# 综合实训 机器学习之特征提炼 总结报告

| 年级 | **2018**级 | 专业 | 软件工程 |
| --- | --- | --- | --- |
| 学号 | 18342115 | 姓名 | 杨玲 |

## 一、实训内容

- 任务一：计算每一种特征与标签的相关程度，并按照相关程度对特征进行排序，输出特征顺序。
- 任务二：实现一个机器学习算法（分类算法），选择合适的特征，训练模型，记录模型预测准确率。
- 任务三：探究特征个数与预测准确率的关系
- 任务四：输出一份实训报告

## 二、实训过程与结果

### **1.**任务一

#### （**1**）实验总体思路与过程

- 因为嵌入法和包装法都是特征选择和算法训练同时进行，在任务一的条件下，就暂时不考虑，所以主要考虑过滤法。
- 又由于F检验过滤只能捕捉线性相关性，且要求数据服从正态分布，不太符合条件，所以主要考虑了卡方过滤和互信息过滤。
- 实验过程主要是调用sklearn库的feature selection来进行卡方过滤、互信息分类、标准化互信息分类和用最大互信息数(MIC)进行分类，按照相关程度对特征进行排序，输出特征顺序。
- 具体过程详见任务一附件代码

#### （**2**）实验结果

以表2-AS为例，各不同的方法得出特征相关程度排序如下，相关程度越高排名越靠前，其他表格的排序结果具体参见“任务一实验报告”

| Number | Feature Name | chi2 | mutual_info | normalized_mutual_info | Maximal Information Coefficient(MIC) |
|---|---|---|---|---|---|
| 1 | max_degree | 16 | 16 | 16 | 16 |
| 2 | fail_node_degree | 9 | 13 | 13 | 14 |
| 3 | fail_neber_degree | 8 | 5 | 2 | 8 |
| 4 | fail_degree_sum | 7 | 10 | 11 | 13 |
| 5 | max_load | 5 | 9 | 10 | 11 |
| 6 | big_load_num | 15 | 14 | 14 | 15 |
| 7 | fail_load_sum | 2 | 3 | 7 | 3 |
| 8 | fail_num | 10 | 7 | 5 | 6 |
| 9 | first_round_fail | 11 | 8 | 4 | 7 |
| 10 | neber_fail_num | 12 | 12 | 6 | 9 |
| 11 | fail_round | 14 | 15 | 15 | 10 |
| 12 | subgraph_num | 13 | 6 | 3 | 2 |
| 13 | fail_node_load | 3 | 4 | 8 | 5 |
| 14 | load_change | 1 | 1 | 9 | 1 |
| 15 | degree_change | 6 | 11 | 12 | 12 |
| 16 | fail_neber_load | 4 | 2 | 1 | 4 |

## **2.**任务二

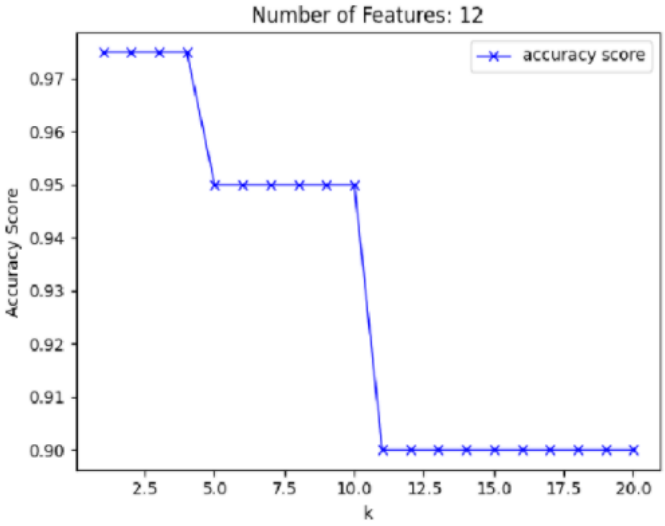
### （**1**）实验总体思路与过程

- 选择相关系数最高的12个特征值
- 按照训练、测试比9:1的比例将数据集拆分为训练集和测试集，每10条数据中选取第二条数据作为测试集
- 调用sklearn库的KNeighborsClassifier来实现KNN算法，并通过改变k的大小来进行训练及测试
- 输出相关结果
- 具体过程详见任务二附件代码

