

## 数据:

来自于kaggle

- Warding totem: An item that a player can put on the map to reveal the nearby area. Very useful for map/objectives control.
- Minions: NPC that belong to both teams. They give gold when killed by players.
- Jungle minions: NPC that belong to NO TEAM. They give gold and buffs when killed by players.
- Elite monsters: Monsters with high hp/damage that give a massive bonus (gold/XP/stats) when killed by a team.
- Dragons: Elite monster which gives team bonus when killed. The 4th dragon killed by a team gives a massive stats bonus. The 5th dragon (Elder Dragon) offers a huge advantage to the team.
- Herald: Elite monster which gives stats bonus when killed by the player. It helps to push a lane and destroys structures.
- Towers: Structures you have to destroy to reach the enemy Nexus. They give gold.
- Level: Champion level. Start at 1. Max is 18.

```
<bound method NDFrame.head of                                     gameId  blueWins  ...  redCSPerMin  redGoldPerMin
0      4519157822          0  ...          19.7          1656.7
1      4523371949          0  ...          24.0          1762.0
2      4521474530          0  ...          20.3          1728.5
3      4524384067          0  ...          23.5          1647.8
4      4436033771          0  ...          22.5          1740.4
...          ...          ...  ...          ...          ...
9874   4527873286          1  ...          22.9          1524.6
9875   4527797466          1  ...          20.6          1545.6
9876   4527713716          0  ...          26.1          1831.9
9877   4527628313          0  ...          24.7          1529.8
9878   4523772935          1  ...          20.1          1533.9

[9879 rows x 40 columns]>

Process finished with exit code 0
```

## 1.了解数据的大致信息

```
data.shape
```

```
(9879, 40)
```

```
data.describe()
```

```
          gameId  blueWins  ...  redCSPerMin  redGoldPerMin
count  9.879000e+03  9879.000000  ...  9879.000000  9879.000000
mean    4.500084e+09    0.499038  ...    21.734923    1648.904140
std     2.757328e+07    0.500024  ...     2.191167    149.088841
min     4.295358e+09    0.000000  ...    10.700000    1121.200000
25%     4.483301e+09    0.000000  ...    20.300000    1542.750000
50%     4.510920e+09    0.000000  ...    21.800000    1637.800000
75%     4.521733e+09    1.000000  ...    23.300000    1741.850000
max     4.527991e+09    1.000000  ...    28.900000    2273.200000

[8 rows x 40 columns]
```

```
data.head
```

```
<bound method NDFrame.head of
0    4519157822    0 ...    19.7    1656.7
1    4523371949    0 ...    24.0    1762.0
2    4521474530    0 ...    20.3    1728.5
3    4524384067    0 ...    23.5    1647.8
4    4436033771    0 ...    22.5    1740.4
...      ...      ... ...      ...      ...
9874  4527873286    1 ...    22.9    1524.6
9875  4527797466    1 ...    20.6    1545.6
9876  4527713716    0 ...    26.1    1831.9
9877  4527628313    0 ...    24.7    1529.8
9878  4523772935    1 ...    20.1    1533.9
```

```
[9879 rows x 40 columns]>
```

```
Process finished with exit code 0
```

## 2.预处理

```
labels=np.array(data["winner"])
features=data.drop("winner",axis=1)
feature_list=list(features.columns)
features=np.array(features)
```

## 3.查看特征之间的相关程度

绘制热力图

```
corr = df.corr()
sns.heatmap(corr,
             xticklabels=corr.columns.values,
             yticklabels=corr.columns.values)
```

进行 T-Test...

## 4.训练模型前需要先对数据集进行划分

```
# 产生x, y, 即特征值与目标值
target_name = 'winner'
x = df.drop('winner', axis=1)
y = df[target_name]

# 将数据分为训练和测试数据集
# 注意参数 stratify = y 意味着在产生训练和测试数据中, 离职的员工的百分比等于原来总的的数据中的
# 离职的员工的百分比
X_train, X_test, y_train, y_test = train_test_split(
    x, y, test_size=0.15, random_state=123, stratify=y)
# 显示前5行数据
df.head()
```

## 5.开始训练

```
# 实例化
dtree = tree.DecisionTreeClassifier(
    criterion='entropy',
    #max_depth=3, # 定义树的深度，可以用来防止过拟合
    min_weight_fraction_leaf=0.01 # 定义叶子节点最少需要包含多少个样本(使用百分比表达)，
    防止过拟合
)
# 训练
dtree = dtree.fit(X_train,y_train)
# 指标计算
dt_roc_auc = roc_auc_score(y_test, dtree.predict(X_test))
print ("决策树 AUC = %2.2f" % dt_roc_auc)
print(classification_report(y_test, dtree.predict(X_test)))
```

