LinearRegression

September 23, 2021

SL Lab with Python 2: Linear Regression

Statistical Learning (Sejong University)

Date: 2021.09.23 (By: S. M. Riazul Islam)

In [13]:
```
# Import various Python packages and modules

import pandas as pd
import numpy as np
import matplotlib.pyplot as plt

from mpl_toolkits.mplot3d import axes3d
# used for 3D plotting

import seaborn as sns
# used for data visualization

from sklearn.preprocessing import scale
# Standardize a dataset

import sklearn.linear_model as skl_lm
from sklearn.metrics import mean_squared_error, r2_score


# statmodels
# Python module that provides classes and functions for the estimation
# of many different statistical models, as well as for
# conducting statistical tests, and statistical data exploration
import statsmodels.api as sm
# Cross-sectional models and methods

import statsmodels.formula.api as smf
# A convenience interface for specifying models
```

```
%matplotlib inline
plt.style.use('seaborn-white')
```

Load Datasets: Advertising, Credi Card, and Auto

Datasets are located at C:/DataSL/

```
In [3]: advertising = pd.read_csv('C:/DataSL/Advertising.csv', usecols=[1,2,3,4])
        advertising.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 200 entries, 0 to 199
Data columns (total 4 columns):
TV           200 non-null float64
Radio        200 non-null float64
Newspaper    200 non-null float64
Sales        200 non-null float64
dtypes: float64(4)
memory usage: 6.3 KB
```

```
In [18]: credit = pd.read_csv('C:/DataSL/Credit.csv', usecols=list(range(1,12)))
         credit['Student2'] = credit.Student.map({'No':0, 'Yes':1})
         # Student2: dummy column created

         credit.head(5)
```

Out[18]:

	Income	Limit	Rating	Cards	Age	Education	Gender	Student	Married	Ethnicity	Balance	Student2
0	14.891	3606	283	2	34	11	Male	No	Yes	Caucasian	333	0
1	106.025	6645	483	3	82	15	Female	Yes	Yes	Asian	903	1
2	104.593	7075	514	4	71	11	Male	No	No	Asian	580	0
3	148.924	9504	681	3	36	11	Female	No	No	Asian	964	0
4	55.882	4897	357	2	68	16	Male	No	Yes	Caucasian	331	0

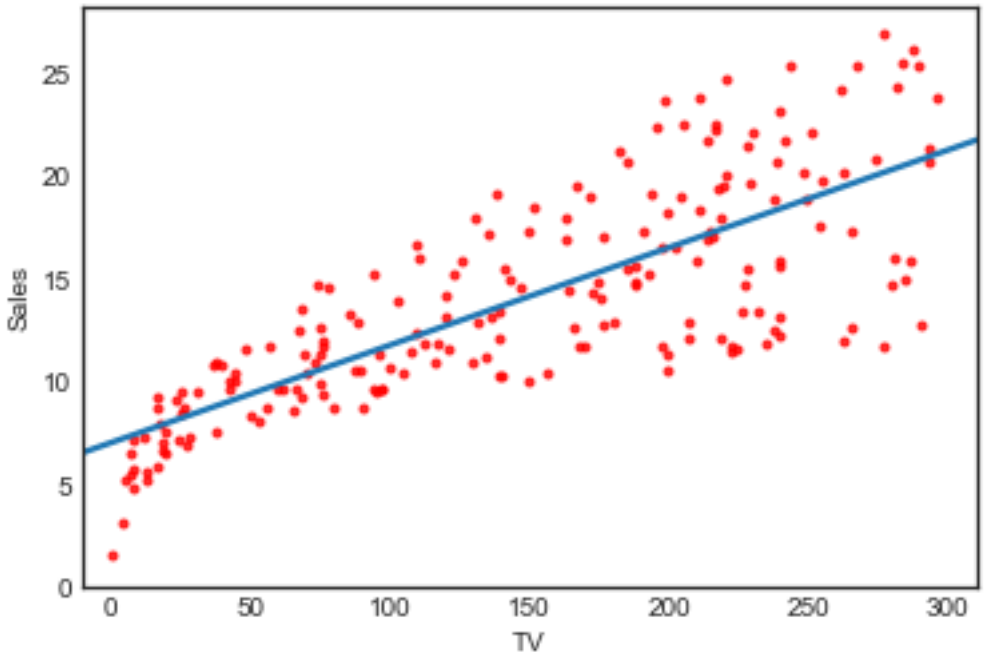
```
In [17]: auto = pd.read_csv('C:/DataSL/Auto.csv', na_values='?').dropna()
         auto.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 392 entries, 0 to 396
Data columns (total 9 columns):
mpg            392 non-null float64
cylinders      392 non-null int64
```

```
displacement    392 non-null float64
horsepower      392 non-null float64
weight          392 non-null int64
acceleration    392 non-null float64
year            392 non-null int64
origin          392 non-null int64
name            392 non-null object
dtypes: float64(4), int64(4), object(1)
memory usage: 30.6+ KB
```

