NCAA 26 Skill Points Model

Alex Swanner

10/30/2025

linkedin.com/in/alexswanner/

Executive Summary

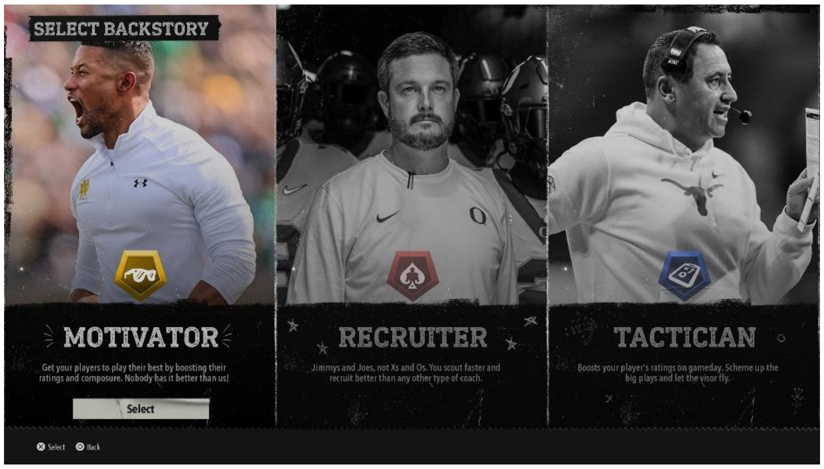
The purpose of this research was to build a model to accurately predict the amount of skill points a player would have in total on the “Training Results” screen in the “Dynasty” game mode from *EA Sports College Football 26*. *EA Sports College Football 26* is a college football video game available on platforms such as Xbox and PlayStation. In the *Dynasty* game mode, you can control a team or teams across multiple seasons starting in 2026. You play games, recruit players, fire coaches, build schedules, and much more. In the offseason, there is a week called “Training Results”. It is during this week you are able to see how many skill points each of your players has accumulated up until now. Skill points are the method in which you can manually upgrade your players. Understanding the best way to maximize these skill points is key when it comes to building your team.

I built a multiple linear regression model in Excel and in Google Colab to predict skill points. I collected my data directly from the Training Results screen in the game. I manually typed in hundreds of entries into excel. I listed names, positions, development traits, and coach skill point boosts that applied to each player. My current model, which consists of five hundred six entries, has an adjusted R-Squared score of .932. This is strong predictive capability. With the

assistance of generative AI (OpenAI) and Claude, I wrapped it up by building a GUI in python where users can input a player's information to receive a prediction for their skill points.

Introduction

In the world of EA *Sports College Football 26*, there are many game modes available. A fan favorite game mode is *Dynasty.* You can control one or many teams across up to 30 seasons. You can play by yourself, or in a shared Dynasty with friends. There are over 100 teams to choose from; you can start with a big program such as Alabama, or a small program like Kennesaw St. You can use existing head coaches and coordinators such as Hugh Freeze or DJ Durkin of Auburn University; there's over 300 real life coaches to choose from. There's also the option of creating your own coach. Coaches are really important in the scope of the game mode. Coaches, along with players, receive points that you can use to upgrade them. There's a coaching tree that consists of different areas of expertise. Some examples are motivator, tactician, and recruiter. Recruiter helps with recruiting; motivator helps with player development, and tactician gives direct gameplay boosts.

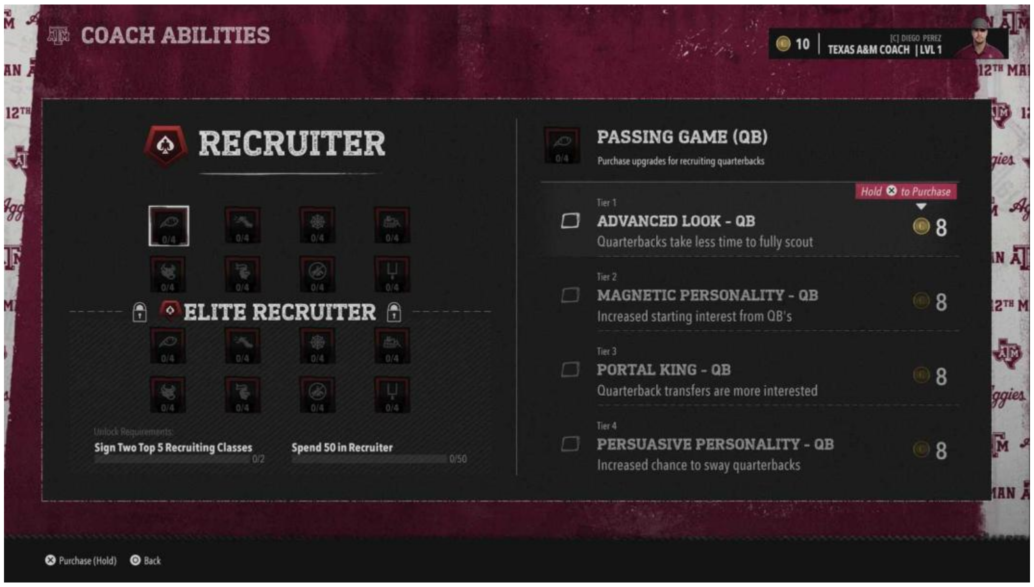


Get your players to play their best by boosting their ratings and composure. Nobody has it better than us!

