APPLICATION OF MEMORY-BASED COLLABORATIVE FILTERING TO PREDICT FANTASY POINTS OF NFL QUARTERBACKS

A Thesis
Submitted to the Graduate Faculty
of the
North Dakota State University
of Agriculture and Applied Science

By

Dienul Haq Ambeg Paramarta

In Partial Fulfillment of the Requirements
for the Degree of
MASTER OF SCIENCE

Major Department:
Computer Science

April 2019

Fargo, North Dakota

North Dakota State University

Graduate School

Title

APPLICATION OF MEMORY-BASED COLLABORATIVE FILTERING TO PREDICT FANTASY POINTS OF NFL QUARTERBACKS

By

Dienul Haq Ambeg Paramarta

The Supervisory Committee certifies that this thesis complies with North Dakota State University's regulations and meets the accepted standards for the degree of

MASTER OF SCIENCE

SUPERVISORY COMMITTEE:

Dr. Juan Li
Chair

Dr. Anne Denton

Dr. Pratap Kotala

Dr. Mila Kryjevskaia

Approved:

May 1, 2019
Date

Dr. Kendall Nygard
Department Chair

ABSTRACT

Subjective expert projections have been traditionally used to predict points in fantasy football, while machine prediction applications are limited. Memory-based collaborative filtering has been widely used in recommender system domain to predict ratings and recommend items. In this study, user-based and item-based collaborative filtering were explored and implemented to predict the weekly statistics and fantasy points of NFL quarterbacks. The predictions from three seasons were compared against expert projections. On both weekly statistics and total fantasy points, the implementations could not make significantly better predictions than experts. However, the prediction from the implementation improved the accuracy of other regression models when used as additional feature.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

LIST OF ABBREVIATIONS

CF ..... Collaborative Filtering

D/ST ..... Defense/Special Teams

INT ..... Interception

MAE ..... Mean Absolute Error

MAPE ..... Mean Absolute Percentage Error

MLB ..... Major League Baseball

MLS ..... Major League Soccer

NBA ..... National Basketball Association

NFL ..... National Football League

NHL ..... National Hockey League

NN ..... Neural Network

QB ..... Quarterback

RB ..... Running Back

RMSE ..... Root Mean Squared Error

SVM ..... Support Vector Machine

TD ..... Touchdown

TE ..... Tight End

WR ..... Wide Receiver

LIST OF APPENDIX TABLES

1. INTRODUCTION

Fantasy football has seen a large growth within the past decade. Fantasy Sports Trade Association (FTSA) reported that fantasy sports is a $7.22 billion industry [1]. Two of the top fantasy football sites is reported to have over $3.2 billion in revenue in 2017 [2]. This number is even more impressive considering that the revenue of the major professional American football league, National Football League (NFL), is estimated around $14 billion [3]. Fantasy football is expected to have an even bigger growth as recent Supreme Court decision paved the way for states to legalize sports gambling [4].

In fantasy football, one important aspect of the game for team owners is picking the right players for their team. The performance of players in real games translate into points in the fantasy game. Selecting the right players requires being able to predict the actual points that real football players will score in a game. Traditionally, to predict the fantasy points, team owners rely on the projections made by experts from various sites such as CBS and ESPN. However, these projections have been shown to be subjective and inaccurate [5].

Recently, computing techniques started to be explored to predict fantasy points, although a few have come from academia. Various regression methods have been used to predict total fantasy points for a player in a season [6]–[8]. Both regression methods and neural network have been explored for weekly fantasy points predictions, albeit with varying degree of successes and limitations [9]–[12]. The complex system created by IBM and ESPN is by far the most promising in the fantasy football prediction domain, although details on the implementations are scarce given its proprietary nature [13].

This paper explores memory-based collaborative filtering (CF) to predict fantasy football points. *CF* techniques have been widely used in the recommender system domain [14], where a

system predicts a user rating or preference based on the past recorded ratings by all users on all items. *CF* uses past behaviors of a user to create a set of neighbors for the user. Then, these neighbors' preference toward an item is used to generate a prediction of the user's preference of the item.

In this paper, both user-based and item-based collaborative filtering were explored to create a system that predicts the fantasy points and statistics of NFL quarterbacks. Ten seasons of quarterbacks' statistics from 2009 to 2018 were used for this study. The ten years of data was analyzed to display relevant trends of quarterbacks from 2009 to 2018 and the importance of the position. Then, the data was used in user-based and item-based CF to predict the weekly fantasy points and individual statistics of quarterbacks. The results were compared to expert projections from four sites: *CBS*, *Fantasy Sharks*, *FF Today*, and *NFL*. Finally, the prediction from the collaborative filtering implementation was used as an additional feature in Support Vector Machine regression models.

2. BACKGROUND

2.1. American Football

2.1.1. Overview

American football is played on rectangular field with dimension of 120 yards long by 53 yards, 1 foot wide [15]. As with many other sports, the goal of the game is for one team to score more points than the opposing team. The game is divided into four 15-minutes long quarters. A team can advance the ball toward the opposing team's end zone and get into the scoring range by throwing the ball or running the ball. A team can score points in the game in multiple ways:

- **Touchdown (TD) – 6 points.** A ball is carried into an opponent's end zone or caught in the end zone.
- **Extra point – 1 point.** A ball is kicked through the uprights of the opponent's goal post after a touchdown.
- **2-pt conversion – 2 points**. A ball is carried into an opponent's end zone or caught in the end zone after a touchdown.
- **Field goal – 3 points.** A ball is kicked through the uprights of the opponent's goalpost.
- **Safety – 2 points**. A player tackles an opposing player in the opposing player's own end zone.

At any one time, only 11 players per team are allowed on the team. A team who has possession of the ball is playing the offense. The offensive unit attempts to move the ball toward the opponent's end zone and score points. The opponent, i.e. the defensive unit, attempts to limit the progress and gain back the possession of the ball. A team can take possession of the ball in several ways:

- **Receiving a kickoff**. A team receives a kickoff at the beginning of each half and after the other team scores
- **Turnover**. A team recovers a ball dropped by the other team (fumble) or catches a ball thrown by the other team (interception/INT)
- **Safety**. Tackling the other team's player within his own end zone
- **Turnover on downs**. The offensive team fails to advance the ball 10 yards in four downs, surrendering the ball to the other team
- **Punt.** The offensive team opts to kick the ball to the other team, as an alternative to turnover on downs

2.1.2. Positions

American football is a sport where the players are highly specialized. Typically, a player is designated to play in only one position. Each position has unique responsibilities and sometimes unique abilities, in terms of what that position is allowed and not allowed to do, that only that position can perform. The three units in a team – offensive, defensive, and special teams – each has unique positions that only exist within those units.

In offensive units, players can fill in one of these positions:

- **Quarterbacks (QB)**. Quarterbacks throw the ball, hand off the ball to other player, or carry the ball themselves. Quarterbacks also act as the leader of the offensive unit and are on the field for almost all offensive plays.
- **Offensive Linemen (OL)**. These players provide protection to and pave ways for quarterback, running back, and other ball carriers against the opposing defensive unit.
- **Running Backs (RB)**. Running backs generally carry and run the ball forward.

- **Wide Receivers (WR)**. Wide receivers run further down the field and catch the ball thrown by the quarterback.
- **Tight Ends (TE)**. Tight ends are hybrid players that can perform the role of receivers and/or linemen at any given time.

2.1.3. National Football League

National Football League (NFL) is the major and most popular American football league [16]. The NFL consists of 32 teams divided into two conferences, National Football Conference (NFC) and American Football Conference (AFC). The conference is further divided into four divisions of four teams each.

Each year, the NFL has three phases: pre-season, regular season, and postseason. Pre-season consists of four games over a period of four weeks. Teams use pre-season games as a way to practice and evaluate the players on their roster. The result of pre-season games is not counted toward the teams' record.

The regular season consists of 17 weeks. Each team plays one game per week for 16 weeks, with one additional "bye" week between those weeks (typically between week 4 to 12) to rest. A team is not guaranteed to play against every other team in the league, except the teams that belong to the same division of that team.

After regular season ends, postseason begins with 12 qualified teams, six from each conference. The six teams in each conference are the four division winners and the next two teams with the best regular season record.

2.2. Fantasy Football

Fantasy Sports has been a growing phenomenon in the United States and around the world. Fantasy Sports Trade Association (FTSA) reported that in 2017, fantasy sports is a $7.22

billion industry, with 59.3 million players in United States and Canada [1]. Daily Fantasy Sports sites such as FanDuel and DraftKings, meanwhile, is reported to have collected just over $3.2 billion in entry fees from their users [2]. This is a substantial amount, considering that the revenue of the professional American football league, National Football League (NFL), is estimated to be around $14 billion [3].

2.2.1. Rules

Fantasy football allows an ordinary person or fan to take on the role as a team owner. A fantasy team owner, or fantasy player, selects a set of football players to create a team that earns points based on the players' performance on the real-life games. This team competes head-to-head in a league consisting of multiple fantasy teams [17].

Like in real football team, a fantasy team consists of players of various positions. Depending on the type of league and scoring standard, the number of players in each position may differ. *Table 2.1.* shows examples of team composition [18], [19]. In a fantasy team, however, each of these players may come from different teams.

Table 2.1. Fantasy team compositions in standard and DFS leagues

Position	*Number of players*	
	Yahoo (traditional)	*DraftKing (DFS)*
QB	1	1
RB	2	2
WR	2	3
TE	1	1
FLEX	1	1
Kicker	1	1
D/ST	1	1

2.2.2. Scoring

The goal of a fantasy football game is to have a team whose players score more points than the players from the other team. The fantasy points of each player in a team is determined

by the statistics that the player attains in a real game. A scoring system is used to map the statistics to fantasy points.

Multiple scoring systems exists, such as standard, PPR (point-per-reception), half-PPR, and Individual Defensive Player (IDP). For this study, the standard scoring system [20] was used. The scoring system for offense is defined on *Table 2.2*.

Table 2.2. Standard scoring system for offensive statistics

Statistics	*Points*
Passing yard	0.04
Passing touchdown	4
Interception	-2
Rushing yard	0.1
Rushing touchdown	6
Receiving yard	0.1
Receiving touchdown	6
2-point conversion	2
Fumble lost	-2

For example, in week 17 of 2018 season, Jameis Winston had the following statistics: 35 pass attempts, 22 pass completions, 345 passing yards, 4 passing touchdowns, 1 interception, 2 rushing attempts, 23 rushing yards, no 2-pt conversion, and no fumble lost. Using the standard scoring system, Jameis Winston scored 30.1 fantasy points.

