机器学习_day1

定义

机器学习是从数据中心自动分析获得规律（模型），并利用规律对未知数据进行预测。

特征工程

- 是什么：使用专业的知识和技巧处理数据，使得特征能在机器学习算法上发挥更好的作用
- 为什么：筛选、处理选择合适的特征
- 怎么用
  - 数据集
    - scikit-learn
    - Kaggle
    - UCI

Scikit-learn的使用

- 特征抽取
  - 定义：包含将任意数据（如文本或图像，类别特征）转换为可用于机器学习的数字特征
  - 字典特征提取
  - 文本特征提取
  - 汉字：jieba分词处理
  - TF-idf文本特征提取
    - 为了处理这种同一个词在很多片文章当中出现的次数都比较多的情况
    - 用以评估一个词对于一个文件集或一个语料库中的其中一份文件的重要程度
- 特征预处理
  - 通过一些转换函数将特征数据 转换成更加适合算法模型的特征数据过程
  - 基本的数据处理，比如缺失值处理（pandas）
  - 无量纲化
    - 归一化（很少使用）
    - 标准化
- 特征降维
  - 在某些限定条件下，降低随机变量的个数
  - 降维方式
    - 特征选择
      - 定义：数据中包含冗余或无关变量（或称特征、属性、指标等），旨在从原有特征中找出主要特征
      - 方式
        - 过滤式（Filter）
          - 低方差特征过滤
          - 相关系数
        - 嵌入式（Embedded）
        - 包裹式（Wrapper）
    - 主成分分析（PCA）
      - 定义：高维数据转化为低维数据的过程，在此过程中可能会舍弃原有数据、创造新的变量
      - 作用：是数据维数压缩，尽可能降低原数据的维数（复杂度），损失少量信息
      - 应用：回归分析或者聚类分析当中

机器学习算法介绍

- 按学习方式分类
  - 监督学习
    - 分类
    - 回归
    - 标注
  - 无监督学习：聚类
  - 半监督和强化学习
- 两种数据类型
  - 离散型数据
  - 连续型数据

jieba分词处理

由于汉字不使用空格符分隔

pip install jieba

jieba.cut()

返回词语组成的生成器

```
8 def cutword():
9     # 将文本用jieba处理
10    content1 = jieba.cut("如梦令其一宋 李清照常记溪亭日暮，沉醉不知归路。兴尽晚回舟，误入藕花深处。争渡，争渡，惊起一滩鸥鹭。")
11    content2 = jieba.cut("如梦令其二宋 李清照昨夜雨疏风骤，浓睡不消残酒。试问卷帘人，却道海棠依旧。知否？知否？应是绿肥红瘦")
12
13    # 把分词后的对象转换成列表，再变成以空格隔开的字符串
14    c1 = ' '.join(list(content1))
15    c2 = ' '.join(list(content2))
16    return c1, c2
```

Sikit-learn的使用

字典特征提取

1.导入模块：`from sklearn.feature_extration.text import DictVectorizer`

2.实例化类DicVectorizer

```
16 def dictvec():
17     """对字典数据进行特征抽取"""
18     # 实例化dictvec
19     dic = DictVectorizer(sparse=False)
```

- sparse表示是否返回sparse矩阵

```
20     # dictvec调用fit_transform
21     # 三个样本的特征数据（字典形式）
22     data = dic.fit_transform([{'city': '北京', 'temperature': 100}, {'city': '上海', 'temperature': 60}, {'city': '深圳', 'temperature': 30}])
```

3.调用 fit_transform 方法输入数据并转换

- 把字典数据转换成one-hot编码

文本特征提取

1.导入模块：`from sklearn.feature_extration.text import CountVectorizer`

2.实例化类CountVectorizer

```
6 def countvec():
7     # 实例化count
8     count = CountVectorizer()
```

```
9     # 对两篇文章特征抽取
10    data = count.fit_transform(["life is short, i like python", "life is too long, i dislike python"])
```

