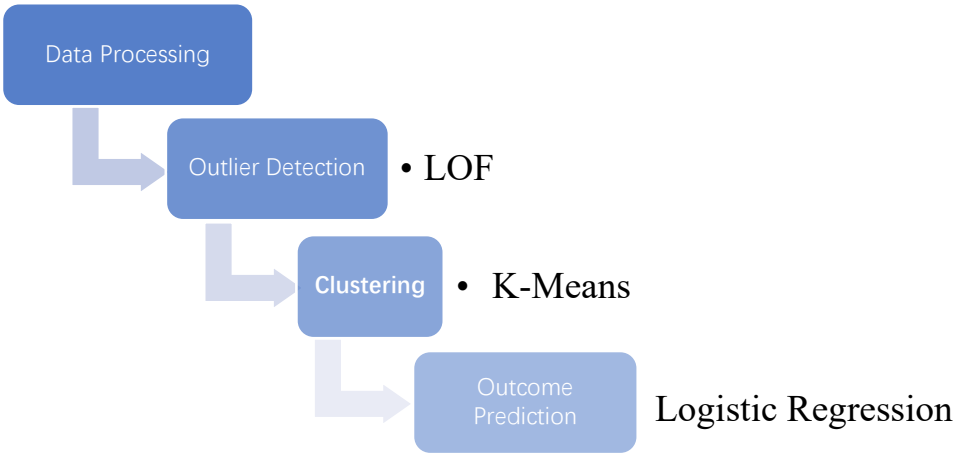
NBA Outstanding Player and Match Result Prediction using Machine Learning

Project Report

Contributors:
Minghao Wang, net ID: mw5945
Jiahang Zhang, net ID: jz7581

1. Project Overview

This report details the application of machine learning techniques to analyze NBA historical data, addressing two core problems: 1) identifying outstanding players through outlier detection and clustering, and 2) predicting game outcomes using classification models. Methods including Local Outlier Factor (LOF), K-Means clustering, Logistic Regression and Principal Component Analysis (PCA). This project underscores the practical utility of machine learning in sports analytics for objective performance evaluation and strategic forecasting.



2. Problem Formulation and Solutions

2.1. Outlier Detection and Excellent Player Selection:

Problem Definition:

The selection of excellent players should be defined as a clustering problem. Based on reviewing player historical data, besides basic personal information, valuable information includes games played, total minutes played, total rebounds, total assists, total steals, total blocks, personal fouls, total turnovers, field goal attempts and makes, free throw attempts and makes. Because labels are lacking and we need the machine to select the best players, clustering is the best method. We adopt the LOF algorithm to detect outliers, which contain information about extremely good players and extremely poor players. By performing K-Means clustering on the outlier information, we can find exceptionally outstanding players.

LOF Algorithm Description:

LOF Algorithm: Local Outlier Factor (LOF) is a classic density-based algorithm. LOF aims to discover abnormal patterns in a dataset, which depends on the density comparison between a sample point and its surrounding neighbors. The core idea of LOF is that abnormality depends on the local environment. The advantages of LOF are its simplicity, intuitiveness, not needing to know the data distribution, and its ability to quantify the degree of abnormality for each sample point.

Local Density: LOF believes that the closer the k-th nearest neighbor of a sample point p is, the more neighbors are near it, and the greater its local density is; conversely, the farther the k-th nearest neighbor is, the smaller its local density is. Therefore, LOF defines the local density of sample point p as the reciprocal of the distance to its k-th nearest neighbor.

Degree of Abnormality: Whether p is abnormal does not depend on p's local density but on the comparison between p's local density and the local densities of its neighbors. For example, if p's local density is small, but the local densities of its neighbors are also small, then p's degree of abnormality is low. Conversely, if p's local density is small, but the local densities of its neighbors are large, then p's degree of abnormality is high.

Algorithm Steps:

Step 1: Calculate the local density of all sample points; calculate the distance from all other points to sample point p, sort in ascending order, take the k-th distance as k-distance; The local density of sample point p $\rho = 1/k - distance$;

Note: If there are k or more points coinciding with p, i.e., the distance to p is 0, then p cannot be calculated; exclude this situation. Alternatively, add a very small value to k-distance to avoid p being incalculable.

Step 2: Calculate the anomaly score for all sample points; denote the set of all points whose distance to sample point p is less than or equal to k-distance as N, i.e., the neighbors within p's k-distance; calculate the local density of all points in N and take their average, denoted as $\rho - mean$; The anomaly score for sample point p: score = $\rho - mean/\rho$.

