

利用随机森林算法判断钓鱼网站模型

一. 数据分析和探索

数据集: Phishing Websites, 数据集描述如图 1-1。

<http://archive.ics.uci.edu/ml/datasets/Phishing+Websites>

Data Set Characteristics:	N/A	Number of Instances:	2456	Area:	Computer Security
Attribute Characteristics:	Integer	Number of Attributes:	30	Date Donated	2015-03-26
Associated Tasks:	Classification	Missing Values?	N/A	Number of Web Hits:	78862

图 1-1

数据集是关于网络钓鱼网站数据, 它包含 30 个属性值, 每个属性值为整型数, 属性取值为 $\{1, 0, -1\}$ 。结果为二元分类, 用 $\{-1, 1\}$ 编码。

这三十个属性在数据集作者的描述中, 这些特征已经被证明为合理有效的。鉴于这些数据的特征与信安专业有关, 所以我们对这 30 个属性进行了研究, 特征图如图 1-2 (打包文件中包含原图以及每个特征的详细介绍)。30 个特征值分别基于地址栏、基于异常特征、HTML 和 JavaScript 的功能、域特征四种, 我们拿到的数据集是贡献者按照规则将每个特征编码为 $-1, 0, 1$, 分别表示这个网站是钓鱼网站、可疑网站和合法网站。

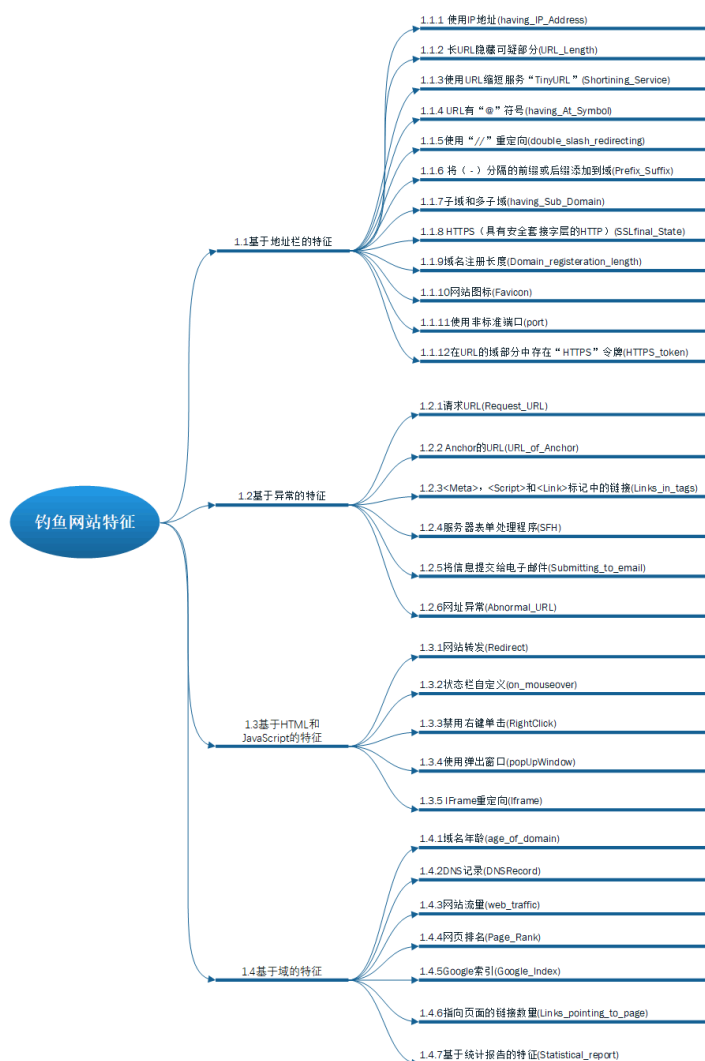
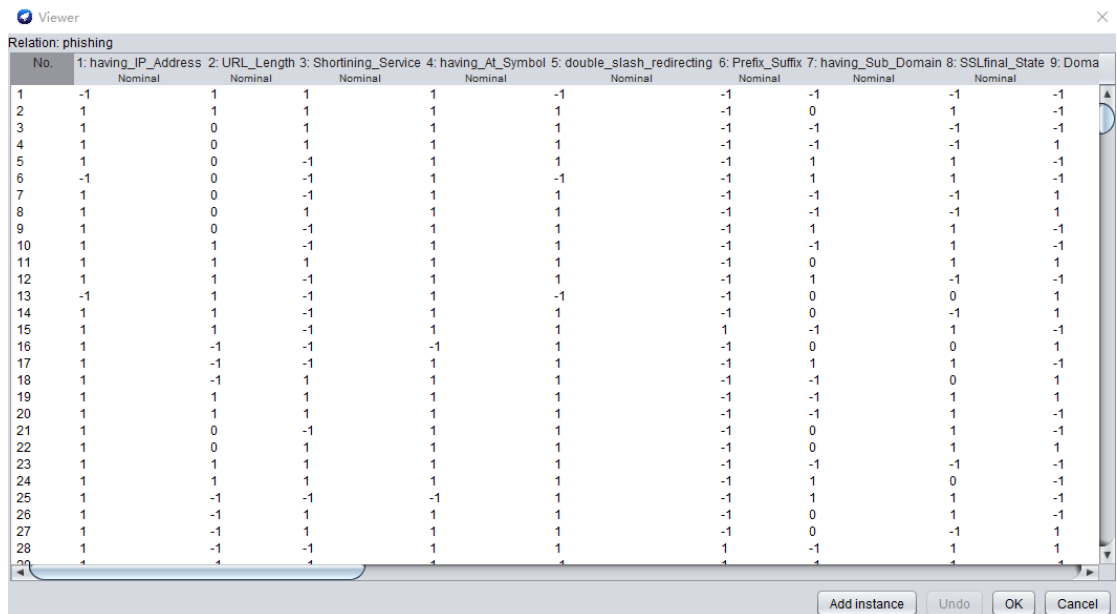


