

# Baseball Project: On Slugging

Sam Caruthers and Devon Kost

3/20/2025

## Introduction

We want to build a model for the UCSB Division I Baseball Team that predicts slugging percentage of hitters based on exit speed and launch angle. Exit speed is the speed the ball has off the bat after the player hit it, as in how much speed did they put on the ball. Angle or launch angle is the angle that the ball flew off of the bat after the hitter made contact. We pick these two metrics because we want the model to not only be interpretable to us, but we want it to be interpretable to the players and exit speed and launch angle are realistically the only two variables that the hitter can control are angle and exit speed.

Prepping the data for modeling. Ensuring that we have enough data in each column to appropriately predict slugging percentage

## Cross validation

The code below ensures that we train all the models fairly using the same cross validation for each so that we can accurately compare their results.

first we split the data in order to have a test set and a train set.

We then build the four models, for this type of AI Algorithm, we believe that RandomForest, XGBoost, DecisionTree, and BaggedTree are the best models to predict slugging percentage. We will run a 5-fold CV.

## Model Comparison

After building the models, we want to compare them via Test RMSE in order to decide which model is the best.

Table 1: Comparison of Model Performance Metrics

Model	RMSE	Rsquared	MAE
DecisionTree	0.8594634	0.3042426	0.6376472
RandomForest	0.8463812	0.3413625	0.5250407
XGBoost	0.8697410	0.2872705	0.6003529
BaggedTree	0.8121666	0.3794595	0.5608769

So after running cross-validation, we are able to test the 4 different models against each other and we find that BaggedTree has the least RMSE, followed by RandomForest, then DecisionTree and finally XgBoost. While we cannot exactly understand yet while BaggedTree has the lowest RMSE, it does intuitively make sense that BaggedTree and RandomForest produce similar results as those two algorithms are the most similar.

*Over the last four years, from research and various baseball analytics articles, We have seen that XGBoost is frequently used for predictive baseball models due to its ability to handle complex interactions and non-linearity*

effectively. Given this, We found it surprising that XGBoost performed the worst among the four models, with the highest error and lowest predictive accuracy. This suggests that the default parameters may not be well-suited for my dataset, leading me to investigate further.

Conversely, Bagged Trees performed the best, achieving the lowest error and a relatively high predictive accuracy. This indicates that an ensemble of bootstrapped decision trees might be a strong approach for predicting slugging percentage (SLGValue) based on Exit Speed and Launch Angle. Since Bagged Trees outperformed XGBoost, my next step is to tune XGBoost hyperparameters to determine if better parameter optimization can improve its performance.

After tuning the XGBoost model, the goal is to determine whether optimizing hyperparameters can improve its predictive performance. Given that XGBoost initially performed the worst in the earlier model comparison, adjusting parameters such as learning rate (`eta`), tree depth (`max_depth`), and regularization (`gamma`) should help address potential issues like overfitting or underfitting.

The tuning grid tests different values for key parameters. The number of boosting rounds (`nrounds`) controls how many trees are built, balancing complexity and generalization. Tree depth (`max_depth`) determines how detailed each tree can be, with shallower trees reducing overfitting risk. The learning rate (`eta`) influences how much each tree contributes to the final prediction, where lower values slow learning but improve stability.

Regularization parameters like `gamma` help prune unnecessary splits in trees, preventing overfitting. The model also introduces randomness through `colsample_bytree` and `subsample`, which select subsets of features and rows, respectively, to improve generalization. The `min_child_weight` parameter ensures that splits occur only when a minimum number of observations are in a leaf node, reducing the likelihood of capturing noise.

Once the tuning process is complete, the model selects the best combination of parameters based on cross-validation results. Printing `xgb_tuned$bestTune` provides insight into which values performed the best.

To understand feature importance, `xgb.importance()` ranks `ExitSpeed` and `Angle` based on their contribution to predictions. The importance plot visualizes which variable has the most influence on expected slugging percentage (`xSLG`), helping validate whether the model is learning meaningful relationships.

If tuning significantly improves performance, XGBoost may become a viable option for predicting slugging percentage. Otherwise, Bagged Trees may still be the better choice, confirming the strength of ensemble-based models for this dataset.

```
##      nrounds max_depth  eta gamma colsample_bytree min_child_weight subsample
## 17         150         3 0.05      0              0.85              1      0.85

## [1] "Tuned XGBoost Regression RMSE: 0.87"

## [1] "Tuned XGBoost Regression R^2: 0.286"
```

After training the XGBoost model with optimized hyperparameters, the results still did not outperform the Bagged Tree model. Despite tuning parameters like tree depth, learning rate, and regularization, XGBoost continued to show higher error and lower predictive accuracy than expected.

