Capstone 2 Project Milestone Report 1
Social Network Outreach Analysis
Springboard Data Science Career Track
Eric Cruz

Problem Statement

The problem statement is whether given a roster list for a college tennis team, can a LinkedIn profile be found for each name, and is it likely to match the person being searched? The compilation of the roster list and a hit rate in the neighborhood of 50% of names accurately found is a great start for a targeted social network outreach campaign.

Obtaining the Data

The deliverable datasets are the set of downloads for a College Tennis Team's historical rosters from the school's website, matched with a search for the roster name/school combination on LinkedIn. The roster download is obtained using a web scraping tool called Octoparse, which has a free version that is adequate for obtaining rosters from the scope of approximately 200 teams in the original New York outreach campaign. This represents approximately 100 schools, for both a Men's and Women's team. The LinkedIn search is obtained using Selenium WebDriver with ChromeDriver, and writing a Python script in a Jupyter Notebook.

- ➢ Octoparse gui-based web scraping tool
    - The tutorial for scraping elements from a web page is straightforward and simple, as just clicking on what you want to scrape created the script behind the scenes. The slightly more complex part is figuring out how to cycle through the available roster years, usually from a drop-down list.
    - The initial collection included about 60 different teams, or 30 schools for both Men and Women. Most schools use one of 3 or 4 different formats for organizing the archived rosters.
    - There was some quasi-scripting involved for a few of the formats, such as creating a list of roster years in Excel based on a pattern of switching a few numbers to represent different years. Those involve defining the set of years to scrape and identifying patterns in the web address organization, but a large portion of websites are organized so all the available years are cycled through automatically.
    - The fields chosen for download are Name, SchoolYear (e.g Freshman, Senior), Home Town, Roster Identification (e.g. Men's 2015 Roster), and the profile Link.
- ➢ Selenium Python Script in Jupyter Notebook for LinkedIn Search
    - A few different tutorials can be found online with a search engine such as

Google, which follow a pattern of using Selenium to log in to LinkedIn with your own account credentials, then perform a keyword search in a search engine such as Google or Bing, then use the profile URL to scrape details from LinkedIn

- A tutorial which uses the Bing API was the easiest to implement, and offered a free trial for limited use keys for evaluating the API product. Aside from the cost of purchasing a tier-based subscription, the result was very disappointing with respect to the Bing Search. As pointed out, it seems ironic that Bing is a Microsoft product, and LinkedIn is also owned by Microsoft and tries to make it difficult to scrape their data. In short, the Bing search started returning different results using the same keywords, with worse results for finding the target person each time. That seems suspicious given the association of both products with Microsoft.

  https://codeburst.io/bulk-researching-linkedin-contacts-with-python-microsofts-bing-api-1f9a19e1c7b7

- The tutorial most useful, in my opinion, uses an ipython terminal which is useful for learning about detecting elements for scraping, but it was not hard to convert to using a Jupyter notebook, which is more in line with the Springboard course curriculum.

  https://www.linkedin.com/pulse/how-easy-scraping-data-from-linkedin-profiles-david-craven/

- Using a Google search instead of a Bing search, which does not require an API for purchase, better results were produced on a known keyword search. However, the search was still a disappointment knowing that the LinkedIn URL string was used in the keyword and the intended result was still less accurate than expected. However, I came to realize that the reason the tutorials use a search engine is they are looking for profiles with characteristics, not a specific person. I was able to use Selenium to make a direct keyword search in LinkedIn, and collect the URL from that search for evaluation of accuracy.
- The keyword search used is the “Name + School”, for example to search myself the keyword is “Eric Cruz Cornell”. There are some online discussions regarding the LinkedIn Terms of Service with respect to volume and crawling, so I proceeded cautiously and applied for a “whitelist exemption” for permission to crawl. A response has not been provided after about a month, but it took just over 2 months to receive an official

denial of my request to obtain the LinkedIn Marketing API toolkit which includes the PeopleSearch API.
    - To summarize the LinkedIn data collection, the base test cases for the Cornell Men's and Women's tennis teams ran a search on over 250 names each, and the fields chosen to collect are Name, Profile URL, Location, Company, School, and Activities.