Jimmys and Joes, not Xs and Os. You scout faster and recruit better than any other type of coach.

Boosts your player's ratings on gameday. Scheme up the big plays and let the visor fly.

These three coaching abilities are the base abilities; when you create a new coach, whether it's a head coach, offensive coordinator, or a defensive coordinator, you will select one of these abilities first. You start at level one and as you do things like sign recruits and go on win streaks, you earn skill points. With these skill points you can upgrade the abilities you have already purchased or try to purchase new abilities. Each of these main abilities has a tree of its own. The base three, recruiter, motivator, and tactician, each have two 'parts. The first part of each of the base three abilities is all unlocked, and you unlock the second part by completing additional in-game goals. All of the trees within the abilities are tied to a position group. There are eight position groups in each ability. The position groups are as follows: QB, RB/FB, WR/TE, Oline, Dline, LB, Secondary, and K/P.


COACH ABILITIES

10 | [C] DIEGO PEREZ TEXAS A&M COACH | LVL 1

RECRUITER

ELITE RECRUITER

Unlock Requirements:
Sign Two Top 5 Recruiting Classes 0/2
Spend 50 in Recruiter 0/50

PASSING GAME (QB)
Purchase upgrades for recruiting quarterbacks

Hold ⊗ to Purchase

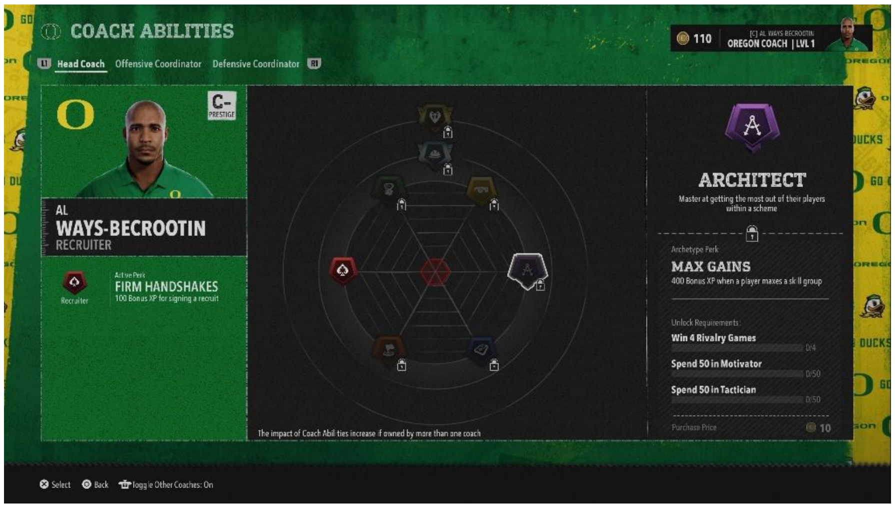
Tier 1
ADVANCED LOOK - QB
Quarterbacks take less time to fully scout
8

Tier 2
MAGNETIC PERSONALITY - QB
Increased starting interest from QB's
8

Tier 3
PORTAL KING - QB
Quarterback transfers are more interested
8

Tier 4
PERSUASIVE PERSONALITY - QB
Increased chance to sway quarterbacks
8

With each position group, there are four tiers. Each additional tier is locked behind the one before it, except for tier 1 of course. There are additional coaching abilities that will be able to be unlocked as you upgrade your coach. These coaching abilities are super important as they can



Active Perk
FIRM HANDSHAKES
100 Bonus XP for signing a recruit

ARCHITECT
Master at getting the most out of their players within a scheme

Archetype Perk
MAX GAINS
400 Bonus XP when a player maxes a skill group

Unlock Requirements:
Win 4 Rivalry Games 0/4
Spend 50 in Motivator 0/50
Spend 50 in Tactician 0/50

Purchase Price 10

The impact of Coach Abilities increase if owned by more than one coach

have a direct influence on how your players can develop. The two main coaching abilities with respect to player skill points are Motivator and Talent Developer. I will explain these later when I introduce my explanatory variables.