### （**2**）实验结果

以表1-BA_cla为例，当k从1到20时，测试的准确率逐渐下降，其他表格的结果具体参见“任务二实验报告”

| k | Accuracy Score |
|---|---|
| 1 | 0.975 |
| 2 | 0.975 |
| 3 | 0.975 |
| 4 | 0.975 |
| 5 | 0.95 |
| 6 | 0.95 |
| 7 | 0.95 |
| 8 | 0.95 |
| 9 | 0.95 |
| 10 | 0.95 |
| 11 | 0.9 |
| 12 | 0.9 |
| 13 | 0.9 |
| 14 | 0.9 |
| 15 | 0.9 |
| 16 | 0.9 |
| 17 | 0.9 |
| 18 | 0.9 |
| 19 | 0.9 |
| 20 | 0.9 |




## 3.任务三

### （1）实验总体思路与过程

思路一：按照相关程度排序（该思路在完成任务二时已实现）

- 特征选择，分别选择相关程度最高的16、12、8、4个特征来进行后续训练和测试
- 按照训练、测试比9:1的比例将数据集拆分为训练集和测试集，每10条数据中选取第二条数据作为测试集
- 调用sklearn库的KNeighborsClassifier来实现KNN算法，并进行训练及测试
- 输出相关结果
- 具体过程详见任务二附件代码

思路二：按照表格顺序排序

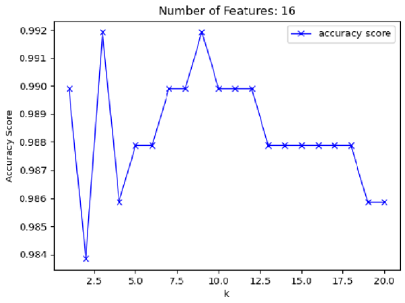
- 按照数据表中的前后顺序，分别选择1-16个特征数量来进行训练和测试
- 按照训练、测试比9:1的比例将数据集拆分为训练集和测试集，每10条数据中选取第二条数据作为测试集
- 调用sklearn库的KNeighborsClassifier来实现KNN算法，并进行训练及测试
- 输出相关结果
- 具体过程详见任务三附件代码

（2）实验结果

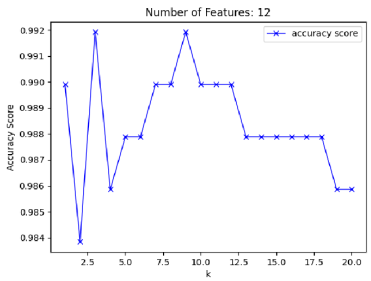
思路一结果

- 以表2-AS_cla为例，总体来说，随着选择特征数量的增多，模型的准确率也随之提高。其他表格的结果具体参见“任务二实验报告”

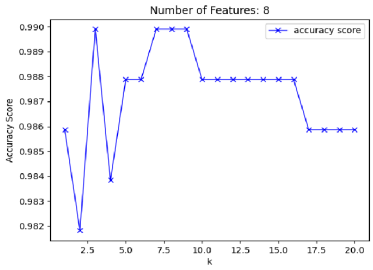
| k | Accuracy Score |
|---|---|
| 1 | 0.98989898989899 |
| 2 | 0.9838383838383838 |
| 3 | 0.9919191919191919 |
| 4 | 0.9858585858585859 |
| 5 | 0.9878787878787879 |
| 6 | 0.9878787878787879 |
| 7 | 0.98989898989899 |
| 8 | 0.98989898989899 |
| 9 | 0.9919191919191919 |
| 10 | 0.98989898989899 |
| 11 | 0.98989898989899 |
| 12 | 0.98989898989899 |
| 13 | 0.9878787878787879 |
| 14 | 0.9878787878787879 |
| 15 | 0.9878787878787879 |
| 16 | 0.9878787878787879 |
| 17 | 0.9878787878787879 |
| 18 | 0.9878787878787879 |
| 19 | 0.9858585858585859 |
| 20 | 0.9858585858585859 |



