实验介绍

1.实验内容

本实验首先介绍一元线性回归的基本思想，然后给出其背后的数学原理。

接着给出两个基于一元线性回归预测房价的示例，

> 第一个示例中，使用一元线性回归预测汽车的重量和每加仑油量可行驶距离；
> 第二个示例中，使用一元线性回归根据房屋面积预测房屋价格。

在这两个示例中，你将综合运Python，pandas，numpy，matplotlib，sklearn。

2.实验目标

通过本实验掌握一元线性回归的基本思想，了解其背后的数学原理。

可以将一元线性回归应用到真实世界中的预测问题中。

学会numpy，matplotlib以及sklearn库的使用。

3.实验知识点

- 一元线性回归基本思想
- 一元线性回归数学原理
- 一元线性回归的应用
- numpy，matplotlib，pandas，sklearn的使用

4.实验环境

- python 3.6.5
- numpy 1.14.5
- matplotlib 2.2.2
- sklearn 0.19.1
- pandas 0.23.1

5.预备知识

- 微积分，偏微分（非必须）
- Linux命令基本操作
- Python编程基础

注意：对一元线性回归背后的数学原理不用达到掌握的层次，事实上，在了解一元线性回归的基本思想后，利用sklearn等工具就可以很便捷的解决相关问题了。

实验原理

1. 回归与机器学习

机器学习监督学习算法分为分类和回归两种，其实就是根据类别标签分布类型而定义的，分类算法用于离散型分布预测，如kNN、决策树、朴素贝叶斯、Adaboost、SVM、Logistic回归都是分类算法；回归算法用于连续型分布预测，针对的是数值型的样本。回归也是统计学中最有力的工具之一，使用回归，可以在给定输入后给出预测的数值。回归的目的就是建立一个

回归方程用来预测目标值，回归的求解就是求这个回归方程的回归系数。在回归方程确定后，预测结果很简单，将输入值带入回归方程即可。

机器学习中的参数模型指的是：通过建模找到一个最大程度拟合数据的模型，通过确定损失函数（或效用函数，统称为目标函数），最优化目标函数来获得机器学习模型的参数。回归分析指的是利用已知的数据，最大程度的拟合样本与输出标记，即产生拟合方程，从而对未知的数据进行预测。在一元线性回归模型中，需要找到一条可以最大程度拟合输入和输出的直线。

2. 一元线性回归

一元线性回归是一种简单的模型，但应用广泛，比如简单地预测商品价格、成本评估等，都可以用一元线性模型。比如：告诉你(1,1),(2,2),(3,3)，那么问你(4,?)是多少，很容易复原出来(4,4)，这就是一元线性回归问题的求解。一元线性回归是数据挖掘的基础模型，其中包含了非常重要的数学回归的概念，是学习多元回归，广义线性回归的基础。

若X与y之间存在着较强的相关关系，则有：

$$y=a+bX$$

若a与b的值已知，则给出相应的X值，我们可以得到相应的y的预测值。一元线性回归涉及的参数及含义如下：

* 截距项a
* 斜率b

一元线性回归的问题求解就是求解参数a，b。

3. Sklearn简介

Scikit learn 也简称sklearn，是机器学习领域当中最知名的python模块之一。sklearn包含了很多机器学习的方式：

- Classification 分类
- Regression 回归
- Clustering 非监督分类
- Dimensionality reduction 数据降维
- Model Selection 模型选择
- Preprocessing 数据与处理

使用sklearn可以很方便地让我们实现一个机器学习算法。一个复杂度算法的实现，使用sklearn可能只需要调用几行API即可。所以学习sklearn，可以有效减少我们特定任务的实现周期。sklearn对数据挖掘和机器学习的各类算法进行了较好的封装，基本可以使用fit、predict、score来训练、预测、评价模型。

准备工作

点击屏幕右上方的下载实验数据模块，选择下载linear_regression.tgz到指定目录下，然后再依次选择点击上方的File->Open->Upload,上传刚才下载的数据集压缩包，再使用如下命令解压：