Maximizing player skill points is my favorite way to build my teams. Every off season you will find out exactly how many skill points each of your players has to take advantage of. In last year’s edition of the game, you really didn’t ever see how many points they got. There was no way to manually upgrade the players. It was all done automatically, so as soon as the player had enough skill points to upgrade, it was just done by the computer. In last year’s edition and in this one, each player has “buckets” of attributes you can upgrade. For example, just about every player has a “Quickness” bucket that includes things like speed and acceleration. When you upgrade a bucket, at least 1 of the attributes in the bucket will be increased. Each bucket has a different cost associated with it. Some buckets are cheap, and others are expensive. It scales in that as you get closer to maxing out an attribute bucket, it gets more expensive. Also, it’s important to know that not every bucket can be fully maxed out for every player. Each player has “caps” on their attribute buckets. For example, some players come in at the fastest they will ever be. They come in with a speed rating of ninety, and that’s the best they can get. Other players can come in with a ninety-speed rating and be able to get all the way up to ninety-nine.

In this year's edition of the game, you can manually upgrade players. This was a key feature many fans like me were hoping to see. This allows you to have more control over how your players upgrade and how you function as a team. Want to have a strong running game? You can make sure your offensive linemen upgrade their run block. Want to attack defensives down the field? You can upgrade your QBs' arm strength. Getting these skill points for your players is one of, if not the best, ways to build your team. It's important for getting a competitive edge against your foes in the online dynasties. Understanding what factors really influence player skill points is the main focus of this research.

Data Collection Methodology

All of the data I used to build this model came from my own system, my own game, and my own Xbox. I manually entered five hundred twenty-eight entries into an excel sheet. My data set currently consists of eight teams with about fifty players each. When I first got the idea to build this model, I figured I would just use a player set from a dynasty I was actively participating in. It was an offline dynasty that was with my favorite team, Auburn. I was about 15 years into the future when I got the idea. It's important to note that this data set is unique in that it has the most coaching abilities applied to it. I made my own coach, Alex Swanner, who is level ninety, and neither of my coordinators ever left, so they too have a good number of coaching abilities. This is the source of my first team in the data set. I initially figured I would just add to the data set when I go to the Training Results screen every year, but I realized I wouldn't want to get through the seasons that fast. I wanted new entries sooner than that. I decided on starting a new dynasty, selecting the current head coach, and simming to the Training Results screen. This is the method I used for most of the data collection. The last one hundred-twenty-nine entries are actually from the same team, Georgia. I got these entries from the same dynasty, and they're from two Training Results screens. They are two years apart. They would have been one year apart, but I made a mistake. On year one in the Training Results screen, I recorded my data and decided to sim to the next Draft Results screen. Here I recorded the players that were drafted from the team. This is important; I'll explain later. I then went to the next Training Results screen to see how my players performed this season in terms of skill points. I come to find out that none of the skill points got spent, so this year two Training Results consisted of two years of skill points, not just one. I then had to do the process again, sim to next year's Players Drafted screen, and then to the Training Results screen. I did two sets of entries with Georgia because I was lacking enough Elite players in my data set. They are hard to come by. With Georgia, I spent a season recruiting hoping to find Elite players to record information on. I needed more of this data to train my model better. I used these settings with each setup:

Coach Firing: Off

Injury: Off

Wear and Tear: Off

User Player Transfer Chance: 0

Auto Progress Players: 0

Quarter Length: 5 mins

Default roster

Offline Dynasty

Coach Firing was set to off so that I wasn't fired. Being fired would prevent me from getting to see the Training Results screen for the team I selected. I turned Injury and Wear and Tear off to prevent my players from getting injured. Injuries would affect their playing time which could affect their skill points. Transfer Chance is set to zero so that my players would I wanted to collect data on wouldn't leave. Quarter Length isn't super important as I simmed all the games anyways. Quarter Length applies only when you are actually playing the game. I figured this was something good to keep constant though. Default roster and Offline Dynasty are also for controlling variables as well. In the start of the seasons, I took advantage of the pre order bonus to get more coach ability points. You receive one hundred; I used these to get more of the key coach abilities to get further variation.

I used sources like cfblabs.com to find coaches with a good mix of Motivator and Talent Developer abilities. I first chose Clemson, followed by Michigan and Rutgers. These teams had a good mix of coaches with these abilities.

Finding coaches with these abilities was really important as these abilities are key variables in the models that I built. The variables in my models:

Team- Team that the player plays for

Name- Name of player

Skill Points (Independent Variable)- How many skill points the player had on training results screen

Position- What position the player plays

Year- Class of player, i.e. Freshman, Sophomore, etc.

Dev Trait- Development trait that the player has, direct influence on player development.