One possible reason for this could be that XGBoost tends to perform best with larger datasets or when relationships between variables are highly complex and nonlinear. Given that slugging percentage is driven largely by `Exit Speed` and `Launch Angle`, simpler ensemble methods like Bagged Trees may be more effective in capturing these interactions without excessive parameter tuning.

Since the Bagged Tree model had the lowest error and strongest predictive performance in the initial comparison, We are shifting focus to **fully utilizing Bagged Trees** for expected slugging percentage (`xSLG`) predictions. The next steps will involve refining the Bagged Tree approach and applying it consistently across the dataset to generate reliable results.

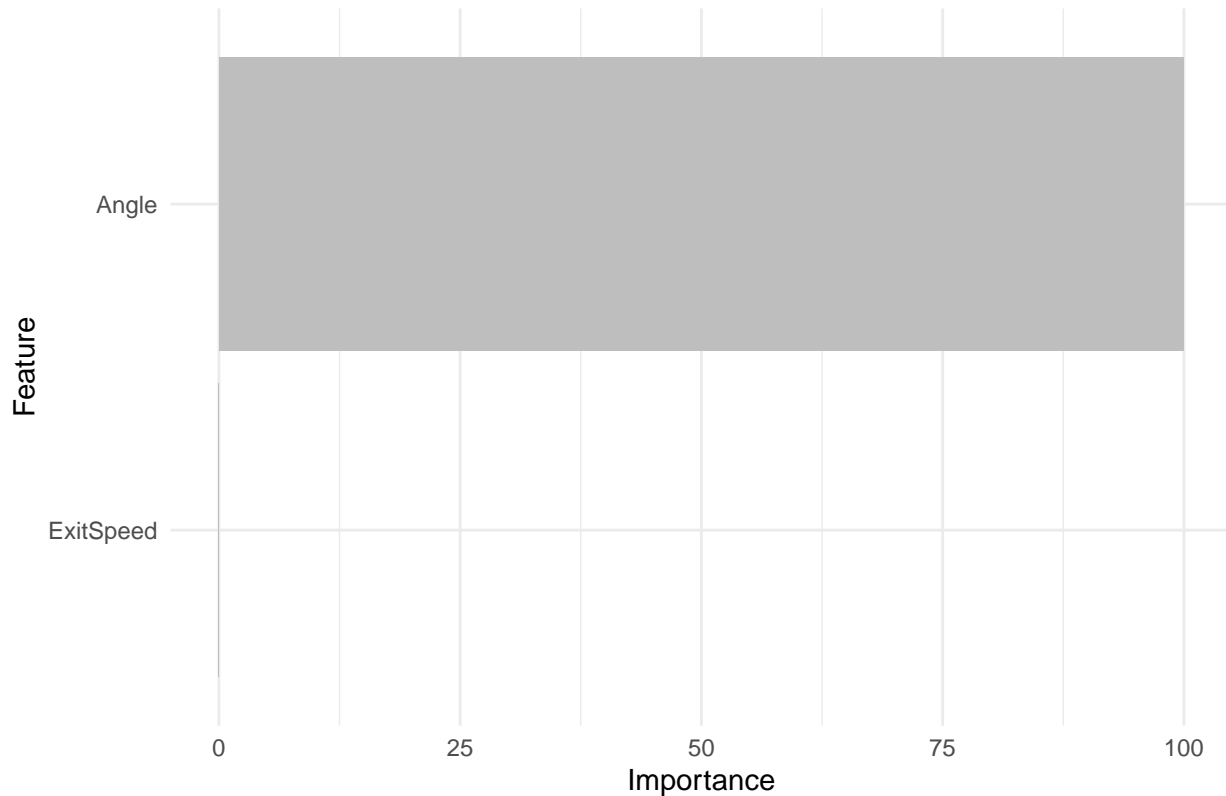
nbagg	RMSE	Rsquared	MAE
10	0.8196697	0.3669798	0.5639179
50	0.8143729	0.3759148	0.5613817

nbagg	RMSE	Rsquared	MAE
100	0.8152554	0.3745467	0.5625929
200	0.8140501	0.3764991	0.5621348

Now we will try to optimize the Random Forest model by tuning hyperparameters like the amount of trees that are made in the model.