2.2.3. Types of League

There are two types of fantasy football leagues: traditional league and daily fantasy sports league [21].

Traditional fantasy football league is a season-long league. Each traditional league typically consists of 8 to 12 fantasy teams. At the beginning of the season, fantasy team owners hold a draft to select the players for their team roster. A player cannot be owned by more than one fantasy team at a time within a league. Each week, team owners select a set of players from their roster for the starting lineup. Throughout the season, fantasy team owners may drop/waive a

player, add a player from free agent, or trade a player with other teams in the same league. A fantasy team owner is traditional league is expected to commit to play the game through the entire season. Some of the most popular sites that players user for traditional season-long fantasy football include ESPN, Yahoo, and CBS.

In daily fantasy sports (DFS) league, a fantasy team owner selects the entire team roster every week. Therefore, a DFS team may have a completely different set of players from one week to the next. A football player in this type of league may be owned by more than one fantasy team at a time. To prevent fantasy team owners to select only the top performing players from each position, a salary cap system is used. Each team is limited to a set amount of salary, and each player has a salary that is indicative of the player's projected performance in the upcoming match. A team owner in a DFS league such as FanDuel or DraftKings may compete head-to-head against another fantasy team or compete for the highest score in a league with large number of teams. A fantasy team owner in DFS league is not committed to participate every week throughout the season.

3. PREVIOUS WORK

3.1. Challenges

Unique challenges present in both human and computer fantasy football points prediction.

With any sports, players progress and regress as they go through their career. A player who had performed well the previous season may not perform as well during the next season. Conversely, a rookie player who might still try to get the hang of the game may break out and perform over the expectation next season. Incorporating window of time is important in fantasy football prediction.

Human and computer predictions may assume that player progression is smooth, therefore predictions can use the last *n* games or matches to plot the rate of player progression. This is more difficult at the beginning of the season when there have not been sufficient games played yet. Thus, predictions have to fall back to the games played last season to estimate the progression, while keeping consideration that there is an offseason period between seasons when data of players or games is limited. The longer the offseason is, the harder it would be to estimate the progression a player has during the offseason. Among the five major American sports leagues (NFL, MLB, NBA, NHL, and MLS), NFL by far has the longest offseason period. In 2013-2014 season, NFL had 183 days of offseason. MLB (baseball) had the next longest offseason with 119 days, while MLS (soccer) had the shortest offseason with 53 days [22].

In addition to longer offseason, the limited number of games that each NFL team has per season makes it harder for fantasy football predictions compared to other major fantasy sports. Each team in NFL plays 16 regular season games [23]. This is significantly fewer than 82 games for an NBA team or 162 games for an MLB team [24], [25].

While predictions can opt to use only current season to avoid dealing with the offseason gap, it is still important to use longer historical data. NFL game plays are constantly changing. It has been well-documented that NFL gameplay is moving from a run-heavy strategy to a more passing-friendly one [26], [27]. Including data from multiple seasons is important to see the rate of change and to predict the outcome of upcoming games.

Lastly, injury impacts on how much playing time a player gets, and therefore how many points a player can score in the fantasy football setting. A player who unexpectedly gets injured during a game will have his playing time cuts short and will likely score fewer points than expected. Conversely, a player who is reported to have injury and expected to have limited playing time may actually get more playing time and score more points than expected. Although NFL teams are required to report player injury, the injury reports that NFL teams produce have been known to be inconsistent across different teams [28], [29].

3.2. Expert Projections

Traditionally fantasy players rely on expert projections to make their decision on which players to draft and/or set as starting players on their team. Websites that host fantasy leagues (e.g. CBS, Yahoo, ESPN) as well as fantasy football enthusiast community websites (e.g. FF Today, Fantasy Sharks) employ experts to make projection on how many points players will score for each week and for the entire season.

These expert projections can be subjective and may vary wildly. For example, ESPN's expert projections have been shown to overestimate player points by at least 25% half the time [5]. Websites such as FantasyPros and Fantasy Football Analytics offer free and premium services to fantasy players that weigh and aggregate the projections by experts, in attempt to

generate more accurate predictions for their users. These aggregated projections tend to give better predictions than the projections of individual sites and/or expert [30].

3.3. Computer Predictions

Unlike other fields such as fantasy soccer, baseball, or basketball, fantasy football is harder to quantify and predict [9]. Based on the number of games played in a season, American football has a very limited number of data available. Each NFL team plays only 16 games per season, while in Major League Baseball, each team plays 162 games per season. Additionally, the performance of a player in a team depends heavily on the performance of other players on the team, more so than in other domains such as baseball. This makes it difficult to predict the performance of a player who has different set of teammates throughout the season.

3.3.1. Season Total Points Predictions

There have been studies focused on applying algorithms to predict how many points a player will score for the entire season. Knoche used Support Vector Machine (SVM) to predict season total points for quarterback, wide receiver, running back, and tight end with R squared scores of 0.67, 0.72, 0.54, and 0.71 respectively [31]. Hart used a Random Forest model that is able to achieve R squared scores of 0.84 and 0.78 for quarterback and running back respectively [6]. Autoregressive Moving Average (ARMA) model has been used to predict season's score for quarterback [7]. Porter used Autoregressive Integrated Moving Average (ARIMA) model to predict entire season scores for all players with an impressive mean absolute percentage error (MAPE) of 3.53%, although the study only contained players with at least 15 games in their career [8]. This type of season points prediction is useful to assist fantasy players in traditional league for the fantasy draft at the beginning of the season. However, the lack of weekly points

prediction makes this type of prediction less helpful for fantasy players in setting their weekly starting lineups in both traditional and DFS leagues.

3.3.2. Weekly Points Predictions

In attempt to predict weekly points of quarterback, Lutz used Support Vector Regression and Neural Network [10]. Lutz put an emphasis on the importance of predicting the top 24 quarterbacks, as in a traditional league of 12 teams, it is likely that only about 24 quarterbacks would be considered for the starting lineup. Lutz's SVR model achieves RMSE of 7.833 for the top 24 quarterbacks, while his Neural Network with one hidden layer creates a model achieving RMSE of 7.868.

King & Leboulluec compared several different models to predict weekly points of quarterbacks[11]. In the study, King et al used Random Forest, Principal Components Regression, Boosted Tree, and Support Vector Regression. King et al used many input features such as player's height and weight, years of experience, NFL combine results, and multiple expert projections. King et al PCR model is able to achieve RMSE of 4.24, while their SVR, Boosted Tree, and Random Forest model have RMSE of 7.30, 7.70, and 7.72 respectively.

Parikh applied K-Nearest Neighbors (KNN) and Support Vector Machine (SVM) to create different model for each player position [9]. For all positions except quarterback, both KNN and SVM models are able to generate predictions that outperform benchmark expert projections. Parikh also found that his SVM model results in a higher accuracy level than the KNN model in all player positions.

Landers & Duperrouzel used least squares and gradient boosted decision trees as regression methods to predict weekly points for all positions [12]. Landers et al also used additional feature such as FanDuel's player salary on top of standard game statistics such as

player's average points. Landers et al acknowledged that feature such as player's salary is, in a sense, FanDuel's expert projections as player with higher salary is expected to score more points. Landers et al found that boosted decision tree performs better with R squared value of 0.417, while their least squares model provides an R squared value of 0.401. Additionally, the model achieves highest scoring teams against 300,000 randomly selected fantasy teams in 5 out of the 11 weeks tested in the 2016 season.

With the exception of Lutz's study, one of the main pitfalls for the weekly points prediction models above is that they rely exclusively on the statistics of the previous games during the same season. For example, Landers et al used the first 6 weeks of a season to create models that predict the rest of the season [12]. King et al used the first 7 weeks as the training set, while reserving only week 13 to 16 for testing [11]. These models cannot be used for early part of the season as they relied exclusively on the results of the matches from first few weeks of the season for training. In traditional league, this portion of the season equates to almost half of the regular fantasy football season.

Perhaps the most extensive application of both machine learning and deep learning toward fantasy football is the collaboration between IBM and ESPN [13]. IBM used a series of Watson AI pipeline that includes Neural Networks, regression, and simulation to build the system. Watson first collects related articles, video, and podcasts from ESPN and other trusted sources. For each article, Watson rates whether there is a positive sentiment or negative sentiment toward a player. These sentiments are aggregated by players into an overall sentiment score. The articles are also fed into 4 models of neural network with 98 layers to determine the probability of a boom (player exceeding expectation), bust (player underperforming expectation), player playing with an injury, and player playing meaningful minutes.

Watson predicts the fantasy points for players through regression using the sentiment scores, the player's state classifications (boom, bust, injury, and meaningful minutes), ESPN expert projections, and other relevant data as inputs. Watson also runs simulation to predict the maximum and minimum points that the players can get. Watson continually runs this pipeline throughout the season, producing weekly forecast and predictions of players.

The collaboration between IBM and ESPN is significant. First, the system takes account the probability of injury, something that is rarely done in fantasy football predictions by machine. Second, the produced output is not just the predicted points; Watson also produces probability of boom or bust, maximum and minimum points, and the injury probability for players so fantasy team owners can decide on their own. And finally, the predicted points by Watson achieves RMSE of 6.78 for all positions, an impressive score considering the volatility of fantasy football.

4. DATA SET

The data for this study is obtained from *github.com/derek-adair/nflgame* [32]. The data set contains raw data that is extracted directly from NFL. The data set contains 10 seasons of play to play statistics and results from all pre-season, regular, and post-season games from 2009-2018.

4.1. Pre-Processing

A few pre-processing steps were performed to the raw data available from the site. Aggregation by player and game was performed on the offensive play-by-play statistics. This step resulted 51,687 rows of offensive statistics, with each row representing a player's statistics in a game. There were over 3,477 unique players and 25 statistics/features. These features are shown on *Table 4.1*.

