ]", type(data[['Rooms','Rooms']]) )
```

```
data[ 'Rooms' ]             is <class 'pandas.core.frame.DataFrame'>
data[ ['Rooms'] ]           is <class 'pandas.core.series.Series'>
data[ ['Rooms', 'Rooms'] ]  is <class 'pandas.core.frame.DataFrame'>
```

Для построения простейшего графика достаточно добавить к конструкции выше команду **.plot()**

In [260]:
```
data[ ['Price','Rooms','Address'] ][10:20].plot(
    figsize=(10,5) # размер графка
)
```

Out[260]:
```
<matplotlib.axes._subplots.AxesSubplot at 0x20f9e2b0>
```



Для объекта "Series" используем метод **value_counts()** для подсчета количества уникальных значений

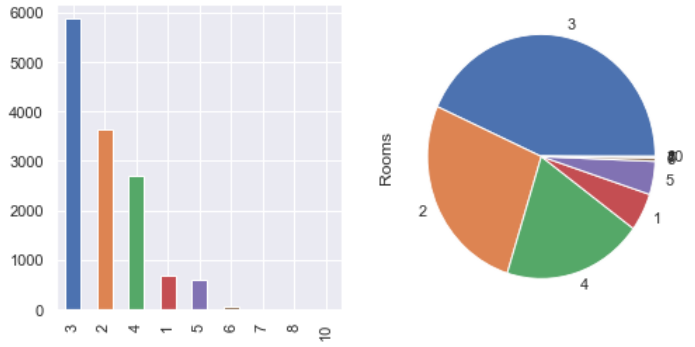
In [261]:
```
data['Rooms'] .value_counts()
```

Out[261]:
```
3     5881
2     3648
4     2688
1      681
5      596
6       67
7       10
8        8
10       1
Name: Rooms, dtype: int64
```

Для наглядного отображения построим пару графиков и разместим их горизонтально рядом друг с другом:

- Столбчатую диаграмму, командой **.plot(kind='bar')**
- Круговую диаграмму, командой **.plot(kind='pie')**

In [262]:
```
# https://pandas.pydata.org/pandas-docs/stable/user_guide/visualization.html
plt.figure(figsize=(8,4)) # создать полотно для графиков с заданным размером

plt.subplot(121) # 121 = 1,2,1 - кол. строк, кол. столбцов, индекс графика
data['Rooms'].value_counts().plot(kind='bar')

plt.subplot(122) # 122 = 1,2,2 - кол. строк, кол. столбцов, индекс графика
data['Rooms'].value_counts().plot(kind='pie', fontsize=11)

plt.show()
```



Для объединения столбцов в новый DataFrame используем **pd.concat()**

In [263]:
```
pd.concat(
    {
        "Rooms_Сырые": data['Rooms'].value_counts(normalize=False),
        "Rooms_Нормализованные": data['Rooms'].value_counts(normalize=True)
    },
    axis=1 # axis - способ объединения 0 - как строки, 1 - как столбцы
)
```

Out[263]:

	Rooms_Нормализованные	Rooms_Сырые
3	0.433	5881
2	0.269	3648
4	0.198	2688
1	0.050	681
5	0.044	596
6	0.005	67
7	0.001	10
8	0.001	8
10	0.000	1

In [264]:
```
data['Rooms'].value_counts(normalize=True).sum()
```

Out[264]:
```
1.0
```

Фильтрация строк

К объектам-массивам "Series" можно применить условный оператор, результатом станет массив с значениями True/False для каждой строки.

In [265]:
```
type(data['Rooms'])
```

Out[265]:
```
pandas.core.series.Series
```

In [266]:
```
# Проверить, выполняется ли условие  "== 3" для значений признака 'Rooms'
(data['Rooms'] == 3) [10:20]
```

Out[266]:
```
10    False
11     True
12    False
13    False
14    False
15    False
16     True
17     True
18     True
19    False
Name: Rooms, dtype: bool
```

Для получения конкретных строк, для которых выполняется заданное условие следует подставить полученный массив.

In [267]:
```
data[ data['Rooms'] == 3 ]
```

Out[267]:

	Suburb	Address	Rooms	Type	Price	Method	SellerG	Date	Distance	Postcode	...	Bathroom	Car	Lar
2	Abbotsford	5 Charles St	3	h	1465000.000	SP	Biggin	4/03/2017	2.500	3067.000	...	2.000	0.000	13
3	Abbotsford	40 Federation La	3	h	850000.000	PI	Biggin	4/03/2017	2.500	3067.000	...	2.000	1.000	9
6	Abbotsford	124 Yarra St	3	h	1876000.000	S	Nelson	7/05/2016	2.500	3067.000	...	2.000	0.000	24
11	Abbotsford	40 Nicholson St	3	h	1350000.000	VB	Nelson	12/11/2016	2.500	3067.000	...	2.000	2.000	21
16	Abbotsford	42 Henry St	3	h	1200000.000	S	Jellis	16/07/2016	2.500	3067.000	...	2.000	1.000	11
17	Abbotsford	78 Yarra St	3	h	1176500.000	S	LITTLE	16/07/2016	2.500	3067.000	...	1.000	1.000	13
18	Abbotsford	196 Nicholson St	3	h	955000.000	S	Collins	17/09/2016	2.500	3067.000	...	1.000	0.000	18
21	Abbotsford	13/11 Nicholson St	3	t	900000.000	S	Beller	18/03/2017	2.500	3067.000	...	2.000	2.000	
22	Abbotsford	138/56 Nicholson St	3	u	1090000.000	S	Jellis	18/03/2017	2.500	3067.000	...	2.000	2.000	429
27	Abbotsford	48 Abbotsford St	3	h	1447500.000	PI	Nelson	22/08/2016	2.500	3067.000	...	3.000	1.000	16
31	Abbotsford	166 Gipps St	3	h	1290000.000	S	Biggin	25/02/2017	2.500	3067.000	...	2.000	2.000	14
32	Abbotsford	60 Stafford St	3	h	1290000.000	S	Biggin	25/02/2017	2.500	3067.000	...	1.000	1.000	16
35	Abbotsford	45 Yarra St	3	h	1195000.000	SP	Jellis	27/11/2016	2.500	3067.000	...	2.000	1.000	12
38	Airport West	154 Halsey Rd	3	t	840000.000	PI	Nelson	3/09/2016	13.500	3042.000	...	2.000	1.000	30
39	Airport West	50 Bedford St	3	h	730000.000	VB	Nelson	3/12/2016	13.500	3042.000	...	2.000	1.000	
40	Airport West	50 Bedford St	3	h	770000.000	SP	Nelson	4/03/2017	13.500	3042.000	...	2.000	1.000	
42	Airport West	1/80 Hawker St	3	t	700000.000	S	Brad	4/03/2017	13.500	3042.000	...	2.000	2.000	23
43	Airport West	1/37 Hillside Gr	3	h	600000.000	S	Maddison	4/03/2017	13.500	3042.000	...	1.000	1.000	29
45	Airport West	54 Marshall Rd	3	h	720000.000	S	Barry	6/08/2016	13.500	3042.000	...	1.000	1.000	62
51	Airport West	3 Deidre Ct	3	h	895000.000	PI	Rendina	10/09/2016	13.500	3042.000	...	1.000	6.000	106
52	Airport West	13 Etzel St	3	h	805000.000	S	Nelson	10/12/2016	13.500	3042.000	...	1.000	1.000	
53	Airport West	1/43 Cameron St	3	u	752000.000	S	Nelson	11/02/2017	13.500	3042.000	...	1.000	1.000	26
56	Airport West	2/40 Earl St	3	t	700000.000	PI	Brad	15/10/2016	13.500	3042.000	...	2.000	2.000	16
57	Airport West	180 Parer Rd	3	h	830000.000	S	Barry	16/04/2016	13.500	3042.000	...	1.000	2.000	97
59	Airport West	43 Hillside Gr	3	h	670000.000	SP	Barry	16/07/2016	13.500	3042.000	...	2.000	2.000	65
61	Airport West	138 Victory Rd	3	h	1042000.000	S	Nelson	16/07/2016	13.500	3042.000	...	2.000	5.000	61
62	Airport West	75 King St	3	h	910000.000	S	McDonald	17/09/2016	13.500	3042.000	...	1.000	1.000	77
64	Airport West	478 Fullarton Rd	3	h	810000.000	PI	Barry	18/03/2017	13.500	3042.000	...	3.000	2.000	55
65	Airport West	144 Marshall Rd	3	h	715000.000	S	Rendina	18/03/2017	13.500	3042.000	...	2.000	2.000	42
66	Airport West	106 Parer Rd	3	h	790000.000	S	Nelson	18/03/2017	13.500	3042.000	...	1.000	3.000	61
...	...	...	...	...	...	...	...	...	...	...	...	...	...	
13505	Mulgrave	2229 Dandenong Rd	3	h	940000.000	S	Biggin	26/08/2017	18.800	3170.000	...	2.000	3.000	70

	Suburb	Address	Rooms	Type	Price	Method	SellerG	Date	Distance	Postcode	...	Bathroom	Car	Lar
13506	Niddrie	3 Grandview Rd	3	h	845000.000	SP	Brad	26/08/2017	10.400	3042.000	...	1.000	1.000	34
13507	Niddrie	44 Shaw St	3	h	1305000.000	SP	Frank	26/08/2017	10.400	3042.000	...	1.000	3.000	65
13514	Ormond	57 Holloway St	3	h	1510000.000	S	Buxton	26/08/2017	11.400	3204.000	...	2.000	3.000	54
13517	Parkdale	14 Robert St	3	h	1060000.000	S	Barry	26/08/2017	21.500	3195.000	...	1.000	2.000	60
13525	Preston	61 Dundas St	3	h	910000.000	SP	Nelson	26/08/2017	8.400	3072.000	...	1.000	2.000	27
13526	Reservoir	1E Black St	3	t	640000.000	PI	Ray	26/08/2017	12.000	3073.000	...	1.000	1.000	22
13529	Reservoir	4 Newton St	3	h	800000.000	PI	Ray	26/08/2017	12.000	3073.000	...	1.000	2.000	58
13532	Ringwood	115 Loughnan Rd	3	h	700000.000	S	Ray	26/08/2017	19.900	3134.000	...	2.000	2.000	60
13534	Rosanna	179 Beverley Rd	3	h	1224000.000	S	Nelson	26/08/2017	8.900	3084.000	...	1.000	2.000	60
13539	Scoresby	1 millgrove St	3	h	890000.000	S	Ray	26/08/2017	22.200	3179.000	...	1.000	2.000	72
13540	Seaford	137 East Rd	3	h	512000.000	S	hockingstuart	26/08/2017	35.400	3198.000	...	1.000	1.000	57
13542	Spotswood	20 Watt St	3	h	965000.000	S	RT	26/08/2017	6.200	3015.000	...	1.000	2.000	47
13544	Strathmore	4 Collegian Av	3	h	1848000.000	S	McDonald	26/08/2017	8.200	3041.000	...	1.000	3.000	82
13545	Sunbury	31 Barkly St	3	h	516000.000	S	Raine	26/08/2017	31.700	3429.000	...	2.000	3.000	58
13546	Sunbury	7 Fullbrook Dr	3	h	480000.000	PI	Leeburn	26/08/2017	31.700	3429.000	...	2.000	2.000	64
13548	Sunbury	64 Stewarts La	3	h	605000.000	S	One	26/08/2017	31.700	3429.000	...	2.000	2.000	75
13549	Sunshine	9 Appleby Ct	3	h	717500.000	S	Barry	26/08/2017	10.500	3020.000	...	1.000	1.000	55
13550	Sunshine North	5 Berkshire Rd	3	h	595000.000	SP	Barry	26/08/2017	10.500	3020.000	...	1.000	nan	58
13552	Sunshine West	16 Mudford St	3	h	640000.000	S	Bells	26/08/2017	10.500	3020.000	...	1.000	1.000	61
13554	Surrey Hills	46 Durham Rd	3	h	1715000.000	S	Noel	26/08/2017	10.200	3127.000	...	1.000	2.000	43
13556	Tarneit	27 McMahon Cr	3	h	350000.000	VB	S&L	26/08/2017	18.400	3029.000	...	1.000	1.000	46
13559	Templestowe Lower	16 Benambra Dr	3	h	1190000.000	S	hockingstuart	26/08/2017	12.400	3107.000	...	1.000	1.000	72
13562	Thornbury	7 Ballantyne St	3	h	1450000.000	S	Woodards	26/08/2017	7.000	3071.000	...	1.000	1.000	31
13563	Thornbury	201 Gooch St	3	h	1271000.000	S	Nelson	26/08/2017	7.000	3071.000	...	1.000	2.000	47
13564	Tullamarine	7 Londrew Ct	3	h	540000.000	S	Barry	26/08/2017	12.900	3043.000	...	1.000	1.000	60
13570	Wantirna South	34 Fewster Dr	3	h	970000.000	S	Barry	26/08/2017	14.700	3152.000	...	2.000	2.000	67
13574	Westmeadows	9 Black St	3	h	582000.000	S	Red	26/08/2017	16.500	3049.000	...	2.000	2.000	25
13576	Williamstown	77 Merrett Dr	3	h	1031000.000	SP	Williams	26/08/2017	6.800	3016.000	...	2.000	2.000	33
13577	Williamstown	83 Power St	3	h	1170000.000	S	Raine	26/08/2017	6.800	3016.000	...	2.000	4.000	43

5881 rows × 21 columns

1.5. Группировка

Объекты "DataFrame" позволяет группировать по одному или нескольким столбцам и применять к группированным данным агрегатные функции (sum, mean, median, count, max, min, ...)

In [268]:
```
data.groupby(['Suburb','Rooms'])['Price'].min() [0:20]
```

Out[268]:
```
Suburb        Rooms
Abbotsford    1        300000.000
              2        480000.000
              3        850000.000
              4       1330000.000
Aberfeldie    1        280000.000
              2        373000.000
              3        726000.000
              4       1150000.000
              5       1472000.000
Airport West  2        440000.000
              3        510000.000
              4        765000.000
              5        755000.000
              6        725000.000
Albanvale     2        415000.000
              3        506000.000
              4        655000.000
Albert Park   1        442500.000
              2        647000.000
              3       1300000.000
Name: Price, dtype: float64
```

In [269]:
```
data.groupby(['Rooms','Suburb'])['Price'].min() [0:10]
```

Out[269]:
```
Rooms  Suburb
1      Abbotsford       300000.000
       Aberfeldie       280000.000
       Albert Park      442500.000
       Albion           145000.000
       Alphington       316000.000
       Armadale         280000.000
       Ascot Vale       390000.000
       Balaclava        280000.000
       Balwyn           390000.000
       Bentleigh East   370000.000
Name: Price, dtype: float64
```

1.6. Замена значений

Для регрессионной модели необходимо, чтобы используемые значения были числовыми. Но в таблицах часто встречаются признаки строкового типа (Названия городов, видов, моделей и т.д.). Один из способов передать эти признаки в обработку - это заменить из числовыми значениями. Применительно к "DataFrame" и "Series" доступны методы **replace()** и **map()**

In [270]:
```
# Создим тестовый объект DataFrame
ddf = pd.DataFrame (
    {
        u"День недели":["Пн","Вт","Ср","Чт", "Пт", "Сб", "Вс"],
        u"Температура":[20,20,30,10,10,20,30],
    }
)
print (ddf)
```

