Report – Project 1

111550027 鐘正崴

Dataset link

https://github.com/zhweiii/AI_Capstone_HW1

Introduction

In this project, I use AI to **predict which NBA playoff team will win the championship**. For this dataset, I used Python's 'nba_api' and referenced data from Basketball Reference to build it. The dataset includes various statistics from16 playoffs teams, such as regular season performance, number of championships, and number of star players.

For algorithms, I applied two supervised learning methods: SVR (Support Vector Regression) and RF (Random Forest); as well as K-Means, an unsupervised learning method. Notably, both SVR and RF successfully predicted the 2024 NBA champion – Boston Celtics.

Data documentation

- Data description

Feature name	type	description
TeamName	string	name of each playoff team
Conference	string	East / West
PlayoffRank	int	rank in regular season (1 ~ 8)
SEASON	int	years from 1997 to 2024
W_PCT	int	win %
PLUS_MINUS	int	overall point differential in season
FG_PCT	int	Field Goal %
FG3_PCT	int	3 - Point Field Goal %
FT_PCT	int	Free Throw %
AST	int	assist

REB	int	rebound
TOV	int	turn over
STL	int	steal
BLK	int	block
AllStar_count	int	The number of players selected as All-Stars that season (excluding those who missed the All-Star game due to injury).
Finals_Appearances_Last5	int	The number of times a team has reached the Finals in the past five years
ChampionNumber	int	The total number of championships a team has won in its history (including under previous team names).
Label	int	Whether the team won the champion label ratio of 0 : 1 is 420 : 28

- Source
  - nba_api: a Python package that allows you to access NBA statistics and data from NBA.com
  - Basketball reference: a comprehensive basketball statistics website that provides detailed historical and current NBA data
- Dataset size
  - Column size: 18 (details on table above)
  - Row size: 448 (16 teams each year from 1997 to 2024)
    - Process of data collection (execute get_data/getTeamData.py to finish 1~3)
    - Used nba_api to retrieve information (Conference, TeamName, PlayoffRank, SEASON) of the teams ranked 1st to 8th in the regular season from 1997 to 2024. (Due to the play-in tournament, the 9th-seeded Pelicans and Hawks in the 2022 playoffs replaced the original 8th-seeded Clippers and Cavaliers in the Western and Eastern Conferences.)
    - Based on season to collect team statistics with nba_api , including: W_PCT, PLUS_MINUS, FG_PCT, FG3_PCT, FT_PCT, AST, REB, TOV, STL, BLK
    - merge the datasets using team_id
    - Modify SEASON label format (ex. from 2023-24 to 2024)
    - use Basketball Reference to collect or calculate additional data for each team: AllStar_count, ChampionNumber, ChampionNumber, Label

Method

1. **data split**: Use data from 1997 to 2023 for training and 2024 for testing
2. **data resampling**: Since the number of champion teams (label = 1) is significantly smaller compared to non-champions (label = 0) at a ratio of 1:15, resampling is needed to address the data imbalance. To achieve this, I use resample from scikit-learn to balance the dataset (replicate data of label 1), ensuring a more effective training process for the model.
3. **supervised learning**: in this section, considering that there is only one champion each year, I used a regression approach to predict the probability of each team winning the championship. Then, I selected the team with the highest probability from the same season as the predicted champion. For evaluation, I chose to use MSE as the calculation method.

   - K-Fold Cross Validation
     - splits training data into K (set K = 10 in this project) subsets and helps ensure that the model is evaluated on different subsets of the data, providing a more reliable measure of its performance
     - function used: KFold, make_scorer, cross_val_score from scikit-learn
   - MSE (Mean Squared Error)
     - a common metric used to evaluate the performance of regression models. It measures the average squared difference between the predicted and actual values.

$$\text{MSE} = \frac{1}{N}\sum_{i=0}^{N}(y_i - \hat{y}_i)^2$$

     - function used: mean-squared-error from scikit-learn
   - Random Forest Regressor (w/o data resampling)
     - an ensemble method that builds multiple decision trees and combines their predictions to produce a more accurate and robust result.
     - function used: RandomForestRegressor from scikit-learn
     - parameter

hyperparameter name	value
max_depth	10
n_estimators	300
criterion	friedman_mse
max_features	sqrt
random_state	42

