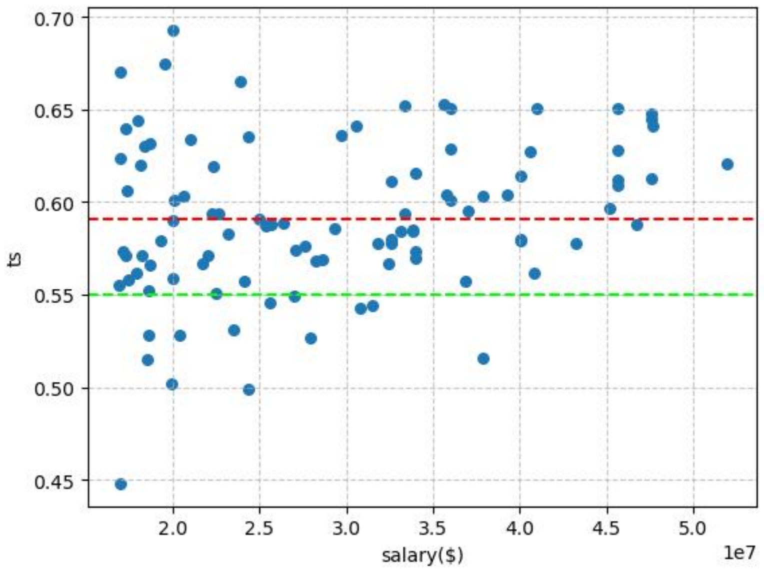
```
import pandas as pd
import matplotlib.pyplot as plt
import matplotlib


file_path = r'./salary2.xls'
df = pd.read_excel(file_path, sheet_name = "Sheet1")
list1 = df.values.tolist()

x_values = [player[3] for player in list1]      #球员年薪
y_values = [player[4] for player in list1]      #球员真实命中率
y_average=sum(item[4] for item in list1)/98    #高薪球员平均真实命中率
total_average=0.55                              #全联盟平均真实命中率

plt.scatter(x_values,y_values,s=30)
plt.xlabel("salary($)")
plt.ylabel("ts")

plt.axhline(y_average,color='red',linestyle='--')
plt.axhline(total_average,color='lime',linestyle='--')
plt.grid(True, linestyle='--', alpha=0.7)
plt.show()
print("高薪球员平均真实命中率为{}%".format(100*y_average))
```

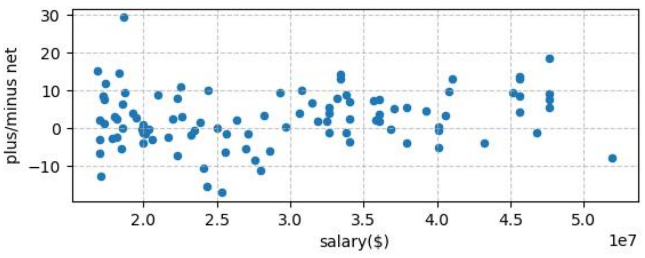


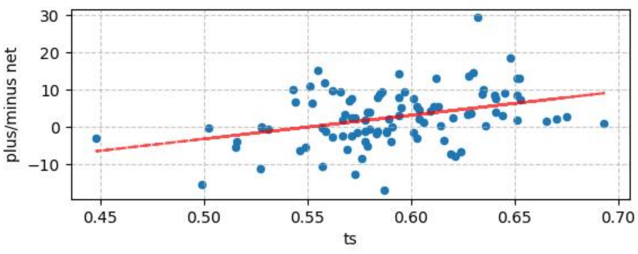
```
sum_above=0
sum_below=0
total_below=0
for player in list1:
    if player[4] < total_average:
        total_below += 1
    if player[3] >= 35000000:
        if player[4] >= y_average:
            sum_above += 1
    else:
        if player[4] >= y_average:
            sum_below += 1
print("高薪球员真实命中率不足全联盟平均水平的有{}个，占全部高薪球员的{}%".format(total_below,100*total_below/98))
print("年薪超过 3500 万的球员真实命中率超过高薪球员平均水平的有{}个，占全部年薪超过 3500 万的球员的{}%".format(sum_above,100*sum_above/28))
print("年薪低于 3500 万的高薪球员真实命中率超过高薪球员平均水平的有{}个，占全部年薪低于 3500 万的高薪球员的{}%".format(sum_below,100*sum_below/70))
```

```
高薪球员真实命中率不足全联盟平均水平的有 13 个，占全部高薪球员的 13.26530612244898%
年薪超过 3500 万的球员真实命中率超过高薪球员平均水平的有 21 个，占全部年薪超过 3500 万的球员的 75.0%
年薪低于 3500 万的高薪球员真实命中率超过高薪球员平均水平的有 24 个，占全部年薪低于 3500 万的高薪球员的 34.285714285714285%
```

```
from sklearn.linear_model import LinearRegression
import numpy as np

plt.figure(1)

plt.subplot(2,1,1)

y_values = [player[8] for player in list1]     #球员净正负值

plt.scatter(x_values,y_values,s=20)
plt.xlabel("salary($)")
plt.ylabel("plus/minus net")
plt.grid(True, linestyle='--', alpha=0.7)

plt.figure(2)

plt.subplot(2,1,2)
x_values = [player[4] for player in list1]     #球员真实命中率

X_train = np.array(x_values).reshape((len(x_values), 1))
Y_train = np.array(y_values).reshape((len(y_values), 1))
lineModel = LinearRegression()
lineModel.fit(X_train, Y_train)
Y_predict = lineModel.predict(X_train)
a1 = lineModel.coef_[0][0]
b = lineModel.intercept_[0]
print("y=%.4f*x+%.4f" % (a1,b))
```