```
  День недели  Температура
0          Пн           20
1          Вт           20
2          Ср           30
3          Чт           10
4          Пт           10
5          Сб           20
6          Вс           30
```

In [271]:
```
# Составим словарь для замен - Ключ:Значение
day_dict = {"Пн":1,"Вт":2,"Ср":3,"Чт":4, "Пт":5, "Сб":6, "Вс":7}

# Используем метод replace() для отдельного столбца
ddf[u"День недели"].replace(day_dict)
```

Out[271]:
```
0    1
1    2
2    3
3    4
4    5
5    6
6    7
Name: День недели, dtype: int64
```

In [272]:
```
# Получим уникальные значения из отдельного столбца
ddf[u"Температура"].unique()
```

Out[272]:
```
array([20, 30, 10], dtype=int64)
```

In [273]:
```
# Составим словарь для замен - Ключ:Значение
weather_dict = {20: "Тепло", 30: "Жарко", 10: "Холодно", }

ddf[u"Температура"].map(weather_dict)
```

Out[273]:
```
0      Тепло
1      Тепло
2      Жарко
3    Холодно
4    Холодно
5      Тепло
6      Жарко
Name: Температура, dtype: object
```

In [274]:
```
# Используем метод map()
ddf[u"Температура"].map("+{} t Co".format)
```

Out[274]:
```
0    +20 t Co
1    +20 t Co
2    +30 t Co
3    +10 t Co
4    +10 t Co
5    +20 t Co
6    +30 t Co
Name: Температура, dtype: object
```

2. Формирование выборки для обучения и проверки

2.1. Первичный анализ

Получим информацию о таблице с данными.

In [275]:
```
nrow = data.shape[0]
ncol = data.shape[1]
print ("Итого: " + str(nrow) + " записи, " + str(ncol) + " столбец (признак).")
```

```
Итого: 13580 записи, 21 столбец (признак).
```

In [276]:
```
print(data.info())
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 13580 entries, 0 to 13579
Data columns (total 21 columns):
Suburb           13580 non-null object
Address          13580 non-null object
Rooms            13580 non-null int64
Type             13580 non-null object
Price            13580 non-null float64
Method           13580 non-null object
SellerG          13580 non-null object
Date             13580 non-null object
Distance         13580 non-null float64
Postcode         13580 non-null float64
Bedroom2         13580 non-null float64
Bathroom         13580 non-null float64
Car              13518 non-null float64
Landsize         13580 non-null float64
BuildingArea     7130 non-null float64
YearBuilt        8205 non-null float64
CouncilArea      12211 non-null object
Lattitude        13580 non-null float64
Longtitude       13580 non-null float64
Regionname       13580 non-null object
Propertycount    13580 non-null float64
dtypes: float64(12), int64(1), object(8)
memory usage: 2.2+ MB
None
```

На этом этапу становиться заметны пропуски, т.к. кол-во значений присутствующих в столбцах не совпадает. Обработку пропущенных (NA - not available) выполним позднее.

In [277]:
```
# Получить количество пропущенных значений для каждого столбца, как сумма всех True
data.isnull().sum()
```

Out[277]:
```
Suburb              0
Address             0
Rooms               0
Type                0
Price               0
Method              0
SellerG             0
Date                0
Distance            0
Postcode            0
Bedroom2            0
Bathroom            0
Car                62
Landsize            0
BuildingArea     6450
YearBuilt        5375
CouncilArea      1369
Lattitude           0
Longtitude          0
Regionname          0
Propertycount       0
dtype: int64
```

Метод **describe()** показывает основные статистические характеристики данных по каждому числовому признаку (типы int64 и float64):

- count - число непропущенных значений,
- mean - среднее арифметическое
- std - стандартное отклонение (среднеквадратическое),
- min, max - минимальное и максимальное значение
- 0.25, 0.50, 0.75 квартили.
- unique - кол-во уникальных значения
- top - самое часто встречающееся значение
- freq - кол-во значений 'top'

In [278]:
```
data.describe() # описание для числовых значений
```

Out[278]:

	Rooms	Price	Distance	Postcode	Bedroom2	Bathroom	Car	Landsize	BuildingArea	YearBuilt	Lattitude	Longtit
count	13580.000	13580.000	13580.000	13580.000	13580.000	13580.000	13518.000	13580.000	7130.000	8205.000	13580.000	13580.
mean	2.938	1075684.079	10.138	3105.302	2.915	1.534	1.610	558.416	151.968	1964.684	-37.809	144.
std	0.956	639310.724	5.869	90.677	0.966	0.692	0.963	3990.669	541.015	37.274	0.079	0.
min	1.000	85000.000	0.000	3000.000	0.000	0.000	0.000	0.000	0.000	1196.000	-38.183	144.
25%	2.000	650000.000	6.100	3044.000	2.000	1.000	1.000	177.000	93.000	1940.000	-37.857	144.
50%	3.000	903000.000	9.200	3084.000	3.000	1.000	2.000	440.000	126.000	1970.000	-37.802	145.
75%	3.000	1330000.000	13.000	3148.000	3.000	2.000	2.000	651.000	174.000	1999.000	-37.756	145.
max	10.000	9000000.000	48.100	3977.000	20.000	8.000	10.000	433014.000	44515.000	2018.000	-37.409	145.

In [279]:
```
data.describe(include=['object', 'bool'])  # описание для не числовых значений
```

Out[279]:

	Suburb	Address	Type	Method	SellerG	Date	CouncilArea	Regionname
count	13580	13580	13580	13580	13580	13580	12211	13580
unique	314	13378	3	5	268	58	33	8
top	Reservoir	13 Robinson St	h	S	Nelson	27/05/2017	Moreland	Southern Metropolitan
freq	359	3	9449	9022	1565	473	1163	4695

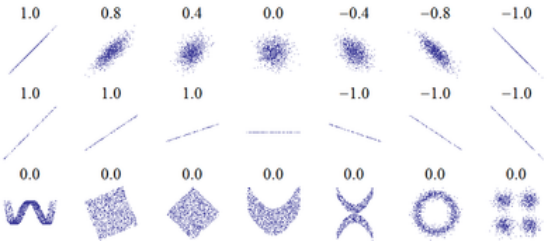
In [280]:
```
print data['Type'].unique() # уникальные значения
print data['Type'].nunique() # кол-во уникальных значения
```

```
['h' 'u' 't']
3
```

2.2. Корреляция

- **Коэффициент корреляции r-Пирсона**

  Мера линейной взаимосвязи переменных, при нелинейной взаимосвязи непоказателен. Не устойчив к выбросам в выборке. Учитывает числовые значения. Переменные должны иметь нормальное распределение. Изменяется в пределах от -1 до +1.



Перед дальнейшими действиями следует обработать пропущенные значения.

In [281]:
```
data.isna().sum()
```

Out[281]:
```
Suburb             0
Address            0
Rooms              0
Type               0
Price              0
Method             0
SellerG            0
Date               0
Distance           0
Postcode           0
Bedroom2           0
Bathroom           0
Car               62
Landsize           0
BuildingArea    6450
YearBuilt       5375
CouncilArea     1369
Lattitude          0
Longtitude         0
Regionname         0
Propertycount      0
dtype: int64
```

Возможные варианты:

- отбросить (исключить) строки в которых отсутствуют значения. **.dropna(how='any')**
- заполнить отсутствующее значение. Для числовых значений можно использовать среденее **.mean()**.

In [282]:
```
data_origin = data
```

In [283]:
```
data = data.dropna(how='any')
data.to_csv('./melbourne-housing-snapshot/melb_data_clear.csv')
# inplace=True - означает, что операция примениться к исходному набору данных

#data['Car'].fillna(data['Car'].mean(), inplace=True)
#data['BuildingArea'].fillna(data['BuildingArea'].mean(), inplace=True)
#data['YearBuilt'].fillna(data['YearBuilt'].mean(), inplace=True)
#data['CouncilArea'].fillna(data['CouncilArea'].mean(), inplace=True)
```

In [284]:
```
len(data)
```

Out[284]: 6196

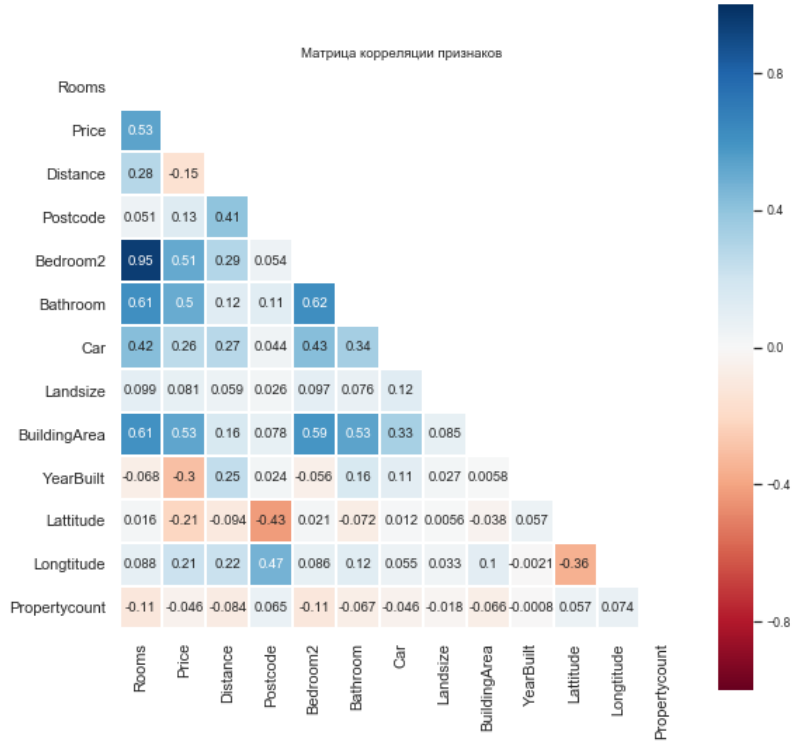
Постоим матрицу корреляции, вызвав метод **.corr(method='pearson')**

In [285]:
```
corrdf = data.corr(method='pearson') # method= {‘pearson’, ‘kendall’, ‘spearman’} или функция
print corrdf
```

```
               Rooms  Price  Distance  Postcode  Bedroom2  Bathroom    Car  \
Rooms          1.000  0.534     0.284     0.051     0.952     0.614  0.423
Price          0.534  1.000    -0.149     0.126     0.515     0.504  0.262
Distance       0.284 -0.149     1.000     0.406     0.290     0.125  0.274
Postcode       0.051  0.126     0.406     1.000     0.054     0.110  0.044
Bedroom2       0.952  0.515     0.290     0.054     1.000     0.618  0.426
Bathroom       0.614  0.504     0.125     0.110     0.618     1.000  0.341
Car            0.423  0.262     0.274     0.044     0.426     0.341  1.000
Landsize       0.099  0.081     0.059     0.026     0.097     0.076  0.118
BuildingArea   0.608  0.531     0.160     0.078     0.593     0.534  0.334
YearBuilt     -0.068 -0.305     0.246     0.024    -0.056     0.158  0.109
Lattitude      0.016 -0.214    -0.094    -0.425     0.021    -0.072  0.012
Longtitude     0.088  0.212     0.222     0.471     0.086     0.116  0.055
Propertycount -0.111 -0.046    -0.084     0.065    -0.109    -0.067 -0.046

               Landsize  BuildingArea  YearBuilt  Lattitude  Longtitude  \
Rooms             0.099         0.608     -0.068      0.016       0.088
Price             0.081         0.531     -0.305     -0.214       0.212
Distance          0.059         0.160      0.246     -0.094       0.222
Postcode          0.026         0.078      0.024     -0.425       0.471
Bedroom2          0.097         0.593     -0.056      0.021       0.086
Bathroom          0.076         0.534      0.158     -0.072       0.116
Car               0.118         0.334      0.109      0.012       0.055
Landsize          1.000         0.085      0.027      0.006       0.033
BuildingArea      0.085         1.000      0.006     -0.038       0.104
YearBuilt         0.027         0.006      1.000      0.057      -0.002
Lattitude         0.006        -0.038      0.057      1.000      -0.358
Longtitude        0.033         0.104     -0.002     -0.358       1.000
Propertycount    -0.018        -0.066     -0.001      0.057       0.074

