实验报告

一周乐园队：廖佳怡（SA23006058），陈柯舟（SA23006018），郑善乐（SA23229096）

实验要求

社交媒体舆论场虚假账号检测：http://www.crowdhmt.com/crowdcompetition/settings

任务介绍

主要目标是检测社交媒体中的虚假账号。这些虚假账号会模仿人类行为，利用点赞、评论、转发、发帖等行为与人类进行交互，从而达到以假乱真、引导舆论导向的目的。因此，检测社交媒体中的虚假账号成为整顿互联网秩序，建设良好网络环境的迫切之需。

给定数据集 $D = d_1, d_2, \ldots, d_n$ ，其中d_i表示一条社交媒体账户，n 表示账户总数。该任务要求通过对数据集建模，检测出数据集中存在的虚假账号信息。

数据集介绍

评测数据由2484条账户信息构成。账户信息来自于知名社交媒体平台Twitter。我们通过数据爬虫技术抓去了Twitter上具有影响力的部分账户信息。对于保留的账户数据，我们都进行了人工筛查，以保证数据的质量。

评测方式

对测试集中的账户标签进行预测。将模型预测的账号类型集合 $P_a = a_1^P, \ldots, a_n^P$ 和所标注的账号类型集合 $T_a = a_1^T, \ldots, a_n^T$ 进行对比，计算模型预测F1指标，并作为最终评价指标。

实验步骤

我们总共对三类方案进行尝试，下面将分别介绍这三类方案。

一、方案一（多模态特征+MLP）

由于比赛数据提供了多种模态（布尔特征，数字特征，文本特征，图片特征）的账号特征，此方案将各类特征（布尔特征，数字特征，图片特征）向量化，并通过拼接起来作为该账号的特征，由训练一个MLP来进行二分类。

数据处理

1. 数字特征
   - 数据统计

```
statis={}
for feature in num_feature_list:
    t=np.array(data_cal[feature])
    statis[feature]=(t.min(),t.max())
```

   - 按最大值和最小值归一化

```
each_data["user"][feature]=(each_data["user"][feature]-statis[feature]
[0])/(statis[feature][1]-statis[feature][0])
```

2. 布尔特征

   01整数化

```
each_data["user"][feature]=int(each_data["user"][feature])
```

3. 图片特征
   - 首先根据url从网上下载图片

```
import requests
import json
import os

def get_img(url,path):
    if url:
        ps=url.split(".")[-1]
        ps=ps.lower()
        if ps in ["jpg","png","jpeg","gif"]:
            print(url)
            response = requests.get(url)
            with open(path+"."+ps, 'wb') as fw:
                fw.write(response.content)

def get_img2(url,path):
    if url:
        print(url)
        response = requests.get(url)
        with open(path+".jpg", 'wb') as fw:
            fw.write(response.content)

raw_data_dir = './ref/raw/'
processed_data_dir = './ref/processed/'
for file_name in ['test']:
```

```
    with open(os.path.join(raw_data_dir,file_name+".json"), 'r') as fr:
        data = json.load(fr)
        print(len(data))
        for each_data in data:
            id_str=each_data["user"]["id_str"]
            if "profile_image_url" in each_data["user"]:
                profile_image_url=each_data["user"]["profile_image_url"]

 get_img(profile_image_url,os.path.join(processed_data_dir+"profile_image",id_s
tr))
            if "profile_banner_url" in each_data["user"]:
                profile_banner_url=each_data["user"]["profile_banner_url"]

 get_img2(profile_banner_url,os.path.join(processed_data_dir+"profile_banner",i
d_str))
```

- 然后进行图片读取和预处理

```
self.transform = transforms.Compose([
            transforms.Resize((224, 224)),  # ViT通常需要224x224的输入
            transforms.ToTensor(),  # 将PIL图像转换为张量
            transforms.Normalize(mean=[0.485, 0.456, 0.406], std=[0.229, 0.224,
0.225]),  # 归一化，这些值是预训练模型的训练数据的均值和标准差
        ])
```

- 用ViT(google/vit-base-patch16-224)来encode图片

模型结构

MLP:

```
for i in range(layer_num):
  input_dim=2*hidden_dim+18 if i==0 else hidden_dim
  output_dim=hidden_dim if i!=layer_num-1 else class_num
  mlp.append(nn.Linear(input_dim, output_dim))
  if i!=layer_num-1:
  mlp.append(nn.ReLU())
```

损失函数

```
loss_function = nn.BCELoss(reduction='mean')
```

参数调整

1. 调整输入特征的模态类型（是否包含图片特征）
2. 调整MLP隐层维度:
    - 除最后一层外，其它层维度与输入层保持一致，如各层维度为[18,18,2]
    - 呈逐渐收缩的形态，如各层维度为[18,12,6,2]
3. 调整MLP层数，层数在[2,3,4]中搜索
4. max epochs in [10, 20, 30, 40, 50], lr in [1e-2, 1e-3, 1e-4, 1e-5]

