Pitch Classification Models

Brett Castro - Paulo Julio

University of California, Riverside

bcast107@ucr.edu pjuli003@ucr.edu

https://github.com/brettcastro2004/Pitch-Classification-Model

https://colab.research.google.com/drive/1tYqSuoRPoAPq1qC2YrK-uu9rYO5SC_Bn?usp=sharing

This is a pitch classification model and analysis featuring data cleaning, normalization, correlation and feature visualization, KNN, ANN, PCA, and an ensemble model. Additional information for the dataset and source code access are available in the referenced repository.

Background

This project involves building classification models that examine pitch characteristics and labels a given pitch into one of 8 classes. We utilize pitching data provided by MLB's Statcast technology through Baseball Savant. The model will be evaluated based upon its ability to accurately classify the type of pitch thrown given its physical behavior. Given the variability of pitcher characteristics present in MLB, our goal is to create a generalized model that is able to classify pitches with moderately high accuracy.

Sam Sharpe writes about MLB's own patented classification system in a breakdown article, "MLB pitch classification". He describes the multi-layered model, PitchNet, which first runs the pitch through a generic pitching model, then a pitcher specific model that compares the predicted pitch to the player's repertoire, and also considers other factors such as weather and climate data in its prediction. That model utilizes both supervised and unsupervised learning to achieve a 97% or higher accuracy. Our model intends to fulfill only the purpose of the first layer of the PitchNet system, a generic pitch model that labels a pitch given purely mechanical properties of the pitch.

Dataset Description

To ensure a high quality set of data, we use Baseball Savant. Baseball Savant is directly owned by MLB and is the home for Statcast, a database for every game, at-bat, and pitch thrown in an entire season, amounting to hundreds of millions of data points. It is accessible here:

https://baseballsavant.mlb.com/statcast_search

The full implementation of the Statcast system is explained in the article "Introducing Statcast 2020: Hawk-Eye and Google Cloud" by Ben Jedlovec. In short, this system uses cameras and radar technology to track swings, base runners, and thrown or hit balls.

The dataset is tabular and includes detailed measurements of individual pitches, with over 100 tracked variables for each pitch. We intend to use only 5 features and 40,000 pitches to train the model, with 5000 instances per pitch type. This sample size gives us data from hundreds of pitchers and keeps us from considering too much noise or over-training the models.

The features we will be utilizing for classification include: velocity, spin rate, horizontal movement (pfx_x), vertical movement (pfx_z), and spin axis. To fully understand what these feature labels mean in the context of a baseball game, we encourage looking through the official MLB documentation:

https://baseballsavant.mlb.com/csv-docs

Though there are more pitches in existence, today's game sees very few in play. The pitches we will consider for labeling are four-seam fastball [FF], sinker [SI], sweeper [ST], slider [SL], cutter [FC], curveball [CU], splitter [FS], and changeup [CH].

We will train 2 models, a K-nearest neighbors model and an Artificial Neural Network. We will also conduct PCA to reduce dimensions and observe how accurate the models are in reduced feature space. To conclude, an ensemble model composed of both the KNN and the ANN models will be created, with the objective of achieving the highest accuracy of all models.

To begin, we analyze the features themselves. We start by examining correlation between features. After seeing how the features relate to one another, we will create a hypothesis for how the models might behave.

Data Gathering and Normalization

We gather 5000 clean samples of data from each of the 8 pitch type datasets (originally containing 25000 pitches each) for 40,000 total pitches.

pitch_type	player_name	release_speed	pfx_x	pfx_z	release_spin_rate	spin_axis	p_throws
FF	Sears, JP	90.7	0.83	1.11	1895.0	132.0	L
FF	Sears, JP	90.7	0.66	1.28	1878.0	131.0	L

Figure 1: Sample of data before normalization

Identifying and Handling Missing Data

- Due to how precise pitches are, we cannot classify any pitches that are missing data. Thus, we remove any pitches that are missing data for any of our 5 features from the working dataset.

We normalize the data to consider all pitchers as right handed (in order to avoid any issues due to differences between right and left handed pitchers). We normalize left handed pitchers into right handed pitchers by flipping all horizontal movement for left handed pitchers (positive values all become negative and vice versa). We then mirror the spin axis in degrees using (new axis = (360 - old axis) % 360). All data is normalized using z-score normalization. The resulting DataFrame is stored as a csv file.

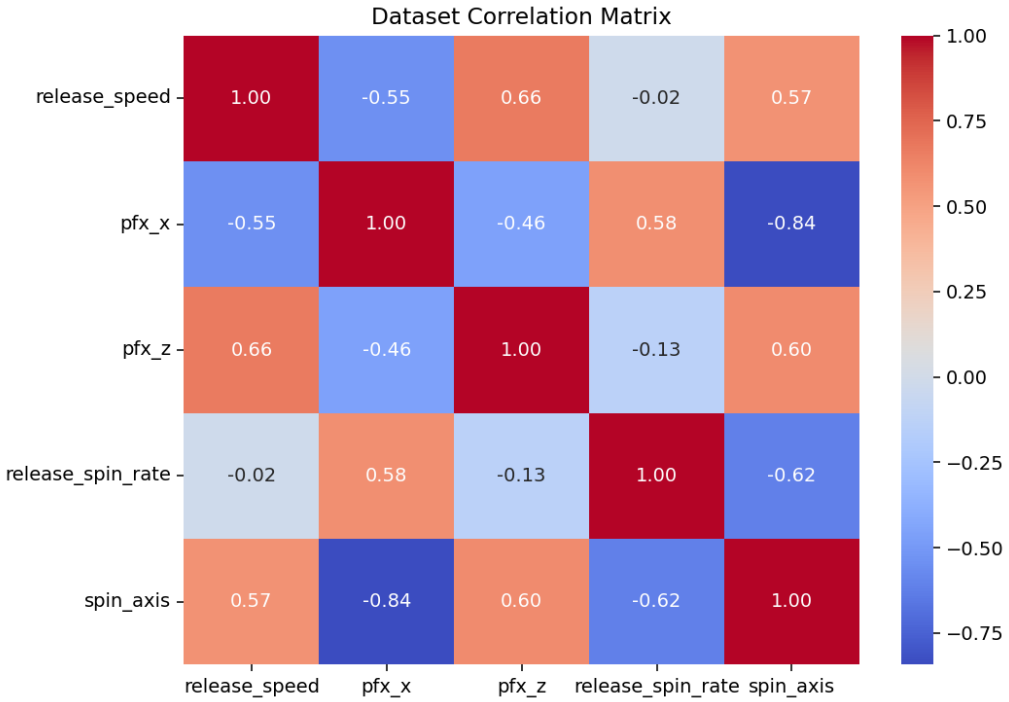
pitch_type	player_name	release_speed	pfx_x	pfx_z	release_spin_rate	spin_axis	p_throws
FF	Sears, JP	0.5659711320019010	-0.686306400815928	1.146563805484540	-0.6510898625810860	-1.1024996436555400	L

Figure 2: Sample of data after normalization

Dataset Analysis: Correlation and Visualization

We use a heatmap to visually display the linear correlation between features. We see that there is a strong positive correlation between vertical movement (pfx_z) and release speed, spin rate and horizontal movement (pfx_x), release speed and spin axis, and spin axis and vertical movement. In contrast, there is a strong negative correlation between horizontal movement and spin axis, release speed, and vertical movement, as well as spin axis and spin rate. Release speed and vertical movement both have a weak correlation with spin rate.

We examine three of these feature relationships to form a hypothesis describing that different pitch types which tend to behave similarly might be confused.



Spin Axis vs Horizontal Break

We examine the effect of spin axis on horizontal movement, the most negatively correlated relationship. In particular, we are looking at how certain pitches move towards or away from the pitcher's arm side, and how spin axis relates to that movement. To clarify how spin axis is defined, we include a chart that graphically displays spin axis angles, provided by Driveline Baseball:
https://www.drivelinebaseball.com/wp-content/uploads/2019/09/spin_axis_green_line.webp