Step 3: Analyze the degree of abnormality:

If the anomaly score is close to 1, it indicates that the local density of sample point p is similar to that of its neighbors.

If the anomaly score is less than 1, it indicates that p is in a relatively dense area and is not likely an outlier.

If the anomaly score is much greater than 1, it indicates that p is relatively distant from other points and is likely an outlier.

2.2. Game Results Prediction

Problem Definition: Predicting game results should be a **classification problem**, i.e., predicting game outcomes based on the status attributes of the opposing teams. I downloaded data for every game (preseason + regular season + playoffs) and team ability evaluations for the years 2022-2024 from basketball-reference.com. Useful information extracted includes the final scores of both teams in each game, which team is home/away, team offensive efficiency, defensive efficiency, etc. Through feature engineering, using the win/loss of each game as the label, a predictive model can

be obtained through training. I used PCA and Pearson heatmap to select features, and used logistic regression methods to learn from the data.

Algorithm Description:

Logistic Regression: Assuming data follows a Bernoulli distribution, parameters are solved by maximizing the likelihood function using gradient descent to achieve the purpose of binary classification of data. The main advantages of this model are good interpretability; if feature engineering is done well, the model effect is also very good; training speed is relatively fast; output results are also easy to adjust.

Algorithm Steps:

Step 1: Calculate the predicted output result: Build a predictive model $Y = WX + b$ to predict the data result.

Step 2: Construct the loss function based on the original label and Y.

Step 3: Calculate the gradients of Y, W, b.

Step 4: Update W and b using gradient descent, repeat the above steps until the minimized cost function $J(W, b)$ is obtained.

3. Algorithms Implementation and Results:

3.1. Outlier Detection and Excellent Player Selection:

Data Reading:

Data information was downloaded and saved in txt files, separated by ','. Python was used to read the data into a dictionary and convert it into a DataFrame structure. Career data for players in the regular season and playoffs were separately counted. Yearly data was not chosen because an outstanding performance in one season cannot represent a player's true level; the average accumulated over many years better demonstrates a player's excellence.

	ilkid	firstname	lastname	leag	gp	...	fgm	fta	ftm	tpa	tpm
3754	ZEVENPH01	Phil	Zevenbergen	N	8	...	15	2	0	0	0
3755	ZIDEKGE01	George	Zidek	N	135	...	161	166	130	4	1
3756	ZOETJI01	Jim	Zoet	N	7	...	1	0	0	0	0
3757	ZOPFBI01	Bill	Zopf	N	53	...	49	36	20	0	0
3758	ZUNICMA01	Matt	Zunic	N	56	...	98	109	77	0	0

Last 5 player read-in information

Data Cleaning: Exclude missing (NAN) or blank data.

Feature Construction: Find features that can reflect the excellence of players. Based on existing player data, we constructed player averages per game: minutes played, points, turnovers, rebounds, assists, blocks, field goal percentage, and free throw percentage.

minutes_ave	pts_ave	to_ave	reb_ave	asts_ave	stl_ave	blk_ave	fg_acc	ft_acc
17.78	7.90	0.0	2.91	1.17	0.0	0.0	0.39	0.71
12.52	14.80	0.0	1.60	2.02	0.0	0.0	0.34	0.77
19.74	6.76	0.0	4.09	1.21	0.0	0.0	0.37	0.77
4.56	1.48	0.0	0.44	0.95	0.0	0.0	0.35	0.75
7.51	2.23	0.0	0.87	1.38	0.0	0.0	0.36	0.56

Newly constructed data

Feature Organization:

Delete intermediate features used for processing, retain final features for learning, and simultaneously remove potentially erroneous information and useless data, such as players with average playing time less than 3 minutes (definitely not excellent players). Also, considering that steals, blocks, and turnovers were not recorded before 1970, the dataset was divided based on whether steals, blocks, and turnovers were all 0 to find outliers separately, improving algorithm robustness.

Outlier Detection: Put the obtained data features into the written outlier detection function for detection.