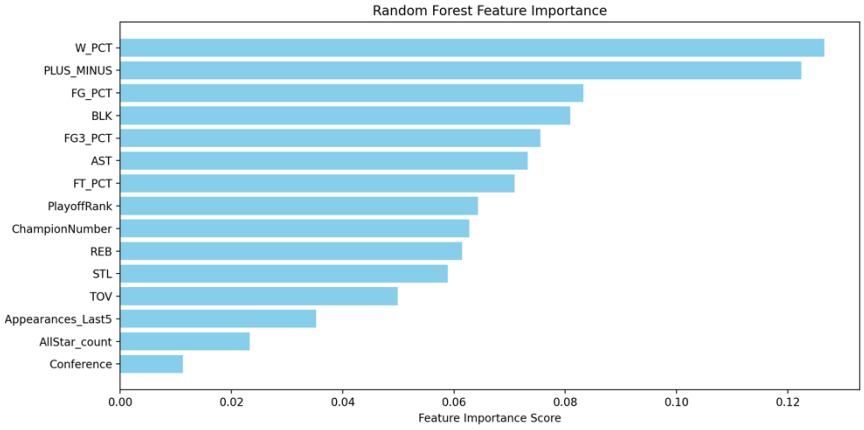
- result (successfully predict Celtics as champion with highest prob)

```
--- Random Forest ---
Highest Champion Prob:
      SEASON TeamName   RF_Prob
432    2024  Celtics  0.394667
```

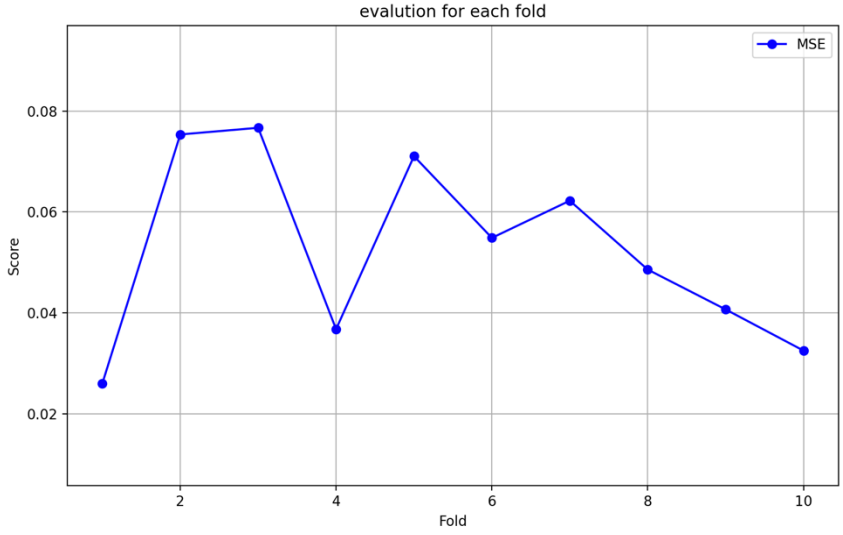
- MSE (test data and avg of cross-validated data)

```
evaluation:
MSE: 0.0397
Cross-validated MSE: 0.0525
```

- feature importance (W_PCT is the most important feature in RF model)



- Cross Validation



- <u>SVR (with data resampling)</u>
  - It is based on the Support Vector Machine (SVM), which is traditionally used for classification. SVR aims to find a function that best fits the data within a margin of tolerance. Also, it is effective for non-linear regression

problems and widely used when the relationship between features and the target is complex.