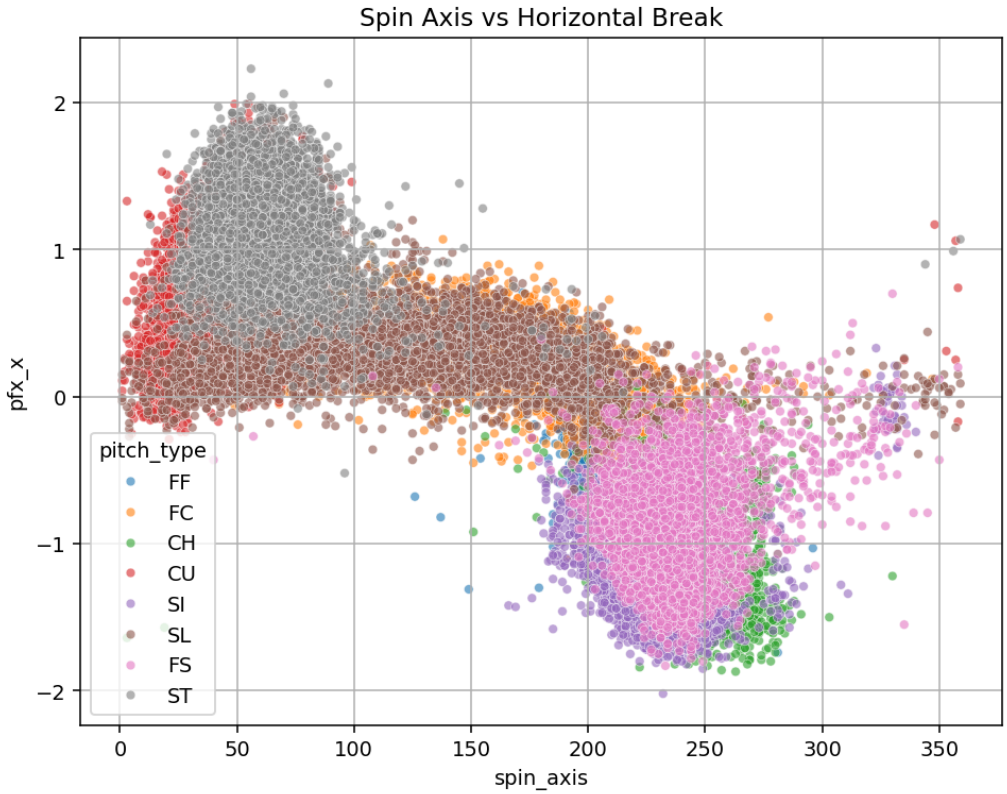


Figure 3: Spin Axis vs Horizontal Movement

From the graph, we gather that spin axis and horizontal movement can be described as having a negative linear correlation overall. This description is not entirely accurate though. The graph is asymptotic and shaped like a graph of cotangent. The graph indicates from feature relationships is that a pitch thrown with a more positive spin axis tends to break more negatively horizontal (we describe this as "arm side" movement). A pitch thrown with a less positive spin axis breaks more positively horizontal (called "glove side" movement").

The graph demonstrates a significant overlap among many pitch clusters. Pitches such as CH, FS, and SI break to the pitchers "arm side," and tend to have a lot of overlap. We see pitches that break to the "glove side" includes CU, ST, FC, and SL. FF are less distinct compared to other clusters, and in baseball are intended to not break significantly on the horizontal axis, so their behavior is mostly pitcher dependent, but generally close to 0 induced movement.

Velocity vs Vertical Movement

We now examine the relationship between velocity and vertical movement.

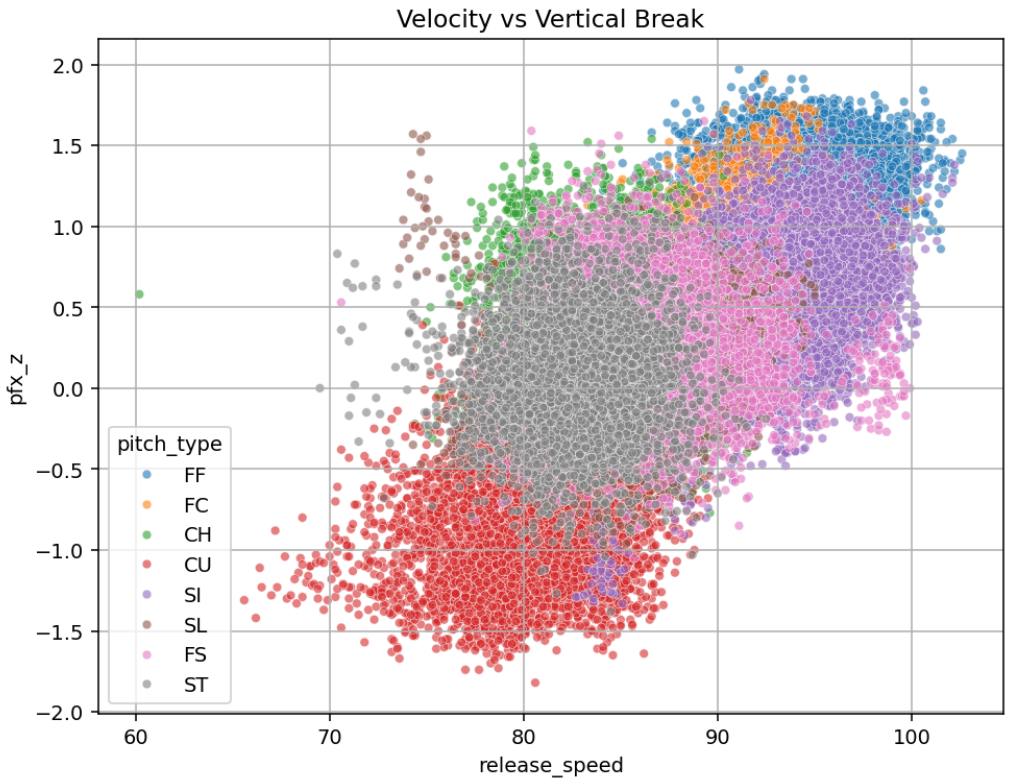


Figure 4: Velocity vs Vertical Break

The slowest pitch, CU, breaks downwards more than any other pitch. FF appears to rise slightly due to its backspin paired with its high velocity, fighting gravity while in motion. Most other pitches tend to move vertically to varying degrees, with a clear positive linear relationship between velocity and vertical break overall. Distinguishable clusters are maintained.

Spin Rate vs Induced Break

Examining the relationship between spin rate and the magnitude of induced break, defined as

$$\text{Total Induced Break} = \sqrt{(\text{pfx\_x}^2 + (\text{pfx\_z})^2},$$

allows us to see how the spin rate on a ball relates to its overall movement.

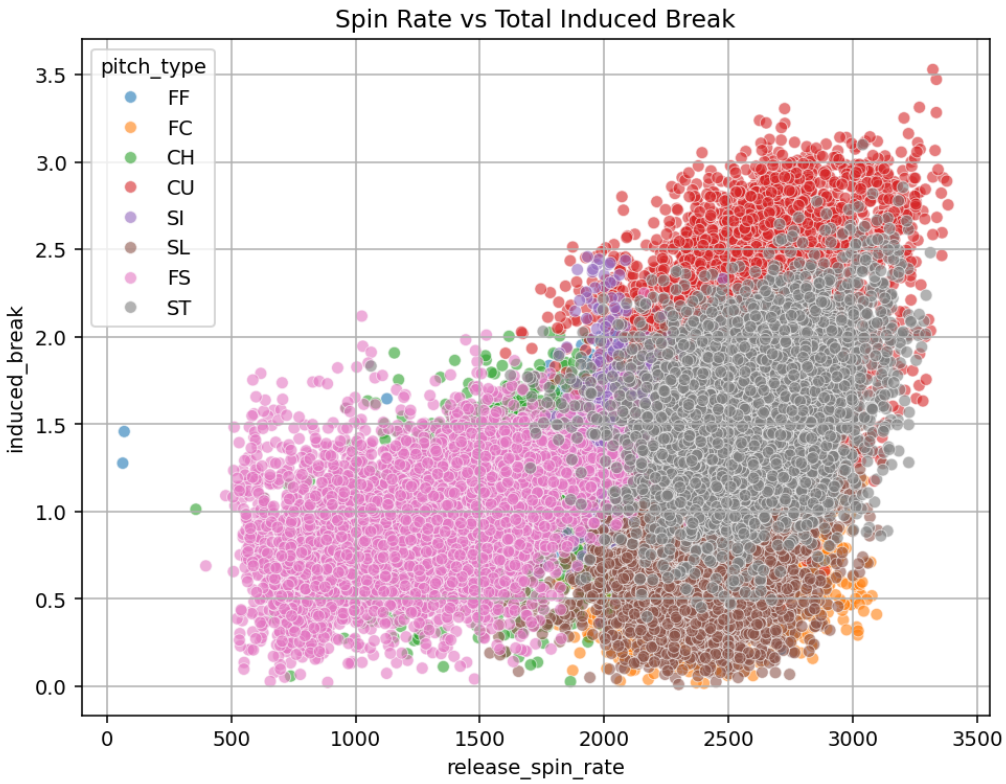


Figure 5: Spin Rate vs Induced Break

CU exhibit significant vertical break, up to 36 inches, the most of any pitch type, and also have the highest spin rate, with reaching over 3000 rpm. FS and CH have very low spin rates, typically around 1200 rpm. These pitches can move approximately 18 inches. SI are more tightly confined, with an average spin rate around 2500 rpm, and average break at 15-18 inches.

Analyzing this in context gives us an understanding of how break can be produced in different ways. CU break while having a very high spin rate, while FS and CH break with a very low spin rate. These differences will provide our models with a solid foundation while learning pitch patterns.