Linear Regression Models

In [14]:
```
sns.regplot(advertising.TV, advertising.Sales, order=1, ci=None, scatter_kws={'color'
# Plot data and a linear regression model fit
# If order is greater than 1, use numpy.polyfit to estimate a polynomial regression
# ciint in [0, 100] or None, optional

plt.xlim(-10,310)
plt.ylim(bottom=0);
```



In [32]:
```
# Regression coefficients (Simple Linear Regression using Least Squares)
regr = skl_lm.LinearRegression()

X = scale(advertising.TV, with_mean=True, with_std=False).reshape(-1,1)
```

```
        # reshape(-1,1) converts 1D X to 2D X of    no-of-rows as it had and 1 column

        y = advertising.Sales

        regr.fit(X,y)
        print(regr.intercept_)
        print(regr.coef_)
```

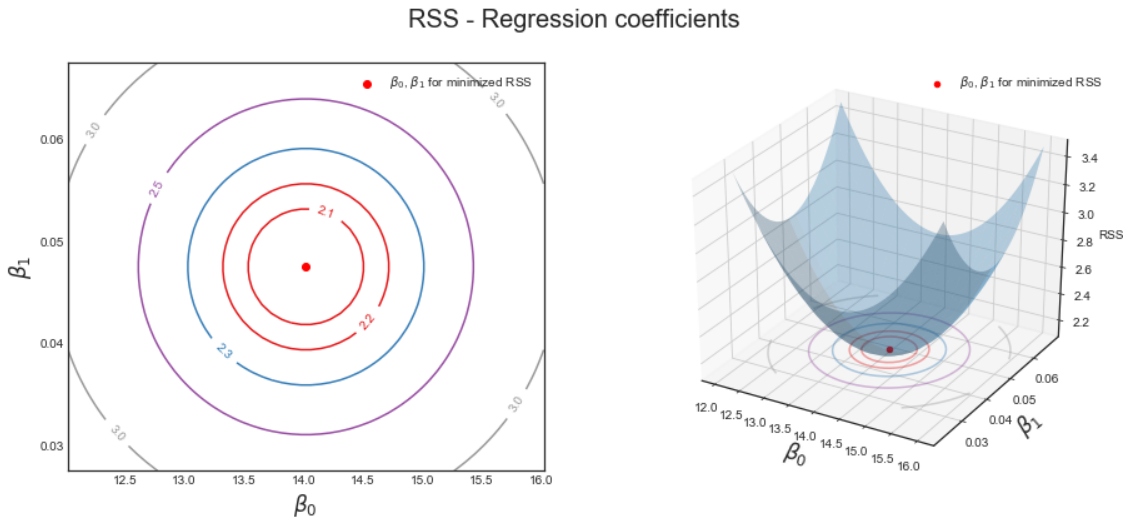
```
14.0225
[0.04753664]
```

In [34]:
```
        # Create grid coordinates for plotting
        B0 = np.linspace(regr.intercept_-2, regr.intercept_+2, 50)
        B1 = np.linspace(regr.coef_-0.02, regr.coef_+0.02, 50)
        xx, yy = np.meshgrid(B0, B1, indexing='xy')
        # create a rectangular grid out of an array of x values and an array of y values.
        # If we like to make a grid where we have a point at each point in both the x and y d


        Z = np.zeros((B0.size,B1.size))

        # Calculate Z-values (RSS) based on grid of coefficients
        for (i,j), v in np.ndenumerate(Z):
            Z[i,j] =((y - (xx[i,j]+X.ravel()*yy[i,j]))**2).sum()/1000

            # xx[i,j: B0
            # yy[i,j: B1
            # ravel: Return a contiguous flattened array.

        # Minimized RSS
        min_RSS = r'$\beta_0$, $\beta_1$ for minimized RSS'
        min_rss = np.sum((regr.intercept_+regr.coef_*X - y.values.reshape(-1,1))**2)/1000
        min_rss
```

