基于机器学习的2021欧洲杯预测

一、实验目的

1、运用神经网络预测比赛结果

2、改进模型，提升正确率

二、实验原理

1、本实验历史数据来源于kaggle，本届数据自己整理的。两者作用不同，前者更能表征一支球队的数十年的综合实力，代表了球队底蕴；后者来源于小组赛，因此更能表征当前的实力状态，对预测的作用更大。

2、本实验使用ANN神经网络，考虑到性能和适用性，直接使用sklearn调包实现。不同类型的数据输入到神经网络中，他们和不同权值相乘，并通过激活函数输出。神经网络使用BP算法更新权值，这个过程也称为学习。

三、实验内容

①基于历史数据

最初使用的数据集是一个来自kaggle的欧洲杯历史数据：UEFA Euro Cup (1960-2016)，它包含几十年来每届欧洲杯的比赛结果、各队胜利次数等数据。

1、首先是对大量数据进行整理。数据集中缺少比赛结果一项，可通过进球数来添加。读入All Matches文件，选择主/客场球队，主/客场进球四个数据构成比赛比分match_scores数据集，再使用pandas的loc方法加上一列数据result。

```
match_statistics = pd.read_csv('All Matches.csv')
match_scores=match_statistics[['HomeTeamName','AwayTeamName','HomeTeamGoals',
'AwayTeamGoals']]
match_scores.loc[match_scores['HomeTeamGoals']>match_scores['AwayTeamGoals'],
'results']= 'win'
match_scores.loc[match_scores['HomeTeamGoals']<match_scores['AwayTeamGoals'],
'results']= 'lose'
match_scores.loc[match_scores['HomeTeamGoals']==match_scores['AwayTeamGoals']
, 'results']= 'draw'
match_scores.head()#查看前5个表项
```

	HomeTeamName	AwayTeamName	HomeTeamGoals	AwayTeamGoals	results
0	France	Yugoslavia	4	5	lose
1	Czechoslovakia	Soviet Union	0	3	lose
2	Czechoslovakia	France	2	0	win
3	Soviet Union	Yugoslavia	2	1	win
4	Spain	Hungary	2	1	win

2、只有比赛结果还不够，胜利场次、进球数、积分都是评价一支球队很重要的数据。再读入另外两个数据集，收集有用的数据，剔除球场、天气等无用数据。使用pandas的concat拼接得到球队状态team_status数据集。

```
general_statistics=pd.read_csv('Participated Teams General
Statistics.csv',index_col=0)
national_appearance=pd.read_csv('National Teams Appearance.csv',index_col=0)
national_appearance=national_appearance[['Appearances','Record
streak','Active streak']]
team_status=pd.concat([general_statistics,national_appearance],axis=1)
team_status=team_status[:-2]
team_status.head()
```

	Participations	Played	Win	Draw	Loss	Goal_For	Goal_Against	Goal_Difference	Points	Points/match	Appearances	Record streak	Active streak
Germany	12.0	49.0	26.0	12.0	11.0	72.0	48.0	24.0	90.0	1.84	13	13	13
France	9.0	39.0	20.0	9.0	10.0	62.0	44.0	18.0	69.0	1.77	10	8	8
Spain	10.0	40.0	19.0	11.0	10.0	55.0	36.0	19.0	68.0	1.70	11	7	7
Italy	9.0	38.0	16.0	16.0	6.0	39.0	27.0	12.0	64.0	1.68	10	7	7
Portugal	7.0	35.0	18.0	9.0	8.0	49.0	31.0	18.0	63.0	1.80	8	7	7

3、现在有了各球队的历史交战记录和各球队的比赛数据，我们要让模型学习二者之间的关系。使用reindex把比赛数据拼接到主客队后面，使对阵双方的比赛数据在写同一行，形成对比。

```
home_team_df=team_status.reindex(match_scores['HomeTeamName'])
away_team_df=team_status.reindex(match_scores['AwayTeamName'])
home_away_df=pd.concat([home_team_df.reset_index(),away_team_df.reset_index()],axis=1)
home_away_df=home_away_df.dropna()#去除重复项
home_away_df=home_away_df.reset_index(drop=True)
home_away_df.head()
```