3.调用 fit_transform 方法输入数据并转换

TF-IDF文本特征提取

公式：$tf \times idf = tfidf$

特征预处理 无量纲化

sklearn.preprocessing.MinMaxScaler(feature_range=(a, b))

- (a, b) 是指归一化之后的特征值范围

MinMaxScaler.fit_transform(X)

归一化（并不常用）

```
38 def minmaxscaler():
39     """对约会对象数据进行归一化处理"""
40     # 读取数据选择要处理的特征
41     dating = pd.read_csv("./dating.txt")
42     data = dating[['milage', 'Liters', 'Consumtime']]
43     # 实例化，进行fit_transform
44     mm = MinMaxScaler(feature_range=(0, 9))
45     data = mm.fit_transform(data)
46     print(data)
47
48     return None

54 if __name__ == '__main__':
55     minmaxscaler()
```

sklearn.preprocessing.StandardScaler()

处理之后每列来说所有的数据都聚集在均值0附近标准差为1的范围

StandardScaler.fit_transform(X)

标准化

```
form sklearn.preprocessing import StandardScaler
38 def standardscaler():
39     """对约会对象数据进行标准化处理"""
40     # 读取数据选择要处理的特征
41     dating = pd.read_csv("./dating.txt")
42     data = dating[['milage', 'Liters', 'Consumtime']]
43     # 实例化，进行fit_transform
44     std = StandardScaler()
45     data = std.fit_transform(data)
46     print(data)
48     return None
49
51 if __name__ == '__main__':
52     standardscaler()
```

降维

特征选择

Filter（过滤式）

1.导入模块

`from sklearn.feature_selection import VarianceThreshold`

```
8 from sklearn.feature_selection import VarianceThreshold
51 def varthreshold():
52     """使用方差法进行股票的指标的过滤"""
53     factor = pd.read_csv("./stock_day.csv")
54     # 使用VarianceThreshold
55     var = VarianceThreshold(threshold=0.0)
56     # 9列指标低方差过滤
57     # data = var.fit_transform(factor.iloc[:, 1:10])
58     data = var.fit_transform(factor)
59
60     print(data)
61     print(data.shape)
62     return None
63
65 if __name__ == '__main__':
66     varthreshold()
```

训练集差异低于threshold的特征将被删除。默认值是保留所有非零方差特征，即删除所有样本中具有相同值的特征。

皮尔森相关系数

反映变量之间相关关系密切程度的统计指标

API：`from scipy.stats import pearsonr`

```
9 from scipy.stats import pearsonr

65 def pearson():
66     """对股票的一些常见财务指标进行相关性计算"""
67     factor = ['open', 'high', 'low', 'volume', 'price_change', 'p_change', 'ma5', 'ma10', 'ma20', 'v_ma5', 'v_ma10', 'v_ma20', 'turnover']
68     data = pd.read_csv('./stock_day.csv')
69     # 循环获取两个指标
70     for i in range(len(factor)):
71         for j in range(i, len(factor)-1):
72             # 第一次open,j+1
73             print("指标1:%s 和指标2:%s 的相关系数计算：%f" % (factor[i], factor[j+1], pearsonr(data[factor[i]], data[factor[j+1]])[0])
74
75     return None
76
79 if __name__ == '__main__':
80     pearson()
```

主成分分析（PCA）

定义：高维数据转化为低维数据的过程，在此过程中可能会舍弃原有数据、创造新的变量

API：sklearn.decmposition.PCA(n_components=None)

n_components: 小数_表示保留百分之多少的信息（90%以上）

整数：减少到多少特征

PCA.fit_transform(X)

```
80 def pc_a():
81     """主成分分析进行降维"""
82     data = pd.read_csv('./stock_day.csv')
83     pca = PCA(n_components=0.95)
84     data = pca.fit_transform(data[['high', 'open']])
85     print(data)
86
87     return None
88
89 if __name__ == '__main__':
90     pc_a()
```