NFL Play by Play 2009-2017 (v4)数据分析与预处理

1 数据读取和属性分类

利用 Python 中的 pandas 库进行 csv 数据文件的读取：

```python
def read_csv(path, na_values=None):
    '读取csv数据集'
    return pd.read_csv(path, na_values=na_values, low_memory=False)

csv_path='./NFL Play by Play 2009-2017 (v4).csv'
dataFrame=read_csv(csv_path,None)
```

对属性进行分类，分为标称属性和数值属性：

```python
#标称属性
name_category = ['Drive', 'qtr', 'down', 'SideofField', 'ydstogo', 'GoalToGo', 'FirstDown', 'posteam',
                 'DefensiveTeam', 'PlayAttempted', 'sp', 'Touchdown', 'ExPointResult', 'TwoPointConv',
                 'DefTwoPoint', 'Safety', 'Onsidekick', 'PuntResult', 'PlayType', 'Passer', 'Passer_ID',
                 'PassAttempt', 'PassOutcome', 'PassLength', 'QBHit', 'PassLocation', 'InterceptionThrown',
                 'Interceptor', 'Rusher', 'Rusher_ID', 'RushAttempt', 'RunLocation', 'RunGap', 'Receiver',
                 'Receiver_ID', 'Reception', 'ReturnResult', 'Returner', 'BlockingPlayer', 'Tackler1', 'Tackler2',
                 'FieldGoalResult', 'Fumble', 'RecFumbTeam', 'RecFumbPlayer', 'Sack', 'Challenge.Replay',
                 'ChalReplayResult', 'Accepted.Penalty', 'PenalizedTeam', 'PenaltyType', 'PenalizedPlayer',
                 'HomeTeam', 'AwayTeam', 'Timeout_Indicator', 'Timeout_Team', 'Season']
#数值属性
name_value = ['TimeUnder', 'TimeSecs', 'PlayTimeDiff', 'yrdln', 'yrdline100', 'ydsnet', 'Yards.Gained',
              'AirYards', 'YardsAfterCatch', 'FieldGoalDistance', 'Penalty.Yards', 'PosTeamScore', 'DefTeamScore',
              'ScoreDiff', 'AbsScoreDiff', 'posteam_timeouts_pre', 'HomeTimeouts_Remaining_Pre', 'AwayTimeouts_Remaining_Pre',
              'HomeTimeouts_Remaining_Post', 'AwayTimeouts_Remaining_Post', 'No_Score_Prob', 'Opp_Field_Goal_Prob',
              'Opp_Safety_Prob', 'Opp_Touchdown_Prob', 'Field_Goal_Prob', 'Safety_Prob', 'Touchdown_Prob', 'ExPoint_Prob',
              'TwoPoint_Prob', 'ExpPts', 'EPA', 'airEPA', 'yacEPA', 'Home_WP_pre', 'Away_WP_pre', 'Home_WP_post',
              'Away_WP_post', 'Win_Prob', 'WPA', 'airWPA', 'yacWPA']
```

2 数据可视化和摘要

2.1 数据摘要

标称数据：

对于标称属性，使用 pandas 中的 value_counts 函数统计每个标称属性的所有可能取值的频数。

```python
def count(dataFrame, columns, dropna=False, format_width=30):
    '标称属性，给出每个可能取值的频数'
    format_text = '{{:<{0}}}{{:<{0}}}'.format(format_width)
    for col in columns:
        print('标称属性 <{}> 频数统计'.format(col))
        print(format_text.format('value', 'count'))
        print('- ' * format_width)

        counts = pd.value_counts(dataFrame[col].values, dropna=False)
        for i, index in enumerate(counts.index):
            if pd.isnull(index): # NaN?
                print(format_text.format('-NaN-', counts.values[i]))
            else:
                print(format_text.format(index, counts[index]))
        print('--' * format_width)
        print()

count(dataFrame,name_category)
```