	HomeTeamName	Participations	Played	Win	Draw	Loss	Goal_For	Goal_Against	Goal_Difference	Points	...	Draw	Loss	Goal_For	Goal_Against	Goal_
0	Spain	10.0	40.0	19.0	11.0	10.0	55.0	36.0	19.0	68.0	...	2.0	4.0	11.0	14.0	
1	Hungary	3.0	8.0	2.0	2.0	4.0	11.0	14.0	-3.0	8.0	...	6.0	14.0	30.0	43.0	
2	Hungary	3.0	8.0	2.0	2.0	4.0	11.0	14.0	-3.0	8.0	...	2.0	8.0	22.0	25.0	
3	Netherlands	9.0	35.0	17.0	8.0	10.0	57.0	37.0	20.0	59.0	...	3.0	8.0	14.0	20.0	
4	Belgium	5.0	17.0	7.0	2.0	8.0	22.0	25.0	-3.0	23.0	...	11.0	10.0	40.0	35.0	

4、加上比赛结果，去除无法量化的'HomeTeamName'和'AwayTeamName'，这样就得到了所需要的数据集。它由对阵双方的比赛数据和结果组成，模型可以通过学习知道各个数据对结果的影响。

```
euro_cup_data=pd.concat([home_away_df,match_scores.iloc[:,-1]],axis=1).drop(['HomeTeamName','AwayTeamName'],axis=1)
euro_cup_data=euro_cup_data.dropna()
euro_cup_data.head()
```

	Participations	Played	Win	Draw	Loss	Goal_For	Goal_Against	Goal_Difference	Points	Points/match	...	Loss	Goal_For	Goal_Against	Goal_Difference
0	10.0	40.0	19.0	11.0	10.0	55.0	36.0	19.0	68.0	1.70	...	4.0	11.0	14.0	-3.0
1	3.0	8.0	2.0	2.0	4.0	11.0	14.0	-3.0	8.0	1.00	...	14.0	30.0	43.0	-13.0
2	3.0	8.0	2.0	2.0	4.0	11.0	14.0	-3.0	8.0	1.00	...	8.0	22.0	25.0	-3.0
3	9.0	35.0	17.0	8.0	10.0	57.0	37.0	20.0	59.0	1.69	...	8.0	14.0	20.0	-6.0
4	5.0	17.0	7.0	2.0	8.0	22.0	25.0	-3.0	23.0	1.35	...	10.0	40.0	35.0	5.0

5、这里需要考虑的一个问题是不同的数据大小不一致，如Participations一般不会多于10次，但Goal_For却是比较大的。数据的量纲不同可能会对结果造成影响，因此需要先对数据归一化处理，本实验使用的是Min-Max归一化，公式如下：

$$x^* = \frac{x - \min}{\max - \min}$$

```
scores_temp=euro_cup_data.iloc[:,:-1]
euro_cup_normal=(scores_temp - scores_temp.min()) / (scores_temp.max() - scores_temp.min())
euro_cup_normal.head()
```

	Participations	Played	Win	Draw	Loss	Goal_For	Goal_Against	Goal_Difference	Points	Points/match	...	Draw	Loss	Goal_For	Goal_
0	0.818182	0.804348	0.730769	0.6875	0.692308	0.760563	0.733333	0.864865	0.752809	0.820359	...	0.1250	0.230769	0.140845	0
1	0.181818	0.108696	0.076923	0.1250	0.230769	0.140845	0.244444	0.270270	0.078652	0.401198	...	0.3750	1.000000	0.408451	0
2	0.181818	0.108696	0.076923	0.1250	0.230769	0.140845	0.244444	0.270270	0.078652	0.401198	...	0.1250	0.538462	0.295775	0
3	0.727273	0.695652	0.653846	0.5000	0.692308	0.788732	0.755556	0.891892	0.651685	0.814371	...	0.1875	0.538462	0.183099	0
4	0.363636	0.304348	0.269231	0.1250	0.538462	0.295775	0.488889	0.270270	0.247191	0.610778	...	0.6875	0.692308	0.549296	0

6、数据集构建完成，接下来可以进行预测了。x是归一化后的比赛数据，y是比赛结果。确定要比较的两支队伍，在数据集中查找他们的数据并归一化。

```
x=euro_cup_normal
y=euro_cup_data['results']
team1='England'
team2='Italy'
predict=pd.concat([home_team_df.loc[team1].iloc[0],home_team_df.loc[team2].iloc[0]])
predict_normal=(predict - predict.min()) / (predict.max() - predict.min())
```

7、调用多层感知机分类器MLPClassifier，使用predict_proba计算每种结果的概率。另外，使用5折交叉验证来计算正确率。

```
model_1=MLPClassifier(max_iter=3000)
model_1.fit(x,y)
prob_model_1=model_1.predict_proba(np.atleast_2d(predict_normal))
#print('比赛结果：%s' % (model_1.predict(np.atleast_2d(predict))[0]))#模型预测
print('赢球概率：%.3f' % (prob_model_1[0][0]))
print('平局概率：%.3f' % (prob_model_1[0][1]))
print('输球概率：%.3f' % (prob_model_1[0][2]))
print('交叉验证正确率：%.3f' % (np.mean(cross_val_score(model_1,x,y,cv=5))))#5折交叉验证
```

```
赢球概率：0.434
平局概率：0.350
输球概率：0.216
交叉验证正确率：0.389
```