After trying to tune the amount of trees in the random forest, we found that 500 trees is the sweetspot that minimizes RMSE (tried 300, 800). We additionally find that out of the 5, mtry = 1 was the best model so we will use that to optimize the model. We will now run feature importance to double check that both angle and exit speed are contributing to the model as intended.

### Feature Importance Plot



So since unlike our xgBoost we find that exitSpeed actually has no use in the random forest AI model, so we will run a forward subset selection to find another variable to predict alongside angle. We will redefine our model with just angle, and use six other remaining variables that are useful. The nature of our data has a lot of categorical variables that do not predict slugging, and these variables make R unable to run the full model in order to carry out a forward variable selection so we have to remove those. Additionally, our data has outcome variables like “Single, Double, Triple” and we cannot include those either because then we would be using what we want to predict in order to predict what we are already predicting.

### Post-hoc subset selection

Additionally, since our data was so large we found that we could only run the full model when we took a random sample of 10% of the data in order to find the best model available.

```
## Start: AIC=-19.62
## SLGValue ~ Angle
```

```

##
##           Df Sum of Sq    RSS      AIC
## + Distance   1    82.228 479.62 -110.193
## + ExitSpeed   1    57.077 504.77  -80.293
## <none>                561.85  -19.623
## + RelSpeed    1     0.827 561.02  -18.485
## + Direction   1     0.460 561.39  -18.103
##
## Step: AIC=-110.19
## SLGValue ~ Angle + Distance
##
##           Df Sum of Sq    RSS      AIC
## + ExitSpeed   1    15.2877 464.33 -127.14
## + RelSpeed    1     2.3443 477.27 -111.06
## <none>                479.62 -110.19
## + Direction   1     0.2135 479.41 -108.45
##
## Step: AIC=-127.14
## SLGValue ~ Angle + Distance + ExitSpeed
##
##           Df Sum of Sq    RSS      AIC
## + RelSpeed    1     3.3850 460.95 -129.42
## <none>                464.33 -127.14
## + Direction   1     0.1671 464.16 -125.35
##
## Step: AIC=-129.42
## SLGValue ~ Angle + Distance + ExitSpeed + RelSpeed
##
##           Df Sum of Sq    RSS      AIC
## <none>                460.95 -129.42
## + Direction   1  0.087941 460.86 -127.53
##
## Call:
## lm(formula = SLGValue ~ Angle + Distance + ExitSpeed + RelSpeed,
##     data = df_sample)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -1.5021 -0.5620 -0.1622  0.4575  2.8672
##
## Coefficients:
##              Estimate Std. Error t value Pr(>|t|)
## (Intercept)  0.1956939  0.5843549   0.335   0.7378
## Angle       -0.0080244  0.0019828  -4.047 5.89e-05 ***
## Distance     0.0030491  0.0004244   7.185 2.07e-12 ***
## ExitSpeed    0.0129200  0.0028504   4.533 7.08e-06 ***
## RelSpeed    -0.0133607  0.0064739  -2.064  0.0395 *
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.8915 on 580 degrees of freedom
## Multiple R-squared:  0.186, Adjusted R-squared:  0.1803
## F-statistic: 33.12 on 4 and 580 DF, p-value: < 2.2e-16

```

Table 3: Comparison of AIC Values for Candidate Models

Model.Stage	Candidate.Addition	AIC
SLGValue ~ Angle	(none)	-19.62
SLGValue ~ Angle	+ Distance	-110.19
SLGValue ~ Angle	+ ExitSpeed	-80.29
SLGValue ~ Angle	+ RelSpeed	-18.49
SLGValue ~ Angle + Distance	(none)	-110.19
SLGValue ~ Angle + Distance	+ ExitSpeed	-127.14
SLGValue ~ Angle + Distance	+ RelSpeed	-111.06
SLGValue ~ Angle + Distance	+ Direction	-108.45
SLGValue ~ Angle + Distance + ExitSpeed	(none)	-127.14
SLGValue ~ Angle + Distance + ExitSpeed	+ RelSpeed	-129.42
SLGValue ~ Angle + Distance + ExitSpeed	+ Direction	-125.35
SLGValue ~ Angle + Distance + ExitSpeed + RelSpeed	(none)	-129.42
SLGValue ~ Angle + Distance + ExitSpeed + RelSpeed	+ Direction	-127.53

Table 4: Key Summary Statistics for Final Model (Angle + Distance + ExitSpeed + RelSpeed)

