Individual Project Report
Jason Witry

Introduction

Usain Bolt winning the 2008 100 meter race. Who could have predicted it? Given that he lost in 2004, the only defeat in his career, you would be hard-pressed to have said in 2007 that he would take the gold in Beijing. What happened in between? The answer is that Bolt trained meticulously. He even said so himself, "I really enjoy what I do, and I know the hard work pays off. On the track, it's all about staying number one." It's all about the hard work. If you kept up with Bolt's training regime, his diet, hydration habits, and other training indicators, you might be able to have made a convincing prediction that Bolt would shatter world records in 2008. However, how would you collect this data? Would you collect the information from each athlete to compare? Imagine the enormous resources and time required to collect this data, and this method quickly becomes infeasible.

Let's think a little differently; every Olympic Games, data is collected on the athletes, such as height, weight, age, team, country of origin, and more. Since this data is already available and has been for over a century, we do not have to wait to collect training data for all future Olympic Games to get a decent sample size. This is an enormous advantage, so only one question remains; Can we use athlete characteristics to predict Olympic medal winners as a proxy for training data and skill level?

Below is an outline of the work done by the group. I wrote the introduction and problem statement in the paper, and acquired the data from Kaggle. Jing covered the Exploratory Data Analysis and created the analysis figures, from which I considered how the data would affect our preprocessing and modeling decision, such as dropping the year below a threshold. Sarvesh covered the coding of the preprocessing, and we worked together in understanding preprocessing steps needed, such as balancing the dataset. Sarvesh created the models, both Logistic Regression and the Random Forest. I took all of his preprocessing code and edited it as necessary to create the GUI, for instance adding a new train test split after any data imputation.

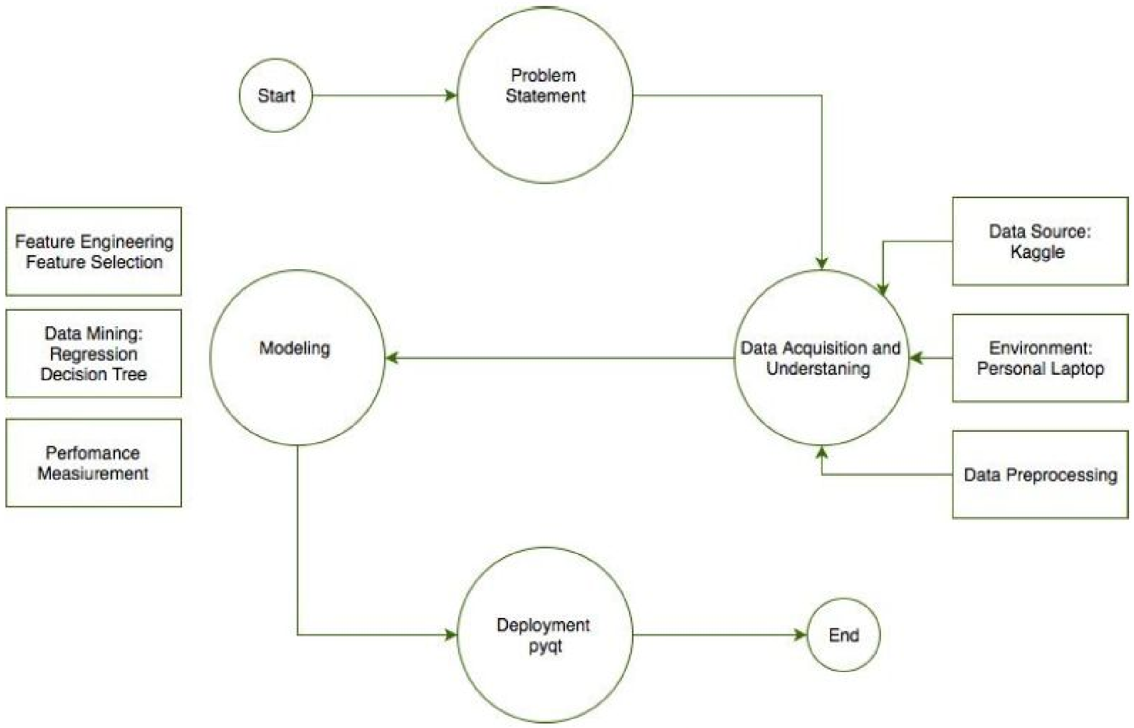

Start
Problem Statement
Data Source: Kaggle
Environment: Personal Laptop
Data Preprocessing
Data Acquisition and Understaning
Modeling
Feature Engineering Feature Selection
Data Mining: Regression Decision Tree
Perfomance Measiurement
Deployment pyqt
End