图 1-2

二. 数据预处理

1. 首先我们用 Weka 软件将下载的数据集. arff 文件转为. csv 文件
用 weka 工具查看数据如下:



No.	1: having_IP_Address	2: URL_Length	3: Shortining_Service	4: having_At_Symbol	5: double_slash_redirecting	6: Prefix_Suffix	7: having_Sub_Domain	8: SSLfinal_State	9: Doma
1	-1	1	1	1	-1	-1	-1	-1	-1
2	1	1	1	1	1	-1	0	1	-1
3	1	0	1	1	1	-1	-1	-1	-1
4	1	0	1	1	1	-1	-1	-1	1
5	1	0	-1	1	1	-1	1	1	-1
6	-1	0	-1	1	-1	-1	1	1	-1
7	1	0	-1	1	1	-1	-1	-1	1
8	1	0	1	1	1	-1	-1	-1	1
9	1	0	-1	1	1	-1	1	1	-1
10	1	1	-1	1	1	-1	-1	1	-1
11	1	1	1	1	1	-1	0	1	1
12	1	1	-1	1	1	-1	1	-1	-1
13	-1	1	-1	1	-1	-1	0	0	1
14	1	1	-1	1	1	-1	0	-1	1
15	1	1	-1	1	1	1	-1	1	-1
16	1	-1	-1	-1	1	-1	0	0	1
17	1	-1	-1	1	1	-1	1	1	-1
18	1	-1	1	1	1	-1	-1	0	1
19	1	1	1	1	1	-1	-1	1	1
20	1	1	1	1	1	-1	-1	1	-1
21	1	0	-1	1	1	-1	0	1	-1
22	1	0	1	1	1	-1	0	1	1
23	1	1	1	1	1	-1	-1	-1	-1
24	1	1	1	1	1	-1	1	0	-1
25	1	-1	-1	-1	1	-1	1	1	-1
26	1	-1	1	1	1	-1	0	1	-1
27	1	-1	1	1	1	-1	0	-1	1
28	1	-1	-1	1	1	1	-1	1	1