部分结果如下：

标称属性 <Drive> 频数统计

value	count
1	19261
7	18088
6	17925
4	17875
5	17822
8	17691
16	17525
17	17459
3	17425
18	17339
10	17193
15	17188
2	17084
9	17033
19	16928
11	16898
14	16881
13	16745
12	16724
20	15622
21	13998
22	12071
23	9933
24	7959
25	5842
26	4200
27	2815
28	1658
29	1077
30	675
31	366
32	237
33	97
34	45
35	9

标称属性 <qtr> 频数统计

value	count
4	112641
2	112317
3	90682
1	89176
5	2872

标称属性 <down> 频数统计

value	count
1.0	138878
2.0	104089
3.0	67398
-NaN-	61154
4.0	36169

数值属性：

对于数值属性，使用 pandas 中 describe()函数给出其最小、最大、均值、中位数、四分位数及缺失值个数：

```python
def describe(dataFrame, columns):
    '数值属性，给出最大、最小、均值、中位数、四分位数及缺失值的个数'
    desc = dataFrame[columns].describe()
    statistic = DataFrame()
    statistic['max'] = desc.loc['max']
    statistic['min'] = desc.loc['min']
    statistic['mean'] = desc.loc['mean']
    statistic['50%'] = desc.loc['50%']
    statistic['25%'] = desc.loc['25%']
    statistic['75%'] = desc.loc['75%']
    statistic['NaN'] = dataFrame[columns].isnull().sum()
    print(statistic)

describe(dataFrame,name_value)
```

部分结果如下：

	max	min	mean	50%	25%	75%	NaN
TimeUnder	15.000000	0.000000	7.374200	7.000000	3.000000	11.000000	0
TimeSecs	3600.000000	-900.000000	1695.268944	1800.000000	778.000000	2585.000000	224
PlayTimeDiff	943.000000	0.000000	20.576762	17.000000	5.000000	37.000000	444
yrdln	50.000000	1.000000	28.488327	30.000000	20.000000	39.000000	840
yrdline100	99.000000	1.000000	48.644081	49.000000	30.000000	70.000000	840
ydsnet	99.000000	-87.000000	25.945517	19.000000	5.000000	43.000000	0
Yards.Gained	99.000000	-74.000000	4.994221	1.000000	0.000000	7.000000	0
AirYards	84.000000	-70.000000	3.264006	0.000000	0.000000	4.000000	0
YardsAfterCatch	90.000000	-81.000000	1.252598	0.000000	0.000000	0.000000	0
FieldGoalDistance	71.000000	18.000000	37.465132	38.000000	29.000000	46.000000	398740
Penalty.Yards	66.000000	0.000000	0.613673	0.000000	0.000000	0.000000	0
PosTeamScore	61.000000	0.000000	10.201424	7.000000	2.000000	16.000000	26904
DefTeamScore	61.000000	0.000000	11.414484	10.000000	3.000000	17.000000	26904
ScoreDiff	59.000000	-59.000000	-1.186590	0.000000	-7.000000	4.000000	24988
AbsScoreDiff	59.000000	0.000000	7.783541	7.000000	3.000000	11.000000	26904
posteam_timeouts_pre	3.000000	0.000000	2.521239	3.000000	2.000000	3.000000	0
HomeTimeouts_Remaining_Pre	3.000000	-3.000000	2.540479	3.000000	2.000000	3.000000	0
AwayTimeouts_Remaining_Pre	3.000000	-1.000000	2.517222	3.000000	2.000000	3.000000	0
HomeTimeouts_Remaining_Post	3.000000	-3.000000	2.520118	3.000000	2.000000	3.000000	0
AwayTimeouts_Remaining_Post	3.000000	-1.000000	2.496367	3.000000	2.000000	3.000000	0
No_Score_Prob	1.000000	0.000000	0.127816	0.024771	0.002791	0.172509	176
Opp_Field_Goal_Prob	0.360177	0.000000	0.094614	0.083088	0.034599	0.149943	176
Opp_Safety_Prob	0.031461	0.000000	0.002495	0.000988	0.000104	0.003845	176
Opp_Touchdown_Prob	0.496874	0.000000	0.139973	0.124032	0.039834	0.226408	176
Field_Goal_Prob	0.994605	0.000000	0.243906	0.231311	0.152443	0.326130	176
Safety_Prob	0.015177	0.000000	0.002634	0.002990	0.001883	0.003582	176
Touchdown_Prob	0.912963	0.000000	0.295940	0.313676	0.191206	0.407684	176