Cleaning and Wrangling the Data

- ➢ Roster Data

The roster page usually includes a headshot photo, which downloads as a player profile link along with a separate label containing the person's name. There is also a field containing the name, but when there is no photo, it is defined as a different field. In the years later than 2001, there are only a few missing photos. After the gap where the data resumes for 1963 back to 1947, there are no photos. The name columns can combined using the fillna() function in order to get all the names in a single column with no missing values. After updating the column names with meaningful descriptions, the roster data contains the following fields:

Name, RosterYear, ClassYear, Hometown, HS, RosterLink

The ClassYear field identifies year in school such as Freshman, Sophomore, etc. The RosterYear identifies the web page such as "2001-02 Men's Tennis Roster". Some school websites include the school name in this label, but others don't. For a person who is on the roster for multiple years, the dataset will have duplicate rows with respect to the Name, Hometown, and HS. The RosterLink appears to be unique for each year, but when clicking the link, the website only renders the most recent from those belonging to the same person.

- ➢ Roster Data Transformations
    - groupby() aggregation : unique, max, count, first, last

In order to collapse multiple observations for the same name, we can use the groupby() function with aggregation "statistics". For RosterYear and ClassYear, we select 'unique' to get a list of the values as a cell in each row under that column. For example, typical 4-year player would have an aggregated ClassYear list of [Freshman, Sophomore, Junior, Senior]. Note that for some schools ClassYear values could include Graduate Student, or Red-shirt. We can use the aggregation 'count' statisic of any column to obtain a value for YearsPlayed, and 'max' to select only one value for the ClassYear field, and 'last' to obtain a single value in each column for the remaining fields.

    - Regex, regular expression text splitting

We can use the regex, or Regular Expression, module in Python to extract the numerical value for the year in RosterYear such as "2001-02 Men's Tennis Roster" into a separate

column. We split on the dash (-) and take the preceding text converted to an integer to obtain 2001. We add 1 to that value to obtain the year pertaining to the Spring season, or the value after the dash which we discarded, as a proxy for graduation year if a Senior year season is present. Although the majority of players do not play all four years, we can obtain the 'max' value from the groupby aggregation to approximate graduation year. We could further extract the maximum ClassYear and calculate the projected graduation year if not already included, but will not do that in the preliminary analysis. This is for consistency when appending the gap roster years from a pdf file of All-Time Letter Winners. That list only has the name and years a varsity letter was earned.

- Nameparser utility function HumanName

The Name obtained from the roster includes the First and Last name as a single string, and for the years 1963 and earlier includes a Middle Initial. The preliminary LinkedIn search performed poorly when including a Middle Initial. Rather than strip it out, it could prove useful to parse it into a separate column for First Name, Last Name, and Middle Initial. Further, there are a handful of Middle Name values that are meaningful, especially for people from countries where the Middle Name is more like part of an extended Last Name. In addition, it is useful to have flexibility to substitute other name types such as Maiden Name vs. Married Name in the search string. The following elements are returned from the HumanName() function, which was used by converting the Name column to a list, and passing each value to a for loop. We can call the elements individually to keep or process them individually.

<HumanName : [

title: ''

first: ''

middle: ''

last: ''

suffix: ''

nickname: ''

]>

In order to distinguish Middle Initials from Middle Names and Initials that precede a Middle Name, we can first strip the period and count the characters to make sure the count is 1. Otherwise, we might mistake a 2-character name for an Initial with a period. We can create a separate column for Middle Initial and Middle Name, such that only one or the other will be populated after checking the character count.

➢ Roster Data Additions

An additional column was added to identify the school, which will be useful for merging

the data from different schools into a consolidated database. Also, a column called Criteria was defined which is used to concatenate with the Name to form the LinkedIn search string. It was used as a short form of College, e.g. “Cornell” instead of “Cornell University”, but the field could contain any value. Further, a Gender classification is added while building the roster list, for the purpose of later combining lists for Men, Women, and different schools.