Out[34]:
```
2.1025305831313514
```

In [31]:
```
        fig = plt.figure(figsize=(15,6))
        fig.suptitle('RSS - Regression coefficients', fontsize=20)

        ax1 = fig.add_subplot(121)
        ax2 = fig.add_subplot(122, projection='3d')

        # Left plot
        CS = ax1.contour(xx, yy, Z, cmap=plt.cm.Set1, levels=[2.15, 2.2, 2.3, 2.5, 3])
        ax1.scatter(regr.intercept_, regr.coef_[0], c='r', label=min_RSS)
        ax1.clabel(CS, inline=True, fontsize=10, fmt='%1.1f')

        # Learn how to Choosing Colormaps in Matplotlib
```

```
# Learn how to Choosing Colormaps in Matplotlib


# Right plot
ax2.plot_surface(xx, yy, Z, rstride=3, cstride=3, alpha=0.3)
ax2.contour(xx, yy, Z, zdir='z', offset=Z.min(), cmap=plt.cm.Set1,
            alpha=0.4, levels=[2.15, 2.2, 2.3, 2.5, 3])
ax2.scatter3D(regr.intercept_, regr.coef_[0], min_rss, c='r', label=min_RSS)
ax2.set_zlabel('RSS')
ax2.set_zlim(Z.min(),Z.max())
ax2.set_ylim(0.02,0.07)

# settings common to both plots
for ax in fig.axes:
    ax.set_xlabel(r'$\beta_0$', fontsize=17)
    ax.set_ylabel(r'$\beta_1$', fontsize=17)
    ax.set_yticks([0.03,0.04,0.05,0.06])
    ax.legend()
```



Confidence Intervals

In [40]:
```
# Table 3.1
est = smf.ols('Sales ~ TV', advertising).fit()
est.summary().tables[1]  ## Check tables[0] and tables[2]
```

Out[40]:
```
<class 'statsmodels.iolib.table.SimpleTable'>
```

In [36]:
```
# RSS with regression coefficients
((advertising.Sales - (est.params[0] + est.params[1]*advertising.TV))**2).sum()/1000
```

Out[36]:
```
2.1025305831313514
```

Table 3.1 and 3.2 using Scikit-learn

R-squared

```
In [41]: regr = skl_lm.LinearRegression()

         X = advertising.TV.values.reshape(-1,1)
         y = advertising.Sales

         regr.fit(X,y)
         print(regr.intercept_)
         print(regr.coef_)
```

```
7.032593549127693
[0.04753664]
```

```
In [42]: Sales_pred = regr.predict(X)
         r2_score(y, Sales_pred)
```

```
Out[42]: 0.611875050850071
```

Multiple Linear Regression

Table 3.3

```
In [43]: est = smf.ols('Sales ~ Radio', advertising).fit()
         est.summary().tables[1]
```

```
Out[43]: <class 'statsmodels.iolib.table.SimpleTable'>
```

```
In [44]: est = smf.ols('Sales ~ Newspaper', advertising).fit()
         est.summary().tables[1]
```

```
Out[44]: <class 'statsmodels.iolib.table.SimpleTable'>
```

Table 3.4

```
In [46]: est = smf.ols('Sales ~ TV + Radio + Newspaper', advertising).fit()
         est.summary().tables[1]
```

```
Out[46]: <class 'statsmodels.iolib.table.SimpleTable'>
```

Table 3.5

```
In [49]: # Correlation Matrix
         advertising.corr()
```

```
Out[49]:                  TV     Radio  Newspaper     Sales
         TV         1.000000  0.054809   0.056648  0.782224
         Radio      0.054809  1.000000   0.354104  0.576223
         Newspaper  0.056648  0.354104   1.000000  0.228299
         Sales      0.782224  0.576223   0.228299  1.000000
```

