南开大学

Python 大作业

实验报告

姓名：聂志强

学号：2012307

学院：网络空间安全学院

专业：信息安全

授课教师：陈晨

基于循环神经网络的新闻检测

一．项目概述

总述：本项目主要采用循环神经网络模型对新闻进行真假预测，为了客观对比循环神经网络训练结果，增加了传统机器学习的五种方法[朴素贝叶斯、决策树、随机森林、KNN、支持向量机(高斯核函数、Sigmoid 核函数、线性核函数)]，通过给定的训练集和测试集观察发现谣言和非谣言比例约为 1：6，而 AUC 非常适合评价样本不平衡中的分类器性能，所以应该主要通过 AUC 指标衡量模型/分类器性能，经数次验证，5 种传统机器学习方法测试集 AUC 指标集中在 0.67—0.70，深度学习测试集 AUC 指标集中在 0.85—0.88，较机器学习方法有大幅提升。

定义：给定一个信息的标题，判别信息真伪。

输入：信息标题

输出：真伪标签（0：消息为真，1：消息为假）

二．开发软件

百度 PaddlePaddle 平台

三．整体流程（仅展示核心代码）

A.机器学习方法：

1. 文本预处理

核心思想：根据已经划分好的训练集和测试集，分别提取训练集和测试集的 title 和 label 保存至 x_train，y_train，x_test，y_test.

```
    for line in train_lines:
        words = line.split('\t')[-1].replace('\n', '')
        label = line.split('\t')[0]
        x_train.append(words)
        y_train.append(label)
```

2. 特征提取

核心思想：

TF-IDF，统计评估一字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。TF-IDF 的主要思想是如果某个词或短语在一篇文章中出现的概率高，并且在其他文章中很少出现，则认为此词或者短语具有很好的类别区分能力，适合用来分类。TF - 词频，指的是某一个给定的词语在该文件中出现的频率 IDF - 逆向文档频率，是一个词语普遍重要性的度量。某一特定词语的 IDF 可以由总文件数目除以包含该词语之文件的数目，再将得到的商取以 10 为底的对数得到

```
    transfer = TfidfVectorizer()
```

3. 设置预估器

(1) 朴素贝叶斯分类器

核心代码：

```
estimator = MultinomialNB(alpha = 1.0)
```

alpha：拉普拉斯平滑系数，防止计算出的分类概率为 0

(2) 决策树分类器

原理：

信息增益越大，则表示使用该特征对数据集划分所获得的“纯度提升”越大。所以信息增益可以用于决策树划分属性的选择，其实就是选择信息增益最大的属性，即利用信息增益来划分属性。特征 A 对训练数据集 D 的信息增益 g(D,A)，定义为集合 D 的信息熵 H(D)与特征 A 给定条件下 D 的信息条件熵 H(D|A)之差，即

公式为 $$g(D,A)=H(D)-H(D|A)$$

信息熵的计算： $$H(D)=-\sum_{k=1}^{K}\frac{|C_k|}{|D|}\log\frac{|C_k|}{|D|}$$

条件熵的计算： $$H(D|A)=-\sum_{i=1}^{n}\frac{|D_i|}{|D|}H(D_i)=-\sum_{i=1}^{n}\frac{|D_i|}{|D|}\sum_{k=1}^{K}\frac{|D_{ik}|}{|D_i|}\log\frac{|D_{ik}|}{|D_i|}$$

核心代码：

```
estimator = DecisionTreeClassifier(criterion="entropy")
```

(3) 随机森林

原理：

随机森林是一个包含多个决策树的分类器，并且其输出的类别是由各个树输出的类别的众数而定。一次随机选出一个样本，重复 N 次（N 来表示训练用例（样本）的个数），有可能出现重复的样本。随机去选出 m 个特征，m <<M（M 表示特征数目），建立决策树（采取 bootstrap 抽样）