Model Strength and Weakness Hypothesis

Examining relationships between features in context allows us to form a hypothesis about our classification models. We hypothesize that highly distinguishable pitches, like CU and FF, will be labeled the most accurately. Pitches that overlap with other clusters, such as CH and FS will be confused for one another. We hypothesize that the following pitch groups will be most affected by overlap: CH/FS, FC/SL, SL/ST.

Training Models

MLB uses a proprietary system called PitchNet to classify pitches in real time. PitchNet utilizes gradient boosting in addition to individual pitcher model, layered with ballpark specific weather and climate data to create a prediction of pitch type. This model will consist of only a generic pitcher model that looks purely at physical ball behavior and generates a label.

Our performance objective for this generic model is to reach at least 80% accuracy. We train and test two separate models. The first model will use K Nearest Neighbors, the second will be an Artificial Neural Network.

We set up a training and test data split, designating 4000 samples as training and 1000 as test data for each pitch type. This is an 80/20 split across all eight classes, with a fixed random seed used to ensure a reproducible split. The full, final configuration profile for all models used is included here:

Table 1: Summary of final model configurations

Model	Configuration
KNN	$k = 9$, distance metric: Euclidean, weights: distance
ANN	Layers: [128, 64, 32, 8]; activations: ReLU (hidden), Softmax (output); optimizer: Adam; batch size: 128; epochs: 100
PCA	2 components, fitted on normalized training features
Ensemble	Probability-weighted average with $\alpha = 0.5$

K-Nearest Neighbors (KNN)

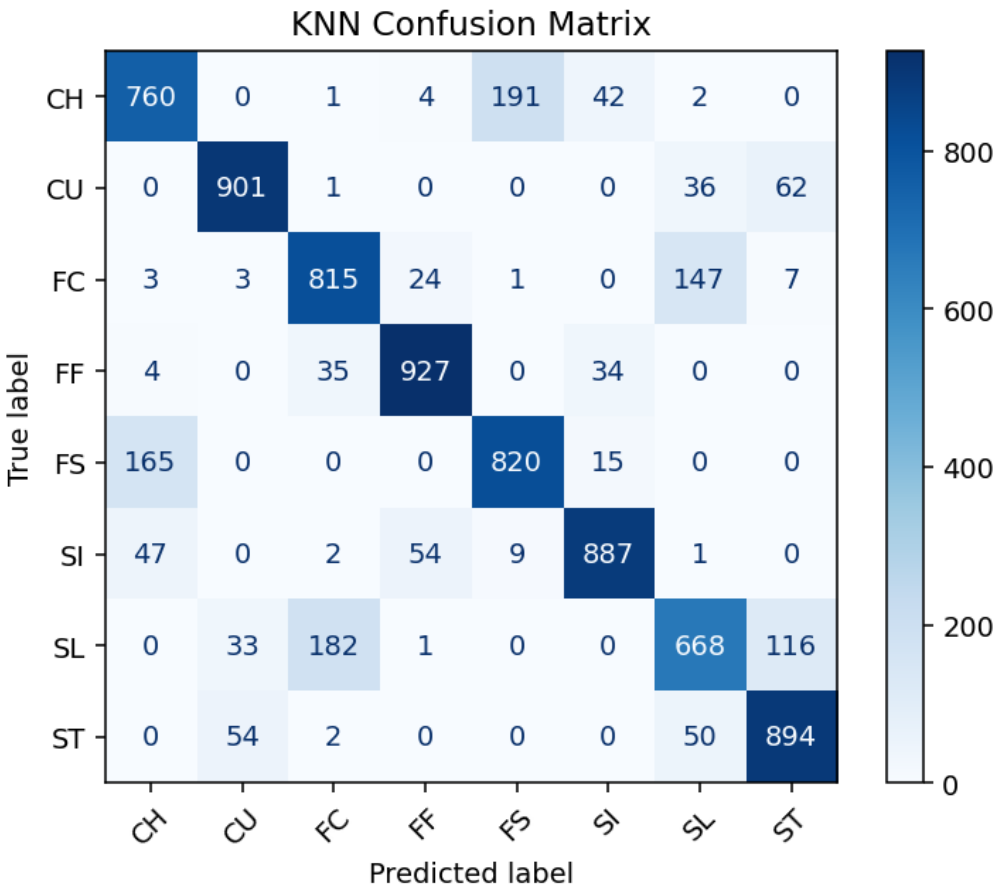
KNN compares a given unclassified pitch with K pitches that have similar features to the given one. The model looks at the most similar pitches to the unclassified pitch in 5 dimensional space, then takes a vote for the labeling of that pitch, ultimately decided by the label with a plurality of votes.

We train KNN via sklearn with a starting selection of K = 7, a standard value.

```
KNN Accuracy: 0.8340
KNN Precision:, 0.8333
KNN Recall:, 0.8340
KNN F1-score:, 0.8332

Classification Report (KNN):
              precision    recall  f1-score   support

          CH       0.78      0.76      0.77      1000
          CU       0.91      0.90      0.91      1000
          FC       0.79      0.81      0.80      1000
          FF       0.92      0.93      0.92      1000
          FS       0.80      0.82      0.81      1000
          SI       0.91      0.89      0.90      1000
          SL       0.74      0.67      0.70      1000
          ST       0.83      0.89      0.86      1000

    accuracy                           0.83      8000
   macro avg       0.83      0.83      0.83      8000
weighted avg       0.83      0.83      0.83      8000
```

This model achieves 83.40% accuracy on our test data. Looking at the individual pitch types, we see variance in accuracy. FF are being classified correctly the most, followed by CU, ST and SI. Other breaking and off-speed pitches are being confused more, with SL being misclassified the most.

Our precision, recall and f1 scores all indicate that the model is classifying the pitches with features that did not have excessive cluster overlap the best. We can consider the confusion matrix of this model to see which pitches are being improperly labeled.



Consistent with our hypothesis, the three pitch groups, CH/FS, FC/SL, and ST/SL are being confused the most. In particular, FS are favored over CH, FC over SL, and ST over SL. We also see some confusion between SI and FF, as well as FF and FC. This aligns with expected behavior, as all three are in the fastball family, and will share some characteristics.

KNN Optimization

Choosing an appropriate K value is essential, as it influences the model's tradeoff between bias and variance. Small K values lead to overfitting and higher variance, while large K values can increase bias and lead to an oversmoothing on class boundaries. To identify a balanced value, testing is conducted with the same data split, and various K values.

We test values between 3 and 21 and record the classification report of each configuration.

```
Test K =  9

KNN Accuracy: 0.8361
KNN Precision:, 0.8352
KNN Recall:, 0.8361
KNN F1-score:, 0.8351

Classification Report (KNN):
              precision    recall  f1-score   support

          CH       0.79      0.76      0.78      1000
          CU       0.91      0.90      0.91      1000
          FC       0.78      0.81      0.80      1000
          FF       0.92      0.93      0.92      1000
          FS       0.81      0.83      0.82      1000
          SI       0.90      0.89      0.90      1000
          SL       0.74      0.66      0.70      1000
          ST       0.83      0.90      0.86      1000

    accuracy                           0.84      8000
   macro avg       0.84      0.84      0.84      8000
weighted avg       0.84      0.84      0.84      8000
```

