Problem Statement:

The problem is being able to predict the outcome of any match-up between two teams in future world cup games in order to best make business decisions related to those predicted results. Can we accurately predict the winner among any two teams in a future world cup? Which team will win any particular match-up? Any company interested in the outcomes of the world cup regarding the various teams to make important business-related decisions can benefit from this enormously. For example, if a company is deciding what team to potentially have a sponsorship deal with or which team to utilize in carrying their logo on their jersey. If the algorithm can accurately predict the winner of any match-up in the world cup, it will help the client decide which team to make a deal with. The client will want to make a deal with the predicted winner for increased exposure and attention to their brand.

Data Cleaning and Preparation:

The data I used consisted of a CSV file found on Kaggle. The CSV file included data regarding all the world cup matches since 1930. For each match, there were many data fields describing the match such as home team name, away team name, stadium, stage, attendance, referee, city, match ID, home team goals, away team goals, and more. I started by converting the CSV file into a Pandas data frame with 'Year' as the index. With Year as the index, I only included data from 2002 onwards, as data from world cups before that was too old to provide valuable predictive power and insights on future world cups. Afterwards, I noticed there were multiple duplicate rows. It was very important to remove these redundant records as to not skew the data. Thus, I dropped the duplicate rows and the number of matches/records was reduced from 270 to 256. There was no need to remove records with missing values as all the relevant data

was there. I now had a Pandas data frame with data from 2002 onwards with no duplicates and was ready to perform exploratory data analysis.

Exploratory Data Analysis:

I initially wanted to get some insight into the winning percentages by year for the world cup teams in the dataset. Also, I was only interested in teams that qualified for all four world cups (2002, 2006, 2010, 2014), so teams that only played from 2006 onwards for example were excluded. I used the winning percentage rather than simply wins because I wanted to take in to account how many games each team won with respect to how many games they played. I knew this would give me valuable information regarding the success of each team year by year. In order to do this, I created four separate data frames, one for each world cup of 2002, 2006, 2012, and 2014. For each data frame, I calculated the winning percentage for each team that qualified for all world cups since 2002 and then concatenated the 4 data frames into one.

Resulting Data Frame:

	Argentina	Brazil	England	France	Germany	Italy	Japan	\
2002	0.333333	1.000000	0.40	0.000000	0.714286	0.250000	0.5	
2006	0.600000	0.800000	0.60	0.571429	0.714286	0.714286	0.0	
2010	0.800000	0.600000	0.25	0.000000	0.714286	0.000000	0.5	
2014	0.636364	0.363636	0.00	0.571429	0.888889	0.333333	0.0	