In [1]:
```
# !tar -zxvf linear_regression.tgz
```

```
linear_regression/
linear_regression/auto_mpg.csv
linear_regression/auto-mpg.data
linear_regression/price_info.csv
```

一元线性回归的数学原理

对于一元线性回归来说，可以看成Y的值是随着X的值变化，每一个实际的X都会有一个实际的Y值，我们叫Y实际，那么我们要求出一条直线，每一个实际的X都会有一个直线预测的Y值，我们叫做Y预测，回归线使得每个Y 的实际值与预测值之差的平方和最小，即（Y1实际-Y1预测）^ 2+（Y2实际-Y2预测）^ 2+ …… +（Yn实际-Yn预测）^ 2的和最小。现在来实际求一下这条线：

$$Q(a,b)=\sum_{i=1}^{n}(Yi-(aXi+b))^2$$

我们都知道直线在坐标系可以表示为Y=aX+b，所以（Y实际-Y预测）就可以写成（Y实际-（aX实际+b））， 于是平方和可以写成a和b的函数。只需要求出让Q最小的a和b的值，那么回归线的也就求出来了。现在说明函数最小值怎么求：首先，一元函数最小值点的导数为零，比如说Y=X^2，X^2的导数是2X，令2X=0，求得X=0的时候，Y取最小值。那么实质上二元函数也是一样可以类推。不妨把二元函数图象设想成一个曲面，最小值想象成一个凹陷，那么在这个凹陷底部，从任意方向上看，偏导数都是0。因此，对于函数Q，分别对于a和b求偏导数，然后令偏导数等于0，就可以得到一个关于a和b的二元方程组，就可以求出a和b了。这个方法被称为最小二乘法。下面是具体的数学演算过程，。先把Q函数表达式展开：

$$
\begin{aligned}
&Q(a,b)=\sum_{i=1}^{n}(Yi-(aXi+b))^2=(Y_1-(aX_1+b))^2+(Y_2-(aX_2+b))^2+.....+(Yn-(aXn+b))^2\\
&=[Y_1^2-2Y_1(aX_1+b)+(aX_1+b)^2]+[Y_2^2-2Y_2(aX_2+b)+(aX_2+b)^2]+....+[Y_n^2-2Y_n(aX_n+b)+(aX_n+b)^2]\\
&=Y_1^2-2Y_1aX_1-2Y_1b+a^2X_1^2+2aX_1b+b^2\\
&+Y_2^2-2Y_2aX_2-2Y_2b+a^2X_2^2+2aX_2b+b^2\\
&+.....\\
&+Y_n^2-2Y_naX_n-2Y_nb+a^2X_n^2+2aX_nb+b^2\\
&=(Y_1^2+...+Y_n^2)-2a(X_1Y_1+...+XnYn)-2b(Y_1+...+Yn)+a^2(X_1^2+...+X_n^2)+2ab(X_1+..+X_n)+nb^2
\end{aligned}
$$

然后利用平均数，把上面式子中每个括号里的内容进一步化简。例如Y^2的平均表示如下：

$$(Y_1^2+...+Y_n^2)/n=\overline{Y^2}$$

则：

$$(Y_1^2+...+Y_n^2)=n\overline{Y^2}$$

上式子两边×n，于是Q最终化简结果为：

$$Q(a,b)=n\overline{Y^2}-2an\overline{XY}-2bn\overline{Y}+a^2n\overline{X^2}+2abn\overline{X}+nb^2$$

然后分别对Q求a的偏导数和b的偏导数，令偏导数等于0：

$$
\begin{aligned}
\frac{\partial Q}{\partial a}&=-2n\overline{XY}+2an\overline{X^2}+2bn\overline{X}=0\\
\frac{\partial Q}{\partial b}&=-2n\overline{Y}+2an\overline{X}+2nb=0
\end{aligned}
$$

进一步化简，可以消掉2n，最后得到关于a，b的二元方程组为：

$$
\begin{aligned}
-\overline{XY}+a\overline{X^2}+b\overline{X}&=0\\
-\overline{Y}+a\overline{X}+b&=0
\end{aligned}
$$

