Teresiah Kahura

Springboard Capstone 1

May 2020

<u>Problem Statement</u>

The prevailing image of a migrant is that of a low-skilled refugee or perhaps an asylum seeker fleeing war.[1] Less attention is paid to high-skilled migrants who are highly mobile and even fewer attention is paid to football players who play for countries other than those of their birth.[2, 3]

As an example, the 2018 Men's World Cup in Russia was won by the French who fielded a team that had two foreign-born players while the runners up team Croatia had four foreign-born players.[4] I therefore intend to evaluate the hypothesis that having foreign-born players on a team leads to better FIFA rankings in men's football. I hope to provide more clarity on the boon to a host nation of having more open migration policies, with an emphasis on the social cohesion and cultural benefits of having successful men's national football teams.[5] The target audience for this analysis is policy makers evaluating and making decisions on migration, national football governing bodies and fantasy football enthusiasts.

<u>Dataset Wrangling:</u>

<u>DataFrame 1</u>

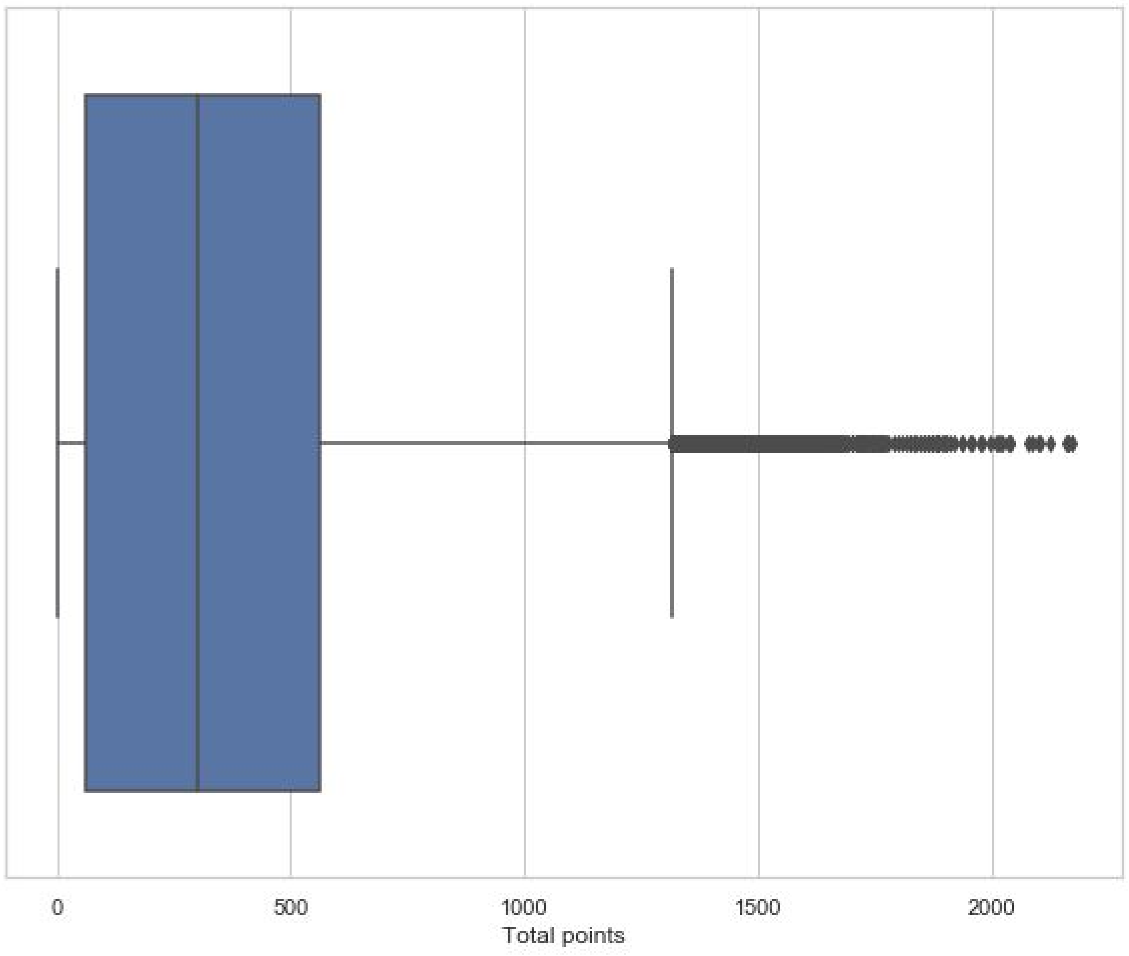
The first dataset was a .csv file compiling FIFA rankings and was found on the Kaggle website.[6] The source material was obtained from the official FIFA website tracking countries which have participated in any FIFA tournament including confederation matches and World Cup matches.