图 2-1

可以确定数据没有缺省值，是离散值，我们认为也不必进行再次独热编码。

2. 对 30 个特征值的取值分布可视化：数据分布直方图

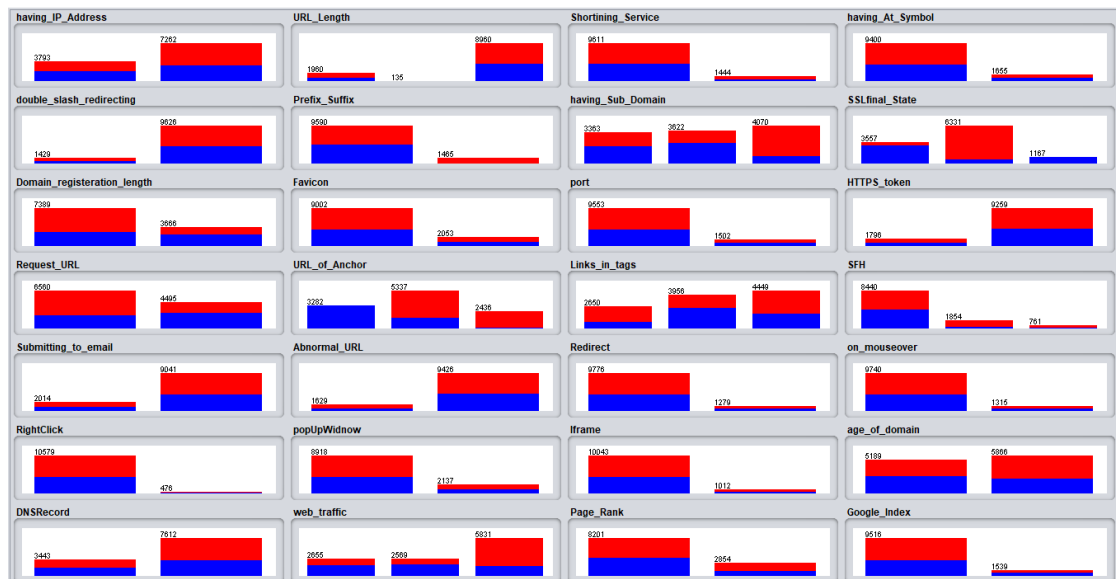


图 2-2

可以发现存在有些特征值对结果的影响比较小。

3. 对于最后的结果取值分布

Name: Result		Type: Nominal	
Missing: 0 (0%)		Unique: 0 (0%)	
		Distinct: 2	
No.	Label	Count	Weight
1	-1	4898	4898.0
2	1	6157	6157.0