Figure 1: project workflow

Individual Work

We developed this algorithm using Jing's exploratory analysis, discovering relationships between variables that could be important to our model. For example, we discovered that over the years, the physical characteristics, age and gender of an Olympic athlete have changed significantly over the years due to various events in human history. This noise is primarily present in older Olympic data, when women were not allowed to compete and the Games were not as much of a household name as they are today. This lead to the idea that we might gain performance by dropping years prior to a threshold year. We also realized several potential connections in our dataset, such as that specific countries may tend to perform better in specific sports or events. This is supported by the graphs that show medals won by country and the breakdown of each countries sports they medal in. The US, for example, has a fantastic swimming team and takes most of its medals from swimming. These are further reason to believe that we may be able to use the current dataset to predict Olympic champions, and it is not as far fetched as it looks at first glance.

My individual work consisted of writing the GUI code, and determining how to implement the preprocessing and modeling code into the GUI. I also wrote the introduction for the paper, the results and conclusions, and edited some of Jing's work on the data exploration to relate it to why we made the preprocessing and modeling choices we did. For example, I wrote that there is noise in the data set from fluctuations in the age, height and weight distributions in earlier years for both Summer and Winter games, which meant that the model improved after removing this noise. I also interpreted why the Logistic Regression did not perform as well as the Random Forest, which was because the data was mainly categorical. My GUI was used to create the confusion matrix figures, with an example shown below of a Random Forest with preprocessing (mean imputation by country and gender, removing years prior to 1990) and oversampling.

Detailed Description

I wrote the code for the entire GUI, writing the overall structure and fitting Sarvesh's functions into the framework. The overall structure of the GUI is that the Window class, the main portion of the GUI, creates all of the visuals such as buttons and drop down menus. In the initialization step, it creates a preprocessing object using the Olympic_Analysis.py module to access the data and all preprocessing and modeling steps. The Window class consisted of code like this:

```
btn0 = QtGui.QPushButton("Reset Dataframe",self)
btn0.clicked.connect(self.Reset_Data)
btn0.resize(btn0.minimumSizeHint())
btn0.move(0,50)
vBoxlayout.addWidget(btn0)
```

for creating buttons, and subroutines such as

```
def Impute_Height_Dataset_Wrap(self):
  self.progress.setRange(0,0)
  self.Imputation = TaskThread(self.prep,6)
  self.Imputation.taskFinished.connect(self.Imputation_Finished)
  self.Imputation.start()
```

to wrap around Sarvesh's functions that I put in the function module, Olympic_Analysis.py. self.prep is the preprocessing object, which is passed into a multi-threaded class that runs the imputation on one thread and the progress bar in the GUI on another.

I created additional subroutines and edited some of Sarvesh's functions to fit within the GUI structure, like so:

```
def Standardize(self):
  cols = self.df1.select_dtypes(include=['float64']).columns.values
  self.df1[cols] = self.sc.transform(self.df1[cols])
  self.X_train[cols] = self.sc.transform(self.X_train[cols])
  self.X_test[cols] = self.sc.transform(self.X_test[cols])
  #X_combined_std = np.vstack((X_train_std, X_test_std))
  #y_combined = np.hstack((y_train, y_test))
```

