

1. Selected key stats to define similarity

We used only pre-NBA stats because that's what scouts use. These include:

- College PPG
- College RPG
- College APG
- College FG%
- Age at Draft
- Height

These define how similar players are before the draft.

2. Standardized the stats

Since stats have different scales (height vs FG%), we used StandardScaler to make all features balanced and comparable.

3. Built a similarity graph using k-nearest neighbors

Each player is a node.

We connect each player to the most similar players based on their features.

This forms a “similarity network” of draft prospects.

4. Created a NetworkX graph

We turned the similarity connections into a graph.

This helps us extract advanced relational features.

5. Extracted graph features

Degree Centrality

How many similar players a player connects to.

Clustering Coefficient

How tightly grouped a player's neighbors are.

These graph features reveal patterns simple stats can't show.

6. Multi-hot encoded player positions

Players with multiple roles ("PF/C") now become:

- $\text{Pos_PF} = 1$
- $\text{Pos_C} = 1$

This captures multiple position players correctly.

7. Selected X (features) and y (target)

X includes:

college stats + age + height + graph features + position indexes

y is:

Top3Likelihood (0.70, 0.85, or 1.00)

8. Train–test split (80/20)

We split:

- 80% → training
- 20% → testing

This lets us evaluate the model properly.

9. Baseline Model Training (Linear Regression)

The next step is to train the first machine-learning model. As shown in class, we begin with Linear Regression as our baseline model.

Linear Regression is the simplest regression algorithm and is used mainly to give us a starting point. The idea is to train this basic model first so we can later compare stronger and better-precision models and see if they actually improve performance.

To train the baseline model, we use the training data, which is X_{train} and y_{train} . The model learns the relationship between our features, such as college stats, age, height, graph centrality, clustering, and the multi-hot encoded positions, and the target value Top3Likelihood.

Once the model is trained, we use it to make predictions on the test set, which is X_{test} . Then, evaluate how well it performed using the common regression metrics that the professor taught in class:

- **MAE (Mean Absolute Error)** – average size of the errors.
- **MSE (Mean Squared Error)** – similar to MAE but penalizes bigger mistakes more.

- **RMSE (Root Mean Squared Error)** – the main metric used, easier to interpret.
- **R² score** – how much of the variation the model explains.

This step gives us our first set of results. It tells us how good a simple model performs and establishes a baseline that the more advanced models should aim to beat.

10. Main Model Training (Random Forest Regressor)

The next step is to use a stronger model to improve prediction accuracy. The model we're going to use for better prediction is the Random Forest Regressor. It is an ensemble model that combines many decision trees. Each tree learns slightly different patterns from the data, and the forest averages all its predictions. This can help the model capture more complex, non-linear relationships between our features and the Top3Likelihood (target variable) because players' draft likelihood depends on multiple interacting factors (performance, physical attributes, positional flexibility, and graph-based similarity), a Random Forest is better suited than a simple linear model.

Now it is time to train it using the same training data, which is `X_train`, `y_train`, and then evaluate its performance on the test data (`X_test`). The evaluation uses the same regression metrics from the lectures:

- **MAE (Mean Absolute Error)**
- **MSE (Mean Squared Error)**
- **RMSE (Root Mean Squared Error)**
- **R² Score**

These metrics allow us to directly compare the Random Forest results with the Linear Regression baseline. In most cases, the Random Forest should have lower error metrics (MAE, MSE, RMSE) and a higher R² score, indicating better predictive power. Whichever of the models performs, it becomes the final model for it to be deployed in Streamlit.

Outcome

Linear Regression:

- MAE: 0.0753
- MSE: 0.0085
- RMSE: 0.0925
- R²: 0.2584

Random Forest:

- MAE: 0.1130
- MSE: 0.0200
- RMSE: 0.1415
- R^2 : -0.7366

After testing and training, the linear regression performed better (had lower errors and a positive R^2 score). The Random Forest model performed worse because we didn't have enough data for it. Random Forest works best with large datasets, but ours is small, so the model ends up memorizing the training data instead of learning real patterns. Also, our target values are simple (only 1.00, 0.85, and 0.70), which makes Linear Regression a better match. Linear Regression handles these simple relationships well, while Random Forest becomes too complex and ends up making worse predictions.

11. Final model choosing aftermath

Once we selected it as the best model, we retrained it again using the entire dataset instead of just the training split. Because the goal of the 80/20 split was to compare models, not to build the final one, retraining on 100% of the data gives the model more information and makes it stronger for real predictions. We do not re-evaluate this final version because testing/evaluation occurs only on the train-test split model, not the final one that will be used for deployment. This gives the model the maximum amount of information before being used in the GUI. We also saved the exact list of feature columns so the GUI can send inputs to the model in the correct order and avoid incorrect predictions. Now, the final model is trained and ready for user predictions in Streamlit.

12. Building graphs

1st graph: To analyze how player attributes relate to each other, we created a correlation heatmap using only the main numeric features (PPG, RPG, APG, FG%, Age, and Height). The heatmap visually shows how strongly each stat increases or decreases together, using colors to represent the strength of the relationship. We excluded the graph-based features and the target variable (Top3Likelihood) because they showed minimal variation and did not correlate meaningfully with the performance metrics. By focusing only on the true numeric attributes, the heatmap becomes much clearer and reveals simple but useful patterns, for example, that better scorers often have moderately higher rebounds or assists, and that age or height vary independently from game performance.

2nd graph: The Predicted vs Actual plot visually evaluates how close the predictions of the Linear Regression model are to the real “Top3Likelihood” values in the test set. Each point represents one player from the test data (20% of 51). Points that fall near the diagonal line indicate accurate predictions, since the predicted value matches the actual value. In our graph, the points form a diagonal pattern, meaning the model is able to capture the general relationship between player attributes and the likelihood of being a top-3 draft pick. While some predictions fall slightly above or below the ideal line (showing small errors), there are no extreme outliers. This confirms that the Linear Regression model performs considerably well for this dataset we have.