最后得出a和b的求解公式如下：

$$
\begin{aligned}
a&=\frac{\overline{X}\,\overline{Y}-\overline{XY}}{(\overline{X})^2-\overline{X^2}}\\
b&=\overline{Y}-a\overline{X}
\end{aligned}
$$

引言

1. 项目文件结构

实验目录下有3个文件：

price_info.csv
auto-mpg.data
auto_mpg.csv

其中：price_info.csv为房屋价格预测的数据集；auto-mpg.data为车辆行驶距离预测的数据集，auto_mpg.csv是该数据集的csv版本。

2. 开始实验

接下来的实验步骤将引导你学习一元线性回归的相关知识。

首先引入我们系统需要用到的Python的相关库。

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

sklearn实现简单的一元线性回归

这里，我们首先用一些模拟数据做一元线性回归，首先我们使用方程y=2x+1作为参考曲线，在输出给定计算结果时，加一些随机干扰，让结果距离曲线有点距离。

输入x分别为：1, 2, 3, 4, 5

输出y应该为：3, 5, 7, 9, 11

添加干扰后y的值为：3.1, 5.2, 6.8, 8.8, 11.1

接着我们使用sklearn的线性回归类对x和y进行拟合，sklearn中线性回归类的原型为：

LinearRegression(fit_intercept=True,normalize=False,copy_X=True,n_jobs=1)

其中各个参数的含义如下：

- fit_intercept:是否有截据，如果没有，则直线过原点。默认为有截距。
- normalize:是否将数据归一化，默认是不做归一化。
- copy_X:默认为True，当为True时，X会被copied,否则X将会被覆写。
- n_jobs:默认值为1，计算时使用的核数。

一般使用时，就用默认参数就可以。输入以下内容：

In [2]:
```
#设定X向量
x = [[1], [2], [3], [4], [5]]
#设定Y向量
y = [[3.1], [5.2], [6.8], [8.8], [11.1]]
#创建线性回归模型
model = LinearRegression()
#使用X和Y进行拟合
model.fit(x, y)
```

Out[2]:
```
LinearRegression()
```
**In a Jupyter environment, please rerun this cell to show the HTML representation or trust the notebook.
On GitHub, the HTML representation is unable to render, please try loading this page with nbviewer.org.**

In [3]:
```
#输出线性方程的斜率，即b的值
model.coef_
```

Out[3]:
```
array([[1.96]])
```

In [4]:
```
#输出线性方程的截距，即a的值
model.intercept_
```

Out[4]:
```
array([1.12])
```

In [3]:
```
#在x=6时，预测y的值
predicted = model.predict([[6]])[0]
#输出y的结果
predicted
```

```
[[ 1.96]]
[ 1.12]
[ 12.88]
```

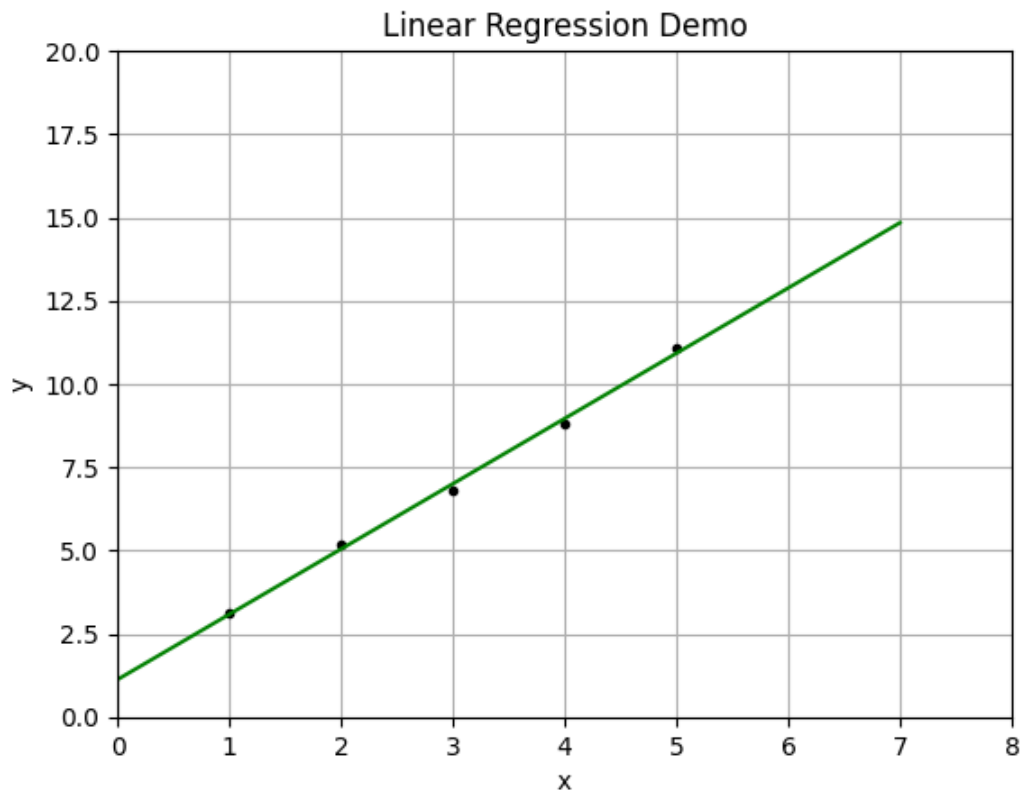
LinearRegression将方程分为两个部分存放，coef_存放回归系数，intercept_则存放截距，因此要查看方程，就是查看这两个变量的取值。

