Milestone 2: Feature Engineering, Feature Selection, and Data Modeling Timeline

By: Kuan-Chen, Chen

1. Previous of Milestone2

For this milestone, our objective is to develop machine learning models that predict potential earnings across various sports based on fundamental user attributes such as height, weight, age, and playing position. We plan to implement several algorithms, including random forests, gradient boosting, and neural networks, comparing their performance to identify the most effective approach for this prediction task.

Since directly measuring athletic potential or performance level is challenging, we will use salary as a proxy measure for success. This approach is justified by the strong correlation between higher salaries and superior athletic performance in professional sports. Extensive research in sports economics has consistently demonstrated that salaries generally reflect an athlete's contribution to team success, particularly in leagues with competitive markets and salary caps.

Our analysis will focus on major professional sports including basketball (NBA), baseball (MLB), football (NFL), soccer (various leagues), and tennis, allowing us to compare how physical attributes translate to financial success across different sporting contexts. We will evaluate model performance using metrics such as Mean Absolute Error (MAE) and Root Mean Squared Error (RMSE) to ensure accurate salary predictions.

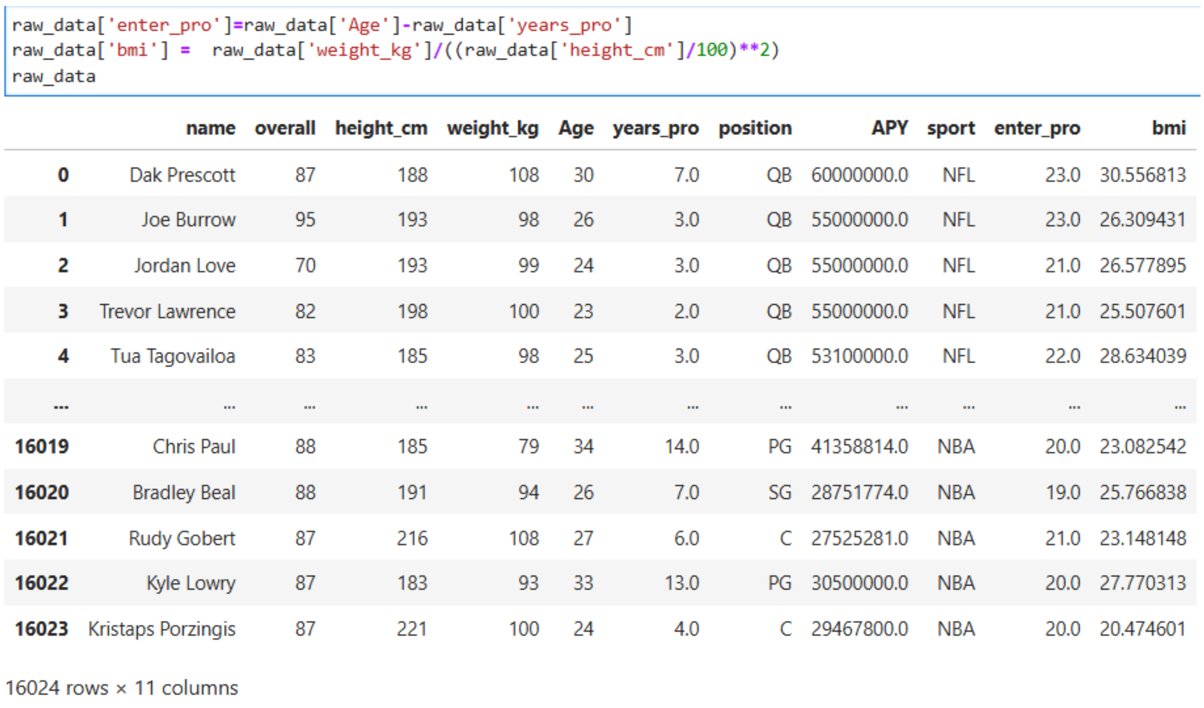
Therefore, our models will predict the potential salary a user could earn in different sports as the target variable. This information can be valuable not only for aspiring athletes deciding which sport to pursue but also for talent scouts, sports analysts, and team management professionals looking to identify potentially undervalued players.

2. Feature Engineering

Since we are using the same basic features (height, weight, age, and position) across different sports, we recognized the need to enhance our model with additional predictive variables. To improve model performance, we incorporated two supplementary features: BMI (Body Mass Index) and enter_pro (age at which an athlete entered professional leagues).

The enter_pro feature provides valuable insight because athletes who join professional leagues at a younger age typically demonstrate exceptional talent. This early professional entry often indicates superior skill levels that correlate with higher earning potential throughout their careers.

Similarly, BMI serves as an important discriminating factor since optimal body composition varies significantly across different sports. For example, the ideal BMI for a basketball player differs substantially from that of a gymnast or a football lineman, making this metric particularly useful for sport-specific salary predictions.



```
raw_data['enter_pro']=raw_data['Age']-raw_data['years_pro']
raw_data['bmi'] =  raw_data['weight_kg']/((raw_data['height_cm']/100)**2)
raw_data
```

	name	overall	height_cm	weight_kg	Age	years_pro	position	APY	sport	enter_pro	bmi
0	Dak Prescott	87	188	108	30	7.0	QB	60000000.0	NFL	23.0	30.556813
1	Joe Burrow	95	193	98	26	3.0	QB	55000000.0	NFL	23.0	26.309431
2	Jordan Love	70	193	99	24	3.0	QB	55000000.0	NFL	21.0	26.577895
3	Trevor Lawrence	82	198	100	23	2.0	QB	55000000.0	NFL	21.0	25.507601
4	Tua Tagovailoa	83	185	98	25	3.0	QB	53100000.0	NFL	22.0	28.634039
...	...	...	...	...	...	...	...	...	...	...	...
16019	Chris Paul	88	185	79	34	14.0	PG	41358814.0	NBA	20.0	23.082542
16020	Bradley Beal	88	191	94	26	7.0	SG	28751774.0	NBA	19.0	25.766838
16021	Rudy Gobert	87	216	108	27	6.0	C	27525281.0	NBA	21.0	23.148148
16022	Kyle Lowry	87	183	93	33	13.0	PG	30500000.0	NBA	20.0	27.770313
16023	Kristaps Porzingis	87	221	100	24	4.0	C	29467800.0	NBA	20.0	20.474601

16024 rows × 11 columns

3. Feature Selection

A. Data Preprocessing and Feature Engineering

Before training our models, we implemented several crucial preprocessing steps to prepare our categorical position data for machine learning algorithms.

Since different sports use distinct position classification systems, we needed a standardized approach to encode this information effectively.

B. Position Encoding

We explored multiple encoding strategies for the position variables. Initially, we tested one-hot encoding, which creates binary features for each position category. However, our experiments showed that this approach did not significantly improve model performance despite increasing feature dimensionality. Therefore, we opted for a more efficient label encoding method using custom mapping dictionaries.

For NBA and NFL data, we created specialized mapping systems that consolidate similar positions into functional groups. This approach was particularly valuable for basketball data, where there is considerable overlap between certain positions. For example, point guards (PG) and shooting guards (SG) share many similar attributes, with many players capable of performing both roles. By grouping these positions together, we could test whether this domain-specific assumption improved prediction accuracy.