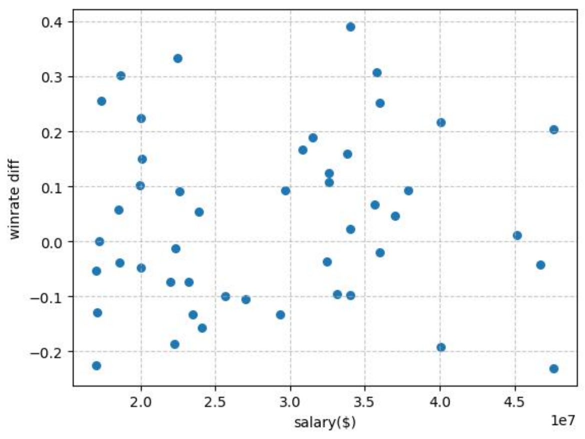
```
plt.plot(X_train,Y_predict, c='red',linestyle='--',alpha=0.7)
plt.scatter(x_values,y_values,s=20)
plt.xlabel("ts")
plt.ylabel("plus/minus net")
plt.grid(True, linestyle='--', alpha=0.7)
plt.show()


sum_below=0
sum_above=0


for player in list1:
    if player[3] >= 30000000:
        if player[8] >= 0:
            sum_above += 1
    else:
        if player[8] >= 0:
            sum_below += 1
print(sum_above/44)
print(sum_below/54)
```





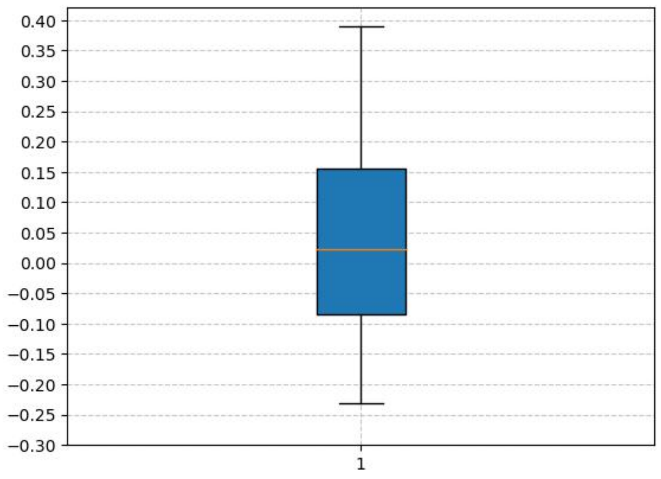
```
import csv


file_path = r'./standings.xls'
df = pd.read_excel(file_path, sheet_name = "Sheet1")


dictionary = df.to_dict('records')
team_to_win = {record['Team']: record['Win'] for record in dictionary}
```

```
team_to_loss = {record['Team']: record['Loss'] for record in dictionary}

winrate_matrix = []
for player in list1:
    if player[6] >= 5:              #球员缺席场数
        abs_winrate = round(player[7] / player[6],4)         #球员缺席时球队胜率
        pre_winrate = round((team_to_win[player[2]] - player[7])/(team_to_win[player[2]] + team_to_loss[player[2]] - player[6]),4)
        #球员出勤时球队胜率
        winrate_matrix.append([player[1],player[2],player[3],pre_winrate,abs_winrate,round(pre_winrate - abs_winrate,4),player[8]])
        #在新表格中引入球员名字，所属球队，薪资，出勤时球队胜率，缺席时球队胜率，两者差值，净正负值

csv_file_path = 'winrate2.csv'
with open(csv_file_path, 'w', newline='') as csvfile:
    csv_writer = csv.writer(csvfile)

    csv_writer.writerow(['Name', 'Team', 'Salary','pre_winrate','abs_winrate','diff','PM'])
    for row in winrate_matrix:
        csv_writer.writerow(row)

x_values = [player[2] for player in winrate_matrix]
y_values = [player[5] for player in winrate_matrix]

plt.scatter(x_values,y_values,s=30)
plt.xlabel("salary($)")
plt.ylabel("winrate diff")
plt.grid(True, linestyle='--', alpha=0.7)

plt.show()
```