Table 4.1. Features available in the dataset

Feature	*ID*	*Used*	*Feature*	*ID*	*Used*
Pass attempt	00	Yes	*Receiving target*	13	No
Pass completed	01	Yes	*Receiving reception*	14	No
Pass yard	02	Yes	*Receiving yard*	15	No
Pass touchdown	03	Yes	*Receiving touchdown*	16	No
Pass interception	04	Yes	*Receiving longest*	17	No
Pass 2-point attempt	05	Yes	*Receiving longest touchdown*	18	No
Pass 2-point made	06	Yes	*Receiving 2-point attempt*	19	No
Rushing attempt	07	Yes	*Receiving 2-point made*	20	No
Rushing yards	08	Yes	*Fumbles total*	21	Yes
Rushing touchdown	09	Yes	*Fumbles own team recovery*	22	No
Rushing longest	10	No	*Fumbles total recovery*	23	No
Rushing 2-point attempt	11	Yes	*Fumbles recovery yards*	24	No
Rushing 2-point made	12	Yes	*Fumbles lost*	25	Yes

Note that players in this data were not exclusively players who hold an offensive position in their team. Defensive players were included in this data if they had logged at least one of the features above in a game.

Since this study focused on the quarterback position, non-quarterback players needed to be filtered out. Although the original data set has player's position attribute (e.g. QB, WR), this attribute is unreliable primarily because the value of this attribute has been deleted for players who have retired as of the beginning of 2018.

Alternatively, the number of pass attempts a player had in a game can be used as an indicator whether a player is a quarterback. A player who has attempted multiple passes in a game is likely a quarterback. Similar to a previous study, the threshold of five for passing attempt was used to get quarterbacks from the data set[10]. This threshold effectively filtered out backup quarterbacks with no significant play time and non-quarterbacks who attempted a pass in trick or misdirection plays. This step yielded 5,472 rows of player by game statistics.

Additionally, only 14 out of 25 features were used in the final data set. The "receiving" features had low variance, with only 62 out of 5,472 rows had non-zero values. Moreover, most projections do not have receiving features as one of the projected features, as receiving is not an important aspect of a quarterback.

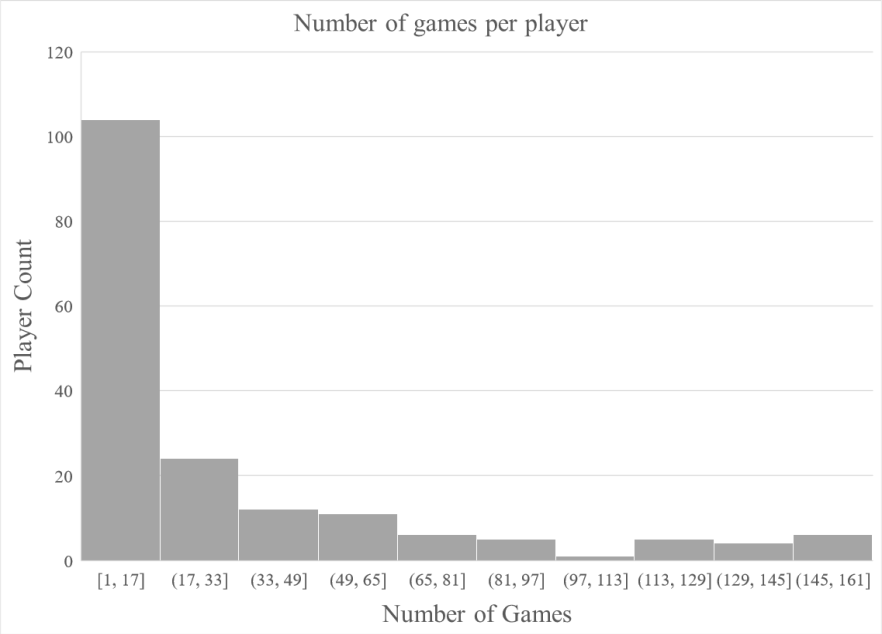

Figure 4.1. Histogram of number of games per player

The majority of players in the final data set have not had significant playing time in more than 16 games, as shown on *Figure 4.1*. This is expected, as there are few quarterbacks who have started multiple seasons throughout their career. Most quarterbacks have either started only for a few games before they lost their starting position, or they have only seen significant playing time in backup capacity. Only one player, Phillip Rivers, has played in every game in the last 10 seasons. The list of all players in the final dataset used in this study is in *Appendix A.1*.

After filtering, the final data set had 178 unique players with 14 features. Players in the dataset had an ID attribute that will be used as user. Each team had an abbreviation, but few teams who had relocated within the past two seasons had multiple abbreviations. Therefore, each team was assigned an ID as shown in *Table 4.2*. Item was a combination of team and feature IDs. For an example, item with an ID of 0301 represented the statistic of pass completed against Buffalo Bills. With 32 teams, this translated into 448 items that the 178 users could theoretically "rate."

Table 4.2. List of teams with abbreviation and encoded ID

Team Name	*Abbreviation*	*Team ID*	*Team Name*	*Abbreviation*	*Team ID*
Arizona Cardinals	ARI	00	*Los Angeles Chargers*	LAC, SD	16
Atlanta Falcons	ATL	01	*Los Angeles Rams*	LA, STL, LAR	17
Baltimore Ravens	BAL	02	*Miami Dolphins*	MIA	18
Buffalo Bills	BUF	03	*Minnesota Vikings*	MIN	19
Carolina Panthers	CAR	04	*New England Patriots*	NE	20
Chicago Bears	CHI	05	*New Orleans Saints*	NO	21
Cincinnati Bengals	CIN	06	*New York Giants*	NYG	22
Cleveland Browns	CLE	07	*New York Jets*	NYJ	23
Dallas Cowboys	DAL	08	*Oakland Raiders*	OAK	24
Denver Broncos	DEN	09	*Philadelphia Eagles*	PHI	25
Detroit Lions	DET	10	*Pittsburgh Steelers*	PIT	26
Green Bay Packers	GB	11	*Seattle Seahawks*	SEA	27
Houston Texans	HOU	12	*San Francisco 49ers*	SF	28
Indianapolis Colts	IND	13	*Tampa Bay Buccaneers*	TB	29
Jacksonville Jaguars	JAX, JAC	14	*Tennessee Titans*	TEN	30
Kansas City Chiefs	KC	15	*Washington Redskins*	WAS	31

To take an account of player progression, a timestamp field was created. Timestamp *t* was a function of season *s* and week *wk* when the corresponding game was played.

$$t = f(s, wk) = 25 * (s - 2009) + wk$$

A timestamp period of eight was added between seasons to penalize the offseason period. One season therefore equaled to 25 timestamp units (17 for regular season weeks plus 8 for offseason). For instance, a game that was played in week 15 of 2009 season had a timestamp of 15, while a game that was played in week 2 of 2018 season had a timestamp of 227.

The final data set was in the form of comma separated values (CSV) file. The CSV file had four columns: user(player), item (team-feature), rating, and timestamp. A snapshot of the CSV file is shown in *Figure 4.2*.

```
user, item, rating, timestamp
2506109,3000,43,1
2506109,3001,33,1
2506109,3002,363,1
...,...,...,...
...,...,...,...
2560858,2721,2,242
2560858,2725,2,242
```

Figure 4.2. A snapshot of the final csv file used

4.2. Data Trend

The ten seasons data from 2009 to 2018 shows trend in the NFL that affects fantasy football.

4.2.1. Passing and Rushing Yards by Season

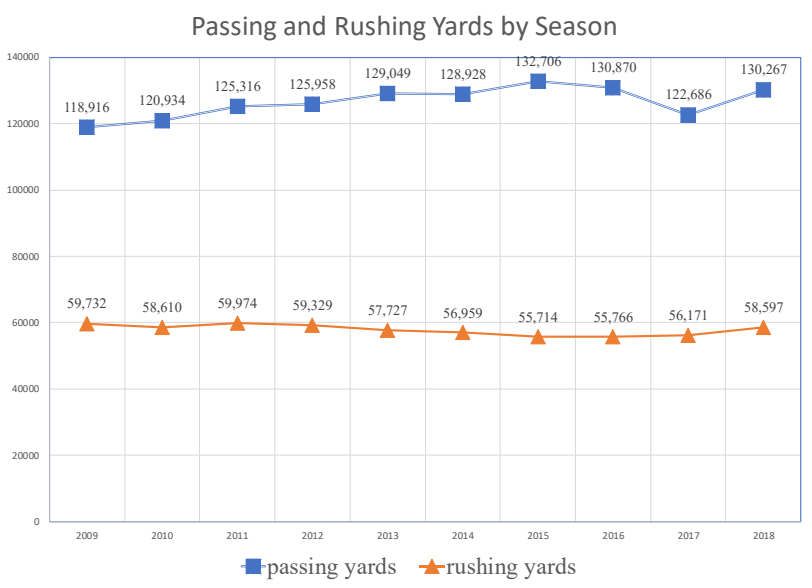

Figure 4.3. Total passing yards and rushing yards by season

It has been known that the NFL is evolving into more of a passing league and leaving its run-heavy past [26], [27]. The total passing and rushing yards throughout the season from 2009 to 2018 shows this trend. The amount of passing yards has steadily increased throughout the ten seasons, except for the 2017 season. The amount of rushing yards, meanwhile, has slowly decreased with the exception of 2017 and 2018 seasons.

4.2.2. Fantasy Points per Player by Game Aspect

Each statistic/feature was assigned to the aspect of the game it belonged to. For example, passing yard and interception were grouped into passing, rushing yards and rushing touchdowns were grouped into running, receiving yards and receiving two points conversions were grouped into receiving, and so on. Then for each grouping, fantasy points were calculated using the standard scoring system.

Assuming that each position is exclusively responsible for the aspect of the game it is designed for, positions were grouped into the three aspects. Quarterback (QB) was assigned into passing, running back (RB) was assigned into running, and wide receiver/tight end (WR/TE) were assigned into receiving. It is important to note that this is not the case in the real game, as there are many QBs who can run as well as RBs who can receive and catch the ball.

Using the typical compositions of 1 QB, 2 RBs, 2 WRs, 1 TE, 1 FLEX (RB/WR/TE), there was one player responsible for the fantasy points in the passing category, 2.5 for running, and 3.5 for receiving. *Figure 4.4.* shows that passing category is the most important category in fantasy football. It also shows that quarterback, being the position responsible for the points accumulated in passing category, is the most important player in a fantasy team.