Figure 6: KNN Confusion Matrix with K = 9

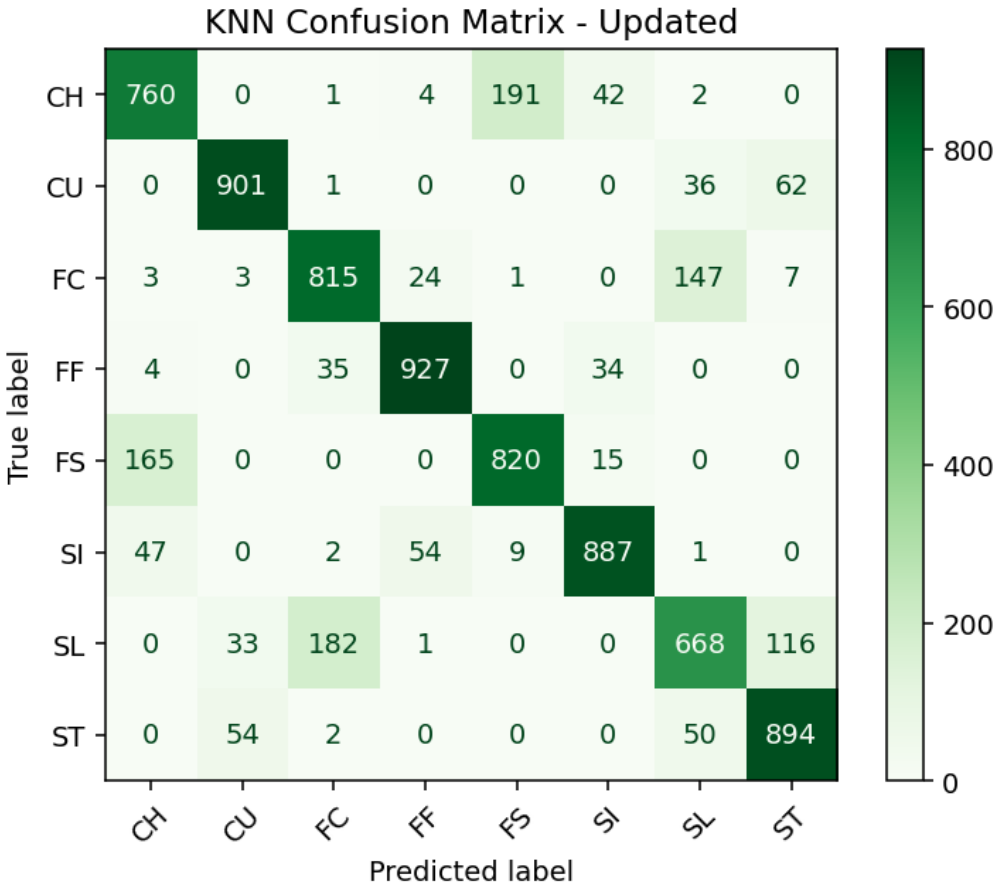


Figure 7: KNN Confusion Matrix

After conducting this range of values, we see a noteworthy trend. Accuracy increases from K = 3 to K = 9. As K approaches 9, accuracy peaks with K = 9 at 83.61%, accuracy then decreases until K

```
Test K =  16

KNN Accuracy: 0.8383
KNN Precision:, 0.8374
KNN Recall:, 0.8383
KNN F1-score:, 0.8372

Classification Report (KNN):
              precision    recall  f1-score   support

          CH       0.79      0.77      0.78      1000
          CU       0.92      0.90      0.91      1000
          FC       0.79      0.82      0.80      1000
          FF       0.91      0.93      0.92      1000
          FS       0.82      0.83      0.82      1000
          SI       0.90      0.89      0.89      1000
          SL       0.75      0.67      0.71      1000
          ST       0.82      0.91      0.86      1000

    accuracy                           0.84      8000
   macro avg       0.84      0.84      0.84      8000
weighted avg       0.84      0.84      0.84      8000
```

= 12. From K = 12, accuracy increases until it peaks at 83.83% when K = 16. The confusion matrix will show where the improvements were achieved, and allows us to confirm if overfitting is occurring.

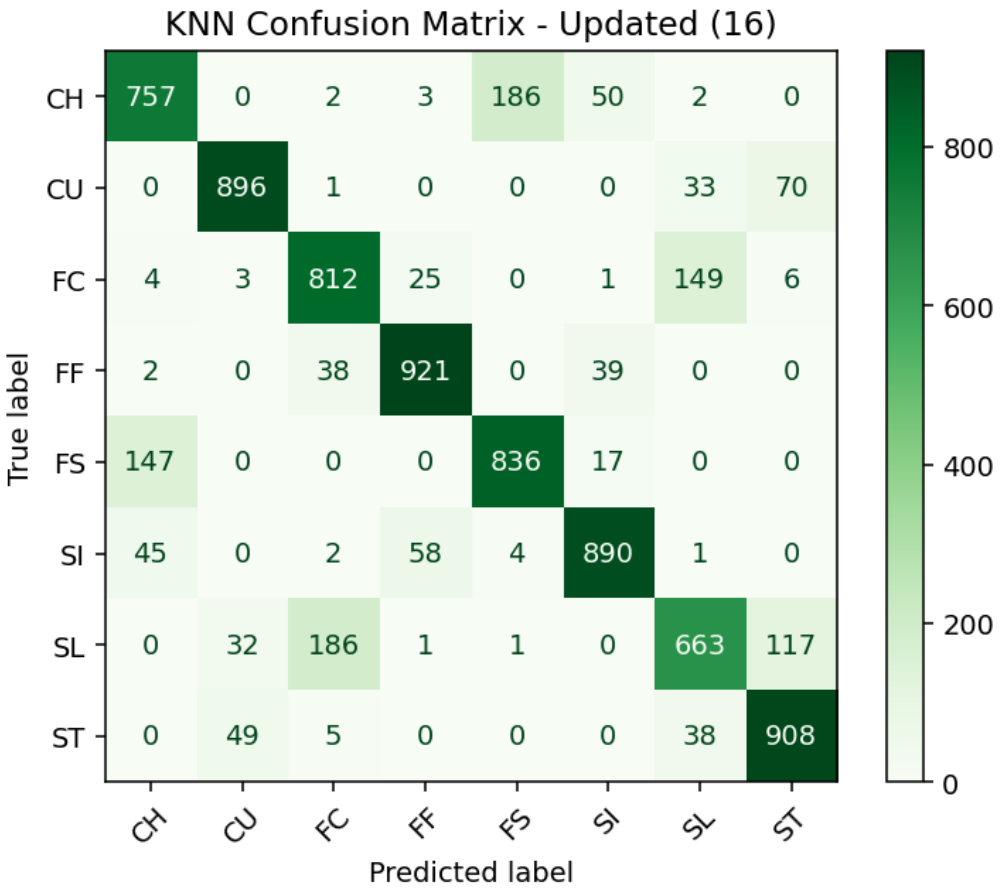


Figure 8: KNN Confusion Matrix with K = 16

There is a noticeable improvement for classifying ST and FS but a marked regression for all other

pitch types. This suggests overfitting is occurring, as the model is now considering more data than is useful for classifying most pitch types. The most balanced K value appears to be 9, as it achieves the most broadly accurate results without overfitting, and we select it for all subsequent modeling.

Artificial Neural Network (ANN)

We move to the second model, an ANN with 4 layers. The first dense layer has 128 neurons, the second 64, the third 32, and the final layer is the 8 neuron SoftMax output layer. Each layer deeply analyzes patterns within the data, specifically looking at how the features interact. For example, layer 1 may look at how fastballs tend to have the highest velocity and high spin rate. It may also notice that CU also have high spin rate, but low velocity. The model uses these types of relationships to distinguish pitch types and improve accuracy throughout the training process.

As the pitch goes to the next layer, the model examines patterns and refines itself via its activation function, ReLU. In the final layer, a new activation function, Softmax, considers the outputs of the previous dense layer and converts it into a list of probabilities. The label with the highest probability is chosen. During training, the loss function, binary crossentropy, measures the difference between the predicted label probability from the true label, and is used when the optimizer updates the model weights. We can use loss, accuracy, and the overall classification report to determine how the model is performing. The model is based off of the provided sample code. After trying various combination of neurons, learning rates, activation functions, and batch sizes, the consistently highest performing parameters are used.

One distinction to note from the sample code is the activation function used. Instead of the sigmoid function, we use Softmax for multi-class classification. Our model assigns each pitch exactly one label from the 8 potential choices. Softmax is specifically designed for cases such as this one. In the article "Multi-label classification for beginners with codes," Mehul Gupta explains this, "Sigmoid is preferred for multi-label classification because it allows for independent probabilities for each label, while softmax is designed for exclusive class assignments, such as in multi-class classification."

```
ANN Test Accuracy: 0.8341
250/250 ━━━━━━━━━━━━━━━━━━━━ 0s 1ms/step

Classification Report:
              precision    recall  f1-score   support

          CH       0.76      0.80      0.78      1000
          CU       0.90      0.91      0.90      1000
          FC       0.78      0.79      0.79      1000
          FF       0.91      0.94      0.93      1000
          FS       0.84      0.79      0.81      1000
          SI       0.92      0.87      0.90      1000
          SL       0.72      0.69      0.70      1000
          ST       0.84      0.88      0.86      1000

    accuracy                           0.83      8000
   macro avg       0.83      0.83      0.83      8000
weighted avg       0.83      0.83      0.83      8000
```

Accuracy is very slightly lower than the updated KNN model at 83.41%, and 0.01% higher than the initial KNN model. This model behaves very similar to KNN, where we observe a high accuracy of for CU, FF, and SI. We see that the accuracy of some pitch types such as CH and SL is also low in this model.