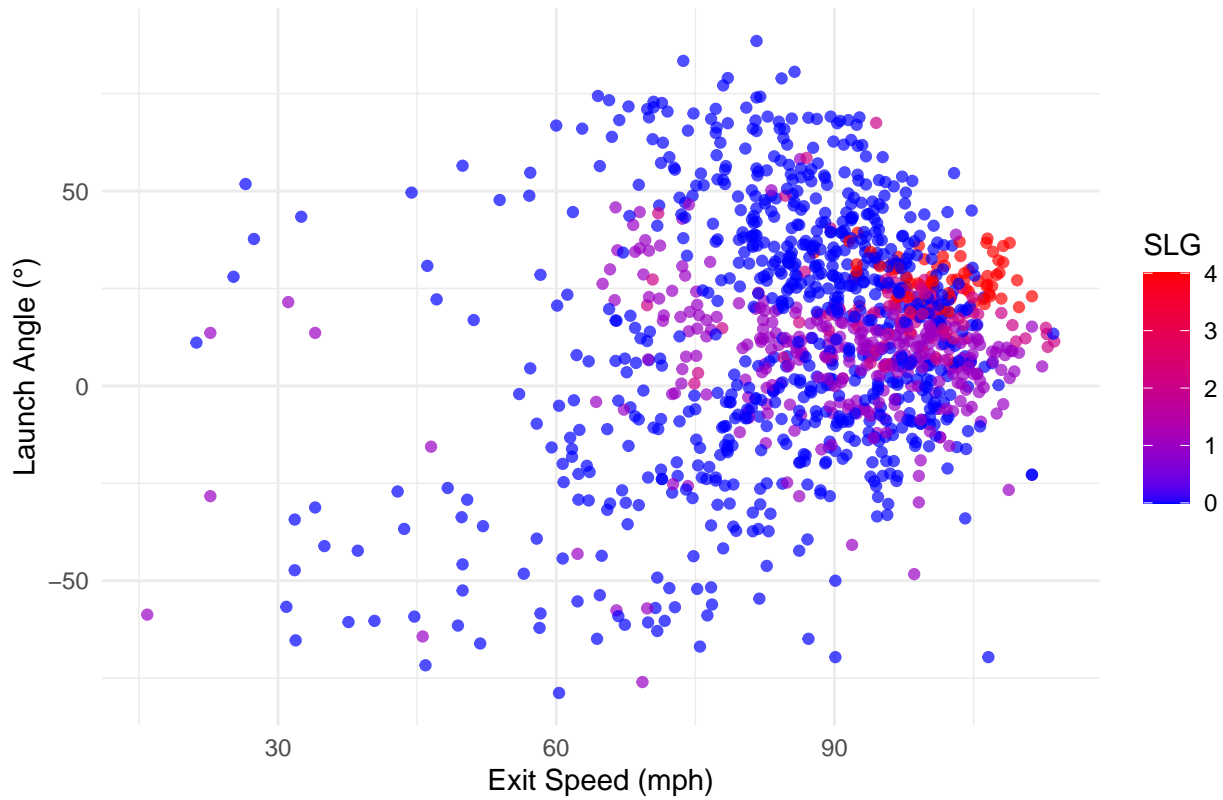
Statistic	Value
Residual Standard Error	0.8915
Multiple R-squared	0.1859
Adjusted R-squared	0.1803
F-statistic (df)	33.12 (4, 580)
p-value	< 2.2e-16

After running this model we see that a combination of angle, distance, exitspeed, and relspeed generate a better prediction for slugging percentage, so we will compare that to the other model of just angle and exitspeed. Of course, release speed is something largely out of control of the batter, as they cannot make the pitcher throw harder or slower, but it is still interpretable and helpful for the hitters because they know when they are facing a pitcher who throws harder or slower.

So this suggests that our optimal model is `SLGValue ~ Angle + Distance + ExitSpeed + RelSpeed` because it generates the lowest AIC.

We made both the original model and the one with the two extra predictors to see how they compare in predicting Slugging percentage.