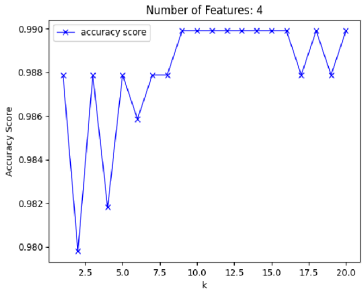

| k | Accuracy Score |
|---|---|
| 1 | 0.98989898989899 |
| 2 | 0.9838383838383838 |
| 3 | 0.9919191919191919 |
| 4 | 0.9858585858585859 |
| 5 | 0.9878787878787879 |
| 6 | 0.9878787878787879 |
| 7 | 0.98989898989899 |
| 8 | 0.98989898989899 |
| 9 | 0.9919191919191919 |
| 10 | 0.98989898989899 |
| 11 | 0.98989898989899 |
| 12 | 0.98989898989899 |
| 13 | 0.9878787878787879 |
| 14 | 0.9878787878787879 |
| 15 | 0.9878787878787879 |
| 16 | 0.9878787878787879 |
| 17 | 0.9878787878787879 |
| 18 | 0.9878787878787879 |
| 19 | 0.9858585858585859 |
| 20 | 0.9858585858585859 |




| k | Accuracy Score |
|---|---|
| 1 | 0.9858585858585859 |
| 2 | 0.9818181818181818 |
| 3 | 0.98989898989899 |
| 4 | 0.9838383838383838 |
| 5 | 0.9878787878787879 |
| 6 | 0.9878787878787879 |
| 7 | 0.98989898989899 |
| 8 | 0.98989898989899 |
| 9 | 0.98989898989899 |
| 10 | 0.9878787878787879 |
| 11 | 0.9878787878787879 |
| 12 | 0.9878787878787879 |
| 13 | 0.9878787878787879 |
| 14 | 0.9878787878787879 |
| 15 | 0.9878787878787879 |
| 16 | 0.9878787878787879 |
| 17 | 0.9858585858585859 |
| 18 | 0.9858585858585859 |
| 19 | 0.9858585858585859 |
| 20 | 0.9858585858585859 |




| k | Accuracy Score |
|---|---|
| 1 | 0.9878787878787879 |
| 2 | 0.9797979797979798 |
| 3 | 0.9878787878787879 |
| 4 | 0.9818181818181818 |
| 5 | 0.9878787878787879 |
| 6 | 0.9858585858585859 |
| 7 | 0.9878787878787879 |
| 8 | 0.9878787878787879 |
| 9 | 0.98989898989899 |
| 10 | 0.98989898989899 |
| 11 | 0.98989898989899 |
| 12 | 0.98989898989899 |
| 13 | 0.98989898989899 |
| 14 | 0.98989898989899 |
| 15 | 0.98989898989899 |
| 16 | 0.98989898989899 |
| 17 | 0.9878787878787879 |
| 18 | 0.98989898989899 |
| 19 | 0.9878787878787879 |
| 20 | 0.98989898989899 |