I loaded the file as a DataFrame named [df1]. The resulting DataFrame had 9 columns, all of which had the date range 2019-12-19 to 1992-12-31. There were about 60,000 rows. Inspection of the DataFrame revealed no missing values. I then noticed that the DataFrame had a column named "previous_points" with similar values to the column named "total_points" the only difference being the presence of extra zero digits so I decided to eliminate the former.

For my analysis I decided to focus on the columns named “country_full”, “rank_date”, and “total_points”. I then renamed the “country_full” column to “country” and ‘rank_date’ to ‘date’ in order to prepare the DataFrame for an inner merge with the second DataFrame. I also dropped duplicate values.

I created a box-plot of the “total_points” column and found it to have an outlier value at around 2000 points. After inspecting the “total_points” column, I realized that the rankings in 2018 were higher than in all other years (including 2019) which is a bit counterintuitive as shown in the next plot.

a. Box plot of the ‘total-points’ column



I came to the conclusion that this might be because that year was when the world cup was held. It’s likely that there’s a points bump for countries that participate in the tournament. I decided to use the “rank” column instead to track a country’s progress. I eventually decided to focus only on three [df1] columns for the final statistical analysis: ‘rank’, ‘date’ and ‘country’. The whittled down DataFrame still had about 60,000 rows.

DataFrame 2

The second set of data was obtained after a search through Google’s dataset website. The .xlsx file was created by three researchers at the University of Rotterdam.[7] They pulled data from player Wikipedia profiles and other sources to create tabular data of players who have participated in each FIFA world cup tournament from 1930 until 2018. Special distinction was made between those playing for countries other than their country of birth and those playing for their countries of birth and effort was made to distinguish first-generation migrants and second-generation migrants.

I loaded the file as [df2] . The resulting DataFrame had 12 columns and about 10,000 rows. Close inspection revealed no outliers. Some column names had whitespace in between and in front of names which made it difficult to slice so I trimmed the whitespace. The columns “NationalityFather”, “NationalityMother”, “NationalityGrandmother”, and “NationalityGrandfather” had significant chunks of missing values, but I decided these were not relevant to the analysis so I discarded them.

The columns to be included in statistical analysis are “NameFootballPlayer”, “'International”, “FIFAWorldCup” and “Foreign-born”. I renamed the “International” column to “country” and the “FIFAWorldCup” column (which contains the year the player participated in the tournament) to “date” in preparation for merging.