评估指标

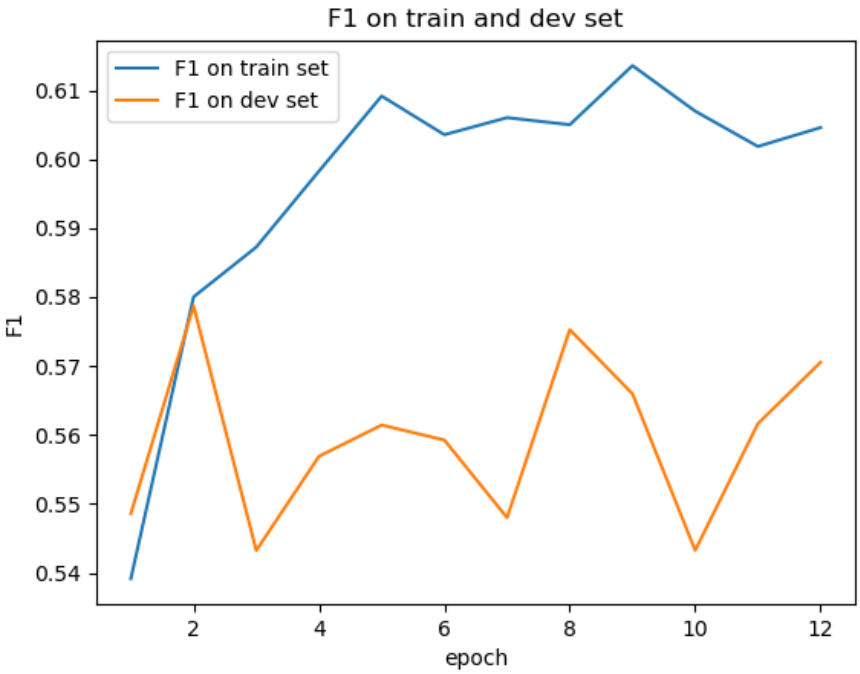
验证集上用F1 score进行模型选择。

二、方案二（文本数据处理，Catboost）

1. BERT处理文本特征（description）的尝试

由于比赛数据中存在不同结构的数据，这其中包含了文本（description），因此首先我们判断文本的语义特征能否在机器人检测中起作用。以用户特征中的description为输入，选取预训练的BERT-base编码，采用BCE loss作为损失函数。

训练过程中发现以依靠文本语义不能时使验证集上的F1上升，如下图所示：



可以看出无法使用文本语义检测账号是否为机器人。原因可能有如下三点：

1. 账号的description是多语言的，其中包含英语，阿拉伯语，韩语等等，BERT作为英文单语言预训练模型无法正确理解不同语言的信息。

2. 很多账号的description不存在，使得训练数据有一定缺失。
3. description的语义本身与账号是否为机器人没有相关性。

因此在后续的模型设计中，我们不再尝试提取description中的语义，而是把账号是否存在description作为输入。

2. Catboost

（Catboost 中的数据处理代码基于郑善乐XGBoost）

模型选择：

使用集成学习算法Catboost分类器。Catboost是以对称决策树（oblivious trees）为基学习器实现的参数较少、支持类别型变量和高准确性的GBDT框架。

```
import catboost as cb
classfier = cb.CatBoostClassifier(**params)
```

特征处理：

数字特征不变，布尔特征用1，0表示，description分别用1，0代表是否有内容。

参数调整：

对catboost 的'learning_rate'（学习率），'max_depth'（最大树深度），'boosting_type'（boost类型）进行调整。

```
params = {
    'loss_function': 'Logloss',
    'learning_rate': 0.01,
    'max_depth': 6,
    'boosting_type':'Ordered',
}
```

评估指标

验证集上用F1 score进行模型选择。

三、方案三（XGBoost）

1. 数据处理

数据读取与特征提取：`extract_features` 函数从每个用户数据中提取了多个特征，包括粉丝数、关注数、列表数、收藏数、状态数等，以及一些布尔特征（例如用户是否被验证、是否使用地理标记等）。这些特征被用于训练分类模型。

1. **特征选择**:
    - 选择了与用户的Twitter账号相关的19个特征。
    - 这些特征包括用户的社交媒体统计数据（如粉丝数、关注数、收藏数等）和账号设置（如是否启用地理定位、是否验证用户等）。
2. **数据类型转换**:

- 对于非数值特征（如布尔类型的特征），代码通过`int()`函数将它们转换为整数（0或1）。

3. **特征组合**:
   - 所有选定的特征被组合成一个特征向量，用于表示每个用户。
4. **标签处理**:
   - 如果数据项中包含“label”字段，该字段也被提取出来。
   - 标签用于监督学习，帮助模型学习区分不同类别（在本案例中是bot和human）。

2. 模型结构

XGBoost模型：选用了XGBoost作为分类器，这是一个高效的梯度增强库。XGBoost对于不平衡数据集表现良好，且对于各种特征类型（如数值型和类别型）处理效果都不错。