               Propertycount
Rooms                 -0.111
Price                 -0.046
Distance              -0.084
Postcode               0.065
Bedroom2              -0.109
Bathroom              -0.067
Car                   -0.046
Landsize              -0.018
BuildingArea          -0.066
YearBuilt             -0.001
Lattitude              0.057
Longtitude             0.074
Propertycount          1.000
```

Для того, чтобы визуально оценить величину коэффициентов корреляции постоим тепловую карту по значениям взаимозависимости признаков выборки.

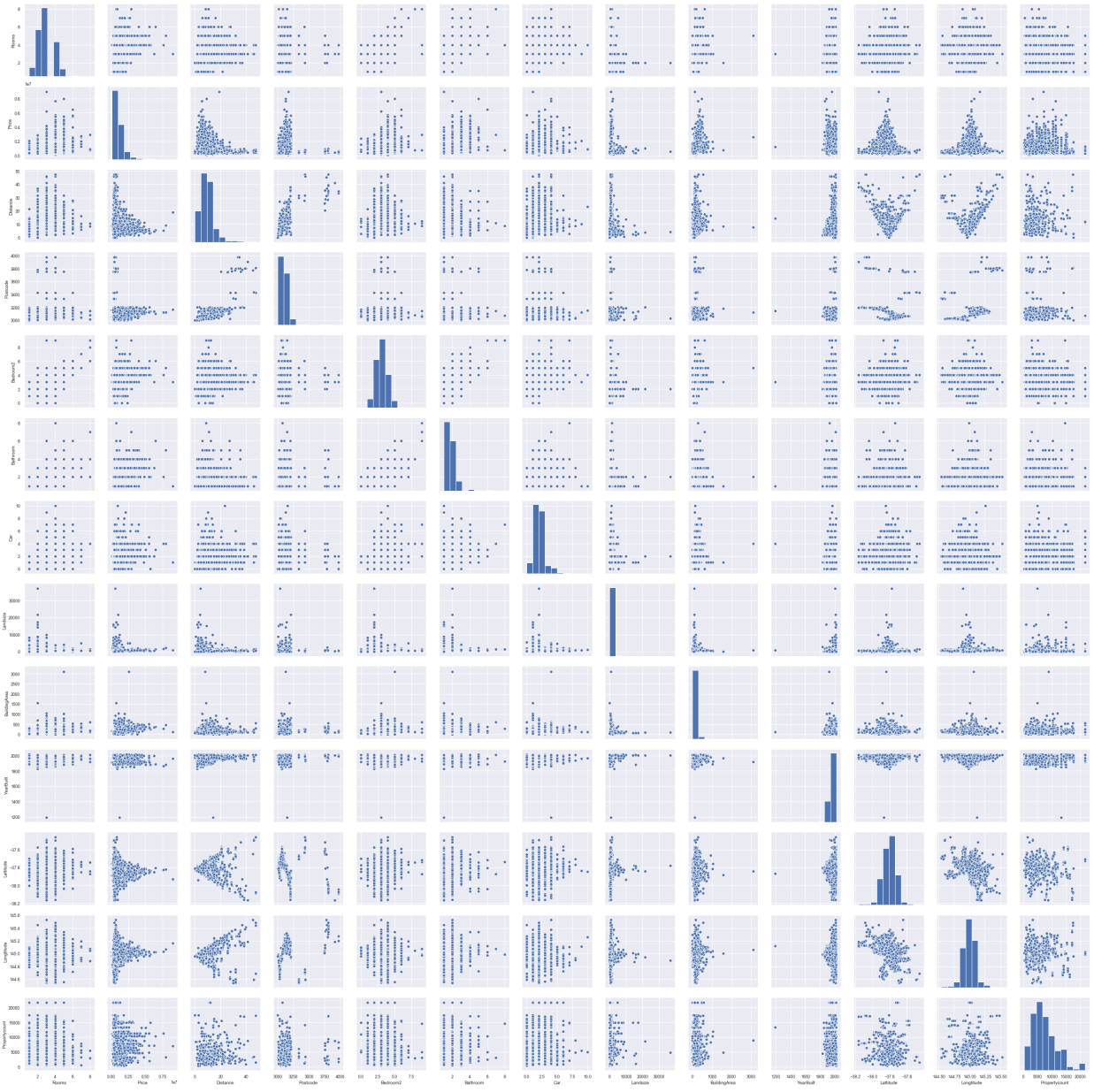
In [286]:
```
import matplotlib.pyplot as plt
##cmap = sns.diverging_palette(220, 10, as_cmap=True) ## сгенерировать цветовую карту
sns.set(style="white")# цвет фона

# построить "маску" для отображения только половины корреляционной матрицы
mask = np.zeros_like(corrdf, dtype=np.bool)
mask[np.triu_indices_from(mask)] = True

fig, ax = plt.subplots(figsize=(9,9))
sns.set(font_scale=0.8) # размер шрифта

# Построить тепловую карту
sns.heatmap(data=corrdf, vmin=-1, vmax=1, center=0,
            annot=True,
            cmap = 'RdBu',
            mask=mask,   # применить маску
            square=True, # форма ячейки - квадрат
            linewidths=1.0, # зазор между ячейки
            #cbar_kws={"shrink": .5}  # уменьшить размер легенды
           )
plt.title(u'Матрица корреляции признаков');
# cmap = https://matplotlib.org/3.1.0/tutorials/colors/colormaps.html
```



Построим диаграммы между каждой парой признаков

In [287]:
```
%config InlineBackend.figure_format = 'png' # формат изображений PNG, для сокращения времени отрисовки графиков
sns.pairplot(data)
```

Out[287]: <seaborn.axisgrid.PairGrid at 0x1f264b00>



2.3. Выбор целевого значения и признаков для анализа данных

На основе информации о взаимозависимости и степени влияния признаков выберем целевое значение:

- 'Price' - Стоимость.

Признаки для анализа следует выбирать с высоким коэфф. корреляции с целевым значением, но при этом как можно более не связанные между собой:

- 'YearBuilt' - Год посройки
- 'Landsize'- Размер участка
- 'Rooms' - Кол-во комнат

Выведем коэфф. корреляции для выбранных признаков

In [288]:
```
features = [u'Rooms', u'YearBuilt', u'Landsize']
target = [u'Price']

for features_name in features:
    print "{:<9}, {} = {:>6.3f}".format(features_name, target, corrdf[features_name][target[0]])
```

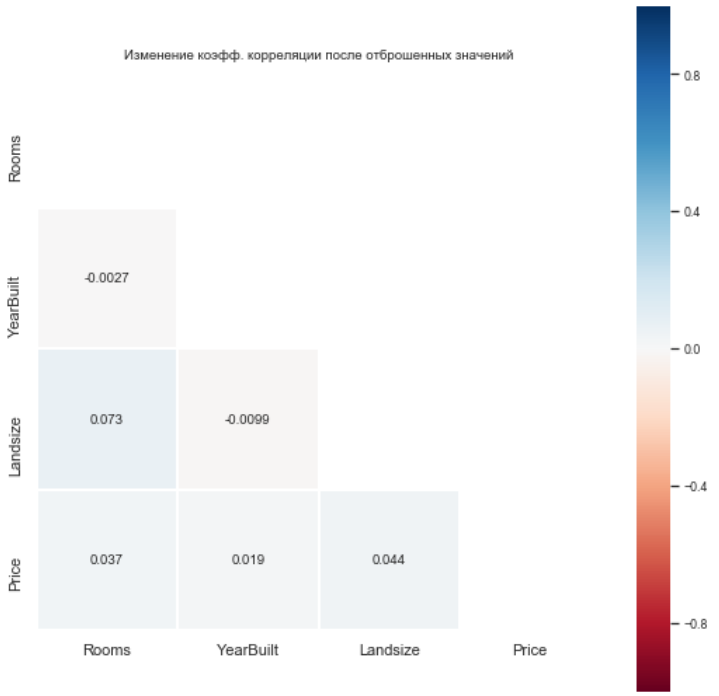
```
Rooms    , [u'Price'] =  0.534
YearBuilt, [u'Price'] = -0.305
Landsize , [u'Price'] =  0.081
```

In [289]:
```
test_corrdf2 = data[features + target].corr(method='pearson') # method= {‘pearson’, ‘kendall’, ‘spearman’} или функция
print test_corrdf2
```

```
           Rooms  YearBuilt  Landsize  Price
Rooms      1.000     -0.068     0.099  0.534
YearBuilt -0.068      1.000     0.027 -0.305
Landsize   0.099      0.027     1.000  0.081
Price      0.534     -0.305     0.081  1.000
```

Посмотрим, насколько повлияли отброшенные значения на коэфф. корреляции для выбранных признаков

In [290]:
```
import matplotlib.pyplot as plt
sns.set(style="white") # цвет фона

delta_corrdf = data[ features + target ].corr(method='pearson') - data_origin[features + target].corr(method='pearson')

# построить "маску" для отображения только половины корреляционной матрицы
mask = np.zeros_like(delta_corrdf, dtype=np.bool)
mask[np.triu_indices_from(mask)] = True

fig, ax = plt.subplots(figsize=(9,9))
sns.set(font_scale=0.8) # размер шрифта

# Построить тепловую карту
sns.heatmap(data=delta_corrdf,
            vmin=-1, vmax=1, center=0,
            annot=True,
            cmap = 'RdBu',
            mask=mask,   # применить маску
            square=True, # форма ячейки - квадрат
            linewidths=1.0, # зазор между ячейки
            #cbar_kws={"shrink": .5}  # уменьшить размер легенды
           )
plt.title(u'Изменение коэфф. корреляции после отброшенных значений');
```

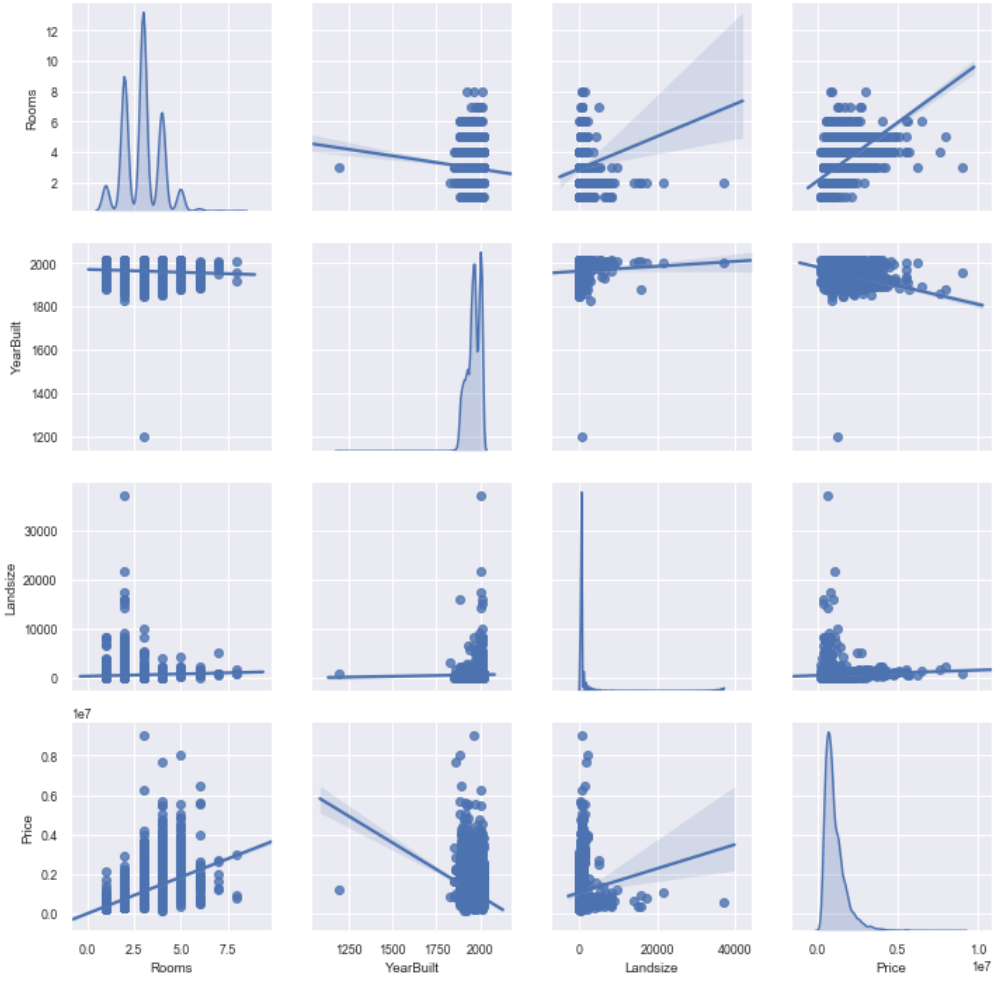


In [291]:
```
delta_corrdf[target]
```

Out[291]:

	Price
Rooms	0.037
YearBuilt	0.019
Landsize	0.044
Price	0.000

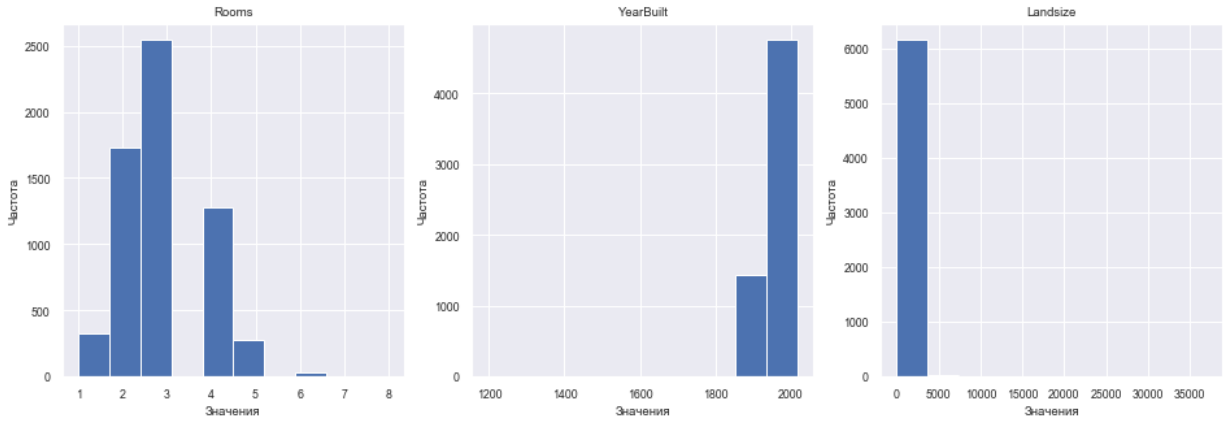
In [292]:
```
# График признаков попарно с разделением по типу
%config InlineBackend.figure_format = 'png' # формат изображений PNG, для сокращения времени отрисовки графиков
sns.pairplot(data[features + target],
             #hue = 'Type', # Признак разделения
             kind="reg", # Fit linear regression models to the scatter plots:
             diag_kind = 'kde', # Признак разделения тип графика
             );
```



2.4. Просмотр характеристик выбранных признаков

In [293]:
```
# Строим гистограммы по каждому признаку:
#data[features].hist() #Упрощенный вывод графиков

plt.figure(figsize(16, 5))
plot_number = 0
for feature_name in features:
    plot_number += 1
    plt.subplot(1,3,plot_number)
    plt.hist(data[feature_name])