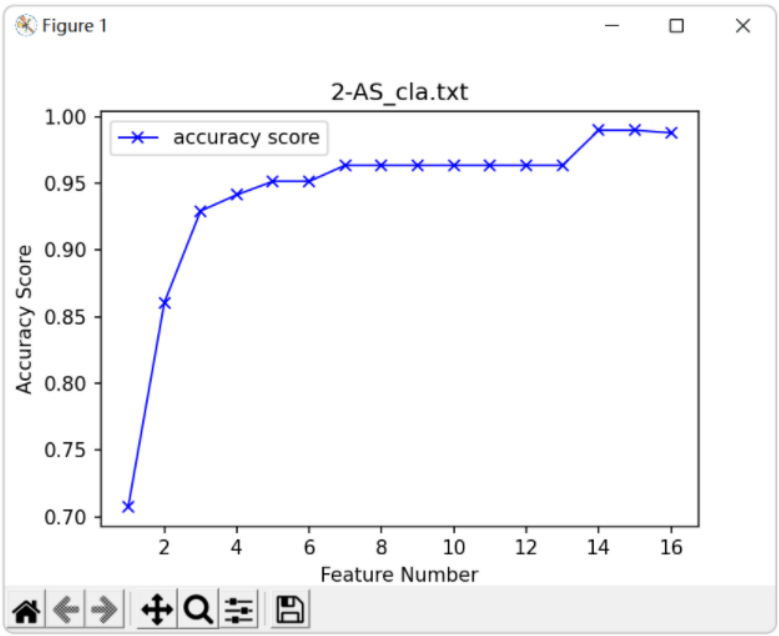



思路二结果

- 以表2-AS_cla为例，随着选择特征数量的增多，模型的准确率也随之提高。其他表格的结果具体参见“任务三实验报告”

最终结果：

| Feature Number | Accuracy Score |
|---|---|
| 1 | 0.7070707070707071 |
| 2 | 0.8606060606060606 |
| 3 | 0.9292929292929293 |
| 4 | 0.9414141414141414 |
| 5 | 0.9515151515151515 |
| 6 | 0.9515151515151515 |
| 7 | 0.9636363636363636 |
| 8 | 0.9636363636363636 |
| 9 | 0.9636363636363636 |
| 10 | 0.9636363636363636 |
| 11 | 0.9636363636363636 |
| 12 | 0.9636363636363636 |
| 13 | 0.9636363636363636 |
| 14 | 0.98989898989899 |
| 15 | 0.98989898989899 |
| 16 | 0.9878787878787879 |



## 三、实训总结与分析

- 从任务一实验结果可以看出，不同的过滤方法得出的结果是不同的，毕竟每种方法的原理都不同，且都涉及到不同调整方法的超参数。
    - 一般来说，过滤法更快速，但更粗糙；包装法和嵌入法更精确，比较适合具体到算法去调整，但计算量比较大，运行时间长。
    - 当数据量很大的时候，优先使用方差过滤和互信息法调整，再上其他特征选择方法。
    - 使用逻辑回归时，优先使用嵌入法。
    - 使用支持向量机时，优先使用包装法。
    - 不知从何开始时，一般从过滤法开始，再看具体数据具体分析。
- 从任务二实验结果可以看出，对于KNN模型，选择的k不同，模型的准确率也会不同，对于大部分表来说，k=3时，模型的准确率是最高的。
- 从任务三实验结果可以看出，选择的特征数量不同，模型的准确率也会不同。
    - 总体来说，随着选择的特征数量变多，模型的准确率也会上升。但随着选择特征数量的增多，提高幅度会越小。个别相关系数不高的特征向量可能反而会导致准确率下降。
    - 一般来说，当选择的特征数量大于等于3时，模型的准确率都会高于90%。

## 四、参考代码

**1.**任务一

- 卡方过滤

```
import numpy as np
import pandas as pd
from sklearn.feature_selection import chi2
from sklearn import metrics
from minepy import MINE
from prettytable import PrettyTable

#特征名称
feature_head = ['max_degree','fail_node_degree','fail_neber_degree','fail_degree_sum',
                    'max_load','big_load_num','fail_load_sum','fail_num',
                    'first_round_fail','neber_fail_num','fail_round','subgraph_num',
                    'fail_node_load','load_change','degree_change','fail_neber_load']
number = [1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16]

#按照得到结果进行排序
```

```
def sort_func(data, result):
    for i in range(data.shape[0]):
        count = 0
        for j in range(data.shape[0]):
            if(data[j] > data[i]):
                count = count + 1
        result[i] = count + 1

#卡方检验
def chi2_func(features, label, order):
    print("\n卡方过滤计算结果:")
    result1 = SelectKBest(chi2, k = 16)  # 选择K个最好的特征，返回选择特征后的数据
    result1.fit_transform(features, label)

def main_func():
    # 导入数据集
    data = pd.read_table('2-AS.txt',index_col=False)
    features = data[feature_head]
    label = data["LCC"]

    #卡方分布
    order1 = np.zeros(16, 'i')
    chi2_func(features, label, order1)

main_func()
```