Figure 3.5 - Multiple Linear Regression

```
In [50]: regr = skl_lm.LinearRegression()

         X = advertising[['Radio', 'TV']].as_matrix()
         y = advertising.Sales

         regr.fit(X,y)
         print(regr.coef_)
         print(regr.intercept_)
```

```
C:\Users\Preload\Anaconda3\lib\site-packages\ipykernel_launcher.py:3: FutureWarning: Method .as
  This is separate from the ipykernel package so we can avoid doing imports until


[0.18799423 0.04575482]
2.9210999124051398
```

```
In [51]: # What are the min/max values of Radio & TV?
         # Use these values to set up the grid for plotting.
         advertising[['Radio', 'TV']].describe()
```

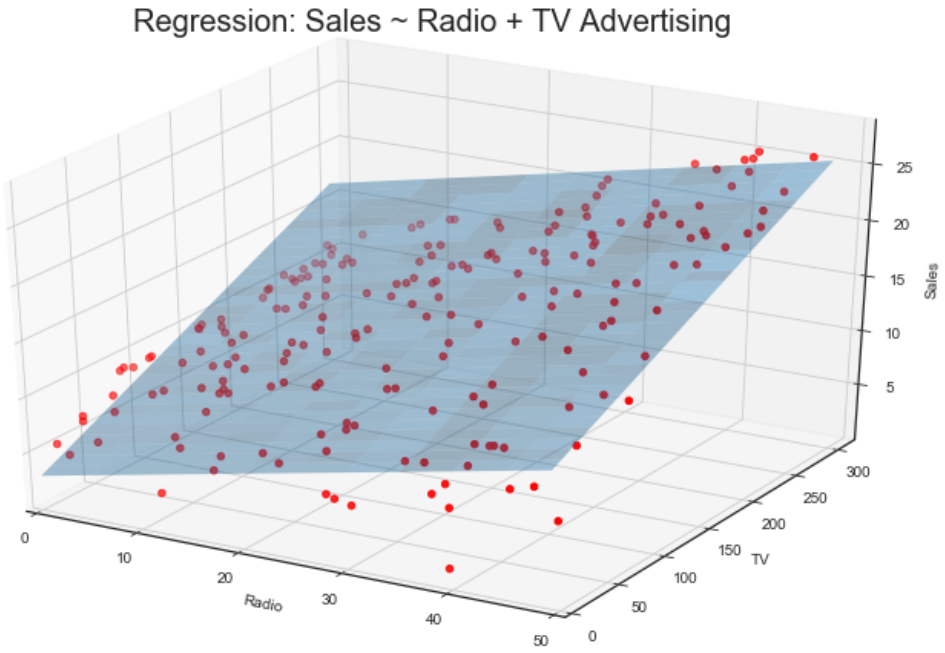
Out[51]:

	Radio	TV
count	200.000000	200.000000
mean	23.264000	147.042500
std	14.846809	85.854236
min	0.000000	0.700000
25%	9.975000	74.375000
50%	22.900000	149.750000
75%	36.525000	218.825000
max	49.600000	296.400000

```
In [52]: # Create a coordinate grid
         Radio = np.arange(0,50)
         TV = np.arange(0,300)

         B1, B2 = np.meshgrid(Radio, TV, indexing='xy')
         Z = np.zeros((TV.size, Radio.size))

         for (i,j),v in np.ndenumerate(Z):
                 Z[i,j] =(regr.intercept_ + B1[i,j]*regr.coef_[0] + B2[i,j]*regr.coef_[1])

In [53]: # Create plot
         fig = plt.figure(figsize=(10,6))
         fig.suptitle('Regression: Sales ~ Radio + TV Advertising', fontsize=20)

         ax = axes3d.Axes3D(fig)
```

```
        ax.plot_surface(B1, B2, Z, rstride=10, cstride=5, alpha=0.4)
        ax.scatter3D(advertising.Radio, advertising.TV, advertising.Sales, c='r')

        ax.set_xlabel('Radio')
        ax.set_xlim(0,50)
        ax.set_ylabel('TV')
        ax.set_ylim(ymin=0)
        ax.set_zlabel('Sales');
```

```
C:\Users\Preload\Anaconda3\lib\site-packages\mpl_toolkits\mplot3d\axes3d.py:671: MatplotlibDepr
The `ymin` argument was deprecated in Matplotlib 3.0 and will be removed in 3.2. Use `bottom` i
  alternative='`bottom`', obj_type='argument')
```