The All-Time Letter Winner list in pdf format was copied into Excel manually, and wrangled in order to calculate the latest year a letter was earned, and number of years a letter was earned. The names were separated into FirstName and LastName to go along with the names obtained in FirstLast format. The list was checked against the roster download so only names missing from the list were merged into the full roster list.

The full list of 16 columns resulting from the roster wrangling are listed here:

FirstLast, RosterYear, ClassYear, Hometown, HS, RosterLink, Year, Name, First, Last, MidInit, MidName, YearsPlayed, College, Criteria, and Gender

➢ LinkedIn Data

The output of the Roster Data Wrangling step is a list of names and criteria to search with a LinkedIn query. The lists are used in a for loop to simulate the following activity, which is defined in the commercial use limits published at the link below:

https://www.linkedin.com/help/linkedin/answer/52950/commercial-use-limit?lang=en

- Searching profiles by name using the search box located at the top of every page on LinkedIn.com

For example, the base case for testing is my own Name & Criteria “Eric Cruz Cornell”. The data download includes Headline, Location, Company, School, and Activities. Only the current employer name is extracted, but for School and Activities, the full list is returned because people don’t necessarily list their undergraduate school first and sometimes not at all, and same for activities with the word of interest being “tennis”. The profile link is downloaded as “SearchResult” and the name associated with the search is labelled “LName”. For the purpose of indexing, the name used in the search is returned from the for loop in list format, and has to be converted back as part of the data wrangling procedures. To summarize, the list of columns extracted from the search are:

Name, LName, SearchResult, Headline, Location, Company, School, and Activities

Note that many of the searches do not return a result, and many others return a result with an altogether different name. This will be discussed in the exploratory data

analysis.

- ➢ LinkedIn Data Transformations

While building the model, an iterative feature selection process was used, which will be described in the exploratory data analysis section. For the purpose of building the datasets to be run through the model, the final selection of features is built as a Data Wrangling Phase on the LinkedIn data, so it can be passed along to the model to obtain a prediction of whether the SearchResult is considered a good match or not.

The following features are built from boolean values converted to integers based on the the LinkedIn data collected:

School_yn - This indicates whether the LinkedIn profile includes the school. Since the search query criteria used is the school name, it seems likely that the profile found would include the school. However, there are a surprising number of verified search results where the person does not include the undergraduate school where they played on the tennis team. Also, even for a search result that returned for a different name, several profiles did not include the school used in the search criteria

tennis_yn - This indicates whether “tennis” was listed in the Activities section, as you might expect many players to identify their participation on the team.

SameTennis_yn - This feature is built by adding the 0,1 value of the School_yn and tennis_yn features, where 1 indicates True for those features. This measure is designed to give additional weight to profiles which identify both or neither of those features

SameName_yn - This indicates whether the name associated with the LinkedIn search result is exactly the same as the input name.

As many of the names associated with the profiles found are similar but not exact, the FuzzyWuzzy python library was used to calculate how closely the returned name string matches the search name. This is a quick way to distinguish slightly different spellings such as Will vs William, or Pete vs Peter. There are 5 different ratios available, and all were tested. Rather than explain the subtle differences between the matching methods, all were kept in the final feature selection for reasons explained in the feature selection analysis. The names these are: ratio, partial_ratio, token_sort_ratio, token_set_ratio, and WRatio. The result for each is returned as a value between 0 and 100, which is the equivalent of a percentage score on that scale.

At this stage, we merge the Roster data with the enhanced LinkedIn features and include an additional feature “Decade” calculated as a function of the “Year”, to complete the set of features selected for the model.