- 互信息分类

```
import numpy as np
import pandas as pd
from sklearn.feature_selection import SelectKBest
from sklearn.feature_selection import mutual_info_classif
from sklearn import metrics
from minepy import MINE
from prettytable import PrettyTable

#特征名称
feature_head = ['max_degree','fail_node_degree','fail_neber_degree','fail_degree_sum',
                    'max_load','big_load_num','fail_load_sum','fail_num',
                    'first_round_fail','neber_fail_num','fail_round','subgraph_num',
                    'fail_node_load','load_change','degree_change','fail_neber_load']
number = [1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16]

#按照得到结果进行排序
def sort_func(data, result):
    for i in range(data.shape[0]):
        count = 0
        for j in range(data.shape[0]):
            if(data[j] > data[i]):
                count = count + 1
        result[i] = count + 1
```

```
#互信息
def mutual_info_func(features, label, order):
    print("\n互信息分类计算结果:")
    result2 = mutual_info_classif(features, label)


def main_func():
    # 导入数据集
    data = pd.read_table('2-AS.txt',index_col=False)
    features = data[feature_head]
    label = data["LCC"]

    #互信息
    order2 = np.zeros(16, 'i')
    mutual_info_func(features, label, order2)

main_func()
```

- 标准化互信息分类

```
import numpy as np
import pandas as pd
from sklearn.feature_selection import SelectKBest
from sklearn.feature_selection import mutual_info_classif
from sklearn import metrics
from minepy import MINE
from prettytable import PrettyTable

#特征名称
feature_head = ['max_degree','fail_node_degree','fail_neber_degree','fail_degree_sum',
                    'max_load','big_load_num','fail_load_sum','fail_num',
                    'first_round_fail','neber_fail_num','fail_round','subgraph_num',
                    'fail_node_load','load_change','degree_change','fail_neber_load']
number = [1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16]

#按照得到结果进行排序
def sort_func(data, result):
    for i in range(data.shape[0]):
        count = 0
        for j in range(data.shape[0]):
            if(data[j] > data[i]):
                count = count + 1
        result[i] = count + 1

#标准化互信息
def normalized_mutual_info_func(features, label, order):
    print("\n标准化互信息分类计算结果:")
    i = 0
    result_NMI = np.zeros(16)
    for index, row in features.iteritems():
        result_NMI[i] = metrics.normalized_mutual_info_score(features[index], label)
        i = i + 1
```

```
def main_func():
    # 导入数据集
    data = pd.read_table('2-AS.txt',index_col=False)
    features = data[feature_head]
    label = data["LCC"]

    #标准化互信息
    order3 = np.zeros(16, 'i')
    normalized_mutual_info_func(features, label, order3)

main_func()
```

- 最大互信息数(MIC)

```
import numpy as np
import pandas as pd
from sklearn.feature_selection import SelectKBest
from sklearn.feature_selection import mutual_info_classif
from sklearn import metrics
from minepy import MINE
from prettytable import PrettyTable

#特征名称
feature_head = ['max_degree','fail_node_degree','fail_neber_degree','fail_degree_sum',
                    'max_load','big_load_num','fail_load_sum','fail_num',
                    'first_round_fail','neber_fail_num','fail_round','subgraph_num',
                    'fail_node_load','load_change','degree_change','fail_neber_load']
number = [1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16]

#按照得到结果进行排序
def sort_func(data, result):
    for i in range(data.shape[0]):
        count = 0
        for j in range(data.shape[0]):
            if(data[j] > data[i]):
                count = count + 1
        result[i] = count + 1

#最大互信息系数MIC
def mic(x, y):
    m = MINE()
    m.compute_score(x, y)
    return (m.mic(), 0.5)  # 选择 K 个最好的特征，返回特征选择后的数据

def mic_func(features, label, order):
    print("\n最大互信息系数(MIC)计算结果:")
    mic_select = SelectKBest(lambda X, y: tuple(map(tuple, np.array(list(map(lambda x:
mic(x, y), X.T))).T)), k=16)
    mic_select.fit_transform(features, label)  #选择K个最好的特征，返回特征选择后的数据
    mic_scores = mic_select.scores_  # 特征与最大信息系数的对应
```

```
def main_func():
    # 导入数据集
    data = pd.read_table('2-AS.txt',index_col=False)
    features = data[feature_head]
    label = data["LCC"]

    #最大互信息系数(MIC)
    order4 = np.zeros(16, 'i')
    mic_func(features, label, order4)

main_func()
```