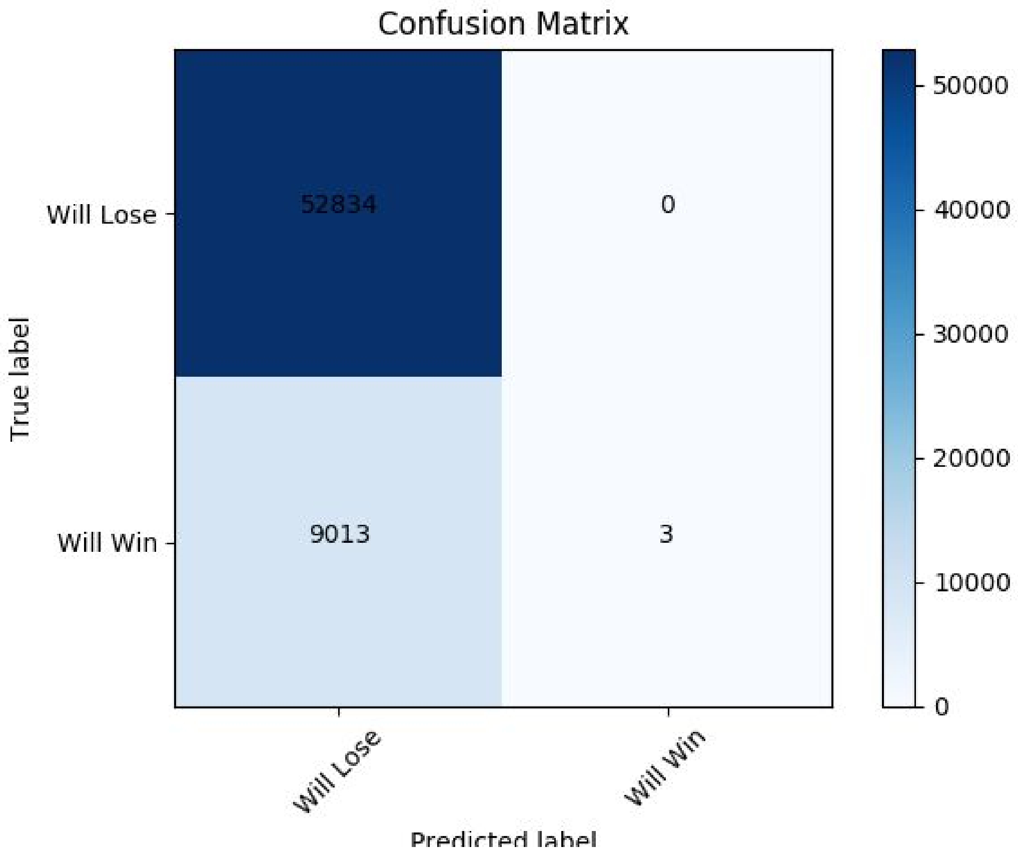
where the above standardizes the variable according to a fit described in the following train-test split subroutine:

```
def Train_Test_Split(self):
  self.df1_nona = self.df1.dropna()
  X = self.df1_nona[self.train_cols].copy()
  y = self.df1_nona['Medal']
  self.X_train, self.X_test, self.y_train, self.y_test = train_test_split(X, y, test_size=0.3, random_state=0)
  cols = self.df1.select_dtypes(include=['float64']).columns.values
  self.sc.fit(self.X_train[cols])
```
Because after splitting the data into training and test, we must recompute the standardization fit to be available to the user if they choose to standardize after imputing. The majority of the code I wrote was in the GUI module, organizing the subroutines into a format compatible with the GUI. In this sense, all of the Olympic_Events_GUI.py code was written or found by me, and the Olympic_Analysis.py module consists of preprocessing and modeling code written by Sarvesh, with edits from me to make the entire code compatible.