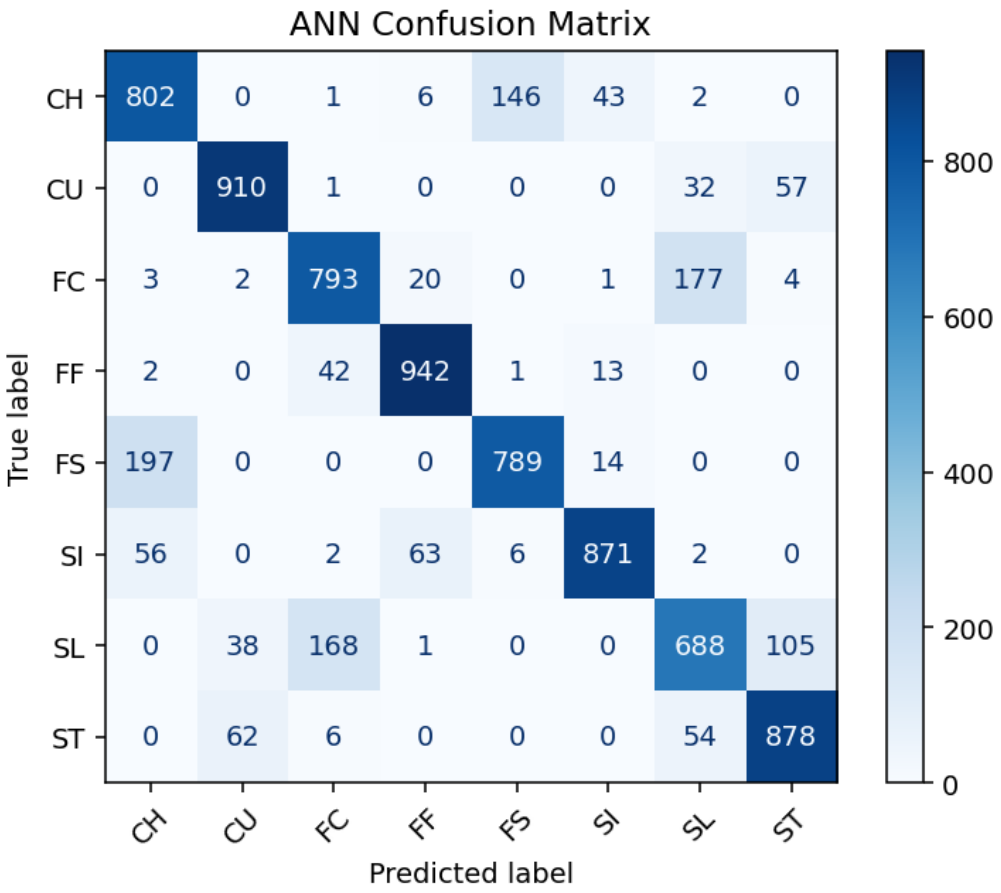


Figure 9: ANN Confusion Matrix

The same pitch types are being misclassified. This model tends to favor CH over FS, does slightly better with fastballs and SL, but worse with ST, FC, and SI. This implies that the models have different strengths in feature recognition, which foreshadows a solution by incorporating both models together to allow each to compensate for weaknesses of the other.

Principal Component Analysis (PCA

Before we move to an ensemble model, we first want to consider what happens if we move from the original sample space to a reduced dimension space. Our question is whether or not condensing our features from 5 to 2 will be more accurate in our models. We hypothesize that condensing features will result in a less accurate model due to the evident nonlinear relationships between features within our pitch data. PCA is unable to capture these nonlinear relationships, which will result in inaccurate labeling at a much higher rate.

To conduct PCA we ensure our training split is clean, convert the data to the reduced dimension space, and train the KNN and ANN models using the same parameters as before.

KNN

```
KNN PCA Accuracy: 0.6661

Classification Report (KNN PCA):
              precision    recall  f1-score   support

          CH       0.58      0.60      0.59      1000
          CU       0.78      0.78      0.78      1000
          FC       0.70      0.72      0.71      1000
          FF       0.75      0.72      0.73      1000
          FS       0.68      0.66      0.67      1000
          SI       0.65      0.67      0.66      1000
          SL       0.59      0.52      0.55      1000
          ST       0.61      0.66      0.63      1000

    accuracy                           0.67      8000
   macro avg       0.67      0.67      0.67      8000
weighted avg       0.67      0.67      0.67      8000
```

As hypothesized, dimension reduction results in a significantly lower accuracy in classification compared to using the full feature space in KNN. It appears that breaking pitches and off speed pitches are hurt the most. Our precision, recall, and f1 scores suggest that the model is not labeling accurately. This is likely caused by the principal components not capturing the complex nonlinear relationships present within the original 5 dimensional feature space.

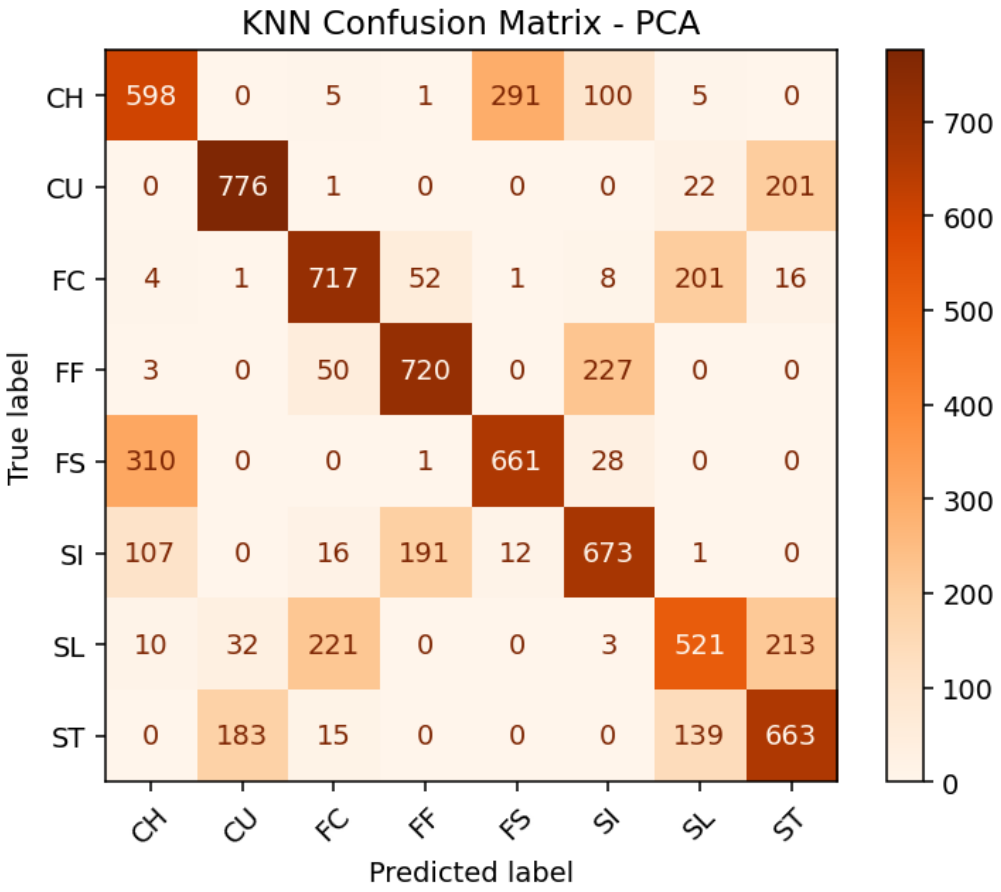


Figure 10: KNN PCA Confusion Matrix

The confusion matrix confirms that pitches are being confused for other pitch types that they overlap with at a higher rate. CH and FS are again confused the most, and even more so than in the

original KNN model. The pitches with the largest break, CU and ST, are also now being confused at a high rate. We see similar trends between FC and SL, as well as SI and FF. The results indicate that movement and velocity are the dominant features in the reduced feature space.

We now analyze the ANN under PCA to see if this behavior continues between models.

ANN

```
ANN PCA Accuracy: 0.6981
250/250 ━━━━━━━━━━━━━━━━━━━━ 1s 2ms/step

Classification Report:
              precision    recall  f1-score   support

          CH       0.60      0.74      0.66      1000
          CU       0.88      0.72      0.79      1000
          FC       0.73      0.73      0.73      1000
          FF       0.78      0.70      0.74      1000
          FS       0.77      0.61      0.68      1000
          SI       0.67      0.71      0.69      1000
          SL       0.62      0.58      0.60      1000
          ST       0.63      0.79      0.70      1000

    accuracy                           0.70      8000
   macro avg       0.71      0.70      0.70      8000
weighted avg       0.71      0.70      0.70      8000
```

