**CSC 495 Final Report**

Jacob Schuh

**Introduction**

In the sport of baseball there are many types of pitches thrown. Each pitch has different characteristics such as speed, spin of the ball, horizontal break, etc. Each of these pitches are achieved by changing the type of grip that a pitcher uses while pitching, when analyzing the data of games, it is very useful to know the type of pitch for every pitch thrown, but after the fact we don’t know the exact grip of every pitch. To get this information you need to have someone go through each pitch and classify them. This can be time-consuming and the person who does it will need to understand the characteristics of these pitches. With the help of machine learning we can build a models classify these pitches almost instantly.

In recent years, teams have understood that using the new technologies at their disposal can allow them to better themselves. One of these technologies that EKU has started to utilize is Trackman. Trackman is a system that can give teams comprehensive ball-tracking data for every pitch and play of a game. This data can be used by teams to better understand how they are playing, and changes that they can make in real time. However, it requires a person with an understanding of baseball to collect the pitch types during the game. Using machine earning we can practically eliminate the need for someone to collect the data in real time.

With the advancement of machine learning we can train models to categorize these pitches based off the observable data from the Trackman. For this application, supervised learning methods need to be implemented, because for each observation we label it with what type of pitch it is. It will also need to be a classification algorithm as we need to classify each method as a different type of pitch. Most basic classification methods are simply a binary choice, so a method to determine the type of pitch thrown would need to be able to classify between multiple different target values.

In this project I will be training methods to accurately classify the type of pitch thrown based off observable ball-tracking data. The machine learning models I plan to use are DecisionTree, RandomForest, SVM, and KNN models. Each of these can classify every observation into multiple factors in a supervised learning model. With the use of machine learning this problem will be effectively solved.

**Abstract**

Analyzing and using data that is collected for a team is an invaluable resource to baseball teams. With advanced ball-tracking data having every pitch correctly labeled in incredibly important for doing even simple data analysis. My goal for this project is to train classification models to predict the type of pitch thrown. I will be using a RandomForest model, a decisionTree model, a Support Vector Machine model, and a K-Nearest Neighbors model. For these models I used 2160 pitches from 7 games over the EKU baseball team’s 2022 season. I also determined the best model to use for this application and built an interactive application to demonstrate it. This project has allowed me to look at a complex problem of classifying the type of pitch thrown based on ball-tracking data, and to use machine learning methods to solve this problem.

**Methods**

The data used for this project was collected using the Trackman Software, collected from 7 different games between February - May 2022. The dataset includes 2160 pitches. These pitches were put into an R data frame, normalized, randomized, and partitioned into a 70/30 training and testing split. Each pitch observation in the data frame had 18 variables recorded. They were: RelSpeed, VertRelAngle, HorzRelAngle, SpinRate, SpinAxis, Tilt, RelHeight, RelSide, Extension, VertBreak, InducedVertBreak, HorzBreak, PlateLocHeight, PlateLocSide, ZoneSpeed, VertApprAngle, HorzApprAngle, and ZoneTime.

Decision Tree Model:

A decision tree is a supervised learning algorithm that is used for classification and regression modeling. Decision trees look like flowcharts, starting at the root node with a specific question of data, that leads to branches that hold potential answers. The branches then lead to decision (internal) nodes, which ask more questions that lead to more outcomes. This goes on until the data reaches what’s called a terminal (or “leaf”) node and ends.

The code to use this model in the project:

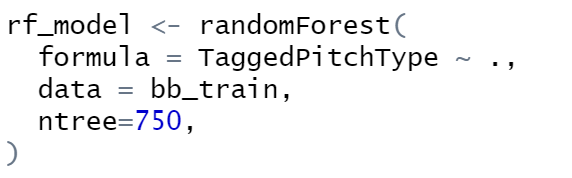
A picture containing text, font, screenshot, white

Description automatically generated

Random Forest Model:

Random forest models consist of many individual decision trees that operate as one. Each individual tree in the random forest spits out a class prediction and the class with the most votes become our model’s prediction.

The code to use this model in the project:



SVM Model:

Support vector machine is a supervised learning system and is used for classification and regression problems. A support vector machine works by defining the best hyperplane (boundary between classifications). Through the use of different “kernels”, they can make higher-dimensional data linearly separable, such as by using the radial or sigmoid kernels.

The code to use this model in the project:

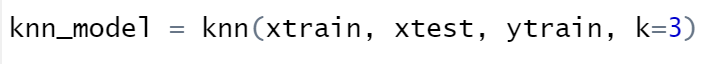
A black text on a white background

Description automatically generated with low confidenceThe best parameters of the SVM model are using a radial kernel, and a cost = 10.