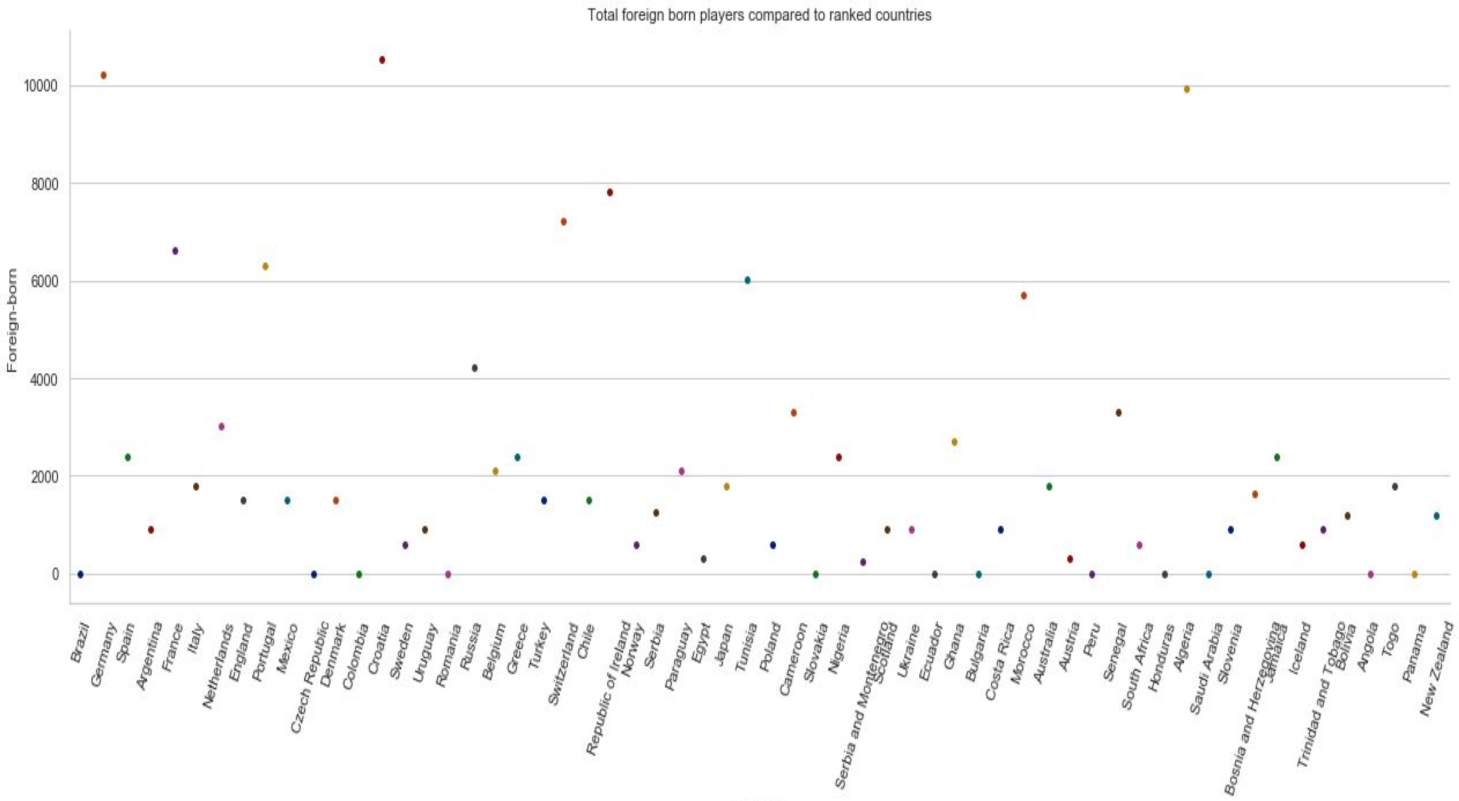
I did an inner merge on [df1] and [df2] on the “country” column in each respective DataFrame. I then dropped the extra date column from [df1]. After merging, I realized the new date column had the digits 1970 added onto the beginning of each date entry. After consulting with my mentor, we decided to do the merge with both original date columns as numeric type. I changed the date column in df1 from object to integer type then continued with the merge with df2 date column which was already in numeric form. This resulted in all the rows in the date column reverting back to the four digit year form.

The merged DataFrame [new_df] now had five columns: ‘date_y’ and ‘rank’, which were numeric, ‘country’, ‘NameFootballPlayer’ which were strings and ‘Foreign-born’ which was Boolean. In preparation for statistical analysis I converted the rank column into category type to minimize the memory requirement. After performing preliminary statistical analyses (described in detail below), I realized the new DataFrame consisted of inflated data most likely from the merging process.

In an effort to minimize the number of rows (at this point numbering at about 2 million) I sliced the rows to only focus on dates from January 1994 onwards. This is because the date range on [df2] stretched back until 1930 and the new FIFA ranking system wasn’t operational until 1993. This cut [new_df] rows by about half. I also went back to the original DataFrames [df1] and [df2] and dropped duplicates which resulted in about 1000 fewer rows in the final merged DataFrame.

I grouped [new_df] according to the ‘country’ column and aggregated the ‘rank’ column with the mean function and the ‘Foreign-born’ column with the sum function. I then made the plot shown next.