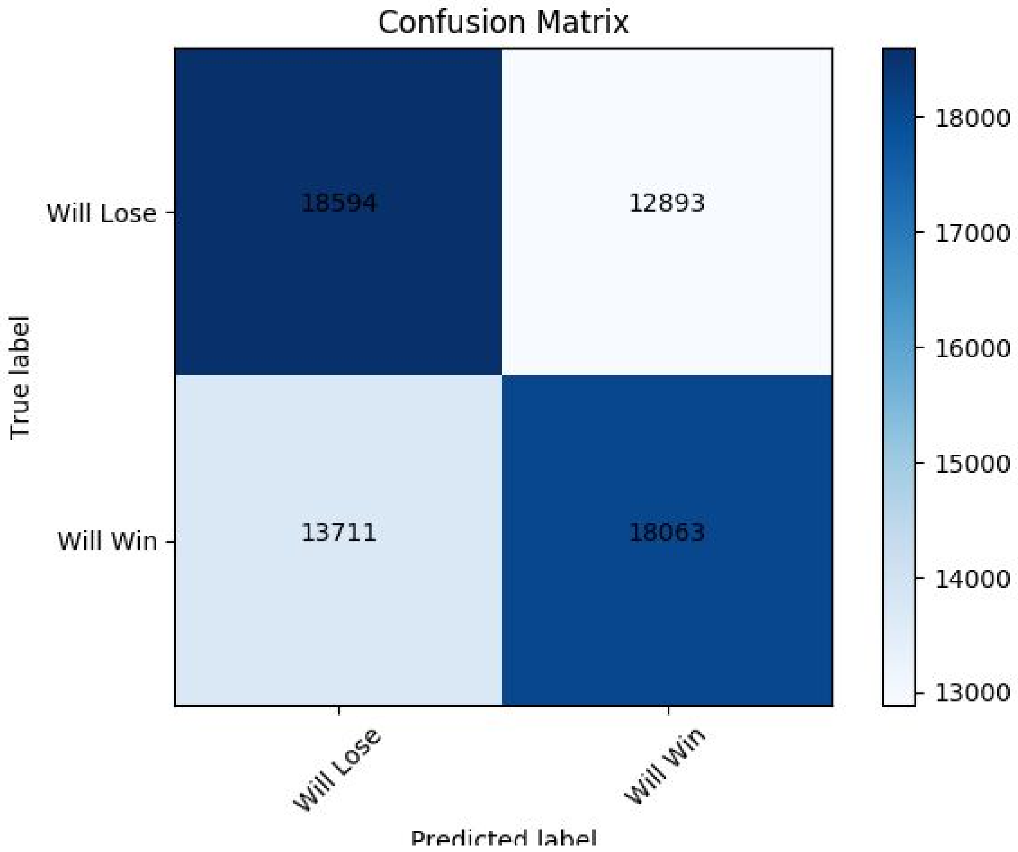
I also provided the explanations that link Jing's data exploration to our model and preprocessing choices. For example, I explained that because the medal by sport varies between countries, this could be an indication that certain countries excel at specific sports. I wrote the problem statement and the introduction in the paper, stating that essentially our goal was to use readily available and previously gathered data to accomplish the prediction, rather than difficult to access and expensive/time-consuming to collect data on individual athletes training and skill level.

Results

We can see clearly that preprocessing the data improves the model significantly. Without preprocessing, the models suffer from an imbalanced dataset and many missing values. In the case of Logistic Regression, the confusion matrix without preprocessing is shown in the figure below.