1. **基于决策树的集成方法**:
   - XGBoost是一个集成学习方法，它使用多个决策树来进行预测。
   - 每个新的树都是在前一个树的基础上构建的，以此来改正前者的错误。
2. **梯度提升**:
   - XGBoost通过梯度提升框架进行工作，即逐步添加新模型，每次添加都是为了减少整体模型的损失函数。
   - 在每一步，根据之前所有树的总和预测的残差，构建新的树。
3. **模型参数**:
   - `max_depth`控制决策树的最大深度，较小的值可以防止过拟合。
   - `min_child_weight`定义了子节点所需的最小权重，这同样有助于防止过拟合。
   - `gamma`也称为最小切分损失，用于控制树的进一步增长。
   - `subsample`和`colsample_bytree`分别控制用于构建树的样本和特征的比例，有助于增加模型的泛化能力。
4. **目标函数和评估指标**:
   - `objective`设置为`binary:logistic`，表明这是一个二分类问题，输出的是概率。
   - `eval_metric`设为`logloss`，也就是逻辑损失，用于评估模型的预测质量。
5. **训练和验证**:
   - 使用了提早停止（early stopping）来防止过拟合。如果在一定轮数内验证集上的性能没有提高，训练将停止。
   - 训练中使用了训练集和验证集来监控模型的性能。

3. 损失函数

二元逻辑回归损失（binary:logistic）：选择了逻辑回归损失作为优化目标，这适用于二分类问题。该损失函数计算的是预测概率和实际标签之间的差异。

4. 参数调整

```
params = {
    'objective': 'binary:logistic',
    'eval_metric': 'logloss',
    'eta': 0.1,
    'max_depth': 4,  # 减少树的深度
    'min_child_weight': 6,  # 增加孩子节点中所需的最小权重
    'gamma': 0.5,  # 增加gamma值
    'subsample': 0.8,  # 使用80%的样本来训练每棵树
    'colsample_bytree': 0.8  # 使用80%的特征来训练每棵树
}
```

- `params` 字典中定义了多个超参数，如 `max_depth`（树的最大深度）、`min_child_weight`、`gamma`、`subsample` 和 `colsample_bytree`。这些参数对模型的性能和过拟合有重要影响。
- `max_depth` 被设定为较小的值4，这有助于防止模型过于复杂，从而避免过拟合。
- `min_child_weight`: 设置为6，这个参数用于控制决策树的生长。较大的值可以防止过拟合。
- `gamma`: 设置为0.5，这是树的进一步分裂所需的最小损失减少。较大的值可以使模型更加保守。
- `subsample` 和 `colsample_bytree` 的设置为 0.8，意味着在构建每棵树时，只随机选择 80% 的样本和特征，这有助于增加模型的随机性，减少过拟合风险。
- 使用了早停（early stopping）来避免过度训练。如果在连续 10 轮迭代中验证集的性能没有改善，训练将提前停止。
- 最后在验证集上使用Weighted Average方法计算F1 Score

实验结果

验证集上得分：

- 方案一：0.76
- 方案二：0.772
- 方案三：0.75

测试集上得分：0.816291869906718

团队排名：并列第10

代码整理

code文件夹下

实验收获

在这次社交媒体舆论场虚假账号检测比赛中，我们深入理解了虚假账号在社交媒体中的行为模式和特征，同时也了解了虚假账号对社交媒体环境的潜在影响。我们使用了各种机器学习和深度学习模型，通过训练和调整参数，尝试找到最佳的模型来解决这个问题。以下是我们的主要收获：

1. 数据预处理的重要性：我们发现原始数据中存在一些噪声和异常值，这可能会影响模型的学习效果。因此，我们进行了数据清洗和预处理，包括去除异常值、填充缺失值、对特征进行标准化等。这些预处理步骤显著提高了数据的质量，使模型能更好地学习和理解数据。
2. 特征选择的关键性：我们发现不是所有的特征都对虚假账号的识别有帮助。因此，我们进行了特征选择，只保留了那些对虚假账号识别有显著影响的特征。这降低了模型的复杂性，同时也提高了模型的预测精度。
3. 模型选择和调参的技巧：在实验过程中，我们尝试了多种机器学习和深度学习模型，包括各种机器学习和深度学习模型。我们发现，不同的模型有各自的优点和缺点，需要根据实际问题和数据特性来选择合适的模型。此外，模型的参数设置也对模型的性能有显著影响，因此我们花费了大量时间进行模型调参。
4. 评价指标的理解：我们学习了如何使用F1指标来评价模型的性能。F1指标是精确度和召回率的调和平均值，能同时考虑到模型的查准率和查全率，是一个非常实用的评价指标。
5. 团队合作的力量：在这次实验中，我们团队成员之间进行了紧密的合作和交流，共同解决了许多问题，也从中学习到了很多新的知识和技能。这让我们深刻认识到团队合作的重要性。

总的来说，这次比赛不仅让我们学习到了很多关于虚假账号检测的知识和技能，也提高了我们的团队合作能力和解决实际问题的能力。我们相信，这次比赛的经验将对我们今后的研究有很大的帮助。

团队分工

- 方案一：廖佳怡
- 方案二：陈柯舟
- 方案三：郑善乐