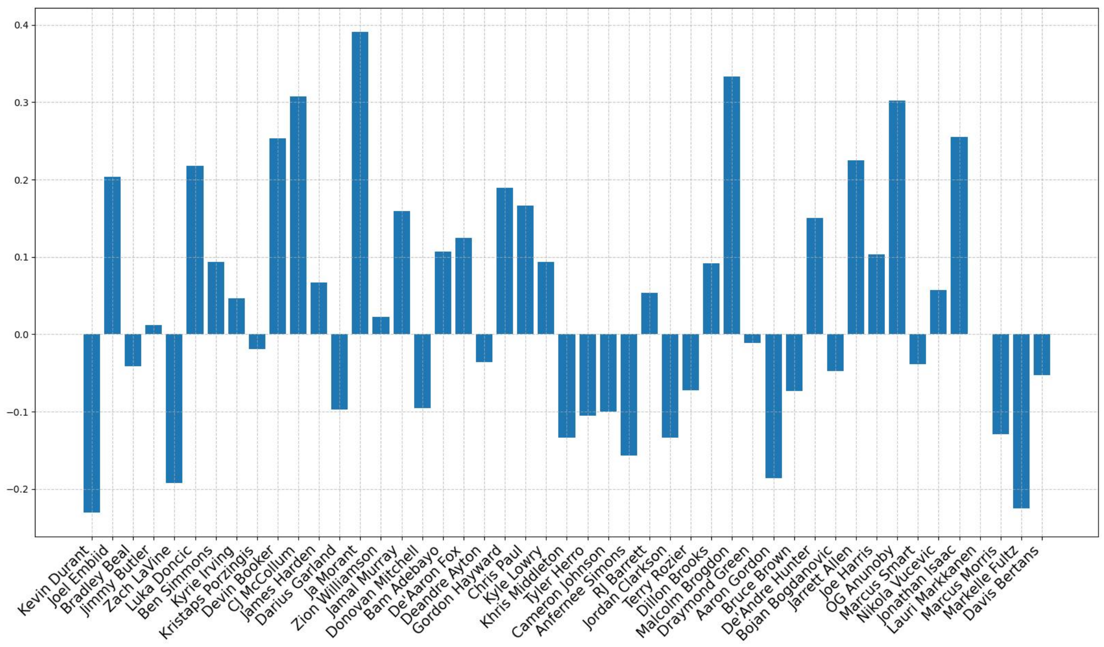


```
boxplot = plt.boxplot(y_values,vert=True, patch_artist=True)
```

```
plt.yticks(np.arange(-0.3, 0.45, 0.05))

plt.grid(True,linestyle='--', alpha=0.7)

plt.show()
```

