Assignment-03 First Step of Machine Learning: Model and Evaluation

Crystal

Part-1 Programming Review 编程回顾

1. Re-code the Linear-Regression Model using scikit-learning(10 points)

<评阅点>：

> - 是否完成线性回归模型 (4')
> - 能够进行预测新数据(3')
> - 能够进行可视化操作(3')

In [1]:
```
from sklearn.linear_model import LinearRegression
import numpy as np

# 随机产生数据
random_data = np.random.random((40, 2))
X = random_data[:, 0]
y = random_data[:, 1]

# 线性回归
def linear_regression(X, y):
    return LinearRegression().fit(X.reshape(-1, 1), y)

reg = linear_regression(X,y)
print("coef:", reg.coef_)
print("intercept:", reg.intercept_)
```

```
coef: [-0.0464506]
intercept: 0.45032734255946394
```

In [2]:
```
arr = np.array([[1, 2, 3, 2, 1, 4], [5, 6, 1, 2, 3, 1]])
arr / 2
```

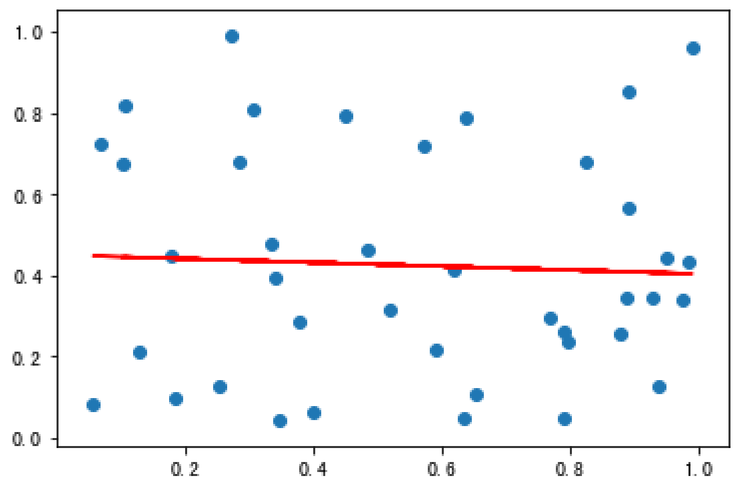
Out[2]:
```
array([[0.5, 1. , 1.5, 1. , 0.5, 2. ],
       [2.5, 3. , 0.5, 1. , 1.5, 0.5]])
```

In [3]:
```
# 数据预测
reg.predict([[0.9]])[0]
```

Out[3]: 0.40852180631115936

In [4]:
```
import matplotlib.pyplot as plt

# 数据可视化
def f(x):
    return reg.coef_ * x + reg.intercept_

plt.scatter(X, y)
plt.plot(X, f(X), color='red')
```

Out[4]: [<matplotlib.lines.Line2D at 0x10eaf52e8>]



2. Complete the unfinished KNN Model using pure python to solve the previous Line-Regression problem. (8 points)

<评阅点>:

> - 是否完成了KNN模型 (4')
> - 是否能够预测新的数据 (4')

In [5]:
```
from scipy.spatial.distance import cosine
def distance(x1, x2):
    return cosine(x1, x2)

def model(X, y):
    return [(Xi, yi) for Xi, yi in zip(X, y)]

def predict(x, k=5):
    most_similars = sorted(model(X, y), key=lambda xi: distance(xi[0], x
))[:k]
    y_hats = [_y for x, _y in most_similars]
    return np.mean(y_hats)

# 数据预测
myself_knn = model(X, y)
predict(0.9)
```

Out[5]:
```
0.2879433987419132
```

3. Re-code the Decision Tree, which could sort the features by salience. (12 points)

<评阅点>

> - 是否实现了信息熵 (1')
> - 是否实现了最优先特征点的选择(5')
> - 是否实现了持续的特征选则(6')

In [6]:
```python
import pandas as pd

mock_data = {
    'gender':['F', 'F', 'F', 'F', 'M', 'M', 'M'],
    'income': ['+10', '-10', '+10', '+10', '+10', '+10', '-10'],
    'family_number': [1, 1, 2, 1, 1, 1, 2],
    'bought': [1, 1, 1, 0, 0, 0, 1],
}
dataset = pd.DataFrame.from_dict(mock_data)
dataset
```