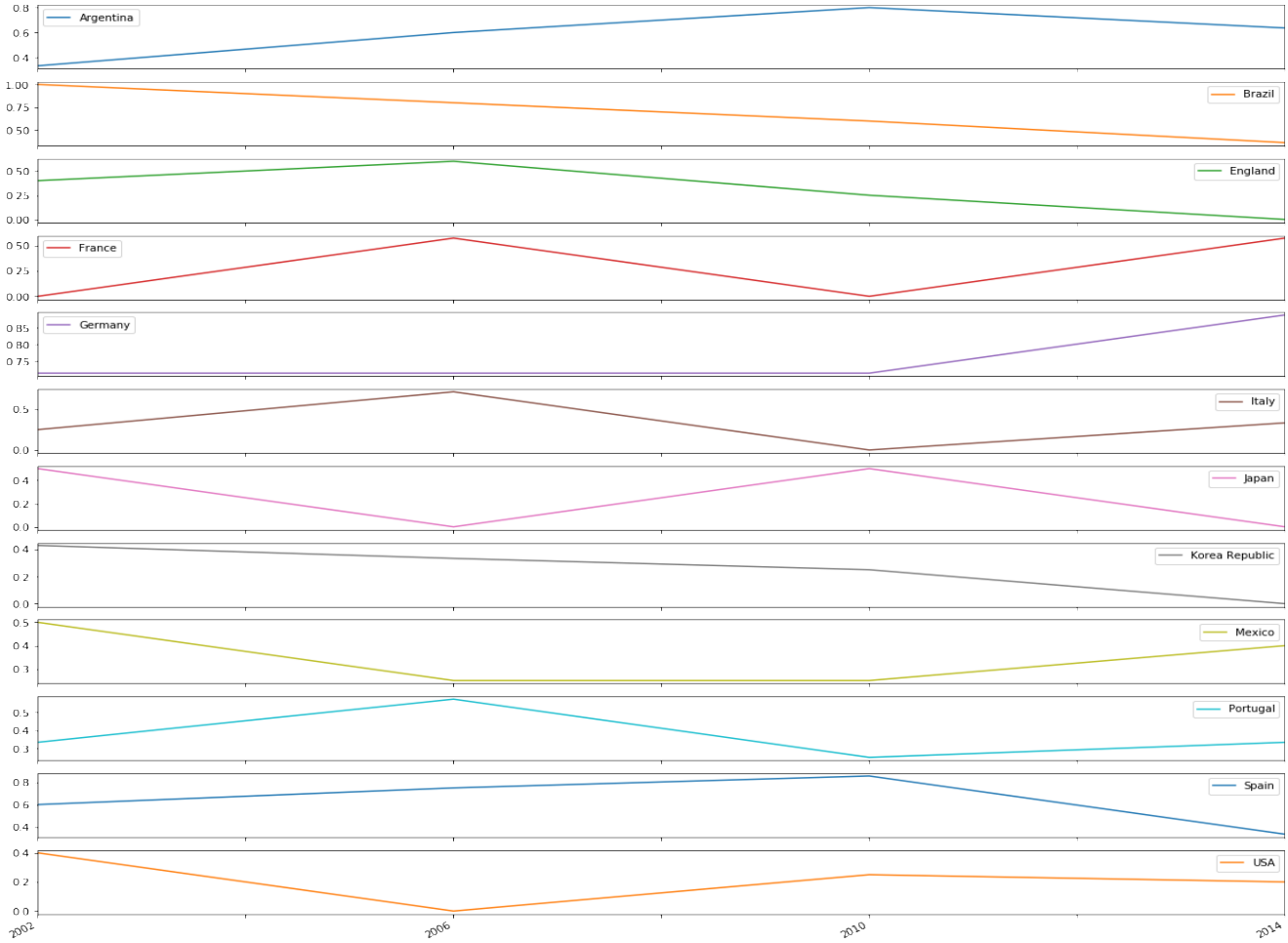
	Korea Republic	Mexico	Portugal	Spain	USA
2002	0.428571	0.50	0.333333	0.600000	0.40
2006	0.333333	0.25	0.571429	0.750000	0.00
2010	0.250000	0.25	0.250000	0.857143	0.25
2014	0.000000	0.40	0.333333	0.333333	0.20

From this data alone, I was able to gather valuable information for various teams. For example, it appeared that Brazil's winning rate had been consistently decreasing and thus showed a strong negative trend. Germany on the other hand was very consistent in their success, even getting slightly better in the 2014 world

cup. Korea Republic, much like Brazil, has been getting consistently worse in their winning success rate, although at a slower rate. The overall winning percentage of each team would later prove to be very useful in determining how likely it is that the team will win against another team in a future world cup. It is important to note however that each team didn't necessarily play the same number of matches every year, and certainly different teams played different number of matches in comparison to one another.

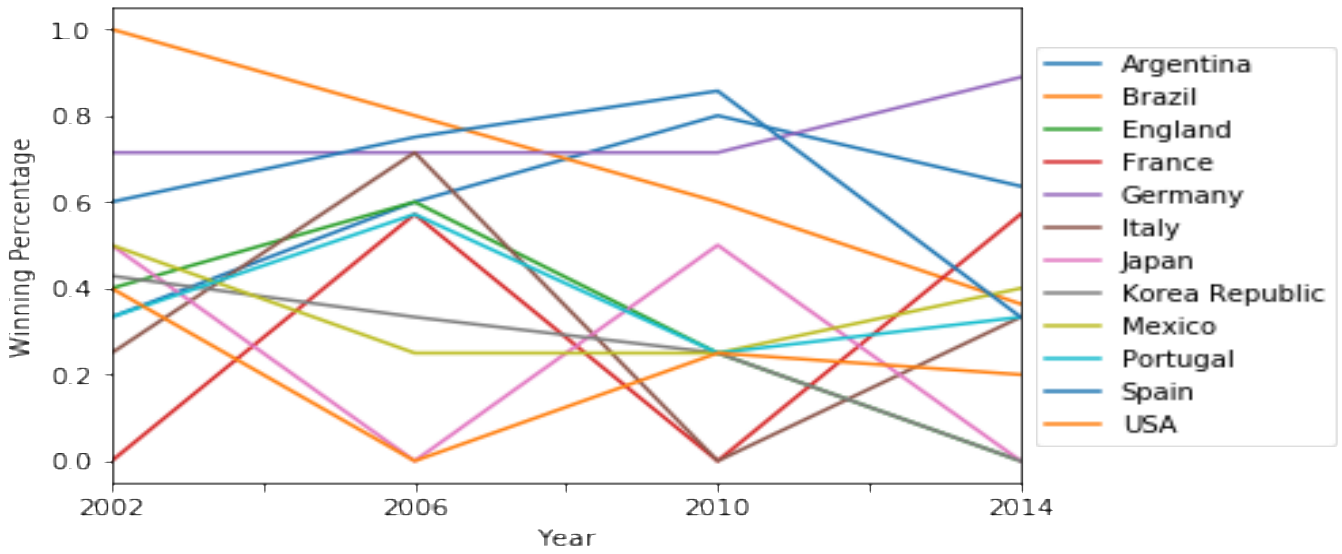
In order to get a visual representation of the winning percentages by year for each team, I plotted it as such. I created a separate subplot for each team with year on the x axis and winning percentage on the y axis. This is provided a nice visual of how each team fared each year in terms of their winning rates.

Winning Percentage of each team in subplots:



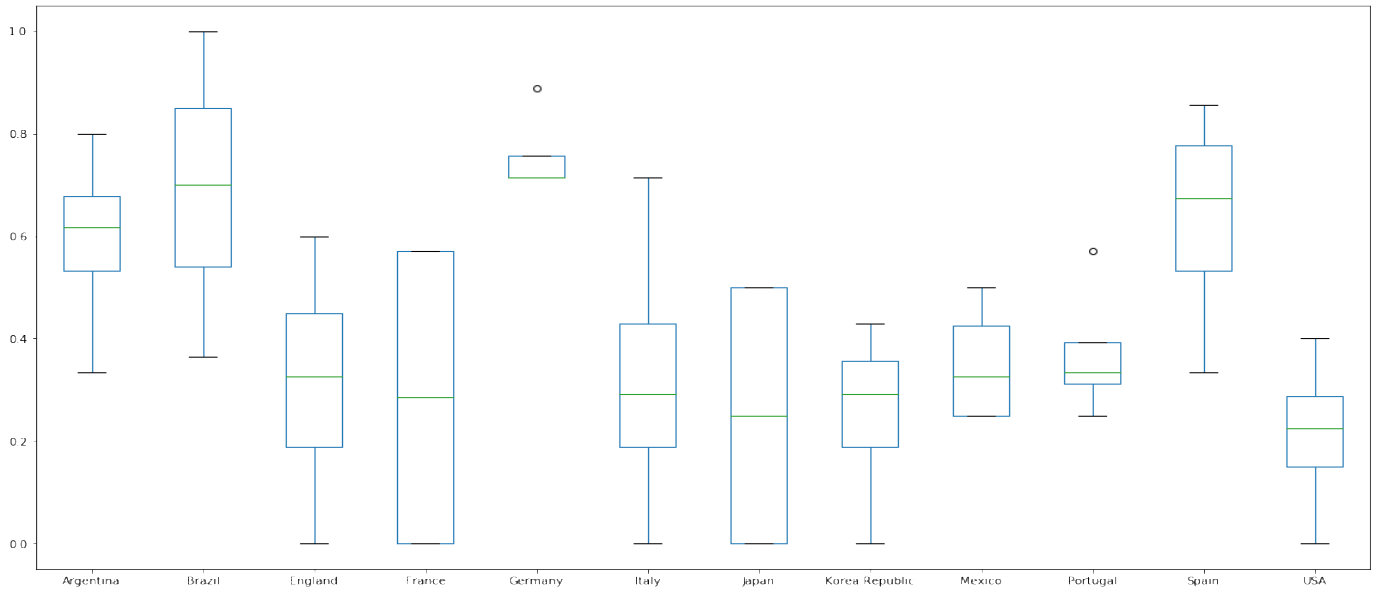
I also wanted to see how the winning percentages all the teams fluctuated over the years on a single plot and compare them to one another.

Winning Percentage of each team in a single plot:



It was important to examine the variance as well. I wanted to see how the winning percentages varied over the years by team. A box plot was the best option to easily visualize this.

Box plot showcasing winning percentage variance:



As we can see, teams such as Italy had tremendous variance while others such as Mexico did not have so much. In general, most teams showcased high variance in the winning percentages through all the world cups. Germany's

variance however was very minimal showing a high level of consistently in their success in each world cup.

Feature Engineering:

The samples/rows/observations for my data frame would be the matches between two teams. The initial problem was the CSV file unfortunately labeled the teams in each match as home and away. The same team could be the home team in some matches and the away team in other matches. This is a problem because in actuality it does not make sense since all the matches for a world cup happen in the same country. Thus, for each world cup, all the teams would be away except for the host country, essentially making the home and away fields not useful. Ultimately, the matches between two teams needed to be matches from the four previous world cups (2002, 2006, 2010, 2014) regardless of which team is labeled as home or away.

To combat this problem and remove the home/away factor, I first assigned a unique ID to each team in the 'home team name' and 'away team name' columns. Each team was now assigned a unique number, regardless of whether that team was labeled as the home team or away team in any particular match. Now I needed to ensure that when two teams played each other, the IDs were displayed in the same order in the data frame the same way each time.

To accomplish this, I set it up so that anytime the ID of the home team is greater than the ID of the away team, it would swap the columns of those respective teams. This ensured it stayed consistent throughout the dataset. This would allow all combinations to be displayed the same way; the team with the smaller ID will always be displayed first and the team with the larger ID will always be displayed second in the data frame. To illustrate, in the initial data frame it could show a match between Ids 17 and 37 (Brazil as home vs Chile as

away) for one row and then 37 and 17 in another row (Chile as home vs Brazil as away). I altered it so that every time these two teams played it would always show up the same way, which in this case would always be 17 and 37 since 17 is the smaller ID and 37 is the larger ID. This allowed for easier calculation of features in any match combination because it takes the away the home/away matchup factor which wasn't useful. Teams with the smaller ID (on the left) were labeled as Team0 and teams with the higher Id (on the right) were labeled as Team1. I created a sorted multi-level index with Team0 being the first index level and Team1 being the second index level.

Afterwards, I created the target variable/column called the 'winning team.' This variable was labeled either zero, one, or two for each match/sample. A zero signified that the winning team was team0 (team on the left), a one signified the winning team in the match was team1 (team on the right), and a two signified that the teams had tied for that particular match. The criteria for whether the team won, lost, or tied was based on the number of goals each team scored. If the 'team 0 goals' column was greater than the 'team 1 goals' column then the correct classification for that sample would be 0 and 1 if vice versa. The correct classification would be 2 if the number of goals in both 'team 0 goals' and 'team 1 goals' variables were equal.

Class labels for target variable:

0 = Team 0 wins.

1 = Team 1 wins.

2 = Tie.

Now that the multi-level index (the samples) and target variables were created, I created the features that I would use to predict the target variable. In total, I created eleven features for each sample/match. Each match would have a

value for each one of these features. I engineered the features that I believed would be best suited in classifying the winning team in other words the target variable. The features were:

-The overall respective win percentages for team0 and team1 in all their respective matches from all world cups 2002 onwards.

-The respective win percentages of team0 and team1 of all the matches in which they played against one another.

-Their tie percentage in all of their matches played against one another.

-The overall tie percentages for team0 and team1 in all their respective matches from all world cups 2002 onwards.

-Employed the recency effect, which indicates that recent events have more impact or more weight on future events than older events. Thus, I generated the win percentage for team0 in all their matches played in the last two world cups and did the same for team1. For those teams that didn't play in the last two world cups, I reused their win percentage overall for all world cups.

-Continuing the recency effect, I generated the tie percentage in all matches played in the last 2 world cups for team 0 as well as team 1.

Data frame with Features (head):

		Team0_Win%_Overall	Team1_Win%_Overall	\
Team0	Team1			
Algeria	Belgium	0.166667	0.714286	
	England	0.166667	0.142857	
	Korea Republic	0.166667	0.142857	
	Russia	0.166667	0.000000	
	Slovenia	0.166667	0.333333	

		Team0_Win%_InMatchUp	Team1_Win%_InMatchUp	\
Team0	Team1			
Algeria	Belgium	0.0	1.0	
	England	0.0	0.0	
	Korea Republic	1.0	0.0	

```
        Russia                          0.0                   0.0
        Slovenia                        0.0                   1.0

                      BothTeams_Tie%_InMatchUp  Team0_Tie%_Overall
\
Team0   Team1
Algeria Belgium                            0.0            0.333333
        England                            1.0            0.333333
        Korea Republic                     0.0            0.333333
        Russia                             1.0            0.333333
        Slovenia                           0.0            0.333333

                      Team1_Tie%_Overall  Team0_Win%_Last2WC  \
Team0   Team1
Algeria Belgium                 0.000000            0.166667
        England                 0.428571            0.166667
        Korea Republic          0.285714            0.166667
        Russia                  0.666667            0.166667
        Slovenia                0.333333            0.166667

                      Team1_Win%_Last2WC  Team0_Tie%_Last2WC  \
Team0   Team1
Algeria Belgium                 0.714286            0.333333
        England                 0.142857            0.333333
        Korea Republic          0.142857            0.333333
        Russia                  0.000000            0.333333
        Slovenia                0.333333            0.333333

                      Team1_Tie%_Last2WC
Team0   Team1
Algeria Belgium                 0.000000
        England                 0.428571
        Korea Republic          0.285714
        Russia                  0.666667
        Slovenia                0.333333
```

Machine Learning:

The main strategy I used to tackle this problem was to use multiple machine learning algorithms and ensemble the results together to get the best results. The first algorithm I used was Extreme Gradient Boosting or XGBoost. This model has proven to be incredibly successful with machine learning problems. The second algorithm I used was the Random Forest Classifier and the third algorithm was the soft-max/multinomial logistic regression. I then

assigned a certain weight to the results of each model, and ensembled them together to determine if I could get an accuracy that was better than each model had individually.

Initially, I created the features(x) and target(y) datasets. The features set was all the features without the 'winning team' target variable. The target set was just the target variable which was the 'winning team' column. I then created the train and test sets. I performed stratified sampling so the classes percentages for all three classes were the same for both the train and test sets. This removed any class imbalance between the train and test sets. This is critical to avoid skewing the results of the machine learning model. I used 20 percent of the data as the validation set with 80 percent as the training set. The next step was to fit my training data onto my first machine learning algorithm which was the XGBoost Classifier. I experimented with the various hyperparameters to ensure I could obtain the best results for the XGBoost Classifier. The hyperparameters I was experimenting with max depth, learning rate, and the number of estimators.

For each hyperparameter, I generated the cross-validation score using accuracy as a metric. I performed five-fold cross validation on the XGBoost model and took the average/mean of the resulting five cross validation scores. I then fit the training data on the model again separately and obtained a classification report, comparing the y test set with the y predictions. I experimented with many different values for each of the hyperparameters. The values that gave me the highest results in the classification report as well as the mean cross validation score were a max depth of 3, number of estimators of 250, and the default learning rate of 0.1. The mean five-fold cross validation score for these hyperparameter values was 94 percent. Afterwards, I obtained the predicted probabilities of each match/sample belonging to particular class (0,1, or 2) and put them into a Pandas data frame with one column for each class label.

XGBoost Classifier Classification Report:

	precision	recall	f1-score	support
0	1.00	0.90	0.95	20
1	1.00	1.00	1.00	20
2	0.86	1.00	0.92	12
micro avg	0.96	0.96	0.96	52
macro avg	0.95	0.97	0.96	52
weighted avg	0.97	0.96	0.96	52

I then proceeded to my second algorithm, the Random Forest Classifier. I repeated the same steps as that I did for XGBoost with the hyperparameters max depth and number of estimators. The values that gave me the best scores on the classification report and mean five-fold cross-validation score were 3 for max depth and 250 for the number of estimators. The mean five-fold cross-validation score with these hyperparameter values was 94 percent. I once again obtained the predicted probabilities of each match/sample belonging to particular class (0,1, or 2) and put them into a Pandas data frame with one column for each class label.

Random Forest Classifier Classification Report:

	precision	recall	f1-score	support
0	1.00	0.90	0.95	20
1	1.00	1.00	1.00	20

2	0.86	1.00	0.92	12
micro avg	0.96	0.96	0.96	52
macro avg	0.95	0.97	0.96	52
weighted avg	0.97	0.96	0.96	52

The last algorithm I used was soft-max/multinomial logistic regression. Soft-max logistic regression was necessary due to the fact that the number of classes exceeded two and thus it was not a binary classification problem. I set the optimization algorithm for the logistic regression to stochastic gradient descent to best minimize the loss function. I experimented with different values for the hyperparameter "C", which is the inverse of regularization strength. I proceeded to obtain the classification report comparing y test with y predictions, the predicted probabilities of each class, and the mean five-fold cross-validation score. The default hyperparameter value for C of 1.0 gave me the best results. The mean five-fold cross-validation score was 95 percent.

Soft-max Logistic Regression Classification Report:

	precision	recall	f1-score	support
0	1.00	0.95	0.97	20
1	1.00	1.00	1.00	20
2	0.92	1.00	0.96	12
micro avg	0.98	0.98	0.98	52
macro avg	0.97	0.98	0.98	52
weighted avg	0.98	0.98	0.98	52

After performing these three machine learning algorithms, I examined the results of each model and experimented applying different weights to each of them. I then ensembled the results using the different weight combinations to see if I can get an even better accuracy. To do this, I multiplied the weights to the predicted probabilities for each class (0,1,2) generated by each of the machine learning models on the test set.

After this, I put the ensemble results (predicted probabilities of the models with the assigned weights) into a Pandas data frame that assigned a column for the predicted probabilities of each class label. Thus, there were 3 columns, one with the predicted probabilities of class 0, one for the predicted probabilities of class 1, and one for the predicted probabilities of class 2.

The class label with the highest probability is essentially the prediction of the model for that particular record/sample. Thus, I took the class label that had the highest prediction probability for each record in the ensembled results. I then compared these predictions to the y test set ground truth and obtained a classification report. I repeated these steps with multiple different weight combinations for each of the three models. However, the highest accuracy I could achieve was 96 percent with a .7 weight on logistic regression and .15 weight on both the XGBoost Classifier and Random Forest Classifier. Other weight combinations produced a 96 percent accuracy as well however that was the highest accuracy I could achieve. The weighted ensemble model essentially ended up duplicating the results that the XGBoost Classifier and Random Forest Classifier gave individually. This was due to the lack of samples/matches to work with overall. This was also less than the 98 percent accuracy I obtained using soft-max logistic regression by itself. Thus, using soft-max/multinomial logistic regression alone proved to be a best option rather than using any combination of weights for each of the models.

Weighted Ensemble Model Classification Report:

	precision	recall	f1-score	support
0	1.00	0.90	0.95	20
1	1.00	1.00	1.00	20
2	0.86	1.00	0.92	12
micro avg	0.96	0.96	0.96	52
macro avg	0.95	0.97	0.96	52
weighted avg	0.97	0.96	0.96	52