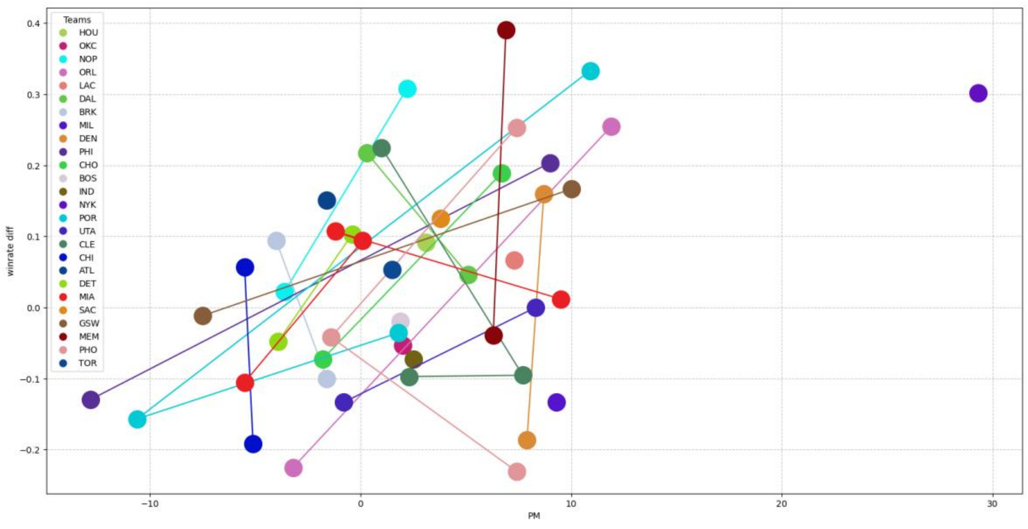


```
players = [player[0] for player in winrate_matrix]


plt.figure(figsize=(20, 10))

plt.bar(players,y_values)

plt.xticks(rotation=45, ha='right', fontsize=15)

plt.grid(True, linestyle='--', alpha=0.7)

plt.show()
```



```
x_values = [player[6] for player in winrate_matrix]
```

```
team_colors = {}
teams = [player[1] for player in winrate_matrix]
for team in set(teams):
    team_colors[team] = np.random.rand(3,)
colors = [team_colors[team] for team in teams]

plt.figure(figsize=(20, 10))
plt.grid(True, linestyle='--', alpha=0.7)
plt.xlabel("PM")
plt.ylabel("winrate diff")
legend_labels = [plt.Line2D([0], [0], marker='o', color='w', markerfacecolor=team_colors[team], markersize=10, label=team) for team in set(teams)]
plt.legend(handles=legend_labels, title='Teams')
for team in set(teams):
    indices = [i for i, x in enumerate(teams) if x == team]
    plt.plot([x_values[i] for i in indices], [y_values[i] for i in indices], color=team_colors[team], linestyle='-', marker='o', markersize=20)
plt.show()
```



```
winrate_matrix.sort(key=lambda x: x[1])
list1.sort(key=lambda x: x[2])
csv_file_path = 'sorted_winrate.csv'
with open(csv_file_path, 'w', newline='') as csvfile:
    csv_writer = csv.writer(csvfile)


    csv_writer.writerow(['Name', 'Team', 'Salary','pre_winrate','abs_winrate','diff','PM'])
    for row in winrate_matrix:
        csv_writer.writerow(row)
csv_file_path = 'sorted_ts.csv'
with open(csv_file_path, 'w', newline='') as csvfile:
    csv_writer = csv.writer(csvfile)

    csv_writer.writerow(['Name', 'Team', 'salary','ts','PM'])
    for row in list1:
        csv_writer.writerow([row[1],row[2],row[3],row[4],row[8]])
```

爬虫 1：获取球队战绩

```
import scrapy
from scrapy import Selector

class StandingSpider(scrapy.Spider):
    name = "standing"
    allowed_domains = ["www.basketball-reference.com"]
    start_urls = ["https://www.basketball-reference.com/"]

    def parse(self, response):
        standings = response.css("#confs_standings_E > tbody > tr")
        #confs_standings_W > tbody > tr
        for standing in standings:
            teamname = standing.css("th > a::text").get()
            win = standing.css("td:nth-child(4)").get()
            lose = standing.css("td:nth-child(5)").get()
            yield{
                "teamname":teamname,
                "win":win,
                "lose":lose,
            }
```

爬虫 2：获取球员薪资信息

```
import scrapy
from scrapy import Selector


class Salary1Spider(scrapy.Spider):
    name = "salary1"
    allowed_domains = ["hoopshype.com"]
    start_urls = ["https://hoopshype.com/salaries/players/"]

    def parse(self, response):
        player_rows = response.css("#content-container > div >
div.hoopshype-salaries-wrap.tabs.tabs-noinit.hoopshype-salaries-players > div.hh-salaries-ranking > table > tbody")
```

```
for player_row in player_rows:
    player_name = player_row.css("td.name>a::text").get()
    salary_2023_24 = player_row.css("td.hh-salaries-sorted").extract_first()

    yield {
        "player_name": player_name,
        "salary_2023_24": salary_2023_24,
    }
```