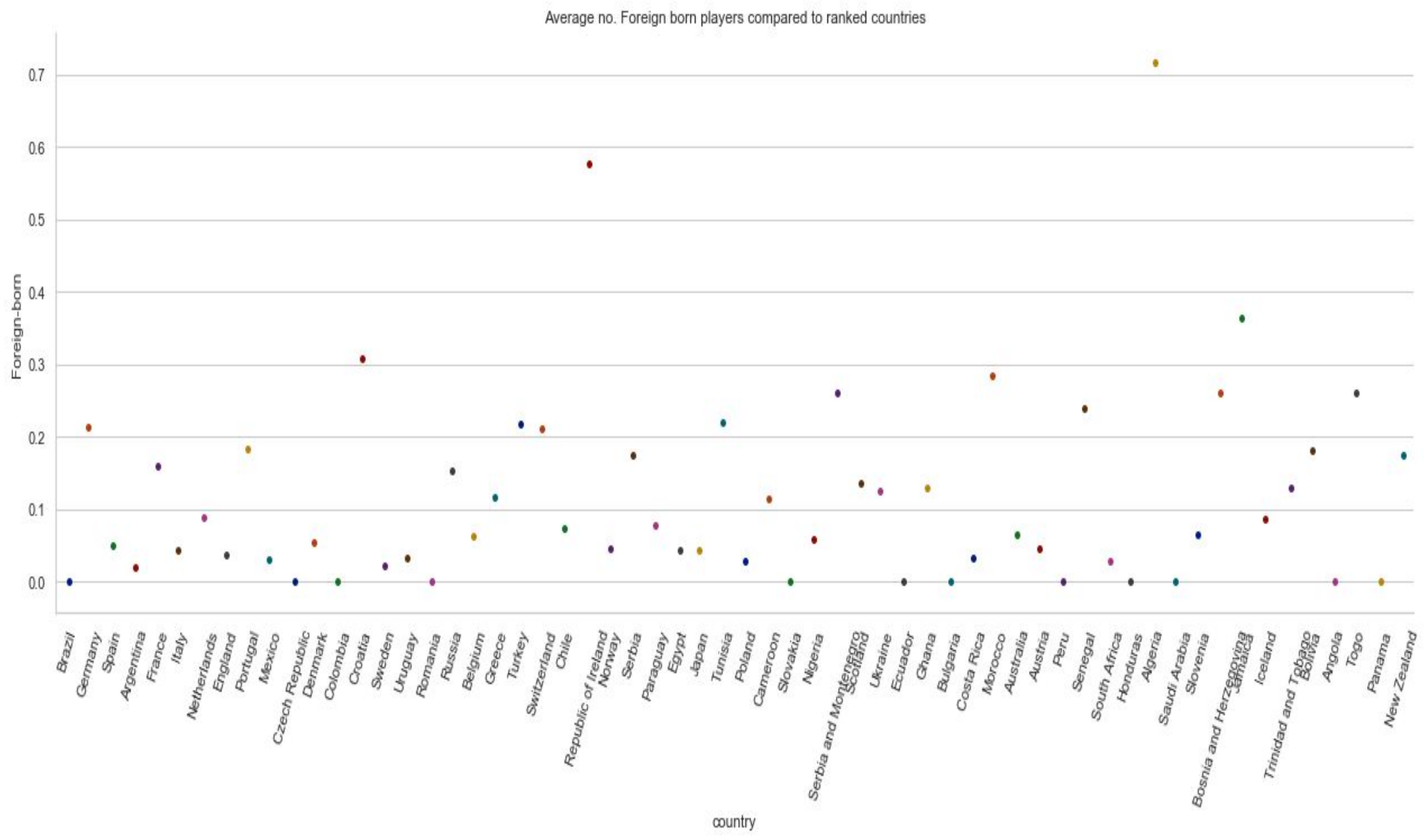
b. <u>Categorical plot of the 'Foreign-born' column totals vs ranked countries</u>



The image shows the number of foreign-born players playing for countries sorted by FIFA rank; highest average rank being Brazil and lowest average rank being New Zealand. This image shows no clear relationship on the rank of countries having no foreign-born players vs countries that do.

