Data mining and machine learning have become a major interest of mine since being exposed to the topics in our Data Mining class. When presented an opportunity to build a project to solve a business problem or inquiry a personal interest, I knew I wanted to relate the project to a sport. I thought baseball would be a great sport to run models on and do analysis on because it produces a lot of data. There are so many statistics that come from a baseball game that can be analyzed to improve play and managerial processes within a game. I have a great interest and respect for pitchers, so I made the decision to look at data for a given pitch and determine if that pitch will result in a strike or ball. Obviously, there are more outcomes in an at bat than just strike or ball including hits, home runs, walks, etc. However, for this project I bucketed any form of a pitch that would be a strike and any form of a pitch that would be a ball. So, hits, fouls, and called strikes I am considering a strike, and called balls or hit by pitch I am considering a ball.

To build out the project and have the capability to do analysis, I needed to find a good data set that would help me solve this problem. Finding data was slightly harder than I expected, because I did not know the best places to find data in somewhat of a format I could use and manipulate. I ended up finding data on a baseball game between the Seattle Mariners and Houston Astros. The data had many attributes that detailed advanced statistics about the game at hand. The data came from a site called data.world and it was a perfect for me to do analysis on and start to manipulate it into a format that I could use for data mining. Unfortunately, the data dictionary was not updated for the data attributes within the data set, so I was left to try and interpret that field names and what statistics they corresponded to. Ultimately, this left some the fields unusable because I did not know what the data was referring to. On the bright side, the data set did contain enough information to where I was still able to have enough statistics to analyze for my purposes within this project.

The data set needed to be cleaned, but I underestimated the amount of data that needed to be cleaned on my first pass. The data had the result of the given pitch, but I needed to manipulate this field to relate to my predictions. I did this by creating a vlookup in excel to turn any form of a strike to a strike, and any form of a ball to a ball. I next moved the prediction field to the last column which is a best practice when using Weka. Next, I eliminated fields that were useless for my purposes. Those fields included unique identifiers, dates, times within the game, and some other fields that had the same value throughout each record of the data set. I also wanted to create numeric and discretized versions of my data set so I could compare accuracy and determine if numeric attributes or bucketed attributes work better in the algorithms I was going to be using. I used the discretize preprocess method to create a bucketed data set the was developed from the original data set with numeric values. This concluded my first pass of data cleaning and I attempted to transition into finding a baseline model. To do so, I saved the excel file I was using for data cleaning into a CSV file. This way I could open the CSV file in Weka and save it as an arff file which is the file type Weka accepts and uses. Once successful, I ran common baseline models such as OneR and ZeroR on the data set. This is where I realized more data cleaning needed to be done. The models where using attributes I did not have enough information on to understand the basis of the predictions. The attributes where named pz and px. I carried out the rest of this experiment to see if all of the models used these attributes, and they did.

In experiment two, I went back and cleaned that data some more. I decided to remove all fields where I did not know what they meant or I just did not have enough information on. This forced me to remove more data than I would have liked, but there were still enough statistics to continue the data mining process with this data set. Once I felt I had a strong grasp on all of the fields that remained, I decided to redo Experiment one in Experiment two. I was looking for a baseline model that I could compare with more complex models to review accuracy and performance. I ran the data set with the algorithms of OneR and ZeroR again to find a potential baseline candidate. These two algorithms produce simple and comprehensible models to the human eye which make them prime candidates of a baseline. ZeroR predicts the majority class each time, and OneR finds one attribute and forms predictions based on thresholds of that attribute. When finding a baseline, it is helpful to find out what the chance probability is for the predicting attribute. In this case, there are two possible outcomes, ball or strike, so the chance of guessing the correct answer is 50%. I expected to find a baseline slightly higher than this, because the models and algorithms should be producing results that are more effective than mere chance or luck. ZeroR produced a 63% test accuracy by predicting strike for each test instances because strike was the more common class. OneR, to my surprise, did very well with classification rates between 70% and 74%. The models for OneR determined that the zone attribute would be best to base the predictions on. Relating these models back to a baseline, I was afraid that OneR could be a bit too optimistic. For this reason, I decided to use the ZeroR models as a baseline to relate complex models back to.