For FIFA data, we took a different approach due to the extensive variety of soccer positions (exceeding 30 distinct classifications). Rather than creating an overly complex encoding scheme, we simplified by assigning all FIFA positions a single label value of 1. This decision was based on our assessment that the other physical attributes would provide more predictive power than the highly granular position classifications in soccer.

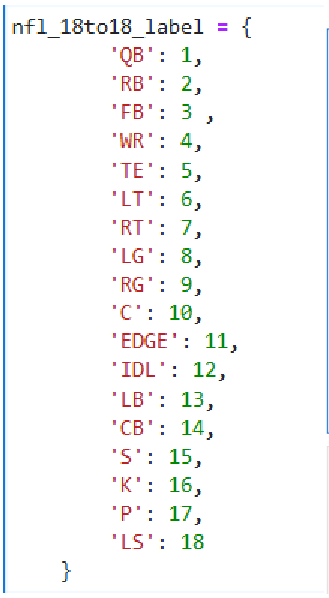
C. Dataset Partitioning

After completing the position encoding process, we split our data into five distinct datasets for separate model training. This partitioning was necessary because the FIFA dataset lacked information regarding players' professional experience duration, making the "years played" feature unavailable for these athletes

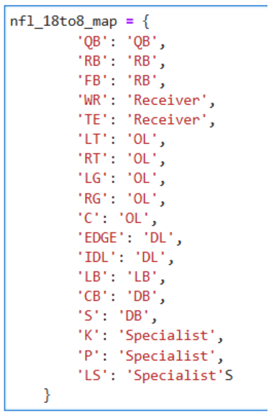
.

D. Position Mapping Dictionaries

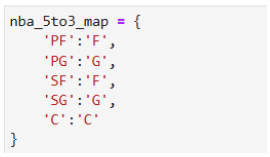
This position consolidation approach allowed us to reduce dimensionality while maintaining the essential functional distinctions between player roles, potentially improving our model's ability to generalize across similar positions.



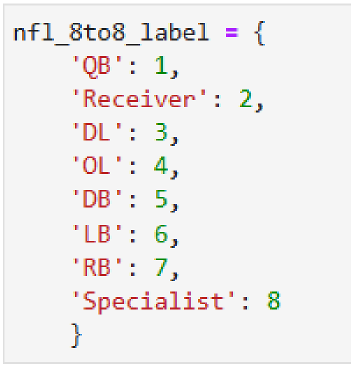
```
nfl_18to18_label = {
            'QB': 1,
            'RB': 2,
            'FB': 3 ,
            'WR': 4,
            'TE': 5,
            'LT': 6,
            'RT': 7,
            'LG': 8,
            'RG': 9,
            'C': 10,
            'EDGE': 11,
            'IDL': 12,
            'LB': 13,
            'CB': 14,
            'S': 15,
            'K': 16,
            'P': 17,
            'LS': 18
      }
```



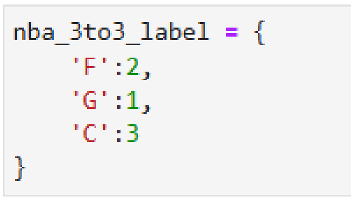
```
nfl_18to8_map = {
            'QB': 'QB',
            'RB': 'RB',
            'FB': 'RB',
            'WR': 'Receiver',
            'TE': 'Receiver',
            'LT': 'OL',
            'RT': 'OL',
            'LG': 'OL',
            'RG': 'OL',
            'C': 'OL',
            'EDGE': 'DL',
            'IDL': 'DL',
            'LB': 'LB',
            'CB': 'DB',
            'S': 'DB',
            'K': 'Specialist',
            'P': 'Specialist',
            'LS': 'Specialist'S
      }
```



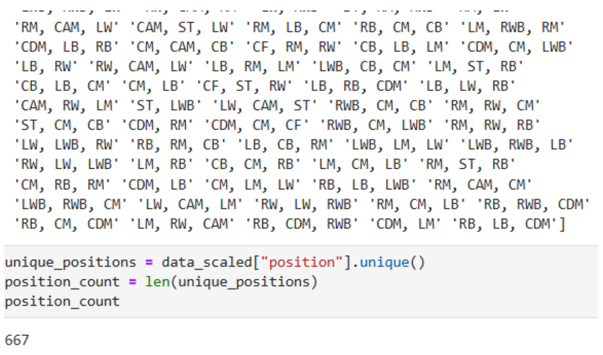
```
nba_5to3_map = {
     'PF':'F',
     'PG':'G',
     'SF':'F',
     'SG':'G',
     'C':'C'
}
```



```
nfl_8to8_label = {
    'QB': 1,
    'Receiver': 2,
    'DL': 3,
    'OL': 4,
    'DB': 5,
    'LB': 6,
    'RB': 7,
    'Specialist': 8
    }
```



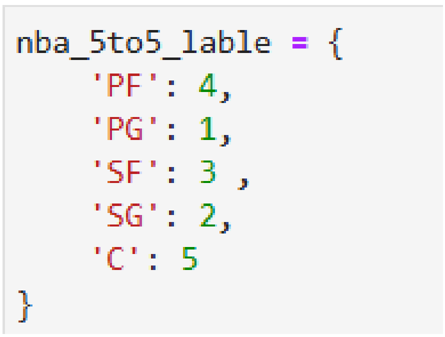
```
nba_3to3_label = {
    'F':2,
    'G':1,
    'C':3
}
```



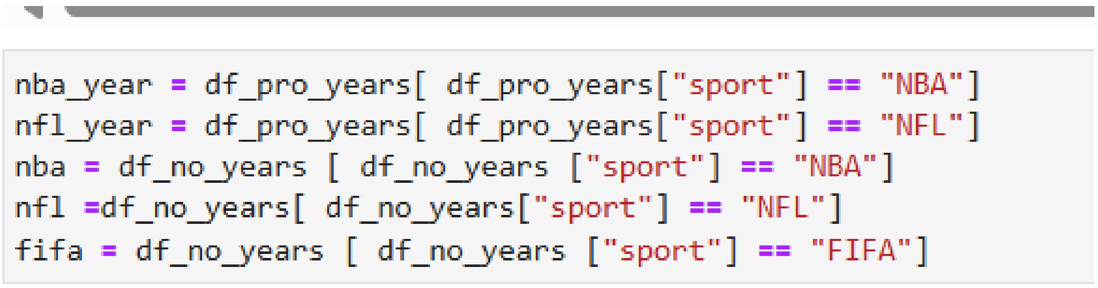
```
'RM, CAM, LW' 'CAM, ST, LW' 'RM, LB, CM' 'RB, CM, CB' 'LM, RWB, RM'
'CDM, LB, RB' 'CM, CAM, CB' 'CF, RM, RW' 'CB, LB, LM' 'CDM, CM, LWB'
'LB, RW' 'RW, CAM, LW' 'LB, RM, LM' 'LWB, CB, CM' 'LM, ST, RB'
'CB, LB, CM' 'CM, LB' 'CF, ST, RW' 'LB, RB, CDM' 'LB, LW, RB'
'CAM, RW, LM' 'ST, LWB' 'LW, CAM, ST' 'RWB, CM, CB' 'RM, RW, CM'
'ST, CM, CB' 'CDM, RM' 'CDM, CM, CF' 'RWB, CM, LWB' 'RM, RW, RB'
'LW, LWB, RW' 'RB, RM, CB' 'LB, CB, RM' 'LWB, LM, LW' 'LWB, RWB, LB'
'RW, LW, LWB' 'LM, RB' 'CB, CM, RB' 'LM, CM, LB' 'RM, ST, RB'
'CM, RB, RM' 'CDM, LB' 'CM, LM, LW' 'RB, LB, LWB' 'RM, CAM, CM'
'LWB, RWB, CM' 'LW, CAM, LM' 'RW, LW, RWB' 'RM, CM, LB' 'RB, RWB, CDM'
'RB, CM, CDM' 'LM, RW, CAM' 'RB, CDM, RWB' 'CDM, LM' 'RB, LB, CDM']
```

```
unique_positions = data_scaled["position"].unique()
position_count = len(unique_positions)
position_count
```

```
667
```



```
nba_5to5_lable = {
     'PF': 4,
     'PG': 1,
     'SF': 3 ,
     'SG': 2,
     'C': 5
}
```



```
nba_year = df_pro_years[ df_pro_years["sport"] == "NBA"]
nfl_year = df_pro_years[ df_pro_years["sport"] == "NFL"]
nba = df_no_years [ df_no_years ["sport"] == "NBA"]
nfl =df_no_years[ df_no_years["sport"] == "NFL"]
fifa = df_no_years [ df_no_years ["sport"] == "FIFA"]
```

