World Cup Prediction

*home pod: Sharanya Venkat happy-herring

Sen Luo
Electrical and Systems Engineering
University of Pennsylvania
luosen@seas.upenn.edu

Jianyu Chen
Electrical and Systems Engineering
University of Pennsylvania
cjyroger@seas.upenn.edu

Haoyi Cheng
Electrical and Systems Engineering
University of Pennsylvania
haoyic@seas.upenn.edu

***Abstract*—In this project, we used machine learning algorithms to predict the outcome of soccer games and apply it to predict the champion of the 2022 FIFA world cup. Compared with traditional soccer prediction, we apply video game FIFA data into the dataset and assign the importance levels to each game. We use decision tree as our baseline method and apply Support Vector Machine, Logistic Regression, Random Forest, Gradient Tree Boosting and Neural Network on this topic. We found the Gradient Tree Boosting model gives the best result and this model predict Argentina will be the world champion of the 2022 Qatar FIFA World Cup.**

I. Motivation

The opening of the FIFA 2022 Qatar World Cup is in less than a week, which means the wait is almost over. It's been more than four years since France win the FIFA 2018 Russia World Cup. The past four years have been rough for all mankind, with the covid-19 pandemic hitting the world. We really need this world cup to bring people all over the world together. As die-hard football fans, all of our team members can't be more excited for the World Cup to take place. Someone in our team supports Argentina, while the other supports Portugal. However, instead of arguing which country could win the World Cup based on our own preferences, can we use data from past football games to predict this year's result? Having studied machine learning for almost the whole semester, can we use some data from the past to train a model? How can we evaluate our model's performance? Is it possible to optimize the loss function? We grow great interest in applying what we learnt so far to predict the up-coming World Cup. Each game contains lots of data and game happens every day in the world. Thus, we have enough training data to train a machine learning model to predict the outcome of the game.

II. Related Work

Dr. James Bond [1] use data from Kaggle to model the outcome of certain pairings between teams, given their rank, points, and the weighted point difference with the opponent. He uses a simple Logistic regression model to predict the outcome. Although he make a surprising correct predict that Belgium will top Brazil (which seems impossible for many people), we believe he do not take important factors in soccer game into consideration.

Lang [2] used Logistic Regression, Decision Trees, Random Forests, and AdaBoost to predict in-game status of soccer game. Though it is not completely the same as the topic of our project, it gave us a intuition of how to improve our model for further prediction.

Sergio Pessoa [3] remove the advantage of away team, predicting the results changing teams from away and home (because there is not home advantage in World Cup), and used as probabilities the mean of the two predictions. He used FIFA ranking from last 20 years and games from last 150 years. But the fact is soccer develop fast in last 30 years, thus we believe games older than 30 years should not be considered.

Octavio Santiago [4] used Generalized Linear Model (GLM) to predict the game, the model reached good result and he considered current conditions of the game and a history of each player of the teams combined in the attack, defense, and Technician. However, the cons of the model is that he trained it on a relatively small dataset.

III. Data Set

The $Dataset_1$ (https://www.kaggle.com/datasets/brenda89/fifa-world-cup-2022) provides a complete overview of all international soccer matches played since the 90s. It contains 23922 matches and 25 useful features. Our project requires 7 features in this database : "date", "home_team", "away_team", "tournament", "home_team_fifa_rank", "away_team_fifa_rank", "home_team_result". The dataset does not have any missing values.

The $Dataset_2$ (https://www.fifaindex.com/) contains attack score, defense score, midfield score, and overall rating score of each national team from 2005 to 2022. We need to use the date, home_team and away_team in $Dataset_1$ to track the rating of each team in a certain period of time. The team rating scores before 2005 are not recorded. So we use the team rating scores from 2005 to 2022 to calculate the average value and treat it as the team rating scores before 2005. Besides, some of the national team of the $Dataset_1$ do not have corresponding ratings in $Dataset_2$. In this case, we assign 65 to their score because these teams are usually not strong. (Stronger team could have the ratings of over 80). Meanwhile, we assign the importance of the game based our soccer knowledge, (e.g., World Cup game has the importance of 1 and Friendly game has the importance of 4).

After merging, we got our raw data with each column respectively representing ”date”, ”home_team”, ”away_team”, ”tournament”, ”home_team_fifa_rank”, ”away_team_fifa_rank”, ”home_team_result”, ”home_team_-mean_defense_score”, ”home_team_mean_midfield_score”, ”home_team_mean_offense_score”, ”away_team_mean_-defense_score”, ”away_team_mean_midfield_score”, ”away_team_mean_offense_score” and ”importance”.

In this problem, we only need the fifa rank to represent each team. Therefore, we can delete the column of ”home_team” and ”away_team”. Meanwhile, ”tournament” can be deleted since ”importance” has already replace the role of it.

Finally, we have 9 columns representing the features, which are ”home_team_fifa_rank”, ”away_team_fifa_rank”, ”home_team_mean_defense_score”, ”home_team_mean_-offense_score”, ”home_team_mean_midfield_score”, ”away_team_mean_defense_score”, ”away_team_mean_-offense_score”, ”away_team_mean_midfield_score” and ”importance”. We have 1 column to present the label, which is ”home_team_result”.(0 implies home_team’s loss, 1 implies home_team’s draw and 3 implies home_team’s win).

In conclusion, we have 23922 samples each with 9 features and 1 label. Here is some summary statistics and visualizations of the data.

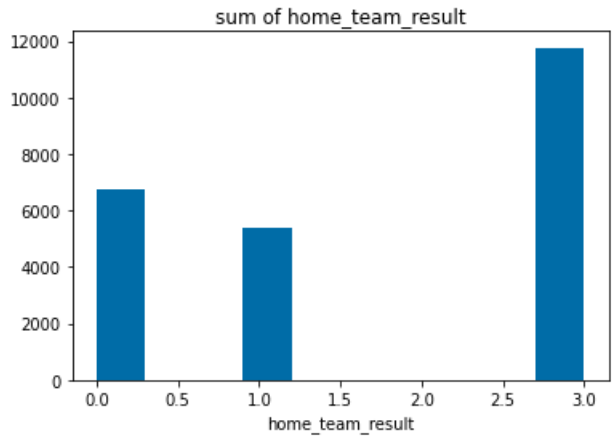

Fig. 1. Visualization of home_team_result

From figure 1, we can see that home_team have almost half of the winning.

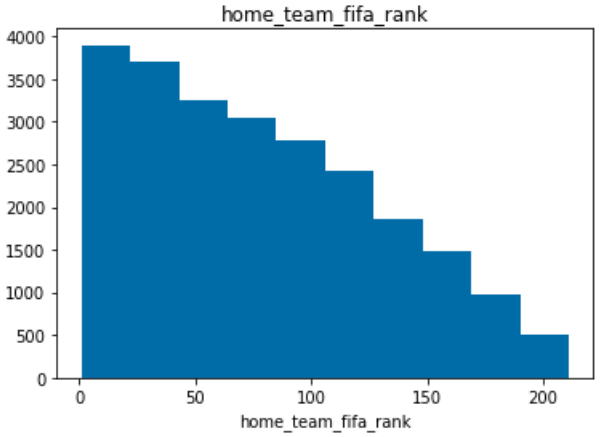

Fig. 2. Visualization of home_team fifa rank

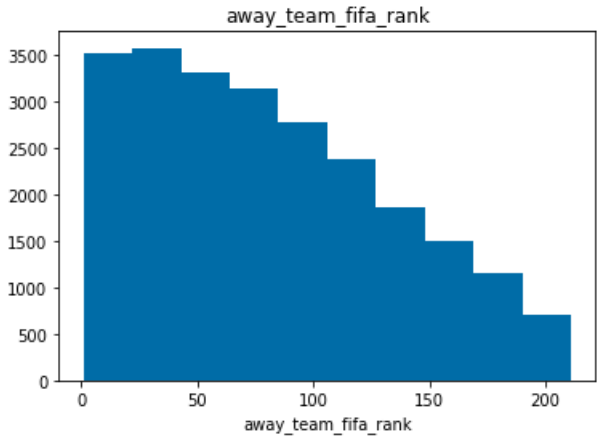

Fig. 3. Visualization of away_team fifa rank

Figure 2 and figure 3 shows that the home_team_rank are relatively higher(smaller) than away_team_rank. This can probably account for why home_team’s win is more than loss.

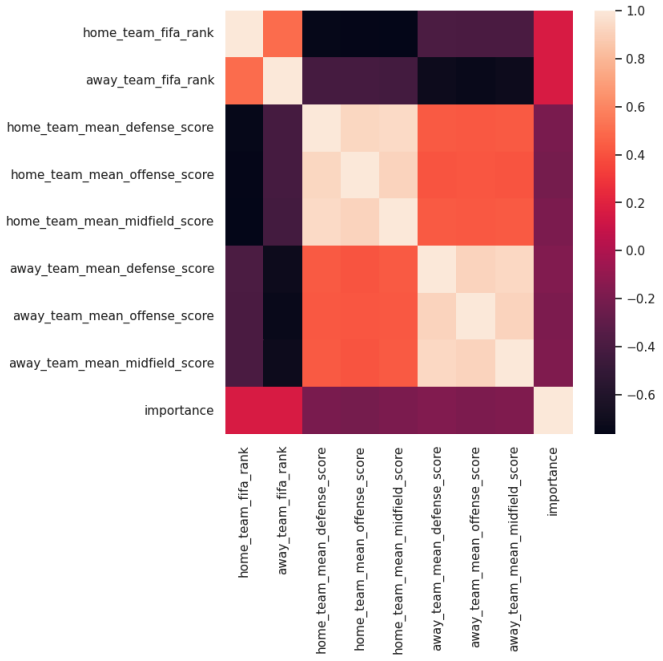

Fig. 4. Correlation heatmap of "X"

Figure 4 shows the Correlation heatmap of "X", it gives us some basic intuition of the data(e.g., if the rank is higher(smaller), the score should be bigger, thus those two features should have negative correlation).

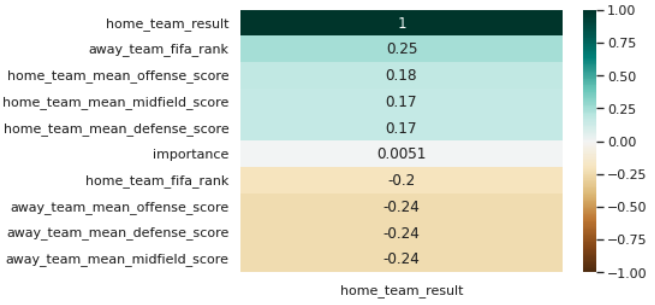

Fig. 5. Correlation heatmap between "X" and "Y"

Figure 5 shows the Correlation heatmap of "X" and "Y", it gives us some basic intuition of the relationship between data and label(e.g., if the home_rank is higher(smaller), home_team_result should be bigger(win)).

IV. Problem Formulation

First, we collect data of all football game from 1990-2022 from Kaggle. As for input features, we use world rank of both home and away team. We also use video game "FIFA"'s national team ratings for both home and away team as input features. Besides, we manually import the feature importance of the game into the data. Thus, the input features will be home-team national team ranking, home-team defense rating, home-team midfield rating, home-team offense rating, away-team national team ranking, away-team defense rating, away-team midfield rating, away-team offense rating, importance of the game and the output will be the result of the home-team(win, draw or loss). Due to the fact that World Cup is held in neutral place, we ignore the impact of home advantage. We will use the result of the games except World Cup games as the training set and use World Cup games as testing set. There will be more than 20000 samples in training set and this prediction is a supervised learning (classification) problem. Thus, we can use machine learning methods to solve this problem. As for model representation, we will train some models using machine learning methods such as Decision Tree, Logistic Regression, Random Forest, Gradient Tree Boosting, SVM and Neural Network. We will use accuracy score, 5-fold cross validation score, F1 score and confusion matrix to evaluate our models.

V. Methods

A. Baseline: Decision Tree

Decision tree is a non-parametric supervised learning algorithm, it is a very useful tool for classification and prediction problems. We choose Decision Tree as our baseline method because Decision Tree method has the lowest test accuracy and cross validation score, despite having the highest train accuracy of 98.34%, which means it has the worst performance for our problem. The reason why decision tree doesn't perform well is that with a fairly large amount of training data, decision tree is easily overfitting so as to have a lower bias and higher variance comparing to other methods.

Package: from **sklearn.tree** import **DecisionTreeClassifier**

B. Logistic Regression

Logistic Regression is a parametric supervised learning algorithm and can be used for classification problems. Since our problem has "win", "draw" and "lose" three categories, we can use multinomial logistic regression. The intuition for logistic regression is to learn a linear relationship from the training dataset and then maps it to a probability distribution using the softmax function. Besides, Logistic regression is very easy to implement, interpret, and very efficient to train.

Package: from **sklearn.linear_model** import **LogisticRegression**

C. Random Forest

Random Forest is a parametric supervised learning algorithm, which is suitable for classification problems. Random forests are composed of decision trees, however, compares to decision tree, random forest uses both ensemble and bagging method, which not only decreases the error, but also avoids overfitting. As a result, Random forest is a better method than our baseline method and will have a better performance on predicting test data.

Package: from **sklearn.ensemble** import **RandomForestClassifier**

D. Gradient Tree Boosting

Gradient Tree Boosting is a parametric supervised learning algorithm and is currently a state-of-the-art method for solving classification problems. It's on average slightly better than random forest method. Gradient Tree Boosting is also an ensemble of trees, the difference between random forest and gradient tree boosting is that gradient tree boosting do stagewise estimation of the model and builds on weak learners.

Package: from **sklearn.ensemble** import **GradientBoostingClassifier**

E. SVM

Support vector machine(SVM) is a parametric supervised learning algorithm, which is a great tool to solve classification problems. The objective of SVM is to find a hyperplane in our 9-dimensional space that distinctly classifies the data points. SVM works relatively well in our problem because there is a clear margin of separation between our three classes. However, SVM is going to be very slow due to our relatively large data sets.

Package: from **sklearn.svm** import **SVC**

F. Neural Network

Neural network is an artificial neural network, composed of artificial neurons or nodes. Neural Nets can be supervised, unsupervised or reinforcement in machine learning. In our problem, we import the **MLPClassifier**, which trains iteratively since at each time step the partial derivatives of the loss function with respect to the model parameters are computed to update the parameters. Our neural network method works well with large input dataset. We can also construct a deeper learning model to a higher degree to improve the performance of our classification problem by increasing the hidden layers.

Package: from **sklearn.neural_network** import **MLPClassifier**

VI. Experiments and Results

A. Random Forest

For Random Forest, first we trained the basic model using RandomForestClassifier package. Then we wanted to find the best parameters from the following hyperparameters:

```
params = {"max_depth": [20,30],
          "min_samples_split": [10],
          "max_leaf_nodes": [175],
          "min_samples_leaf": [5],
          "n_estimators": [300],
          "max_features": ["sqrt"],
          }
```

We tuned the object hyperparameters by doing grid search from sklearn package. We found that the best-performing model has an accuracy of 61.71% on training data, and a 58.50% accuracy on test data. And the best-performing hyperparameters for Random Forest are:

```
params = {"max_depth": [30],
          "min_samples_split": [10,20],
          "max_leaf_nodes": [175,200],
          "min_samples_leaf": [5,10],
          "n_estimators": [250,300],
          "max_features": ["sqrt"],
          }
```

To further analyse our model performance, we use 5-fold cross validation error, confusion matrix and F1 score to evaluate the model. We also import **KFold** and **cross_validate** from sklearn to compute the mean 5-fold cross validation score, which is 58.33%. The confusion matrix and F1 score are as follows:

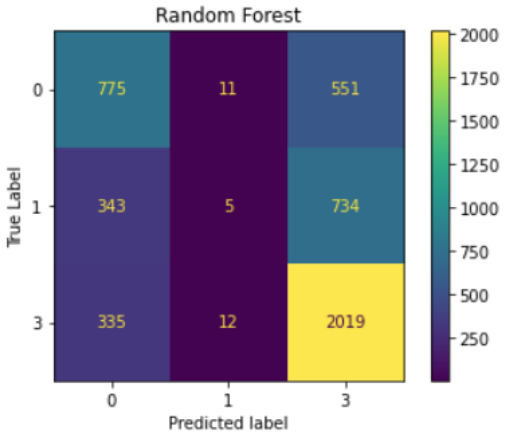

Fig. 6. Random Forest Confusion Matrix

TABLE I
Gradient Tree Boosting F-1 Score with different labels.

	Lose	Draw	Win
F1-Score	0.5556	0.0090	0.7122

From the confusion matrix and F1-score, we can observe that Random Forest model has the lowest F1-score on predicting both losing and winning cases, while has a relatively better prediction on draws than SVM and Neural Network.

B. Gradient Tree Boosting

For Gradient Tree Boosting, we tried to tune the best-performing model using the following hyperparameters:

```
params = {"learning_rate": [0.1,0.3,0.5],
          "min_samples_split": [5,10],
          "min_samples_leaf": [5],
          "max_depth":[3],
          "max_features":["sqrt"],
          "n_estimators":[100]
          }
```

We used cross-validation with k equal to 5 to tune the model and find the best performance. From sklearn package in python, we imported GridSearchCV to perform the cross-validation test. We found the best-performing model has an

accuracy of 59.58% on training data, and a 58.81% accuracy on test data. And the best-performing hyperparameters for gradient tree Boosting are:

```
params = {"learning_rate": [0.1]
          "min_samples_split": [5],
          "min_samples_leaf": [5],
          "max_depth":[3],
          "max_features":["sqrt"],
          "n_estimators":[100]
          }
```

The mean 5-fold cross validation score is 58.35%. To analyze the model performance, we further utilize the confusion matrix and F1 score to evaluate the model.

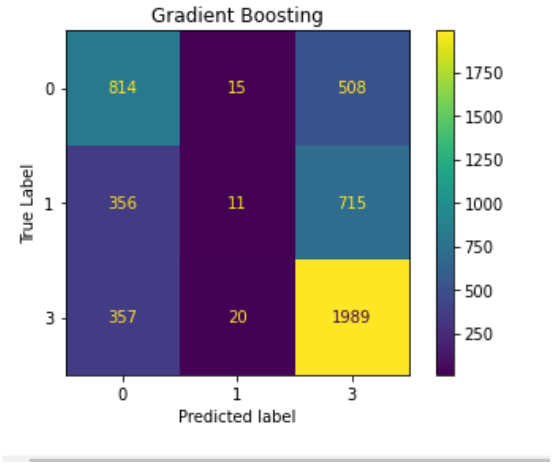

Fig. 7. Gradient Tree Boosting Confusion Matrix

TABLE II
GRADIENT TREE BOOSTING F-1 SCORE WITH DIFFERENT LABELS.

	Lose	Draw	Win
F1-Score	0.5684	0.0195	0.7134

From the confusion matrix and F1-score, it is notable that the Gradient Tree Boosting model has a higher F1-score and accuracy compared to other models. It performs better on predicting wining case and did relatively poorly on predicting draw.

C. SVM

SVM algorithm is trying to find a hyperplane in a high dimensional space that could distinctly classify the data points. We train the SVM model by trying different kernel and regularization parameter C. The hyperparameters we are interested in are:

```
params = {"kernel": ['linear', 'poly',
          'rbf', 'sigmoid'],
          "C": [0.01, 0.1, 1, 10, 50]
          }
```

We used two for loops to go over all the combination of parameters, trained the model, and perform the accuracy test with each combination. And the best-performing hyperparameters are linear kernel and with 0.1 as regularization parameter. The best-performing parameters give an accuracy on training data of 58.05%, and 59.18% accuracy on test data.

```
params = {"kernel": ['linear'],
          "C": [0.1]
          }
```

The mean 5-fold cross validation score for SVM is 58.20%. We then run the confusion matrix and F1 score to evaluate the model.

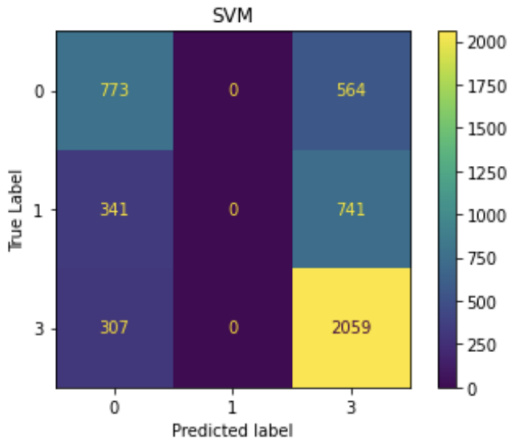

Fig. 8. SVM Confusion Matrix

TABLE III
SVM F-1 SCORE WITH DIFFERENT LABELS.

	Lose	Draw	Win
F1-Score	0.5606	0.	0.7187

From confusion matrix, the SVM model ignores all the draw cases and classified draw as win or lost. In this case, we can take the classification as binary and calculate the precision and recall value. Taking win (labeled as 3) as positive label, we can calculate the precision and recall value as follow:

$$Precision = \frac{TP}{TP + FP} = \frac{2059}{2059 + 741 + 564} = 61.21\% \quad (1)$$

$$Recall = \frac{TP}{TP + FN} = \frac{2059}{2059 + 0 + 307} = 87.02\% \quad (2)$$

D. Neural Network

The last model we tried is neural network. We first train the model with For Gradient Tree Boosting, we tried to tune the best-performing model using the following hyperparameters:

```
params = {"alpha": [0.001, 0.01,
          0.1, 1],
```

```
"hid": [[50, 50, 50],
[100, 100]],
"solver": ['adam', 'sgd'],
"activation":['logistic',
'relu'],
"max_iter":[300]
}
```

We again used nested for loops to go over all the combination of parameters. By comparing the accuracy on test data, we found the the best-performing hyperparameters for neural network are as follow. The accuracy on the training data using the best performing hyperparameters is 58.12%, and it has a 59.37% accuracy on test data.

```
params = {"alpha": [1],
        "hid": [[100, 100]],
        "solver": ['adam'],
        "activation":['relu'],
        "max_iter":[300]
        }
```

The mean 5-fold cross validation score for Neural Network is 58.24%. We also perform the confusion matrix and F1 score for the neural network model. It is worth noticing that the neural network model also performs poorly on correctly labeling the draw data points. The model treats all draw cases as either win or loss. But it performs a good prediction on win and lost data points compared to other models according to F1 score and the accuracy on test data.

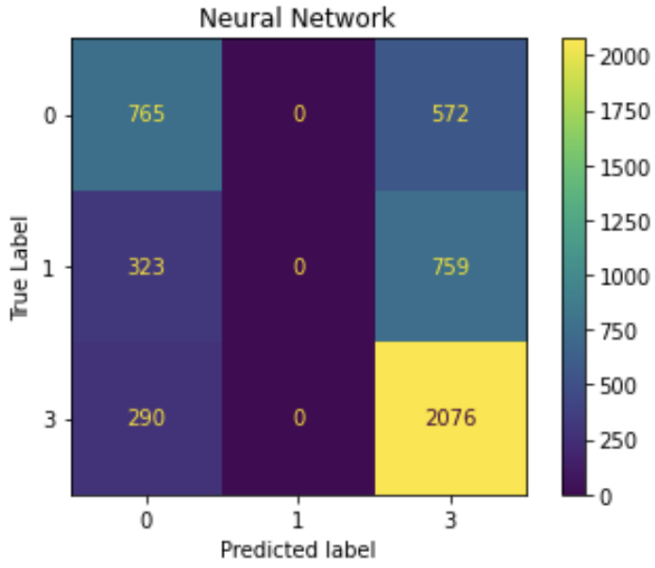

Fig. 9. Neural Network Confusion Matrix

TABLE IV
NEURAL NETWORK F-1 SCORE WITH DIFFERENT LABELS.

	Lose	Draw	Win
F1-Score	0.5635	0.	0.7192

TABLE V
A SUMMARY OF PERFORMANCE ACROSS ALL MODELS.

	Train accuracy	Test accuracy	CV score
Decision Tree	98.34%	46.79%	46.11%
Logistic Regression	58.01%	58.81%	58.15%
Random Forest	61.71%	58.50%	58.33%
Gradient Tree Boosting	59.58%	58.81%	58.35%
SVM	58.05%	59.18%	58.20%
Neural Network	58.12%	59.37%	58.24%

E. World Cup prediction

Finally, we use Gradient Tree Boosting to predict 2022 FIFA World cup, our result indicates Brazil, England, Argentina and Germany will be the final 4 and Brazil will beat Germany in the final. To diminish the effect of home-away team, we predict the results changing teams from away and home (there is not home advantage in World Cup).

	home	away	home_win	away_win	winner
0	Brazil	Argentina	0.522298	0.477702	Brazil
1	Germany	England	0.509951	0.490049	Germany

Fig. 10. Prediction of predicted semi-final game

	home	away	home_win	away_win	winner
0	Brazil	Germany	0.535825	0.464175	Brazil

Fig. 11. Prediction of predicted final game

However, this world cup have happened a lot of "underdog miracle". Morocco and Croatia magically make into final 4. Actual semi-final game will be held in Dec 13, we use our model to predict the semi-final and final game, and model suggests that Argentina will be the 2022 World Cup champion.

	home	away	home_win	away_win	winner
0	Croatia	Argentina	0.367729	0.632271	Argentina
1	Morocco	France	0.286490	0.713510	France

Fig. 12. Prediction of actual semi-final game

	home	away	home_win	away_win	winner
0	Argentina	France	0.540341	0.459659	Argentina

Fig. 13. Prediction of actual final game

VII. CONCLUSION AND DISCUSSION

A. Model Comparison

After comparing the F1 score and accuracy on the test dataset, we conclude that the neural network model yields

the best performance on test data among the other 5 models we have tried. The best performing neural network model has [100, 100] as hidden layers, and takes ADAM as solver and RELU as activation. It has an F1 score being (0.5635, 0., 0.7192) and a test accuracy of 59.37%, which are the highest accuracy. In terms of training models using cross-validation, gradient tree boosting stands out with the highest 5-fold cross-validation accuracy of 58.35%.

B. Model Interpretation

It is notable and worth mentioning that all 6 model performs poorly on correctly classifying the situation when there is a draw. Some of our models, like gradient tree boosting distributed a very small portion of the data points to draw events but has very low accuracy. Other models including neural network and SVM completely ignore all draw cases and wrongly classify them as a win or lose. One possible explanation for this issue can be that having draws in reality often happens randomly. The underlying logic for our models is probability. When things are happening as good as random, it is hard for the model to predict. draw event has no relationship or any causal effect on other features. It is hard for the models to conclude any patterns. Taking the World Cup as an example, draw events happen randomly. A team with a higher overall score could lose a game to a relatively weak team. And there is no significant amount of data in our dataset where two teams with relatively the same ability have a draw. So the patterns for draw events are difficult to find for all classification models.

C. Lesson Learned

Our effort focuses on improving the prediction accuracy of the models. The accuracy we yield at first is 47.26% from the baseline method, decision tree. To improve the accuracy, we have tried different models. Starting from Logistic regression, we are able to gain an accuracy of 58.81%. However, we aim to find a model with at least 65% accuracy. In order to further upgrade the models, we performed k-fold cross-validation and hyperparameter tuning on each model. This attempt slightly increased the best accuracy to 59.37%, but we still failed to break the 60% threshold. After thoroughly investigating, we realized that the quality of the dataset is one of the most crucial factors that affect the model performance. The total amount of data points and the number of related features in the dataset are as important as model choosing and model tuning. Our dataset only has 10 relative features. And the dataset is unevenly distributed among categories. The amount for winning events is twice more than the loss and draw events. If we could modify our dataset and gather more relative features, we might have a better chance to improve our model further.

D. Future Opportunities

For future work, it would be worth spending time on data collection. There are a great number of valuable features that can have a positive relationship with the game outcome. For example, it might be helping to collect information about team members' abilities, the number of investments and funds for teams, and the frequency of training. This information is quite important when predicting the outcome of a game. And they are considered important factors when people try to predict the game outcome. For machine learning, with more features included, the performance of the model can be further enhanced. Due to time and resource limits, it is unable for us to include these features in our model. But it is a potential opportunity for future work.

ACKNOWLEDGMENT

We appreciate the assistance and guidance from Prof. Lyle Ungar and every teaching assistant throughout the semester.

REFERENCES

[1] Dr. James Bond. Soccer world cup 2018 winner, 2022. https://www.kaggle.com/code/agostontorok/soccer-world-cup-2018-winner.

[2] Wild R. Isenko A. et al. Lang, S. Predicting the in-game status in soccer with machine learning using spatiotemporal player tracking data. *Sci Rep*, 16291(12), 2022. https://doi.org/10.1038/s41598-022-19948-1.

[3] Sergio Pessoa. Predicting fifa 2022 world cup with ml, 2022. https://www.kaggle.com/code/sslp23/predicting-fifa-2022-world-cup-with-ml.

[4] Octavio Santiago. Predicting Soccer matches with Machine Learning, 2018. https://levelup.gitconnected.com/predicting-real-soccer-matches-using-fantasy-game-scouts-a3b388edb8aa.