2.2 数据可视化

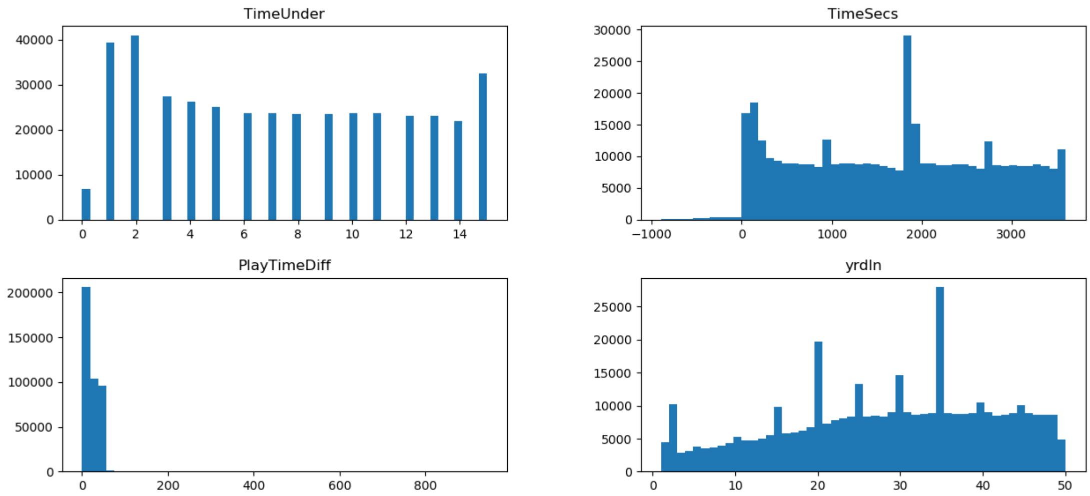
直方图

使用 matplotlib 绘制直方图

```python
def histogram(dataFrame, columns):
    '直方图'
    for i, col in enumerate(columns):
        if i % cell_size == 0:
            fig = plt.figure()
        ax = fig.add_subplot(col_size, row_size, (i % cell_size) + 1)
        dataFrame[col].hist(ax=ax, grid=False, figsize=(15, 15), bins=50)
        plt.title(col)
        if (i + 1) % cell_size == 0 or i + 1 == len(columns):
            plt.subplots_adjust(wspace=0.3, hspace=0.3)
            plt.show()

histogram(dataFrame,name_value)
```