上述代码给出了斜率(1.96)、截距(1.12)、以及输入值为6时的预测结果（12.88）。

接着，我们使用matplotlib将X和Y构成的散点以及表示回归方程的直线绘制出来，可以更加直观的看到线性回归的结果以及各个三点到直线的偏移量。

绘制图像的代码如下所示，以绘制出图像。

In [5]:
```
#启动一个新的图像实例
plt.figure()
#设置图像标题
plt.title('Linear Regression Demo')
#设定坐标轴的X轴和Y轴的标题
plt.xlabel('x')
plt.ylabel('y')
plt.axis([0, 8, 0, 20])
#设定网格开关为显示
plt.grid(True)
#绘制由X向量和Y向量构成的散点图，参数"k."中，其中k表示颜色为卡其色，点表示散点图
plt.plot(x, y, 'k.')
#绘制回归方程的直线，将x的范围设定为0到7，y的值由x进行计算
x1 = [[0], [1], [2], [3], [4], [5], [6], [7]]
y1 = model.predict(x1)
#绘制该直线，参数“g-”表示绘制时颜色选择绿色，-表示点与点直接使用实线连接
plt.plot(x1, y1, 'g-')
#显示图像
plt.show()
```



numpy计算一元线性回归参数

事实上，我们也可以基于numpy计算出一元线性回归方程中参数b和a的值，其数学原理如下：
1.向量x的方差为：

$$var(x) = \frac{\sum_{i=1}^{n}(x_i - \bar{x})^2}{n-1}$$

2.向量x和向量y的协方差为：

$$cov(x,y) = \frac{\sum_{i=1}^{n}(x_i - \bar{x})(y_i - \bar{y})}{n-1}$$

3.由第三小节中，我们知道b的计算方式为：

$$b = \frac{\sum_{i=1}^{n}(x_i - \bar{x})(y_i - \bar{y})}{\sum_{i=1}^{n}(x_i - \bar{x})^2}$$

所以b可以由cov(x,y)/var(x)得到，在知道b的值以后，根据以下公式：

$$a = \bar{y} - b\bar{x}$$

可以计算出a的值，从而求出线性方程的参数。现在我们基于numpy求解线性方程的参数，添加以下代码：

In [5]:
```
#计算X方差
var_x = np.var([1, 2, 3, 4, 5], ddof=1)
#计算X和Y的协方差
cov = np.cov([1, 2, 3, 4, 5], [3.1, 5.2, 6.8, 8.8, 11.1])[0][1]
#协方差除以方差得到拟合方程的斜率
coef = cov / var_x

print("coef based on numpy:", end='')
print(coef)

#计算X的均值
mean_x = np.mean([1, 2, 3, 4, 5])
#计算Y的均值
mean_y = np.mean([3.1, 5.2, 6.8, 8.8, 11.1])
#根据公式计算截距
intercept = mean_y - coef * mean_x

print("intercept based on numpy:", end='')
print(intercept)
```

```
coef based on numpy:1.96
intercept based on numpy:1.12
```

