Capstone Project 1 - Data Wrangling

The purpose of this document is to summarize the data wrangling steps taken to prepare the datasets for analysis. This project will use two files from the Kaggle MLB Pitch Data 2015-2018 dataset. The first file, 'pitches.csv', contains data from every pitch thrown in each game in each of the four seasons. The second file, 'atbats.csv', contains data from every at-bat from every game in each of the four seasons. The ultimate goal of this analysis is to build a model that can predict the likelihood that a pitcher will throw a fastball on the next pitch.

Dimensions

'pitches.csv' == 2,870,000 x 40
'Atbats.csv' == 740,000 by 11

Step 1 - Merge the Datasets

The two .csv files can be merged (using 'pd.merge()') on the common 'ab_id' column. Using the 'nunique()' method in pandas, I was able to determine that the 'pitches.csv' dataset has data recorded from 740,241 unique at-bats whereas the 'atbats.csv' has data recorded from 740,389 unique at-bats. Because the 'pitches.csv' file is the dataset with the 'pitch_type' variable that is the focus of this analysis, I decided to left-join the 'atbats.csv' dataset to the 'pitches.csv' so that the extra at-bats in the 'atbats.csv' file would drop out. Given the large volume of data in the 'pitches.csv' data file, I was comfortable only joining the corresponding at-bats from the 'atbats.csv' file.

The resulting dataframe has a dimension of (2867154, 50).

Step 2 - Address Missing Data

A number of columns in the newly merged dataset contained some 'NaN' values. A majority of the columns in the data set are numerical datatypes (floats and ints). For those datatypes, I elected at this point not to replace the 'NaN' values with a numerical value (such as 0) because I am unsure at this point whether that will be needed for the analysis. I also did not want to replace the 'NaN' values with a 'MISSING' string because it would convert the columns to an 'object' datatype and prevent me from gaining any aggregation information from those columns (e.g., by running a .describe() on the dataframe).

I took a different approach regarding the 'NaN' values in the 'pitch_type' column given that it is the target data of our analysis and it is categorical data of strings, including 'FF' for fastballs. I investigated into whether other columns of the dataset may inform whether the unlabeled pitches were fastballs (especially the 'start_speed' and 'end_speed' columns). Unfortunately, all but five of those entries had 'NaN' values for both start and end speeds, and the five that did have entries appeared to be unreliable because they contained abnormally slow outliers. I therefore decided to replace the ‘NaN’ values with the string ‘MISSING’ in order to make it clear that there is no pitch_type label for those particular pitches.

I also replaced the ‘NaN’ values with ‘MISSING’ string values in the ‘code’ column because it contains categorical data of strings as well.

Step 3 - Create Dummy Variables for Categorical Data

The dataset contains 6 columns of categorical data - ‘pitch_type’, ‘p_throws,’ ‘stand,’ ‘top,’ ‘type’, and ‘code.’

The ‘pitch_type’ column is our target variable and it contains a number of different categories of data, including ‘FF’ for fastball. Because the goal of the project is to predict when a fastball will be thrown, I created a new ‘fastball’ column that contains a value of 1 for every pitch that was labeled with the pitch type ‘FF’ and a value of 0 for every other pitch type category.

The ‘p_throws’, ‘stand’, and ‘top’ columns are all binary categorical data. Both the ‘p_throws’ and ‘stand’ columns contain either an ‘R’ or ‘L’ depending on whether the pitcher and batter are right or left handed, respectively. The ‘top’ column is a boolean datatype that is ‘True’ for every pitch thrown in the top half of an inning and ‘False’ for every pitch thrown in the bottom half of the inning. Therefore, I created binary numerical columns for each of these columns (1 for ‘R’ and ‘True’ values, 0 for ‘L’ and ‘False’ values) titled ‘p_throws_num’, ‘stand_num’, and ‘top_num.’

The final two columns, ‘type’ and ‘code’, are duplicative of the same information regarding the outcome of the pitch. The ‘type’ column provides only three different categories (‘B’ for ball, ‘S’ for strike, and ‘X’ for ball in play), while the ‘code’ column contains more detailed categorical data (e.g., hit on ground, popped up, home run, etc.). In the interest of limiting the total number of columns in the dataset for file size concerns, I elected to only add three dummy columns (‘type_B’, ‘type_S’, and ‘type_X’) to the dataset using the .get_dummies() method and pd.concat(). If it becomes potentially relevant to the analysis, I will add dummy columns for all of the categorical data strings contained in the ‘code’ column.

Step 4 - Save the Clean Dataset

Having addressed all of the missing data issues and properly cleaned the data, I saved the new dataframe as a new file using the ‘.to_csv(‘df_final.csv’)’ method on the fully updated dataframe.