部分结果如下：



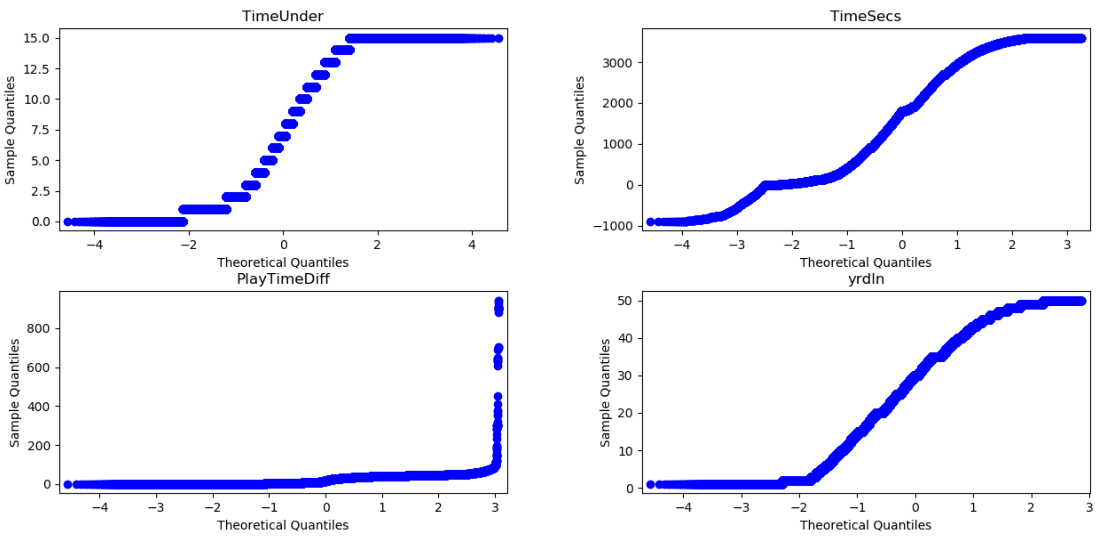
qq 图：

使用 matplotlib 绘制 qq 图

```
def qqplot(dataFrame, columns):
    'qq图'
    for i, col in enumerate(columns):
        if i % cell_size == 0:
            fig = plt.figure(figsize=(15, 15))
        ax = fig.add_subplot(col_size, row_size, (i % cell_size) + 1)
        sm.qqplot(dataFrame[col], ax=ax)
        ax.set_title(col)
        if (i + 1) % cell_size == 0 or i + 1 == len(columns):
            plt.subplots_adjust(wspace=0.3, hspace=0.3)
            plt.show()

qqplot(dataFrame,name_value)
```

部分结果如下：



根据 qq 图可知图像 1、2 和 4 是近似直线的，其对应属性（TimeUnder、TimeSecs、yrdln）为正态分布态。

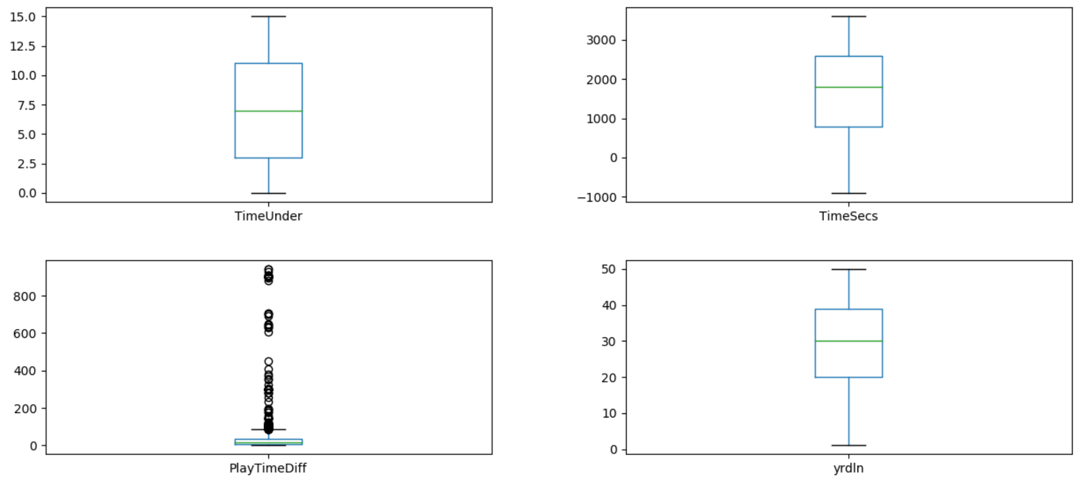
盒图：

使用 matplotlib 绘制盒图，对离群值进行识别：

```
def boxplot(dataFrame, columns):
    '盒图'
    for i, col in enumerate(columns):
        if i % cell_size == 0:
            fig = plt.figure()
        ax = fig.add_subplot(col_size, row_size, (i % cell_size) + 1)
        dataFrame[col].plot.box(ax=ax, figsize=(15, 15))
        if (i + 1) % cell_size == 0 or i + 1 == len(columns):
            plt.subplots_adjust(wspace=0.3, hspace=0.3)
            plt.show()

boxplot(dataFrame,name_value)
```