- function used: SVR, permutation_importance from scikit-learn
- parameters

hyperparameter name	value
kernel	rbf
C	10
epsilon	0.05
gamma	scale

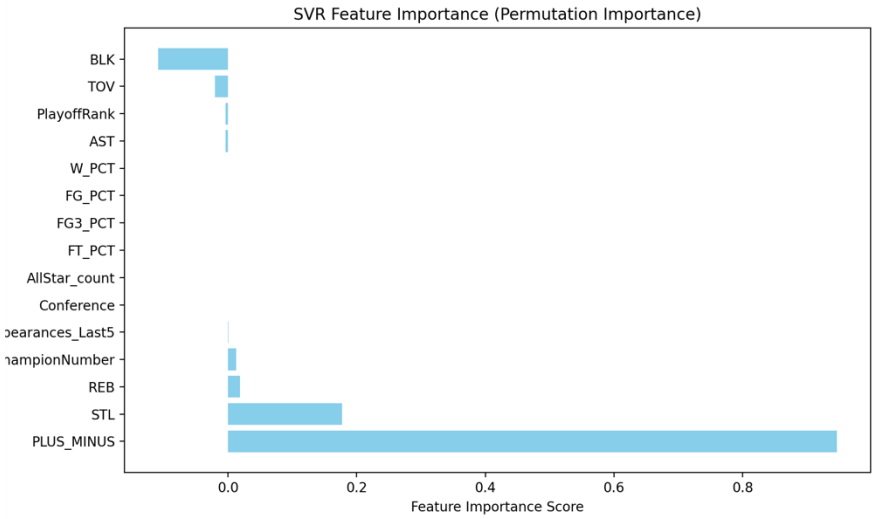
- result (successfully predict Celtics as champion with highest prob)

```
--- SVR ---
Highest Champion Prob:
     SEASON TeamName  SVR_Prob
432    2024  Celtics  0.865341
```

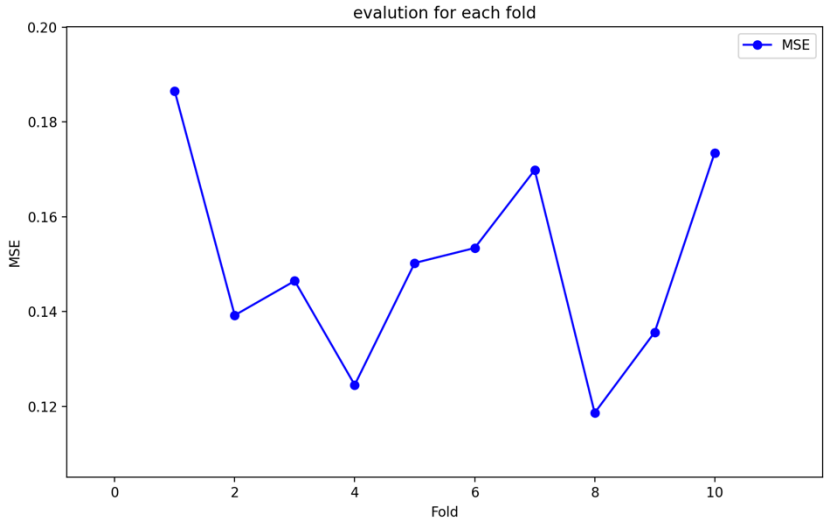
- MSE (test data and avg of cross-validated data)

```
evaluation:
MSE: 0.2139
Cross-validated MSE: 0.1498
```

- permutation importance (PlUS_MINUS is the most important feature in SVR model)



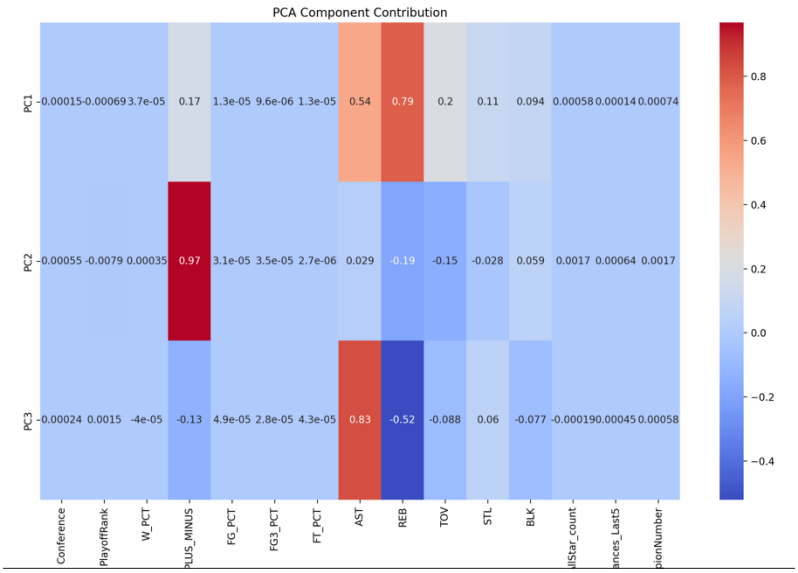
- Cross Validation



4. **unsupervised learning**: In this part, I first use PCA to extract the top 3 most important features, then apply k-Means to group the data into different clusters. Finally, I use these three features as axes to plot the data distribution in 3D space.