4. EDA

To systematically evaluate feature importance and identify relationships between variables, we implemented a specialized correlation analysis function. This tool plays a crucial role in our feature selection process and helps guide our modeling decisions.

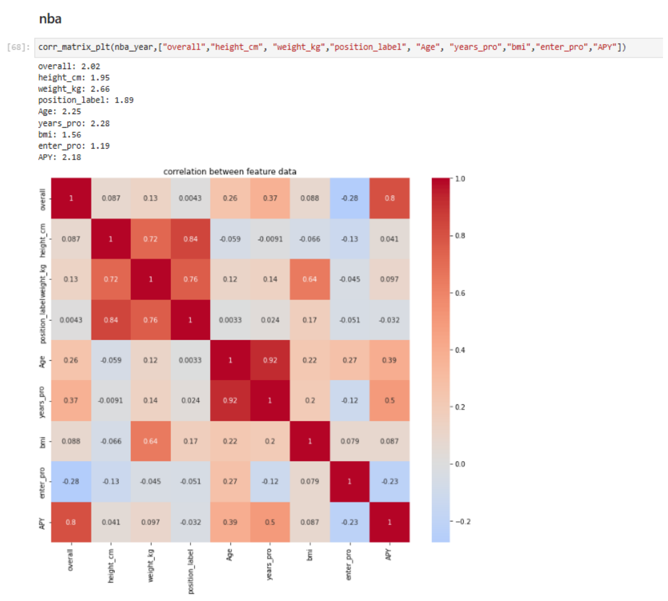
A. Correlation Calculation

The function computes the Pearson correlation coefficients between all features in the dataset and the specified target feature (in our case, salary). This helps us quantify the linear relationship between each predictor variable and the target.

B. Visualization Generation:

The function automatically produces a heatmap visualization of the correlation matrix, allowing for intuitive interpretation of complex relationships between multiple variables. The color intensity in the heatmap corresponds to correlation strength, with darker colors indicating stronger relationships.

For each feature in the dataset, the function also calculates the sum of its correlation coefficients with all other features. This aggregated metric helps us identify which variables have the strongest overall relationship with the entire feature set, potentially highlighting central predictors in our model.


nba

```
[68]: corr_matrix_plt(nba_year,["overall","height_cm", "weight_kg","position_label", "Age", "years_pro","bmi","enter_pro","APY"])
```

```
overall: 2.02
height_cm: 1.95
weight_kg: 2.66
position_label: 1.89
Age: 2.25
years_pro: 2.28
bmi: 1.56
enter_pro: 1.19
APY: 2.18
```

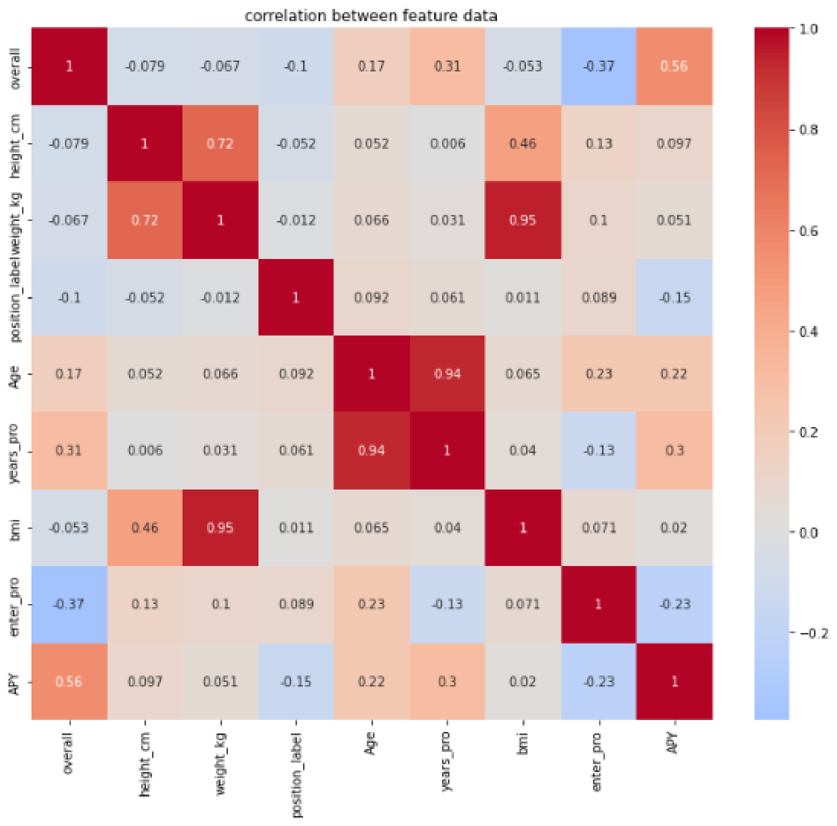
correlation between feature data

	overall	height_cm	weight_kg	position_label	Age	years_pro	bmi	enter_pro	APY
overall	1	0.087	0.13	0.0043	0.26	0.37	0.088	-0.28	0.8
height_cm	0.087	1	0.72	0.84	-0.059	-0.0091	-0.066	-0.13	0.041
weight_kg	0.13	0.72	1	0.76	0.12	0.14	0.64	-0.045	0.097
position_label	0.0043	0.84	0.76	1	0.0033	0.024	0.17	-0.051	-0.032
Age	0.26	-0.059	0.12	0.0033	1	0.92	0.22	0.27	0.39
years_pro	0.37	-0.0091	0.14	0.024	0.92	1	0.2	-0.12	0.5
bmi	0.088	-0.066	0.64	0.17	0.22	0.2	1	0.079	0.087
enter_pro	-0.28	-0.13	-0.045	-0.051	0.27	-0.12	0.079	1	-0.23
APY	0.8	0.041	0.097	-0.032	0.39	0.5	0.087	-0.23	1

nfl

```
[67]: corr_matrix_plt(nfl_year,["overall","height_cm", "weight_kg","position_label", "Age", "years_pro","bmi","enter_pro","APY"])
```