部分结果如下：



3 数据缺失处理

3.1 将缺失部分剔除

根据分析，可填充的数值属性字段有：No_Score_Prob, Opp_Field_Goal_Prob, Opp_Safety_Prob, Opp_Touchdown_Prob, Field_Goal_Prob, 'Safety_Prob', Touchdown_Prob, ExpPts, EPA, airEPA, yacEPA, Home_WP_pre, Away_WP_pre, Home_WP_post, Away_WP_post, Win_Prob, WPA, airWPA, yacWPA。对缺失部分进行剔除

```
cols = ['No_Score_Prob', 'Opp_Field_Goal_Prob', 'Opp_Safety_Prob', 'Opp_Touchdown_Prob', 'Field_Goal_Prob',
        'Safety_Prob', 'Touchdown_Prob', 'ExpPts', 'EPA', 'airEPA', 'yacEPA', 'Home_WP_pre', 'Away_WP_pre',
        'Home_WP_post', 'Away_WP_post', 'Win_Prob', 'WPA', 'airWPA', 'yacWPA']
index = dataFrame[cols].isnull().sum(axis=1) == 0
df_fillna = dataFrame[index]
compare(dataFrame,df_fillna,cols)
```

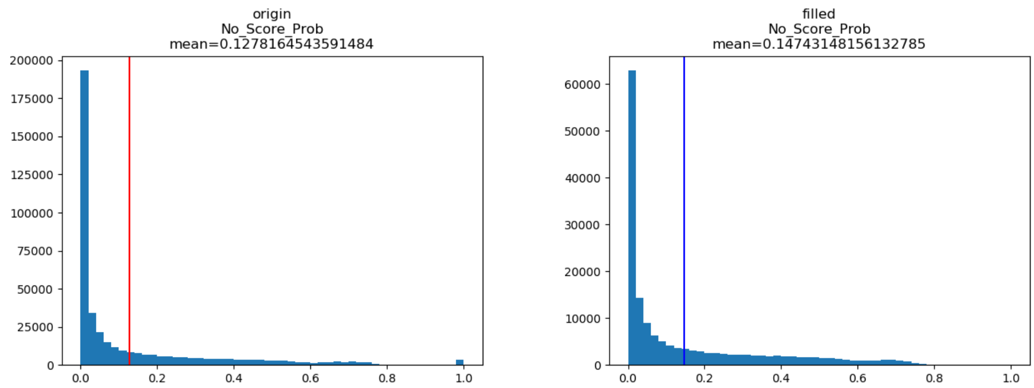
通过直方图比较新旧数据集的数值属性：

```python
def compare(df1, df2, columns, bins=50):
    '直方图比较'
    for col in columns:
        mean1 = df1[col].mean()
        mean2 = df2[col].mean()

        fig = plt.figure()
        ax1 = fig.add_subplot(121)
        df1[col].hist(ax=ax1, grid=False, figsize=(15, 5), bins=bins)
        ax1.axvline(mean1, color='r')
        plt.title('origin\n{}\nmean={}'.format(col, str(mean1)))
        ax2 = fig.add_subplot(122)
        df2[col].hist(ax=ax2, grid=False, figsize=(15, 5), bins=bins)
        ax2.axvline(mean2, color='b')
        plt.title('filled\n{}\nmean={}'.format(col, str(mean2)))
        plt.subplots_adjust(wspace = 0.3, hspace = 10)
        plt.show()

compare(dataFrame,df_fillna,cols)
```