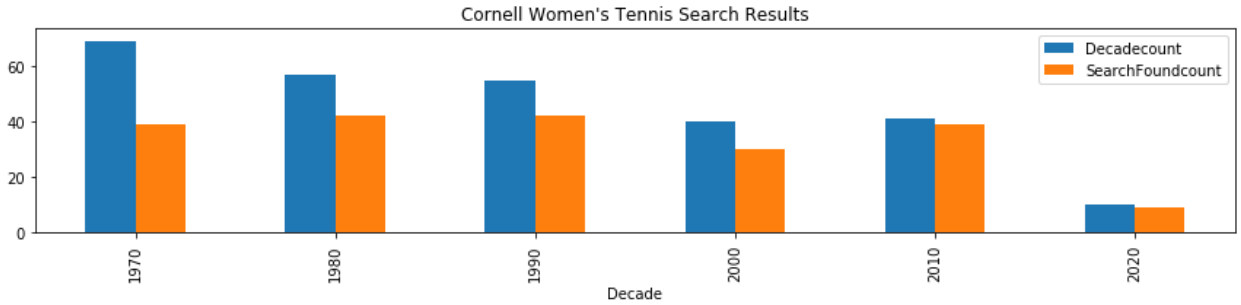
Summary of findings

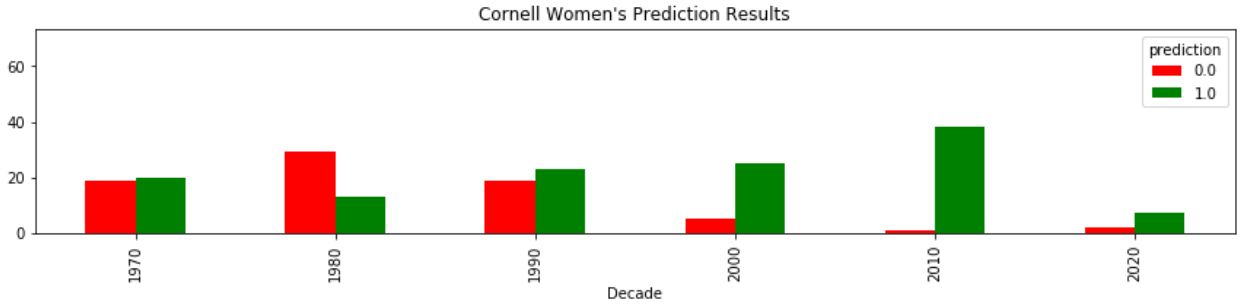
The process described above outlines the collection and wrangling of data to be run through the model in order to provide a prediction of whether each search result actually matches the correct person. As described in the project proposal, the preliminary dataset is a manually curated list of the actual paper rosters and official program supporter contact list. To supplement the contact information, a manual search on each name for Cornell Women's Tennis was conducted on LinkedIn, and this is the labelled data used to build a binary classification model. The test case for gathering the roster and LinkedIn data is the Cornell Men's Tennis program. After obtaining the LinkedIn data and building a predictive model with synthesized features, this data was also manually researched and labelled. The original set of hypotheses are the following:

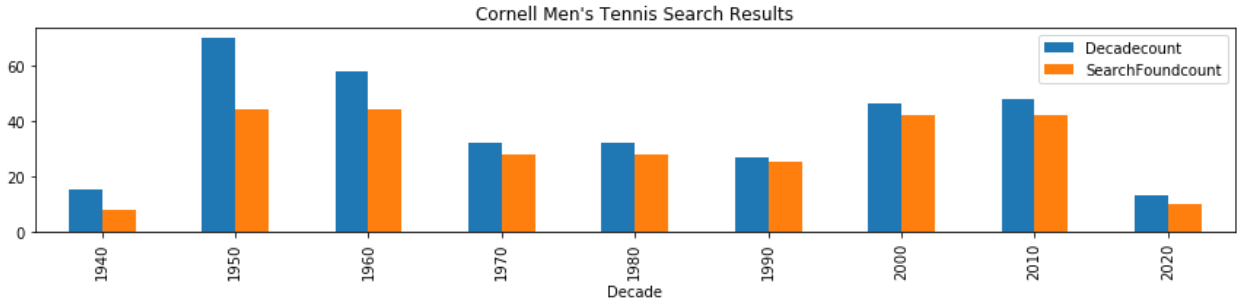
1) LinkedIn profiles are more likely to be found for the younger generation, and the data will reveal a steady decrease in profiles found going back in time.
2) Women will be less likely to be found, especially after a certain age, due to marriage and changing last name to a married name.
3) Profiles which self-identify tennis in the Activities section, and the school associated with the team will be a strong predictor of the prediction accuracy.