The image also shows the aggregation on the 'Foreign-born' column has some atypical values for the sum of foreign born players. It indicates that Germany and Croatia for example have each had a total of over 10,000 players foreign-born players between 1993 and 2018. I decided to change the aggregation function on the 'Foreign-born' column to the mean instead of the sum and created the next plot.

c. Categorical plot of the ‘Foreign-born’ column mean vs. ranked countries



It is now evident that most countries who have competed in the World Cup have had at least 1 player who is foreign-born but there’s no clear relationship between that and the rank of a country.

Hypotheses

- Null hypothesis: there is no difference in ranking between countries that have more foreign-born players vs countries that have no foreign-born players
- Alternate hypothesis: countries which have more foreign-born players have better FIFA rankings

In the new DataFrame the output variable is a country's FIFA ranking (in the 'rank' column) and the input variable of interest is in the column 'Foreign-born'. Since both these columns are categorical, I chose a Chi-squared test to determine whether or not countries that have a higher number of foreign-born players have higher FIFA rankings. The test compares observed frequencies to expected frequencies and from there determines whether an input is independent of the output variable or not. This is determined by comparing the chi-square values: small chi-square values indicate negligible dependence (independence), large chi-square values indicate a gulf between observed and expected frequencies and therefore we can be sure that values are dependent. From there, any independent inputs can be discarded from our analysis.[8]

The resulting chi-squared value was very high: 5,042,748 (the observed frequency is very high as opposed to the expected frequency under the null hypothesis) and the p-value was 0. The large chi-squared value is unusual but the original data set was also very large with about 2,450,000 rows. This inflated data was most likely as a result of the merging process where duplicate entries in the country columns of both original DataFrames resulted in multiple combinations in the merged DataFrame.

At this point, I sliced out and decided to focus on rankings from November 1993 as described above to further minimize the DataFrame, and dropped duplicates from the original DataFrames which halved the original number of rows. After running the test again, the new value was 2,632,748 with a p-value of 0. In tandem with acknowledging the limitations of our result above, we can then reject the null hypothesis that there is no difference in ranking for countries that have more foreign-born players vs countries that don't. This is because if the assumption of the null is true, we would get the result above only 0% of the time. While there is a regrettable limitation, the structure of our original DataFrames doesn't allow any other wrangling that doesn't drastically whittle down our data.

In-Depth Analysis

The new dataframe [new_df] had three non-numeric columns: 'NameFootballPlayer', 'country' and 'Foreign-born' and two numeric columns: 'rank' and 'date' (in datetime format). Given that I was trying to predict a country's rank I decided to approach the problem from a multiclass classification perspective. I chose several classifier models to test my hypothesis, namely: Decision Tree, Random Forest and Gradient Boosting Classifier.

I began by changing the data type of the column 'NameFootballPlayer' to object and the columns 'Foreign-born' and 'country' to category in order to make the models run more efficiently. I then started preprocessing by first defining the matrix of features (all the columns except for ‘rank’) and the target variable (‘rank’ column).

I then used OneHotEncoder to obtain dummy variables on the matrix of features (X) and LabelEncoder to obtain dummy variables on the target variable (y) since we care about preserving the order of the predicted rank. The resulting X had about one million rows and about three thousand columns while y had about one million rows and 1 column.

I then divided both X and y into a training set and a test set with about 70% of the tranche going to both training segments of X and y. The training portion of X had about nine hundred thousand rows and about three thousand columns while the training portion of y had a similar number of rows but only one column. The test portion of X had almost four hundred thousand rows and about three thousand columns while the test portion of y had a similar number of rows and 1 column.

Modeling: Decision Tree

I instantiated an untuned Decision Tree and trained it on the training portions of X and y. I then used the test portion of X to predict a new set of labels. The accuracy of this model was 8.2% with the classification report showing significant discrepancies in the per class classification metrics between the first class and the last class. The dataset showed imbalance as evidenced by the large range in the support values over different classes.[9] For example, class 151 had four occurrences while class 0 had over thirteen thousand occurrences.

