CS Project: Using Machine Learning to predict results in Major League Baseball

The core of my project is to use machine learning tools to predict results in Major League Baseball. Initially, the core of the project will be based around using support vector machines in scikit learn to predict which team will win in a head to head clash. The academic literature I have read suggests SVMs would be the most suitable method for finding a solution. I intend to use supervised learning.

Building upon this core idea, I would like to extend the predictive ability of the model onto other factors, including predicting an over/under value for how many runs would be scored within a game and other similar problems. Beyond this, I would then try to develop the tool further to allow more in-depth predictions based on individual players, such as who is likely to win the following season's MVP.

Once the model is in place, I would then look to build a user interface that would allow a user to enter data, say two teams who are playing in an upcoming game, and to receive the classification result.

The overall purpose of the project would be to provide a quick, easy to use resource for advanced baseball betting.

To implement the project I am planning to code in python, using available libraries such as sci kit learn, numpy and pandas. I will be using existing libraries as building something similar from scratch would probably be a project within itself, and ultimately would probably leave me with something less effective.

My choice of using SVMs is based upon the paper "Predicting Win-Loss outcomes in MLB regular season games – A comparative study using data mining methods" (Soto Valero, 2016), which tested four different data mining methods, both classification and regression based, to try and predict which team would win a head to head match. SVMs returned the best results, with an accuracy of almost 60%. However, the paper above used only statistics based on overall team performance and not individual players. I believe using individual player data would be more effective. For example, if the star pitcher of the New York Yankees made a blockbuster move in the off season to the Pittsburgh Pirates, it would be intuitive to expect the overall performance of the Yankees to decrease and that of the Pirates to increase. However, if the data used for the model only took the previous years results into account this nuance would be missed.

Baseball is an incredibly unpredictable game with many variables: it will never be possible to use a model to predict who will win with, say, 90% accuracy. However, over the past five seasons the team favoured by the book makers has won roughly 58% of their games, the lowest percentage in any of the major American sports leagues. To compare: over the same period if you picked the team playing at home to win every game you would win 53% of the time. With this in mind, a prediction accuracy above 60% would give a long term gambler a significant advantage, and I would *hope* to be able to return a method with an accuracy above 60% by building upon the currently available academic literature.

One advantage to choosing baseball for this project is the mass of in-depth, freely available statistics available on the internet. I intend to scrape the data I need from my projects off these online resources (most notably https://www.baseball-reference.com/). These various websites have already calculated a number of the so-called Sabermetrics, or advanced statistics, saving me time and computation.

Once I have developed the model, including using cross validation to avoid over-fitting, I intend to build a user interface that would allow a user to enter a query and be given a response quickly. I am intending to use the Tkinter package, partly due to its ubiquitousness and partly due to my previous experience of programming in Tcl.

The intended result of my project would be to produce a clean, simple program that would allow users to receive advice on how to bet quickly, and for it to be built upon a machine learning framework they could trust.