核心代码：

```
# 1.随即森林预估器
estimator =RandomForestClassifier()

# 2.交叉验证
# 超参数搜索-网格搜索
param_dict = {"n_estimators": [90,100,110], "max_depth":[500,750,1000]}
estimator = GridSearchCV(estimator, param_dict, cv=3)
```

(4) KNN

原理：

如果一个样本在特征空间中的 k 个最相似(即特征空间中最邻近（通过欧氏距离计算）)的样本中的大多数属于某一个类别，则该样本也属于这个类别。距离公式（欧式距离）：

假设 $a(a1,a2,a3),b(b1,b2,b3)$，则 ab距离 $=\sqrt{(a1-b1)^2+(a2-b2)^2+(a3-b3)^2}$

核心代码：

```
# 1.KNN 算法预估器
estimator = KNeighborsClassifier()

# 2.交叉验证
# 超参数搜索-网格搜索
param_dict = {"n_neighbors": [1, 2, 3, 4, 5]}
estimator = GridSearchCV(estimator, param_dict, cv=10)
estimator.fit(x_train, y_train)
```

(5) 支持向量机

原理：

支持向量机是利用分类间隔的思想进行训练的，它依赖于对数据的预处理，即在更高维的空间表达原始模式。支持向量机将数据从原始空间映射到高维空间的目的就是找到一个最优的分类面从而使得分类间隔 margin 最大，则分别属于两类的原始数据就能够被一个超平面来分隔。当样本集 $D=(x_i,y_i),i\in N$ 在 d 维空间中不是线性可分时，也就是一个超平面无法有效分开这些样本时，可以通过一个函数将样本集从 d 维升到 d + 1 维，从而从高维上来进行划分。这个从 d 维到 d + 1 维的过程就是通过核函数来实现。有以下几种常用核函数：

名称	表达式	参数
线性核	$\kappa(\boldsymbol{x}_i,\boldsymbol{x}_j)=\boldsymbol{x}_i^{\mathrm{T}}\boldsymbol{x}_j$	
多项式核	$\kappa(\boldsymbol{x}_i,\boldsymbol{x}_j)=(\boldsymbol{x}_i^{\mathrm{T}}\boldsymbol{x}_j)^d$	$d\geqslant 1$ 为多项式的次数
高斯核	$\kappa(\boldsymbol{x}_i,\boldsymbol{x}_j)=\exp\left(-\frac{\|\boldsymbol{x}_i-\boldsymbol{x}_j\|^2}{2\sigma^2}\right)$	$\sigma>0$ 为高斯核的带宽(width)
拉普拉斯核	$\kappa(\boldsymbol{x}_i,\boldsymbol{x}_j)=\exp\left(-\frac{\|\boldsymbol{x}_i-\boldsymbol{x}_j\|}{\sigma}\right)$	$\sigma>0$
Sigmoid 核	$\kappa(\boldsymbol{x}_i,\boldsymbol{x}_j)=\tanh(\beta\boldsymbol{x}_i^{\mathrm{T}}\boldsymbol{x}_j+\theta)$	tanh 为双曲正切函数，$\beta>0,\theta<0$

核心代码：

高斯核函数

```
estimator=svm.SVC(kernel='rbf',C=1000)
```

Sigmoid 核函数

```
estimator=svm.SVC(kernel='sigmoid',C=1000)
```

线性核函数

```
model=svm.SVC(kernel='linear',C=1000)
```

4. 交叉验证与网格搜索

交叉验证：将拿到的训练数据，分为训练和验证集。以下图为例：将数据分成 4 份，其中一份作为验证集。然后经过 4 次(组)的测试，每次都更换不同的验证集。即得到 4 组模型的结果，取平均值作为最终结果。

超参数搜索-网格搜索：通常情况下，有很多参数是需要手动指定的（如 k-近邻算法中的 K 值），但是手动过程繁杂，所以需要对模型预设几种超参数组合。每组超参数都采用交叉验证来进行评估。最后选出最优参数组合建立模型