The weighted average F1 score (harmonic mean between precision and recall) which accounts for class imbalance came out to 6%. This value was highest for class 0 with a value of 42%, then declined with the majority of classes recording 0%. The precision (measuring how many are correctly classified in a particular class) also varied widely with a weighted average of 5% . Class 0 had the highest value at 29% which declined over the remaining classes with a majority having a value of 0%.[10]

The recall which indicates how well the classifier can find instances of a certain class from the dataset had a higher weighted average at 8% with the highest value being 72% at class 0. It also declined over subsequent classes. Similar to precision, the majority of the classes had a value of 0%.[10]

I then used RandomSearchCV to tune hyperparameters for the tree. I chose four parameters: max_depth (controlling tree size), criterion (to measure the quality of a split), max_features (indicating the number of features to consider before making the best split) and min_samples_split (which indicates the minimum number of samples required in a node in order to execute a split).[11]

Cross-validating on the above returned max_depth of 30, criterion of ‘gini’, max_features of ‘auto’ (square root of the number of features), and min_samples_split of 2. After fitting the new tuned tree on the training portions of X and y, I used the test portion of X to predict a new set of labels. I then ran the classification report which was very similar to the first untuned tree. The weighted average values for precision, recall and F1 score were 5%, 8% and 6% respectively, similar to the untuned tree. Class 0 had the highest precision, recall and F1 score at 29%, 72% and 42% respectively. These values declined over the remaining classes similar to the first untuned model with the majority of classes having values for each respective metric at or around 0%. The accuracy of the tuned tree came in at 6%, lower than the first tree.

Modeling: Random Forest

I instantiated an untuned Random Forest model and fitted it on the training portions of X and y. I then used the test portion of X to predict a new set of labels. The accuracy of this model was 8.2% similar to the first untuned Decision Tree model. The classification report showed only a modest improvement over the one obtained from the Decision Tree. The weighted average values for precision, recall and F1 score were 5%, 8%, and 6% respectively, similar to the Decision Tree.

All the metrics (F1, precision, recall) were highest for class 0 similar to the previous model. The F1 score for this class came in at 43% only outperforming the Decision Tree by 1%, while the precision score was 30% (similarly outperforming the prior model by 1%). The recall score for this class was 43% (also outperforming the Decision Tree by 1%). For all three metrics, the majority of the classes had a score of either 0% or close to 0% similar to the Decision Tree.

I then used RandomSearchCV to tune hyperparameters for the Random Forest. These were: n_estimators (the number of trees), max_features (the number of features to consider before making the best split), max_depth (controlling tree size) min_samples_split (the minimum number of samples required in a node in order to execute a split), min_samples_leaf (minimum number of samples required in a leaf node), and bootstrap (indicating whether to build bootstrap samples or use the whole dataset).[12]

Cross-validating returned these values for the above parameters: n_estimators of 27, max_features 'auto' (square root of the number of features), max_depth of 22, min_samples_split of 2, min_samples_leaf of 1 and false for bootstrapping. After fitting the new tuned forest on the training portions of X and y, I used the test portion of X to predict a new set of labels. I then obtained the classification report which was very similar to the first untuned forest. The weighted average values were: precision at 4%, recall at 8% and F1 score at 4%.

Class 2 had the highest precision score at 17%, while the highest recall value was found in class 1 at 47%. Class 1 had the highest F1 score at 23%. Similar to the previous models, the majority of the classes had metric values at 0%. The accuracy of the tuned Random Forest was 7.9%, a decline from the untuned forest.

