17.2) Multiple Regression

Vitor Kamada

December 2019

Reference

Tables, Graphics, and Figures from

Computational and Inferential Thinking: The Foundations of Data Science

Adhikari & DeNero (2019): 17.6 Multiple Regression

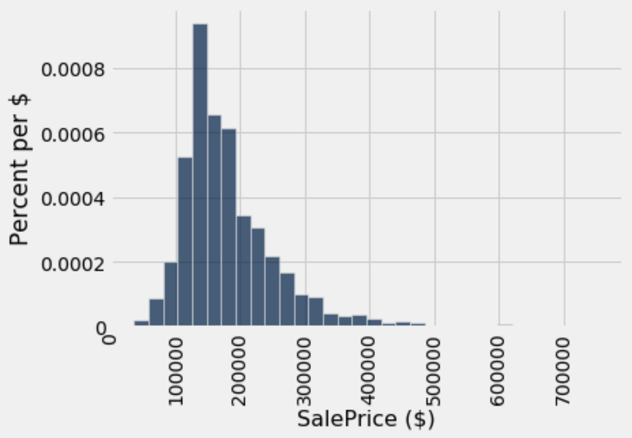
https://www.inferentialthinking.com/

Predicting Home Prices

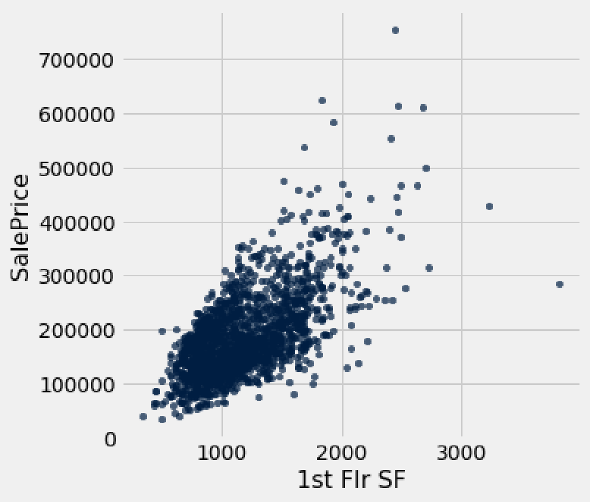
```python
import numpy as np
from datascience import *
path_data = 'https://github.com/data-8/textbook/raw/gh-pages/data/'
all_sales = Table.read_table(path_data + 'house.csv')
sales = all_sales.where('Bldg Type', '1Fam').where('Sale Condition',
    'Normal').select('SalePrice', '1st Flr SF', '2nd Flr SF',
    'Total Bsmt SF', 'Garage Area', 'Wood Deck SF', 'Open Porch SF',
    'Lot Area', 'Year Built', 'Yr Sold')
sales.sort('SalePrice')
```

SalePrice	1st Flr SF	2nd Flr SF	Total Bsmt SF	Garage Area
35000	498	0	498	216
39300	334	0	0	0
40000	649	668	649	250

```
%matplotlib inline
import matplotlib.pyplot as plots
plots.style.use('fivethirtyeight')
sales.hist('SalePrice', bins=32, unit='$')
```



```
sales.scatter('1st Flr SF', 'SalePrice')
```



```python
def standard_units(any_numbers):
    "Convert any array of numbers to standard units."
    return (any_numbers - np.mean(any_numbers))/np.std(any_numbers)

def correlation(t, x, y):
    return np.mean(standard_units(t.column(x))*standard_units(t.column(y)))

for label in sales.labels:
    print('Correlation of', label, 'and SalePrice:\t',
          correlation(sales, label, 'SalePrice'))
```

```
Correlation of SalePrice and SalePrice:  1.0
Correlation of 1st Flr SF and SalePrice:        0.6424662541030225
Correlation of 2nd Flr SF and SalePrice:        0.3575218942800824
Correlation of Total Bsmt SF and SalePrice:     0.652978626757169
Correlation of Garage Area and SalePrice:       0.6385944852520443
Correlation of Wood Deck SF and SalePrice:      0.3526986661950492
Correlation of Open Porch SF and SalePrice:     0.3369094170263733
Correlation of Lot Area and SalePrice:  0.2908234551157694
Correlation of Year Built and SalePrice:        0.5651647537135916
Correlation of Yr Sold and SalePrice:   0.02594857908072111
```

```python
both_floors = sales.column(1) + sales.column(2)
correlation(sales.with_column('Both Floors',
      both_floors), 'SalePrice', 'Both Floors')
```

```
0.782192055613487
```