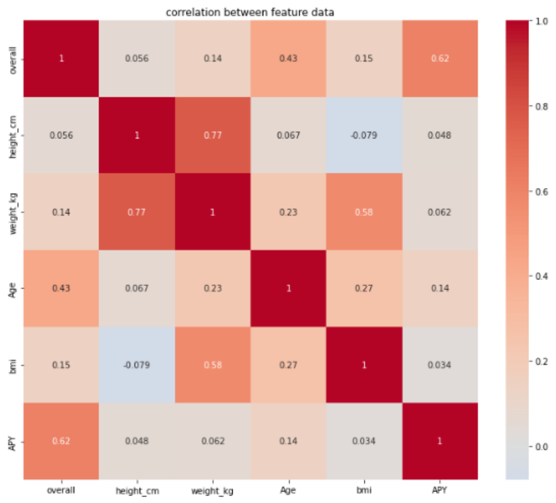
```
overall: 1.71
height_cm: 1.59
weight_kg: 1.99
position_label: 0.57
Age: 1.82
years_pro: 1.82
bmi: 1.67
enter_pro: 1.35
APY: 1.62
```



fifa

```
[66]: corr_matrix_plt(fifa,["overall","height_cm", "weight_kg", "Age","bmi","APY"])
```

```
overall: 1.39
height_cm: 1.02
weight_kg: 1.78
Age: 1.13
bmi: 1.10
APY: 0.90
```



5. Data Modeling

W To effectively predict salary across different sports contexts, we developed a comprehensive modeling strategy that leverages both linear and non-linear approaches. This dual methodology allows us to capture both straightforward linear relationships and complex non-linear interactions within our feature space.

Model Selection and Implementation

We implemented five distinct predictive models to compare their performance characteristics:

Linear Models (3): We deployed three different linear regression variants to establish baseline performance and capture fundamental linear relationships between physical attributes and overall:

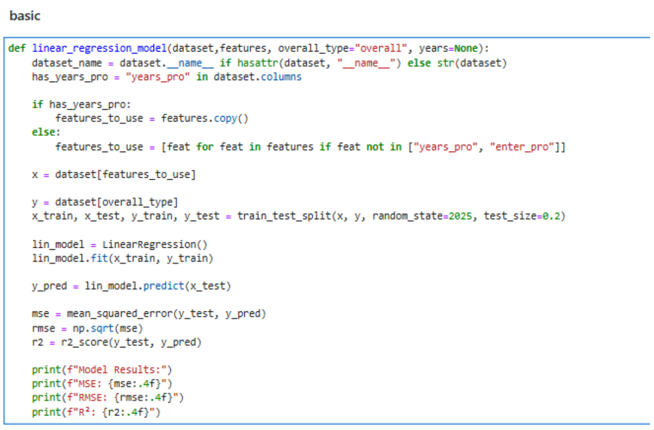
Standard Linear Regression, Ridge Regression and Lasso Regression

Non-Linear Models (2): To capture more complex patterns and interactions, we implemented two tree-based ensemble methods:

Random Forest Regressor and Gradient Boosting Regressor

This diverse model portfolio allows us to evaluate which algorithmic approaches best capture the relationship between physical attributes and earning potential across different sports contexts. For each model, we implemented GridSearchCV to systematically explore the hyperparameter space and identify optimal configurations.

A. linear_regression_model

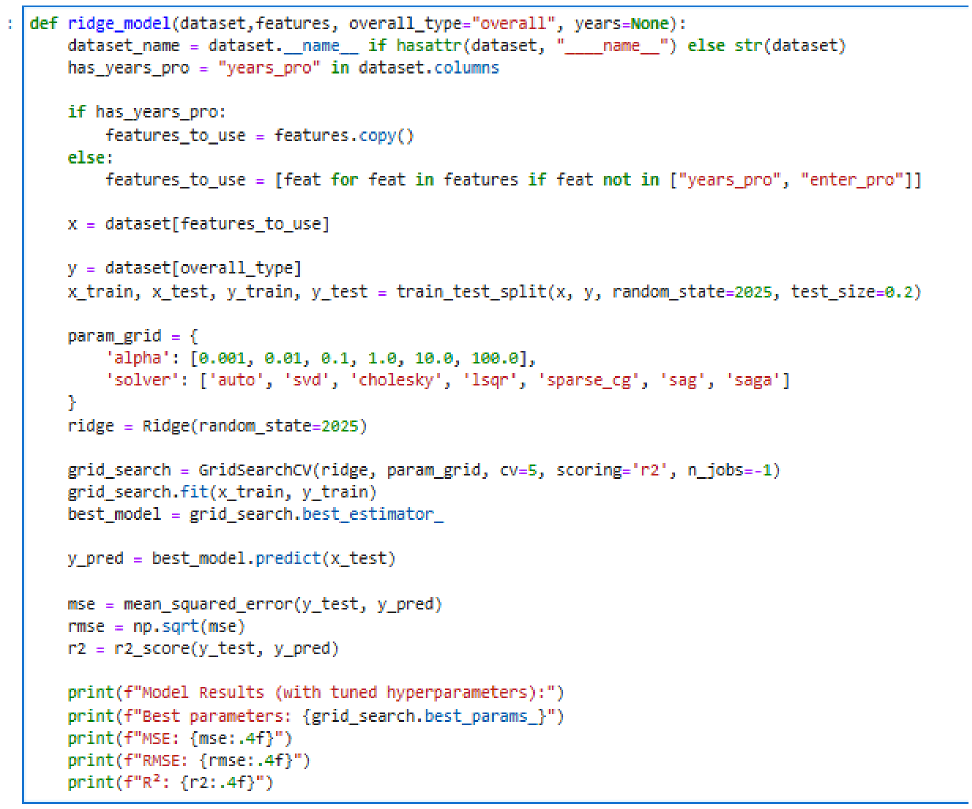


basic

```python
def linear_regression_model(dataset,features, overall_type="overall", years=None):
    dataset_name = dataset.__name__ if hasattr(dataset, "__name__") else str(dataset)
    has_years_pro = "years_pro" in dataset.columns

    if has_years_pro:
        features_to_use = features.copy()
    else:
        features_to_use = [feat for feat in features if feat not in ["years_pro", "enter_pro"]]

    x = dataset[features_to_use]

    y = dataset[overall_type]
    x_train, x_test, y_train, y_test = train_test_split(x, y, random_state=2025, test_size=0.2)

    lin_model = LinearRegression()
    lin_model.fit(x_train, y_train)

    y_pred = lin_model.predict(x_test)

    mse = mean_squared_error(y_test, y_pred)
    rmse = np.sqrt(mse)
    r2 = r2_score(y_test, y_pred)

    print(f"Model Results:")
    print(f"MSE: {mse:.4f}")
    print(f"RMSE: {rmse:.4f}")
    print(f"R²: {r2:.4f}")
```