Actual SLG by Exit Speed and Launch Angle



Predicted xSLG by Exit Speed and Launch Angle (RF Model with 2 Predict

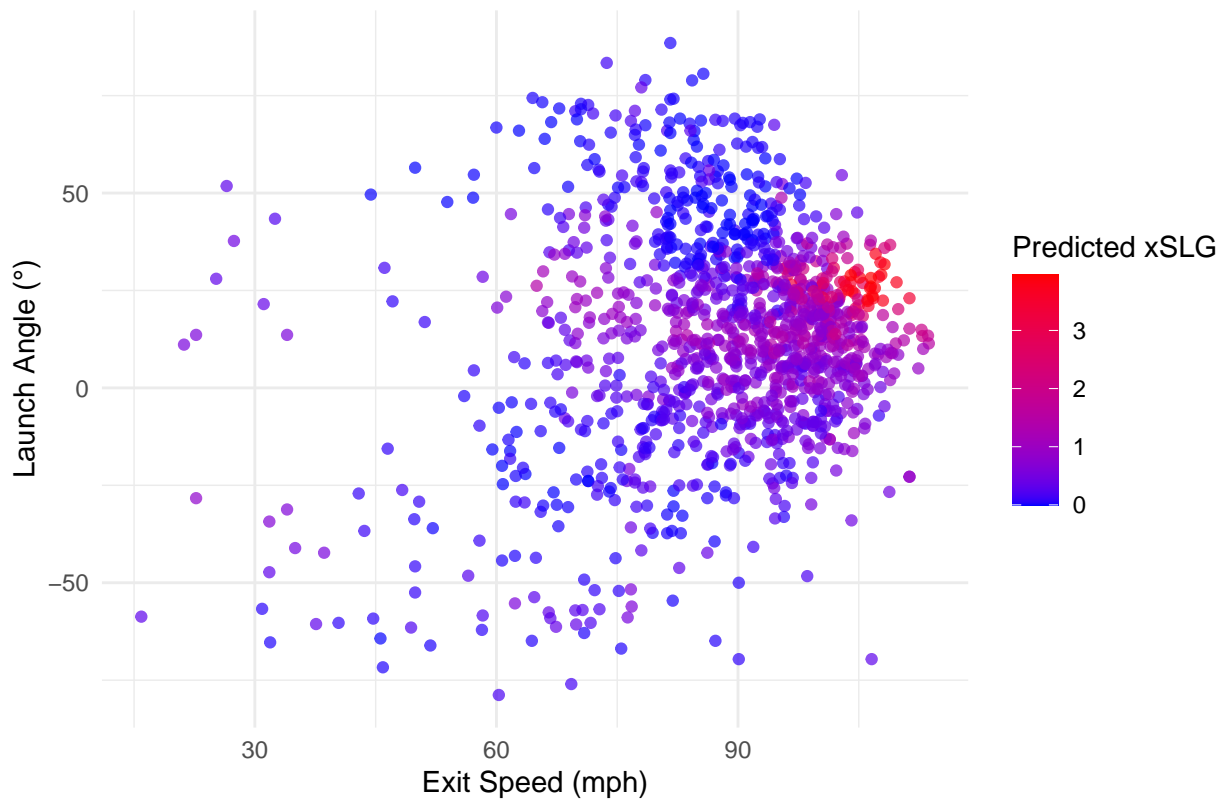




Table 5: Comparison of RMSE Between Random Forest Models

Model	RMSE
Simple RF (2 predictors)	0.8498927
Extended RF (4 predictors)	0.7855394

The first visualization shows the **actual slugging value (SLG)** for each batted ball event based on **Exit Speed and Launch Angle**. The clear trend indicates that **higher exit velocities and optimal launch angles (between 10-40 degrees)** result in higher slugging values, represented by the red-colored points. The second and third graphs represent predicted slugging.

Clearly the second graph matches the actual slugging graph, however the graph with the two variables still produces very similar results to the actual slugging graph. This means that for our application of implementing this model with the players the model with two variables could be more efficient because it is more interpretable for the players.

The dip in RMSE shows that the model with four predictors fits the data better, however the simplicity of the model with two predictors may make more sense for this application so we will proceed with both.

## Predicting rest of season Slugging for UCSB players

We will now have our models use the in progress data we have from this season to have it make predictions on how these players will perform for the rest of the season.

## Summary Table Using the Two-Predictor Model (ExitSpeed + Angle):

Batter	AB	BIP	SLG	xSLG_small	Difference_small
Kim, Isaac	90	69	0.567	0.331	0.236
Crain, Jeramiah	60	38	0.600	0.514	0.086
Vargas, Nathan	157	115	0.459	0.379	0.080
McCollum, LeTrey	153	127	0.444	0.404	0.040
Esquer, Xavier	145	113	0.469	0.450	0.019
Holman, Jack	146	120	0.623	0.606	0.017
Nunez, Corey	136	101	0.265	0.308	-0.043
Mendez, Jonathan	149	112	0.456	0.504	-0.048
Kosciusko, Cole	94	65	0.277	0.392	-0.115
TOTAL	1130	860	0.457	0.433	0.024