5. 模型评估

（1）混淆矩阵：

在分类任务下，预测结果与正确标记之间存在四种不同的组合，构成混淆矩阵，见图 1

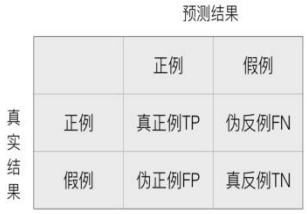


图 1

图 2

图 3

（2）精确率：预测结果为正例样本中真实为正例的比例，见图 2

（3）召回率：真实为正例的样本中预测结果为正例的比例，检验是否查的全，见图 3

（4）F1-score：反映了模型的稳健型

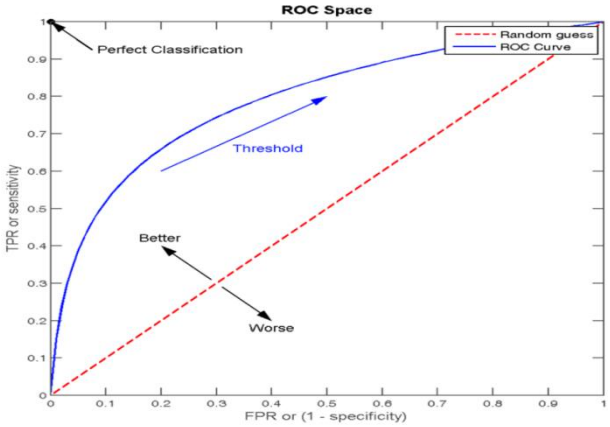
$$F1 = \frac{2TP}{2TP + FN + FP} = \frac{2 \cdot Precision \cdot Recall}{Precision + Recall}$$

（5）ROC 曲线与 AUC 指标：AUC 非常适合评价样本不平衡中的分类器性能

TPR = TP / (TP + FN)：所有真实类别为 1 的样本中，预测类别为 1 的比例
FPR = FP / (FP + FN)：所有真实类别为 0 的样本中，预测类别为 1 的比例
ROC 曲线的横轴就是 FPRate，纵轴就是 TPRate，当二者相等时，表示的意

义则是：对于不论真实类别是 1 还是 0 的样本，分类器预测为 1 的概率是相等的，此时 AUC 为 0.5

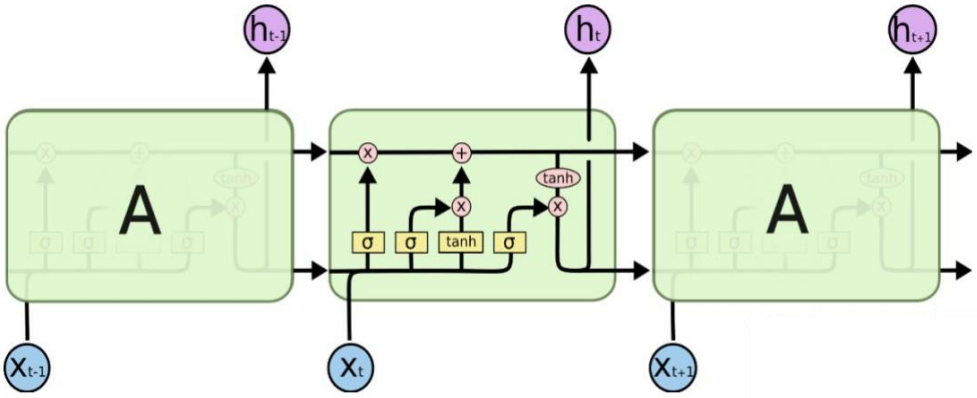
AUC：ROC 曲线面积



B. 深度学习方法—LSTM 循环神经网络

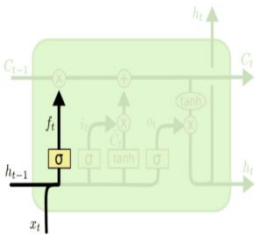
1. 原理概述：

简述：长短时记忆神经网络(LSTM)是循环神经网络的一个变体，可以有效地解决长期依赖问题/梯度消失问题。LSTM 模型的关键是引入了一组记忆单元，允许网络可以学习何时遗忘历史信息，何时用新信息更新记忆单元。