    #plt.xlim( data[feature_name].min(), data[feature_name].max() )

    plt.title(feature_name)
    plt.xlabel(u'Значения')
    plt.ylabel(u'Частота')
    print feature_name, data[feature_name].min(), data[feature_name].max()
```

```
Rooms 1 8
YearBuilt 1196.0 2018.0
Landsize 0.0 37000.0
```



Признак 'YearBuilt'

По графику распределения заметно существенное смещение признака 'YearBuilt'. Предположительно в "пустой" части графика располагается малая часть значений. Проверим минимальное значение и квартили.

In [294]:
```
data[['YearBuilt']].describe()
```

Out[294]:

	YearBuilt
count	6196.000
mean	1964.082
std	38.106
min	1196.000
25%	1940.000
50%	1970.000
75%	2000.000
max	2018.000

In [295]:
```
print ("minimum YearBuilt", data['YearBuilt'].min())
data[data['YearBuilt'] < 1820]
```

```
('minimum YearBuilt', 1196.0)
```

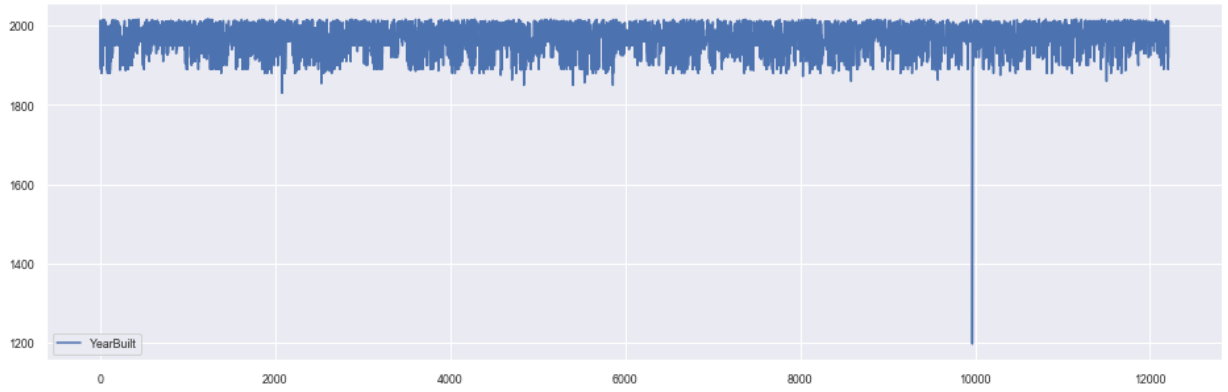
Out[295]:

	Suburb	Address	Rooms	Type	Price	Method	SellerG	Date	Distance	Postcode	...	Bathroom	Car	Landsize	Buil
9968	Mount Waverley	5 Armstrong St	3	h	1200000.000	VB	McGrath	24/06/2017	14.200	3149.000	...	1.000	4.000	807.000	

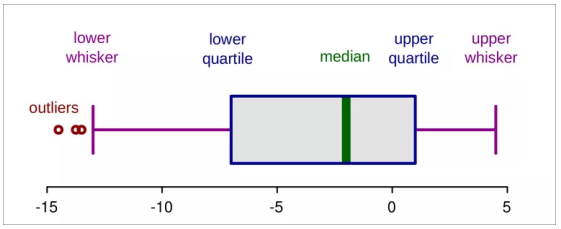
1 rows × 21 columns

In [296]:
```
plt.figure(figsize(16, 5))
data[['YearBuilt']].plot(kind='line')
```

Out[296]:
```
<matplotlib.axes._subplots.AxesSubplot at 0x353b0a58>

<Figure size 1152x360 with 0 Axes>
```

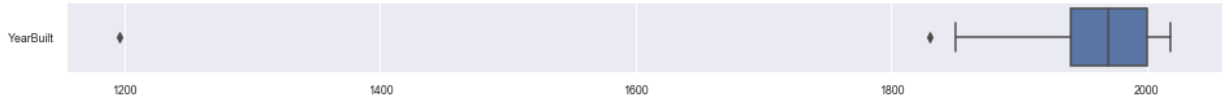


Построим диаграмму размаха для признака



In [297]:
```
plt.figure(figsize(16, 1))
sns.boxplot( data=data[['YearBuilt']], orient="h")
```

Out[297]:
```
<matplotlib.axes._subplots.AxesSubplot at 0x30527668>
```



In [298]:
```
dcount = data['YearBuilt'].count() # кол-во эл. общее
dmean = data['YearBuilt'].mean() # среднее зн. общее
over_count = len(data[data['YearBuilt'] < 1820]) # кол-во эл. ниже порогового
over_mean = data[ data['YearBuilt']  < 1820 ]['YearBuilt'].mean()  # среднее зн. выше порогового

print "В выборке обнаружен \"выброс\" - {} из {} значение(ий) с величиной отличающейся в {:.3f} раз чем у {:.6f} части оставшейся выборки.".format(
    over_count,
    dcount,
    over_mean / dmean,
    (dcount - over_count * 1.0) / dcount )
```

```
В выборке обнаружен "выброс" - 1 из 6196 значение(ий) с величиной отличающейся в 0.609 раз чем у 0.999839 части оставшейся выборки.
```

Данные элементы будут отброшены и не будут учитываться в дальнейшем анализе.

In [299]:
```
# оставить строки, с значение выше порогового
data = data[  ~  (data['YearBuilt'] < 1820) ]
```

In [300]:
```
len(data['YearBuilt'])
```

Out[300]:
```
6195
```

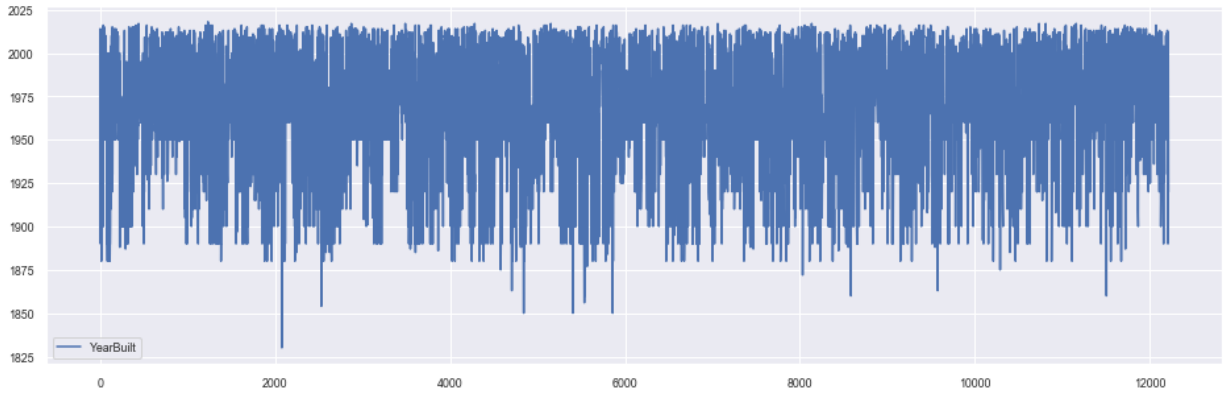
In [301]:
```
data[['YearBuilt']].describe()
```

Out[301]:

	YearBuilt
count	6195.000
mean	1964.206
std	36.838
min	1830.000
25%	1940.000
50%	1970.000
75%	2000.000
max	2018.000

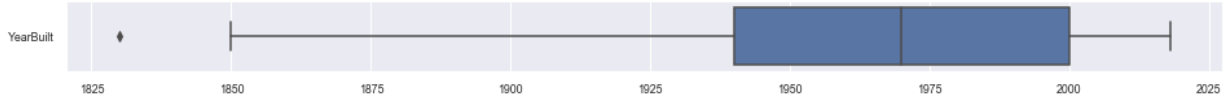
In [302]:
```
plt.figure(figsize(16, 5))
data[['YearBuilt']].plot(kind='line')
```

Out[302]:
```
<matplotlib.axes._subplots.AxesSubplot at 0x35517860>

<Figure size 1152x360 with 0 Axes>
```



In [303]:
```
plt.figure(figsize(16, 1))
sns.boxplot( data=data[['YearBuilt']], orient="h")
```

Out[303]:
```
<matplotlib.axes._subplots.AxesSubplot at 0x3537d588>
```



Признак 'Landsize'

In [304]:
```
data[['Landsize']].describe()
```

Out[304]:

	Landsize
count	6195.000
mean	470.953
std	897.512
min	0.000
25%	152.000
50%	373.000
75%	628.000
max	37000.000

In [305]:
```
print ("minimum Landsize", data['Landsize'].min())
data[data['Landsize'] > 10000]
```

```
('minimum Landsize', 0.0)
```

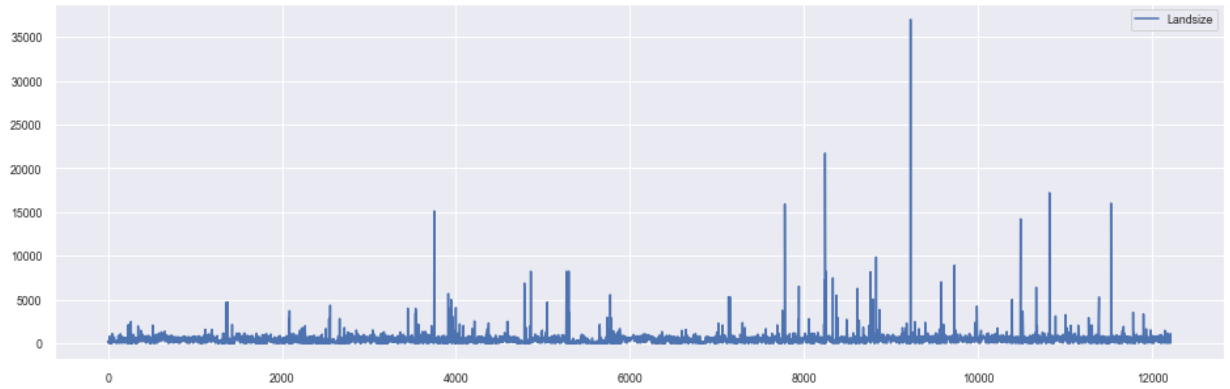
Out[305]:

	Suburb	Address	Rooms	Type	Price	Method	SellerG	Date	Distance	Postcode	...	Bathroom	Car	Land
3750	Maidstone	17/46 Eucalyptus Dr	2	u	352500.000	S	hockingstuart	18/03/2017	9.200	3012.000	...	1.000	1.000	15100
7778	Collingwood	1/78 Oxford St	2	t	945000.000	S	Jellis	8/04/2017	1.600	3066.000	...	1.000	1.000	15900
8241	Port Melbourne	55/4 Seisman Pl	2	u	1030000.000	S	Buxton	29/04/2017	3.800	3207.000	...	2.000	2.000	21715
9223	Maribyrnong	2/6 Horizon Dr	2	u	585000.000	S	Brad	3/06/2017	4.300	3032.000	...	2.000	2.000	37000
10488	Richmond	52/73 River St	2	u	615000.000	PI	hockingstuart	27/05/2017	2.400	3121.000	...	2.000	1.000	14196
10819	South Yarra	308/800 Chapel St	2	u	762500.000	S	hockingstuart	8/07/2017	2.700	3141.000	...	1.000	1.000	17200
11526	Travancore	2115/18 Mt Alexander Rd	2	u	341500.000	S	McGrath	15/07/2017	4.300	3032.000	...	1.000	1.000	16000

7 rows × 21 columns

In [306]:
```
plt.figure(figsize(16, 5))
data[['Landsize']].plot(kind='line')
```

Out[306]:
```
<matplotlib.axes._subplots.AxesSubplot at 0x35ef6710>

<Figure size 1152x360 with 0 Axes>
```



In [307]:
```
dcount = data['Landsize'].count() # кол-во эл. общее
dmean = data['Landsize'].mean() # среднее зн. общее
over_count = len( data[data['Landsize'] > 3500] ) # кол-во эл. выше порогового
over_mean = data[ data['Landsize'] > 3500 ]['Landsize'].mean()  # среднее зн. выше порогового

print "В выборке обнаружен \"выброс\" - {} из {} значение(ий) с величиной отличающейся в {:.3f} больше чем у {:.6f} части оставшейся выборки.".format(
    over_count,
    dcount ,
    over_mean/dmean,
    (dcount - over_count * 1.0) / dcount )
```