Dev Trait #- Number assigned to dev trait to make a numeric column

Snaps- Number of snaps player played

HC_Moti 1- Tier three in part one of the Motivator ability, “[Position group] get an off-season training boost.” For head coach

HC_Moti 2- Tier three in part two of the Motivator ability, “Increase all XP gains for [Position group].” For head coach

OC_Moti 1- Tier three in part one of the Motivator ability, “[Position group] get an off-season training boost.” For offensive coordinator

DC_Moti 1- Tier three in part one of the Motivator ability, “[Position group] get an off-season training boost.” For defensive coordinator

HC_TD 1- Tier one of the Talent Developer ability, “Bonus XP for [Position group] when [Position group] are drafted.” For head coach

HC_TD 2- Tier two of the Talent Developer ability, “Freshman and Sophomore [Position group] progress faster.” For head coach

HC_TD 3- Tier three of the Talent Developer ability “Starting [Position group] will gain XP faster.” For head coach

OC_TD 1- Tier one of the Talent Developer ability, “Bonus XP for [Position group] when [Position group] are drafted.” For offensive coordinator

OC_TD2- Tier two of the Talent Developer ability, “Freshman and Sophomore [Position group] progress faster.” For offensive coordinator

OC_TD 3- Tier three of the Talent Developer ability “Starting [Position group] will gain XP faster.” For offensive coordinator

DC_TD 1- Tier one of the Talent Developer ability, “Bonus XP for [Position group] when [Position group] are drafted.” For defensive coordinator

DC_TD 2- Tier two of the Talent Developer ability, “Freshman and Sophomore [Position group] progress faster.” For defensive coordinator

DC_TD 3- Tier three of the Talent Developer ability “Starting [Position group] will gain XP faster.” For defensive coordinator

XP Penalty- In the main menu there is a slider to penalize manual progression. Functions as a percentage, scales from zero to one hundred.

Development trait is a categorical variable, and in order to use it in regression I need it to be numeric. There are four unique traits: Normal, Impact, Star, and Elite. Normal is the worst and Elite is the best. I decided to assign a number to each development trait, 1 for normal, 2 for impact, 3 for star, and 4 for elite. There isn’t a variable for tier 3 of part 2 in Motivator for the offensive and defensive coordinators as I couldn’t find a coordinator with this ability. Year is a really important variable as it is tied to tier two of the Talent Developer ability. It reads as freshman and sophomores progressing faster. A tricky part of this is not knowing exactly how the game defines freshman and sophomores. For the sake of this research, I included freshman, sophomores, redshirt freshman, and redshirt sophomores.

Also with year, on the Training Results screen, for every player it lists their new class. I decided to do a kind of minus one here. I wanted to see how they developed as the player they were for all of last season, not for who they literally just became. For example, if a player just became a junior, I want to see how they developed all last season as a sophomore. A tricky part of this was not knowing if they had just received their redshirt or not. For example, do I list a redshirt junior as a junior, someone who just got their redshirt, or as a redshirt sophomore, someone who got their redshirt before the season they just played? For consistencies sake, I listed anyone who had RS in their name as someone who already had their redshirt. If I saw redshirt sophomore, I put redshirt freshman. Anyone listed as a redshirt freshman was just listed as a freshman though. Because of this, you won't see anyone listed as just a senior. If I saw senior in the game, I listed them as a junior. If I saw redshirt senior, I listed them as redshirt junior.

Instead of listing every single position, I lumped them together in a similar manner to the coach abilities. I combined FS and SS into just S. LB is MIKE, WILL, and SAM. OL is LT, LG, C, RG, and RT. DL is REDG, DT, LEDG. RB is HB and FB. I kept QB, WR, TE, K, and P on their own though. For Talent Developer tier three, I defined a starter as more than three hundred snaps.

It's also important to know that at the end of the season a player can have their development trait increase. For the sake of this research, I just took the development trait that was displayed on their player card, the one it got upgraded to.

There are two variables I could have tried to include here. Two coaching abilities that do likely influence skill points. I did not include them in this study as tracking them would have been very difficult. One of the coaching abilities is called Architect. Like Talent Developer it has four tiers and eight position groups. Tier one reads: "[Position Group] earn an XP bonus for every three-game win streak." Tier two reads: "[Position Group] earn additional XP for in-game goals." Tracking either of these would have been very difficult. For tier one I would have to track how many three game win streaks the team went on during the course of the season. It's tricky from there though. With the other coaching abilities, it's a yes or no, a one or a zero. With tier one of architect, I would have to somehow account for the team going on multiple three game win streaks. How would I have done that? For tier two, I'm not even sure if there's a way of seeing what the in-game goals are. This is why I did not include either of these coaching abilities in the regression.

For XP Penalty, most of the data is from zero to 50. I did one trial with Minnesota at 90. I intended on getting data at every ten-point increment from 0 to 90, but I didn't. Initially I wanted the model to be trained at ranges from basically zero to ninety-nine, but I realized that most players play at a lower range. For practicality, it was okay to have the model trained at lower penalty values.