Modeling: Gradient Boosting Classifier

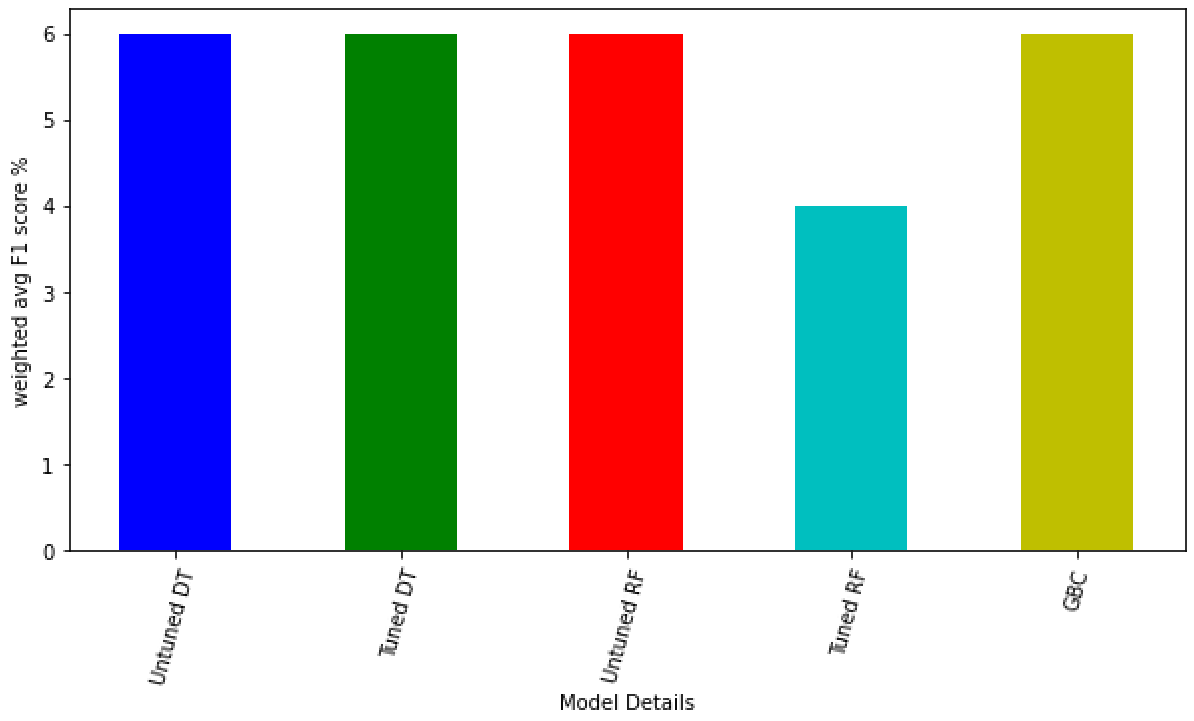
I instantiated a Gradient Boosting Classifier model with only 5 estimators and fitted it on the training portions of X and y. I then used the test portion of X to predict a new set of labels. I obtained the score which came out to be 8.4%. Due to the long run-time of the model and the fact that it didn't perform significantly better than all previous models, I decided not to tune it further and conclude my modeling.

Conclusion

The goal of this project was to evaluate the hypothesis that having foreign-born players on a team leads to better FIFA rankings in men's football. Running the chi-square test of independence suggested indeed that the two might be dependent. This might have been influenced by an inflated data set that arose from the merging process of two data frames. Nevertheless, I decided to train several classifier models to test my hypothesis, namely: Decision Tree, Random Forest and Gradient Boosting Classifier. The goal was to see if these models could predict the rank of a country based on the number of foreign-born players on its team.