```
В выборке обнаружен "выброс" - 48 из 6195 значение(ий) с величиной отличающейся в 16.337 больше чем у 0.992252 части оставшейся выборки.
```

Данные элементы будут отброшены и не будут учитываться в дальнейшем анализе.

In [308]:
```
# оставить строки, с значение ниже порогового
data = data[ ~ (data['Landsize'] > 3500)]
```

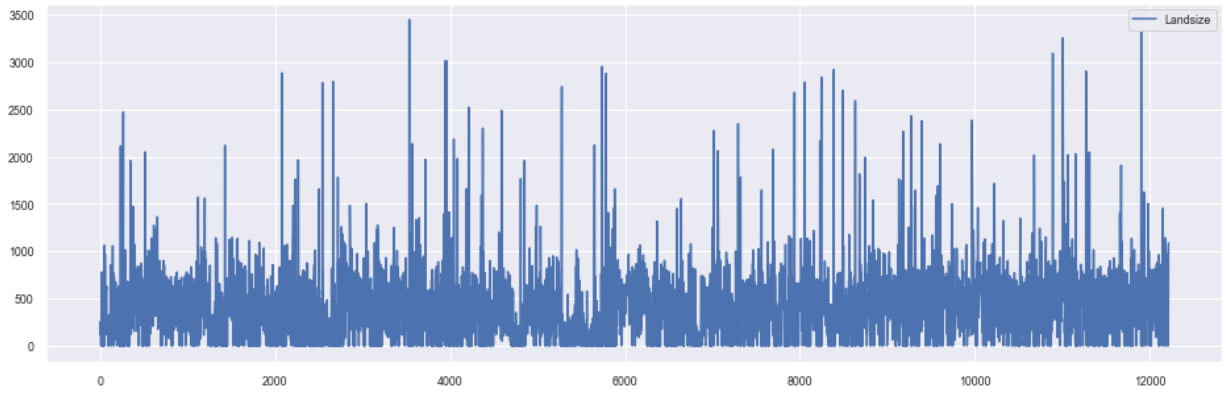
In [309]:
```
data[['Landsize']].describe()
```

Out[309]:

	Landsize
count	6147.000
mean	414.551
std	360.274
min	0.000
25%	151.000
50%	368.000
75%	624.000
max	3448.000

In [310]:
```
plt.figure(figsize(16, 5))
data[['Landsize']].plot(kind='line')
```

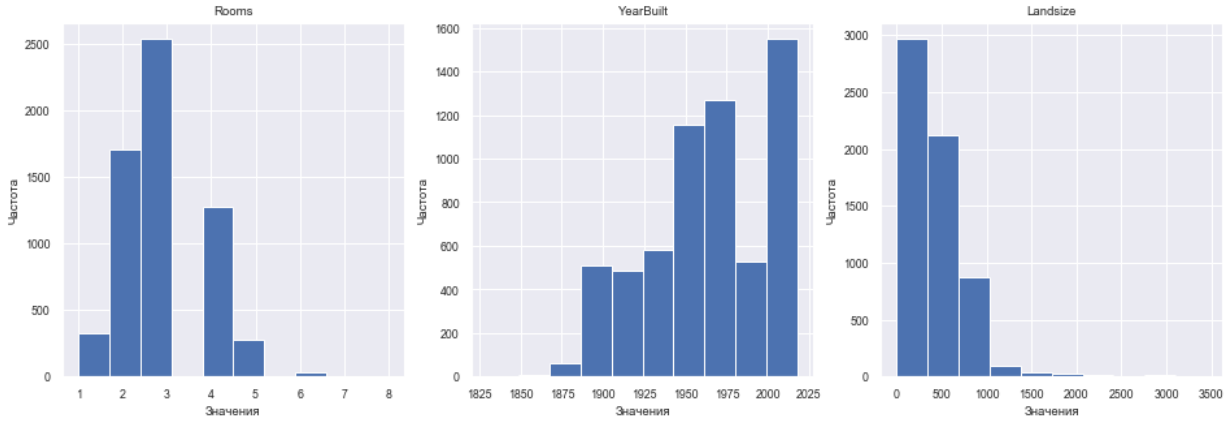
Out[310]:
```
<matplotlib.axes._subplots.AxesSubplot at 0x3728d518>

<Figure size 1152x360 with 0 Axes>
```



Гистограммы по каждому признаку повторно

In [311]:
```
# Строим гистограммы по каждому признаку:
#data[features].hist() #Упрощенный вывод графиков

plt.figure(num=1, figsize=figsize(16, 5))
plot_number = 0
for feature_name in features:
    plot_number += 1
    plt.subplot(1,3,plot_number)
    plt.hist(data[feature_name])
    #plt.xlim( data[feature_name].min(), data[feature_name].max() )

    plt.title(feature_name)
    plt.xlabel(u'Значения')
    plt.ylabel(u'Частота')
    print feature_name, data[feature_name].min(), data[feature_name].max()
```

```
Rooms 1 8
YearBuilt 1830.0 2018.0
Landsize 0.0 3448.0
```



Гистограмма целевой переменной

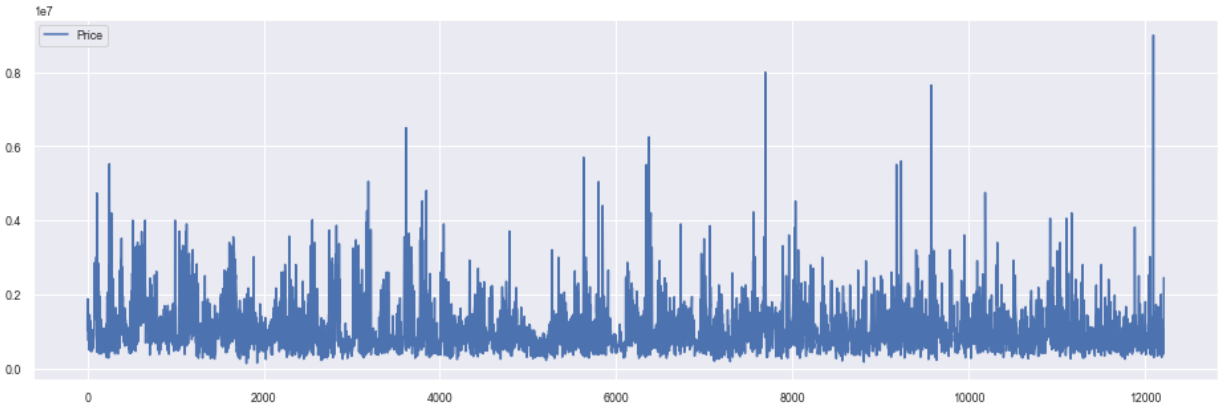
In [312]:
```
data[target].describe()
```

Out[312]:

	Price
count	6147.000
mean	1071375.718
std	675907.683
min	131000.000
25%	622250.000
50%	882000.000
75%	1330000.000
max	9000000.000

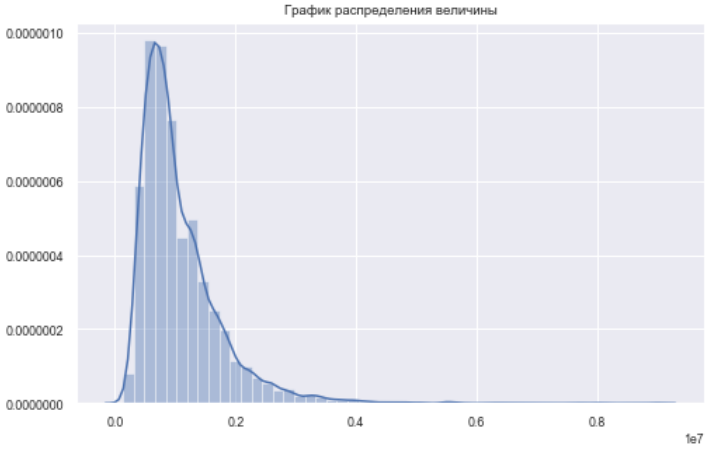
In [313]:
```
plt.figure(figsize(16, 5))
data[target].plot(kind='line')
```

Out[313]:
```
<matplotlib.axes._subplots.AxesSubplot at 0x373edfd0>

<Figure size 1152x360 with 0 Axes>
```



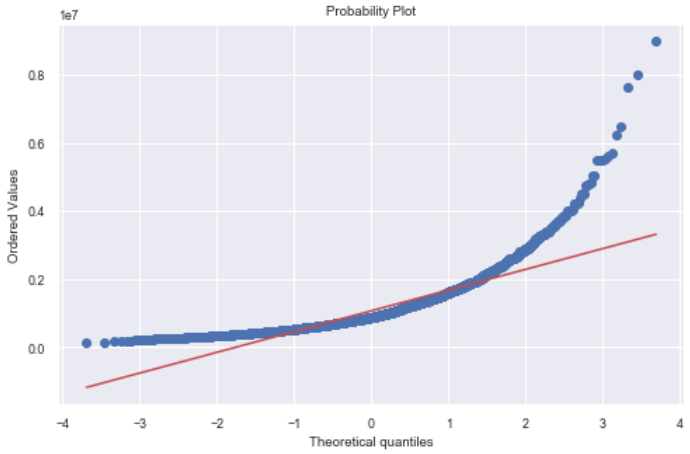
In [314]:
```
plt.figure(figsize=(8,5))
sns.distplot(data[target],  hist=True) # Ядерная оценка плотности (KDE - Kernel density estimation)
plt.title(u'График распределения величины')
print "Коэфф. асимметрии, эксцесса в числовом виде"
```

```
Коэфф. асимметрии, эксцесса в числовом виде
```



In [315]:
```
# диаграмма вероятности
plt.figure(figsize=(8,5))
stats.probplot(data['Price'], plot=plt)
```

Out[315]:
```
((array([-3.68857787, -3.45642624, -3.32863351, ...,  3.32863351,
          3.45642624,  3.68857787]),
  array([ 131000.,  145000.,  170000., ..., 7650000., 8000000., 9000000.])),
 (608888.5303425731, 1071375.718236538, 0.9004195353276806))
```



3. Построение регрессионных моделей

Разделим получившиеся данные на тренировочную (data_x_train, data_y_train) и проверочную части (x_test, y_test)

In [316]:
```
valid_size = 0.3 # доля тестовой части в выборке
rand_seed = 8 # начальное состояние генератора случ. чисел

x_train, x_test, y_train, y_test =  train_test_split(
    data[features], data[target], # исходные данные
    test_size = valid_size,
    random_state=rand_seed,
    shuffle=True # перемешивание
)
```

In [317]:
```
print "Кол-во элементов: \n  x_train: {}, y_train {} \n  x_test:  {}, y_test  {} \n  total x: {}, total y {} ".form
at  (
    len(x_train), len(y_train),
    len(x_test),  len(x_test),
    len(x_train)+len(x_test), len(y_train)+len(x_test),
)
```

```
Кол-во элементов:
  x_train: 4302, y_train 4302
  x_test:  1845, y_test  1845
  total x: 6147, total y 6147
```

In [318]:
```
x_train[:3]
```

Out[318]:

	Rooms	YearBuilt	Landsize
6970	2	1935.000	390.000
7145	3	1975.000	181.000
3569	2	1970.000	0.000

In [319]:
```
y_train[:3]
```

Out[319]:

	Price
6970	1585000.000
7145	835000.000
3569	773000.000

3.1. Линейная регрессия

- Выдвижение рабочей гипотезы
- Построение модели
- Анализ качества и интерпретация модели
- Определение путей изменения модели
- Выдвижение новых гипотез и построение новых моделей.
- Практическое использование модели

In [320]:
```
from sklearn import linear_model
from sklearn.linear_model import LinearRegression
```

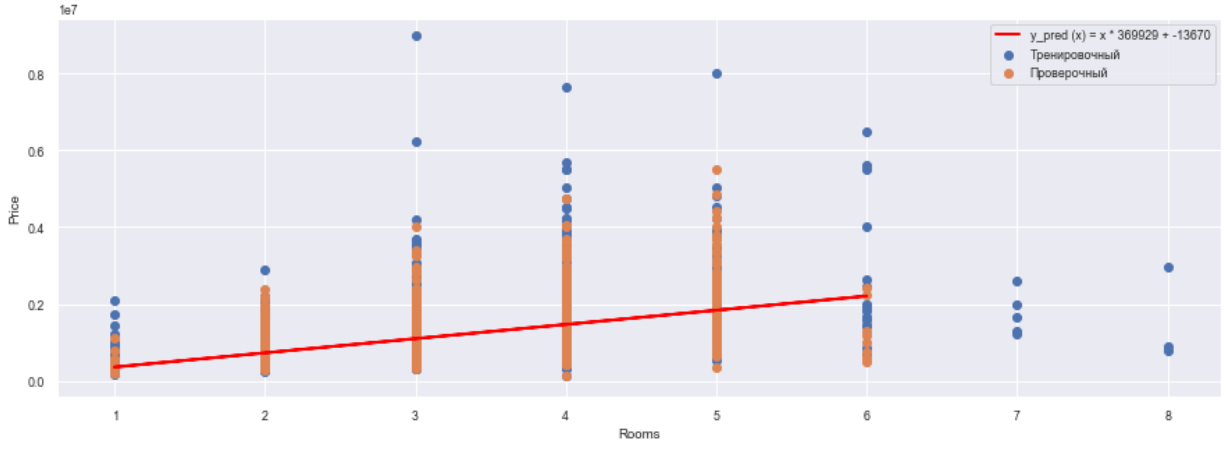
3.1.1. Одномерная модель от признака 'Rooms'

In [321]:
```
lr = linear_model.LinearRegression()

x1 = x_train['Rooms'].values.reshape(-1, 1)
y1 = y_train['Price'].values.reshape(-1, 1)
x2 = x_test['Rooms'].values.reshape(-1, 1)
y2 = y_test['Price'].values.reshape(-1, 1)
```

In [322]:
```
# Вычислить коэфф. на тренировочном наборе
lin_model = lr.fit(x1, y1)
# Получить предсказание на проверочном наборе
y_pred  = lin_model.predict(x2)
```

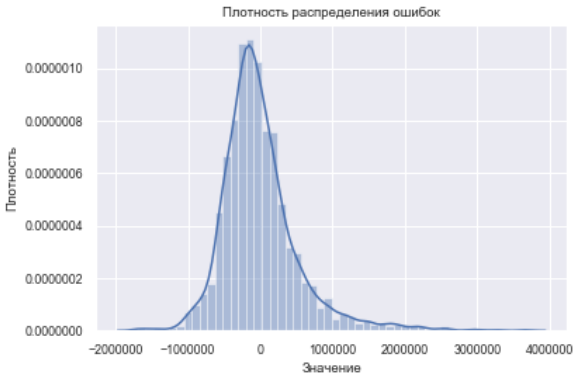
In [323]:
```
plt.figure(figsize(15,5))
fig, ax = plt.subplots()
ax.scatter(x1, y1, label=u'Тренировочный') # тренировочный
ax.scatter(x2, y2, label=u'Проверочный') # проверочный
ax.plot( x2, y_pred, lw=2, color='red',
        label="y_pred (x) = x * {:.0f} + {:.0f}".format(lin_model.coef_[0][0], lin_model.intercept_[0]))
ax.legend(loc="best")
ax.set_xlabel('Rooms')
ax.set_ylabel(target[0])

plt.show()
```

```
<Figure size 1080x360 with 0 Axes>
```



In [324]:
```
plt.figure(figsize=(6,4))
sns.distplot(y2 - y_pred)
plt.title(u'Плотность распределения ошибок');
plt.ylabel(u'Плотность')
plt.xlabel(u'Значение')
```

Out[324]:
```
Text(0.5,0,u'\u0417\u043d\u0430\u0447\u0435\u043d\u0438\u0435')
```



In [325]:
```
print "y_pred (x) = x * {:.0f} + {:.0f}\n".format(lin_model.coef_[0][0], lin_model.intercept_[0])

print "MAE Mean absolute error: {:.0f}".format( mean_absolute_error(y2, y_pred))
print "MSE Mean squared error: {:.0f}".format( mean_squared_error(y2, y_pred))
print "RMSE: {:.0f}".format(np.sqrt( mean_squared_error(y2, y_pred) ))
print "R2 of Linear Regression (1 is perfect): {:.3f}".format( r2_score(y2, y_pred))
```