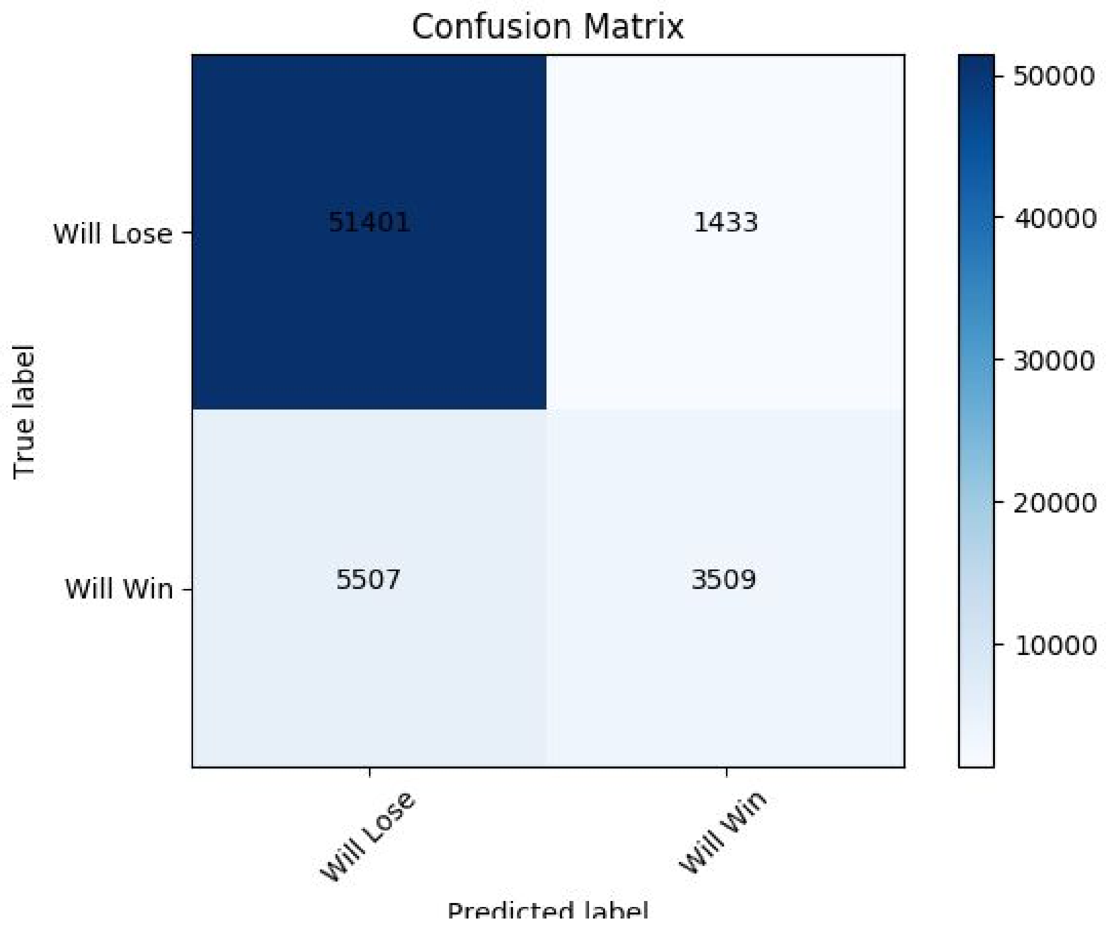


We can see that the model rarely predicts medal winners correctly, and the recall for medal winners is near zero. The precision for medal winners is 1, normally a perfect score, but the confusion matrix shows that we rarely predict someone is a medal winner, so this is not a reliable conclusion. However, once we preprocess the dataset, we find that the predictions improve significantly. In the case of oversampling in addition to mean imputation, standardization, and cutting out years before 1990, logistic regression returns the following confusion matrix.



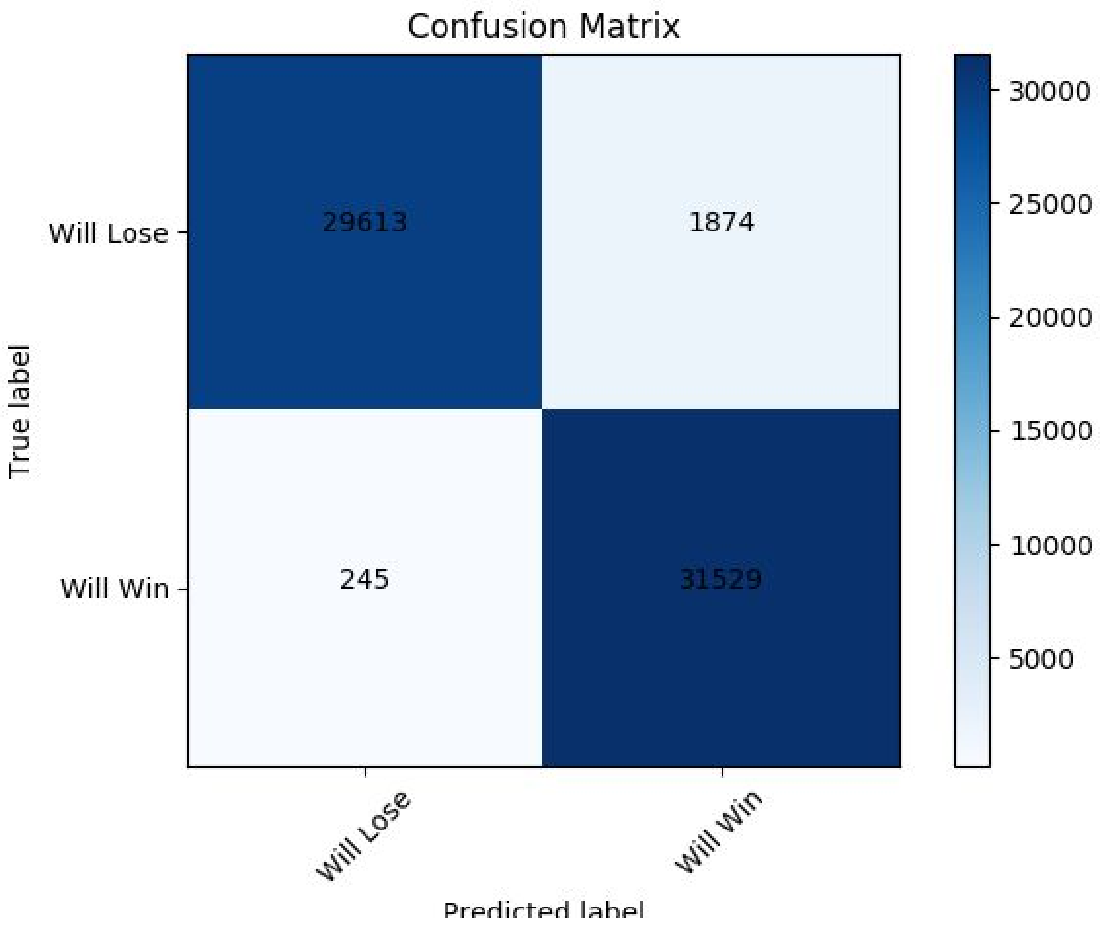
While the recall for winners drops significantly, this is a more reliable prediction, as we do not suffer from the class imbalance. The model also benefits from less noise due to years that are cut off.