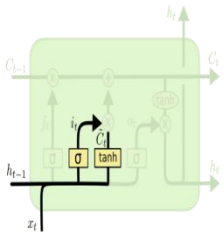
(1) 遗忘门（如图 4）

遗忘门的作用就是用来选择这些信息并“忘记”它们。遗忘门决定了细胞状态 C_{t-1} 中的哪些信息将被遗忘。包含一个 sigmod 神经网络层（神经网络参数为 W_f, b_f），接收 t 时刻的输入信号 x_t 和 t-1 时刻 LSTM 的上一个输出信号 h_{t-1}，这两个信号进行拼接以后共同输入到 sigmod 神经网络层中，然后输出信号 f_t, f_t 是一个 0 到 1 之间的数值，并与 C_{t-1} 相乘来决定 C_{t-1} 中的哪些信息将被保留，哪些信息将被舍弃



$$f_t = \sigma(W_f \cdot [h_{t-1}, x_t] + b_f)$$

图 4



$$i_t = \sigma(W_i \cdot [h_{t-1}, x_t] + b_i)$$

$$\tilde{C}_t = \tanh(W_C \cdot [h_{t-1}, x_t] + b_C)$$

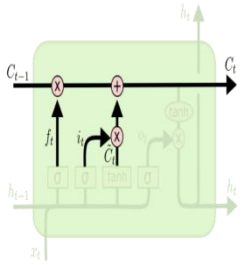
图 5

(2) 记忆门（如图 5）

记忆门包含 2 个部分。第一个是包含 sigmod 神经网络层（输入门,神经网络网络参数为 W_i,b_i）和一个 tanh 神经网络层（神经网络参数为 W_c,b_c）。sigmod 神经网络层接收 X_t 和 h_{t-1} 作为输入，然后输出一个 0 到 1 之间的数值 i_t 来决定哪些信息需要被更新；Tanh 神经网络层的作用是将输入的 X_t 和 h_{t-1} 整合，然后通过一个 tanh 神经网络层来创建一个新的状态候选向量。记忆门的输出由上述两个神经网络层的输出决定，i_t 与 $\tilde{C}_t$ 相乘来选择哪些信息将被新加入到 t 时刻的细胞状态 C_t 中。

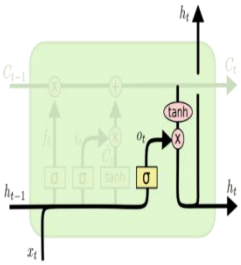
(3) 更新细胞状态（如图 6）

这里将遗忘门的输出与上一时刻的细胞状态相乘来选择遗忘和保留一些信息，将记忆门的输出与从遗忘门选择后的信息加和得到新的细胞状态。这就表示时刻 t 的细胞状态 C_t 已经包含了此时需要丢弃的 t-1 时刻传递的信息和 t 时刻从输入信号获取的需要新加入的信息。C_t 将继续传递到 t+1 时刻的 LSTM 网络中，作为新的细胞状态传递下去



$$C_t = f_t * C_{t-1} + i_t * \tilde{C}_t$$

图 6



$$o_t = \sigma(W_o [h_{t-1}, x_t] + b_o)$$

$$h_t = o_t * \tanh(C_t)$$

图 7

(4) 输出门（如图 7）

输出门就是将 t-1 时刻传递过来并经过了前面遗忘门与记忆门选择后的细胞状态与 t-1 时刻的输出信号 h_{t-1} 和 t 时刻的输入信号 X_t 整合到一起作为当前时刻的输出信号。X_t 和 h_{t-1} 经过一个 sigmod 神经网络层输出一个 0 到 1 之间的数值 o_t。C_t 经过一个 tanh 函数到一个在-1 到 1 之间的数值，并与 o_t 相乘得到输出信号 h_t，同时 h_t 也作为下一个时刻的输入信号传递到下一阶段。