##

## Summary Table Using the Four-Predictor Model (ExitSpeed + Distance + Angle + RelSpeed):

Batter	AB	BIP	SLG	xSLG_big	Difference_big
Kim, Isaac	90	69	0.567	0.318	0.249
Crain, Jeramiah	60	38	0.600	0.498	0.102
Vargas, Nathan	157	115	0.459	0.388	0.071
McCollum, LeTrey	153	127	0.444	0.396	0.048
Esquer, Xavier	145	113	0.469	0.426	0.043
Holman, Jack	146	120	0.623	0.608	0.015
Mendez, Jonathan	149	112	0.456	0.451	0.005
Nunez, Corey	136	101	0.265	0.308	-0.043
Kosciusko, Cole	94	65	0.277	0.341	-0.064
TOTAL	1130	860	0.457	0.417	0.040

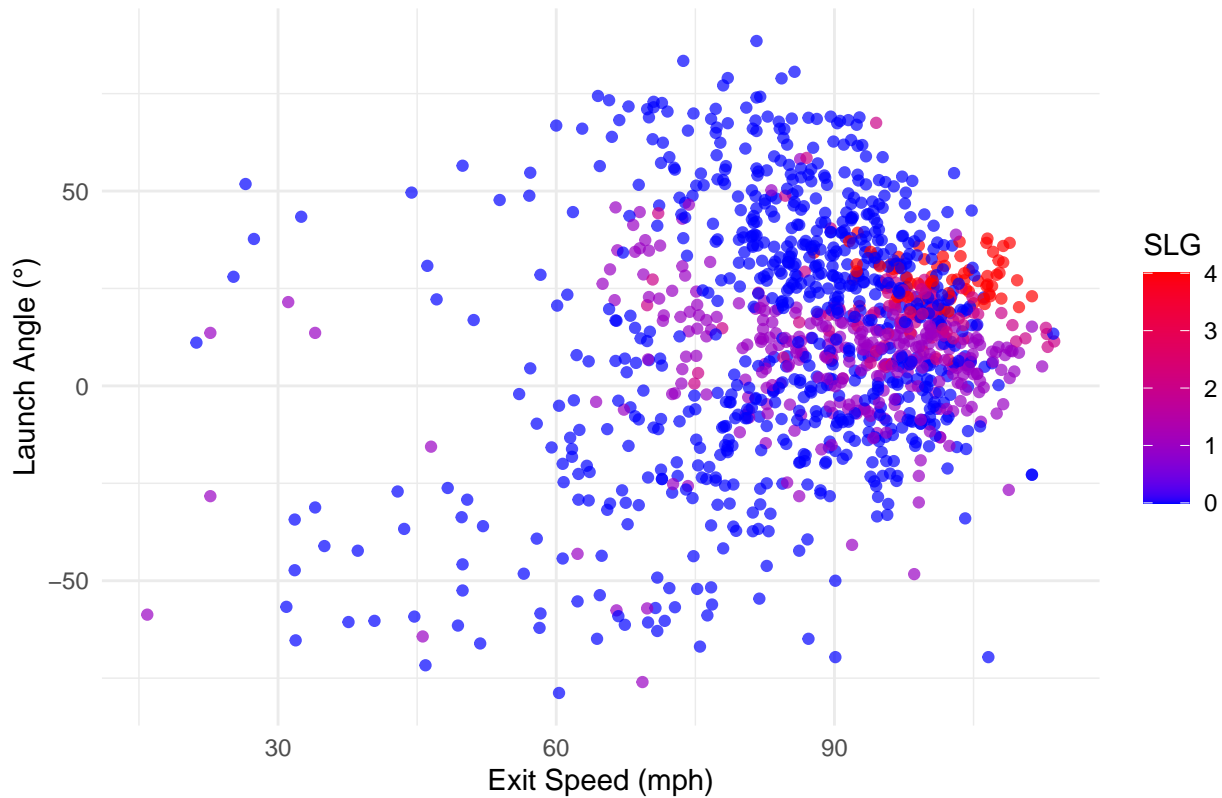
## Visualizations and predictions for BaggedTree model

Since tuning did not provide a better alternative, We will proceed with the **original Bagged Tree model** for: - **Creating a visualization** comparing actual slugging values for each hit versus expected slugging values. - **Calculating expected SLG (xSLG)** on the test dataset to assess model performance. - **Predicting xSLG for the 2025 dataset**, generating insights into future player performance.