B. ridge_model

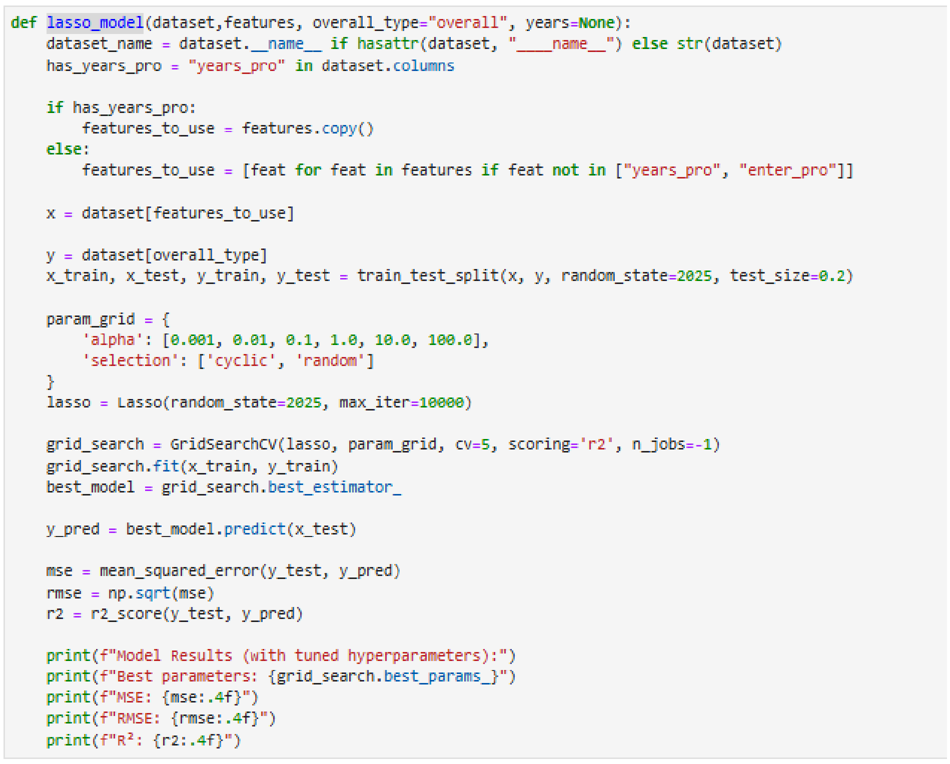
ridge



```
def ridge_model(dataset,features, overall_type="overall", years=None):
    dataset_name = dataset.__name__ if hasattr(dataset, "____name__") else str(dataset)
    has_years_pro = "years_pro" in dataset.columns

    if has_years_pro:
        features_to_use = features.copy()
    else:
        features_to_use = [feat for feat in features if feat not in ["years_pro", "enter_pro"]]

    x = dataset[features_to_use]

    y = dataset[overall_type]
    x_train, x_test, y_train, y_test = train_test_split(x, y, random_state=2025, test_size=0.2)

    param_grid = {
        'alpha': [0.001, 0.01, 0.1, 1.0, 10.0, 100.0],
        'solver': ['auto', 'svd', 'cholesky', 'lsqr', 'sparse_cg', 'sag', 'saga']
    }
    ridge = Ridge(random_state=2025)

    grid_search = GridSearchCV(ridge, param_grid, cv=5, scoring='r2', n_jobs=-1)
    grid_search.fit(x_train, y_train)
    best_model = grid_search.best_estimator_

    y_pred = best_model.predict(x_test)

    mse = mean_squared_error(y_test, y_pred)
    rmse = np.sqrt(mse)
    r2 = r2_score(y_test, y_pred)

    print(f"Model Results (with tuned hyperparameters):")
    print(f"Best parameters: {grid_search.best_params_}")
    print(f"MSE: {mse:.4f}")
    print(f"RMSE: {rmse:.4f}")
    print(f"R²: {r2:.4f}")
```

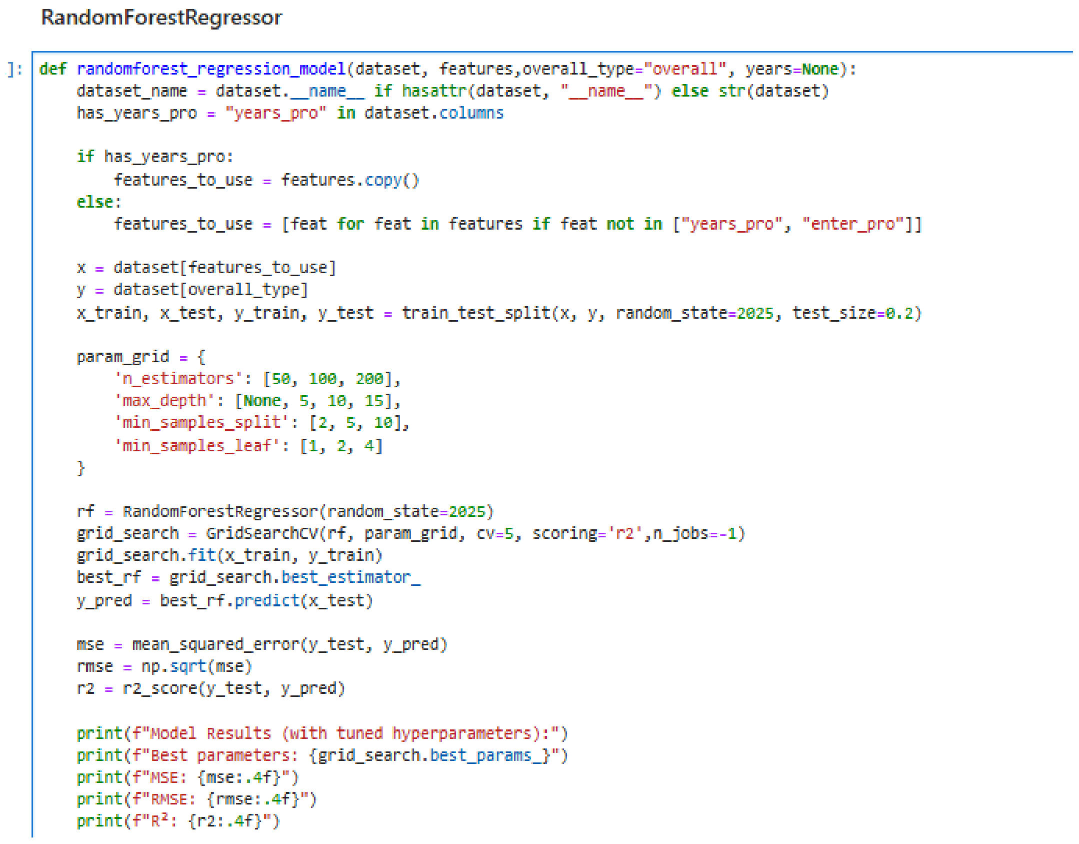
C. lasso_model

lasso



```
def lasso_model(dataset,features, overall_type="overall", years=None):
    dataset_name = dataset.__name__ if hasattr(dataset, "____name__") else str(dataset)
    has_years_pro = "years_pro" in dataset.columns

    if has_years_pro:
        features_to_use = features.copy()
    else:
        features_to_use = [feat for feat in features if feat not in ["years_pro", "enter_pro"]]

    x = dataset[features_to_use]

    y = dataset[overall_type]
    x_train, x_test, y_train, y_test = train_test_split(x, y, random_state=2025, test_size=0.2)

    param_grid = {
        'alpha': [0.001, 0.01, 0.1, 1.0, 10.0, 100.0],
        'selection': ['cyclic', 'random']
    }
    lasso = Lasso(random_state=2025, max_iter=10000)

    grid_search = GridSearchCV(lasso, param_grid, cv=5, scoring='r2', n_jobs=-1)
    grid_search.fit(x_train, y_train)
    best_model = grid_search.best_estimator_

    y_pred = best_model.predict(x_test)

    mse = mean_squared_error(y_test, y_pred)
    rmse = np.sqrt(mse)
    r2 = r2_score(y_test, y_pred)

    print(f"Model Results (with tuned hyperparameters):")
    print(f"Best parameters: {grid_search.best_params_}")
    print(f"MSE: {mse:.4f}")
    print(f"RMSE: {rmse:.4f}")
    print(f"R²: {r2:.4f}")
```