I then took these entries and ran multiple regression tests in excel. I ran Regression using the Data Analysis tool in Microsoft's Excel. I ran a new regression test after each additional full team I got. My results initially were not good, but they continued to improve with each additional team that I added. Here are some of the results I got from running multiple regression in Excel:

37 Entries	
Multiple R	0.835485573
R Square	0.698036142
Adjusted R Square	0.380387547
Standard Error	21.64835577
Observations	37

156 Entries	
Multiple R	0.905093511
R Square	0.819194263
Adjusted R Square	0.798382092
Standard Error	12.93295724
Observations	156

270 Entries	
Multiple R	0.91430196
R Square	0.835948074
Adjusted R Square	0.825573248
Standard Error	11.14380168
Observations	270

399 Entries	
Multiple R	0.922414388
R Square	0.850848303
Adjusted R Square	0.844601112
Standard Error	10.20185504
Observations	399

528 Entries	
Multiple R	0.920791968
R Square	0.847857848
Adjusted R Square	0.843094102
Standard Error	9.946388451
Observations	528

As you can see, just about every metric improved with more entries. These are just the numbers I got from excel.

R-Analysis

Just running the regression in Excel was not enough to get the insight I wanted. I moved to Google Colab to write some R code to get further analysis. I used generative AI, specifically ChatGPT and Claude to assist me in this. I uploaded the data and previewed it using head and str functions. When looking at my data I saw that it needed some cleaning. In some of my entries I had some spaces before or after the entry. This created duplicate category variables. I ran this code to clean it:

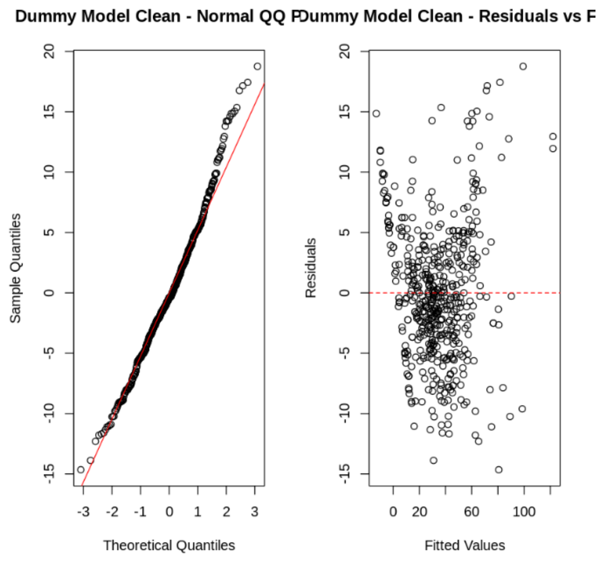
```
#Cleaning Columns

ncaa_data$Position <- trimws(ncaa_data$Position)

ncaa_data$Year <- trimws(ncaa_data$Year)
```

From there, I began building models. I have two key categorical variables, Position and Year. I decided to code dummy variables as a way to kind of turn these categorical variables into numeric variables. The dummy variables are binary indicators that represent different categories within a variable. For example, with Position, it marks a 1 for the position you are, and 0 for the position you aren't. Doing this allows Position and Year to be a part of the regression models. It allows me to see how Position and Year influence a player's skill points.

The first model I built was a full model; it consists of all the variables from the dataset, without the dummies. I then made a model with all the variables plus the dummies. From here I did some more cleaning. I identified outliers and removed them. I classified outliers with just the standard IQR method. I calculated Q1 and Q3 and then formed upper and lower bounds. Anything that fell below Q1 – 1.5 × IQR or above Q3 + 1.5 × IQR were classified as outliers and removed from the dataset. IQR is the interquartile range; this is the distance between Q1 and Q3. I did this for both models, the one without dummies, and the one with them. I identified twenty-five outliers for the full model and twenty-two for the dummy model. After doing this cleaning, I made two new models. The two new models are without outliers. From here I got residuals and calculated a bunch of metrics on all four models. I got this comparison table:

Model	**R²**	**MAE**	**RMSE**	**AIC**
Full Model	0.851	6.67	9.68	3935.15
Dummy Model	0.899	5.66	7.98	3763.01
Full Model (Cleaned)	0.901	5.11	6.66	3375.67
Dummy Model (Cleaned)	0.932	4.39	5.65	3259.85

As you can see, cleaning the data, removing the outliers, helped a lot. My best model of these four is the cleaned model with dummy variables. I decided to get a visual on my residuals with a QQ-Plot.



I then printed out each of my entries, along with their predicted values, actual values, and the residuals. I did this for each of my four models. I also wrote some code to get accuracy ranges. I wanted to see how accurate the models were within certain ranges. The code looks like this:

```
# Accuracy ranges for Dummy_Model_Clean

cat("=== DUMMY MODEL CLEAN ACCURACY ===\n")
ranges <- c(50, 25, 20, 10, 7.5, 5)
for (r in ranges) {
  within_range <- sum(abs(ncaa_dummy_no_outliers$Skill.Points -
ncaa_dummy_no_outliers$Predicted_Dummy) <= r)
  percent_within <- within_range / nrow(ncaa_dummy_no_outliers) * 100
  cat("Number of predictions within", r, "points:", within_range, "\n")
  cat("Percentage of predictions within", r, "points:", round(percent_within, 2), "%\n\n")
}
```

I did this for all my models and got this comparison table:

Accuracy Range	**Full Model (%)**	**Dummy Model (%)**	**Full Model (Cleaned) (%)**	**Dummy Model (Cleaned) (%)**
±50 points	99.81%	100%	100%	100%
±25 points	97.16%	98.67%	100%	100%
±20 points	96.21%	96.97%	100%	100%
±10 points	78.60%	85.61%	82.50%	89.33%
±7.5 points	68.18%	73.30%	71.57%	76.48%
±5 points	52.84%	56.25%	55.47%	58.70%

The 89.33% accuracy within +-10 points is strong predictive performance.