图 2-3

可以看到数据集是相对平衡的。所以最终我们对数据的预处理仅限于数据格式的变换。

三. 分类算法的选择

1. 首先我们运用 weka 软件尝试各种算法，并用 E-charts 将其可视化。

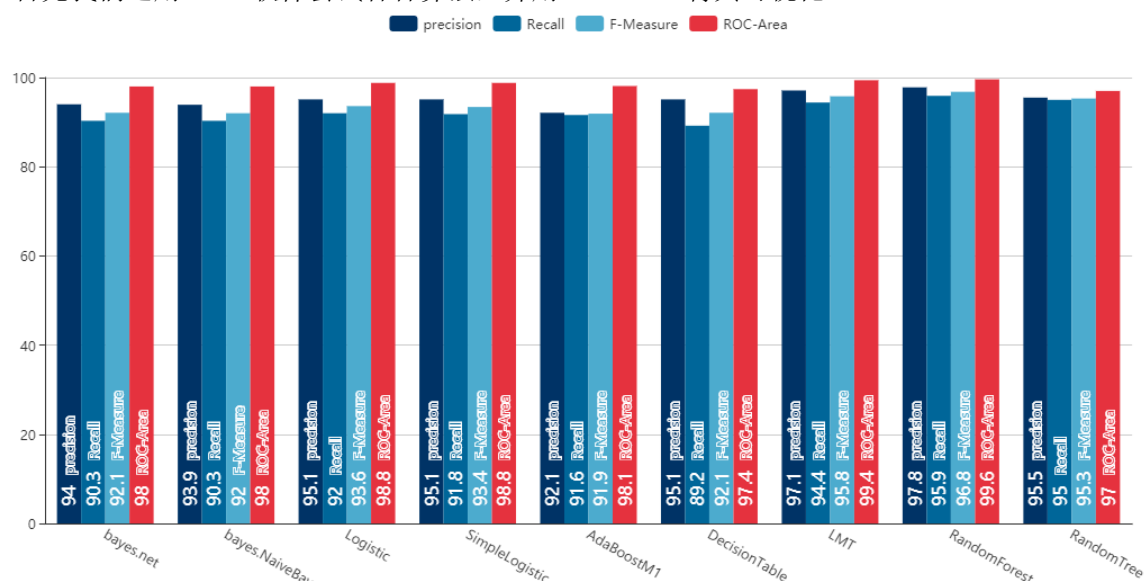


图 3-1

可以分析出我们的数据是非常好的，用 9 中学习算法，准确率都在 90%以上，相比较而言，随机森林的四个指标最好，所以我们选择随机森林的算法。

利用 Weka 内置的 RandomForest 算法,利用 Percentage split 将我们学习的结果如下：

```

=== Summary ===

Correctly Classified Instances      3178           95.8384 %
Incorrectly Classified Instances    138            4.1616 %
Kappa statistic                     0.9156
Mean absolute error                 0.043
Root mean squared error             0.191
Relative absolute error              8.7151 %
Root relative squared error          38.4523 %
Total Number of Instances          3316

=== Detailed Accuracy By Class ===

            TP Rate  FP Rate  Precision  Recall   F-Measure  MCC      ROC Area  PRC Area  Class
            0.950    0.035    0.955     0.950    0.953      0.916    0.970     0.951     -1
            0.965    0.050    0.961     0.965    0.963      0.916    0.970     0.961      1
Weighted Avg.   0.958    0.043    0.958     0.958    0.958      0.916    0.970     0.957

=== Confusion Matrix ===

  a    b  <-- classified as
1395   73 |    a = -1
 65 1783 |    b = 1

```

图 3-2

2. 在文章《Do we Need Hundreds of Classifiers to Solve Real World Classification Problems?》（网址：<http://jmlr.org/papers/v15/delgado14a.html>）测试了 179 种分类模型在 UCI 所有的 121 个数据上的性能，发现 Random Forests 和 SVM 性能最好。最近几年的国内外大赛，包括 2013 年百度校园电影推荐系统大赛、2014 年阿里巴巴天池大数据竞赛以及 Kaggle 数据科学竞赛，参赛者对随机森林的使用占有相当高的比例。一

大部分成功进入答辩的队伍也都选择了 Random Forest 或者 GBDT 算法。所以可以看出, Random Forest 在准确率方面还是相当有优势的。

3. 随机森林算法具有极好的准确率, 能够有效运行在大数据集上, 能够处理具有高维特征的输入样本, 能评估各个特征在分类问题的重要性。很适合我们这个数据集的学习。

四. 随机森林模型建立

此次运用工具 SK-Learn, 用 numpy 和 pandas 包做数据处理。使用 SK-Learn 的集成算法中的 RandomForestClassifier 函数。

1. 导入数据: 从网站上下载的数据为 arff 格式, 导入 arff 包以导入 arff 格式的数据集。并将其转换为 array 格式。
2. 划分测试和训练集: 利用 train_test_split 函数以 7:3 的比例划分训练集和测试集。由于这个公开的数据集没有公开划分测试集和数据集, 所以我们在数据自己定义的划分比例。这样的缺点在于没有对比, 无法与网络中其他人做出的模型用同一个测试集的评估参数比较。
3. 导入训练模型。RandomForestClassifier 函数具有 17 个参数, 用默认值做训练, 用测试集测试评估模型。

```
➤ RandomForestClassifier(bootstrap=True, class_weight=None,
                           criterion='gini', max_depth=None,
                           max_features='auto', max_leaf_nodes=None,
                           min_impurity_decrease=0.0, min_impurity_split=None,
                           min_samples_leaf=1, min_samples_split=2,
                           min_weight_fraction_leaf=0.0, n_estimators=10,
                           n_jobs=1, oob_score=True,
                           random_state=10, verbose=0, warm_start=False)
```

- 使用 OOB Score 和 AUC Score 评价模型

OOB Score: 即采用袋外样本评估模型好坏。默认为 False, 我们用 True。袋外分数反映一个模型拟合后的泛化能力。

此时为: OOB Score(data): 0.951279

AUC Score (data): 0.994785

4. 参数调优

使用 GridSearchCV 函数调参。网格搜索算法通过遍历给定的参数组合, 通过交叉验证的方式来优化模型表现的方法。我们根据默认的参数确定参数的范围, 进行逐个参数的调优。

- 1) 我们首先对 n_estimators 进行网格搜索: 弱学习器的最大迭代次数

```
➤ 运行结果: [mean: 0.97798, std: 0.00298, params: {'n_estimators': 5},
              mean: 0.98237, std: 0.00144, params: {'n_estimators': 10}, mean:
              0.98286, std: 0.00110, params: {'n_estimators': 15}, mean: 0.98370,
              std: 0.00105, params: {'n_estimators': 20}, mean: 0.98416, std:
              0.00092, params: {'n_estimators': 25}, mean: 0.98401, std: 0.00124,
              params: {'n_estimators': 30}, mean: 0.98386, std: 0.00120, params:
              {'n_estimators': 35}, mean: 0.98407, std: 0.00151, params:
              {'n_estimators': 40}, mean: 0.98412, std: 0.00139, params:
              {'n_estimators': 45}] {'n_estimators': 25} 0.9841623601047979
```

➤ 所以在给定的范围 5-45 之间较好的最大迭代次数为 25.