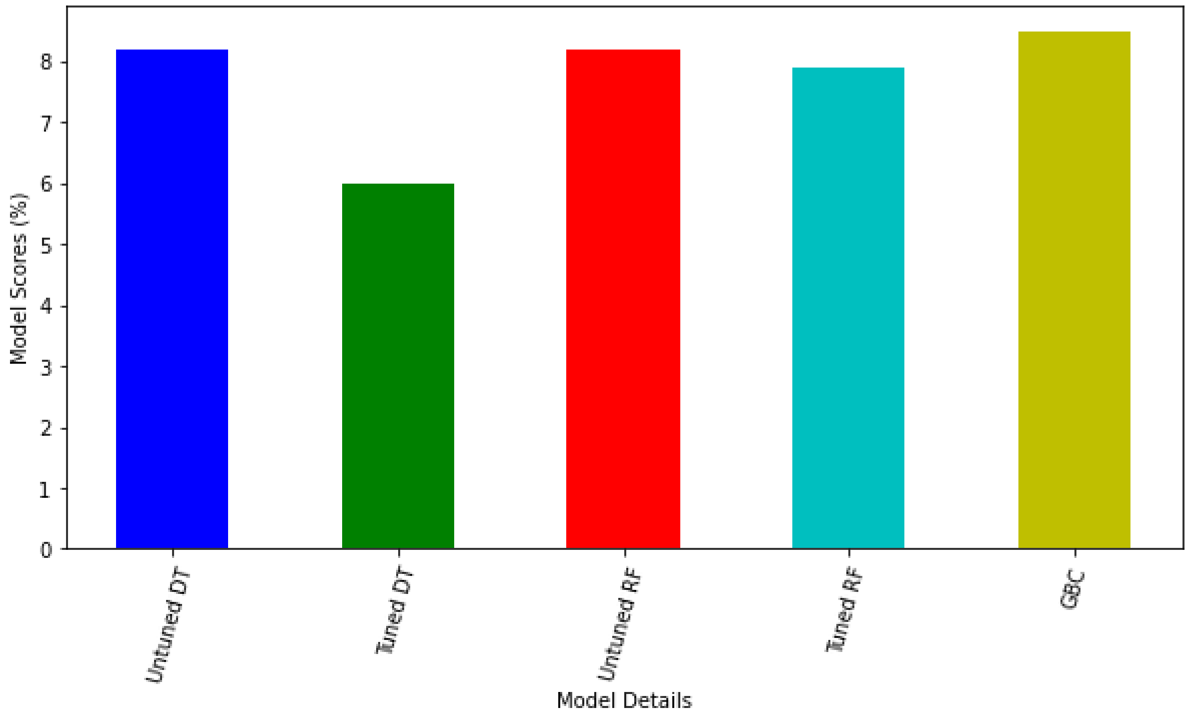
Given that the data showed considerable imbalance, a more useful measure of classifier performance is the F1 score.[10] All three classifiers performed poorly with a weighted average score of around 6% with the tuned Random Forest pulling in a value of 4%.

d. Boxplot of weighted average F1 scores



The classifier scores were similarly modest. The majority of the classifiers had a score hovering at around 8%; the Gradient Boosting Classifier registering the highest score at 8.5%, while the tuned Decision Tree came in last with a score of 6%. Despite the poor performance across the board, the Gradient Boosting Classifier seems to have better capability with this particular dataset with the only limitation being its long training time. Perhaps with careful feature engineering one can minimize the number of rows while increasing the number of dependent features to increase model performance in tandem with decreasing training time.

e. Boxplot of accuracy scores for each model



Future Scope

As mentioned previously, feature engineering to bolster the number of features in the dataset is worth exploring to potentially increase model performance. The dataset I trained on only had one feature which showed dependence (‘Foreign-born’ column).

My dataset consisted of a very limited date range (every four years beginning from 1994) because FIFA instituted a new ranking system at the end of the 1993 season. It would be interesting to explore the problem from a time series perspective if the datetime column would be bolstered.

References

1. https://reimaginingmigration.org/what-are-the-predominant-stereotypes-about-immigrants-today/
2. https://www.iom.int/jahia/webdav/shared/shared/mainsite/about_iom/en/council/88/MC_INF_277.pdf
3. https://www.infomigrants.net/en/post/19733/migrating-for-football-the-harsh-reality-behind-the-dream
4. https://the18.com/soccer-learning/foreign-born-players-in-the-world-cup-final-2018
5. https://www.unicef.org/lebanon/press-releases/football-social-cohesion-project-how-sport-bringing-together-lebanese-and-syrian
6. https://www.kaggle.com/cashncarry/fifaworldranking
7. https://dataverse.harvard.edu/dataset.xhtml?persistentId=doi:10.7910/DVN/TE9KWG
8. https://machinelearningmastery.com/chi-squared-test-for-machine-learning/
9. https://scikit-learn.org/stable/modules/generated/sklearn.metrics.precision_recall_fscore_support.html
10. https://blog.exsilio.com/all/accuracy-precision-recall-f1-score-interpretation-of-performance-measures/
11. https://scikit-learn.org/stable/modules/generated/sklearn.tree.DecisionTreeClassifier.html#sklearn.tree.DecisionTreeClassifier
12. https://scikit-learn.org/stable/modules/generated/sklearn.ensemble.RandomForestClassifier.html