I decided to see about improving the models even more. Each of my variables had a coefficient along with a significance level. I decided to improve the models even further by removing any of the variables that are not significant. I identified these variables as insignificant: HC.Moti.1, OC_Moti.1, HC_TD3, OC_TD2, OC_TD3, and DC_TD1. I then printed out a comparison table like this to see if removing those variables made a difference.

Model	Variables	R^2	MAE	RMSE	AIC
Dummy Model (Clean)	33	0.932	4.39	5.65	3256.27
Dummy Model (Refined)	27	0.931	4.41	5.69	3251.50

As you can see, using Dummy Model Refined didn't actually really improve the model. It has a lower R^2, and a higher MAE. My model that consisted of dummy variables with outliers removed was determined to be my best model. These are the coefficients for that model:

Variable	Estimate	Std. Error	t value	p-value	Significance
(Intercept)	83.393	2.559	32.590	<0.001	***
DevT Impact	-38.170	1.990	-19.182	<0.001	***
DevT Normal	-50.541	2.054	-24.610	<0.001	***
DevT Star	-23.550	1.987	-11.855	<0.001	***
HC_Moti.1	0.382	0.657	0.581	0.561	
HC_Moti.2	2.013	0.767	2.625	0.009	**
OC_Moti.1	0.628	1.128	0.557	0.578	
DC_Moti.1	3.143	0.894	3.516	<0.001	***
HC_TD1	9.575	1.072	8.934	<0.001	***
HC_TD2	2.715	1.106	2.454	0.014	*
HC_TD3	-0.326	1.601	-0.204	0.839	

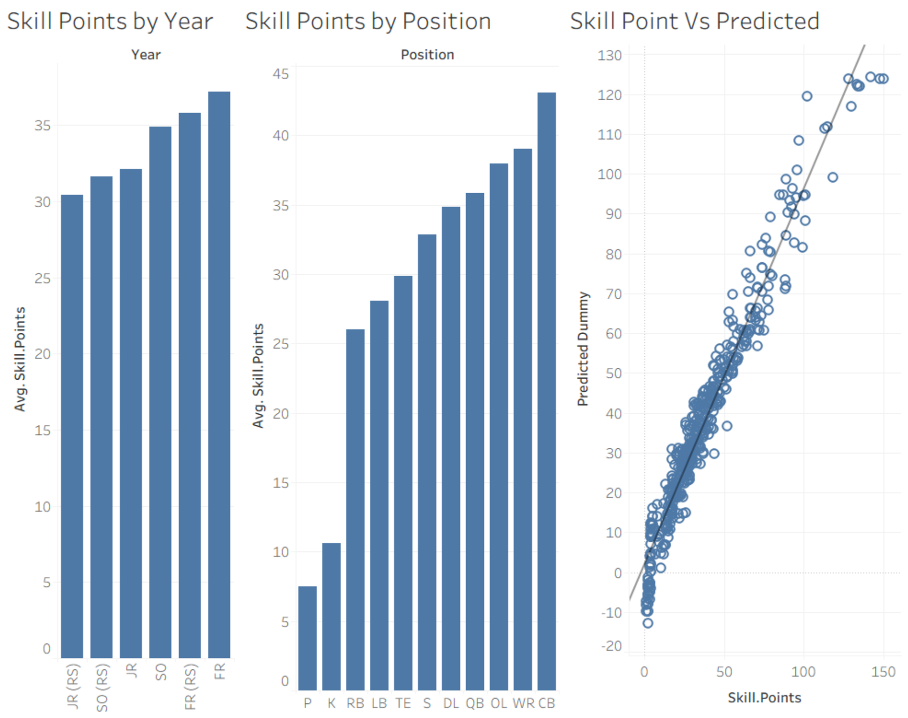
OC_TD1	3.068	0.990	3.097	0.002	**
OC_TD2	1.862	1.481	1.257	0.209	
OC_TD3	2.450	2.085	1.175	0.241	
DC_TD1	-1.486	2.496	-0.595	0.552	
DC_TD2	14.250	2.556	5.575	<0.001	***
DC_TD3	13.068	2.688	4.861	<0.001	***
XP.Penalty	-0.428	0.012	-35.737	<0.001	***
Position_CB	0.567	1.708	0.332	0.740	
Position_DL	-6.448	1.596	-4.040	<0.001	***
Position_K	0.260	3.078	0.084	0.933	
Position_LB	-10.656	1.656	-6.436	<0.001	***
Position_OL	3.175	1.531	2.074	0.039	*
Position_P	0.892	2.869	0.311	0.756	
Position_RB	-4.670	1.630	-2.865	0.004	**
Position_S	-2.172	1.664	-1.305	0.193	
Position_TE	-6.686	1.680	-3.980	<0.001	***
Position_WR	-0.362	1.525	-0.237	0.813	
Year_FR (RS)	-2.881	0.803	-3.588	<0.001	***
Year_JR	-4.769	1.021	-4.669	<0.001	***
Year_JR (RS)	-2.515	0.980	-2.567	0.011	*
Year_SO	-3.163	1.029	-3.075	0.002	**
Year_SO (RS)	-5.085	0.902	-5.638	<0.001	***