3.3 Other Considerations in the Regression Model

Fig. 3.6

```
In [54]: sns.pairplot(credit[['Balance','Age','Cards','Education','Income','Limit','Rating']])
```

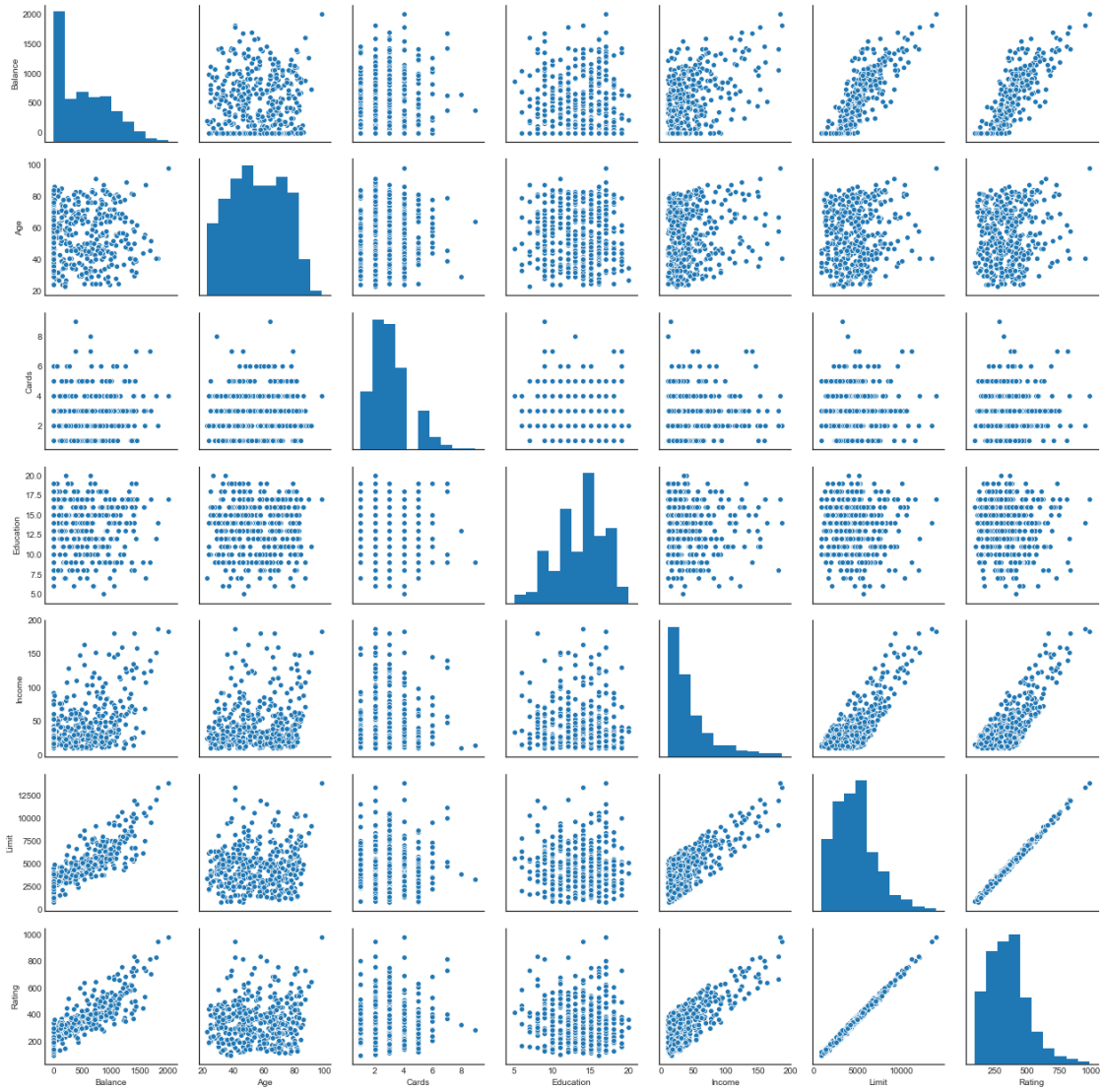


Table 3.7

```
In [56]: est = smf.ols('Balance ~ Gender', credit).fit()
         est.summary().tables[1]
```

```
Out[56]: <class 'statsmodels.iolib.table.SimpleTable'>
```