- PCA (Principal Component Analysis)
  - a dimensionality reduction technique used to simplify datasets by transforming them into a smaller set of uncorrelated variables called principal components
  - function used: PCA from scikit-learn
  - result

    we can find that **PlUS_MINUS / AST / REB** are three principal components in this dataset

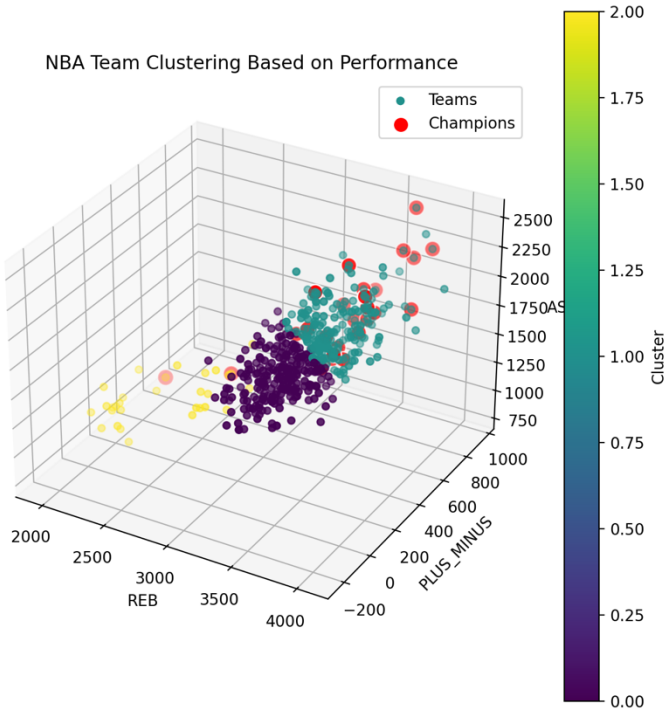


- K-Means
  - groups data points into K distinct clusters based on their features

- function used: KMeans from scikit-learn
- hyperparameter: K = 3
- result

  terrible result since there are many non-champion data points in Cluster 1, even though most of the champions are also included in it

```
--- KMeans Clustering ---
Number of champions in each cluster:
cluster 0 : 2
cluster 1 : 24
cluster 2 : 2
```

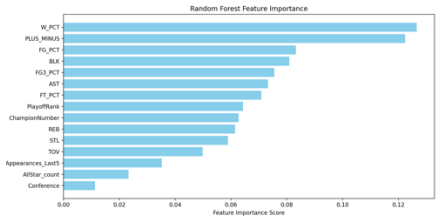
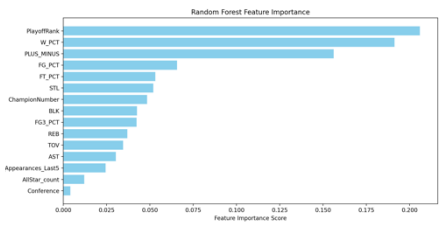
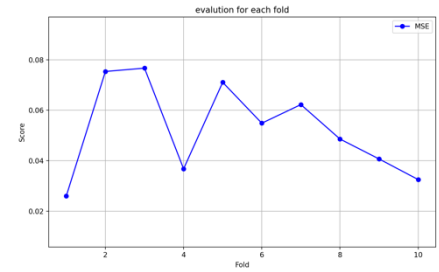
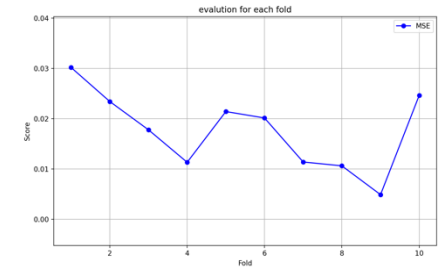