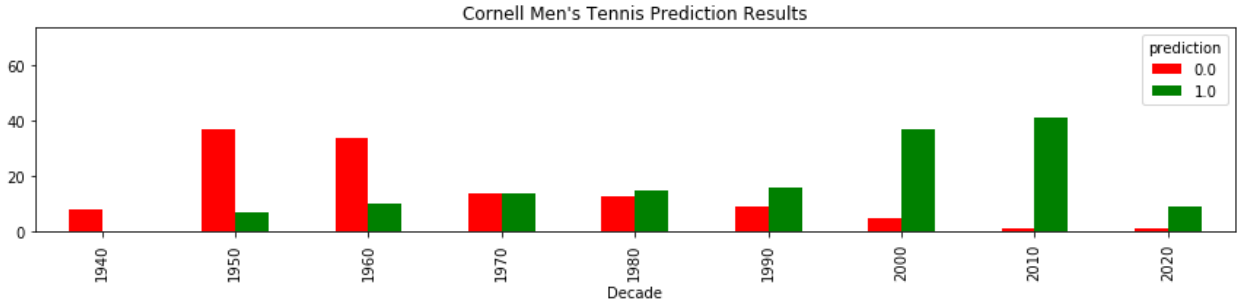
Visuals and statistics to support findings

The result for whether any profile was found did not drop steadily going back in time as I expected, but the prediction of whether the found profile matches the searched person does show that trend. The difference between results for Men and Women is not as pronounced as I expected. The year-by-year result will be shown in the final report, but the decade-level results are depicted with bar graphs to illustrate the general trends on a less busy graph.