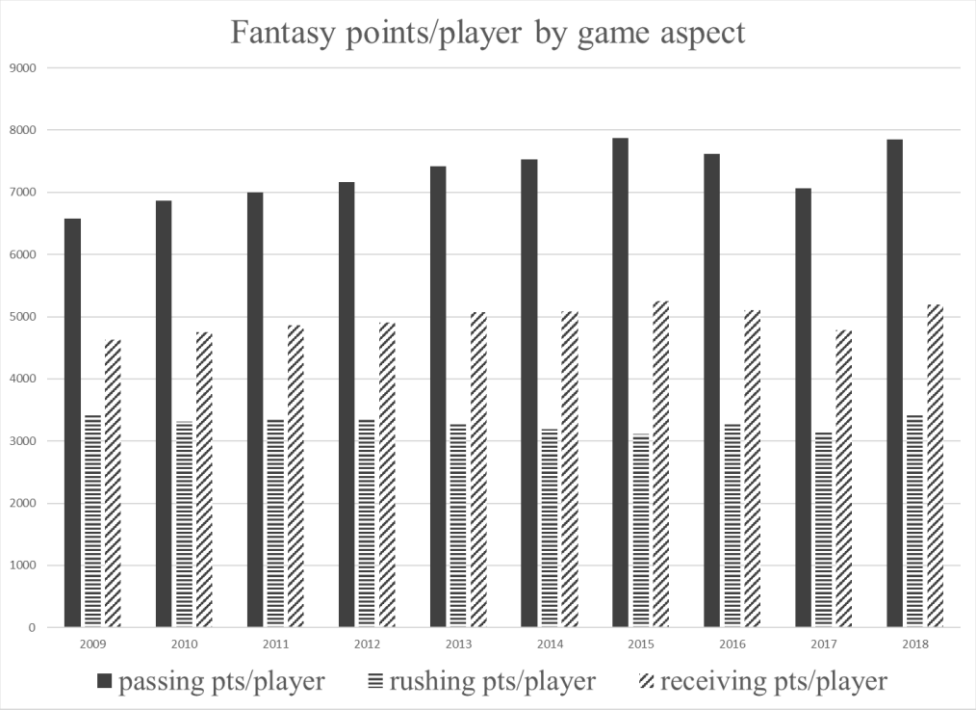

Figure 4.4. Total fantasy points per player by game aspect and season

4.2.3. Overall Quarterback Performance by Season

Using the final filtered data (only players with at least five pass attempts), *Table 4.3.* shows the aggregate statistics of quarterbacks by season. Between 2009 to 2018, quarterbacks have become more efficient overall. Quarterbacks completed more of their pass attempts, while at the same time they threw more touchdowns and less interceptions per pass attempt.

Table 4.3. Cumulative quarterback passing statistics by season

season	*pass attempts*	*pass completions*	*completion rate*	*pass yards/attempt*	*pass TD/100 attempts*	*pass INT/100 attempts*
2009	16,869	10,299	61.05%	6.99	4.16	3.04
2010	17,172	10,436	60.77%	7.01	4.34	2.92
2011	17,323	10,423	60.17%	7.20	4.28	2.88
2012	17,701	10,782	60.91%	7.08	4.24	2.62
2013	18,026	11,047	61.28%	7.13	4.42	2.76
2014	17,802	11,159	62.68%	7.20	4.51	2.51
2015	18,241	11,496	63.02%	7.26	4.61	2.37
2016	18,198	11,483	63.10%	7.16	4.28	2.25
2017	17,411	10,821	62.15%	7.01	4.23	2.43
2018	17,552	11,392	64.90%	7.36	4.75	2.38

Table 4.4. shows quarterbacks rushing statistics. Quarterbacks attempted more run between 2009 to 2018 and gained more yards per attempt. Quarterbacks have been evolving to be more than just a threat in the passing game to a defense, but also a threat in the running game.

Table 4.4. Cumulative quarterback rushing statistics by season

season	*rushing attempts*	*rush yards/attempt*	*rush TD/100 attempts*
2009	1,210	3.39	3.47
2010	1,362	4.08	3.01
2011	1,516	4.00	4.29
2012	1,488	4.29	4.23
2013	1,647	4.43	3.46
2014	1,593	4.18	2.95
2015	1,566	4.22	3.90
2016	1,477	4.10	4.47
2017	1,598	4.46	4.13
2018	1,743	4.47	3.84

Overall, quarterbacks have become more efficient and dynamic. This in turn makes quarterbacks much more valuable in fantasy football.

5. METHODS

In this study, user-based and item-based collaborative filtering were explored and applied to predict the statistics of players in American football games, and effectively, to predict the fantasy points players will score in a given game.

The collaborative filtering implementations in this study took a configuration of parameters. To find the best configuration, parameter optimization for each implementation was performed. Parameter optimization used NFL regular games data for seven seasons from 2009 to 2015. Testing was performed on regular season games during three seasons from 2016 to 2018. Then, the result was compared against expert projections from CBS, FFToday, FantasySharks, and NFL. Standard scoring system was used to calculate fantasy points.

The dataset was limited to players with quarterback positions. As shown in the previous chapter, quarterback is arguably the most important and influential positions, both in real football game and in the fantasy domain. Moreover, unlike other positions in an offensive unit such as running back or wide receiver, quarterback has a consistent amount of playing time because generally only one quarterback usually plays for each team during an entire game.

5.1. Overview of Collaborative Filtering

Collaborative filtering has been widely used in the recommender system domain. Tapestry is one of the earliest implementations of collaborative filtering in recommender system [33]. For a given user, Tapestry generates recommendations based on the explicit opinions of closely related neighbors of that user. Meanwhile, GroupLens used collaborative filtering to create recommender system for news and movies based on the past ratings that users gave [34], [35]. Given a target user and a target item, the algorithm finds other closely-related users/neighbors based on the ratings they have given on the same set of items. The algorithm

then calculates how the target user would rate the target item based on how the neighbors have rated the target item in the past.

Sarwar et al proposed an alternative in the form of item-based collaborative filtering [36]. Instead of finding the closest neighbors for a user, the algorithm precomputes the similarities between items. Given a target user and a target item, the algorithm will then calculate the rating based on how the target user has rated closely-related items to the target item. The algorithm gains popularity in domains where sparsity and scalability are issues such as in the e-commerce domain [37].

Beside user-based and item-based collaborative filtering, there are other methods such as Bayesian network, clustering, and content-based collaborative filtering with wide applications [14]. The output of collaborative filtering can be either binary classification (user will like or dislike the item) or regression (numerical rating prediction).

This study explored whether user-based and item-based collaborative filtering can be applied to predict a player's statistics against an opposing team. The users in this case were the players. The items were the team-features, or the unique combinations of all 32 teams and 14 features.

In user-based collaborative filtering, the prediction of target player against a target item (team-feature) were calculated based on how players who were similar to the target player had performed against the target item. Therefore, for target user *a*, and target item *i*, the predicted score was calculated with the formula:

$$P(a,i) = \overline{r_{a,x}} + \frac{\sum_{u \in U}(r_{u,i} - \overline{r_{u,x}})\, w_{a,u}}{\sum_{u \in U} |w_{a,u}|} \quad (1)$$

x was the feature being coded in the item *i*, *U* was the set of users who had rated (played against) item *i*, $r_{u,I}$ was the rating for user *u* on item *i*, $w_{a,u}$ was the weight or similarity between

users a and $u \in U$, and $\overline{r_{a,x}}$ and $\overline{r_{u,x}}$ were the average ratings (statistics) that user a and u had for item x.

In the item-based collaborative filtering, the prediction was calculated based on how the target player had performed against the target items (team-feature) that were similar to the target item. Therefore, for target user a, and target item i, the predicted score was calculated with the formula:

$$P(a,i) = \frac{\sum_{j \in J} r_{u,j} w_{i,j}}{\sum_{j \in J} |w_{i,j}|} \quad (2)$$

J was the set of items that user a has rated, $w_{i,j}$ was the weight or similarity between items i and $j \in J$, and $r_{a,j}$ was the rating for user a on item j.

5.2. Similarity Functions

The weight or similarity between users or items can be calculated in several ways. Some of the most popular functions to calculate this weight are Pearson correlation, cosine distance, and adjusted cosine distance.

For two users or items a and u, Pearson correlation can be calculated as:

$$w(a,u) = \frac{\sum_{i \in I_a \cap I_u} (r_{a,i} - \overline{r_a})(r_{u,i} - \overline{r_u})}{\sqrt{\sum_{i \in I_a \cap I_u} (r_{a,i} - \overline{r_a})^2} \sqrt{\sum_{i \in I_a \cap I_u} (r_{u,i} - \overline{r_u})^2}} \quad (3)$$

Where $I_a \cap I_u$ is the set of items that both user a and u have rated in user-based algorithm, or the set of users that have rated both item a and u. One disadvantage of Pearson is that it tends to produce a high similarity when the size of the common set is small.

Cosine similarity or cosine distance can be expressed as:

$$w(a,u) = \frac{\sum_{i \in I} r_{a,i} r_{u,i}}{\sqrt{\sum_{i \in I} r_{a,i}^2} \sqrt{\sum_{i \in I} r_{u,i}^2}} \quad (4)$$

Unlike Pearson correlation, cosine distance iterates over the set of all items, whether it has been rated or not. For non-rated items, a default value (typically zero) is assigned. This reduces the possibility of small-sized common set. However, cosine distance fails to take an account that different users may rate items on different scale (a user may tend to rate items higher or lower than others).

Using cosine distance over mean-centered vectors is suggested as an alternative to calculate the similarity. Mean-centered cosine, or adjusted cosine, can be expressed as:

$$w(a,u) = \frac{\sum_{i \in I}(r_{a,i} - \overline{r_a})(r_{u,i} - \overline{r_u})}{\sqrt{\sum_{i \in I}(r_{a,i} - \overline{r_a})^2}\sqrt{\sum_{i \in I}(r_{u,i} - \overline{r_u})^2}} \quad (5)$$

In user-based algorithm, if both users have rated all items, adjusted cosine distance is practically the same as Pearson correlation. This also applies in item-based algorithm where both items have been rated by all users.

In practice, the best similarity function depends on the application [38]. These three similarity functions were tested during parameter optimization to determine the best one for the user-based or item-based collaborative filtering implementations.

5.3. Implementations

The algorithms were implemented in JavaScript on Node.js. Node.js is an open-source, cross-platform, asynchronous event-driven JavaScript runtime environment Node.js allows developers to use JavaScript for server-side scripting and command line tools outside web browsers. [39]. The node package math.js is was to perform various mathematical and matrix operations [40]. RStudio was used for statistical analysis of parameter optimization and testing results [41].