Out[6]:

	gender	income	family_number	bought
0	F	+10	1	1
1	F	-10	1	1
2	F	+10	2	1
3	F	+10	1	0
4	M	+10	1	0
5	M	+10	1	0
6	M	-10	2	1

In [7]:
```
from collections import Counter

def entropy(elements):
    '''群体的混乱程度'''
    counter = Counter(elements)
    probs = [counter[c] / len(elements) for c in set(elements)]
    return - sum(p * np.log(p) for p in probs)

def find_the_optimal_spilter(training_data: pd.DataFrame, target: str) -
> str:
    x_fields = set(training_data.columns.tolist()) - {target}
    spliter = None
    min_entropy = float('inf')

    for f in x_fields:
        values = set(training_data[f])
        for v in values:
            sub_spliter_1 = training_data[training_data[f] == v][target]
.tolist()
            # split by the current feature and one value
            entropy_1 = entropy(sub_spliter_1)
            sub_spliter_2 = training_data[training_data[f] != v][target]
.tolist()
            entropy_2 = entropy(sub_spliter_2)
            entropy_v = entropy_1 + entropy_2
            if entropy_v <= min_entropy:
                min_entropy = entropy_v
                spliter = (f, v)

    print('spliter is: {}'.format(spliter))
    print('the min entropy is: {}'.format(min_entropy))

    return spliter
```

In [8]:
```
find_the_optimal_spilter(training_data=dataset, target='bought')
```

```
spliter is: ('family_number', 2)
the min entropy is: 0.6730116670092565
```

Out[8]:
```
('family_number', 2)
```

In [9]:
```
dataset[dataset['family_number'] == 2]
```

Out[9]:

	gender	income	family_number	bought
2	F	+10	2	1
6	M	-10	2	1

In [10]:
```
sub_data = dataset[dataset['family_number'] != 2]
sub_data
```

Out[10]:

	gender	income	family_number	bought
0	F	+10	1	1
1	F	-10	1	1
3	F	+10	1	0
4	M	+10	1	0
5	M	+10	1	0

In [11]:
```
find_the_optimal_spilter(training_data=sub_data, target='bought')
```

```
spliter is: ('income', '+10')
the min entropy is: 0.5623351446188083
```

Out[11]:
```
('income', '+10')
```

In [12]:
```
sub_data[sub_data['income']=='+10']
```

Out[12]:

	gender	income	family_number	bought
0	F	+10	1	1
3	F	+10	1	0
4	M	+10	1	0
5	M	+10	1	0

In [13]:
```
sub_data[sub_data['income']!='+10']
```

Out[13]:

	gender	income	family_number	bought
1	F	-10	1	1

In [14]:
```
find_the_optimal_spilter(training_data=sub_data[sub_data['income']=='+1
0'], target='bought')
```

```
spliter is: ('family_number', 1)
the min entropy is: 0.5623351446188083
```

Out[14]:
```
('family_number', 1)
```

所以，如果家庭成员是2人，那么就会购买，如果不是2人，则观察收入情况，如果收入是'+10'， 那么他有 1/4 的概率会购买，如果是 '-10', 那么不买

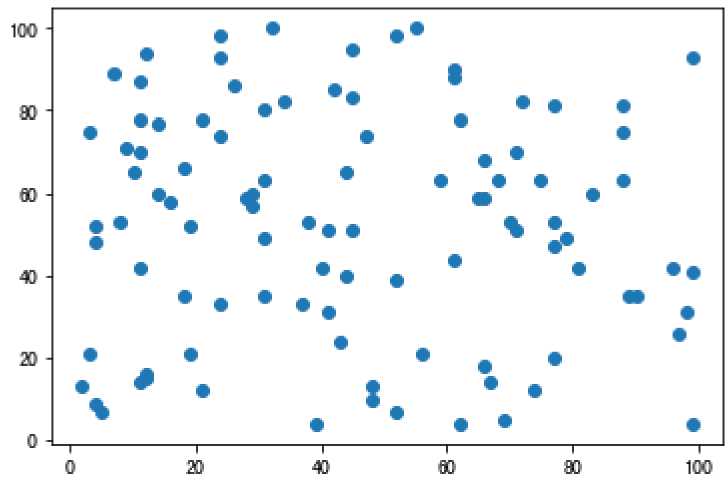
4. Finish the K-Means using 2-D matplotlib (8 points)