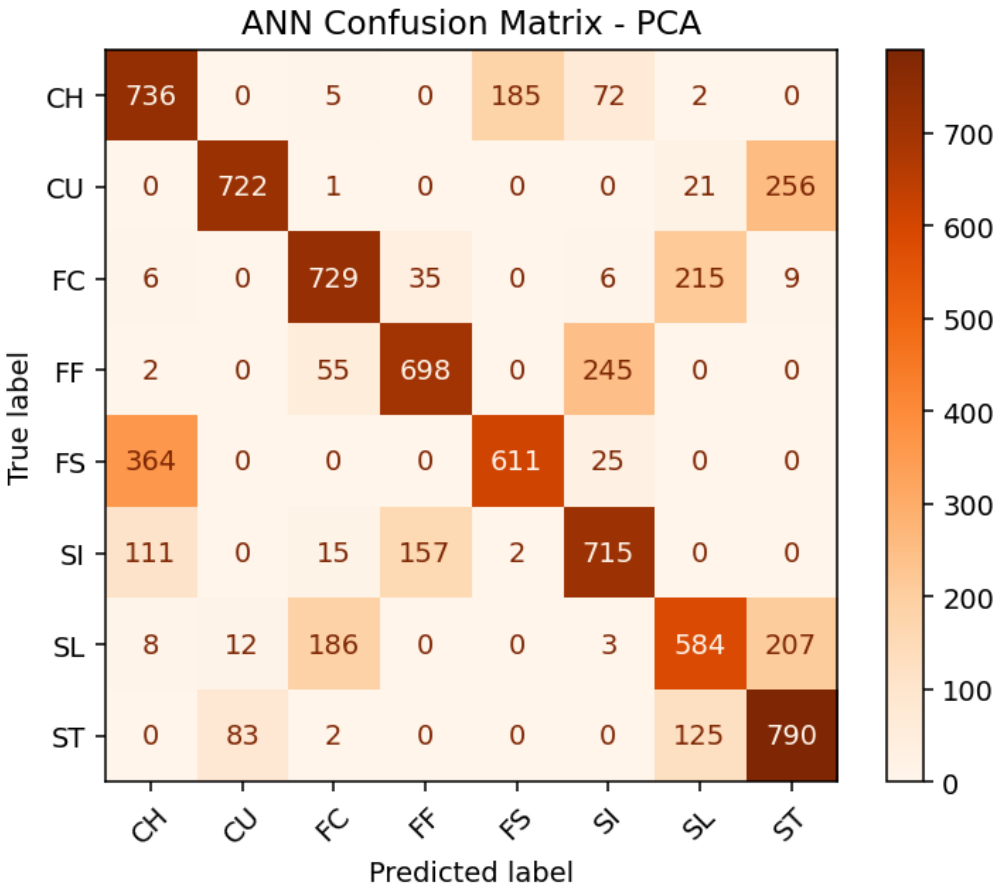


Figure 11: ANN PCA Confusion Matrix

The results of ANN and it's confusion matrix once again confirm the inaccuracy of PCA for this

application. While the model performs slightly better than KNN, it still confuses similar pitch types.

We additionally analyze pitches in the reduced dimension space to see if we can get an idea of how PCA is handling the data. Then, a conclusion may be drawn regarding PCA's performance.

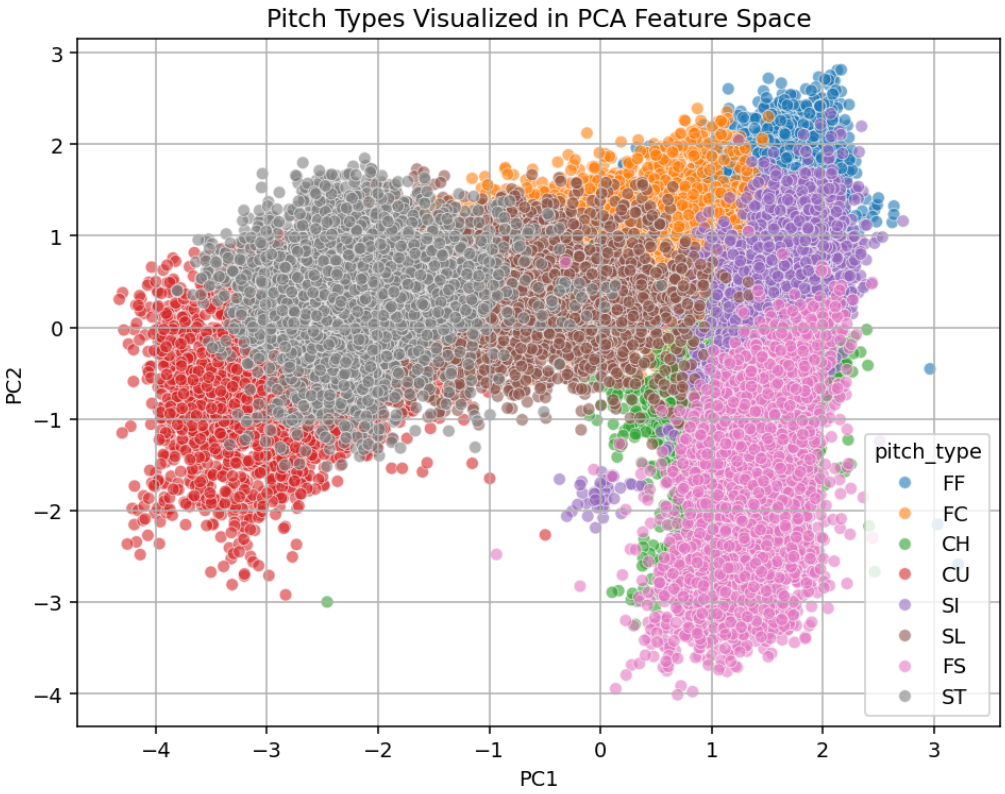


Figure 12: Pitch Types Visualized in PCA Feature Space

The clusters we observe here are extremely dense, and overlap heavily. Some clusters are entirely distinct, and some pitches are not confused for one another at all as the confusion matrix demonstrated, but pitches that are confused are confused very heavily. Ultimately, we conclude that PCA is not a suitable approach for this application.

Ensemble of KNN and ANN

We now return to the original models, and ask the question: can a higher accuracy than either model individually be achieved by combining them together?

Each model, KNN and ANN, tends to accurately classify FF, SI, FC and CU well. We also see that among the most overlapping pitch types, CH and FS, KNN favors FS and ANN favors CH. Another notable fact is the consistent struggle with SL. This issue will be further addressed. We aim to determine whether combining the two predictions together can achieve a higher overall accuracy for all pitch labeling.

Taking the weighted average of the two probabilities generates an ensemble probability, which is used to generate a label.

We define the ensemble probability as

$$p_{\text{ens}} = \alpha\, p_{\text{KNN}} + (1 - \alpha)\, p_{\text{ANN}},$$

where p_{KNN} and p_{ANN} are the class probabilities from the KNN and ANN models, respectively, and

$\alpha \in [0, 1]$ controls the relative contribution of each model. We set alpha equal to 0.5, giving both models equal weight for the first iteration.

In order use this process, we must alter the way we consider KNN. Instead of using the majority vote of neighbors to determine labeling, a probability is required instead. Fortunately, this is available though sklearn. This probability is used in the formula, alongside the ANN label probability to decide the final label.

```
Ensemble Accuracy (alpha=0.5): 0.8431

Classification Report:
              precision    recall  f1-score   support

          CH       0.78      0.79      0.78      1000
          CU       0.91      0.92      0.91      1000
          FC       0.79      0.81      0.80      1000
          FF       0.92      0.94      0.93      1000
          FS       0.83      0.82      0.83      1000
          SI       0.92      0.88      0.90      1000
          SL       0.75      0.69      0.72      1000
          ST       0.85      0.90      0.87      1000

    accuracy                           0.84      8000
   macro avg       0.84      0.84      0.84      8000
weighted avg       0.84      0.84      0.84      8000
```

We see that the answer to our question is yes, combining the models into one ensemble does yield a higher accuracy.

We consider each model equally, with alpha = .5 as this value was ultimately the value that resulted in the highest accuracy. This means that each probability is weighed equally, and the most confident label from both models will be chosen.

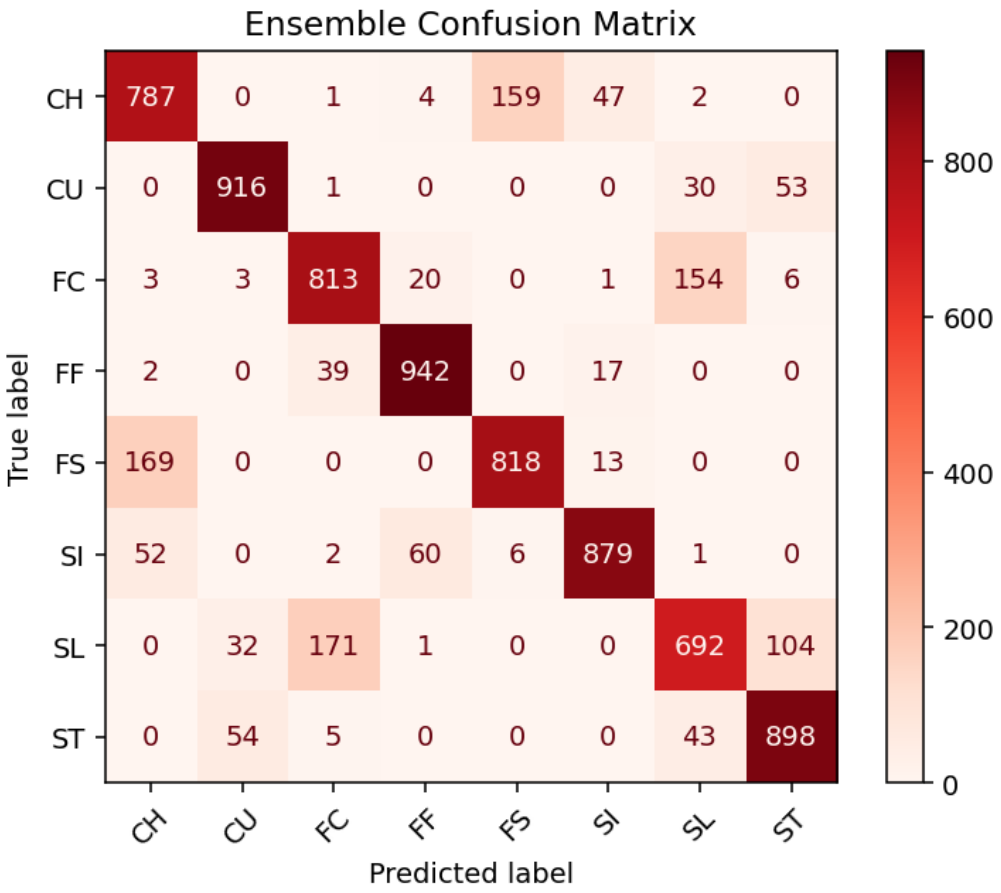


Figure 13: Ensemble Confusion Matrix

The ensemble model is more accurate overall, however, as we examine the individual pitch types, we observe that this model is never more accurate than the more accurate model between KNN and ANN for any individual pitch type. In other words, for any given individual pitch type this model is always between the KNN and ANN accuracy.