Unlike many other regression methods, memory-based collaborative filtering does not require training. Both user-based and item-based implementations took the following parameters:

- *similarityFn:* the similarity functions to use (Pearson, cosine, or mean-centered cosine).
- *minSimilarity:* minimum similarity of weight between users/items *a* and *u* for *u* to be considered as a neighbor of *u*.
- *useNegativeNeighbor:* include neighbors with negative similarity in the pool.
- *neighborSize:* the maximum number of neighbors included in calculation.
- *targetRecentThreshold*: the number of previous period (in timestamp unit) that target user had posted the ratings (statistics of a game) within for the ratings to be included.
- *glbalRecentThreshold*: the number of previous period (in timestamp unit) that all other ratings had to be posted within for the ratings to be included.

These parameters affected the number of training games and neighbors available. If there were not enough training games or neighbors for a target, the mean for an item's feature was used as default value. This default value substitution occurred to rookie/first-year players who had never played before or players who had not played in a long period of time due to reasons such as injury, suspension, or simply lack of skills to be able to play in a game.

This study used the standard scoring system to calculate the fantasy points for a player based on the player's statistics in a game.

5.3.1. User-Based Collaborative Filtering

To predict the fantasy points of target player *a* against a team in the game with timestamp of *t*, the collaborative filtering implementation predicted the set of items $i \in I_{a,t}$ consisting of 14 team-features associated with the game. The following steps were performed in the user-based collaborative filtering:

1) Select a set of ratings *R*, where *U* is the set of all other players and *I* is the set of all items:

$$R = A \cup B$$

$$A = \{ r_{a,i} \mid i \in I, (t - targetRecentThreshold) \leq t_{a,i} \leq (t - 1) \} \quad (6)$$

$$B = \{ r_{u,i} \mid u \in U, i \in I, (t - globalRecentThreshold) \leq t_{u,i} \leq (t - 1) \}$$

a) If A is an empty set (players had not played a game within the specified time period), for each items $i \in I$, return the mean of feature x associated with the item as the predicted result and jump to step 7:

$$P(a, i) = \overline{r_x} \quad (7)$$

$$P(a, t) = \{ P(a, i) \mid i \in I_{a,t} \}$$

2) Normalize $r \in R$ based on its feature x, where $x \in X$ was the set of 14 features as described in Section 4.1:

$$r = \frac{r - \overline{r_x}}{\sigma_x}, \; r \in R \quad (8)$$

3) For each player $u \in U$, calculate the similarity between target player a and player u, $w_{a,u,}$ using the selected similarity function as described in Section 5.2.

4) Select a set of top n neighbors $u \in U_a$, where $n = neighborSize$

$$U_a = \left\{ u \in U \;\middle|\; \begin{cases} us\ NegativeNeighbor, w_{a,u} \geq minSimilarity \\ \neg useNegativeNeighbor, |w_{a,u}| \geq minSimilarity \end{cases} < neighborSize \right\} \quad (9)$$

a) If U_a is an empty set (player a did not have close neighbors), use the mean of each item's feature as described in step 1a and jump to step 7.

5) For each item $i \in I_{a,t}$, the set of items associated with player a against a team in the game at timestamp t, calculate the predicted rating for the item

$$P(a, i) = \overline{r_{a,x}} + \frac{\sum_{u \in U_a} (r_{u,i} - \overline{r_{u,x}}) \, w_{a,u}}{\sum_{u \in U_a} |w_{a,u}|} \quad (10)$$

$$P(a, t) = \{ P(a, i) \mid i \in I_{a,t} \}$$

6) De-normalize each prediction $p_{a,i} \in P(a,t)$

$$p_{a,i} = \left(p_{a,i}\sigma_x\right) + \bar{r_x},\ p_{a,i} \in P(a,t) \tag{11}$$

7) Let s_x be the fantasy point "weight" for a feature/statistic x according to the standard scoring system as described in section. To get the fantasy points for a player a in the game at timestamp t, run the function FP for the set of predicted items

$$FP(a,t) = \sum_{p_{a,i} \in P(a,t)} p_{a,i} s_x \tag{12}$$

Note: some statistics such as pass completions or rush attempts do not contribute to fantasy points in the standard scoring system. These features had fantasy point weight of zero.

5.3.2. Item-Based Collaborative Filtering

For item-based collaborative filtering implementation, the steps performed were similar, albeit with few modifications. To predict the fantasy points of target player a in against a team in the game with timestamp of t and a set of items (team-features) $i \in I$, these following steps were performed in the implementation:

1) Select a set of ratings R, where U is the set of all players and J is the set of all other items:

$$R = A \cup B$$

$$A = \left\{ r_{a,i} \,\middle|\, i \in I, (t - targetRecentThreshold) \leq t_{a,i} \leq (t - 1) \right\} \tag{13}$$

$$B = \left\{ r_{u,ij} \,\middle|\, u \in U, ij \in I \cup J, (t - globalRecentThreshold) \leq t_{u,ij} \leq (t - 1) \right\}$$

a) If A is an empty set (players had not played a game within the specified time period), for each items $i \in I$, return the mean of feature x associated with the item as the predicted result and jump to step 7:

$$P(a,i) = \bar{r_x} \tag{14}$$

$$P(a,t) = \{ P(a,i) \mid i \in I \}$$

2) Normalize $r \in R$ based on its feature x, where $x \in X$ is the set of 14 features:

$$r = \frac{r - \overline{r_x}}{\sigma_x},\ r \in R \tag{15}$$

3) For each item $i \in I$, calculate the similarity between the target item i and another item j, $w_{i,j,}$ using the selected similarity function.

4) Select a set of top n neighbors $j \in J_i$, where $n = neighborSize$

$$J_i = \left\{ j \in J \;\middle|\; \begin{cases} use\ NegativeNeighbor, w_{i,j} \geq minSimilarity \\ \neg useNegativeNeighbor, \left|w_{i,j}\right| \geq minSimilarity \end{cases} < neighborSize \right\} \tag{16}$$

   a) If J_i is an empty set (item i did not have close neighbors), use the mean of the feature associated with item i as described in step 1a and jump to step 7.

5) For each item $i \in I$, the set of items associated with player a against a team in the game at timestamp t, calculate the predicted rating for the item

$$P(a,i) = \frac{\sum_{j \in J_i} r_{a,j}\, w_{i,j}}{\sum_{j \in J_i} |w_{i,j}|} \tag{17}$$

$$P(a,t) = \{\, P(a,i) \mid i \in I \,\}$$

6) De-normalize each prediction $p_{a,i} \in P(a,t)$

$$p_{a,i} = \left(p_{a,i}\sigma_x\right) + \overline{r_x},\ p_{a,i} \in P(a,t) \tag{18}$$

7) Let s_x be the fantasy point "weight" for a feature/statistic x according to the standard scoring system as described in section. To get the fantasy points for a player a in the game at timestamp t, run the function FP for the set of predicted items

$$FP(a,t) = \sum_{p_{a,i} \in P(a,t)} p_{a,i} s_x \tag{19}$$

5.4. Parameter Optimization

To get the best configuration of the parameters, a k-fold cross validation (k=5) on the data from 2009 season to 2015 season. *Table 5.1.* shows the sets of parameters tested for each algorithm.

Table 5.1. Parameters for CF implementations and their possible values

Parameter	*User-based*	*Item-based*
similarityFn	Pearson, cosine, adjusted cosine	Pearson, cosine, adjusted cosine
minSimilarity	0.3, 0.5	0.3, 0.5
useNegativeNeighbor	true, false	true, false
neighborSize	3, 5, 10, 15, 20, 25, 30, max	10, 20, 30, 40, 50, 60, 75, 100, 150, 200, 250, max
targetRecentThreshold	5, 10, 17, 25, 50, max	5, 10, 17, 25, 50, max
globalRecentThreshold	17, 25, 50, 75, max	17, 25, 50, 75, max

The difference between the set of possible *targetRecentThreshold* for user-based and that for item-based was due to the number of possible neighbors. Item-based algorithm was able to utilize all of the items in the dataset as neighbors in a given period. Meanwhile, user-based algorithm could only rely on the players who had played a game within the given period.

The 2009-2015 data was divided into 5 partitions of roughly equal sized based on the number of games and players.

Table 5.2. Number of players and games by partition

Partition	*Players*	*Games*
0	29	773
1	29	765
2	29	769
3	29	768
4	28	773

Some configurations of parameters were not be able to produce prediction (lack of neighbor size or training data). In the parameter optimization, the implementations did not return the features' mean to make prediction. Instead, the implementations returned a null value. Coverage was considered using the number of non-null values the implementations produced. The result of parameter optimization is discussed in Section 6.1.

5.5. Predictions Test

Testing for predictions was performed on the games from 2016 season to 2018 season using the best configuration of parameters. The data from 2009 to 2015 seasons were used for initial training data. Then for every week after, the game data from all previous weeks were added to the training data. For example, for testing games of season 2016 week 1, the training data included data from season 2009 week 1 to season 2015 week 17; for games of season 2017 week 8, the training data was the data from season 2009 week 1 to season 2017 week 7.

The results of the testing were compared to the benchmark expert projections from four sites – CBS, Fantasy Sharks, FF Today, and NFL – in Section 6.2 and 6.3 [42]–[45]. Both the fantasy points as well as the feature predictions were compared.

5.6. Output as Feature

Support vector machine (SVM) regression models were built to test whether the output from the collaborative filtering implementations could be used to supplement other existing models. SVM was picked as it has been used previously by multiple studies [9]–[11]. *libsvm-js* was used to implement to train and test the models [46]. Two kernels were used in the SVM models.

The first model was built using linear kernel. The linear kernel took two hyperparameters, cost (C) and epsilon (ε). The values used for these hyperparameters were taken

from a previous study ($C = 0.25$, $\varepsilon = 0.25$) [10]. The second model used radial basis function (RBF) kernel. The optimal values of the three hyperparameters (cost, epsilon, and gamma) were not disclosed in the previous study. The default values for these hyperparameters were used for the RBF kernel of the SVM model ($C = 1$, $\varepsilon = 0.1$, $\gamma = 1$/number of features).

There were 88 features based on the average statistics of the player, team, and opposing defense for the past one, five, and ten weeks. Features selection was performed using Recursive Feature Elimination function from the *caret* package in R [47]. Features selection resulted in 15 features as show in *Table 5.3*.