2. 具体实现

（1）利用 jieba 进行分词

```
    def cut_word(text):
        text = " ".join(list(jieba.cut(text)))
        return text
```

（2）生成数据字典

核心思想：读取全部数据 - 转换为元组（去重）- 转换为字典（一个词对应一个数字）- 添加未知字符 - 保存到本地

（3）定义长短期记忆网络

核心思想：词嵌入（设立一个 embedding 层，用来把句子中每个词转换成向量 ）- 建立全连接层 - 定义长短期记忆操作 - 池化层 - 输出层（以 softmax 作为全连接激活函数）。其中 softmax 将神经网络输出转换成概率结果，通过概率大小比较即可预测该条新闻为 0/1。池化层：特征提取，通过去掉不重要的样本，进一步减少参数数量此处采取的是最大池化，即取池化窗口的最大值。

$$soft\max(y_i)=\frac{e^{y_i}}{\sum_{j=1}^{n}e^{y_i}}$$

```
    def lstm_net(ipt, input_dim):

        # 设立一个 embedding 层，用来把句子中每个词转换成向量
        emb = fluid.layers.embedding(input=ipt, size=[input_dim, 128], is_sparse=True)
        # 第一个全连接层
        fc1 = fluid.layers.fc(input=emb, size=128)
        # 进行一个长短期记忆操作
        lstm1, _ = fluid.layers.dynamic_lstm(input=fc1,size=128) #size=4*hidden_size, #返回：隐藏状态（hidden state），LSTM 的神经元状态
        # 第一个最大序列池操作 池化层（减少学习参数，防止过拟合）
        fc2 = fluid.layers.sequence_pool(input=fc1, pool_type='max')
        # 第二个最大序列池操作 池化层（减少学习参数，防止过拟合）
        lstm2 = fluid.layers.sequence_pool(input=lstm1, pool_type='max')
        # 以 softmax 作为全连接的输出层，大小为 2,也就是真假 ，达到分类效果
        out = fluid.layers.fc(input=[fc2, lstm2], size=2, act='softmax')

        return out
```

（4）定义损失函数和准确率

核心思想：定义损失函数时，因为是一个分类任务，所以使用的损失函数是交叉熵损失函数： $H_{y'}(y) = -\sum_i y_i' \log(y_i)$ 。使用 fluid.layers.accuracy()接口定义一个输出分类准确率的函数，可以方便在训练的时候，输出测试时的分类准确率，观察模型收敛的情况。

核心代码：

```
cost = fluid.layers.cross_entropy(input=model, label=label)
avg_cost = fluid.layers.mean(cost)
acc = fluid.layers.accuracy(input=model, label=label)
```

（5）定义优化方法

核心思想：这里使用的时 Adagrad 优化方法，Adagrad 其实是对学习率进行了一个约束。即：

$$n_t = n_{t-1} + g_t^2$$

$$\Delta\theta_t = -\frac{\eta}{\sqrt{n_t + \varepsilon}} \cdot g_t$$

此处，对 gt 从 1 到 t 进行一个递推形成一个约束项 regularizer （ε 用来保证分母非 0）：

$$-\frac{1}{\sqrt{\sum_{r=1}^{t}(g_r)^2 + \varepsilon}}$$

在参数空间更为平缓的方向，会取得更大的进步（因为平缓，所以历史梯度平方和较小，对应学习下降的幅度较小），当前期 gt 较小的时候， regularizer 较大，能够放大梯度；后期 gt 较大的时候，regularizer 较小，能够约束梯度。通过多次实验最终学习率取 0.001 最为合适。

核心代码：

```
optimizer = fluid.optimizer.AdagradOptimizer(learning_rate=0.001)
opt = optimizer.minimize(avg_cost)
```

（6）模型训练

核心思想：设定训练轮数 40 - 将数据分成若干批次(batch)，按照批次送入模型，根据优化方法降低 loss - 以列表的形式保存每轮训练的损失率，准确率和输出层的输出