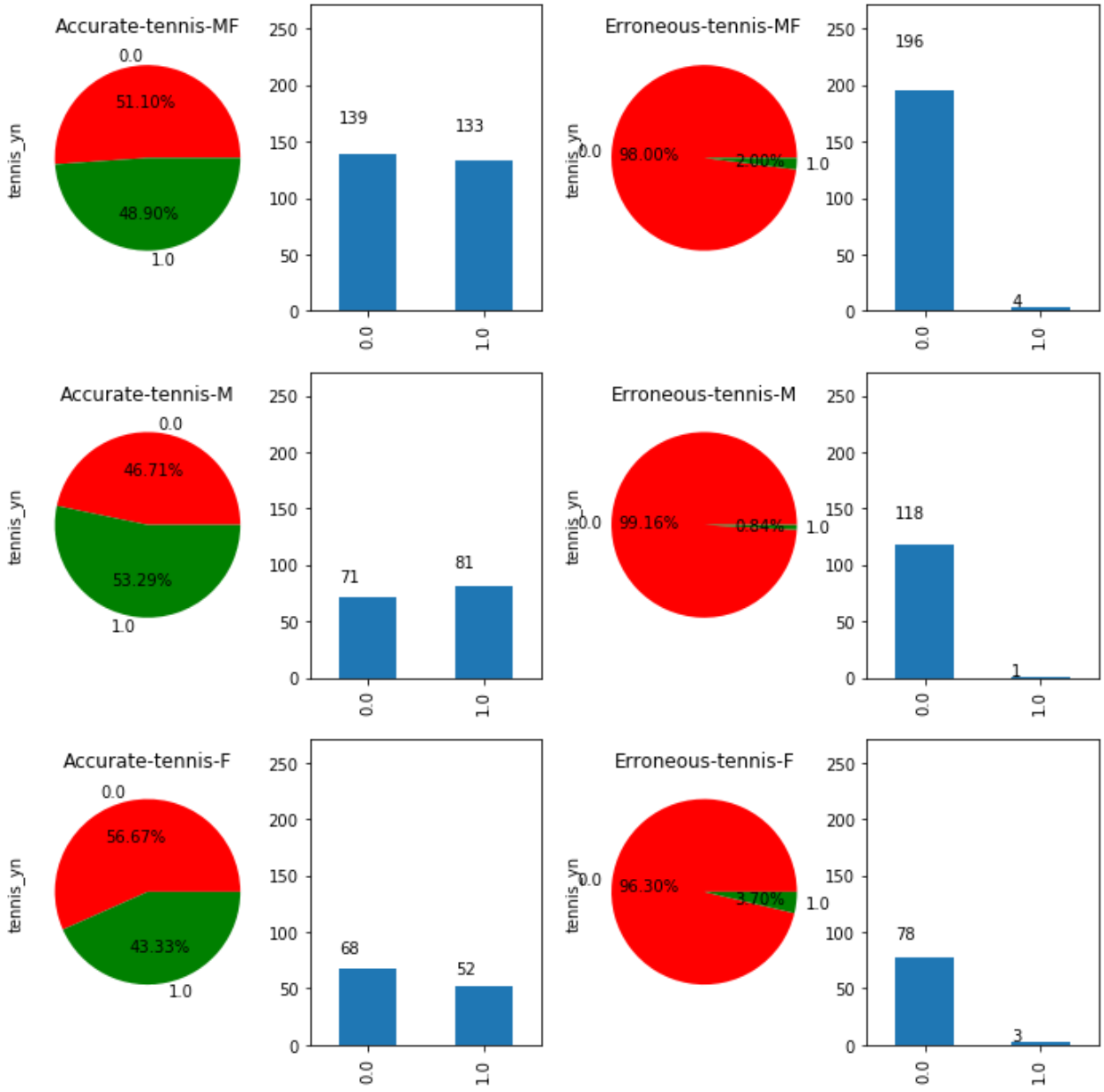




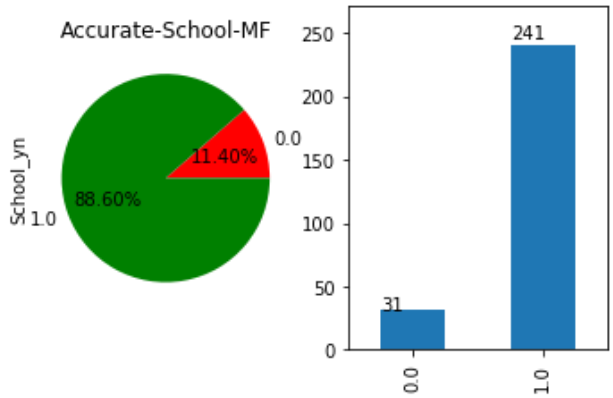


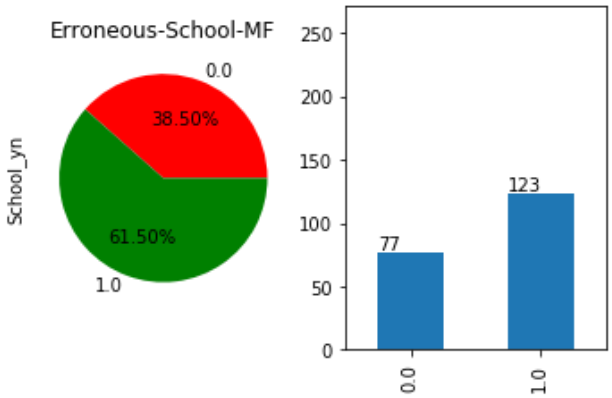


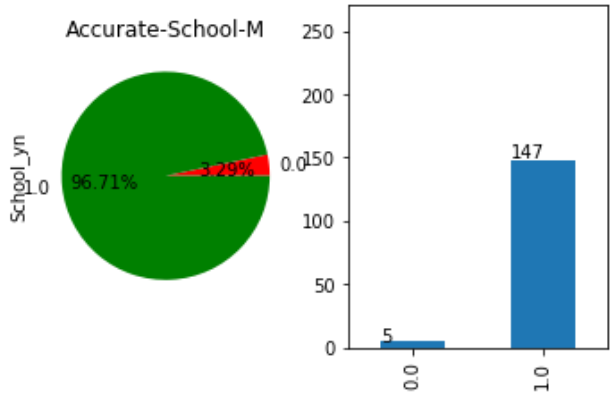
The “tennis” identified in a found profile indicator is highly unlikely to be positive for erroneous predictions, but accurate predictions are more evenly distributed in this regard. Pie graphs are used to show the ratios, and bar graphs annotated with x-values are a quick way to display the distribution count.