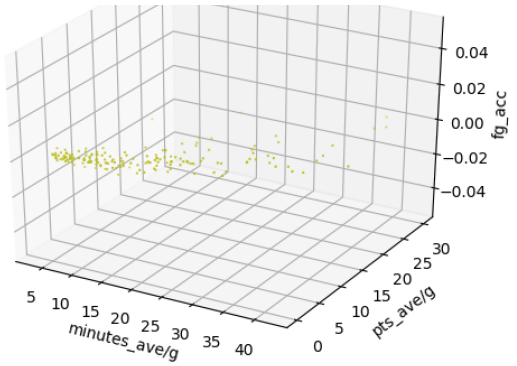
ilkid	name	minutes_ave	pts_ave	to_ave	reb_ave	asts_ave	stl_ave	blk_ave	fg_acc	ft_acc
BLANKLA01	Lance_Blanks	10.00	2.00	1.00	1.00	3.00	3.00	0.00	0.50	0.00
BOLMA01	Manute_Bol	17.10	2.83	0.45	3.76	0.10	0.21	2.66	0.39	0.44
BORRELA01	Lazaro_Borrell	13.00	5.00	0.50	5.50	0.50	0.00	0.00	0.57	0.50
BRYANWA01	Wallace_Bryant	18.00	1.00	1.00	3.50	0.50	0.50	0.50	0.00	1.00
EVANSRE01	Reggie_Evans	18.91	3.73	1.00	7.36	0.45	0.45	0.27	0.41	0.52
FORTSDA01	Danny_Fortson	9.55	3.27	1.18	2.36	0.00	0.27	0.09	0.57	0.80

Partially detected outlier information

It can be seen that the data in the detected outliers are either very good excellent players (the exceptionally outstanding players we are looking for, outstanding in some aspect) or players with consistently poor data.

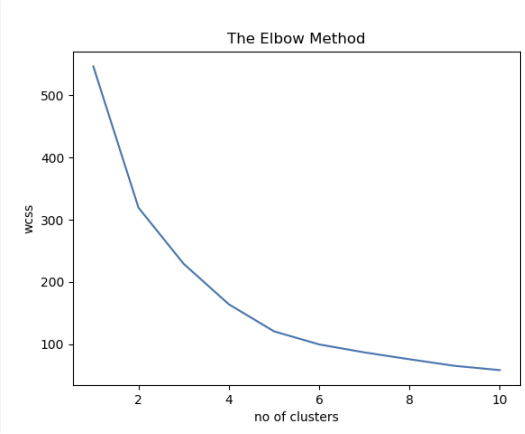
Screening Excellent Players from Outliers:

We select average points per game, average playing time per game, and shooting percentage as features to create a three-dimensional image to first view the distribution of outlier data. These three data points can, to some extent, reflect a player's excellence because the processed data does not differentiate positions such as center, small forward, etc. Using rebounds, assists, and blocks to evaluate players is too one-sided; for example, guards may score high in assists but have few rebounds and blocks, while centers have advantages in rebounds. Playing time, points, and shooting percentage are three factors that comprehensively reflect a player's level.