Table 5.3. Features Used in the SVM Models

Features	*Description*
player_5_score	Average of player's fantasy points in the last 5 games
player_10_score	Average of player's fantasy points in the last 10 games
player_5_passing_yds	Average of player's passing yards in the last 5 games
player_1_score	Player's fantasy points in the last game
player_10_passing_yds	Average of player's passing yards in the last 10 games
player_5_passing_tds	Average of players passing TDs in the last 5 games
player_10_passing_tds	Average of players passing TDs in the last 10 games
player_1_passing_yds	Player's passing TDs in the last game
player_5_passing_cmp	Average of players completed passes in the last 5 games
team_5_offense_yard	Average of team's total yards in the last 5 games
team_5_offense_point	Average of team's point in the last 5 games
player_5_passing_att	Average of player's attempted passes in the last 5 games
team_5_offense_firstDown	Average of team's first downs in the last 5 games
team_5_offense_passYard	Average of team's pass yards in the last 5 games
player_10_passing_cmp	Average of players completed passes in the last 10 games

Both linear and RBF kernel models were trained using 15 features above from the 2016 and 2017 seasons. A second set of models were trained with the data from 2016 and 2017 seasons. The second set of models used 16 features, adding the prediction from the user-based *CF* implementation as another feature, *predictedScore*. Then, all models were tested using games from the 2018 models and the results were compared. The results are shown in Section 6.4.

6. RESULT

RStudio was used to analyze all results data, while Microsoft Excel was used to generate graphs. Beside parameter optimization results, the results of how the implementations performed on predicting both fantasy points and individual feature/statistic were also analyzed.

Root Mean Squared Error (RMSE) and Mean Absolute Error (MAE) were used to measure the accuracy of the results. For a prediction a set of predictions *P* and a set of observations *O* with size of *n*, RMSE is calculated as:

$$RMSE = \sqrt{\frac{\sum_{i=1}^{n}(P_i - O_i)^2}{n}} \tag{20}$$

MAE is calculated as:

$$MAE = \sum_{i=1}^{n}|P_i - O_i| \tag{21}$$

For this study, RMSE was prioritized over MAE, as RMSE gives heavier penalty for predictions that are further away from the actual observations.

Coverage was also considered, as certain configuration of the implementations could not make predictions on all instances due to insufficient training data or neighbors. Coverage was defined as the number of predictions an implementation was able to produce divided by the number of actual observations.

6.1. Parameter Optimization

The preliminary results from parameter optimization did not look promising. For user-based CF, all configurations had MAE score of above 6.00 and RMSE score of above 7.00. For item-based CF, few configurations had fairly low MAE and RMSE scores, although these configurations had very poor coverage. Filtering configurations with at least 50% coverage resulted in configurations with MAE score of above 6.00 and RMSE score of above 7.80. These

poor results could be attributed to fewer available data points, as roughly 20% of players were being held on each run.

The following is a brief overview of how each parameter independently affects the result of the implementations. *Appendix A.2* and *A.3* have more details on the results of parameter optimization.

- **Similarity Function**. Raw, non-adjusted cosine distance performed the best for both user-based and item-based collaborative filtering. In order, adjusted cosine (mean-centered) and Pearson correlation followed behind cosine.
- **Minimum Similarity/Weight**. Two values, 0.3 and 0.5, were being compared as minimum similarity value. For both implementations, the value 0.3 produced better result than 0.5, while achieving higher coverage.
- **Negative Neighbor**. In item-based collaborative filtering, using neighbors with negative weight or similarity did not seem to have a significant effect, although it slightly worsened the accuracy of the result. In user-based, the use of negative neighbors improved the accuracy of the predictions.
- **Neighbor Size**. In user-based CF, using 30 neighbors as predictors produced the best result. For item-based CF, the quality of the predictions increases as the number of neighbors increased, and peaked at 150, and gradually worsened after.
- **Target Recent Threshold**. this parameter dictated how recent the target player's game data must be within to be included in the training data. For user-based, using all of the target player's data, regardless how recent it was, yielded the best result. For item-based, using 50 timestamp units, equaled to two regular seasons and two offseason periods, yielded the best prediction.

- **Global Recent Threshold**. this parameter dictated how recent all other data beside the target player's must be within to be included in the training data. For both implementations, using as much training data as possible, regardless recency, seemed to produce the best result.

It is important to note that using combination of individual best value for each of the parameters may not result in the best result. As parameters interacted with each other, the implementations may produce less accurate prediction or fail to make prediction (poor coverage). *Appendix A.2* and *A.3* show the top 10 configurations for each algorithm based on RMSE and MAE. *Table 6.1.* displays the configuration eventually used in testing, based on the balance of coverage and accuracy.

Table 6.1. Configurations of parameters used for testing

Parameter	*User-based CF*	*Item-based CF*
Similarity Function	cosine	cosine
Minimum Similarity	0.3	0.3
Use Negative Neighbor	true	false
Neighbor Size	30	150
Target Recent Threshold	max	max
Global Recent Threshold	max	max

6.2. Fantasy Points Predictions

Tests were performed on the games from three seasons, 2016 to 2018. Historical expert fantasy points projections were used as benchmark. These projections were scrapped from the corresponding site. Three sites (CBS, Fantasy Sharks, FF Today) had all three testing seasons available on their website, while NFL only had the latest season (2018) available.

Table 6.2. Comparison of accuracy of predictions from various sources

Source	*RMSE*				*MAE*			
	2016	2017	2018	Total	2016	2017	2018	Total
Item-Based	7.55	7.66	8.51	7.92	6.12	6.08	6.75	6.32
User-Based	7.25	8.00	8.29	7.86	5.80	6.36	6.50	6.22
CBS	**6.90**	**7.61**	8.40	**7.66**	**5.46**	**5.97**	6.63	**6.02**
Fantasy Sharks	7.86	8.36	8.63	8.29	6.34	6.86	6.88	6.69
FF Today	8.73	9.23	9.23	9.07	7.12	7.69	7.50	7.44
NFL			**8.15**	8.15			**6.35**	6.23

Overall, user-based CF produced a better result than item-based CF in predicting fantasy points, scoring lower RMSE and MAE in two out of three seasons and in overall. Both implementations failed to beat projections made by CBS and NFL experts. However, the user-based and item-based implementations produced more accurate predictions than projections from fantasy community sites Fantasy Sharks and FF Today.

Table 6.3. shows the mean and standard deviation of fantasy points predictions by different sites, along with the observed scores, while *Figure 6.1.* shows the box-and-whisker plot of the fantasy points. On average, expert projections tend to overestimate the fantasy points of a player. The machine predictions, while having a lower accuracy than CBS and NFL expert projections, stayed closer to the observed actual points in terms of mean and median.

Table 6.3. Mean, standard deviation, and median of each source's predictions compared to those of the observed values

Source	*Mean*	*Standard Deviation*	*Median*
Observed	15.40	8.10	15.12
Item-Based	15.54	3.56	16.88
User-Based	15.21	4.13	16.00
CBS	16.04	5.56	17.00
Fantasy Sharks	18.10	5.00	18.70
FF Today	20.89	3.86	21.80
*NFL**	16.02	4.40	17.03

The standard deviations and boxplot show how volatile fantasy football predictions can be. Fantasy points predictions tend to have much lower variance than the actual observed fantasy points. Additionally, outliers are not uncommon both in observed and predicted fantasy points.

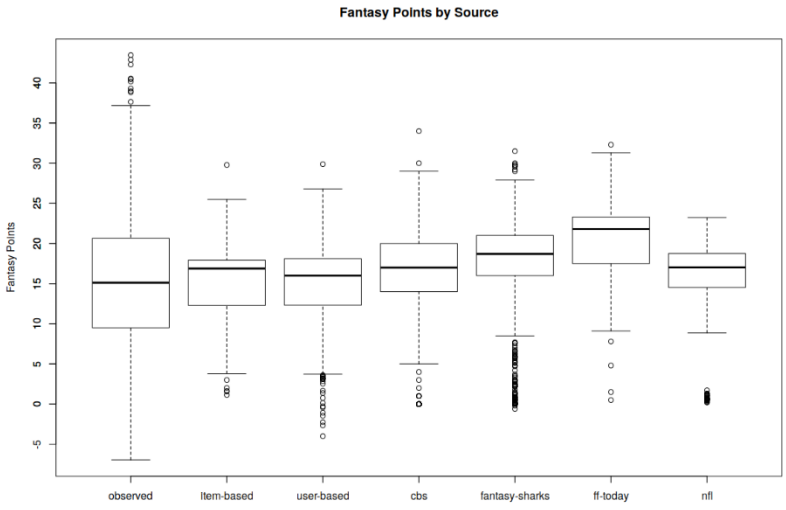

Figure 6.1. Distribution of predicted fantasy points by source compared to observed values

Table 6.4. shows the ratio of predictions for each source that fell within various thresholds. CBS projections lead in all thresholds. In predictions falling within 2.5 points, NFL came in second; while for the rest of the thresholds, user-based CF was able to outperform NFL expert projections. Item-Based CF placed third in the 7.5 points and 10.0 points threshold after user-based CF.

Table 6.4. The ratio of predictions that fall within different thresholds

Source	*Ratio of Predictions Made Within*			
	2.5 points	*5.0 points*	*7.5 points*	*10.0 points*
Item-Based	24.26%	48.09%	65.70%	78.63%
User-Based	25.49%	49.32%	66.63%	79.56%
CBS	28.25%	51.61%	67.53%	80.48%
Fantasy Sharks	23.27%	43.73%	62.57%	77.04%
FF Today	20.00%	38.50%	56.35%	70.55%
*NFL**	26.95%	48.33%	64.87%	78.62%

Some previous machine predictions cited the difficulty of predicting the first few weeks of the season. This part of the season was used as training data. In this study, to predict the first few weeks of the season, data from previous seasons was used.

Figure 6.2. shows the accuracy of different sources of predictions by week of a season. Both CF algorithms did not perform as well on the third and fourth week. The accuracy stabilized as the season progressed, and slightly worsened toward the end of the season.