<评阅点>

> - 是否完成了KMeans模型，基于scikit-learning (3')
> - 是否完成了可视化任务（5'）

In [15]:
```
import random

X1 = [random.randint(0, 100) for _ in range(100)]
X2 = [random.randint(0, 100) for _ in range(100)]
plt.scatter(X1, X2)
```

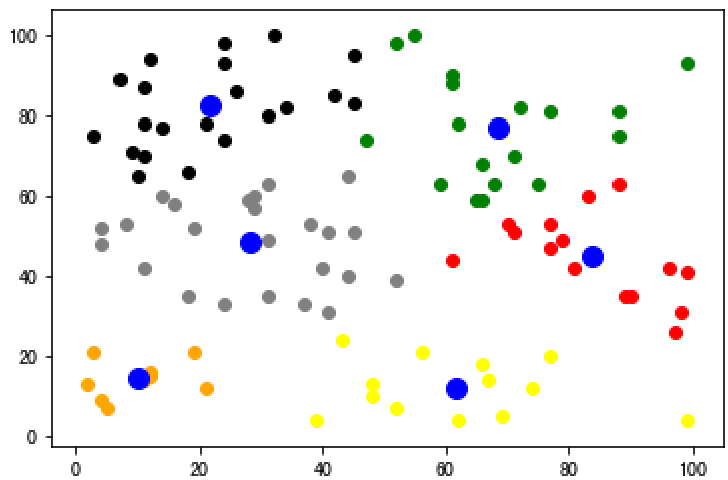
Out[15]: `<matplotlib.collections.PathCollection at 0x1242d8d30>`



In [16]:
```
from sklearn.cluster import KMeans

tranning_data = [[x1, x2] for x1, x2 in zip(X1, X2)]
cluster = KMeans(n_clusters=6, max_iter=500)
cluster.fit(tranning_data)
cluster.cluster_centers_
```

Out[16]:
```
array([[68.44444444, 76.94444444],
       [28.29166667, 48.375     ],
       [61.53846154, 12.        ],
       [ 9.88888889, 14.22222222],
       [21.61904762, 82.19047619],
       [83.73333333, 44.8       ]])
```

In [17]:
```
from collections import defaultdict

color = ['red', 'green', 'grey', 'black', 'yellow', 'orange']
centers = defaultdict(list)

for label, location in zip(cluster.labels_, tranning_data):
    centers[label].append(location)

for i, c in enumerate(centers):
    for location in centers[c]:
        plt.scatter(*location, c=color[i])

for center in cluster.cluster_centers_:
    plt.scatter(*center, s=100, c="blue")
```



Part-2 Question and Answer 问答

1. What's the *model*? why all the models are wrong, but some are useful? (5 points)

Ans: 将实际问题量化，用数据对问题进行建模。没有百分百准确的模型，理论和实际总是会有偏差，但可以一定程度地帮助建模的人理解或者解决一些问题。

<评阅点>

> - 对模型的理解是否正确,对模型的抽象性是否正确(5')

2. What's the underfitting and overfitting? List the reasons that could make model overfitting or underfitting. (10 points)

Ans: underfitting是欠拟合，模型无法捕获数据中的重要区别和模式，结果预测可能相差甚远，原因：模型复杂度过低、特征量过少。overfitting是过拟合，在训练集上表现良好，在测试集上表现糟糕，原因：1.训练集和测试集特征分布不一致、2.数据噪声太大、3.数据量太小、4.特征量太多、5.模型太过复杂

<评阅点>

> - 对过拟合和欠拟合的理解是否正确 (3')
> - 对欠拟合产生的原因是否理解正确(2')
> - 对过拟合产生的原因是否理解正确(5')