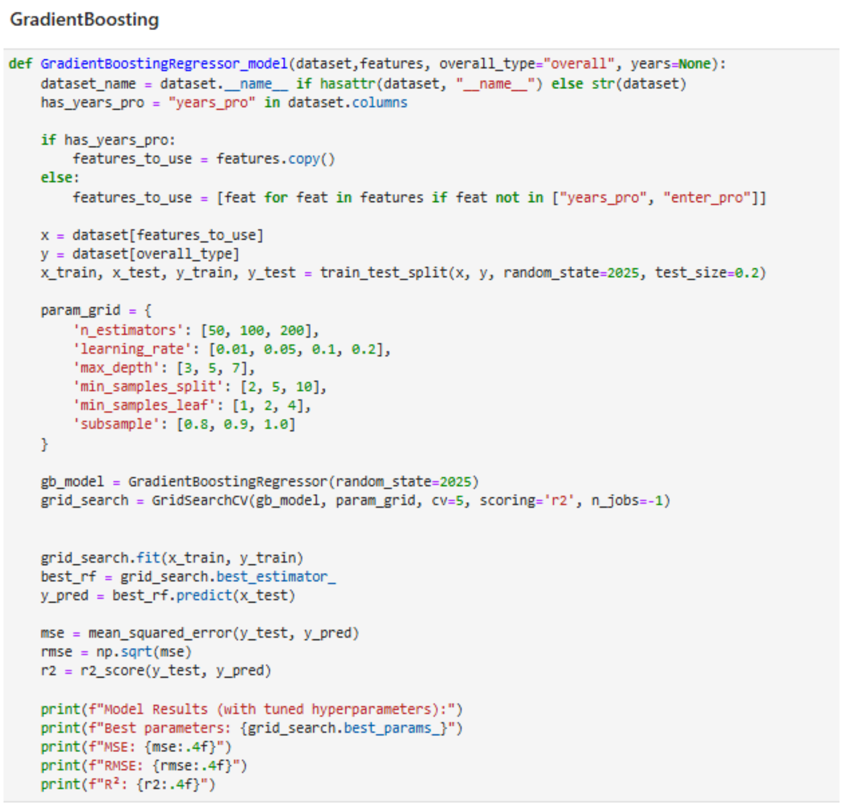
D. RandomForestRegressor



RandomForestRegressor

```python
def randomforest_regression_model(dataset, features,overall_type="overall", years=None):
    dataset_name = dataset.__name__ if hasattr(dataset, "__name__") else str(dataset)
    has_years_pro = "years_pro" in dataset.columns

    if has_years_pro:
        features_to_use = features.copy()
    else:
        features_to_use = [feat for feat in features if feat not in ["years_pro", "enter_pro"]]

    x = dataset[features_to_use]
    y = dataset[overall_type]
    x_train, x_test, y_train, y_test = train_test_split(x, y, random_state=2025, test_size=0.2)

    param_grid = {
        'n_estimators': [50, 100, 200],
        'max_depth': [None, 5, 10, 15],
        'min_samples_split': [2, 5, 10],
        'min_samples_leaf': [1, 2, 4]
    }

    rf = RandomForestRegressor(random_state=2025)
    grid_search = GridSearchCV(rf, param_grid, cv=5, scoring='r2',n_jobs=-1)
    grid_search.fit(x_train, y_train)
    best_rf = grid_search.best_estimator_
    y_pred = best_rf.predict(x_test)

    mse = mean_squared_error(y_test, y_pred)
    rmse = np.sqrt(mse)
    r2 = r2_score(y_test, y_pred)

    print(f"Model Results (with tuned hyperparameters):")
    print(f"Best parameters: {grid_search.best_params_}")
    print(f"MSE: {mse:.4f}")
    print(f"RMSE: {rmse:.4f}")
    print(f"R²: {r2:.4f}")
```

E. GradientBoosting



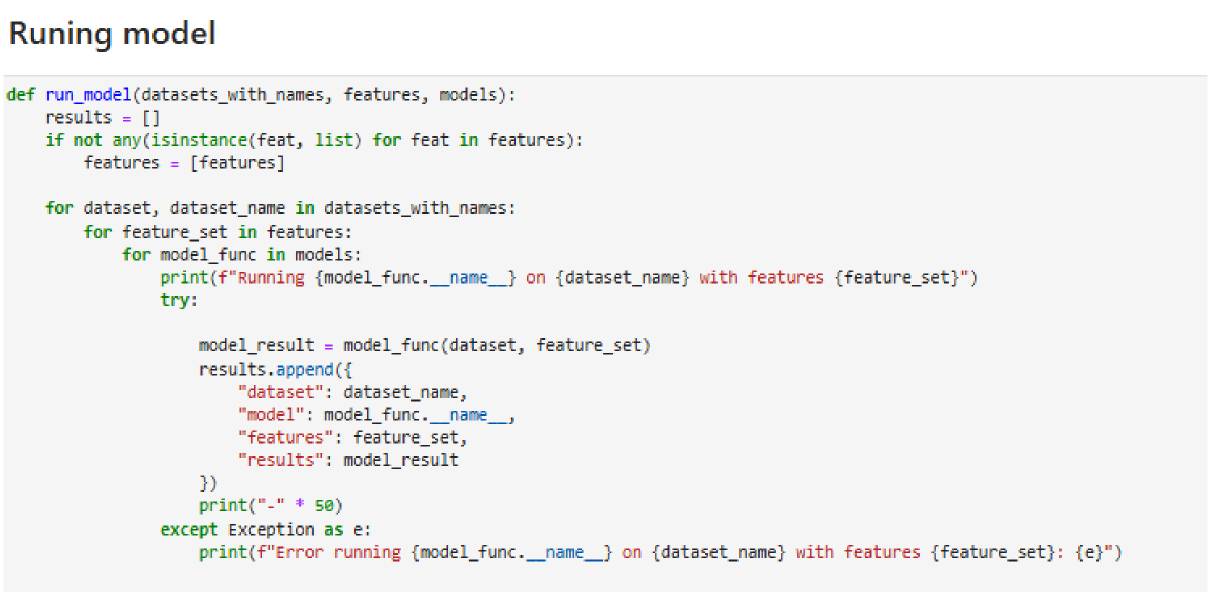
GradientBoosting

```python
def GradientBoostingRegressor_model(dataset,features, overall_type="overall", years=None):
    dataset_name = dataset.__name__ if hasattr(dataset, "__name__") else str(dataset)
    has_years_pro = "years_pro" in dataset.columns

    if has_years_pro:
        features_to_use = features.copy()
    else:
        features_to_use = [feat for feat in features if feat not in ["years_pro", "enter_pro"]]

    x = dataset[features_to_use]
    y = dataset[overall_type]
    x_train, x_test, y_train, y_test = train_test_split(x, y, random_state=2025, test_size=0.2)

    param_grid = {
        'n_estimators': [50, 100, 200],
        'learning_rate': [0.01, 0.05, 0.1, 0.2],
        'max_depth': [3, 5, 7],
        'min_samples_split': [2, 5, 10],
        'min_samples_leaf': [1, 2, 4],
        'subsample': [0.8, 0.9, 1.0]
    }

    gb_model = GradientBoostingRegressor(random_state=2025)
    grid_search = GridSearchCV(gb_model, param_grid, cv=5, scoring='r2', n_jobs=-1)


    grid_search.fit(x_train, y_train)
    best_rf = grid_search.best_estimator_
    y_pred = best_rf.predict(x_test)

    mse = mean_squared_error(y_test, y_pred)
    rmse = np.sqrt(mse)
    r2 = r2_score(y_test, y_pred)

    print(f"Model Results (with tuned hyperparameters):")
    print(f"Best parameters: {grid_search.best_params_}")
    print(f"MSE: {mse:.4f}")
    print(f"RMSE: {rmse:.4f}")
    print(f"R²: {r2:.4f}")
```

This systematic optimization approach ensured that each model was given the best opportunity to perform well on our specific prediction tasks. Additionally, we employed 5-fold cross-validation during the grid search process to ensure robust model performance across different data subsets.