从上图的输出结果中可以看出，基于numpy计算得到的斜率和截距和sklearn拟合曲线给出的结果是一致的。

【实验实战】用一元线性回归预测房价

房价预测的数据集如下：

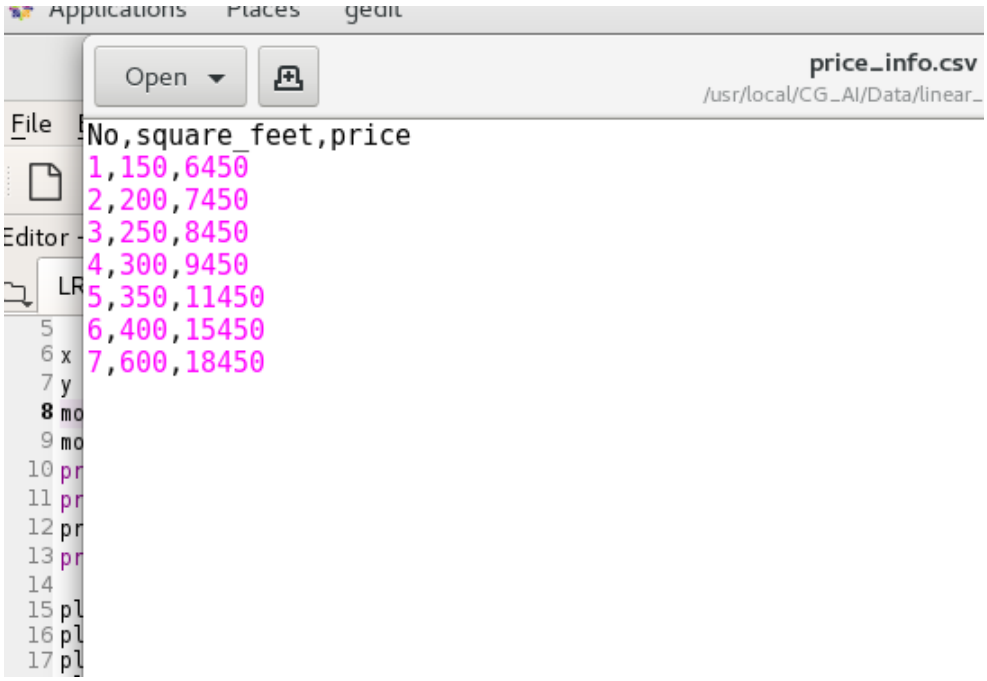
输入编号	平方英尺	价格
1	150	6450
2	200	7450
3	250	8450
4	300	9450
5	350	11450
6	400	15450
7	600	18450

数据已经被存储在price_info.csv文件中，使用以下命令查看文件内容：

In [22]:
```
!cat linear_regression / price_info.csv
```

```
No,square_feet,price
1,150,6450
2,200,7450
3,250,8450
4,300,9450
5,350,11450
6,400,15450
7,600,18450
```

文件内容如下所示：



接下来，我们开始编写线性回归的代码，并预测700英尺时的房价。

利用pandas的read_csv函数加载csv文件，然后依次读取square_feet和price这两列数据，将平方英尺和价格分别放入到X向量和Y向量中，然后将X和Y返回。

In [16]:
```
# 从csv文件中读取数据，分别为：X列表和对应的Y列表
def get_data(file_name):
    # 1. 用pandas读取csv
    data = pd.read_csv('linear_regression/price_info.csv', index_col=0, header=0)
    # 2. 构造X列表和Y列表
    X = [[x] for x in data['square_feet']]
    Y = data['price']
    return X, Y
```

线性回归模型的建立下所示，利用LinearRegression类，可以很方便的建立线性回归模型。

In [17]:
```
# 获取线性回归模型
def get_linear_model(X, Y):
    # 1.创建线性回归模型
    model = LinearRegression()
    # 2.使用X\Y对模型进行拟合
    model.fit(X, Y)
    return model
```