This behavior results from each model considering both probabilities, summing them, and labeling the pitch based off that combined probability. Pitches that are labeled confidently by both models are easiest to classify, but pitches that may be decided by either model based on a slim probabilistic difference can become mislabeled. This is likely happening on edge cases where the models disagree.

Conclusion

The results of our models support our hypothesis, and illustrate how classification challenges arise in multi-class machine learning models. Our initial hypothesis was that each model would struggle to differentiate pitches that showed large feature overlap, while successfully labeling pitches with distinct features. Visualization confirmed there was significant overlap between specific pitch clusters with similar features. This led us to believe that the CH/FS, FC/SL, SL/ST groups would be confused the most. Our models struggled to correctly classify those pitch types compared to pitch types with clusters that did not overlap, such as FF and CU.

To understand why CH and FS were consistently confused in the models, we consider the specific relationships between the pitches and examine their feature profiles. Included is a Pairplot that examines all feature relationships and allows us to observe the overlap for all features.

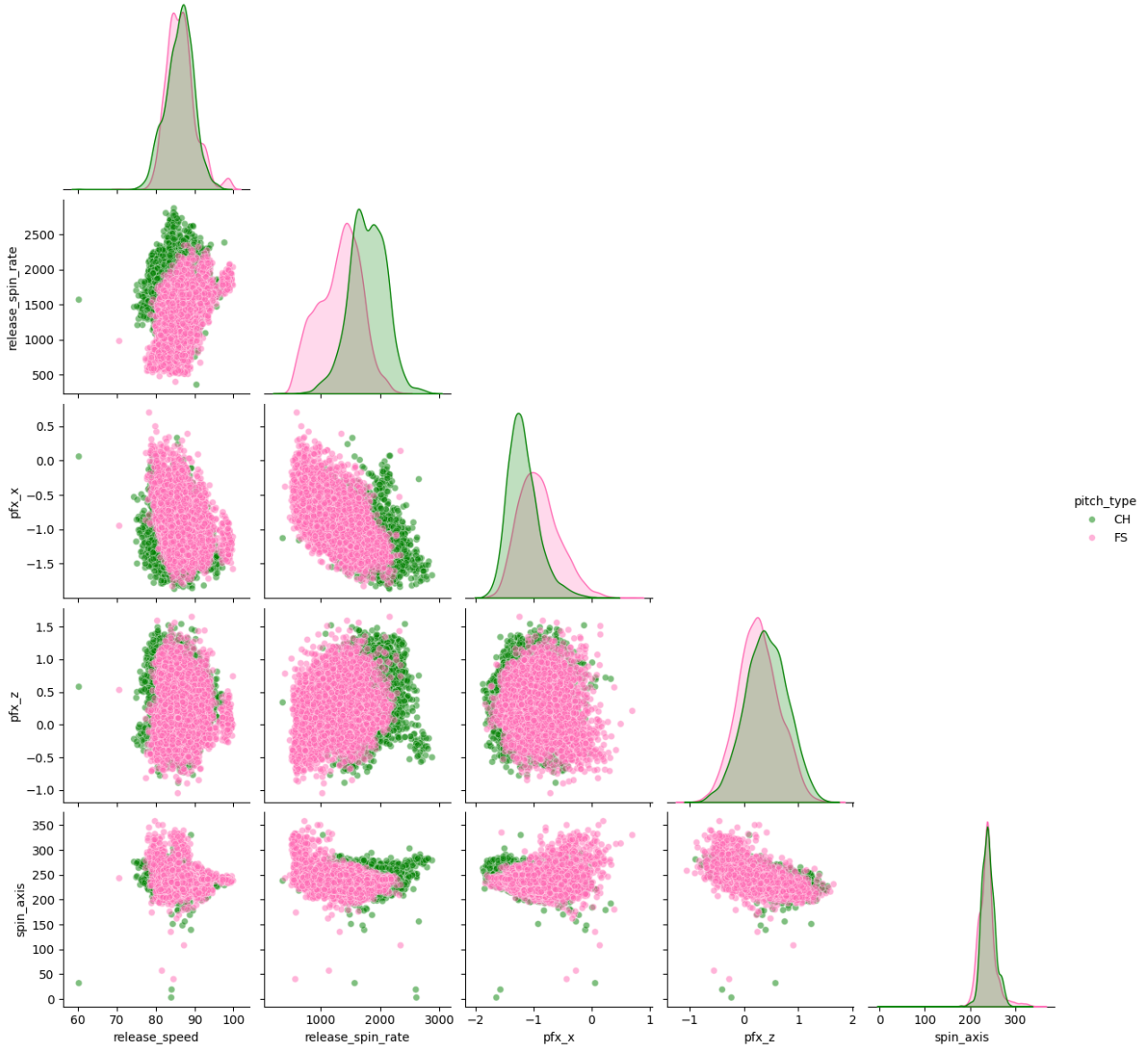


Figure 14: Changeup vs Splitter

Both pitch types show substantial overlap in all feature relationships. Spin rates, overall movement, velocity, and spin axis all fall in comparable ranges. Spin axis and release speed especially are nearly identical. From the model's perspectives, this overlap means that CH and FS lie in the same region of feature space, and cannot be cleanly separated. As a result, classifying these pitch types depends on small variations in the data. For KNN, labels are entirely decided by nearby neighbors, so small differences tilt the labeling decision towards or away from one label. ANN decision-making is limited due to an inability to separate the heavily mixed feature space. This leads to each model favoring a different label for the same pitches, and explains why CH and FS appear as a source of confusion.

SL were a consistently poor performing pitch across all three models and PCA. They were misclassified at the highest rate among pitch types, and were primarily confused for FC and ST. Included are Pairplots for SL against FC and ST.