By applying this rigorous modeling framework across our five distinct datasets (one for each sport), we could effectively compare model performance both within and across sports contexts, providing valuable insights into which algorithms best capture the relationship between physical attributes and earning potential in professional athletics.

6. Performance

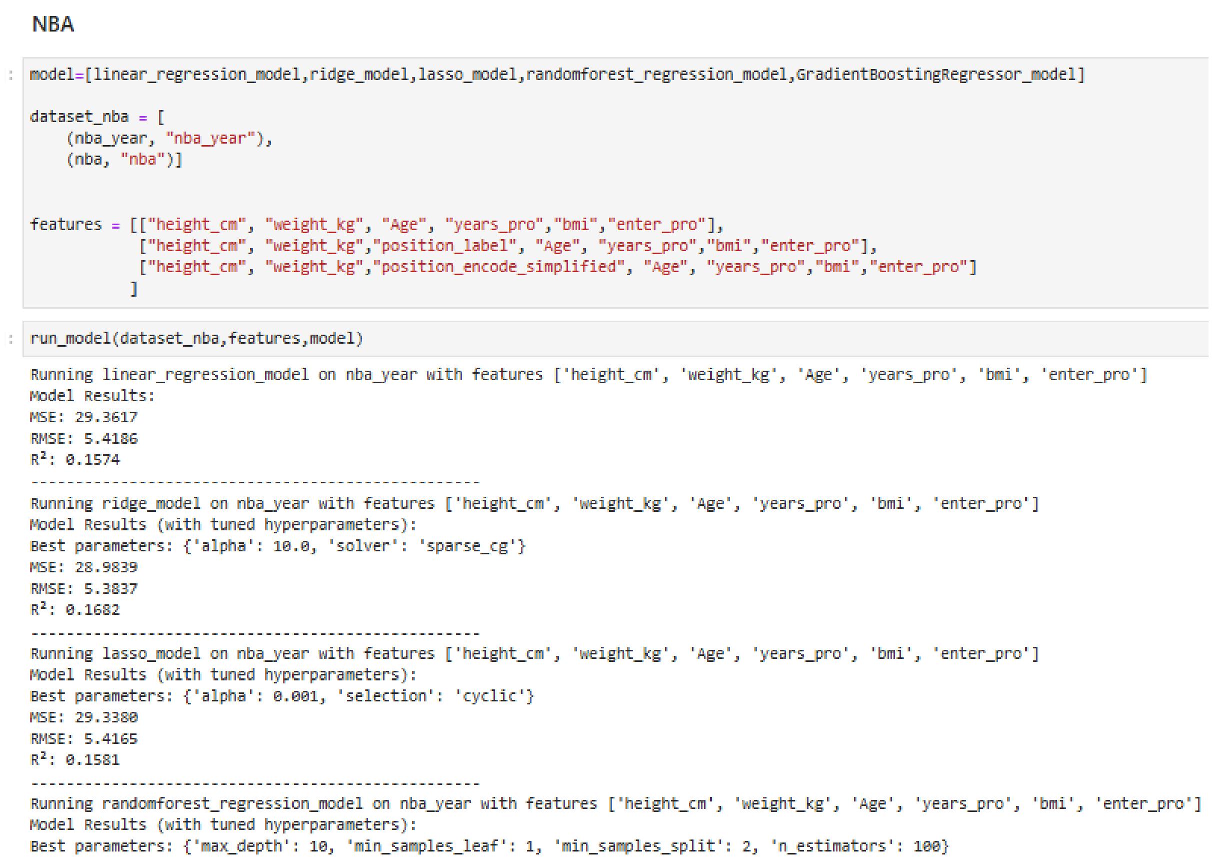


Runing model

```python
def run_model(datasets_with_names, features, models):
    results = []
    if not any(isinstance(feat, list) for feat in features):
        features = [features]

    for dataset, dataset_name in datasets_with_names:
        for feature_set in features:
            for model_func in models:
                print(f"Running {model_func.__name__} on {dataset_name} with features {feature_set}")
                try:

                    model_result = model_func(dataset, feature_set)
                    results.append({
                        "dataset": dataset_name,
                        "model": model_func.__name__,
                        "features": feature_set,
                        "results": model_result
                    })
                    print("-" * 50)
                except Exception as e:
                    print(f"Error running {model_func.__name__} on {dataset_name} with features {feature_set}: {e}")
```

To streamline our experimentation process and ensure consistent evaluation across all sports datasets, we developed a unified model execution framework. This framework significantly enhanced our workflow efficiency by automating the training and evaluation process across multiple models and datasets.

For example, a typical execution of our framework might involve running all five models across our sports datasets with three different feature sets: basic (height, weight, age, position), enhanced (basic features plus BMI), and complete (all features including enter_pro).



NBA

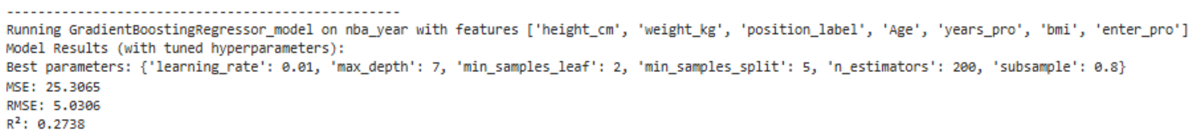
```
model=[linear_regression_model,ridge_model,lasso_model,randomforest_regression_model,GradientBoostingRegressor_model]

dataset_nba = [
    (nba_year, "nba_year"),
    (nba, "nba")]


features = [["height_cm", "weight_kg", "Age", "years_pro","bmi","enter_pro"],
           ["height_cm", "weight_kg","position_label", "Age", "years_pro","bmi","enter_pro"],
           ["height_cm", "weight_kg","position_encode_simplified", "Age", "years_pro","bmi","enter_pro"]
          ]
```

```
run_model(dataset_nba,features,model)
```

```
Running linear_regression_model on nba_year with features ['height_cm', 'weight_kg', 'Age', 'years_pro', 'bmi', 'enter_pro']
Model Results:
MSE: 29.3617
RMSE: 5.4186
R²: 0.1574
--------------------------------------------------
Running ridge_model on nba_year with features ['height_cm', 'weight_kg', 'Age', 'years_pro', 'bmi', 'enter_pro']
Model Results (with tuned hyperparameters):
Best parameters: {'alpha': 10.0, 'solver': 'sparse_cg'}
MSE: 28.9839
RMSE: 5.3837
R²: 0.1682
--------------------------------------------------
Running lasso_model on nba_year with features ['height_cm', 'weight_kg', 'Age', 'years_pro', 'bmi', 'enter_pro']
Model Results (with tuned hyperparameters):
Best parameters: {'alpha': 0.001, 'selection': 'cyclic'}
MSE: 29.3380
RMSE: 5.4165
R²: 0.1581
--------------------------------------------------
Running randomforest_regression_model on nba_year with features ['height_cm', 'weight_kg', 'Age', 'years_pro', 'bmi', 'enter_pro']
Model Results (with tuned hyperparameters):
Best parameters: {'max_depth': 10, 'min_samples_leaf': 1, 'min_samples_split': 2, 'n_estimators': 100}
```