3. What's the precision, recall, AUC, F1, F2score. What are they mainly target on? (12')

Ans:

- precision: 所有预测为正而且预测正确 / 所有预测为正的个数， 重点是找的对。
- recall: 所有预测为正而且预测正确 / 所有真正为正的个数， 重点是找的全。
- AUC: 全称Area Under Curve，被定义为ROC曲线下的面积，取值范围在0.5到1之间, 数值越大，对应的分类器越好。

F_score = $(1+\beta^2)\frac{precision*recall}{\beta^2*precision+recall}$

- F1score: $\beta=1$, 综合考虑Precision和Recall的调和值
- F2score: $\beta>1$, 认为召回率更加重要

<评阅点>

> - 对precision, recall, AUC, F1, F2 理解是否正确(6‘)
> - 对precision, recall, AUC, F1, F2的使用侧重点是否理解正确 (6’)

4. Based on our course and yourself mind, what's the machine learning? (8')

Ans: 帮助人们更智能的完成一些任务，例如运用数据建模，分析，预测。

<评阅点> 开放式问题，是否能说出来机器学习这种思维方式和传统的分析式编程的区别（8'）

5. "正确定义了机器学习模型的评价标准(evaluation)， 问题基本上就已经解决一半". 这句话是否正确？你是怎么看待的？ (8‘)

Ans: 不同的问题应该用不同的模型的去解决，如果模型的准确性不靠谱，那么得出来的结果也不可靠。

<评阅点> 开放式问题，主要看能理解评价指标对机器学习模型的重要性.

Part-03 Programming Practice 编程练习

1. In our course and previous practice, we complete some importance components of Decision Tree. In this problem, you need to build a **completed** Decision Tree Model. You show finish a `predicate()` function, which accepts three parameters **<gender, income, family_number>**, and outputs the predicated 'bought': 1 or 0. (20 points)

In [18]:
```
# 如果家庭成员是2人，那么就会购买，如果不是2人，则观察收入情况，如果收入是'+10'，  那么他有 1/4 的概率会购买，如果是 '-10'，那么会买
def predicate(gender, income, family_number):
    bought = None
    if family_number == 2:
        bought = 1
    else:
        if income == -10:
            bought = 1
        else:
            bought = np.random.choice([1, 0], p = [0.25, 0.75])
    return bought
```

In [19]:
```
# 家庭成员=1,  收入为“-10”, 性别男
predicate("M", -10, 1)
```

Out[19]: 1

In [20]:
```
# 家庭成员=2,  收入为“+10”, 性别男
predicate("M", +10, 2)
```

Out[20]: 1

In [21]:
```
# 家庭成员=1,  收入为“+10”, 性别女
predicate("F", +10, 1)
```

Out[21]: 0

<评阅点>

> - 是否将之前的决策树模型的部分进行合并组装， predicate函数能够顺利运行(8')
> - 是够能够输入未曾见过的X变量，例如gender, income, family_number 分别是： <M, -10, 1>, 模型能够预测出结果 (12')

1. 将上一节课(第二节课)的线性回归问题中的Loss函数改成"绝对值"，并且改变其偏导的求值方式，观察其结果的变化。(19 point)