For my next experiment, experiment 3, I decided to run the data set with algorithms that develop rule sets. I decided to start with rule sets after finding the baseline because they are some of the easiest models for myself and I think for the human eye to comprehend. The algorithms I chose to use within the rule set group were OneR, JRip, and Prism. Before conducting the experiment, I knew Prism could not handle missing values and algorithms that I was going to use in later experiments also could not handle missing values. For this reason, I needed to manipulate the data set so that there were no missing values within the data. I chose to do this in Weka because the software provides preprocess methods to handle, remove, or replace missing values throughout the data set. I attempted to try two different preprocess methods, removing and replacing. When I used the remove missing values method, it ended up removing too many records and not leaving me enough data to have a useful experiment. For this reason, I could not use the remove missing values preprocess method. However, the replace missing values method worked great and I was able to develop a discretized data set with missing values replaced. The replacement method uses means to populate the missing values with values that are about average throughout the data set. After creating the new version, I was ready to conduct experiment 3 and develop rule set models to find out what identifiers classify if a pitch will be a strike or a ball. I had OneR models from the baseline experiment that had accuracies between 70 and 74 percent. These models gave me a take home message of if a pitch is overly high, then it is likely a ball. If the pitch is not overly high, then predict a strike. The OneR models gave some relevant insight in the fact that a high pitch will often result in a ball because batters can recognize a pitch coming in high and umpires will often call the pitch a ball. However, the models did not utilize low zone numbers to determine if a pitch is very low then it could also be a ball. Next, I used JRip to develop a more complex model than OneR. JRip did extremely well with test accuracies between 80 and 81 percent. Although the algorithm is more complex than OneR, the resulting rule set from JRip was not extremely complex. JRip used two attributes, zone and last pitch in at bat indicator, to develop its model. It was very interesting to see that we could classify correctly at an 80% rate by using just these two fields. The take home message from the JRip models was that if the pitch was in a zone that was fairly high, and it was not the last pitch in the at bat, then it was likely a ball. Otherwise if the pitch was not relatively high or it is the last pitch in the at bat, then predict strike. As the human in the loop, this model derived excellent insight because batters have an easier time distinguishing if a pitch will be too high compared to too low in the strike zone. Also, if the pitch is going to be the last pitch in the at bat, pitchers are much more likely to force a pitch into the strike zone so they do not walk the batter. I was extremely pleased and excited about the models that JRip was able to produce. The last algorithm used in experiment 3 was Prism. As stated before, Prism cannot handle missing values. Therefore, the only applicable data set that could be used with Prism was the discretized version with missing values replaced. The resulting model was very complex, and the performance was low at a 61% test accuracy rate. The model showed that the algorithm was taking into account batter and pitcher names in a lot of the rules. This could derive good insight on if a batter will do well against a given pitch with other factors included, but in terms of overall accuracy the model underperformed compared to others. However, I was able to derive some relevant information from the complex model such as if the pitch is not too high and not to low, then it is more likely to be a strike than ball. If the pitch was a knuckle curveball in the first three innings, then it is more likely to be ball. This makes sense because it can take pitchers time to command their breaking pitches because they are harder to control. The model also used other attributes such as pitch type confidence, batter stance, ball spin direction, ball spin rate, and others. Overall, experiment 3 had some highs and some lows. OneR and JRip performed well and gave me solid information about indications of a strike. Prism did not do as well and it performed worse than my baseline model, but it did give me a more detailed look into the data where I could analyze more complex situations within the game.

My next approach was algorithms within the tree model group. Rule sets and trees are somewhat similar in the fact the tress often have a rule for a given node in the tree. Then stemming from that node will be more rule nodes or a leaf where a prediction is made. The models I chose to use for this project for tree modeling were Decision Stump, J48, and ID3. There was actually a fairly strong correlation between OneR and Decision Stump, JRip and J48, and Prism and ID3. The accuracies were similar in each paring. Decision Stump is similar to OneR because it uses one attribute to develop the tree model with leaves stemming from the root node. Decision stump also used the zone attribute like OneR and determined that if a pitch is high at all then predict ball and strike otherwise. This approach worked well with test accuracies between 72-74%. I next tried J48 and it performed very well with test accuracies between 79-81%. J48 also used the zone attribute, but it paired it with other fields that I thought could be very useful such as pitcher pitch type confidence, break angle, last pitch in at bat indicator, and strikes and balls within at bat. This model derived valuable information from the data such as if the pitcher confidence with a given pitch is high then it will likely be a strike. Or if there is a lot of break on the ball then it will more likely be a ball because it is harder to control. J48 model’s gave me some of the most insightful messages out of all of the models in the project. The last algorithm used in experiment 4 was ID3. Like Prism, ID3 had a complex model which included fields like batter name and pitcher name as well as could only be ran with the missing values replaced data set. The accuracy was the lowest of all models, but the underlying details give some good information such as how a certain pitch type will do against a given batter. Overall, experiment 4 was another success because I continued to derive valuable information from all the models. Even though ID3 did not perform very well, I still believe this experiment was an overall success.

The last experiment I conducted, experiment 5, included algorithms that were not necessarily from the same group, but instead algorithms I thought would be useful to try and potentially give valuable information. The algorithms I chose to use were Naïve Bayes, IB1, and KStar. Naïve Bayes’ model is fairly complex and it is hard to comprehend as the human in the loop. The accuracies with these models ranged between 72-73%. The take home messages I was able to derive from the models were that the slider was extremely effective in the game because 60 out of 90 sliders thrown were strikes. Also the average strike had a break angle less than or equal to 15 were as balls had a higher average break angle. The next two algorithms, IB1 and KStar, use similarity and nearest neighbor approaches to find similar training instances to relate to the given test instance. IB1 finds the training instance that is most similar to the test instance and classifies the test instance based on that training instance. KStar finds a group of similar test instances through a similarity function using an entropy-distance based function. Both models produced accuracies below the baseline which gave me great insight that my data does not have a lot of similarities in classification. Indicators for a strike for one pitch do not always correlate as a strike with similar indicators for another pitch. I consider experiment 5 as another success because again, I was able to derive relevant and useful information. Although IB1 and KStar did not perform well, it told me that the data is not interchangeable when making predictions for strike or ball.

In conclusion, I trust the majority of the models that were produced in the experiments and I have no problems accepting patterns that I did not know before. I have very pleased with the information I was able to derive from the models. My major take home messages are that there are a lot of details that can go into a pitch and similar details do not always produce the same result. However, pitches that are thrown too high and too low often produce a ball because batters are able to recognize the location easier. Pitches that are not too high and not too low often produce more strikes even if they are outside of the strike zone. Pitches with a lot of break and spin on the ball often produce a ball as well. If the pitch is the last pitch in the at bat, there is a high likelihood that the pitch will be a strike because pitchers do not want to give up a walk. All of these insights are valuable and can be applied to certain situations in a baseball game to help managers and players know what to do and what to expect in those scenarios. If I had more time on this project, I would first want to gather more data so I could have a broader scope of what makes a pitch a strike. I would also want to try and classify more than just strike or ball, but instead what indicators lead to a strike, hit, or ball.