部分结果如下：



在直方图中，左边的红色垂线表示旧数据集的均值，右边的蓝色垂线表示剔除有缺失的数据得到的新数据集的均值。

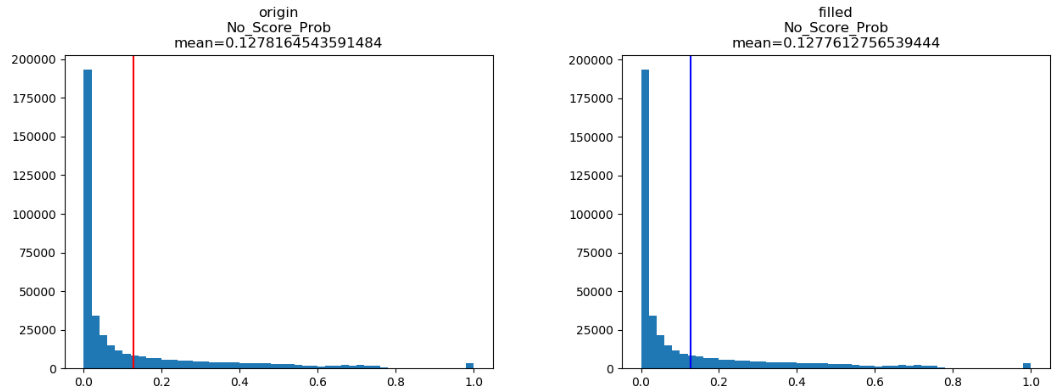
3.2 用最高频率值来填补缺失值

找到每个属性中出现次数最多的值，用这个值填充这个属性中所有的缺失值：

```python
# 建立原始数据的拷贝
df_filled = dataFrame.copy()
# 对每一列数据，分别进行处理
for col in cols:
    # 计算最高频率的值
    most_frequent_value = df_filled[col].value_counts().idxmax()
    # 替换缺失值\n",
    df_filled[col].fillna(value=most_frequent_value, inplace=True)

compare(dataFrame,df_filled,cols)
```

在直方图中，左边的红色垂线表示旧数据集的均值，右边的蓝色垂线表示剔除有缺失的数据得到的新数据集的均值。



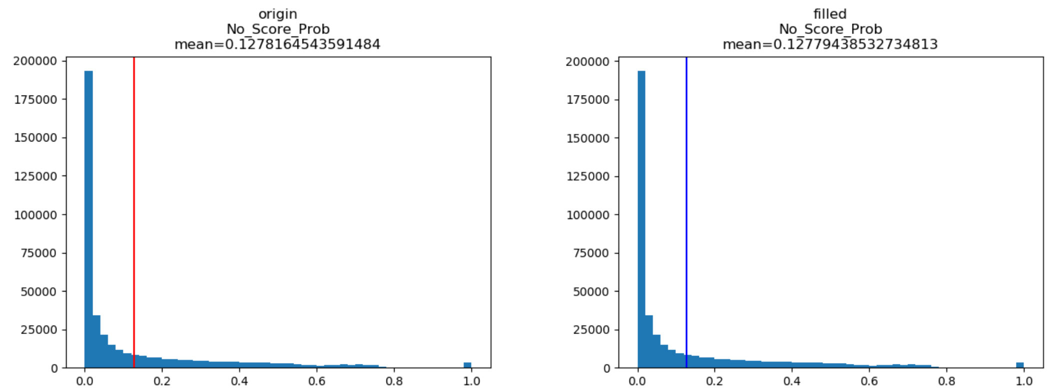
3.3 通过属性的相关关系来填补缺失值

使用 pandas 中的 interpolate()函数，对于每个数值属性进行插值计算，利用得到的插值填充缺失值：

```
# 建立原始数据的拷贝
df_filled_inter = dataFrame.copy()
    # 对每一列数据，分别进行处理
    for col in cols:
        df_filled_inter[col].interpolate(inplace=True)

compare(dataFrame,df_filled_inter,cols)
```

在直方图中，左边的红色垂线表示旧数据集的均值，右边的蓝色垂线表示剔除有缺失的数据得到的新数据集的均值。



3.4 通过数据对象之间的相似性来填补缺失值

无