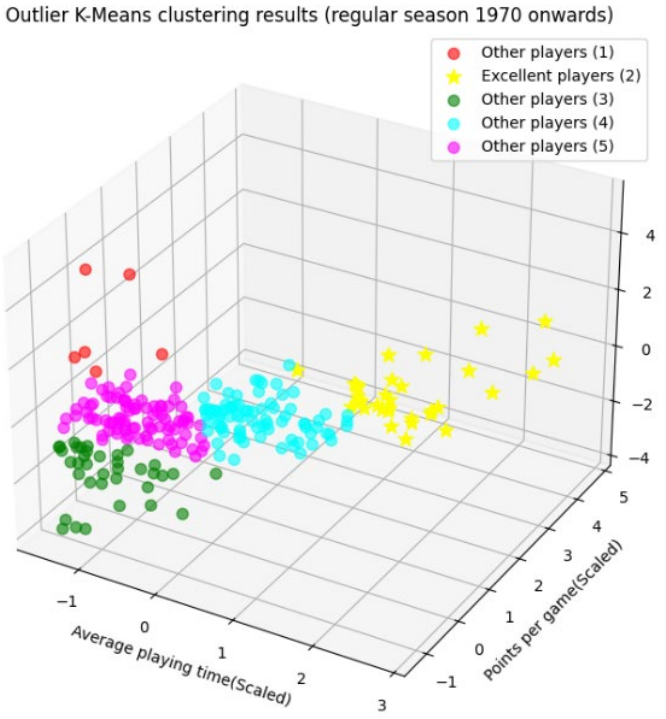


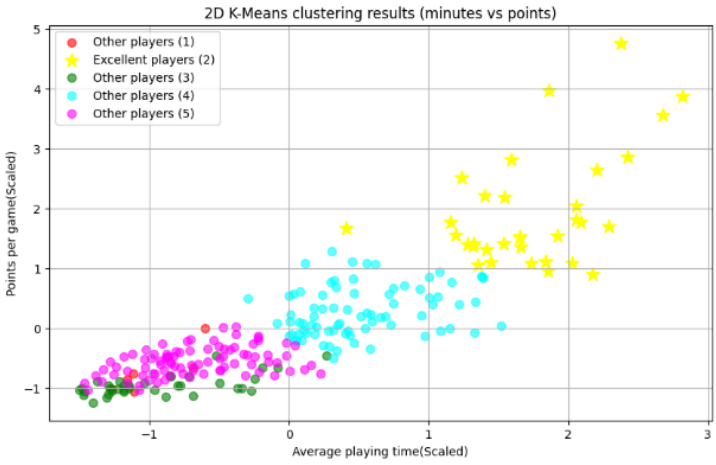
It can be found that most of the outliers found have low points and playing time, i.e., they are poor-performing players. The very few prominent points with high points, playing time, and shooting percentage are the excellent players we are interested in.

We use the K-Means clustering method to screen out excellent players, where the number of clusters is determined by the elbow method.



It can be seen that the data shows a clear turning point at 5 clusters, so 5 is chosen as the number of clusters.





From the graph, it can be seen that excellent players are grouped into one category.

Based on the K-Means results, the cluster containing excellent players is selected by having the highest average points, thus obtaining the final excellent players. Regular season and playoff data are merged separately to obtain the excellent player sets. By comparing the regular season excellent player set with the playoff excellent player set and taking the intersection, we can find players who have performed exceptionally well in both over their careers.

Results:

- **LOF (contamination=0.1):** Identified 25 excellent regular-season players, 10 excellent playoff players, and 3 players excellent in both (e.g., George Mikan).
- **LOF (contamination=0.2):** A less strict threshold identified 36, 21, and 7 players respectively, now including legends like Wilt Chamberlain and Michael Jordan.

Outputs:

```
Total time cost: :     0.08s
There are: 36 Players who performed well in the regular season:
['Alvan_Adams', 'Lucius_Allen', 'Elgin_Baylor', 'Bill_Bridges', 'Wilt_Chamberlain', 'Walter_Davis', 'Michael_Dickerson', 'Vlade_Divac',
'George_Gervin', 'Dan_Issel', 'LeBron_James', 'John_Johnson', 'Magic_Johnson', 'Michael_Jordan', 'Jason_Kidd', 'Andrei_Kirilenko', 'Lafay
ette_Lever', 'Stephon_Marbury', 'Rodney_Mccray', 'George_Mcginnis', 'Dikembe_Mutombo', 'Emeka_Okafor', 'Bob_Pettit', 'Kevin_Porter', 'Alv
in_Robertson', 'Oscar_Robertson', 'Bob_Rule', 'Cazzie_Russell', 'Jerry_Sloan', 'John_Stockton', 'Wes_Unseld', 'Norm_Vanlier', 'Bill_Walto
n', 'Scott_Wedman', 'Lenny_Wilkens', 'Freeman_Williams']
There are: 21 Players who performed well in the playoffs:
['Billyray_Bates', 'Mookie_Blaylock', 'Wilt_Chamberlain', 'Bob_Cousy', 'Dave_Cowens', 'Steve_Francis', 'Dan_Issel', 'Magic_Johnson', 'Mic
hael_Jordan', 'Jason_Kidd', 'Pete_Maravich', 'George_Mikan', 'Eddie_Miller', 'Calvin_Murphy', 'Bob_Pettit', 'Bill_Russell', 'Dolph_Schaye
s', 'Charlie_Scott', 'Randy_Smith', 'Andrew_Toney', 'Norm_Vanlier']
There are: 7 players who performed well across the all seasons:
['Wilt_Chamberlain', 'Dan_Issel', 'Magic_Johnson', 'Michael_Jordan', 'Jason_Kidd', 'Bob_Pettit', 'Norm_Vanlier']
```

3.2. Game Outcomes Prediction:

Data Reading: Data information was downloaded and saved in txt files, separated by ','. Python was used to read the data into a dictionary and convert it into a DataFrame structure.

Data Cleaning: Exclude missing (NAN) or blank data.

Feature Construction: Currently, we only have useful information: the home team and the score of each game. First, obtain the result of each game from the scores and store it in list HW, and add it as a data column. Then, based on the scores, obtain the point differential (net points) for the home team and the points earned by the home team in each game, where the winning team earns 1 point per win, and the losing team earns 0 points. Data accumulates with games. Also, based on the game date, I can obtain the game day and week number, determining which game week each match belongs to.