Training vs Test Dataset

Split the data randomly into a training and test set of equal size

```
train, test = sales.split(1001)
print(train.num_rows, 'training and',
    test.num_rows, 'test instances.')
```

```
1001 training and 1001 test instances.
```

```python
def predict(slopes, row):
    return sum(slopes * np.array(row))

example_row = test.drop('SalePrice').row(0)
print('Predicting sale price for:', example_row)
example_slopes = np.random.normal(10, 1, len(example_row))
print('Using slopes:', example_slopes)
print('Result:', predict(example_slopes, example_row))
```

```
Predicting sale price for: Row(1st Flr SF=897, 2nd
Using slopes: [ 9.33383893 11.10946696  9.71270344
  9.57206353  9.80311921  8.82001355]
Result: 139272.08312238153
```

```python
print('Actual sale price:', test.column('SalePrice').item(0))
print('Predicted sale price using random slopes:',
      predict(example_slopes, example_row))
```

```
Actual sale price: 115000
Predicted sale price using random slopes: 139272.08312238153
```

Root Mean Squared Error (RMSE)

```
train_prices = train.column(0)
train_attributes = train.drop(0)
def rmse(slopes, attributes, prices):
    errors = []
    for i in np.arange(len(prices)):
        predicted = predict(slopes, attributes.row(i))
        actual = prices.item(i)
        errors.append((predicted - actual) ** 2)
    return np.mean(errors) ** 0.5

def rmse_train(slopes):
    return rmse(slopes, train_attributes, train_prices)

print('RMSE of all training examples using random slopes:',
      rmse_train(example_slopes))
```

```
RMSE of all training examples using random slopes: 63970.098907489504
```

Multiple Regression

```
best_slopes = minimize(rmse_train,
    start=example_slopes, smooth=True, array=True)
print('The best slopes for the training set:')
Table(train_attributes.labels).with_row(list(best_slopes)).show()
print('RMSE of all training examples using the best slopes:',
      rmse_train(best_slopes))
```

1st Flr SF	2nd Flr SF	Total Bsmt SF	Garage Area	Wood Deck SF	Open
69.2031	75.186	51.6933	41.7921	33.336	31

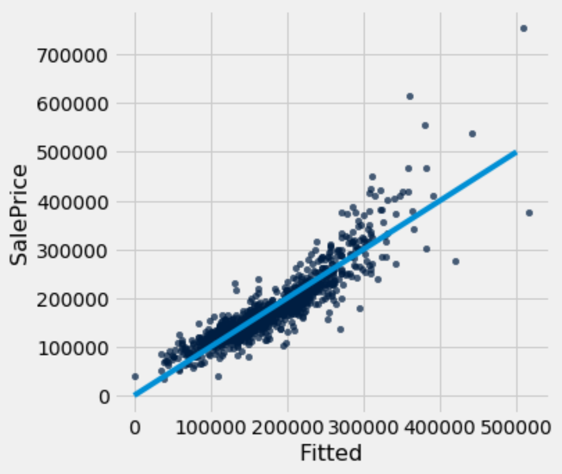
```
RMSE of all training examples using the best slopes: 31074.572851167875
```

Test Dataset

```
test_prices = test.column(0)
test_attributes = test.drop(0)

def rmse_test(slopes):
    return rmse(slopes, test_attributes, test_prices)

rmse_linear = rmse_test(best_slopes)
print('Test set RMSE for multiple linear regression:', rmse_linear)
```

```
Test set RMSE for multiple linear regression: 31876.061362588185
```

```
def fit(row):
    return sum(best_slopes * np.array(row))

test.with_column('Fitted', test.drop(0).apply(fit)).scatter('Fitted', 0)
plots.plot([0, 5e5], [0, 5e5]);
```



```
test.with_column('Residual',
  test_prices-test.drop(0).apply(fit)).scatter(0, 'Residual')
plots.plot([0, 7e5], [0, 0]);
```