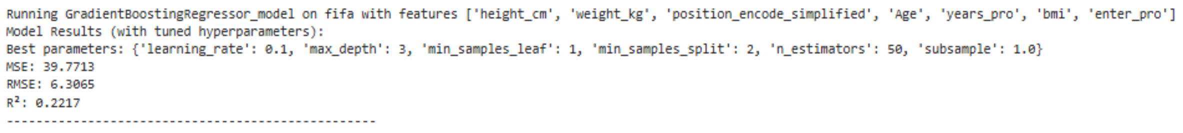
After executing our comprehensive modeling framework across all three sports datasets (NBA, NFL, and FIFA), we successfully identified optimal modeling configurations for each context. This systematic approach yielded valuable insights into which models, hyperparameters, and feature combinations best predict earning potential in different professional sports environments.

A. NBA



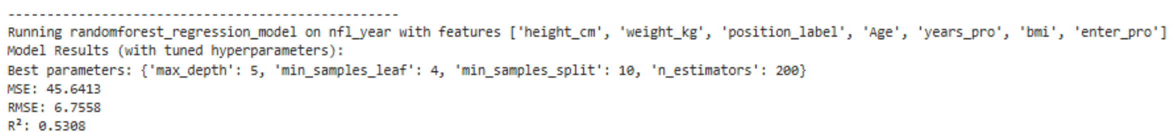
```
--------------------------------------------------
Running GradientBoostingRegressor_model on nba_year with features ['height_cm', 'weight_kg', 'position_label', 'Age', 'years_pro', 'bmi', 'enter_pro']
Model Results (with tuned hyperparameters):
Best parameters: {'learning_rate': 0.01, 'max_depth': 7, 'min_samples_leaf': 2, 'min_samples_split': 5, 'n_estimators': 200, 'subsample': 0.8}
MSE: 25.3065
RMSE: 5.0306
R²: 0.2738
```

B. FIFA



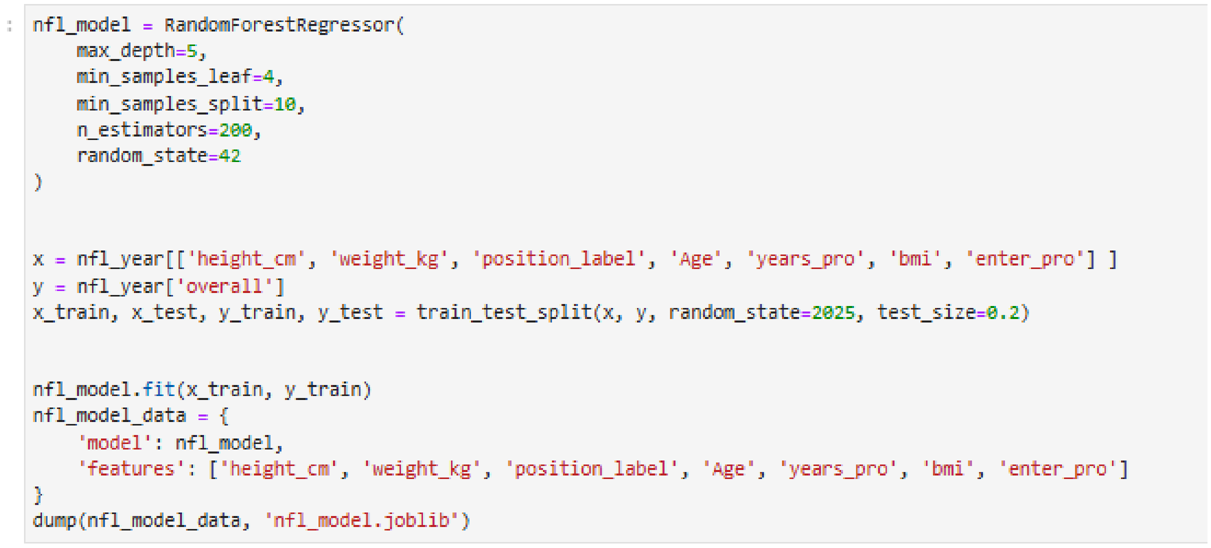
```
Running GradientBoostingRegressor_model on fifa with features ['height_cm', 'weight_kg', 'position_encode_simplified', 'Age', 'years_pro', 'bmi', 'enter_pro']
Model Results (with tuned hyperparameters):
Best parameters: {'learning_rate': 0.1, 'max_depth': 3, 'min_samples_leaf': 1, 'min_samples_split': 2, 'n_estimators': 50, 'subsample': 1.0}
MSE: 39.7713
RMSE: 6.3065
R²: 0.2217
--------------------------------------------------
```

C. NFL



```
--------------------------------------------------
Running randomforest_regression_model on nfl_year with features ['height_cm', 'weight_kg', 'position_label', 'Age', 'years_pro', 'bmi', 'enter_pro']
Model Results (with tuned hyperparameters):
Best parameters: {'max_depth': 5, 'min_samples_leaf': 4, 'min_samples_split': 10, 'n_estimators': 200}
MSE: 45.6413
RMSE: 6.7558
R²: 0.5308
```

After we get all the best parm, he save them into joblib.



```
nfl_model = RandomForestRegressor(
    max_depth=5,
    min_samples_leaf=4,
    min_samples_split=10,
    n_estimators=200,
    random_state=42
)


x = nfl_year[['height_cm', 'weight_kg', 'position_label', 'Age', 'years_pro', 'bmi', 'enter_pro'] ]
y = nfl_year['overall']
x_train, x_test, y_train, y_test = train_test_split(x, y, random_state=2025, test_size=0.2)


nfl_model.fit(x_train, y_train)
nfl_model_data = {
    'model': nfl_model,
    'features': ['height_cm', 'weight_kg', 'position_label', 'Age', 'years_pro', 'bmi', 'enter_pro']
}
dump(nfl_model_data, 'nfl_model.joblib')
```

7. Key Insights

One of our most interesting findings concerned NBA players, where physical attributes demonstrated less predictive power than initially hypothesized. Our analysis revealed that players with nearly identical physical profiles (height, weight, age, and BMI) often exhibited dramatically different performance levels and, consequently, earning potential.

This observation aligns with the widely acknowledged importance of skill development, basketball IQ, and specialized technical abilities in basketball success. While certain physical thresholds remain important for specific positions (particularly center and power forward), our models suggest that beyond these baseline requirements, other factors become more determinative of salary level.

For example, our data included multiple guards of similar height and weight whose salaries differed by millions of dollars, highlighting how shooting ability, playmaking skills, and defensive acumen—attributes not directly captured in basic physical measurements—significantly influence market value in professional basketball.