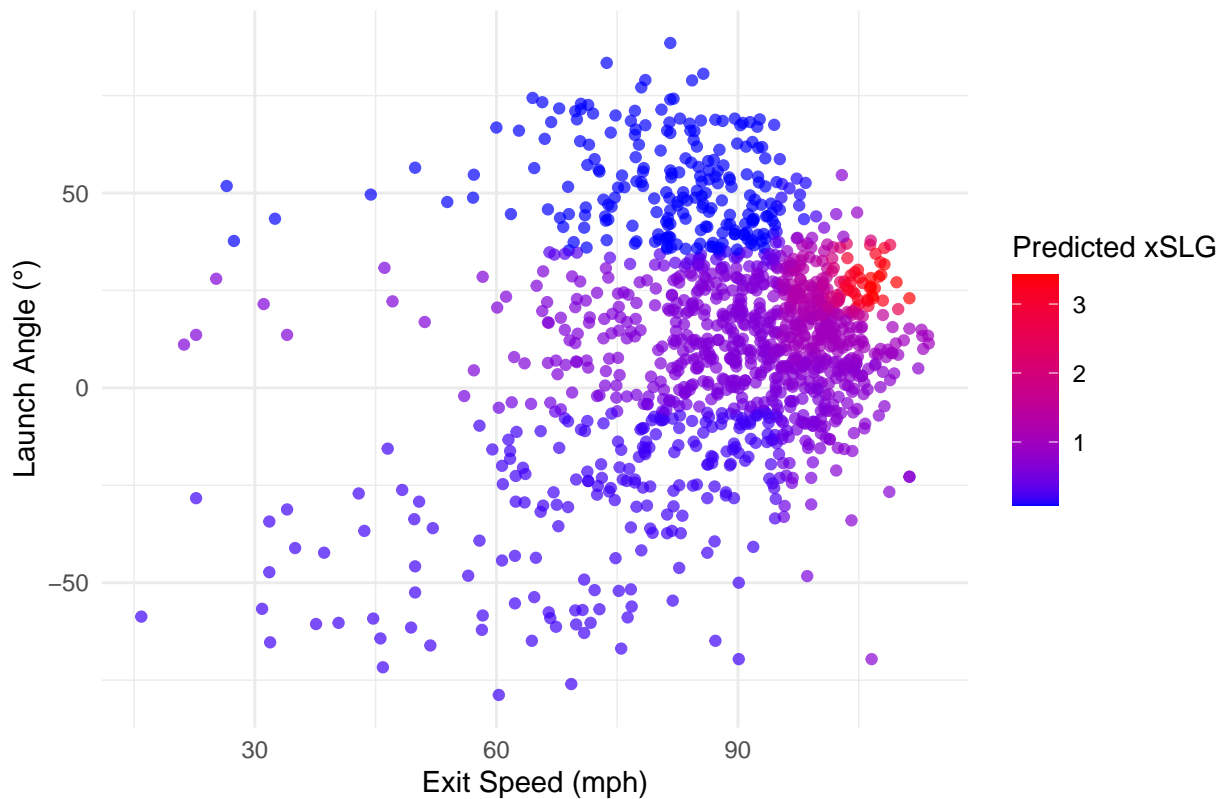
With this approach, We can ensure that my predictions remain accurate and interpretable while leveraging the best-performing model from my initial comparisons.



Actual SLG by Exit Speed and Launch Angle



Predicted xSLG by Exit Speed and Launch Angle (Bagged Tree)



The second visualization represents the **predicted expected slugging value (xSLG)** using the **Bagged**

**Tree model.** The model captures the general trend of actual SLG, correctly identifying areas where slugging percentage is expected to be higher. However, some differences exist between the actual and predicted values, particularly in extreme cases where SLG is either very high or very low.

By comparing these two plots, We can assess how well the model generalizes **batted ball outcomes** and identify potential areas where adjustments could improve predictions. The next step is to use this model to **calculate xSLG for the test dataset and extend predictions to the 2025 dataset**, allowing for a better understanding of player performance and expected outcomes.

Batter	BIP	SLG	xSLG	Difference
Oakley, Nick	57	0.702	0.703	-0.001
Brethowr, Ivan	38	0.474	0.802	-0.328
Darby, Zander	38	0.711	0.585	0.126
Parker, Aaron	36	0.889	0.581	0.308
Kirtley, Christian	34	0.676	0.632	0.044
Nunez, Corey	32	0.156	0.451	-0.295
Sprinkle, Jordan	31	0.484	0.490	-0.006
McCollum, LeTrey	28	0.571	0.599	-0.028
Willits, Bryce	28	0.536	0.480	0.056
TOTAL	1248	0.624	0.625	-0.001

The table summarizes **actual slugging percentage (SLG)** and **expected slugging percentage (xSLG)** for each batter with at least 28 balls in play (BIP). This threshold ensures that only players with a meaningful sample size are included in individual comparisons, while the **TOTAL row accounts for all batters** in the dataset.