Experiments

1. data without resampling in supervised learning

- RF regressor result

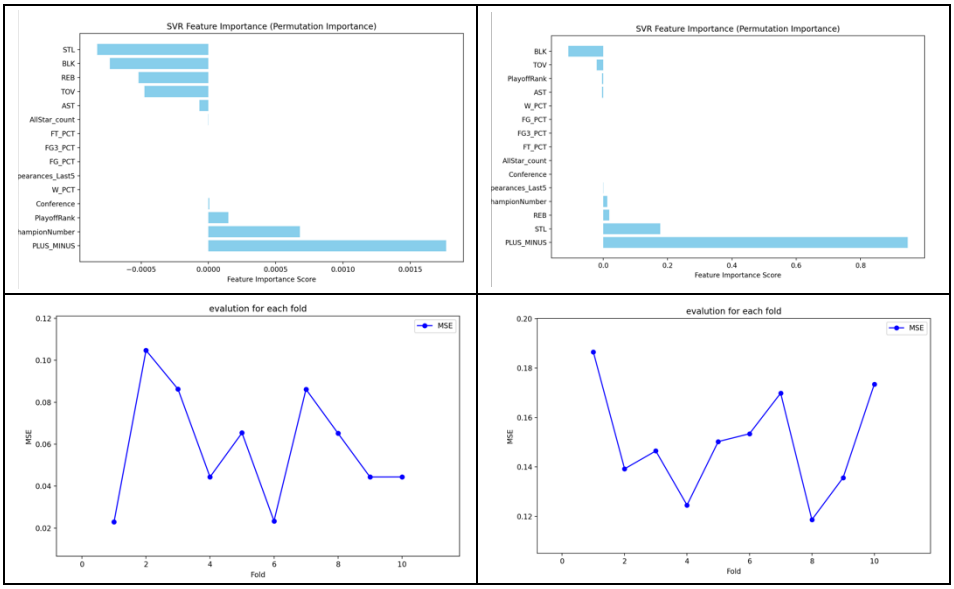
  Compared to the original method without resampling, the probabilities predicted by the Random Forest model tend to be higher and closer to 1. I believe this is because the number of samples labeled as 1 increased, causing the regression values to rise as well. However, the predicted result is incorrect.

w/o resampling	with resampling
--- Random Forest --- Highest Champion Prob: SEASON TeamName RF_Prob 432 2024 Celtics 0.394667	--- Random Forest --- Highest Champion Prob: SEASON TeamName RF_Prob 434 2024 Nuggets 0.605202
evaluation: MSE: 0.0397 Cross-validated MSE: 0.0525	evaluation: MSE: 0.0818 Cross-validated MSE: 0.0176
	
	

- SVR result

  It can be observed that the SVR model without resampling predicts almost all championship probabilities as 0 due to the data imbalance. As a result, even selecting the maximum value has no reference value.

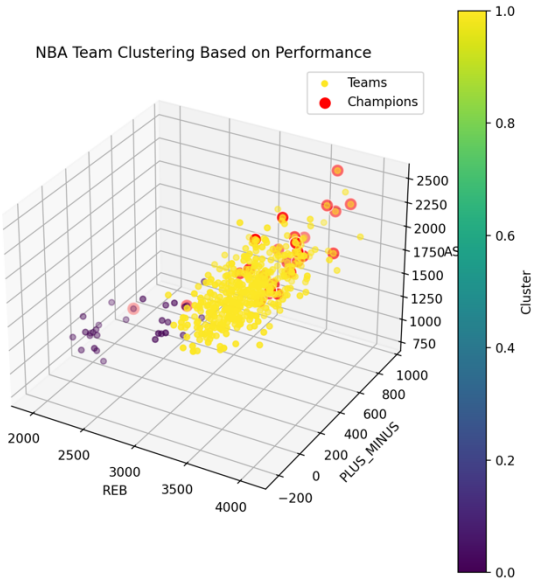
w/o resampling	with resampling
--- SVR --- Highest Champion Prob: SEASON TeamName SVR_Prob 442 2024 Pacers 0.051823	--- SVR --- Highest Champion Prob: SEASON TeamName SVR_Prob 432 2024 Celtics 0.865341
evaluation: MSE: 0.0586 Cross-validated MSE: 0.0587	evaluation: MSE: 0.2139 Cross-validated MSE: 0.1498