（7）测试集验证

核心思想：利用训练集训练完成的模型进行测试集验证，计算损失率，准

确率，F1 召回率，精确率，并绘制 ROC 曲线计算 AUC。

（8）保存模型

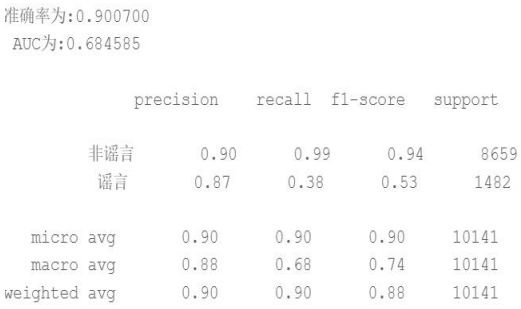
（9）模型检验：用训练好的模型进行预测并输出预测结果

四．结果评测

1. 机器学习——朴素贝叶斯

测试集评价指标与运行截图：

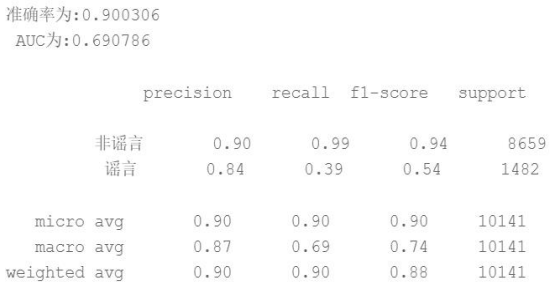
Accuracy ： 0.90
Precision： 非谣言 - 0.90 谣言 - 0.87
Recall ： 非谣言 - 0.99 谣言 - 0.38
F1 ： 非谣言 - 0.94 谣言 - 0.53
AUC ： 0.68



2. 机器学习——决策树

测试集评价指标与运行截图：

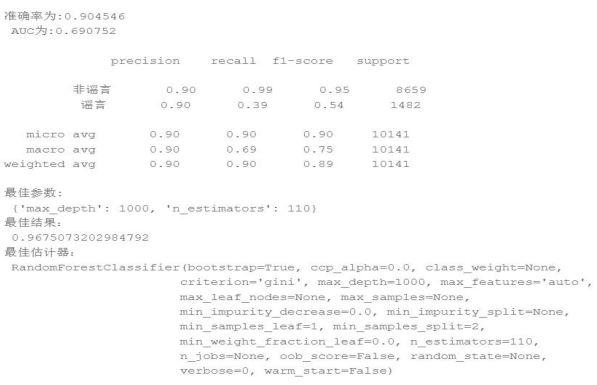
Accuracy ： 0.90
Precision： 谣言 - 0.90 非谣言 - 0.84
Recall ： 谣言 - 0.99 非谣言 - 0.39
F1 ： 谣言 - 0.94 非谣言 - 0.54
AUC ： 0.69



3. 机器学习——随机森林（含网格搜索和交叉验证）

测试集评价指标与运行截图：

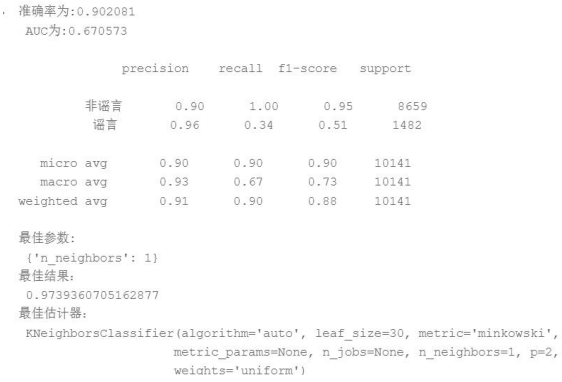
Accuracy ： 0.90
Precision： 谣言 - 0.90 非谣言 - 0.90
Recall ： 谣言 - 0.99 非谣言 - 0.39
F1 ： 谣言 - 0.95 非谣言 - 0.54
AUC ： 0.69