Assume that the target funciton is a linear function

$$y = kx + b$$

$$loss = \frac{1}{n} \sum |(y_i - \hat{y_i})|$$

$$loss = \frac{1}{n} \sum |y_i - (kx_i + b_i)|$$

$$\frac{\partial loss}{\partial k} = \frac{1}{n} \sum (|x_i|)$$

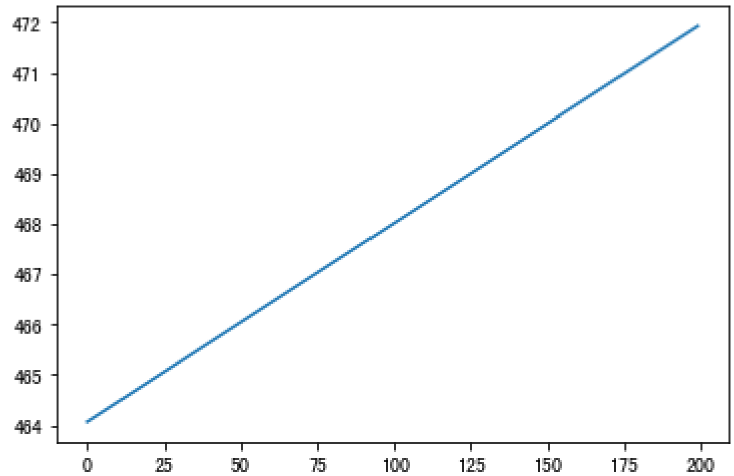
$$\frac{\partial loss}{\partial b} = \frac{1}{n}$$

In [22]:
```
from sklearn.datasets import load_boston

#define target function
def price(rm, k, b):
    return k * rm + b

# define loss function
def loss(y,y_hat):
    return sum(abs(y_i - y_hat_i) for y_i, y_hat_i in zip(list(y),list(y
_hat)))/len(list(y))

# define partial derivative
def partial_derivative_k(x, y, y_hat):
    n = len(y)
    gradient = 0
    for x_i, y_i, y_hat_i in zip(list(x),list(y),list(y_hat)):
        gradient += abs(x_i)
    return 1/n * gradient

def partial_derivative_b(y, y_hat):
    n = len(y)
    return 1/n

# load data
dataset = load_boston()
x,y=dataset['data'],dataset['target']
X_rm = x[:,5]

# initialized parameters
k = random.random() * 200 - 100  # -100 100
b = random.random() * 200 - 100  # -100 100
learning_rate = 1e-3
iteration_num = 200
losses = []

for i in range(iteration_num):
    price_use_current_parameters = [price(r, k, b) for r in X_rm]  # \ha
t{y}
    current_loss = loss(y, price_use_current_parameters)
    losses.append(current_loss)

    k_gradient = partial_derivative_k(X_rm, y, price_use_current_paramet
ers)
    b_gradient = partial_derivative_b(y, price_use_current_parameters)

    k = k + (-1 * k_gradient) * learning_rate
    b = b + (-1 * b_gradient) * learning_rate

best_k = k
best_b = b
best_k, best_b
```

Out[22]:
```
(-61.63271975733362, -62.093162614284815)
```

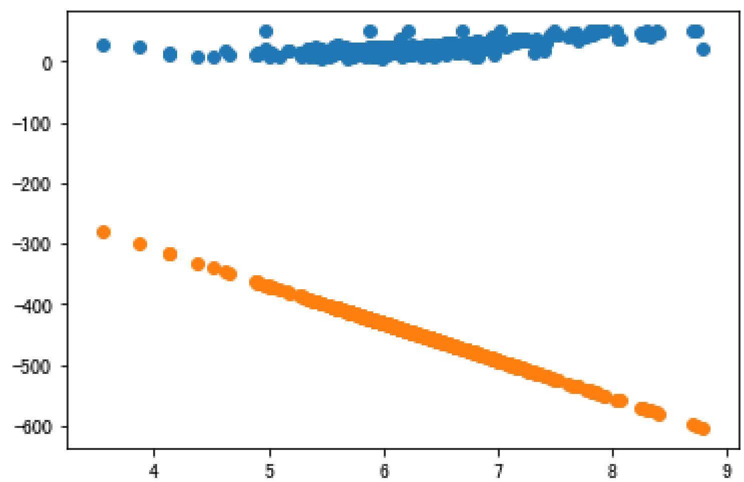
In [23]:
```
plt.plot(list(range(iteration_num)),losses)
```

Out[23]: [<matplotlib.lines.Line2D at 0x124dc0f60>]



In [24]:
```
price_use_best_parameters = [price(r, best_k, best_b) for r in X_rm]

plt.scatter(X_rm,y)
plt.scatter(X_rm,price_use_current_parameters)
```

Out[24]: <matplotlib.collections.PathCollection at 0x124286cf8>



结果比较随机。

<评阅点>

- 是否将Loss改成了“绝对值”(3')
- 是否完成了偏导的重新定义(5')
- 新的模型Loss是否能够收敛 (11’)