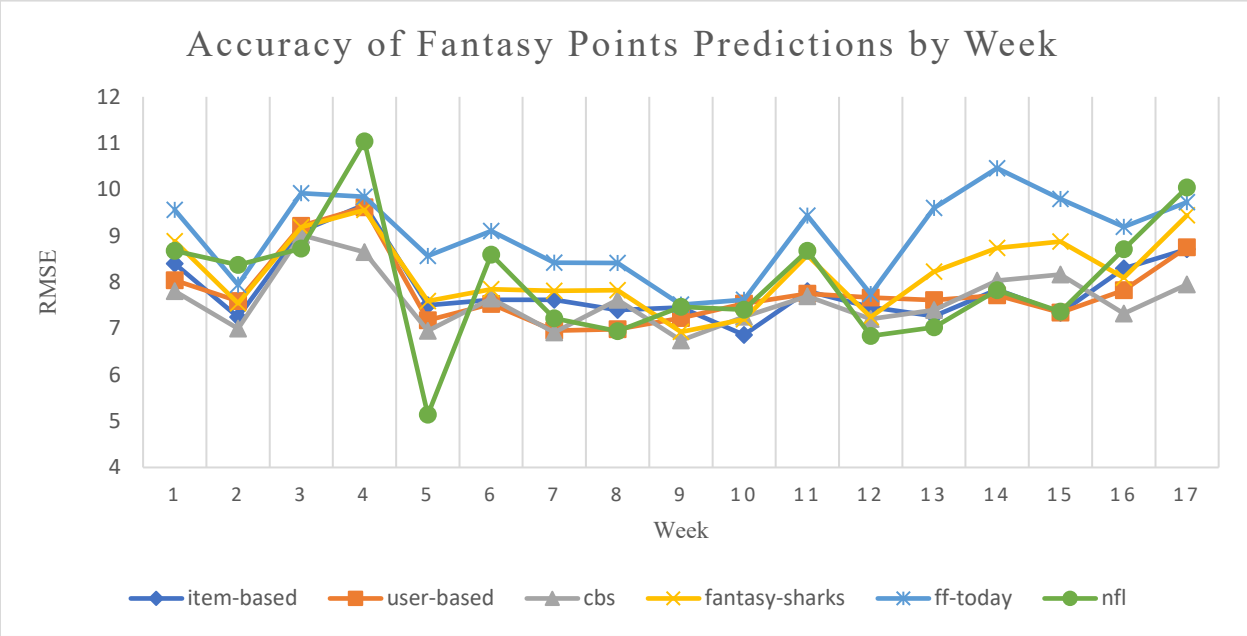

Figure 6.2. Accuracy of predictions by week and source

Additionally, the upward spikes on the figure shows that the accuracy of different sources of predictions generally followed the same trend. In third and fourth weeks of the season, as well as toward the end of the season, the accuracy of predictions dipped This phenomenon may be caused by a set of players who significantly outperform or underperform general expectations during those weeks. Meanwhile, predictions generally stabilized during the middle part of the season.

6.3. Feature Predictions

While other regression methods only produce the total fantasy points predictions, the implementations of collaborative filtering presented on this paper also produced the predictions for individual features/statistics.

The accuracy of these predictions was compared against benchmark expert projections. However, the statistics that the expert projections offered differ depending on the sites.

Table 6.5. Accuracy of individual feature predictions by source. Lighter color indicates lower RMSE score relative to other RMSE score of other sources.

Feature	*Item-Based*	*User-Based*	*CBS*	*Fantasy Sharks*	*FF Today*	*NFL*
Pass attempt	9.553	9.931	9.443	9.875	9.082	
Pass completion	6.583	6.693	6.406	6.628	6.222	
Pass yards	81.069	81.491	78.768	80.272	75.468	81.357
Pass TD	1.115	1.136	1.102	1.135	1.178	1.160
INT	1.004	0.955	0.963	0.936	1.035	0.912
Pass 2-pt attempt	0.443	0.420				
Pass 2-pt made	0.279	0.266				0.296
Rushing attempt	2.588	2.113	2.111	2.193	2.473	
Rushing yards	16.954	14.654	14.632	14.952	14.571	15.858
Rushing TD	0.384	0.356	0.352	0.353	0.386	0.365
Rushing 2-pt attempt	0.123	0.125				
Rushing 2-pt made	0.092	0.096				
Fumbles lost	0.497	0.472	0.460	0.466		0.493
Fumbles total	0.781	0.767				

Table 6.5. shows how the CF implementations performed compared to expert projections on individual features, based on the RMSE score for each individual feature. User-based CF had better accuracy than item-based CF, scoring lower RMSE in 9 out of 14 features. However, it is not conclusive that the CF implementations were able to produce better predictions on individual features than expert projections.

6.4. Support Vector Machine Models

Four support vector machine (SVM) models were trained and tested. The first set of models used 15 features, while the second set of models added the predicted score from user-based CF implementation as another feature. *Table 6.6.* shows the result of the test of the models using games from the 2018 season.

Table 6.6. Comparison of accuracy of predictions by various SVM models and the user-based CF implementation

Model	*RMSE*	*MAE*
svm-linear	8.499	6.747
svm-linear-plus	8.321	6.592
svm-rbf	8.469	6.671
svm-rbf-plus	8.438	6.618
user-based CF	8.294	6.502

The *svm-linear* and *svm-rbf* models represented the SVM models with linear and RBF kernel respectively. The *svm-linear-plus* and *svm-rbf-plus* models added the *predictedScore* as a feature from user-based CF. For both kernels, adding the prediction from user-based CF implementation improved the RMSE and MAE scores of the models. Surprisingly, none of the SVM models produced better result than the user-based CF implementation. This may be due to lack of hyperparameters optimization.

Figure 6.3. shows the weekly RMSE scores of the four SVM models. Except for week 4 of the 2018 season, the models that added predicted score as a feature consistently performed

better. It's worth noting that both CF implementation and expert projections performed poorly during week 4 of the 2018 season.

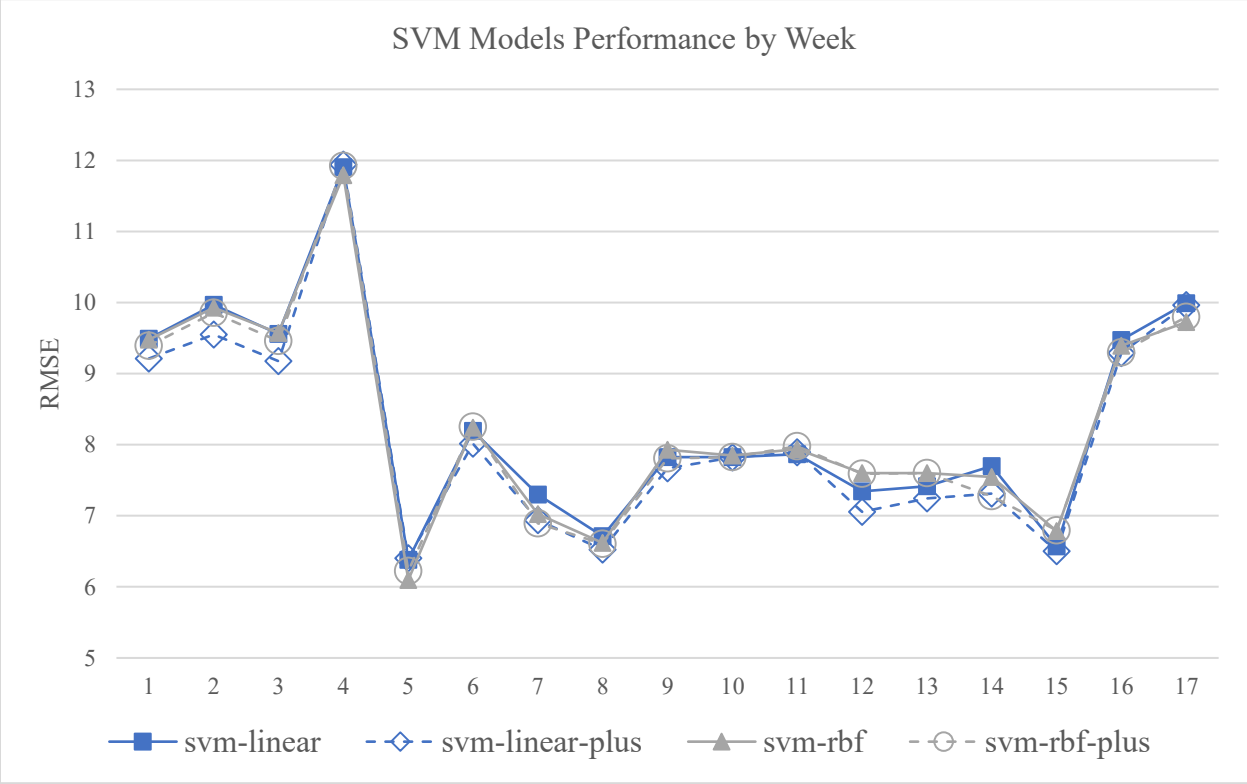

Figure 6.3. Weekly accuracy of SVM models

7. DISCUSSION

7.1. Accuracy of Methods

Expert projections have been used widely as the primary way for fantasy owners in determining the right players for their team. This study attempted to provide an alternative way in creating predictions of fantasy players' performance, as expert projections' accuracy has been shown to be inaccurate [5].

Based on the results, it is not conclusive that the CF implementations were able to generate predictions for quarterback fantasy points that were more accurate than the benchmark expert projections. While their accuracy was better than expert projections in some areas, CBS expert projections was able to outperform the predictions made by the collaborative filtering implementations.

However, the CF methods are not necessarily useless in fantasy football settings. The machine predictions can be used as another data point in fantasy football predictions to consider, as aggregate predictions tend to perform better than predictions by single source [30].

Additionally, both CF methods were able to produce predictions for the games during the early part of the season. Previous studies used the early part of the season as training data [9], [11], [12]. The CF methods presented used data from previous seasons, while accounting for inactivity during offseason, to create predictions for the early part of the season.

Meanwhile, the accuracy of the CF implementations on predicting individual features was not impressive. The CF methods could not consistently outperform the expert projections on predicting individual features.

Overall, based on RMSE and MAE scores of fantasy points predictions, user-based CF is more promising than item-based CF. The user-based CF also produced higher ratio of predictions that fell within various thresholds.

7.2. Limitations

This study has a few limitations. First, the data only contained statistics for players who had seen a sufficient time at quarterback position (pass attempt of at least 5 in a game). Therefore, the scope of this study was limited to quarterback positions while ignoring possible interactions that a quarterback might have with other positions on the team, such as running back and wide receivers. Considering that football is a team game, including the performance of players from other positions could be helpful in future study.

Second, player injury does not factor into predictions. As previously discussed, injury plays an important role in the amount of playing time players have, and in turn the amount of projected points. However, due to limited availability of historical injury report, player injury report was not considered.

Moreover, the predictions also did not factor player's attributes such as weight, height, speed, and other athletic abilities. This data could be gathered from various sites and might be useful in creating a hybrid memory and model-based collaborative filtering model. In an effort to keep simplicity, these attributes were not used in this study.