- 此时评估参数为:

```
RandomForestClassifier(bootstrap=True, class_weight=None, criterion='gini',
                        max_depth=None, max_features='auto', max_leaf_nodes=None,
                        min_impurity_decrease=0.0, min_impurity_split=None,
                        min_samples_leaf=1, min_samples_split=2,
                        min_weight_fraction_leaf=0.0, n_estimators=30, n_jobs=1,
                        oob_score=True, random_state=10, verbose=0, warm_start=False)
OOB Score(data):0.967433
AUC Score (data): 0.994339
```

2) 对决策树最大深度 `max_depth` 和内部节点再划分所需最小样本数 `min_samples_split` 进行网格搜索

- 部分运行结果如下:

```
mean: 0.98129, std: 0.00362, params: {'max_depth': 90,
'min_samples_split': 5}, mean: 0.98129, std: 0.00362, params:
{'max_depth': 90, 'min_samples_split': 6}, mean: 0.98129, std:
0.00362, params: {'max_depth': 90, 'min_samples_split': 7}, mean:
0.98129, std: 0.00362, params: {'max_depth': 90, 'min_samples_split':
8}, mean: 0.98129, std: 0.00362, params: {'max_depth': 90,
'min_samples_split': 9}] {'max_depth': 20, 'min_samples_split': 2}
0.9812931520940518
```

- 可以得到 `max_depth` 在 10-100 的范围内最优的为 20, 对于内部节点再划分所需最小样本数 `min_samples_split`, 我们暂时不能一起定下来, 因为这个还和决策树其他的参数存在关联。
- 此时评估参数为:

```
OOB Score(data):0.964590
AUC Score (data): 0.996696
```

3) 对内部节点再划分所需最小样本数 `min_samples_split` 和叶子节点最少样本数 `min_samples_leaf` 一起调参

- 部分运行结果:

```
mean: 0.98680, std: 0.00596, params: {'min_samples_leaf': 9,
'min_samples_split': 7}, mean: 0.98680, std: 0.00596, params:
{'min_samples_leaf': 9, 'min_samples_split': 8}, mean: 0.98680,
std: 0.00596, params: {'min_samples_leaf': 9, 'min_samples_split':
9}] {'min_samples_leaf': 1, 'min_samples_split': 4}
0.9940510366501213
```

- 可以得到 `min_samples_split` 在 2-10 范围优值为 4, `min_samples_leaf` 在 1-10 范围内优值为 1。
- 此时评估参数为:

```
max_depth=20, max_features='auto', max_leaf_nodes=None,
predictions[k].sum(axis=1)[: , np.newaxis])
min_impurity_decrease=0.0, min_impurity_split=None,
min_samples_leaf=1, min_samples_split=4,
min_weight_fraction_leaf=0.0, n_estimators=25, n_jobs=1,
oob_score=True, random_state=10, verbose=0, warm_start=False)
OOB Score(data):0.966400
AUC Score (data): 0.996200
```

4) 最后在对最大特征数 max_features 做调参.

➤ 部分运行结果:

```
mean: 0.99259, std: 0.00136, params: {'max_features': 26}, mean:
0.99250, std: 0.00140, params: {'max_features': 28}]
{'max_features': 4} 0.9945125597954896
```

➤ 可以得到 max_features 在 2-30 之间优值为 4

➤ 此时评估参数为:

```
OOB Score(data):0.963815
AUC Score (data): 0.996014
```

5. 最终我们的模型参数为:

```
<bound method BaseEstimator.get_params of RandomForestClassifier(bootstrap=True, class_weight=None, criterion='gini',
max_depth=20, max_features=4, max_leaf_nodes=None,
min_impurity_decrease=0.0, min_impurity_split=None,
min_samples_leaf=1, min_samples_split=4,
min_weight_fraction_leaf=0.0, n_estimators=25, n_jobs=1,
oob_score=True, random_state=10, verbose=0, warm_start=False)>
```