Table 3.8

```
In [57]: est = smf.ols('Balance ~ Ethnicity', credit).fit()
         est.summary().tables[1]
```

```
Out[57]: <class 'statsmodels.iolib.table.SimpleTable'>
```

Table 3.9 - Interaction Variables

```
In [58]: est = smf.ols('Sales ~ TV + Radio + TV*Radio', advertising).fit()
         est.summary().tables[1]

Out[58]: <class 'statsmodels.iolib.table.SimpleTable'>
```

Figure 3.7 - Interaction between qualitative and quantative variables

```
In [59]: est1 = smf.ols('Balance ~ Income + Student2', credit).fit()
         regr1 = est1.params
         est2 = smf.ols('Balance ~ Income + Income*Student2', credit).fit()
         regr2 = est2.params

         print('Regression 1 - without interaction term')
         print(regr1)
         print('\nRegression 2 - with interaction term')
         print(regr2)
```

```
Regression 1 - without interaction term
Intercept    211.142964
Income         5.984336
Student2     382.670539
dtype: float64

Regression 2 - with interaction term
Intercept          200.623153
Income               6.218169
Student2           476.675843
Income:Student2     -1.999151
dtype: float64
```

```
In [60]: # Income (x-axis)
         income = np.linspace(0,150)

         # Balance without interaction term (y-axis)
         student1 = np.linspace(regr1['Intercept']+regr1['Student2'],
                                regr1['Intercept']+regr1['Student2']+150*regr1['Income'])
         non_student1 =  np.linspace(regr1['Intercept'], regr1['Intercept']+150*regr1['Income']

         # Balance with iteraction term (y-axis)
         student2 = np.linspace(regr2['Intercept']+regr2['Student2'],
                                regr2['Intercept']+regr2['Student2']+
                                150*(regr2['Income']+regr2['Income:Student2']))
         non_student2 =  np.linspace(regr2['Intercept'], regr2['Intercept']+150*regr2['Income']

         # Create plot
         fig, (ax1,ax2) = plt.subplots(1,2, figsize=(12,5))
```

```
        ax1.plot(income, student1, 'r', income, non_student1, 'k')
        ax2.plot(income, student2, 'r', income, non_student2, 'k')

        for ax in fig.axes:
            ax.legend(['student', 'non-student'], loc=2)
            ax.set_xlabel('Income')
            ax.set_ylabel('Balance')
            ax.set_ylim(ymax=1550)
```

```
C:\Users\Preload\Anaconda3\lib\site-packages\matplotlib\axes\_base.py:3610: MatplotlibDeprecati
The `ymax` argument was deprecated in Matplotlib 3.0 and will be removed in 3.2. Use `top` inst
  alternative='`top`', obj_type='argument')
C:\Users\Preload\Anaconda3\lib\site-packages\matplotlib\axes\_base.py:3610: MatplotlibDeprecati
The `ymax` argument was deprecated in Matplotlib 3.0 and will be removed in 3.2. Use `top` inst
  alternative='`top`', obj_type='argument')
```