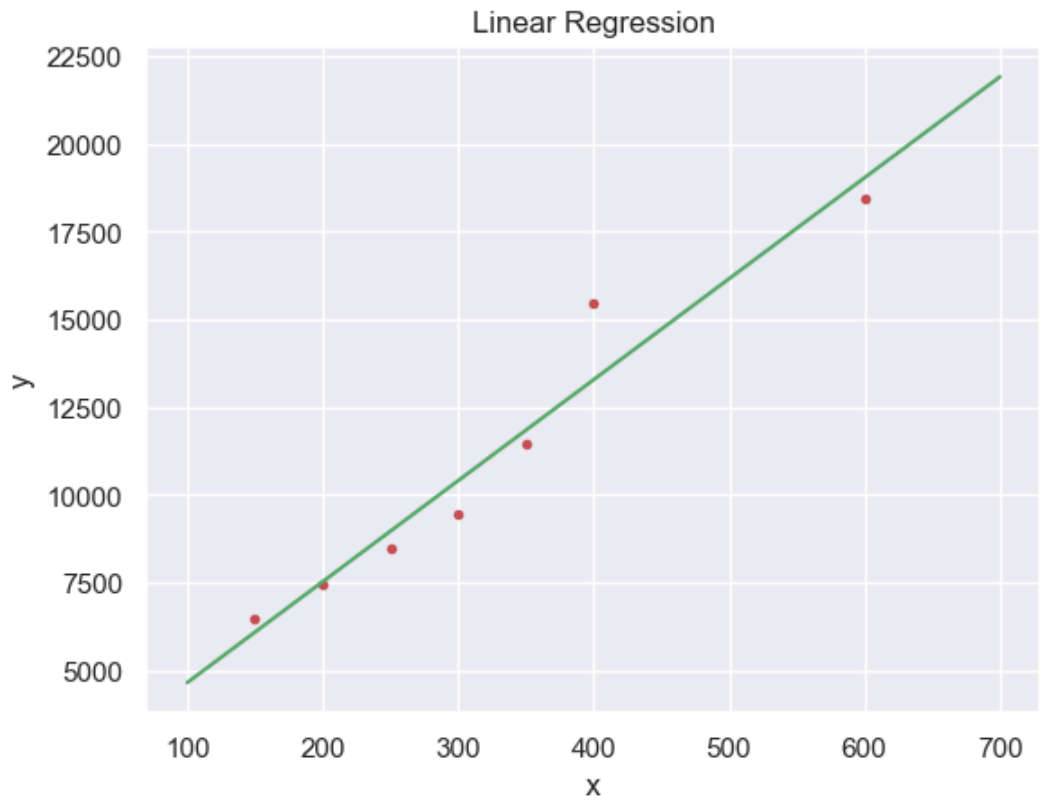
绘制预测结果的代码如下所示：

In [32]:
```
# 绘出图像
def show_linear_line(X, Y, model):
    # 1. 绘出已知数据散点图
    import seaborn
    seaborn.set()
    plt.figure()
    plt.plot(X, Y, 'r.')
    # 2. 绘出预测直线，并设定图像标题和坐标轴的标题
    plt.title('Linear Regression')
    plt.xlabel('x')
    plt.ylabel('y')
    plt.grid(True)
    x1 = [[i] for i in range(100, 800, 100)]
    plt.plot(x1, model.predict(x1), 'g-')
    plt.show()
```

主函数的控制流程如下所示：

In [33]:
```
def main():
    # 1. 读取数据
    X, Y = get_data('linear_regression/price_info.csv')
    # 2. 模型训练
    model = get_linear_model(X, Y)
    # 3. 获取700平方英尺时的预测价格
    print('700平方英尺价格', model.predict([[700]])[0])
    # 4. 绘图
    show_linear_line(X, Y, model)


if __name__ == '__main__':
    main()
```

```
700平方英尺价格 21915.42553191489
```