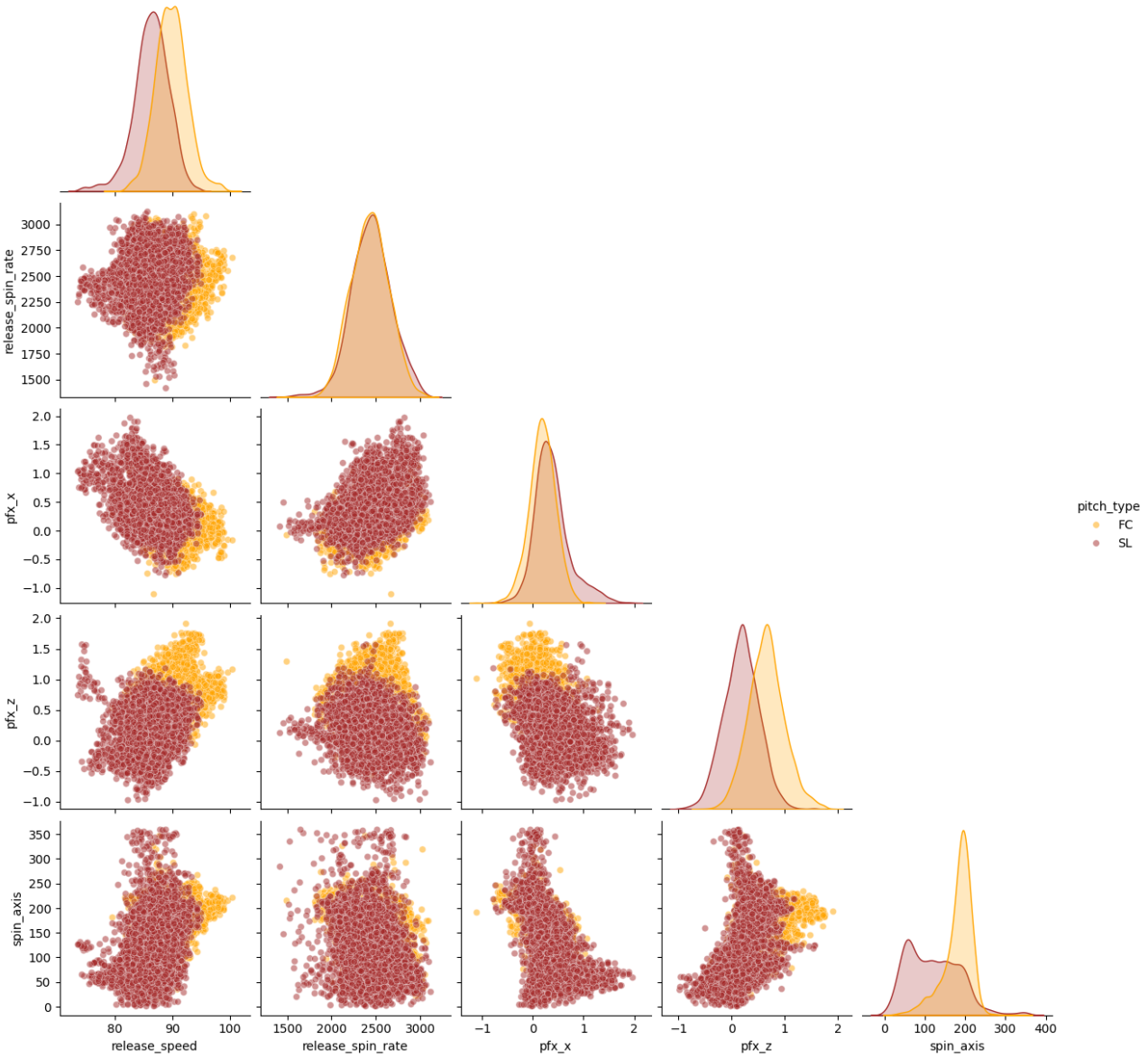


Figure 15: Slider vs Cutter

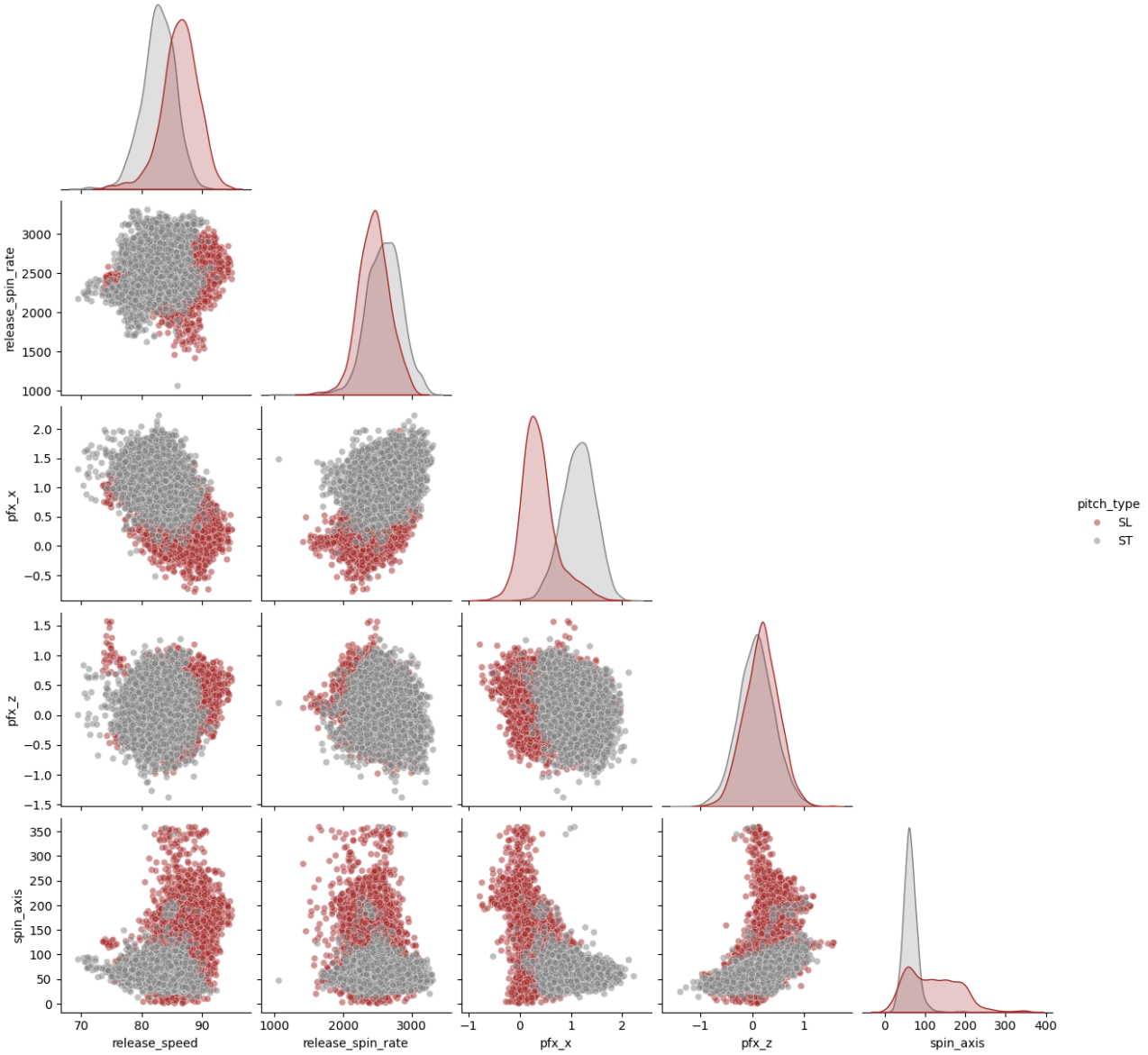


Figure 16: Slider vs Sweeper

SL are spread across the feature space, and share similar horizontal and vertical movement profiles with both FC and ST on opposite ends. While FC and ST themselves are mostly independent from one another, SL bridge the two and span across both clusters. This indicates that the measured features do not provide a unique profile for SL relative to these two pitch types.

From a modeling perspective, the overlap makes labeling difficult, as clusters cannot be easily separated. KNN sees SL with various feature identities, and many times those fall into regions dominated by ST or FC, as both of those pitches have tighter clusters. ANN decision-making is also limited for the same reason, as training the model to isolate SL becomes difficult when they exist in a space where other pitch types are present and more dense. This was further confirmed when we conducted PCA, and the compressed feature space led to SL being even further misclassified by both models into FC and ST.

In both of these cases, we observe particular pitch types with overlapping features which are difficult to separate, leading to model confusion when labeling. A solution to this, and the approach that MLB takes in their own classification system is to incorporate the player specific pitch arsenal layer. By reducing the label pool to only consider pitches that a pitcher throws, it is possible to avoid potential

confusion caused by overlapping features. This gives the model the ability to label pitches that overlap with other pitch types in the generic model more accurately.

What we can gather from this exercise in machine learning is that we successfully implemented a sequence of machine learning tasks that aided us in visualizing and classifying pitch data. Using these tasks, we found clear trends within how the models handled the data, and came to conclusions about how pitch classification works, and how it could be optimized. We conclude that pitch classification is a realistic and implementable system that can be highly accurate, and a next step to garner even greater accuracy would be to consider pitcher repertoire, to apply the correct label in cases where a pitch could belong to multiple classes. Even without this optimization, a generic pitch model such as ours performing with an accuracy of 84% is very good. We are able to separate and correctly label distinct pitches, and are even able to classify many pitches with overlapping features to a lesser degree. Our findings demonstrate that even with some unavoidable mislabeling, supervised learning models can handle multi-class classification problems, such as classifying pitches in baseball, with useful and accurate results.

References

1. Baseball Savant — MLB Advanced Media,
https://baseballsavant.mlb.com/.

2. MLB Pitch Classification Blog Sharpe, Sam. "MLB Pitch Classification." MLB Technology Blog, 28 Apr. 2017,
https://technology.mlblogs.com/mlb-pitch-classification-64a1e32ee079.

3. MLB Statcast + Hawk-Eye + Google Cloud Jedlovec, Ben. "Introducing Statcast 2020: Hawk-Eye and Google Cloud." MLB Technology Blog, 3 March 2020,
https://technology.mlblogs.com/introducing-statcast-2020-hawk-eye-and-google-cloud-a5f5c20321b8.

4. Drive Line Baseball "Mastering The Axis of Rotation: A Thorough Review of Spin Axis in Three Dimensions," by Drive Line Baseball, 06 Sep. 2019,
https://www.drivelinebaseball.com/wp-content/uploads/2019/09/spin_axis_green_line.webp.

5. Multi-Label Classification for Beginners
Gupta, Mehul. "Multi-label classification for beginners with codes." Medium, 8 Nov. 2023,
https://medium.com/data-science-in-your-pocket/multi-label-classification-for-beginners-with-codes-6b098cc76f99.