The baseline for the development trait variables is elite, so Impact, Normal, and Star are all in comparison to it. The baseline for the position variables is QB, so all the other positions are in comparison to how Quarterbacks develop. Finally, The baseline for the year variables is FR, so all of the other years are in comparison to how freshmen develop.

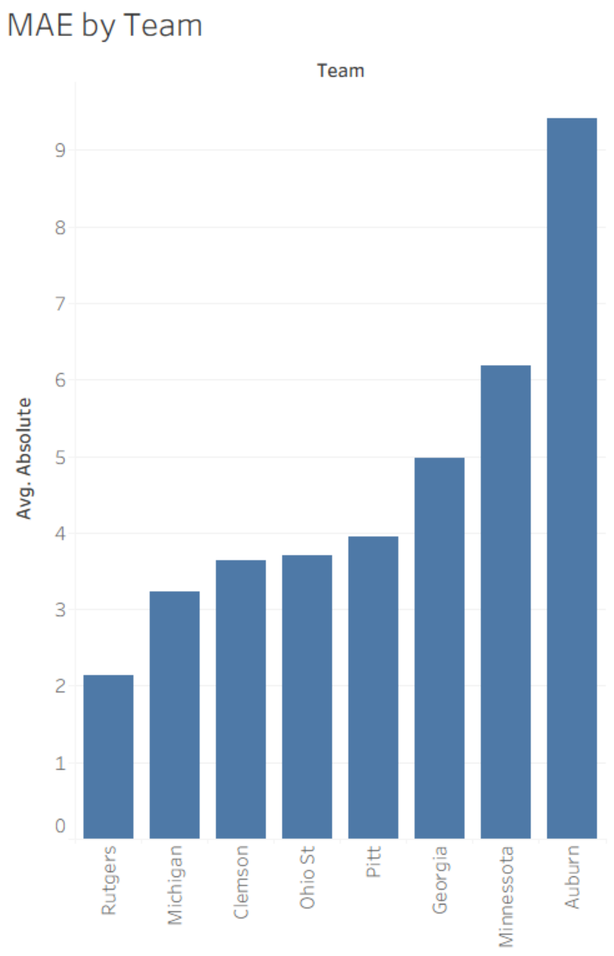
It is important to note that some coaching ability coefficients are negative (HC_TD3: -0.33, DC_TD1: -1.49). This does not mean these abilities cause players to lose skill points in the game; rather, these variables were statistically insignificant ($p > 0.05$) and their coefficients are likely due to random variation in the sample. In practice, coaching abilities can only provide neutral or positive effects on player development. I chose to retain all variables in the final model rather than remove the insignificant ones, as the Refined model (which excluded these variables) did not show meaningful improvement in predictive accuracy. The full model with all coaching abilities provides a more complete picture of the coaching system, even if some individual effects are not statistically significant.