运行代码，可以看到相应的图片结果以及在700平方英尺时，房屋的预测价格为21915.43。

【实验实战】用一元线性回归预测车辆行驶距离

在本小节中，通过一元线性回归基于车辆的重量来预测车辆单位加仑油量可以行驶的距离，使用的数据集为“车辆MPG数据集”，英文名为“auto-mpg.data”。

该数据集共有9列，其中第1列为单位加仑油量行驶距离（英里），第5列为车辆重量（kg）。

为了方便处理，本实验已经将该数据集转换为csv文件格式，即文件“auto_mpg.csv”，这样你可以直接参考上一小节中处理csv文件的代码。

通过对数据的初步分析，发现车辆重量和车辆单位加仑油量形式距离之间存在着线性相关关系。你的任务是：

基于sklearn给出可以根据车辆重量预测其形式距离的一元线性回归方程

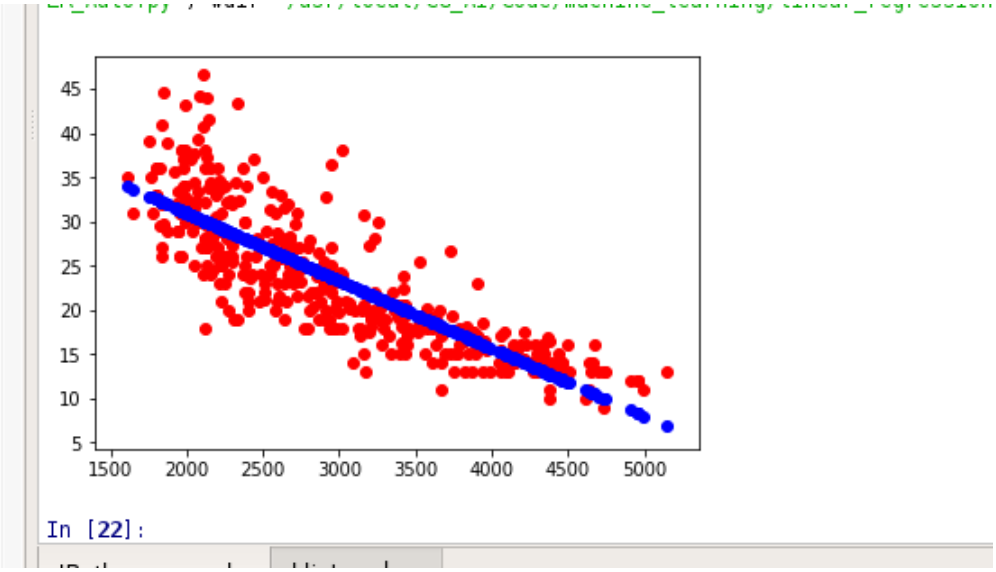
给出方程的斜率和截距

绘制出由原始数据构成的散点图以及根据回归方程得到预测结果后的散点图。

最后车辆重量为3000时的预测行驶距离。

原始数据构成的散点图指的是使用X和Y绘制图像，在得到回归方程f后，可得Y'=f(X)，根据（X,Y'）绘制的图像即根据回归方程得到预测结果后的散点图。

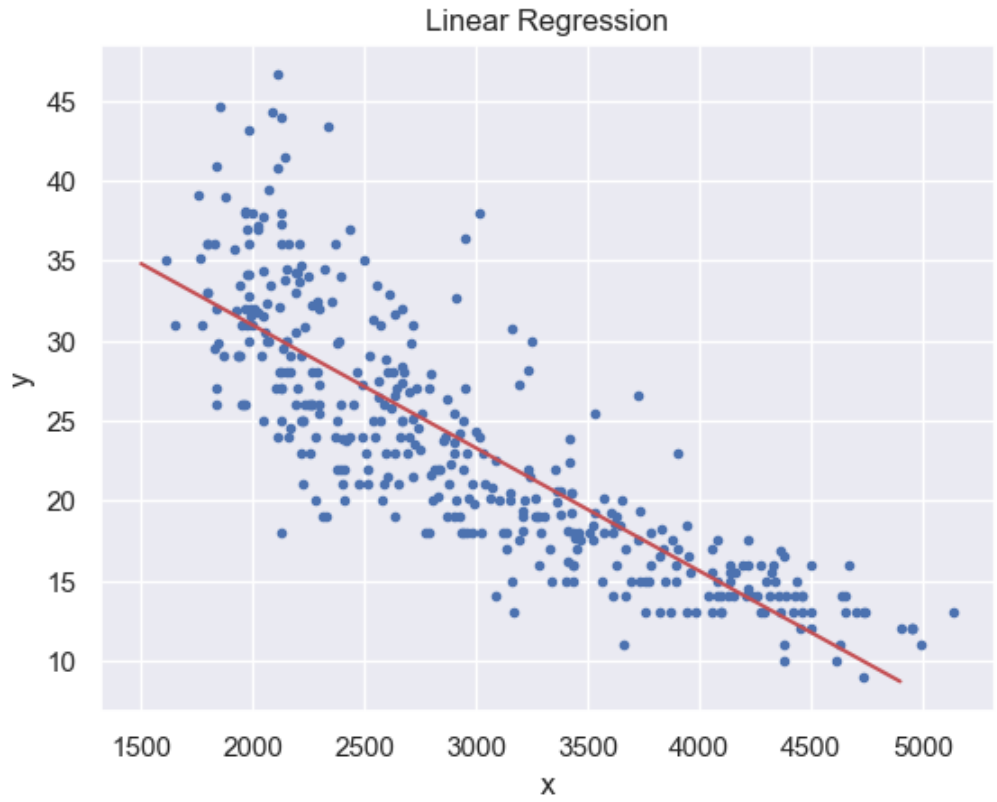
最终绘制的图像效果应该如下图所示。



In [ ]:
```
# cat linear_regression/auto_mpg.csv
```

In [35]:
```
# 从csv文件中读取数据，分别为：X列表和对应的Y列表
def get_data(file_name):
    # 1. 用pandas读取csv
    origin = pd.read_csv('linear_regression/auto_mpg.csv', header=0)
    # 2. 构造X列表和Y列表
    data = origin[['mpg', 'weight']]
    X = [[i] for i in data['weight']]
    Y = data['mpg']
    return X, Y


# 获取线性回归模型
def get_linear_model(X, Y):
    # 1.创建线性回归模型
    model = LinearRegression()
    # 2.使用X\Y对模型进行拟合