```
y_pred (x) = x * 369929 + -13670

MAE Mean absolute error: 374250
MSE Mean squared error: 285469691724
RMSE: 534294
R2 of Linear Regression (1 is perfect): 0.318
```

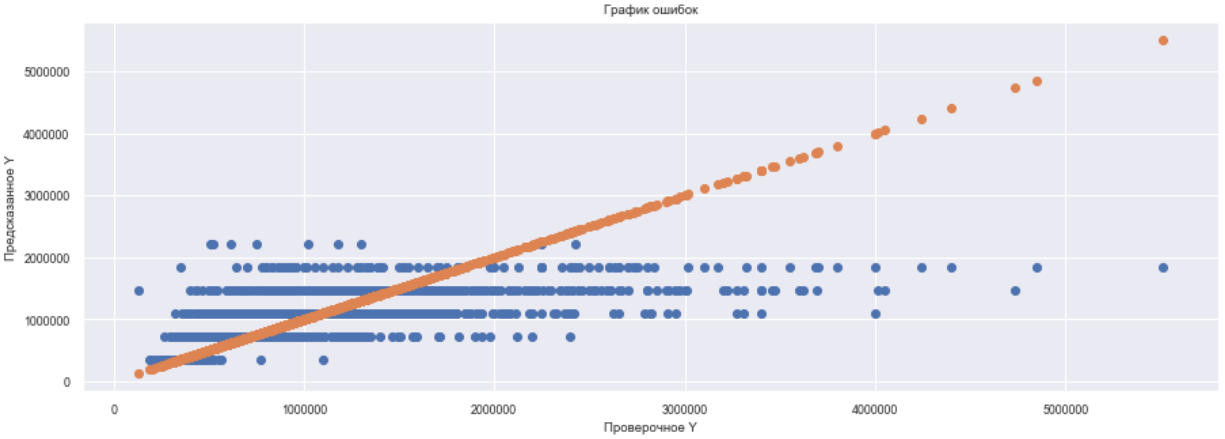
In [326]:
```
df = { u'__Проверочное':  y2.reshape(1,-1)[0],
       u'_Предсказанные': y_pred.reshape(1,-1)[0],
       u'Ошибка':  (y2 - y_pred).reshape(1,-1)[0]}
df = pd.DataFrame(df)
df.head()
```

Out[326]:

	__Проверочное	_Предсказанные	Ошибка
0	1913000.000	1096116.396	816883.604
1	1403000.000	1096116.396	306883.604
2	1631000.000	1096116.396	534883.604
3	680000.000	1096116.396	-416116.396
4	1250000.000	1096116.396	153883.604

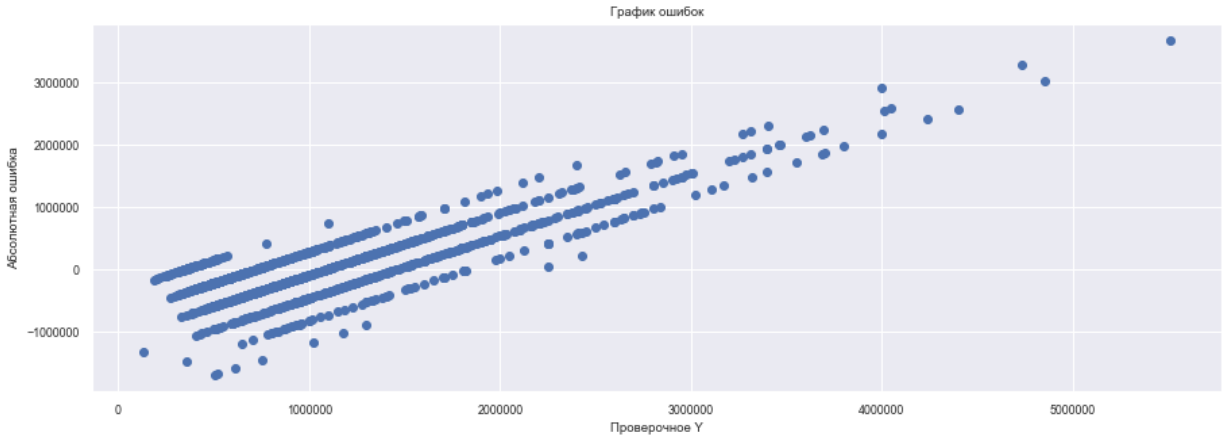
In [327]:
```
plt.scatter(y2,y_pred)
plt.scatter(y2,y2)
plt.title(u'График ошибок');
plt.xlabel(u'Проверочное Y')
plt.ylabel(u'Предсказанное Y')
```

Out[327]:
```
Text(0,0.5,u'\u041f\u0440\u0435\u0434\u0441\u043a\u0430\u0437\u0430\u043d\u043d\u043e\u0435 Y')
```



In [328]:
```
plt.scatter(y2,y2- y_pred)

plt.title(u'График ошибок')
plt.xlabel(u'Проверочное Y')
plt.ylabel(u'Абсолютная ошибка')
```

Out[328]:
```
Text(0,0.5,u'\u0410\u0431\u0441\u043e\u043b\u044e\u0442\u043d\u0430\u044f \u043e\u0448\u0438\u0431\u043a\u0430')
```



In [ ]:
```
```

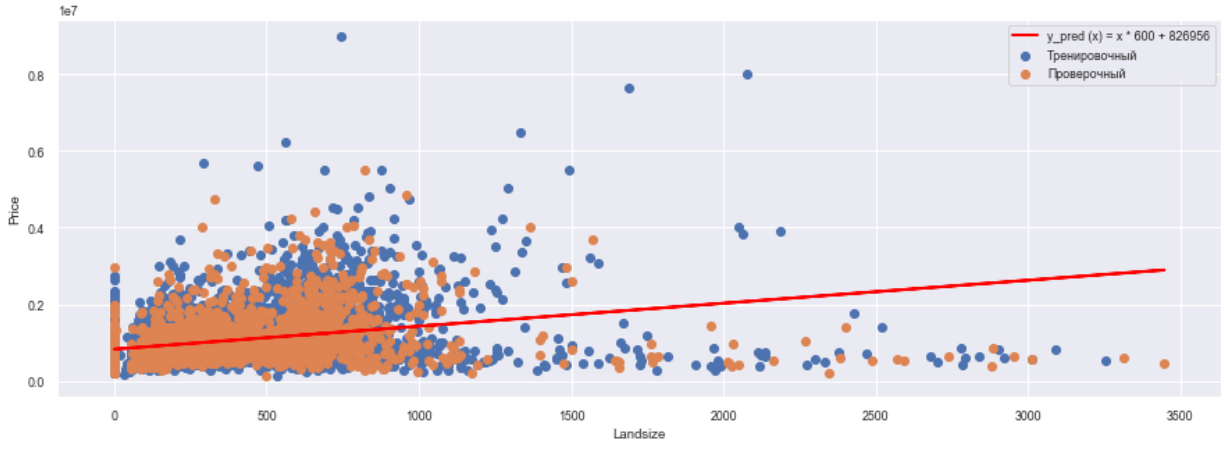
3.1.2. Одномерная модель от признака 'Landsize'

In [329]:
```
lr = linear_model.LinearRegression()

x1 = x_train['Landsize'].values.reshape(-1, 1)
y1 = y_train['Price'].values.reshape(-1, 1)
x2 = x_test['Landsize'].values.reshape(-1, 1)
y2 = y_test['Price'].values.reshape(-1, 1)
```

In [330]:
```
lin_model = lr.fit(x1, y1)
y_pred  = lin_model.predict(x2)
```

In [331]:
```
plt.figure(figsize(15,5))
fig, ax = plt.subplots()
ax.scatter(x1, y1, label=u'Тренировочный') # тренировочный
ax.scatter(x2, y2, label=u'Проверочный') # проверочный
ax.plot( x2, y_pred, lw=2, color='red',
        label="y_pred (x) = x * {:.0f} + {:.0f}".format(lin_model.coef_[0][0], lin_model.intercept_[0]))
ax.legend(loc="best")
ax.set_xlabel('Landsize')
ax.set_ylabel(target[0])

plt.show()
```

```
<Figure size 1080x360 with 0 Axes>
```



In [332]:
```
plt.figure(figsize=(10,4))
sns.distplot(y2 - y_pred)
plt.title(u'Плотность распределения ошибок');
plt.ylabel(u'Плотность')
plt.xlabel(u'Значение')
```

Out[332]:
```
Text(0.5,0,u'\u0417\u043d\u0430\u0447\u0435\u043d\u0438\u0435')
```

In [333]:
```
print "y_pred (x) = x * {:.0f} + {:.0f}\n".format(lin_model.coef_[0][0], lin_model.intercept_[0])

print "MAE Mean absolute error: {:.0f}".format( mean_absolute_error(y2, y_pred))
print "MSE Mean squared error: {:.0f}".format( mean_squared_error(y2, y_pred))
print "RMSE: {:.0f}".format(np.sqrt( mean_squared_error(y2, y_pred) ))
print "R2 of Linear Regression (1 is perfect): {:.3f}".format( r2_score(y2, y_pred))
```

```
y_pred (x) = x * 600 + 826956

MAE Mean absolute error: 451103
MSE Mean squared error: 400406715581
RMSE: 632777
R2 of Linear Regression (1 is perfect): 0.043
```

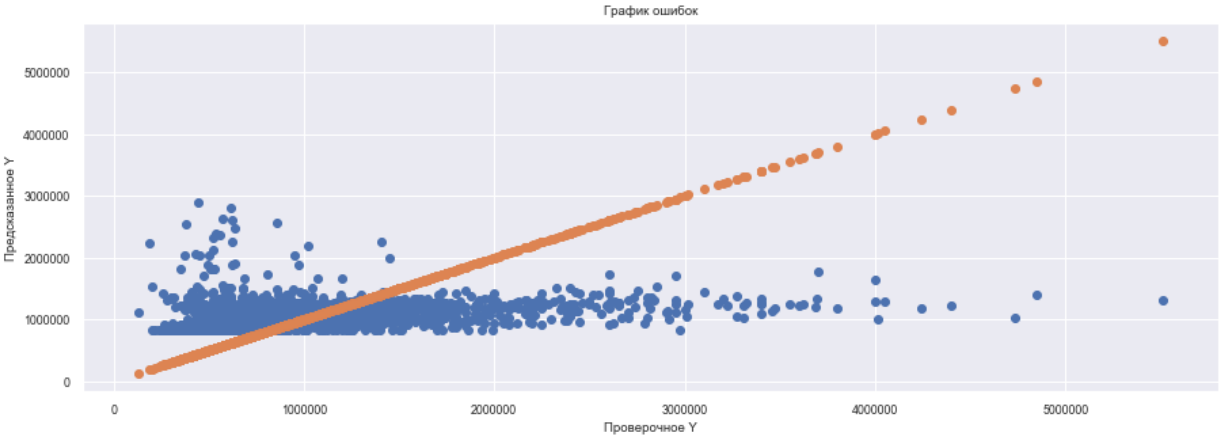
In [334]:
```
df = { u'__Проверочное':  y2.reshape(1,-1)[0],
       u'_Предсказанные': y_pred.reshape(1,-1)[0],
       u'Ошибка':  (y2 - y_pred).reshape(1,-1)[0]}
df = pd.DataFrame(df)
df.head()
```

Out[334]:

	__Проверочное	_Предсказанные	Ошибка
0	1913000.000	1193463.628	719536.372
1	1403000.000	2267194.319	-864194.319
2	1631000.000	1296037.901	334962.099
3	680000.000	1080092.063	-400092.063
4	1250000.000	968520.047	281479.953

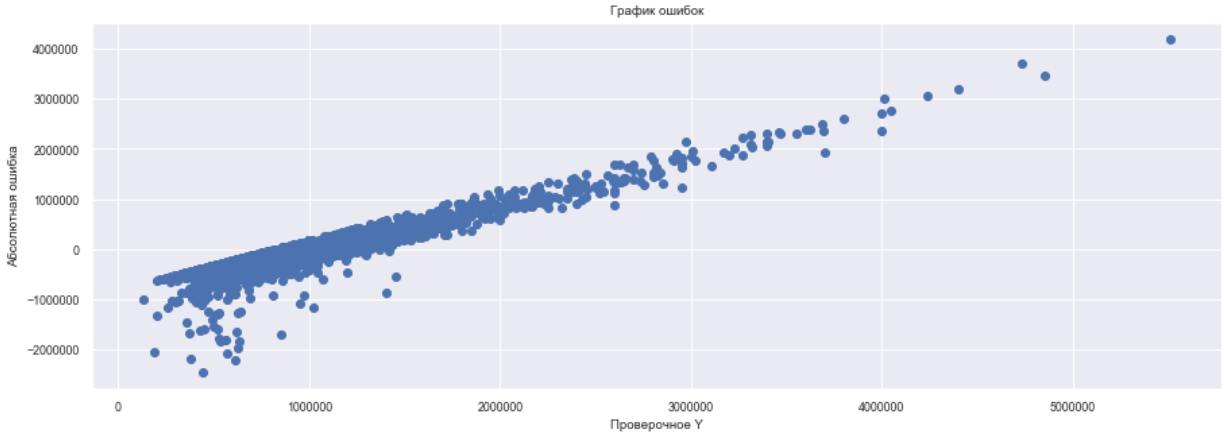
In [335]:
```
plt.scatter(y2,y_pred)
plt.scatter(y2,y2)
plt.title(u'График ошибок');
plt.xlabel(u'Проверочное Y')
plt.ylabel(u'Предсказанное Y')
```

Out[335]:
```
Text(0,0.5,u'\u041f\u0440\u0435\u0434\u0441\u043a\u0430\u0437\u0430\u043d\u043d\u043e\u0435 Y')
```



In [336]:
```
plt.scatter(y2,y2- y_pred)