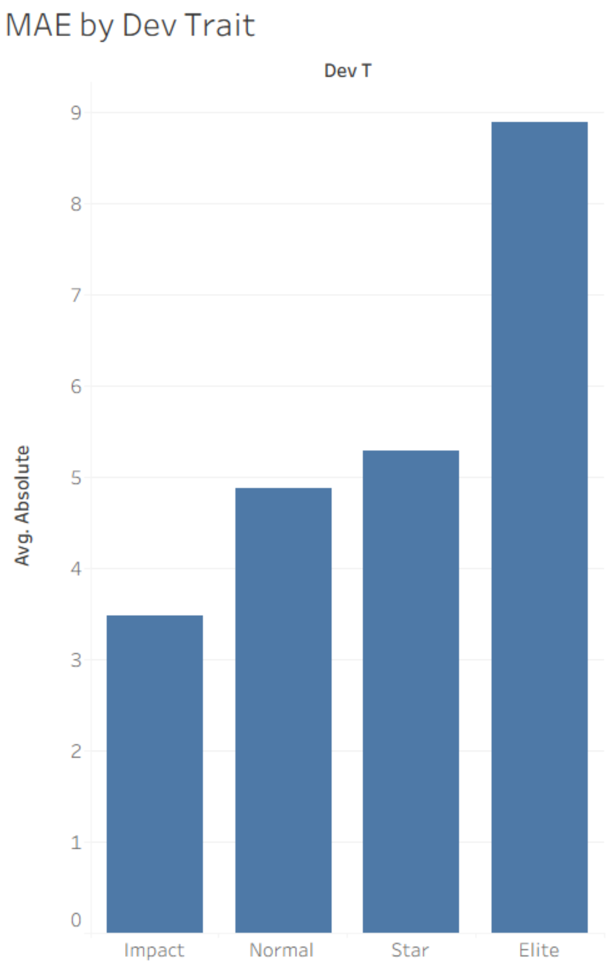
Visualizations

I exported my data from R into a .csv file that I then imported into Tableau. I made two dashboards. The first one focuses on skill points, specifically skill points by player year and actual skill points compared to the model predictions. The second one is a look at MAE compared to team, dev trait, and position.



Here you can see that the underclassmen outperform their older counterparts. This is likely due to the fact that there is an available coaching ability (TD_2) to help just the Freshmen and Sophomores. Based just on the graph though, maybe redshirt sophomores aren't included in that? Maybe it is just freshmen, redshirt freshmen, and true sophomores that get the boost. We can see what positions earn a lot of skill points and which ones don't. Punters and kickers definitely struggle while corners and receivers are living in luxury. It is important to note that of all the positions, I have the fewest number of punters and kickers as there's only 2-3 per team. Finally, the skill points vs predicted chart. I'm really proud of this as you can clearly see that the model fits well. It falls off a little bit past maybe 60 or 70, but overall, really solid.





When looking at this dashboard, it's apparent that the data for Auburn is different from the rest, and that's because it is. I actually ran some code to see if all of the data from Auburn should be removed. I found that the MAE actually improved by an entire point, but the R^2 actually would have gone down. I decided to just keep the Auburn data. It wouldn't hurt to have the model trained on some data that is further down the line. Players who are further into their dynasty may want to use this model as well. It's also no surprise that the Elite dev trait had the highest MAE as it had a small sample size (N = 11). Conversely, it makes sense that Impact would have the lowest MAE as it had the highest number of entries (N= 250).

Web Application

To really top off this research, with the assistance of Claude AI, I built a web application that fellow Dynasty players can use to predict their players' skill points. The application was built using Streamlit. It uses a python framework that takes scripts into interactive web applications. This allowed me to take my model and turn it into a user-friendly application that anyone can access simply with a URL link. The app is hosted on Streamlit Cloud. This is a free deployment platform that makes the model accessible to everyone. In the app users input information such as a player's position, year, development trait, etc. and they immediately receive a skill point prediction. The app also displays numbers about how accurate the prediction will be. It provides a range as well as a likelihood for the prediction.

Not only is this great because everyone in the community can have access to the model, but by users using this tool, they can actually help to train the model even further by providing more entries for the dataset that the model is trained on. With each submission, users are requested to input the actual amount of skill points their player receives in the off season. Granted, not everyone who uses the model will be on the training results screen, but it's still a cool function. All of this data is stored in a Google Sheets database through Google Cloud's API system. This allows for a secure connection between the application and the database. This ensures I won't have to manually input any more information on my end. The Google Cloud service account credentials are stored as 'secrets' in Streamlit Cloud rather than in the code itself. This keeps the database protected while allowing the app to function correctly.

This web app approach transforms the project from a one-time analysis into an evolving system. As more and more users take advantage of the app, the dataset will grow well beyond five-hundred-six. With each significant increase in the dataset, I can rerun my models to train them even better. This creates a sort of feedback loop where users will be rewarded for using my application with an even better model.

Conclusion

The Training Results screen is a really pivotal part of the offseason. You get to manually develop your players; this gives you a lot of flexibility with molding your team into a powerhouse. Understanding how many skill points each of your players will have to work with can give you a competitive advantage over other users and computer-controlled teams. The model I built is strong and accurate with an R-Squared of .932 and an MAE of 4.39.

This model is not perfect though; it has its limitations. Some complex variables such as additional coaching abilities were omitted. My data pool needs further improvement as I am lacking elite players, punters, and kickers. Building upon this data pool could train my model to be increasingly accurate. I intend on improving this model by adding more players to the data set.

Moving beyond this I look forward to seeing increases in my dataset from other users in the NCAA 26 community. As players use the application, the model will continue to evolve and improve. This approach has the potential to provide some new and in-depth insights that I could not have realized alone, at least not in a short amount of time.

The complete project, including all code, data, and documentation is available on GitHub at: https://github.com/alexswannerr/NCAASkillPointsResearch

AI Assistance Disclosure

This research was conducted with the assistance of generative AI tools, specifically Claude AI (Anthropic) and ChatGPT (OpenAI). These tools were used to support R code development, Python application building, debugging, and deployment guidance.