KNN Model:

KNN is a simple, supervised machine learning (ML) algorithm that can be used for classification or regression tasks - and is also frequently used in missing value imputation. It is based on the idea that the observations closest to a given data point are the most "similar" observations in a data set, and we can therefore classify unforeseen points based on the values of the closest existing points.

The code to use this model in the project:



**Results**

|  |  |
| --- | --- |
| **Model** | **Accuracy** |
| Random Forest | 96.7% |
| Support Vector Machine | 94.3% |
| K Nearest Neighbors | 87.8% |
| Decision Tree | 90.5% |

As you can see in the table above the machine learning models trained for this classification purpose were extremely accurate. With 96.7% accuracy on the test set the Random Forest model does an excellent job of predicting the type of pitch thrown with the data given. These models fully met my expectations for this project.

**Random Forest Model Output:**

Output:

Confusion matrix:

Table

Description automatically generated

Chart, scatter chart

Description automatically generated

This plot shows the importance of every variable in the randomForest model

Accuracy:

> accuracy

[1] 0.9671362

> error

[1] 0.03286385

**SVM Model Output:**

Output:

Graphical user interface, text, application

Description automatically generated

Table

Description automatically generated

Accuracy:

> accuracy

[1] 0.943662

**KNN Model Output**:

Output:

A screenshot of a computer

Description automatically generated with low confidence

A screenshot of a computer

Description automatically generated with medium confidence

Accuracy:

87.79%

**Decision Tree Output:**

Output:

A picture containing text, font, screenshot, number

Description automatically generated

Accuracy:

Accuracy : 0.9045

95% CI : (0.8791, 0.9262)

No Information Rate : 0.579

P-Value [Acc > NIR] : < 2.2e-16

**Caveats**

While I have been able to show great results, there are a few issues with this project I would like to address. First is the accuracy of the data because I didn’t collect the data myself, I have no way to know if the data is completely accurate. Using the trackman software someone entering in pitches can enter them wrong. This happens for many reasons such as people recording the pitches can’t always find the difference between two-types of pitches, or some record two different pitches as the same type.

Another issue is having too small of a data set. While 2000 pitches are a very large amount, having more would likely give us much more accurate and versatile methods. For a machine learning model to be able to determine something it needs to have seen at least one instance of it before. One example of this problem is the absence of the “splitter” pitch. This is a pitch type characterized by a sharp vertical movement, and an extremely low spin rate. This pitch is somewhat rarely thrown, but if the model I have created attempted to determine this pitch it would determine that it is not the right pitch. Without a more complete data set, our models are much less versatile.

The fix for these issues is to collect each pitch in a setting in which whoever is recording the data knows exactly what pitch is being thrown to have complete control over the accuracy of the data. To fix the size of the data set, we would simply need to collect more data, and make sure to collect data that specifically identifies every pitch we want to be able to determine.

Neither of these issues in anyway invalidates our findings, though with the knowledge of having a more accurate dataset, as well as having a larger and more complete dataset, we could create much more accurate models that are much more useful for their purpose.

**Conclusions**

Over the course of this project, I developed classification machine learning models to determine the type of pitch thrown, given advanced ball-tracking data. The most accurate of the models that I built was the Random Forest Model that can determine a pitch using ball-tracking data 96.7% of the time. This project has allowed me to learn much more about how the Random Forest, SVM, KNN, and Decision Tree models work for classification problems. I have also built a Shiny App to better exhibit the ability of these machine learning models.

**References**

Breiman, L. (2001), Random Forests, Machine Learning 45(1), 5-32.

Breiman, L (2002), “Manual On Setting Up, Using, And Understanding Random Forests V3.1”, https://www.stat.berkeley.edu/~breiman/Using\_random\_forests\_V3.1.pdf.

Rong-En Fan and Pai-Hsune Chen and Chih-Jen Lin (2005), “Working Set Selection Using the Second Order Information for Training SVM” https://www.csie.ntu.edu.tw/~cjlin/papers/quadworkset.pdf

Ripley, B. D. (1996) Pattern Recognition and Neural Networks. Cambridge.

Venables, W. N. and Ripley, B. D. (2002) Modern Applied Statistics with S. Fourth edition. Springer.

Breiman L., Friedman J. H., Olshen R. A., and Stone, C. J. (1984) Classification and Regression Trees. Wadsworth.