For a Random Forest model using 10 predictors, we see the following results with no preprocessing.



The precision and recall for the non-medal winners is high, as expected from our imbalanced dataset. Similar to the logistic regression model, the precision and recall are lower. However, the Random Forest model seems to perform better on the dataset than Logistic Regression in this case, as we end up with more predictions for medal winners.

A Random Forest with preprocessing returns the following results.



We see a drastically improved precision and recall for each class prediction, above 0.9 for each. This is the best model we found for our data.

Summary and Conclusion

In conclusion, a Random Forest is the best model for our dataset, along with the preprocessing steps of considering years after 1990, mean imputation, and upsampling our non-medal winners. The Random Forest performs better than Logistic Regression due to the fact that our dataset is primarily categorical variables. Random Forests handle categorical variables in a much more natural way than Logistic Regression. Considering years after 1990 not only removes noise in the age, height and weight distributions, it also makes our predictions more relevant for future data like the 2020 Olympics. Imputing the missing values with the means prevents the loss of potentially relevant information, and our method using the Country of Origin and Sex is better than simply using the dataset mean. Finally, upsampling our dataset balances the classes, which is an important step for the Random Forest model, especially when it comes to pruning the trees. We attempted to mitigate this by setting the max depth rather than the minimum sample pruning techniques, which could lead to challenges of its own. Interestingly, letting the Random Forest algorithm run to completion without pruning does not seem to cause overfitting in our dataset. This suggests that our splits seem to divide the dataset well, and there are not many nodes with significant class imbalances in them.

As a result, we can predict Olympic medal winners with the data collected each Olympic Games, and yield a meaningful result that returns a high accuracy, precision and recall for our classes. It seems that there are significant relationships outside of training regimens that make a champion. However, given that training regimens result in different physiques for an athlete, the training regimen likely influences the height and weight relationship of an athlete. Also, younger athletes are in general stronger and more fit than older athletes, which can be important for physical events. Conversely, older athletes may perform better in more skilled based events, such as Archery. The age in this case likely reflects years perfecting the skill, which is an important feature for many events. In summary, we were able to accomplish the task of predicting Olympic champions using data that is readily available and cheap to gather, meaning future predictions of Olympic winners likely do not need to gather data on training regimen, years perfecting the skill, and other difficult information to gather.

Future steps will be to apply our model to the 2020 Olympic winners and test how well we predict them. We also may be able to extend our model to predict which medals athletes will win (bronze, silver or gold) using the Random Forest. The Random Forest can naturally handle the ordinality of this new target.

For the GUI, using a different method to impute the values may result in better performance. As it stands, the imputation of all the variables is a very time-consuming process. It is a for loop through all of the data, where we could use a default function like df.fillna() to impute the values.

Percentage of code copied from Internet:

42.34%

References

https://www.kaggle.com/heesoo37/120-years-of-olympic-history-athletes-and-results#noc_regions.csv

https://wiki.python.org/moin/PyQt4

https://scikit-learn.org/stable/

https://www.worlddata.info/average-bodyheight.php

https://en.wikipedia.org/wiki/List_of_average_human_height_worldwide

http://www.wecare4eyes.com/averageemployeeheights.htm