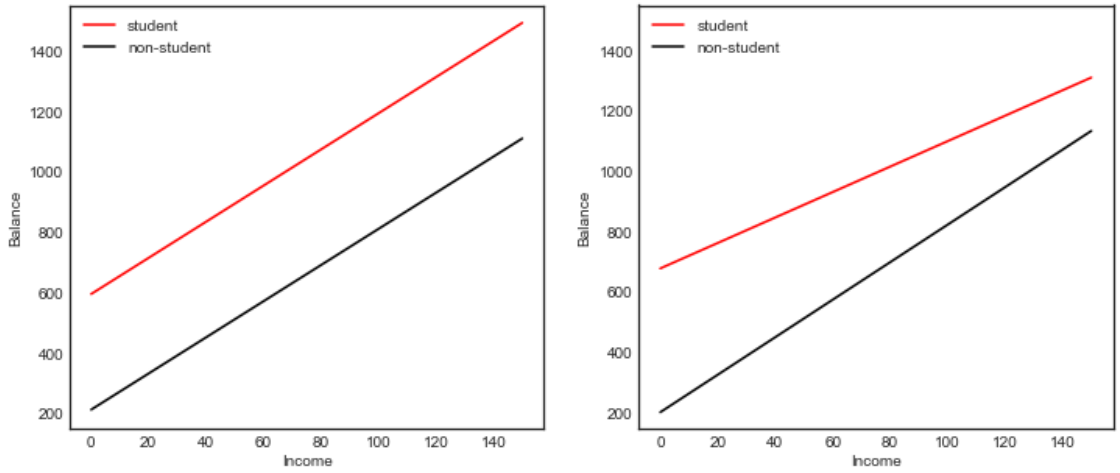


Figure 3.8 - Non-linear relationships

```
In [61]: # With Seaborn's regplot() you can easily plot higher order polynomials.
         plt.scatter(auto.horsepower, auto.mpg, facecolors='None', edgecolors='k', alpha=.5)
         sns.regplot(auto.horsepower, auto.mpg, ci=None, label='Linear', scatter=False, color=
         sns.regplot(auto.horsepower, auto.mpg, ci=None, label='Degree 2', order=2, scatter=Fa
         sns.regplot(auto.horsepower, auto.mpg, ci=None, label='Degree 5', order=5, scatter=Fa
         plt.legend()
         plt.ylim(5,55)
         plt.xlim(40,240);
```

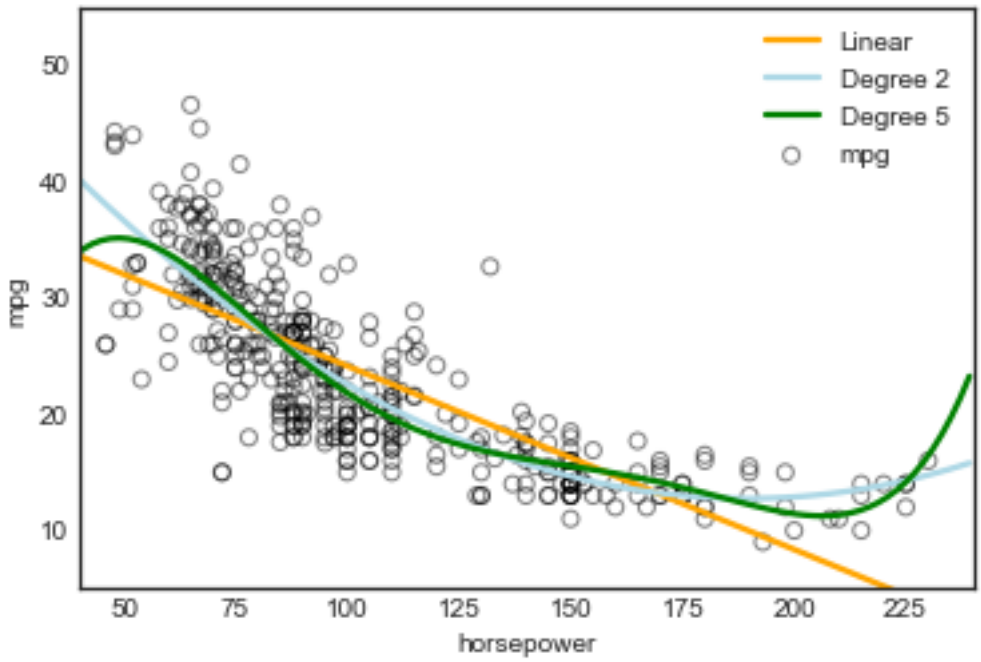


Table 3.10

```
In [62]: auto['horsepower2'] = auto.horsepower**2
         auto.head(3)
```

```
Out[62]:     mpg  cylinders  displacement  horsepower  weight  acceleration  year  \
         0  18.0          8         307.0       130.0    3504          12.0    70
         1  15.0          8         350.0       165.0    3693          11.5    70
         2  18.0          8         318.0       150.0    3436          11.0    70

            origin                       name  horsepower2
         0       1  chevrolet chevelle malibu      16900.0
         1       1          buick skylark 320      27225.0
         2       1         plymouth satellite      22500.0
```

```
In [63]: est = smf.ols('mpg ~ horsepower + horsepower2', auto).fit()
         est.summary().tables[1]
```

```
Out[63]: <class 'statsmodels.iolib.table.SimpleTable'>
```