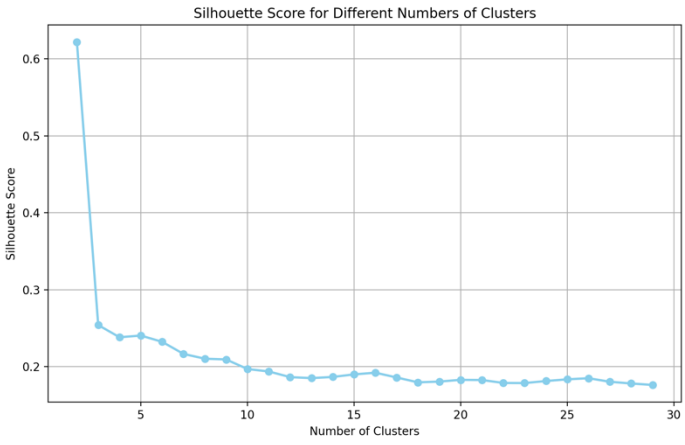


2. K-Means with different K

- Using silhouette_scores to evaluate the degree of separation in the data for different values of k, observe which k provides the best separation.
- Result

  It can be observed that the best separation occurs when k = 2, but after visualization, Cluster 1 still contains too much non-champion teams. I think it is due to the imbalanced data





Discussion

1. data with / without resampling

   Before doing this experiment, I already knew that using unbalanced data without resampling would lead to poor prediction results, but I didn't expect it to be this bad for SVR, which highlights the importance of data balancing. However, since my initial RF Regressor was trained using data without resampling, the predictions aligned with expectations after tuning the parameters, but the probabilities were relatively low. On the other hand, when using resampling, the predicted probabilities increased, yet the prediction results were not ideal. I believe this is the complexity of AI—every change in parameters, training data, and other factors requires careful fine-tuning

2. K-Means with different K

   From this experiment, I believe I should try other unsupervised learning methods. It is possible that my data is too imbalanced or that the features are not distinctive enough, making it difficult for K-Means to effectively cluster the data, regardless of the chosen value of k

3. Future experiments

   Add more features that could potentially impact the results, try different unsupervised learning methods, and fine-tune the RF model trained with resampling

References

- https://www.basketball-reference.com
- https://github.com/swar/nba_api/tree/master/src/nba_api
- https://scikit-learn.org/stable/modules/generated/sklearn.svm.SVR.html
- https://scikit-learn.org/stable/modules/generated/sklearn.ensemble.RandomForestRegressor.html
- https://scikit-learn.org/stable/modules/generated/sklearn.cluster.KMeans.html
- https://scikit-learn.org/stable/modules/generated/sklearn.decomposition.PCA.html