## 2.任务二

```
import numpy as np
import pandas as pd
from sklearn.neighbors import KNeighborsClassifier  # 引入KNN分类器
from sklearn.metrics import accuracy_score
from prettytable import PrettyTable
import matplotlib.pyplot as plt

#特征名称
feature_head_12 = ['fail_neber_degree','fail_degree_sum',
                   'max_load','fail_load_sum','fail_num',
                   'first_round_fail','neber_fail_num','subgraph_num',
                   'fail_node_load','load_change','degree_change','fail_neber_load']

def main_func():
    # 导入数据集
    data = pd.read_table('1-AS_cla.txt', sep='\t', index_col=False)
    features = data[feature_head_12]
    label = data["LCC"]

    train_data_feature = pd.DataFrame()
    train_data_label = pd.DataFrame()
    test_data_feature = pd.DataFrame()
    test_data_label = pd.DataFrame()

    # 将源数据分为训练集和测试集
    for i in range(features.shape[0]):
        temp = pd.DataFrame([label.loc[i]], columns=['LCC'])
        if ((i + 1) % 10 == 2):
            test_data_feature = test_data_feature.append(features.loc[i],
ignore_index=True)
            test_data_label = test_data_label.append(temp, ignore_index=True)
        else:
            train_data_feature = train_data_feature.append(features.loc[i],
ignore_index=True)
            train_data_label = train_data_label.append(temp, ignore_index=True)

    accuracy_score_list = []    #准确率
```

```
    #KNN
    for k in range(20):
        knn = KNeighborsClassifier(n_neighbors = k + 1)  # 调用KNN分类器
        knn.fit(train_data_feature.values, train_data_label.values)  # 训练KNN分类器
        test_pred = knn.predict(test_data_feature)
        score = accuracy_score(test_pred, test_data_label)
        accuracy_score_list.append(score)

main_func()
```

### 3.任务三

```
import numpy as np
import pandas as pd
from sklearn.neighbors import KNeighborsClassifier  # 引入KNN分类器
from sklearn.metrics import accuracy_score
from prettytable import PrettyTable
import matplotlib.pyplot as plt

def main_func():
    # 导入数据集
    data = pd.read_csv("2-USAirlines_cla.txt", sep='\t', index_col=False)
    label = data.iloc[:, 17]

    accuracy_score_list = []  # 准确率
    for number in range (2,18):
        features = data.iloc[:, 1:number]

        train_data_feature = pd.DataFrame()
        train_data_label = pd.DataFrame()
        test_data_feature = pd.DataFrame()
        test_data_label = pd.DataFrame()

        # 将源数据分为训练集和测试集
        for i in range(features.shape[0]):
            temp = pd.DataFrame([label.loc[i]], columns=['LCC'])
            if ((i + 1) % 10 == 2):
                test_data_feature = test_data_feature.append(features.loc[i],
ignore_index=True)
                test_data_label = test_data_label.append(temp, ignore_index=True)
            else:
                train_data_feature = train_data_feature.append(features.loc[i],
ignore_index=True)
                train_data_label = train_data_label.append(temp, ignore_index=True)

       #KNN
        knn = KNeighborsClassifier()  # 调用KNN分类器
        knn.fit(train_data_feature.values, train_data_label.values)  # 训练KNN分类器
        test_pred = knn.predict(test_data_feature)
        score = accuracy_score(test_pred, test_data_label)
```

```
        accuracy_score_list.append(score)

main_func()
```

## 五、参考文献


机器学习如何计算特征的重要性_简介机器学习中的特征工程

机器学习如何计算特征的重要性_机器学习之特征工程

**sklearn.neighbors.KNeighborsClassifier — scikit-learn 1.0.1 documentation**