    model.fit(X, Y)
    return model


# 绘出图像
def show_linear_line(X, Y, model):
    import seaborn
    seaborn.set()
    # 1. 绘出已知数据散点图
    plt.figure()
    plt.plot(X, Y, 'b.')
    # 2. 绘出预测直线，并设定图像标题和坐标轴的标题
    plt.title('Linear Regression')
    plt.xlabel('x')
    plt.ylabel('y')
    plt.grid(True)
    x1 = [[i] for i in range(1500, 5000, 100)]
    plt.plot(x1, model.predict(x1), 'r-')
    plt.show()


def main():
    # 1. 读取数据
    X, Y = get_data('linear_regression/auto_mpg.csv')
    # 2. 模型训练
    model = get_linear_model(X, Y)
    # 3. 获取车辆重量为3000时的预测行驶距离
    print('车辆重量为3000时的预测行驶距离为', model.predict([[3000]])[0])
    # 4. 绘图
    show_linear_line(X, Y, model)


if __name__ == '__main__':
    main()
```

```
车辆重量为3000时的预测行驶距离为 23.287534228486237
```



实验总结

1. 掌握内容

通过本实验，你应该至少掌握了机器学习的以下几点：

- 了解机器学习和回归的关系
- 一元线性回归的基本思想
- 一元线性回归的数学原理
- 熟练的使用sklearn提供的一元线性回归库
- 可以将一元线性回归应用到实际问题的解决中

参考文献与延伸阅读

参考资料:

1.哈林顿，李锐. 机器学习实战 : Machine learning in action[M]. 人民邮电出版社, 2013.
2.周志华. 机器学习:Machine learning[M]. 清华大学出版社, 2016.

延伸阅读

1.李航. 统计学习方法[M]. 清华大学出版社, 2012.