Program Code

Prediction.py

```
# Load dataset
df = pd.read_csv("data/data.csv")
df["Conference"] = df["Conference"].map({"East": 0, "West": 1})

# Split data into train and test
train_data = df[df["SEASON"] < 2024]
test_data = df[df["SEASON"] >= 2024]

X = df.drop(columns=["Label", "SEASON", "TeamName"])
X_train = train_data.drop(columns=["Label", "SEASON", "TeamName"])
y_train = train_data["Label"]
X_test = test_data.drop(columns=["Label", "SEASON", "TeamName"])
y_test = test_data["Label"]

df_majority = train_data[train_data["Label"] == 0]
df_minority = train_data[train_data["Label"] == 1]

df_minority_upsampled = resample(df_minority,
                                 replace=True,
                                 n_samples=len(df_majority),
                                 random_state=42)

train_data_balanced = pd.concat([df_majority, df_minority_upsampled])

X_train_balanced = train_data_balanced.drop(columns=["Label", "SEASON", "TeamName"])
y_train_balanced = train_data_balanced["Label"]
```

```
# Random Forest Regressor
rf_model = RandomForestRegressor(max_depth=10, n_estimators=300, criterion='friedman_mse',
                                 max_features='sqrt', random_state=42)
rf_model.fit(X_train, y_train)
rf_probs = rf_model.predict(X_test)

# Add predicted probabilities to test data
test_data = test_data.copy()
test_data.loc[:, "RF_Prob"] = rf_probs
rf_champions = test_data.loc[test_data.groupby("SEASON")["RF_Prob"].idxmax()]
print("--- Random Forest ---")
print("Highest Champion Prob:")
print(rf_champions[["SEASON", "TeamName", "RF_Prob"]])

# evaluation
mse = mean_squared_error(y_test, rf_probs)
```

```python
# # Cross-validation
split = 10
kf = KFold(n_splits=split, shuffle=True, random_state=42)
mse_scorer = make_scorer(mean_squared_error, greater_is_better=False)
mse_cv_scores = cross_val_score(rf_model, X_train, y_train, cv=kf, scoring=mse_scorer)

# Plot cross-validated MSE
plt.figure(figsize=(10, 6))
plt.plot(range(1, split + 1), -mse_cv_scores, marker='o', linestyle='-', color='b', label='MSE')
plt.legend()
plt.xlabel('Fold')
plt.ylabel('Score')
plt.margins(y=0.4)
plt.title('evalution for each fold')
plt.grid(True)
plt.show()
```

```python
# Feature Importance
feature_importance = rf_model.feature_importances_
feature_names = X_train.columns
importance_df = pd.DataFrame({'Feature': feature_names, 'Importance': feature_importance})
importance_df = importance_df.sort_values(by='Importance', ascending=False)

plt.figure(figsize=(12, 6))
plt.barh(importance_df["Feature"], importance_df["Importance"], color="skyblue")
plt.xlabel("Feature Importance Score")
plt.ylabel("Features")
plt.title("Random Forest Feature Importance")
plt.gca().invert_yaxis()
plt.show()
```

```python
# SVR
svr_model = SVR(kernel='rbf', C=10.0, epsilon=0.05, gamma='scale')
svr_model.fit(X_train_balanced, y_train_balanced)
svr_probs = svr_model.predict(X_test)
svr_probs = np.tanh(svr_probs)

# print("svr_probs:", svr_probs)

# Add predicted probabilities to test data
test_data = test_data.copy()
test_data.loc[:, "SVR_Prob"] = svr_probs
rf_champions = test_data.loc[test_data.groupby("SEASON")["SVR_Prob"].idxmax()]
print("--- SVR ---")
print("Highest Champion Prob:")
print(rf_champions[["SEASON", "TeamName", "SVR_Prob"]])
```

```python
# Permutation Importance
result = permutation_importance(svr_model, X_test, y_test, n_repeats=10, random_state=42)
sorted_idx = result.importances_mean.argsort()

# Plot feature importance
plt.figure(figsize=(10, 6))
plt.barh(np.array(X_train_balanced.columns)[sorted_idx], result.importances_mean[sorted_idx], color="skyblue")
plt.xlabel("Feature Importance Score")
plt.ylabel("Features")
plt.title("SVR Feature Importance (Permutation Importance)")
plt.gca().invert_yaxis()
plt.show()
```

```python
# PCA
pca = PCA(n_components=3)
X_pca = pca.fit_transform(X)

# visualize PCA
plt.figure(figsize=(10, 6))
sns.heatmap(pca.components_, annot=True, cmap="coolwarm", xticklabels=X.columns, yticklabels=["PC1", "PC2", "PC3"])
plt.title("PCA Component Contribution")
plt.show()
```

```python
# KMeans Clustering
kmeans = KMeans(n_clusters=3, random_state=42)
df["Cluster"] = kmeans.fit_predict(X)

# check which cluster has the highest number of champions
print("--- KMeans Clustering ---")
print("Number of champions in each cluster:")
result = df.groupby("Cluster")["Label"].sum()
print(f"cluster 0 : {result[0]}")
print(f"cluster 1 : {result[1]}")
print(f"cluster 2 : {result[2]}")

# Plot result
feature1 = "REB"
feature2 = "PLUS_MINUS"
feature3 = "AST"

# Mark the points where label == 1
champions = df[df["Label"] == 1]

# Assuming df and champions are already defined
plt.figure(figsize=(8, 8))
ax = plt.axes(projection='3d')
scatter = ax.scatter(df[feature1], df[feature2], df[feature3], c=df["Cluster"], cmap="viridis", label="Teams")
ax.scatter(champions[feature1], champions[feature2], champions[feature3], c='red', label="Champions", edgecolors='w', s=100)
ax.set_xlabel(feature1)
ax.set_ylabel(feature2)
ax.set_zlabel(feature3)
plt.title("NBA Team Clustering Based on Performance")
plt.colorbar(scatter, label="Cluster")
plt.legend()
plt.show()
```