正确率只有38.9%，并不理想。

分析原因：历史数据是从1960到2016，时间跨度太大了，每支球队都有巅峰和低谷，用60年前的数据来预测今天的比赛结果是不合理的。因此可以考虑使用本届欧洲杯小组赛的数据，这更能看出目前的实力和状态。

②基于本届数据

第二个数据集来自本届小组赛，包括每场比赛的对阵结果group_matchs和比赛数据team_status，和之前历史数据很类似。

1、处理数据的方法和之前相同，最终得到的数据集如下

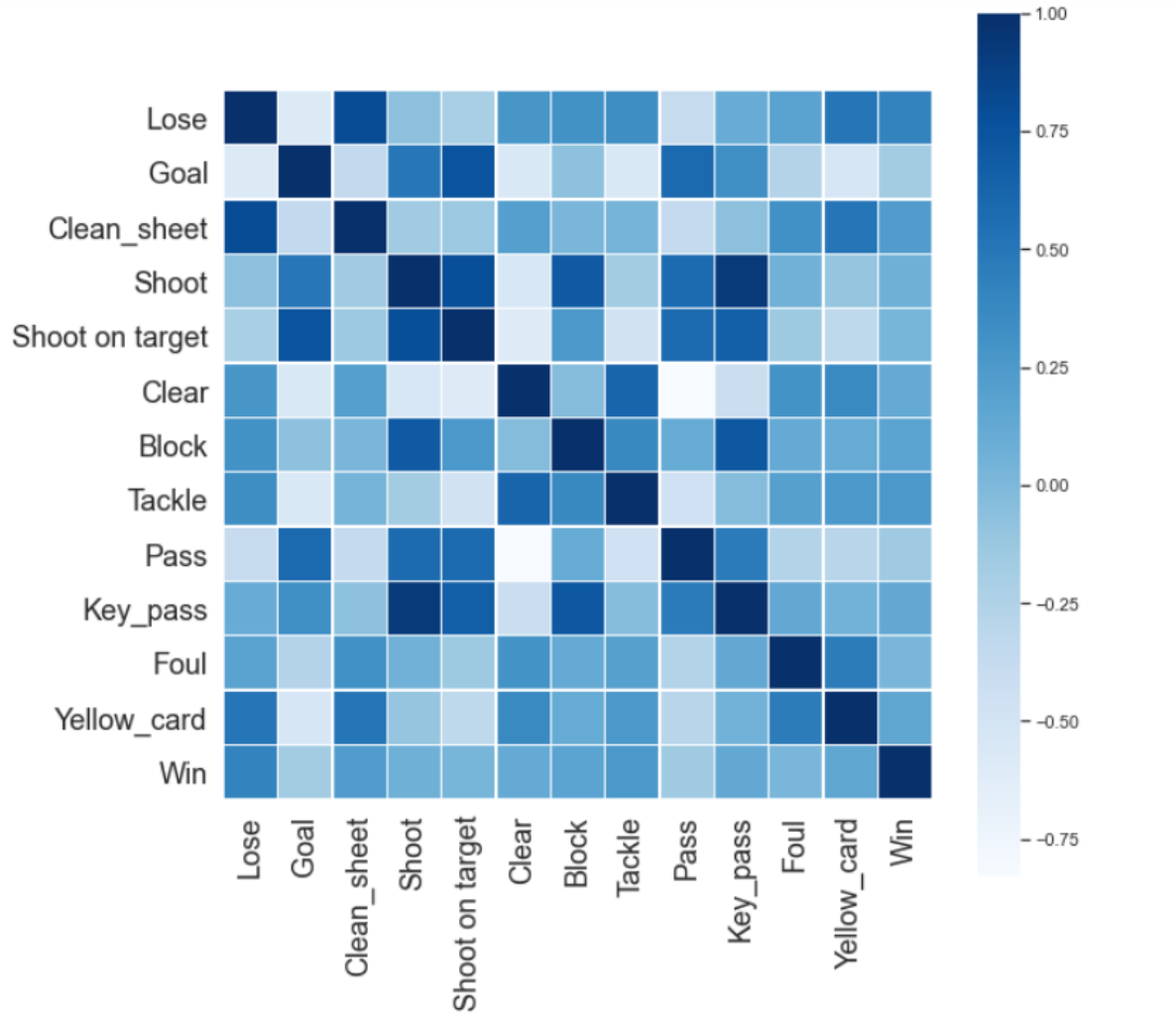
```
#min-max映射归一化数据，消除大数据对小数据的影响
scores_temp=euro_cup_data.iloc[:,:-1]
euro_cup_normal=(scores_temp - scores_temp.min()) / (scores_temp.max() - scores_temp.min())
euro_cup_normal.head()
```