Day	h_pts_add	v_pts_add	h_score_add	v_score_add	Week
224	726	640	76	76	33
227	650	89	71	71	33
230	669	70	72	72	33
233	62	677	62	62	34
235	39	700	62	62	34

New data obtained from the last 5 games

We consider features reflecting the team's historical performance in the season itself. We select the results of the team's previous three games as features: hm1, hm2, hm3 represent the win/loss results of the home team's previous three games, with a win recorded as 1 and a loss as 0. vm1, vm2, vm3 respectively reflect the win/loss results of the away team's previous three games, with a

win recorded as 1 and a loss as 0.

hm1	hm2	hm3	vm1	vm2	vm3
0	1	1	1	0	0
1	1	0	1	1	0
1	1	1	0	1	1
0	0	1	1	1	1
0	0	0	1	1	1

Historical win/loss data obtained from the last 5 games

Based on the game week and cumulative point differential, game information is compiled, and the weekly average of the above data is obtained, which better reflects a team's state.

	hPTS_avew	vPTS_avew	hSco_avew	vSco_avew
1307	22.00	19.39	2.30	2.30
1308	19.70	2.70	2.15	2.15
1309	20.27	2.12	2.18	2.18
1310	1.82	19.91	1.82	1.82
1311	1.15	20.59	1.82	1.82

Weekly average scores for the last 5 games

Import relevant information from team-ranking, including each team's ranking, offensive efficiency, defensive efficiency, and other features. We select MOV/A, ORtg/A, DRtg/A, NRtg/A as features of interest and add them to the game information.

	Rk	Team	Conf	Div	W	L	W/L%	MOV	ORtg	DRtg	NRtg	MOV/A	ORtg/A	DRtg/A	NRtg/A
0	1	Houston Rockets	W	SW	65	17	.793	8.48	115.48	106.84	8.64	8.21	115.45	107.08	8.37
1	2	Toronto Raptors	E	A	59	23	.720	7.78	114.60	106.56	8.03	7.30	114.50	106.96	7.54
2	3	Golden State Warriors	W	P	58	24	.707	5.98	114.31	108.43	5.88	5.79	114.33	108.62	5.70
3	4	Utah Jazz	W	NW	48	34	.585	4.30	109.07	104.56	4.51	4.47	108.92	104.24	4.68
4	5	Philadelphia 76ers	E	A	52	30	.634	4.50	110.28	105.77	4.51	4.30	110.12	105.81	4.31

Top five team ranking information

From the initial features, we have constructed many features. Some of these are intermediate features that we need to discard. Because for the first few weeks of games, each team's historical win/loss information is insufficient, so we intend to discard incomplete data. At this point, data feature construction is complete.

```
['hm1', 'hm2', 'hm3', 'vm1', 'vm2', 'vm3', 'hPTS_avew', 'vPTS_avew', 'hSco_avew', 'vSco_avew', 'hMOV', 'hNRtg', 'vMOV', 'vNRtg']
```

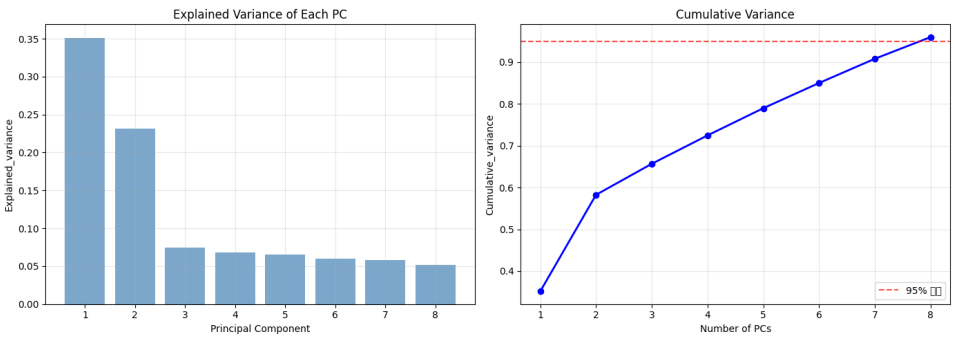
Constructed effective features

Data Processing:

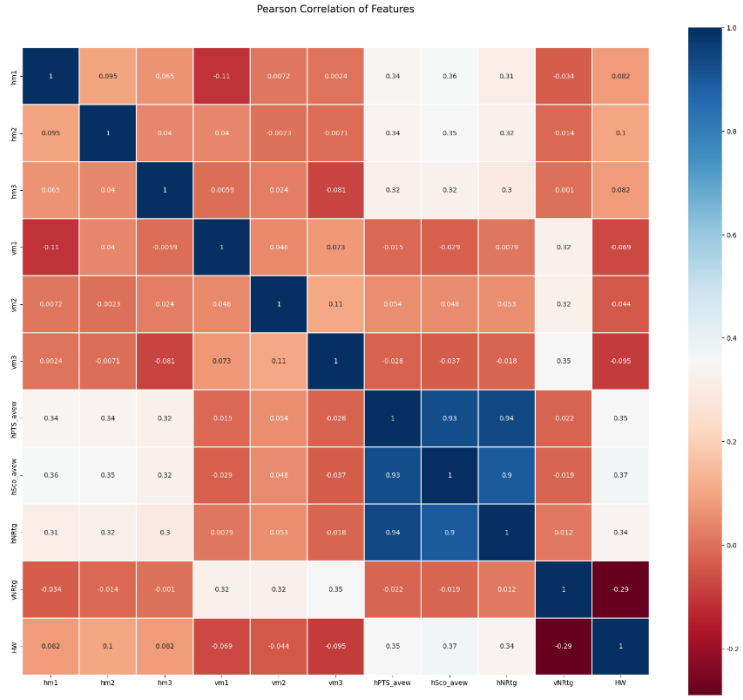
We adopt three feature dimensionality processing methods:

- No dimensionality processing is performed; the original 14 features are used.
- Dimensionality processing is performed using PCA, selecting the first 8 principal components.

  Cumulative explained variance ratio: 0.9598



- Draw a Pearson correlation heatmap, and remove highly correlated features (> 0.9) to reduce redundant dimensions (examples include (vMOV, vNRtg), (hMOV, hNRtg), (hSco_avew, vSco_avew)).



After division, based on the data, convert HW (home win) into training labels, and other data into training features. Split into training set and test set for data testing.

Outputs:

```
Logistic regression learning results:
Accuracy_train:0.703691
Accuracy_test:0.671937
Time cost:  : 0.27s
      hm1  hm2  hm3  vm1  vm2  vm3  hPTS_avew  hSco_avew     hNRtg     vNRtg
1665  0.0  1.0  1.0  1.0  0.0  0.0   0.134636  -0.346248 -0.064793 -1.678828
3546  1.0  1.0  1.0  1.0  0.0  0.0   0.816103   1.714357  0.100571  1.036569
2247  1.0  0.0  0.0  1.0  1.0  0.0  -0.208567  -0.176742 -0.413367 -0.375991
2987  1.0  1.0  0.0  0.0  1.0  1.0   0.647560   0.689627 -0.064793 -1.678828
2166  1.0  0.0  0.0  1.0  0.0  0.0   1.585508   1.553912  1.262882 -1.686657
...   ...  ...  ...  ...  ...  ...        ...        ...       ...       ...
443   0.0  0.0  0.0  1.0  1.0  0.0  -0.160743  -0.002390 -0.047918  0.269133
11    0.0  0.0  0.0  0.0  0.0  0.0  -1.698830  -1.519131 -1.899951 -0.768291
1759  0.0  1.0  1.0  1.0  1.0  0.0  -0.867865  -0.541676 -0.917636  0.019057
1064  0.0  1.0  0.0  0.0  0.0  0.0  -0.075646   0.069798  0.941687 -0.385560
1353  0.0  1.0  0.0  1.0  1.0  0.0   0.177614   0.418926  0.225810  1.093601

[3034 rows x 10 columns]
1665    1.0
3546    1.0
2247    1.0
2987    1.0
2166    0.0
       ...
443     0.0
11      1.0
1759    0.0
1064    0.0
1353    1.0
Name: HW, Length: 3034, dtype: float64
[1. 1. 1. ... 0. 1. 1.]
```

Prediction Results:

- **No dimensionality processing:**

  Accuracy_train:0.699077

  Accuracy_test:0.690382

  Time cost: : 0.28s

- **PCA:**

  raining set accuracy: 0.6889

Test set accuracy: 0.7207
Time Cost： : 0.01s

- **Select features using Pearson correlation heatmap:**

  Accuracy_train:0.690508
  Accuracy_test:0.714097
  Time cost： : 0.03s

Summary:

- Reported prediction accuracy reaches about **70%**.
- Using PCA for feature dimensionality processing offers good runtime efficiency.