plt.title(u'График ошибок')
plt.xlabel(u'Проверочное Y')
plt.ylabel(u'Абсолютная ошибка')
```

Out[336]:
```
Text(0,0.5,u'\u0410\u0431\u0441\u043e\u043b\u044e\u0442\u043d\u0430\u044f \u043e\u0448\u0438\u0431\u043a\u0430')
```



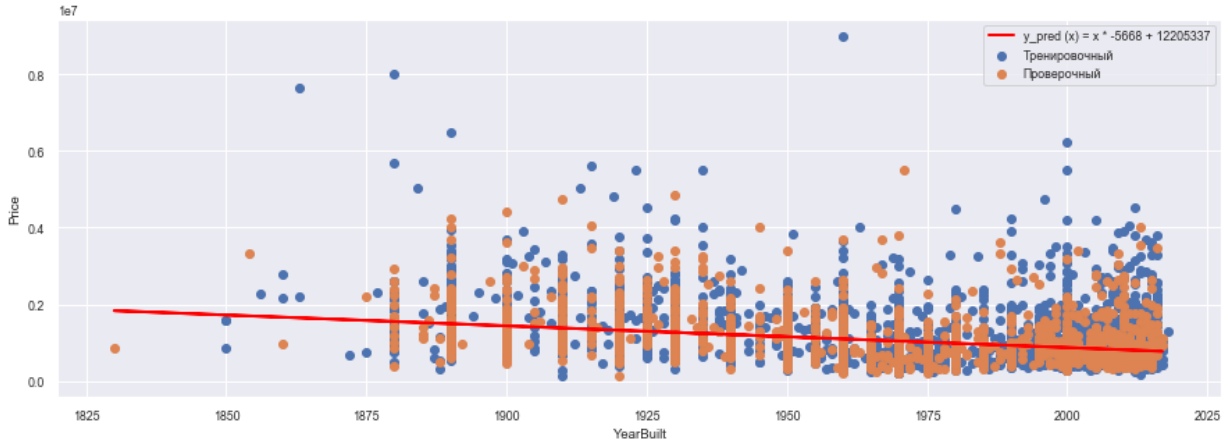
3.1.3. Одномерная модель от признака 'YearBuilt'

In [337]:
```
lr = linear_model.LinearRegression()

x1 = x_train['YearBuilt'].values.reshape(-1, 1)
y1 = y_train['Price'].values.reshape(-1, 1)
x2 = x_test['YearBuilt'].values.reshape(-1, 1)
y2 = y_test['Price'].values.reshape(-1, 1)
```

In [338]:
```
lin_model = lr.fit(x1, y1)
y_pred  = lin_model.predict(x2)
```

In [339]:
```
plt.figure(figsize(15,5))
fig, ax = plt.subplots()
ax.scatter(x1, y1, label=u'Тренировочный') # тренировочный
ax.scatter(x2, y2, label=u'Проверочный') # проверочный
ax.plot( x2, y_pred, lw=2, color='red',
        label="y_pred (x) = x * {:.0f} + {:.0f}".format(lin_model.coef_[0][0], lin_model.intercept_[0]))
ax.legend(loc="best")
ax.set_xlabel('YearBuilt')
ax.set_ylabel(target[0])

plt.show()
```

```
<Figure size 1080x360 with 0 Axes>
```



In [340]:
```
plt.figure(figsize=(6,4))
sns.distplot(y2 - y_pred)
plt.title(u'Плотность распределения ошибок');
plt.ylabel(u'Плотность')
plt.xlabel(u'Значение')
```

Out[340]:
```
Text(0.5,0,u'\u0417\u043d\u0430\u0447\u0435\u043d\u0438\u0435')
```

In [341]:
```
print "y_pred (x) = x * {:.0f} + {:.0f}\n".format(lin_model.coef_[0][0], lin_model.intercept_[0])

print "MAE Mean absolute error: {:.0f}".format( mean_absolute_error(y2, y_pred))
print "MSE Mean squared error: {:.0f}".format( mean_squared_error(y2, y_pred))
print "RMSE: {:.0f}".format(np.sqrt( mean_squared_error(y2, y_pred) ))
print "R2 of Linear Regression (1 is perfect): {:.3f}".format( r2_score(y2, y_pred))
```

```
y_pred (x) = x * -5668 + 12205337

MAE Mean absolute error: 435533
MSE Mean squared error: 370676653413
RMSE: 608832
R2 of Linear Regression (1 is perfect): 0.114
```

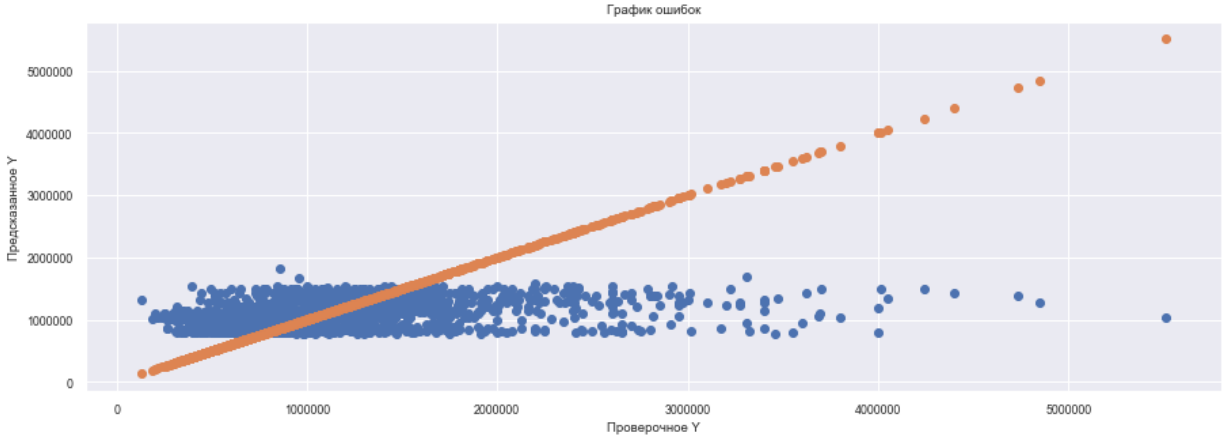
In [342]:
```
df = { u'__Проверочное':  y2.reshape(1,-1)[0],
       u'_Предсказанные': y_pred.reshape(1,-1)[0],
       u'Ошибка':  (y2 - y_pred).reshape(1,-1)[0]}
df = pd.DataFrame(df)
df.head()
```

Out[342]:

	__Проверочное	_Предсказанные	Ошибка
0	1913000.000	1266225.700	646774.300
1	1403000.000	982829.039	420170.961
2	1631000.000	1152867.036	478132.964
3	680000.000	1096187.703	-416187.703
4	1250000.000	846798.642	403201.358

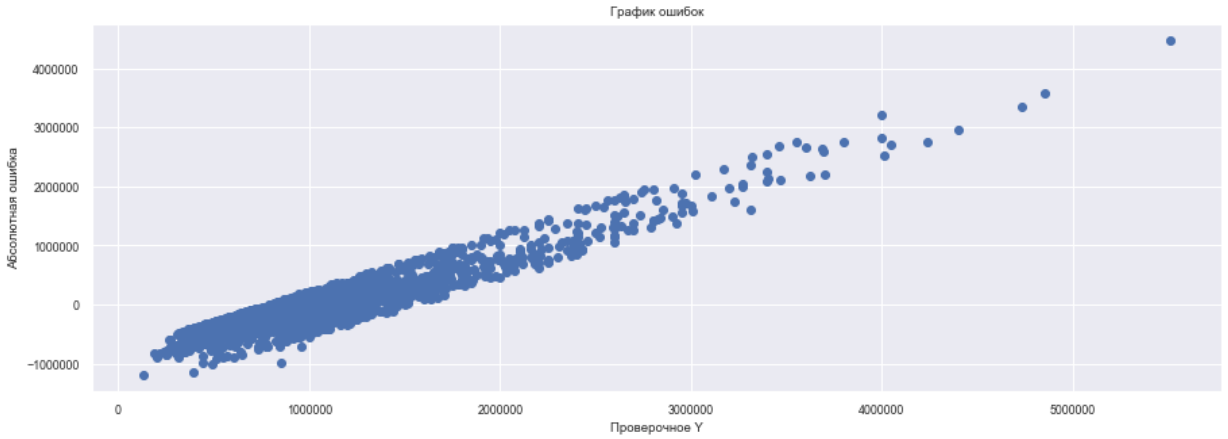
In [343]:
```
plt.scatter(y2,y_pred)
plt.scatter(y2,y2)
plt.title(u'График ошибок');
plt.xlabel(u'Проверочное Y')
plt.ylabel(u'Предсказанное Y')
```

Out[343]:
```
Text(0,0.5,u'\u041f\u0440\u0435\u0434\u0441\u043a\u0430\u0437\u0430\u043d\u043d\u043e\u0435 Y')
```



In [344]:
```
plt.scatter(y2,y2- y_pred)

plt.title(u'График ошибок')
plt.xlabel(u'Проверочное Y')
plt.ylabel(u'Абсолютная ошибка')
```

Out[344]:
```
Text(0,0.5,u'\u0410\u0431\u0441\u043e\u043b\u044e\u0442\u043d\u0430\u044f \u043e\u0448\u0438\u0431\u043a\u0430')
```



3.1.4. Многомерная модель

In [345]:
```
import numpy as np
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.datasets import load_boston
from sklearn.metrics import mean_squared_error, r2_score

bh_data = load_boston()
print(bh_data.keys())
boston = pd.DataFrame(bh_data.data, columns=bh_data.feature_names)

#print(data.describe)
```

```
['filename', 'data', 'target', 'DESCR', 'feature_names']
```

In [346]:
```
lin_reg_mult = LinearRegression()
lin_reg_mult.fit(x_train, y_train)
y_pred = lin_reg_mult.predict(x_test)
```

In [347]:
```
print "y_pred (x) =\n   x1 *{} + {:.0f}\n".format(lin_reg_mult.coef_, lin_reg_mult.intercept_[0])
print "MAE Mean absolute error: {:.0f}".format( mean_absolute_error(y_test, y_pred))
print "MSE Mean squared error: {:.0f}".format( mean_squared_error(y_test, y_pred))
print "RMSE: {:.0f}".format(np.sqrt( mean_squared_error(y_test, y_pred) ))
print "R2 of Linear Regresson (1 is perfect): {:.3f}".format( r2_score(y_test, y_pred))
```

```
y_pred (x) =
   x1 *[[ 3.30103284e+05 -5.12367003e+03  2.02050874e+02]] + 10082741

MAE Mean absolute error: 351151
MSE Mean squared error: 253160763804
RMSE: 503151
R2 of Linear Regresson (1 is perfect): 0.395
```

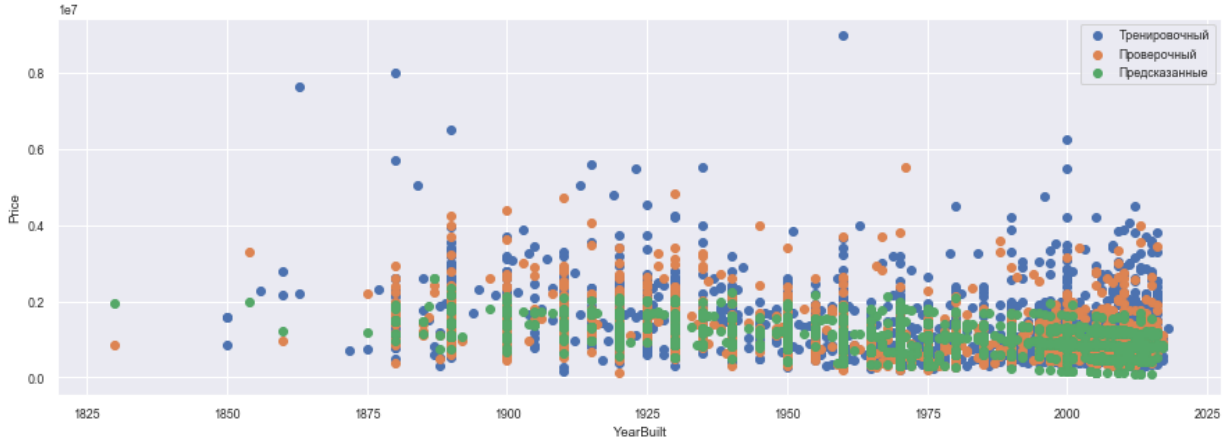
In [348]:
```
(y_test - y_pred).head()
```

Out[348]:

	Price
6324	605179.520
4592	-10308.042
423	391102.221
12022	-435922.764
7424	397100.180

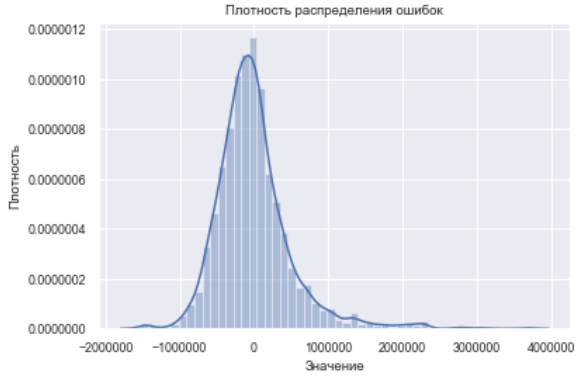
In [349]:
```
plt.figure(figsize(15,5))
fig, ax = plt.subplots()
ax.scatter(x_train['YearBuilt'], y_train, label=u'Тренировочный') # тренировочный
ax.scatter(x_test['YearBuilt'], y_test, label=u'Проверочный') # проверочный
ax.scatter(x_test['YearBuilt'], y_pred, label=u'Предсказанные') # проверочный