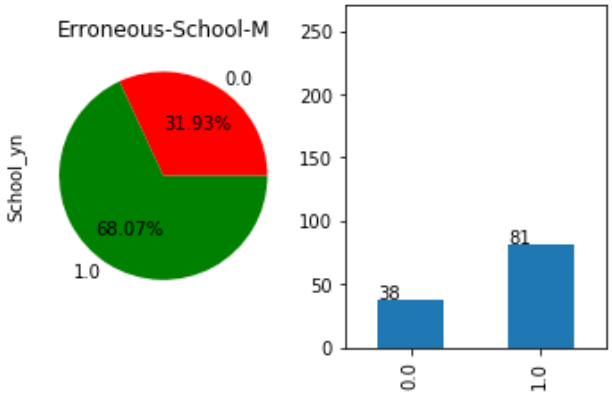


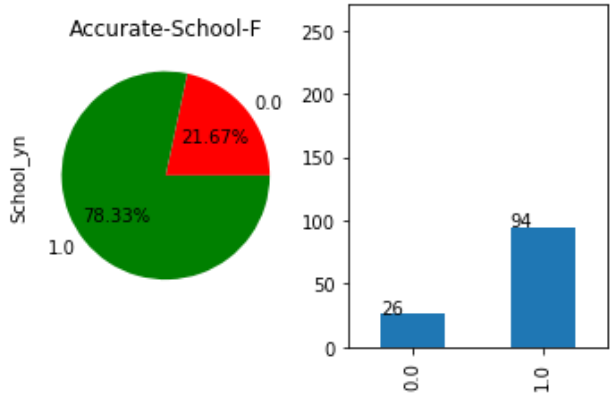
The “school” identified in a found profile indicator is highly likely to be positive for accurate predictions. The ratio can be misleading for erroneous predictions because even a wrong name match is likely to find someone based on the school being used in the search criteria. So these results confirm this feature as a good indicator of prediction accuracy.

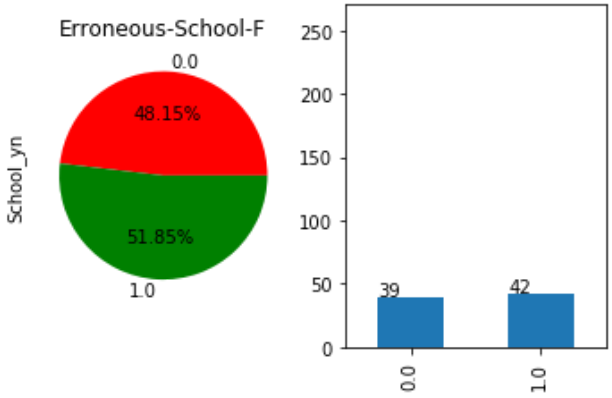

Accurate-School-MF
School_yn
0.0
1.0
11.40%
88.60%
250
200
150
100
50
0
241
31
0.0
1.0


Erroneous-School-MF
School_yn
0.0
1.0
38.50%
61.50%
250
200
150
100
50
0
123
77
0.0
1.0


Accurate-School-M
School_yn
0.0
1.0
3.29%
96.71%
250
200
150
100
50
0
147
5
0.0
1.0


Erroneous-School-M
School_yn
0.0
1.0
31.93%
68.07%
250
200
150
100
50
0
81
38
0.0
1.0


Accurate-School-F
School_yn
0.0
1.0
21.67%
78.33%
250
200
150
100
50
0
94
26
0.0
1.0


Erroneous-School-F
School_yn
0.0
1.0
48.15%
51.85%
250
200
150
100
50
0
42
39
0.0
1.0