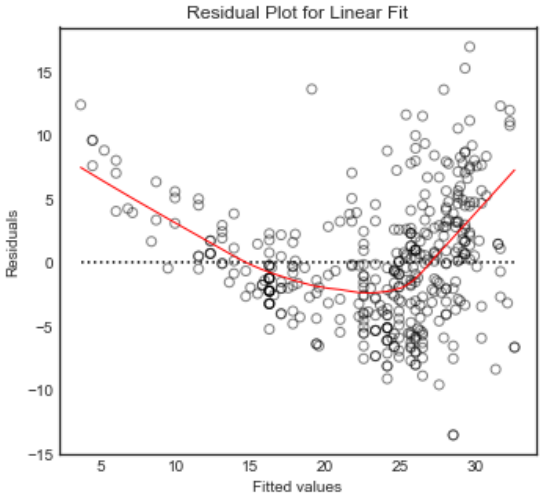
Figure 3.9

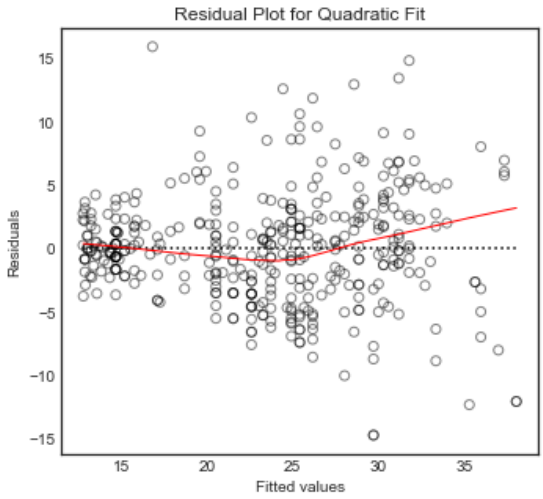
```
In [65]: regr = skl_lm.LinearRegression()

         # Linear fit
         X = auto.horsepower.values.reshape(-1,1)
```

```
y = auto.mpg
regr.fit(X, y)

auto['pred1'] = regr.predict(X)
auto['resid1'] = auto.mpg - auto.pred1

# Quadratic fit
X2 = auto[['horsepower', 'horsepower2']].as_matrix()
regr.fit(X2, y)

auto['pred2'] = regr.predict(X2)
auto['resid2'] = auto.mpg - auto.pred2
```

```
C:\Users\Preload\Anaconda3\lib\site-packages\ipykernel_launcher.py:12: FutureWarning: Method .a
  if sys.path[0] == '':
```

In [66]:
```
fig, (ax1,ax2) = plt.subplots(1,2, figsize=(12,5))

# Left plot
sns.regplot(auto.pred1, auto.resid1, lowess=True,
            ax=ax1, line_kws={'color':'r', 'lw':1},
            scatter_kws={'facecolors':'None', 'edgecolors':'k', 'alpha':0.5})
ax1.hlines(0,xmin=ax1.xaxis.get_data_interval()[0],
           xmax=ax1.xaxis.get_data_interval()[1], linestyles='dotted')
ax1.set_title('Residual Plot for Linear Fit')

# Right plot
sns.regplot(auto.pred2, auto.resid2, lowess=True,
            line_kws={'color':'r', 'lw':1}, ax=ax2,
            scatter_kws={'facecolors':'None', 'edgecolors':'k', 'alpha':0.5})
ax2.hlines(0,xmin=ax2.xaxis.get_data_interval()[0],
           xmax=ax2.xaxis.get_data_interval()[1], linestyles='dotted')
ax2.set_title('Residual Plot for Quadratic Fit')

for ax in fig.axes:
    ax.set_xlabel('Fitted values')
    ax.set_ylabel('Residuals')
```


Residual Plot for Linear Fit
Residuals
Fitted values
15
10
5
0
−5
−10
−15
5
10
15
20
25
30


Residual Plot for Quadratic Fit
Residuals
Fitted values
15
10
5
0
−5
−10
−15
15
20
25
30
35