ax.legend(loc="best")
ax.set_xlabel('YearBuilt')
ax.set_ylabel(target[0])

plt.show()
```

```
<Figure size 1080x360 with 0 Axes>
```



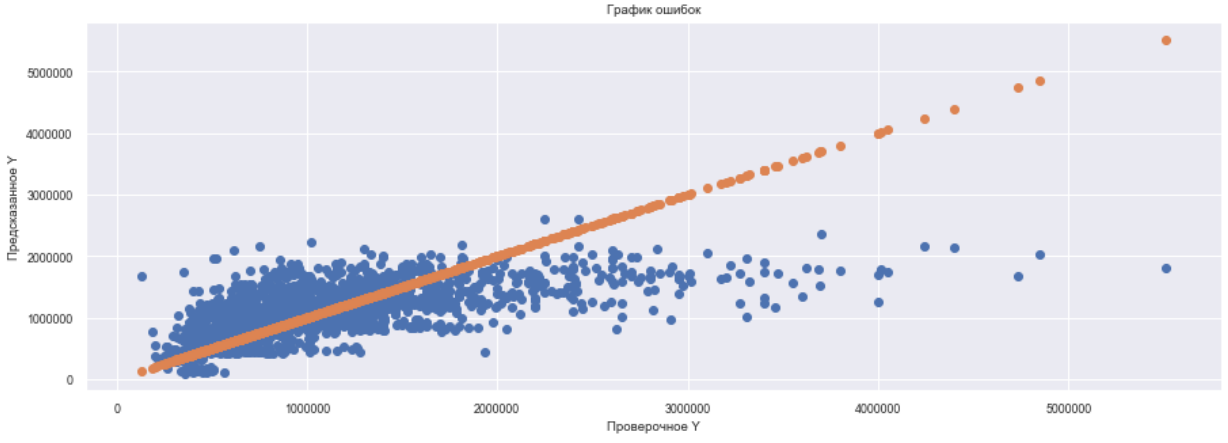
In [350]:
```
plt.figure(figsize=(6,4))
sns.distplot((y_test - y_pred))
plt.title(u'Плотность распределения ошибок');
plt.ylabel(u'Плотность')
plt.xlabel(u'Значение')
```

Out[350]:
```
Text(0.5,0,u'\u0417\u043d\u0430\u0447\u0435\u043d\u0438\u0435')
```



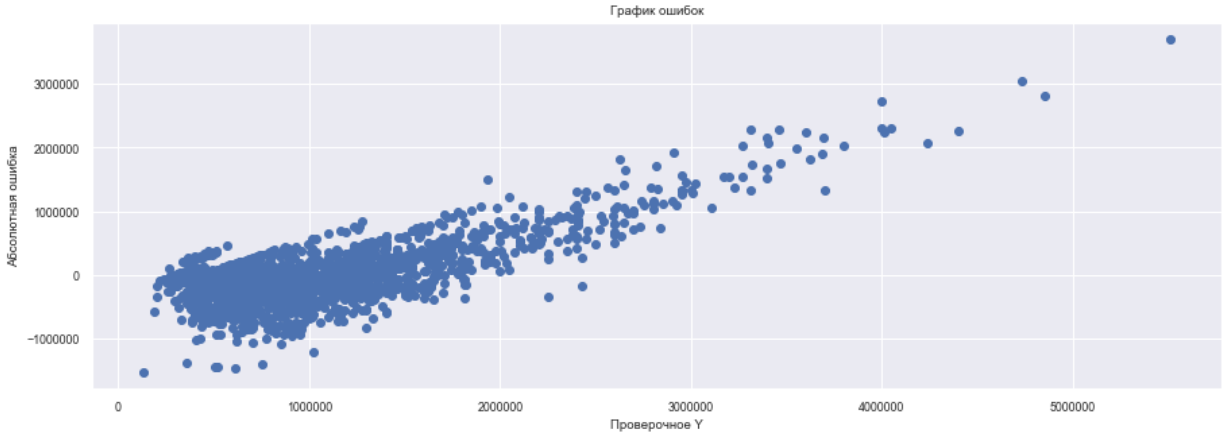
In [351]:
```
plt.scatter(y_test,y_pred)
plt.scatter(y_test,y_test)
plt.title(u'График ошибок');
plt.xlabel(u'Проверочное Y')
plt.ylabel(u'Предсказанное Y')
```

Out[351]:
```
Text(0,0.5,u'\u041f\u0440\u0435\u0434\u0441\u043a\u0430\u0437\u0430\u043d\u043d\u043e\u0435 Y')
```



In [352]:
```
plt.scatter(y_test, (y_test - y_pred) )

plt.title(u'График ошибок')
plt.xlabel(u'Проверочное Y')
plt.ylabel(u'Абсолютная ошибка')
```

Out[352]:
```
Text(0,0.5,u'\u0410\u0431\u0441\u043e\u043b\u044e\u0442\u043d\u0430\u044f \u043e\u0448\u0438\u0431\u043a\u0430')
```



In [353]:
```
import statsmodels.api as sm
sm_x_train = sm.add_constant(x_train) # adding a constant
model = sm.OLS(y_train, sm_x_train)  # OLS - Ordinary Least Squares
results = model.fit()

print('Parameters: {:.0f}', results.params)
print('R2: ', results.rsquared)
print(results.summary())
```

```
('Parameters: {:.0f}', const       10082740.705
Rooms         330103.284
YearBuilt      -5123.670
Landsize         202.051
dtype: float64)
('R2: ', 0.3547634985170308)
                            OLS Regression Results
==============================================================================
Dep. Variable:                  Price   R-squared:                       0.355
Model:                            OLS   Adj. R-squared:                  0.354
Method:                 Least Squares   F-statistic:                     787.7
Date:                Mon, 09 Sep 2019   Prob (F-statistic):               0.00
Time:                        07:36:33   Log-Likelihood:                -62987.
No. Observations:                4302   AIC:                         1.260e+05
Df Residuals:                    4298   BIC:                         1.260e+05
Df Model:                           3
Covariance Type:            nonrobust
==============================================================================
                 coef    std err          t      P>|t|      [0.025      0.975]
------------------------------------------------------------------------------
const       1.008e+07   4.53e+05     22.275      0.000     9.2e+06     1.1e+07
Rooms       3.301e+05   9568.636     34.498      0.000    3.11e+05    3.49e+05
YearBuilt  -5123.6700    229.408    -22.334      0.000   -5573.429   -4673.911
Landsize     202.0509     26.313      7.679      0.000     150.463     253.639
==============================================================================
Omnibus:                     2994.874   Durbin-Watson:                   2.006
Prob(Omnibus):                  0.000   Jarque-Bera (JB):            90777.833
Skew:                           2.919   Prob(JB):                         0.00
Kurtosis:                      24.734   Cond. No.                     1.08e+05
==============================================================================

Warnings:
[1] Standard Errors assume that the covariance matrix of the errors is correctly specified.
[2] The condition number is large, 1.08e+05. This might indicate that there are
strong multicollinearity or other numerical problems.
```

Нелинейная регрессия. Многомерная модель

Линейная регрессия

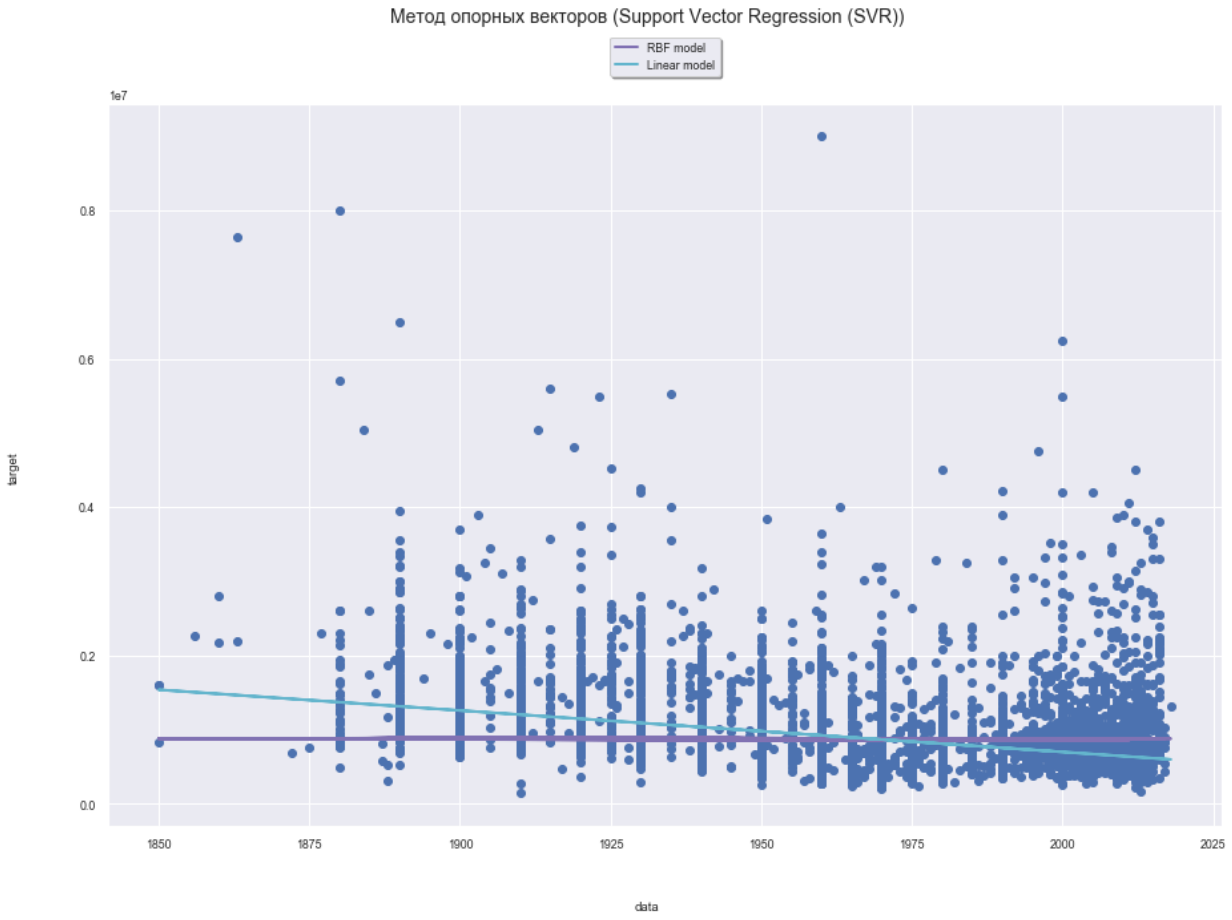
- linear_model.LinearRegression() - МНК метод наименьших средних квадратов
- linear_model.Ridge() - наименьших средних квадратов с регуляризацией
  - пример https://towardsdatascience.com/ridge-regression-for-better-usage-2f19b3a202db (https://towardsdatascience.com/ridge-regression-for-better-usage-2f19b3a202db)
- linear_model.Lasso() Линейная модель обучена с L1 до регуляризатора (он же Лассо)
- Обобщённый метод наименьших квадратов (ОМНК, GLS — англ. Generalized Least Squares)

Метод опорных векторов (Support Vector Regression (SVR))

- Support Vector Regression (SVR) https://scikit-learn.org/stable/auto_examples/svm/plot_svm_regression.html (https://scikit-learn.org/stable/auto_examples/svm/plot_svm_regression.html)

In [354]:
```
#print(__doc__)

import numpy as np
from sklearn.svm import SVR
import matplotlib.pyplot as plt

# ##############################################################################
# Generate sample data
X = x_train[['YearBuilt']]
y = y_train

# ##############################################################################
# Add noise to targets
#y[::5] += 3 * (0.5 - np.random.rand(8))

# ##############################################################################
# Fit regression model
svr_rbf = SVR(kernel='rbf', C=100, gamma=0.1, epsilon=.1)
svr_lin = SVR(kernel='linear', C=100, gamma='auto')
svr_poly = SVR(kernel='poly', C=100, gamma='auto', degree=3, epsilon=.1,
               coef0=1)

# ##############################################################################
# Look at the results
lw = 2


#svrs = [svr_rbf, svr_lin] # , svr_poly]
kernel_label = ['RBF', 'Linear', 'Polynomial']
model_color = ['m', 'c', 'g']
```

In [355]:
```
fig, axes = plt.subplots(nrows=1, ncols=1, figsize=(15, 10), sharey=True)
axes.scatter(X, y)
axes.legend(loc='upper center', bbox_to_anchor=(0.5, 1.1),
                    ncol=1, fancybox=True, shadow=True)

axes.plot(X, svr_rbf.fit(X, y).predict(X), color='m', lw=lw,
                  label='{} model'.format('RBF'))

axes.plot(X, svr_lin.fit(X, y).predict(X), color='c', lw=lw,
                  label='{} model'.format('Linear'))

axes.legend(loc='upper center', bbox_to_anchor=(0.5, 1.1),
                    ncol=1, fancybox=True, shadow=True)

fig.text(0.5, 0.04, 'data', ha='center', va='center')
fig.text(0.06, 0.5, 'target', ha='center', va='center', rotation='vertical')
fig.suptitle(u"Метод опорных векторов (Support Vector Regression (SVR))", fontsize=14)
plt.show()
```



In [ ]:

In [ ]: