Predicting the Games of English Premier League 2019-2020

Eric Chagoya, echagoya@uci.edu
Mars Dai, bomingd@uci.edu
Brady Hong, bradyh3@uci.edu
Github: https://github.com/EricChagoya/SoccerPredictions

1. Introduction and Problem Statement

This project develops a system to predict the outcome of each game, wins, losses, or draws, for the English Premier League 2019-2020 season. The main features we looked at are each game's statistics for instance the possessions, shots, touches, goals of a single game for the home and away team. We looked at the average statistics for the previous season such as average shots, shots on target, yellow, and red cards for each team. Some of the methods used to create data tables are utilizing publicly available datasets, calling APIs, and web scraping. The models we used to solve this problem are the K Nearest Neighbors classifier and logistic regression. The evaluation method we used is a RPS (Ranked Probability Scores).

2. Related Works

In the field of predicting soccer match outcomes, incorporating domain knowledge has been proven effective[1]. We decided to extract recency features in predicting the outcome of the match. The recency features are extracted by collecting a team's in-game statistics from the number of most recent played matches. We could locate these most recent games using the data of the match variable that is included in the data set.

3. Data sets

https://www.premierleague.com

The first dataset has the in-game statistics for each game in the past 10 seasons. We got this data set from premierleague.com by web scraping. The features we have in the first data set are date, possession percentage, shots on target, tackles, clearances, and corners for both the home team and away team. This data set contains a total of 3800 rows and 16 columns. Each row in the data set represents a single match. The response variable is WDL, its value could be "W" which means the home team won the match, "L" which means the home team lost or "D" which means the game ended with a draw. Offense and defense of a team plays a key role in determining the outcome of the match. In Figure 1, variables like possession percentage, shots on target corners can represent how good a team's offense is while variables like tackles, clearances can shed insights on how well a team is doing defensively.

[1] Berrar, D., Lopes, P. & Dubitzky, W. Incorporating domain knowledge in machine learning for soccer outcome prediction. Mach Learn 108, 97–126 (2019). https://doi.org/10.1007/s10994-018-5747-8

	date	home_team	away_team	home_score	away_score	home_possession_%	away_possession_%	home_shots_on_target	away_shots_on_target
0	08/14/2010	Aston Villa	West Ham United	3	0	56.8	43.2	5.0	1.0
1	08/14/2010	Blackburn Rovers	Everton	1	0	30.4	69.6	2.0	4.0
2	08/14/2010	Bolton Wanderers	Fulham	0	0	46.5	53.5	5.0	4.0
3	08/14/2010	Chelsea	West Bromwich Albion	6	0	59.5	40.5	12.0	1.0
4	08/14/2010	Sunderland	Birmingham City	2	2	44.1	55.9	2.0	6.0

home_tackles	away_tackles	home_clearances	away_clearances	home_corners	away_corners	WDL
27.0	18.0	24.0	36.0	15.0	7.0	W
15.0	16.0	48.0	44.0	1.0	3.0	W
26.0	18.0	54.0	52.0	4.0	8.0	D
16.0	20.0	10.0	24.0	3.0	2.0	W
9.0	16.0	49.0	39.0	3.0	5.0	D

Figure 1: **Webscrapped Data**

https://www.reddit.com/r/PremierLeague/

One of the most popular forums where fans passionately discuss the EPL is the PremierLeague subreddit from reddit.com. We fetched the data from this subreddit to measure the popularity of each team in EPL. Some of the features are total posts, comments, scores, positive and negative posts. Total post is the total number of posts that contain the team name. Total comments are the total number of comments of posts that contain the team name. The score is the total number of upvotes minus downvotes. This dataset has 140 rows with 7 columns.

For the positive and negative posts feature, we trained the word2vec model using the texts of the posts. The model was able to distinguish between positive and negative words. The output from the method most_similar() contains a vector of similar words and their similarity scores. A post was deemed to be positive if there were positive words and negative if there were more negative words. If they tied, then it was not counted as positive nor negative.

	A	B	C	D	E	F	G
1	Season	Team	Total Post	Total Comments	Score	Postivie	Negative
2	2013-2014	Arsenal	148	782	474	78	1
3	2014-2015	Arsenal	114	366	217	26	1
4	2015-2016	Arsenal	139	365	291	17	0
5	2016-2017	Arsenal	286	1086	423	27	0
6	2017-2018	Arsenal	648	3539	1027	62	0
7	2018-2019	Arsenal	385	4524	2375	103	1
8	2019-2020	Arsenal	587	5218	5995	198	0
9	2013-2014	Aston Villa	71	526	282	60	1
10	2014-2015	Aston Villa	47	241	175	21	1
11	2015-2016	Aston Villa	37	51	21	3	0

Figure 2: Reddit Data

https://www.football-data.co.uk/englandm.php

The dataset from the Football-data UK has historical data for the English Premier League since the 1990s. We collected and combined the data from the past 17 years. It has one row for each game played for a total of 6462 games. It has 19 features, with almost all of them being numerical. Some of the more interesting features that are not commonly found in other datasets are the referees for each game and corner counts. Figure 3 provides a subset of the data.

	A	B	C	D	E	F	G	H
1	HomeTeam	AwayTeam	HomeGoals	AwayGoals	Referee	HomeCorners	AwayCorners	HomeYellow
2	Arsenal	Everton	2	1	M Halsey	6	9	1
3	Birmingham	Tottenham	1	0	R Styles	1	4	3
4	Blackburn	Wolves	5	1	J Winter	6	2	1
5	Fulham	Middlesbrough	3	2	G Poll	7	6	1
6	Leicester	Southampton	2	2	M Riley	2	7	3
7	Man United	Bolton	4	0	P Durkin	8	4	0
8	Portsmouth	Aston Villa	2	1	G Barber	7	9	2
9	Charlton	Man City	0	3	M Dean	4	7	2
10	Leeds	Newcastle	2	2	R Martin	4	8	4

Figure 3: Data UK Dataset

Figure 4 shows a correlation heatmap for the features in the first dataset shown in figure 1. The graph shows that home possession %, home shots on target, and home corners are positively correlated with home scores while away team tackles, away team clearance and away team possession percentage are negatively correlated with home score. This is because the better the away team defense the harder it is for the home team to score. Moreover, shots on target are positively correlated with team possession percentage since the longer you have the ball, you will have more chances to shoot the ball. At last, corner kicks are positively correlated with shots on target, because corner kicks are considered a decent opportunity to score the goal.

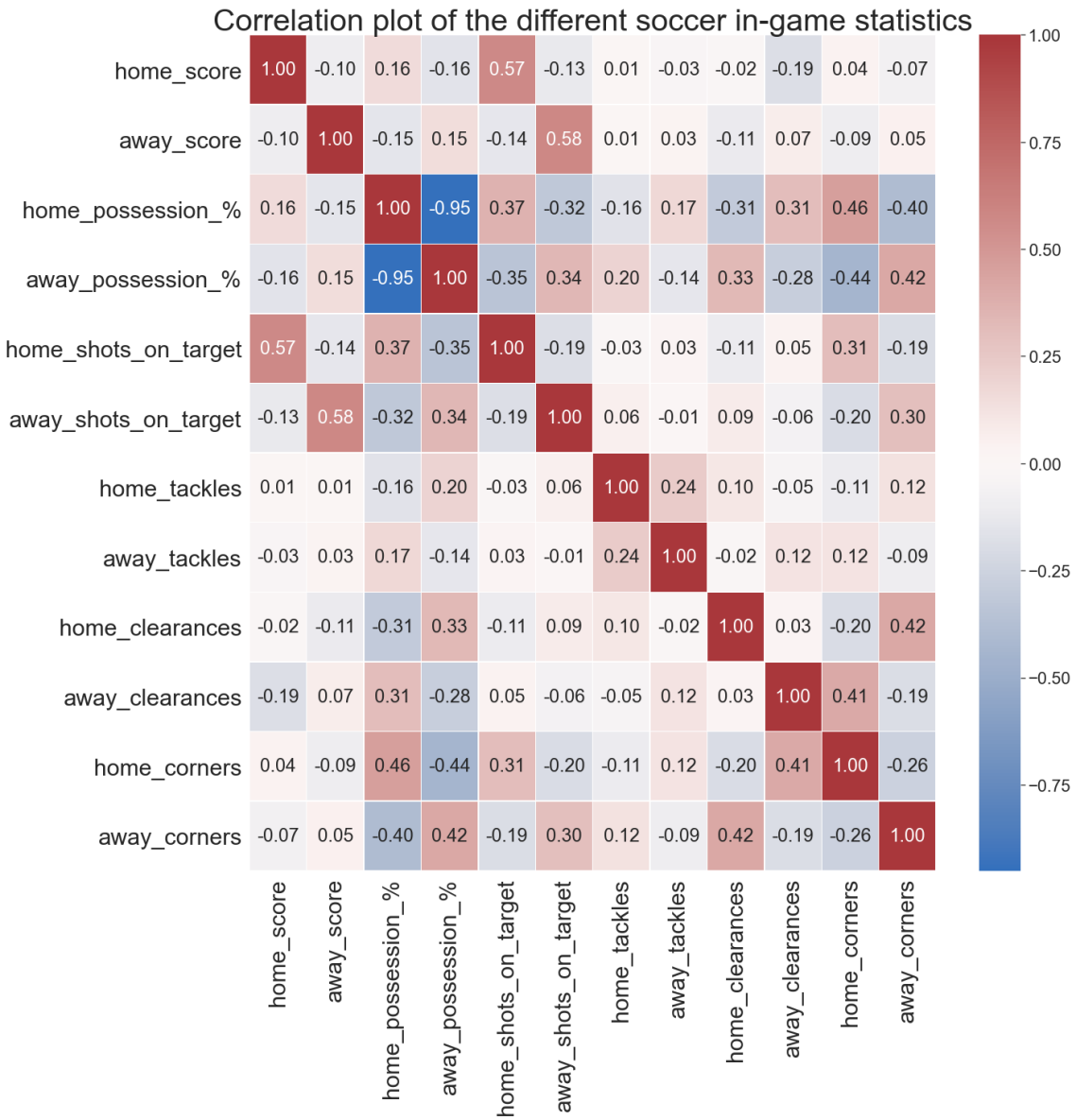

Figure 4: Correlation plot

On the heatmap, it shows the number of posts that each team in the EPL got. that specific teams got many posts for each season consecutively like Arsenal, Chelsea, Liverpool, Manchester City, and Manchester United. They are the most popular teams in the EPL for many consecutive years. How many posts overall the team got could convey how popular they are. There is a good possibility that the popular team gets into the top 4. Looking at the 2019-2020 season, Liverpool, Manchester City, Manchester United, and Chelsea took 1st through 4th. Therefore, it shows that the total posts of the team can be a potentially good feature to predict the outcome of each match.

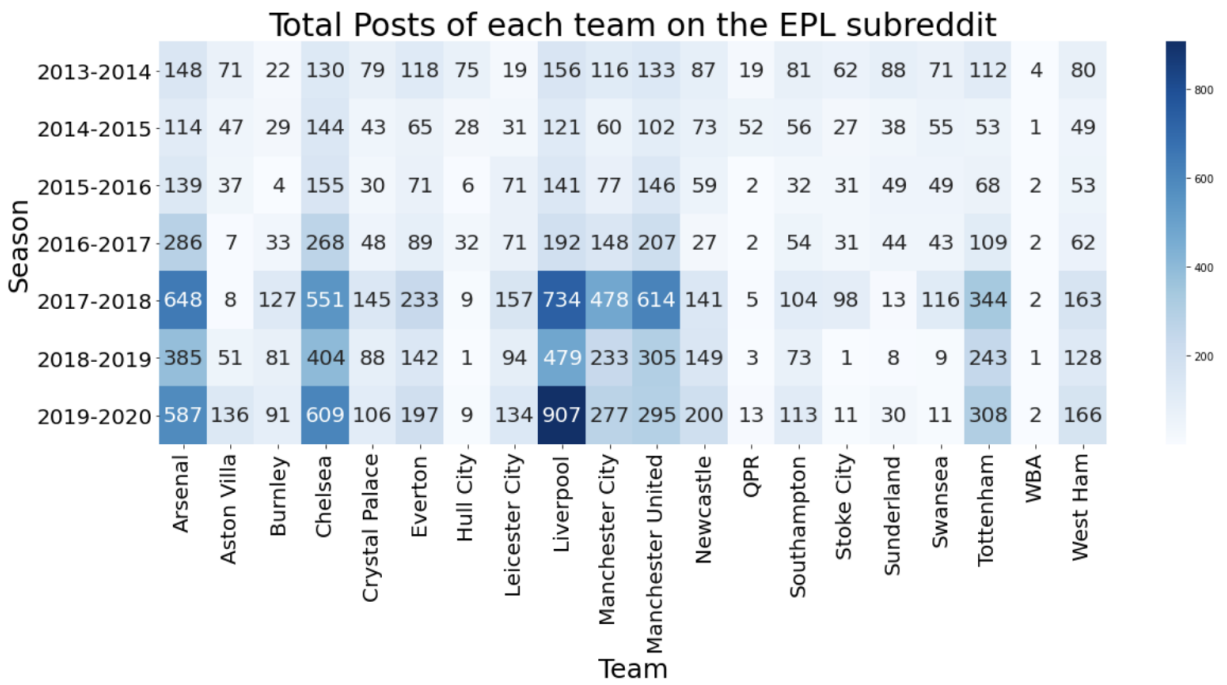

Figure 5: Heatmap of Reddit Data

4. Overall Technical Approach

We used the Pushshift API for gathering the data from Reddit. The Pushshift API allows us to search posts from a specific time. This API finds all the queries that we type into the parameter. In order to collect data from premierleague.com, we modified the Web scraping code written by Otavio[2]. We are currently using Amazon RDS for PostgreSQL for our cloud computing database. It was easy to set up and has backups in case the data gets corrupted. If the group ever decides to improve on the process, Amazon RDS has easy scalability.

For the recency feature extraction, there are 4 feature groups per team. The attacking strength feature group represents a team's ability to score goals. We calculate this feature by finding many goals the team scores for each game. The defensive strength feature group represents a team's ability to prevent goals by the opponent. We calculate this feature by how many scores the team allows goals from the opponent for each match. The home advantage feature group is used to determine a team's home-field advantage based on attacking and defensive strengths. The strength of the opposition group is calculated by the average goal difference the opponent achieved in prior matches. In addition to those four features, we added shots on target, and possession for the feature extraction.

These 6 features are called for each team for the past n games. A higher n means we look at more past games for both teams while a smaller n looks at less. The hard part is finding a big enough n to have enough relevant data, but small enough that the last game is still important in deciding the outcome of that game.

[2] https://github.com/otavio-s-s/data_science/tree/master/Premier%20League%20Scraping

	home_team	away_team	t-3 home_attacking	t-3 home_defending	t-3 home_strength	t-3 home_home	t-3 home_posession	t-3 home_shots_on_target
1534	Everton	Manchester United	2	0	-0.6666666666666670	1	32.7	8
1539	Manchester United	Chelsea	0	2	1.0	-1	53.9	4
1549	Manchester United	Chelsea	2	1	0.0	1	42.3	4
1553	Manchester United	Crystal Palace	0	4	0.3333333333333330	-1	51.7	1
1560	Manchester United	Leicester City	1	1	-1.6666666666666700	1	51.7	5
1565	West Ham United	Manchester United	2	2	-1.6666666666666700	1	49.3	3
1569	Manchester United	Arsenal	1	2	0.0	1	71.3	3
1573	Newcastle United	Manchester United	1	0	-0.3333333333333330	-1	19.8	3
1577	Manchester United	Liverpool	0	2	1.3333333333333300	-1	53.5	3
1593	Manchester United	Aston Villa	1	1	-0.6666666666666670	1	54.8	4
1595	Manchester United	Tottenham Hotspur	0	1	-0.3333333333333330	-1	68.1	3
1603	Manchester United	Everton	1	1	-1.0	1	32.0	2

Figure 6: Depth of the Recency Feature Extraction

Total features: 12 * depth of the recency feature extraction

For our baseline model, we used a k nearest neighbor classifier, because kNN is the simplest and most commonly used algorithm for classification problems. It is non-parametric , meaning that it does not make any assumptions on the underlying data distribution. Berrar uses this algorithm to solve a similar problem and receives good results. kNN works as follows, it calculates the distances between a query and all the examples in the data, then it will pick the K number of examples that is closest to the query, then votes for the most frequent label. Thus, Choosing the correct value for k is critical.

Logistic regression model is another classifier that is commonly used to solve labling problems. Assumptions of a multinomial logistic regression includes

Data level: The dependent variable should be dichotomous in nature for binary regression.
Error Term: The error term is assumed independently.
Linearity: Does not assume a linear relationship, but between the odd ratio and the independent variable, there should be a linear relationship.
No outliers: Assumes that there should be no outliers in data.
Large sample: Uses the maximum likelihood method, so a large sample size is required for logistic regression.

(Allison)

5. Software

Software Used	Experience
Reddit API Script	The inputs were the data of each post in EPL subreddit. The output was one file that combined data with 7 features.
Web Scraper	The input is the match ID for each match. The output was one file that contains match data from the 2010 to 2020 season.

Script for Cleaning Data from DataUK	The inputs were all the data from the 2003 season up until 2020. The output was one file that combined all the data.
PostgreSQL	The inputs are the individual datasets. The output is the combined dataset.
Model	The input is the combined dataset. It produces a model predicting the outcome of a match. It shows graphs of the model's accuracy.

Figure 10: Software Created

6. Experiments and Evaluation

Comparing two models by lowest mean RPS for each game depth, The best result was using K-nearest neighbors with game depth of 5 according to Figure 7. We aggregated recency features with reddit score and total posts and used that dataset.

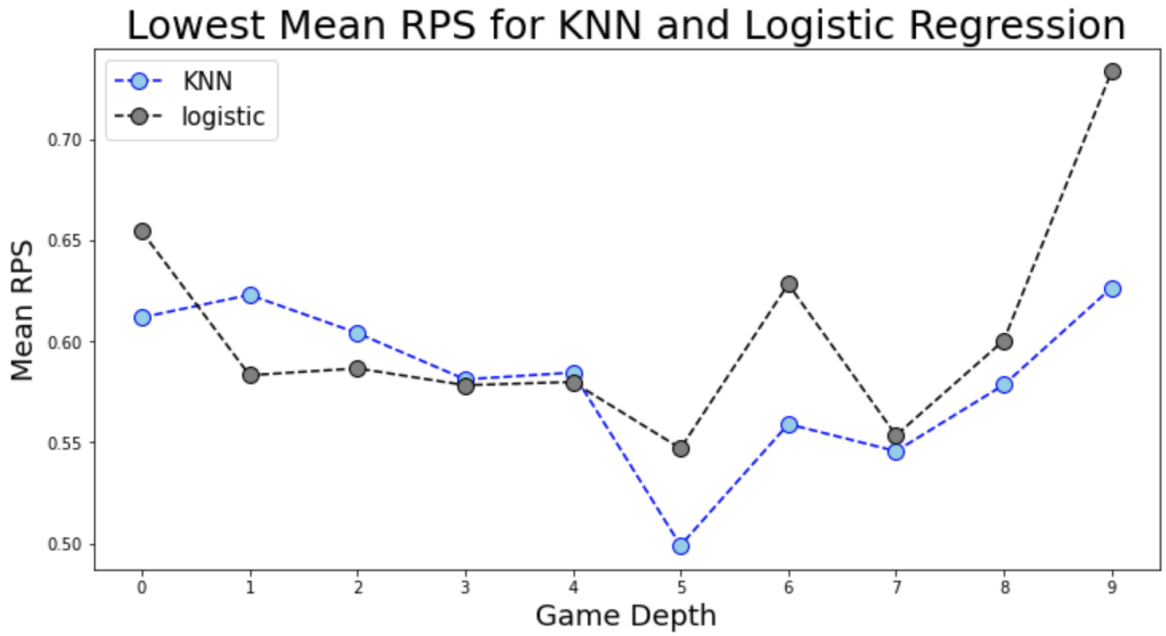

Figure 7: RPS for Different Game Depth

For the k-nearest neighbor, we tested k values from 1 to 200 and evaluated each K with mean RPS. According to Figure 8, k = 17 had the lowest mean RPS = 0.499 when we chose game depth of 5. We have selected 0.08 percent of the data which are 350 games to be our test data.

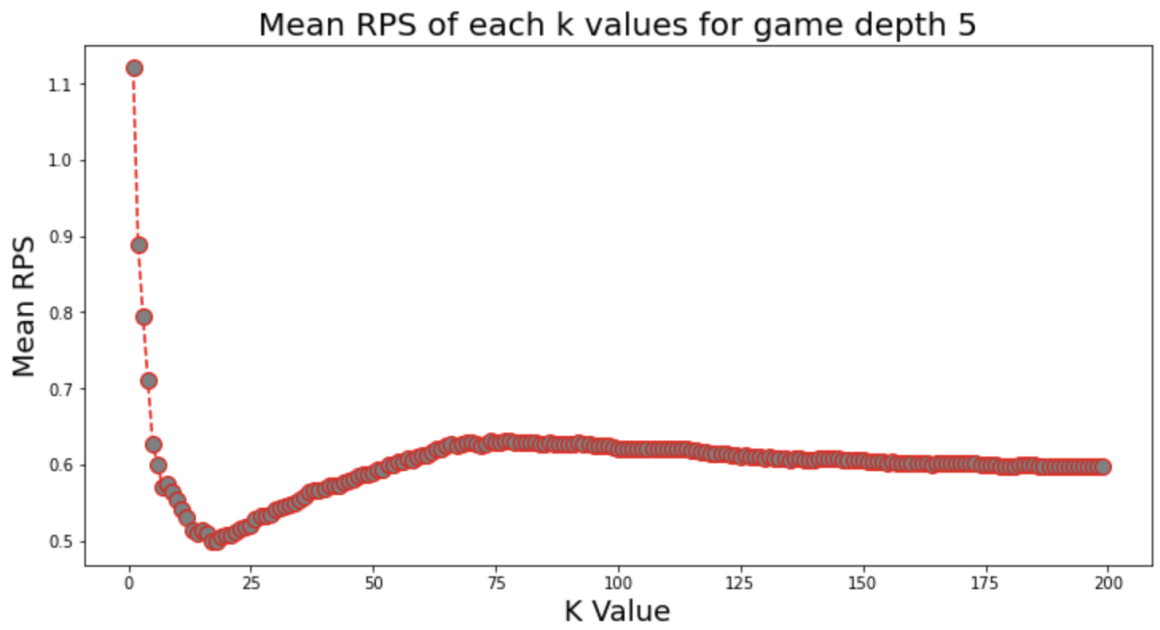

Figure 8: RPS for different k

For the logistic regression model, we have cleaned our data and made sure all the assumptions are checked out. We then trained a logistic regression model using the default parameters from sklearn and the same features. It performed the best with a mean RPS of 0.547.

For the evaluation method we choose to use a metric called ranked probability scores that is also from the paper written by Berrar et al. The equation is defined as:

$$\mathrm{RPS} = \frac{1}{r-1}\sum_{i=1}^{r-1}\left(\sum_{j=1}^{i}(p_j - a_j)\right)^2$$

Figure 9: Ranked Probability Score Formula

r means the number of possible outcomes, for us it is 3 which are A, D and H. For each match, we will calculate the predicted probabilities for A(away team wins the game), D(draws) and H(home team wins the game) and store these 3 probabilities into a vector p = (p_a, p_d, p_h) with p_a + p_d + p_h = 1. if the real outcome is a win for the home team, then a=(1,0,0).

7. Notebook Evaluation

The Notebook has three major components. The first part is used for exploratory data analysis, we investigate the relationship between the features we have in our data set and the outcome of the match. This part helps us to select the most appropriate feature to train our model. The second component is for data cleaning, we do most of our data extraction here and also prepare the needed data points for training the model. At last, we train two models in the last part of our notebook. Figure 12 in the appendix has the code for training the model and finding the best k value and n value for the model.

Section 8: Members Participation

Eric mostly dealt with cleaning up the data, and dealing with inconsistencies each dataset had. He also played a big role in actively reviewing each presentation, script, and report. Brady worked on aggregating data from reddit and extracted recency features from the combined dataset. He also worked on data visualization for the reddit data and models as well. Mars did exploratory data analysis, research on similar topics and trained and evaluated the models. Despite each of us having our own roles, we actively helped each other. Some problems we needed each others help on was not knowing how to rotate a figure, steering the direction of the model, or even basic things like the type of joins to use.

Tasks	Team Members
Aggregate Data from Reddit	Brady
Web Scraper	Mars
Cleaning, Transforming, and Joining Data	Eric
Recency Feature Engineering	Brady
Model Creation	Mars
Data Visualization	Mars - 0.33 Brady- 0.33 Eric- 0.33
Custom Notebook	Mars
Editing Scripts and Drafts	Eric

Figure 11: Team Member Tasks

We learned how important having someone with domain expertise is. Their intuition guided us on what features we should be paying more attention to and which features look more important than they actually are.

9. Discussion and Conclusion

One of the most difficult things we had to deal with was the amount of freedom we were given. In the vast majority of our coursework, we are told to follow the instructions to the letter, with the solution being ideas we have learned in the past week. There is value in being given instructions in this manner, but it inhibits our problem-solving to ideas we have recently seen instead of thinking about similar problems we have seen in different courses or projects. Having more freedom led us to more dead ends, but more opportunities to learn. It made us reflect more on similar types of problems. It made us realize how stubborn we were being. Thousands

of people are trying to solve the same problem and we could have done a better job of seeing what they did right and wrong.

If we had more time and resources, we would look into finding bigger datasets. When researching, we found that most projects had at least 20,000 games because they looked at all Leagues, while our project was just limited to the Premier League. We would also look into other models. For this project, we only played around with knn and logistic regression, but there are numerous models that try to look at game predictions like Bradley Terry, ELO, BART, etc. Finally, we could have done a better job incorporating soccer fan's opinions from social media. Soccer fanatics have watched thousands of hours watching the sport, learning trends, keeping up with player injuries or illness. These people have valuable information that would otherwise be difficult to quantify or difficult to find the datasets.

Works Cited

Berrar, Daniel, et al. “Incorporating Domain Knowledge in Machine Learning for Soccer Outcome Prediction.” Machine Learning, Springer US, 7 Aug. 2018, link.springer.com/article/10.1007/s10994-018-5747-8.

Allison, P. D. (1999). Comparing logit and probit coefficients across groups. Sociological Methods and Research, 28(2), 186-208.

Additional Graphs

```
d = {}
d1 = {}
e = {}
knns = []
log_mean = []
np.random.seed(31415)
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler
from sklearn.neighbors import KNeighborsClassifier
from sklearn.linear_model import LogisticRegression
for i in range(67,176,12):
    x1 = dt_drop.iloc[:,67:i]
    x3 = pd.concat([x1,x], axis = 1)
    dt_new = pd.concat([x2,x3], axis = 1)
    print(dt_new)
    X = dt_new.iloc[:, 3:-1].values
    y = dt_new.iloc[:, dt_new.shape[1]-1].values
    X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.08)
    scaler = StandardScaler()
    scaler.fit(X_train)
    X_train = scaler.transform(X_train)
    X_test = scaler.transform(X_test)
    error = []
    p_y = []
    for j in y_test:
        if j == 'A':
            p_y.append(np.array([1, 0, 0]))
        elif j == 'D':
            p_y.append(np.array([0, 1, 0]))
        else:
            p_y.append(np.array([0, 0, 1]))
```

```
    # Calculating error for K values between 1 and 40
    for k in range(1, 200):
        knn = KNeighborsClassifier(n_neighbors=k)
        knn.fit(X_train, y_train)
        pred_i = knn.predict(X_test)
        p_test = knn.predict_proba(X_test)
        rps=[]
        for m in range(len(p_test)):
            temp = rankedProbabilityScore(p_test[m], p_y[m])
            rps.append(temp)
        error.append(np.mean(rps))
    a = min(error)
    knns.append(error.index(a))
    e[i] = error
    d[i] = a
    for n in range(0,len(error)):
        if error[n]==a:
            d1[n] = i
    clf = LogisticRegression(random_state=0).fit(X_train, y_train)
    pred_clf = clf.predict_proba(X_test)
    rps_log =[]
    for i in range(len(pred_clf)):
        temp = rankedProbabilityScore(pred_clf[i], p_y[i])
        rps_log.append(temp)
    log_mean.append(np.mean(rps_log))
```

Figure 12