	Win	Lose	Goal	Clean_sheet	Shoot	Shoot on target	Clear	Block	Tackle	Pass	...	Clean_sheet	Shoot	Shoot on target	Clear	Block
0	0.000000	1.000000	0.000000	1.000	0.456522	0.5	0.552941	0.588235	0.571429	0.347856	...	0.000	1.000000	0.75	0.164706	0.882353
1	0.333333	0.333333	0.285714	0.250	0.239130	0.3	0.635294	0.470588	0.357143	0.272476	...	0.625	0.608696	0.65	0.317647	0.764706
2	0.333333	0.666667	0.571429	0.500	0.804348	1.0	0.211765	0.823529	0.357143	0.478562	...	0.375	0.000000	0.05	0.917647	0.529412
3	1.000000	0.000000	0.857143	0.125	0.282609	0.7	0.270588	0.176471	0.190476	0.477870	...	0.875	0.130435	0.20	0.470588	0.176471
4	0.666667	0.000000	0.142857	0.000	0.065217	0.2	0.117647	0.117647	0.380952	0.480636	...	0.375	0.239130	0.40	0.341176	0.235294

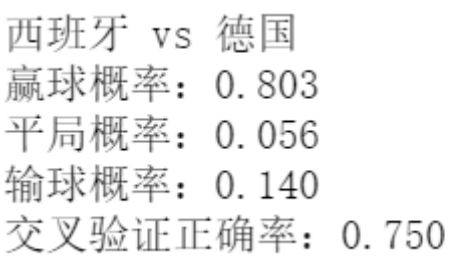
5 rows × 26 columns

2、画热力图是数据可视化的常见方法，通过热力图能直观地看出各种数据的相关程度。可以看出除了Key_pass和Shoot相关程度较高，其他数据都能较好地反映出结果

```
#绘制热力图，查看各特征相关性
train_matrix = euro_cup_normal.iloc[:,1:14].corr()
sns.set()
f, ax = plt.subplots(figsize=(10,10))
sns.heatmap(train_matrix, annot=False, square=True, cmap="Blues",
linewidths=.5, vmax=1, ax=ax)
plt.xticks(fontsize=18)
plt.yticks(fontsize=18)
plt.show()
```



3、再次运行神经网络，正确率达到了75%，比较理想



四、实验结果

1、使用正确率更高的小组赛数据集来训练模型，并预测淘汰赛结果（节选）

2、最终预测结果如下，**西班牙**将获得2021欧洲杯冠军

五、实验心得

第一次完整地完成了一个机器学习项目，感想是大部分时间都用在前期处理数据上了，尤其是对pandas不熟悉，很多步骤都是试出来的。同时也能看出数据集对一个预测模型成功率的影响巨大，历史数据只能达到30%多的正确率，而换成小组赛数据则高达75%。

本实验的预测完成于6月底，小组赛刚进行完，预测最终西班牙击败德国夺冠。本实验报告写于7月10号，决赛将于后天进行，由英格兰对阵意大利。具体晋级图如下所示，和上面的预测结果相比差别还是挺大的。所以……即便该模型有75%的正确率，但预测结果还是不尽人意。

归结原因主要有2个。一是模型原因，数据只有3场小组赛显然还是太少了，而且每个小组强弱不等，小组赛数据不足以完全表现出各个球队的实力。另外预测模型也是直接调用感知机实现的，未进行过多的调参，可能模型的效果还未达到最佳。

二是现实原因，足球比赛变数太大了，很多东西如球员状态、临场战术无法量化，这就导致了结果的偏差。例如，赛前谁也想不到世界杯冠军法国会输给倒在第一轮淘汰赛，因为队内球员之间的矛盾和教练突然改踢三中卫是模型无法预测的。这也是足球的魅力吧。

结论是：赌球有风险，买球需谨慎。

参考

https://www.kaggle.com/jaykumar1607/uefa-euro-cup-19602016

https://github.com/itsmuriuki/FIFA-2018-World-cup-predictions

https://www.bilibili.com/video/BV1Ts411E7kP