导入依赖包

```
In  [1]: import os
         import pandas as pd
         import numpy as np

         from collections import defaultdict

         from sklearn.tree import DecisionTreeClassifier
         from sklearn.cross_validation import cross_val_score
         from sklearn.preprocessing import LabelEncoder
         from sklearn.preprocessing import OneHotEncoder
```

```
f:\software\python35\lib\site-packages\sklearn\cross_validation.py:41: DeprecationWarning: This module was deprecated in version 0.
18 in favor of the model_selection module into which all the refactored classes and functions are moved. Also note that the interfa
ce of the new CV iterators are different from that of this module. This module will be removed in 0.20.
  "This module will be removed in 0.20.", DeprecationWarning)
```

加载数据集

- 修改头部属性名称
- 将日期字符串格式转为日期对象格式

* 注意：在Windows下，读取csv文件要重新编码

```
In  [10]: data_folder = "D:\dataset\Data"
          data_filename = os.path.join(data_folder, "basketball", "leagues_NBA_2014_games_games.csv")
          dataset = pd.read_csv(data_filename, encoding='gbk')

          dataset.columns = ["Date", "Start Time", "Visitor Team","VisitorPts", "Home Team", "HomePts", "OT?", "Notes"]
          dataset['Date'] = pd.to_datetime(dataset['Date'])
```

提取新特征（1）

- 找出主场获胜的球队

In [11]:
```
dataset["HomeWin"] = dataset["VisitorPts"] < dataset["HomePts"]
y_true = dataset["HomeWin"].values
won_last = defaultdict(int)
# 初始化
dataset["HomeLastWin"] = False
dataset["VisitorLastWin"] = True
for index, row in dataset.sort_values("Date").iterrows():
    home_team = row["Home Team"]
    visitor_team = row["Visitor Team"]
    row["HomeLastWin"] = won_last[home_team]
    row["VisitorLastWin"] = won_last[visitor_team]
    dataset.iloc[index] = row
    won_last[home_team] = row["HomeWin"]
    won_last[visitor_team] = not row["HomeWin"]
```

使用分类器

In [12]:
```
clf = DecisionTreeClassifier(random_state=14)

X_previouswins = dataset[["HomeLastWin", "VisitorLastWin"]].values

scores = cross_val_score(clf, X_previouswins, y_true, scoring='accuracy')
print("Accuracy: {0:.1f}%".format(np.mean(scores) * 100))
```

```
Accuracy: 57.6%
```

添加新特征（2）

创建一个叫作“主场队是否通常比对手水平高”的特征，并使用2013赛季的战绩作为特征取值来源。

如果一支球队在2013赛季排名在对手前面，我们就认为它的水平更高。

In [15]:
```
## 读取13赛季数据
standings_filename = os.path.join(data_folder, "basketball", "leagues_NBA_2013_standings_expanded-standings.csv")
# 同样的，注意修改编码
standings = pd.read_csv(standings_filename, skiprows=[0], encoding='gbk')

dataset["HomeTeamRanksHigher"] = 0
for index, row in dataset.iterrows():
    home_team = row["Home Team"]
    visitor_team = row["Visitor Team"]

    if home_team == "New Orleans Pelicans":
        home_team = "New Orleans Hornets"
    elif visitor_team == "New Orleans Pelicans":
        visitor_team = "New Orleans Hornets"

    home_rank = standings[standings["Team"] == home_team]["Rk"].values[0]
    visitor_rank = standings[standings["Team"] == visitor_team]["Rk"].values[0]
    row["HomeTeamRanksHigher"] = int(home_rank > visitor_rank)
    dataset.iloc[index] = row
```

再次使用分类器

In [16]:
```
clf = DecisionTreeClassifier(random_state=14)

X_homehigher = dataset[["HomeLastWin", "VisitorLastWin","HomeTeamRanksHigher"]].values

scores = cross_val_score(clf, X_homehigher, y_true, scoring='accuracy')
print("Accuracy: {0:.1f}%".format(np.mean(scores) * 100))
```

```
Accuracy: 60.8%
```

添加新特征（3）

统计两支球队上场比赛的情况，作为另一个特征。

虽然球队排名有助于预测（排名靠前的胜算更大），但有时排名靠后的球队反而能战胜排名靠前的。

In [17]:
```
last_match_winner = defaultdict(int)
dataset["HomeTeamWonLast"] = 0

for index, row in dataset.iterrows():
    home_team = row["Home Team"]
    visitor_team = row["Visitor Team"]

    teams = tuple(sorted([home_team, visitor_team]))

    row["HomeTeamWonLast"] = 1 if last_match_winner[teams] == row["Home Team"] else 0
    dataset.ix[index] = row

    winner = row["Home Team"] if row["HomeWin"] else row ["Visitor Team"]
    last_match_winner[teams] = winner
```

再再次使用分类器

In [20]:
```
X_lastwinner = dataset[["HomeTeamRanksHigher", "HomeTeamWonLast"]].values

clf = DecisionTreeClassifier(random_state=14)

scores = cross_val_score(clf, X_lastwinner, y_true, scoring='accuracy')
print("Accuracy: {0:.1f}%".format(np.mean(scores) * 100))
```

```
Accuracy: 58.8%
```

检验分类效果

决策树在训练数据量很大的情况下，能否得到有效的分类模型。

我们将会为决策树添加球队，以检测它是否能整合新增的信息。

- 用 LabelEncoder转换器 把字符串类型的球队名转化为整型
- 用 OneHotEncoder转换器 把整数转换为二进制数字

```
In [26]: encoding = LabelEncoder()
         encoding.fit(dataset["Home Team"].values)

         home_teams = encoding.transform(dataset["Home Team"].values)
         visitor_teams = encoding.transform(dataset["Visitor Team"].values)
         X_teams = np.vstack([home_teams, visitor_teams]).T
```

```
In [27]: onehot = OneHotEncoder()
         X_teams_expanded = onehot.fit_transform(X_teams).todense()

         clf = DecisionTreeClassifier(random_state=14)

         scores = cross_val_score(clf, X_teams_expanded, y_true, scoring='accuracy')

         print("Accuracy: {0:.1f}%".format(np.mean(scores) * 100))
```

```
Accuracy: 59.0%
```

结果分析

正确率接近60%，比基准值要高，但是没有之前的效果好。

原因可能在于特征数增加后，决策树处理不当。

```
In [ ]:
```