4. 机器学习——KNN（添加网格搜索和交叉验证）

测试集评价指标与运行截图：

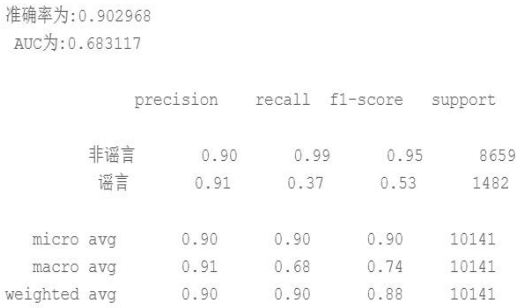
Accuracy ： 0.90
Precision: 非谣言 - 0.90 谣言 - 0.96
Recall ： 非谣言 - 1.00 谣言 - 0.34
F1 ： 非谣言 - 0.95 谣言 - 0.51
AUC ： 0.67



5. 机器学习——支持向量机（高斯核函数）

测试集评价指标与运行截图：

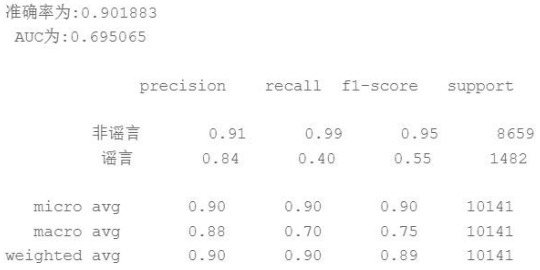
Accuracy ： 0.90
Precision: 谣言 - 0.90 非谣言 - 0.91
Recall ： 谣言 - 0.99 非谣言 - 0.37
F1 ： 谣言 - 0.95 非谣言 - 0.53
AUC ： 0.68



6. 机器学习——支持向量机（Sigmoid 核函数）

测试集评价指标与运行截图：

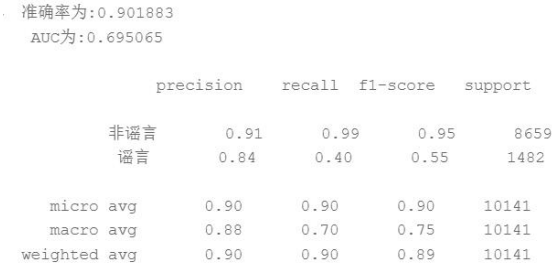
Accuracy ： 0.90
Precision: 非谣言 - 0.91 谣言 - 0.84
Recall ： 非谣言 - 0.99 谣言 - 0.40
F1 ： 非谣言 - 0.95 谣言 - 0.55
AUC ： 0.70