Since the **TOTAL SLG and xSLG are equal**, this confirms that the model is calibrated correctly across the dataset. The **Difference column** highlights whether a player is overperforming ( $SLG > xSLG$ ) or underperforming ( $SLG < xSLG$ ) relative to expected outcomes based on **Exit Speed and Launch Angle**.

- **Positive Difference:** The batter has a higher actual SLG than expected, possibly due to favorable defensive positioning, small sample variance, or other unmodeled factors.
- **Negative Difference:** The batter's actual SLG is lower than expected, suggesting potential bad luck, strong defensive plays, or a need for adjustments in approach.

This table helps identify **which hitters may sustain their performance** and which ones might **regress positively or negatively** based on expected outcomes.

```
## [1] "Before filtering: Rows = 1130"
## [1] "Missing ExitSpeed = 344"
## [1] "Missing Angle = 344"
## [1] "After filtering BIP: Rows = 786"
```

Batter	AB	BIP	Strikeouts	AvgEV	BABIP	SLG	xSLG	Difference
Kim, Isaac	90	69	21	83.2	0.377	0.567	0.365	0.202
Crain, Jeremiah	60	38	22	96.8	0.421	0.600	0.483	0.117
Vargas, Nathan	157	115	42	89.0	0.339	0.459	0.414	0.045
Holman, Jack	146	120	26	90.4	0.342	0.623	0.586	0.037
McCollum, LeTrey	153	127	26	89.7	0.386	0.444	0.422	0.022
Esquer, Xavier	145	113	32	89.5	0.265	0.469	0.460	0.009
Mendez, Jonathan	149	112	37	88.2	0.321	0.456	0.502	-0.046
Nunez, Corey	136	101	35	86.1	0.307	0.265	0.325	-0.060
Kosciusko, Cole	94	65	29	88.4	0.277	0.277	0.364	-0.087

Batter	AB	BIP	Strikeouts	AvgEV	BABIP	SLG	xSLG	Difference
TOTAL	1130	860	270	88.8	0.333	0.457	0.440	0.017

This table provides a breakdown of **actual slugging percentage (SLG)**, **expected slugging percentage (xSLG)**, and **additional hitting metrics** for UCSB Baseball's main hitters across the **2024 Fall and 2025 season**. The purpose of this analysis is to **predict future performance trends** and identify which players may be overperforming or underperforming based on expected outcomes.

Strikeouts were included since they are accounted for in SLG but were **excluded from xSLG predictions model** because they always result in zero total bases. **BABIP (Batting Average on Balls in Play)** is also included, as it provides additional insight into whether a hitter has been particularly lucky or unlucky in their batted ball outcomes.

From the table, **Isaac Kim** has the highest positive difference between SLG and xSLG, indicating he may be overperforming. This suggests he has accumulated **a lot of weak base hits or BIP that are normally outs**, meaning he is **likely to regress** over the rest of the season. Similarly, **Jack Holman** also has a **positive difference**, likely influenced by his **six home runs in the first two weeks of the season**, an extremely rare feat that suggests he is **due for some regression** as well.

Conversely, the **unluckiest hitter is Cole Kosciusko**, who has a **mid-tier average exit velocity**, suggesting that **soft contact is not the issue**. Instead, he has had **many well-hit BIP that should have been hits but resulted in outs or fewer bases than expected**, meaning his performance may improve as the season progresses.

---

## Future Considerations for Improving Expected SLG Models

While this model provides valuable insights into expected performance, **there are ways it can be improved** in the future. One key limitation is that **MLB has advanced tracking technology** that includes **baserunner speed data**, which plays a significant role in expected stats. A **slower runner is less likely to beat out throws or take extra bases**, meaning that **xSLG calculations in professional baseball are more refined**.

However, **this technology has not yet been implemented at the college level**. As tracking systems continue to evolve, **colleges could integrate baserunning metrics** into expected stat models, making projections more accurate. In future years, if this data becomes available, it could be incorporated into **college-level player evaluations** to better assess player strengths, weaknesses, and potential trends in performance.

For applying this model, the lowest RMSE was generated from the Random Forest model with four predictors, however between the models with two predictors the baggedTree was better than the Random Forest model with two predictors, so for simplicity it still may be worthwhile to present the players with the baggedTree model. We will also track these slugging predictions as the season progresses to see which model was more accurate.