用这个模型 fit 训练集, 并将模型用 pickle 保存。

6. 考虑降维处理数据

特征重要程度图

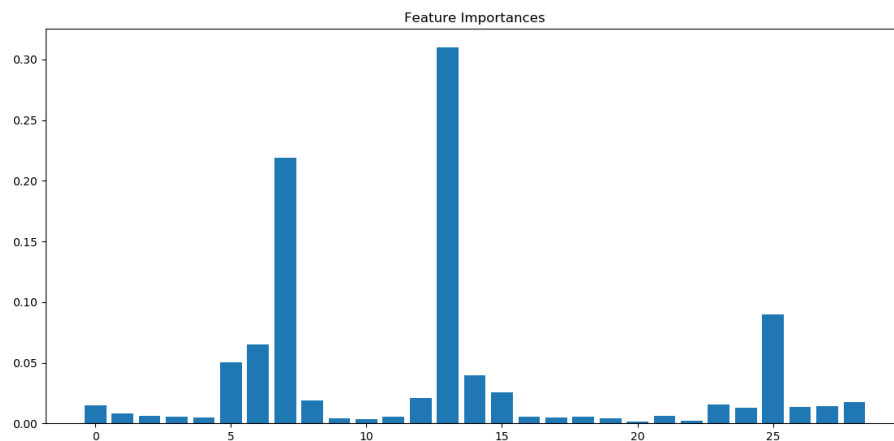


图 4-1

不同特征的重要性存在一定差别, 考虑减少特征, 选用循环特征小贱 RF 算法, 但是最终的准确度等参数并没有明显提高。考虑到数据集本身的特征, 就像在数据分析中所言, 这 30 个特征是数据收集者做过证明对结果有影响的特征, 所以减少特征不会对模型有很好的提升。其次, 与随机森林算法本身有关。随机森林对多特征有很好的效果。

7. 随机森林可视化

为了更加清晰的分析决策树生成的过程, 使用 graphviz 包对决策树可视化

五. 模型评估

利用测试集对模型进行评估

score:0.971661				
	precision	recall	f1-score	support
0	0.98	0.96	0.97	1521
1	0.96	0.98	0.97	1796
avg / total	0.97	0.97	0.97	3317

图 5-1

- 1. score 为 0.963531
- 2. precision、recall、F1、support 值如图 5-1
- 3. 混淆矩阵为

[[1454 67]
[27 1769]]

- 4. ROC 曲线，如图 5-2:

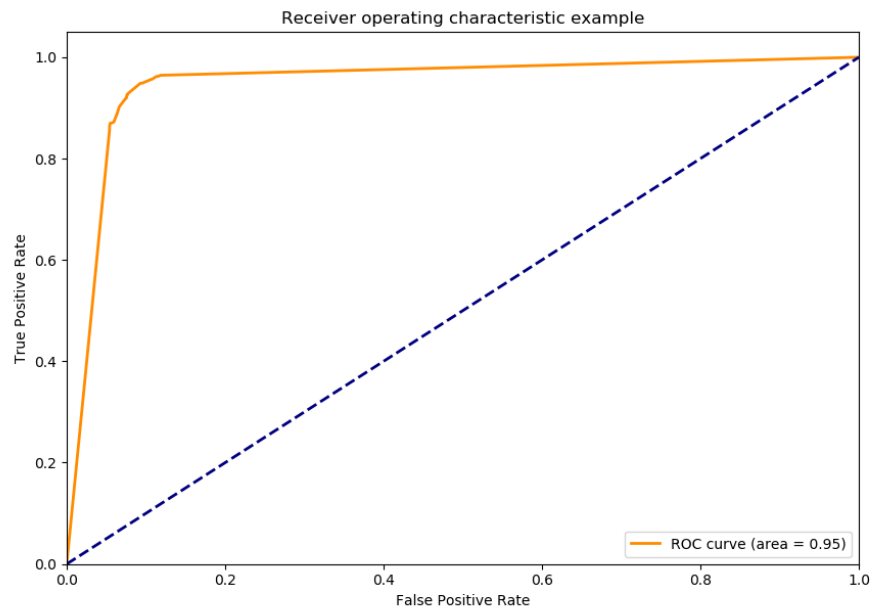


图 5-2

5. 整体评价：可以看到模型的效果还是不错的，各项参数与调参之前和用 Weka 直接调用的算法要好一些，但是因为算法本身的准确度就很高，所以提升的效果不是很明显。

六. 项目分工

本小组共计三人

组长：于林林（515030910078）：分类算法选择、模型建立与评估、文档撰写

组员：王 晨（515030910106）：数据分析与探索、数据预处理

周奕雯（5140809049）：数据可视化、评估模型可视化