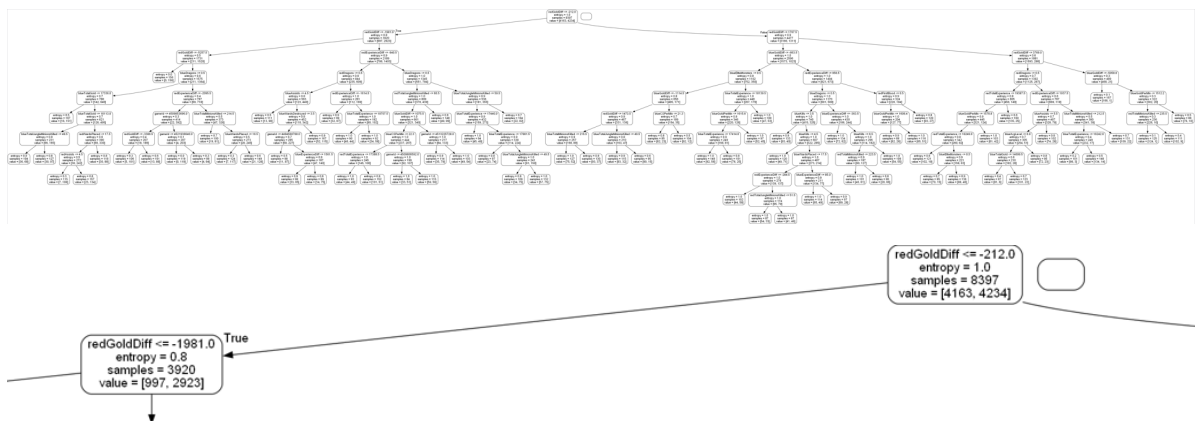
决策树 AUC = 0.72

	precision	recall	f1-score	support
0	0.75	0.70	0.72	786
1	0.69	0.74	0.71	696
accuracy			0.72	1482
macro avg	0.72	0.72	0.72	1482
weighted avg	0.72	0.72	0.72	1482

## 6.树模型的可视化

```
from sklearn.tree import export_graphviz
import pydot
```

```
export_graphviz(dtree,out_file="tree.dot",feature_names=feature_list,rounded=True,
precision=1)
(graph,)= pydot.graph_from_dot_file("tree.dot")
graph.write_png("tree.png")
```





可以看到有多个分支

## 7.最后看看预测的效果

```

true_data=pd.DataFrame(data={"id":[i for i in
range(100)],"actual":y_test[:100]})
plt.plot(true_data["id"],y_test[:100],"bp",label="actual")
plt.plot(true_data["id"],dtree.predict(X_test)[:100],"rp",label="prediction")

plt.legend()
plt.xlabel("id")
plt.ylabel("bluewin")
plt.title("Actual and Predicted Values")
plt.show()

```

## 8.用户输入

```

lis = ['blueFirstBlood'
        , 'blueKills'
        , 'blueDeaths'
        , 'blueAssists'
        , 'blueEliteMonsters'
        , 'blueDragons'
        , 'blueHeralds'
        , 'blueTowersDestroyed'
        , 'blueTotalGold'
        , 'blueTotalMinionsKilled'
        , 'redAssists'
        , 'redEliteMonsters'
        , 'redDragons'
        , 'redHeralds'
        , 'redTowersDestroyed'
        , 'redTotalGold'
        , 'redTotalMinionsKilled']

lol_data = []
for i in range(17):
    print("Please Input " + lis[i] + " : ", end='')
    s = int(input())
    lol_data.append([s])

lol_data = np.array(lol_data)
header = np.array(lis)
with open('lol.csv', 'w', newline='') as f:
    f_csv = csv.writer(f)

```

```

f_csv.writerow(header)
f_csv.writerows(lol_data.T)

want_data = pd.read_csv(r'lol.csv')

print()
if dtree.predict(want_data) == 1:
    print("BlueTeamWin, Probability: " + str(dtree.predict_proba(want_data)[0][1]))
else:
    print("BlueTeamLose, Probability: " + str(dtree.predict_proba(want_data)[0][0]))

import os

os.remove('lol.csv')

```



这是我们截取到的比赛第10min的画面，通过反复观看录像用人眼获取数据并输入：

```
Please Iuput blueFirstBlood : 1
Please Iuput blueKills : 1
Please Iuput blueDeaths : 2
Please Iuput blueAssists : 2
Please Iuput blueEliteMonsters : 2
Please Iuput blueDragons : 2
Please Iuput blueHeralds : 1
Please Iuput blueTowersDestroyed : 8
Please Iuput blueTotalGold : 15000
Please Iuput blueTotalMinionsKilled : 303
Please Iuput redAssists : 2
Please Iuput redEliteMonsters : 0
Please Iuput redDragons : 4
Please Iuput redHeralds : 1
Please Iuput redTowersDestroyed : 6
Please Iuput redTotalGold : 15200
Please Iuput redTotalMinionsKilled : 289
```

BlueTeamWin, Probability: 0.6215686274509804

最终得到T1决赛胜利的概率为百分之62(虽然最终还是输了)

## 不懂

---

roc\_auc\_score()

```
dt_roc_auc = roc_auc_score(y_test, dtree.predict(X_test))
```