7. 机器学习——支持向量机（线性核函数）

测试集评价指标与运行截图：

Accuracy ： 0.90
Precision: 非谣言 - 0.91 谣言 - 0.84
Recall ： 非谣言 - 0.99 谣言 - 0.40



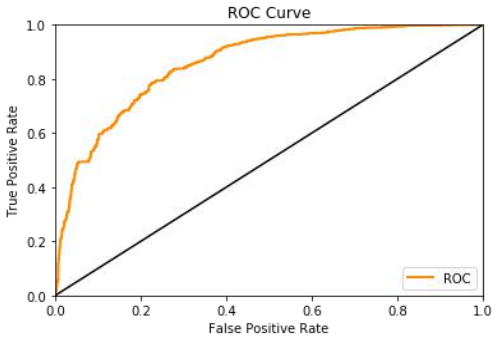
F1　　：　非谣言 - 0.95　谣言 - 0.55
AUC　　：　0.70

8. 深度学习——循环神经网络

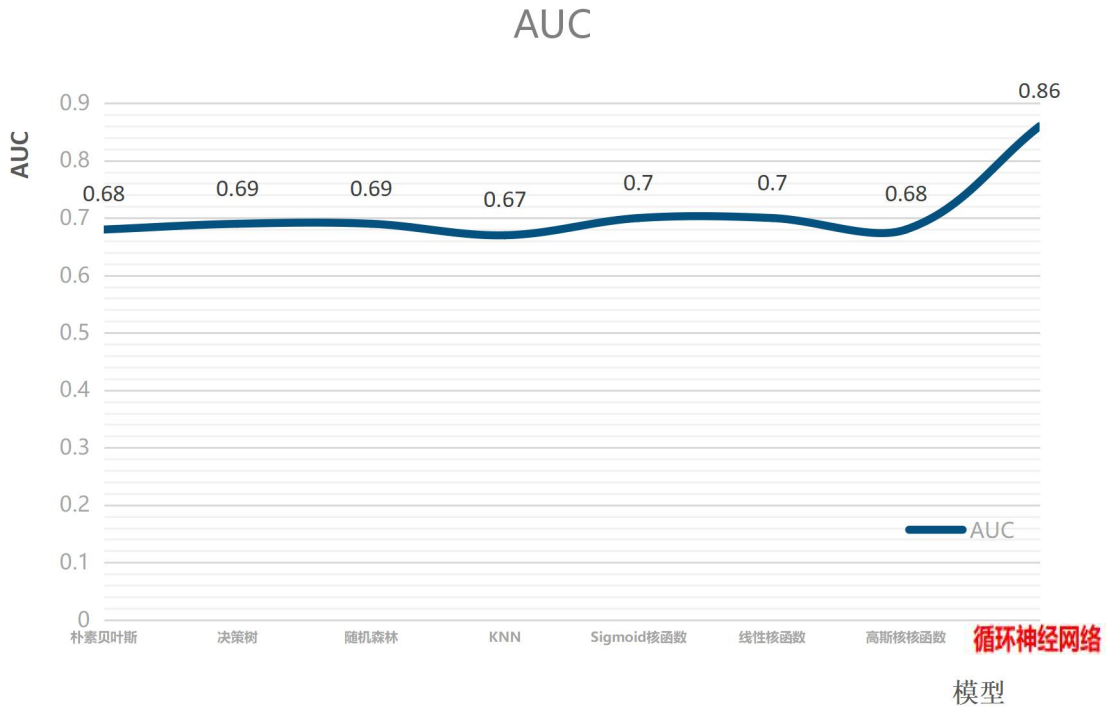
测试集评价指标与运行截图：

Accuracy ：　0.89
Precision：　非谣言 - 0.92　谣言 - 0.67
Recall　：　非谣言 - 0.96　谣言 - 0.48
F1　　：　非谣言 - 0.94　谣言 - 0.56
AUC　　：　0.86

```
Test:, Cost:0.28266, ACC:0.89242
              precision    recall  f1-score   support

        非谣言       0.92      0.96      0.94      8659
         谣言       0.67      0.48      0.56      1482

    accuracy                           0.89     10141
   macro avg       0.79      0.72      0.75     10141
weighted avg       0.88      0.89      0.88     10141

AUC: 0.8593043378921776
```



9. 综合对比



模型		AUC	准确率	精确率		召回率		F1	
				非谣言	谣言	非谣言	谣言	非谣言	谣言
朴素贝叶斯		0.68	0.90	0.90	0.87	0.99	0.38	0.94	0.53
决策树		0.69	0.90	0.90	0.84	0.99	0.39	0.94	0.54
随机森林		0.69	0.90	0.90	0.90	0.99	0.39	0.95	0.54
KNN		0.67	0.90	0.90	0.96	1.00	0.34	0.95	0.51
支持向量机	Sigmoid 核函数	0.70	0.90	0.91	0.84	0.99	0.40	0.95	0.55
	线性核函数	0.70	0.90	0.91	0.84	0.99	0.40	0.95	0.55
	高斯核核函数	0.68	0.90	0.90	0.91	0.99	0.37	0.95	0.53
循环神经网络		0.86	0.89	0.92	0.67	0.96	0.48	0.94	0.56