Lastly, only regular season games were used in the study. Pre-season games were not included, as teams use these games to practice and evaluate their players. Experienced and starting players often also do not see significant amount of playing time during these games, and sometimes skip participation altogether. Meanwhile, post-season games were not included as only few selected teams participate in the post-season.

For the SVM models, no hyperparameters optimizations were performed. The hyperparameters used were either the default values for the kernel or the optimal values found in previous study. Although the user-based CF implementation performed better than the SVM models in this study, it is not conclusive whether the collaborative filtering implementation is a better predictor than the SVM models.

7.3. Future Work

While the result of the presented collaborative filtering implementations did not produce predictions with better accuracy than current expert projections, there are some future work and tweaks that may be useful.

This study focused on only the quarterback position. Future study may include players from other positions. This may be helpful considering that football is a team game. Including positions that are not the target player's position may take the interaction between these positions into account when predicting the player's fantasy points. At the same time, unlike quarterback position, other positions such as RB, WR, and TE have inconsistent on-field playing time that needs to be considered in the model.

The CF methods described in this study only used the statistics that the players had during a game. It did not use other features, such as player's weight, height, and speed. Including these features in the model may be helpful in creating a more accurate model for players, especially for players that do not have previous data available such as first-year players.

Other types or flavors of collaborative filtering may also be explored. The CF implementations on this paper set the players and team-feature into a matrix with 448 items. Dimensionality reduction, through methods like Singular Value Decomposition (SVD), has been

shown to produce better result than high dimensions collaborative filtering [48]. Methods such as SVD may be explored in this domain in the future.

Lastly, the predictions made by the CF methods can be used in other models as another data point or feature for predictions. Previous works had used outside projections to create a model for prediction [11], [13], [30]. As shown on Section 6.4, the predicted score from user-based CF can improve the Support Vector Machine models with linear and RBF kernels. In future study, the predicted score by collaborative filtering may be useful as an added feature in other regression models or neural networks.

8. CONCLUSION

In this paper, user-based and item-based collaborative filtering were implemented to predict NFL quarterbacks' individual statistics and fantasy points. The data from seven seasons were used to get the best parameters for *CF* implementations.

The implementations were tested on three seasons of games and the results were compared to expert projections from four sites. In both predicting weekly fantasy points and individual statistics of quarterbacks, neither implementations were significantly better than the traditional expert projections. The *CF* implementations were able to generate predictions for games during the first few weeks of a season. User-based *CF* tends to produce more accurate results than item-based *CF* in both fantasy points and individual statistics predictions.

Future work may involve including other positions in addition to quarterback in the predictions, as well as using other features beside players' statistics in the dataset. Additionally, the output of the predictions may also be combined into another system to generate better predictions.

REFERENCES

[1] FTSA, “Press Release: Fantasy Sports Now a $7 Billion Industry,” *FTSA*, 20-Jun-2017. .

[2] D. Gouker, “How Much Money Did DraftKings, FanDuel Make In The Past Year?,” *Legal Sports Report*, 19-Oct-2017. .

[3] E. Novy-Williams, “Trump Trash Talk Can’t Touch NFL, Packers Financials Reveal,” 16-Jul-2018.

[4] P. Marinova, “What Legal Sports Betting Means For the Future of FanDuel & DraftKings,” *Fortune*, 15-May-2018. [Online]. Available: http://fortune.com/2018/05/15/sports-betting-fanduel-draftkings/. [Accessed: 28-Feb-2019].

[5] G. Reda and M. Stringer, “Are ESPN’s fantasy football projections accurate?,” *Datascope Analytics*, 09-Dec-2014. [Online]. Available: //datascopeanalytics.com/blog/are-espns-fantasy-football-projections-accurate/. [Accessed: 18-Feb-2019].

[6] J. Hart, “Fantasy Football + Artificial Intelligence Cheat Sheet!,” *Towards Data Science*, 27-Aug-2017. [Online]. Available: https://towardsdatascience.com/fantasy-football-artificial-intelligence-cheat-sheet-ac172a23e2e1. [Accessed: 18-Feb-2019].

[7] Optimized Financial System, “Fantasy Football Prediction Using Vectorial ARMA Models | Optimized Financial Systems,” 19-May-2014. .

[8] J. W. Porter, “Predictive Analytics for Fantasy Football: Predicting Player Performance Across the NFL,” *University of New Hampshite Scholars’ Repository*, p. 28, 2018.

[9] N. (Neena S.) Parikh, “Interactive tools for fantasy football analytics and predictions using machine learning,” Thesis, Massachusetts Institute of Technology, 2015.

[10] R. Lutz, “Fantasy Football Prediction,” *arXiv:1505.06918 [cs]*, May 2015.

[11] N. King and A. LeBoulluec, “Projecting a Quarterback’s Fantasy Football Point Output for Daily Fantasy Sports using Statistical Models,” *International Journal of Computer Applications*, vol. 164, no. 4, pp. 22–27, Apr. 2017.

[12] J. R. Landers and B. Duperrouzel, “Machine Learning Approaches to Competing in Fantasy Leagues for the NFL,” *IEEE Transactions on Games*, pp. 1–1, May 2018.

[13] A. Baughman, “Watson: Behind the code,” *IBM Developer*, 06-Feb-2019. .

[14] X. Su and T. M. Khoshgoftaar, “A Survey of Collaborative Filtering Techniques,” *Adv. in Artif. Intell.*, vol. 2009, pp. 4:2–4:2, Jan. 2009.

[15] K. Bosner, “How American Football Works,” *HowStuffWorks*, 15-Jan-2004. [Online]. Available: https://entertainment.howstuffworks.com/football.htm. [Accessed: 19-Feb-2019].

[16] J. Alder, “How Are the NFL League and Playoffs Organized?,” *ThoughtCo*, 09-Apr-2018. [Online]. Available: https://www.thoughtco.com/how-the-nfl-is-organized-1335412. [Accessed: 19-Feb-2019].

[17] J. Sablich, “A Beginner’s Guide to Playing Fantasy Football,” *The New York Times*, 24-Aug-2017.

[18] “DraftKings NFL Rules & Scoring,” *DraftKings - Daily Fantasy Sports for Cash*. [Online]. Available: https://www.draftkings.com/help/rules/nfl, https://www.draftkings.com/help/rules/nfl. [Accessed: 21-Feb-2019].

[19] Yahoo, “Default league settings, scoring, and stats in Fantasy Football | Yahoo Help - SLN6489.” [Online]. Available: https://help.yahoo.com/kb/SLN6489.html. [Accessed: 21-Feb-2019].

[20] B. Perniciaro, “The Ultimate Fantasy Football Scoring & Points System Guide,” *https://www.cheatsheetwarroom.com/blog*. .
[21] “Season Long vs. Daily Fantasy Sports,” *MyFantasyLeague*. [Online]. Available: http://home.myfantasyleague.com/fantasy-sports-legislation/season-long-vs-daily-fantasy-sports/. [Accessed: 17-Feb-2019].
[22] C. Gaines, “Major League Baseball’s Season Is Not As Long As You May Think,” *Business Insider*, 23-Sep-2014. [Online]. Available: https://www.businessinsider.com/major-league-baseball-season-2014-9. [Accessed: 21-Feb-2019].
[23] “Creating the NFL Schedule | NFL Football Operations.” [Online]. Available: https://operations.nfl.com/the-game/creating-the-nfl-schedule/. [Accessed: 21-Feb-2019].
[24] H. Keyser, “Why Are Baseball Seasons 162 Games Long?,” 11-Sep-2014. [Online]. Available: http://mentalfloss.com/article/58831/why-are-baseball-seasons-162-games-long. [Accessed: 21-Feb-2019].
[25] “NBA Frequently Asked Questions,” *NBA.com*. [Online]. Available: http://www.nba.com/news/faq. [Accessed: 21-Feb-2019].
[26] A. Powell-Morse, “Evolution of the NFL Offense: An Analysis of the Last 80 Years,” *Best Tickets Blog*, 30-Sep-2013. [Online]. Available: http://www.besttickets.com/blog/evolution-of-nfl-offense/. [Accessed: 21-Feb-2019].
[27] S. Wyche, “Passing league: Explaining the NFL’s aerial evolution - NFL.com,” 05-Jul-2012. [Online]. Available: http://www.nfl.com/news/story/09000d5d82a44e69/article/passing-league-explaining-the-nfls-aerial-evolution. [Accessed: 21-Feb-2019].
[28] ProFootballDoc, “How ‘questionable,’ ‘doubtful’ designations differ among NFL teams,” *sandiegouniontribune.com*, 06-Sep-2018. [Online]. Available: https://www.sandiegouniontribune.com/sports/profootballdoc/sd-sp-pfd-questionable-doubtful-nfl-injury-report-0906-htmlstory.html. [Accessed: 21-Feb-2019].
[29] A. Zonfrelli, “Inaccuracies in the Injury Report Across the NFL,” *The Harvard Sports Analysis Collective*, 06-Nov-2013. .
[30] I. Petersen, “Who Has the Best Fantasy Football Projections? 2017 Update,” *Fantasy Football Analytics*, 20-Mar-2017. [Online]. Available: https://fantasyfootballanalytics.net/2017/03/best-fantasy-football-projections-2017.html. [Accessed: 18-Feb-2019].
[31] R. Knoche, “Fantasy Football Predictions — A First Look · Dealing Data,” 21-Aug-2016. [Online]. Available: http://www.dealingdata.net/2016/08/31/Fantasy-Football-Preview/. [Accessed: 18-Feb-2019].
[32] derek-adair, *A maintained fork of burntsushi/nflgame: An API to retrieve and read NFL Game Center JSON data. It can work with real-time data, which can be used for fantasy football. - derek-adair/nflgame*. 2018.
[33] D. Goldberg, D. Nichols, B. M. Oki, and D. Terry, “Using Collaborative Filtering to Weave an Information Tapestry,” *Commun. ACM*, vol. 35, no. 12, pp. 61–70, Dec. 1992.
[34] J. A. Konstan, B. N. Miller, D. Maltz, J. L. Herlocker, L. R. Gordon, and J. Riedl, “GroupLens: applying collaborative filtering to Usenet news,” *Communications of the ACM*, vol. 40, no. 3, pp. 77-, Mar-1997.
[35] P. Resnick, N. Iacovou, M. Suchak, P. Bergstrom, and J. Riedl, “GroupLens: An Open Architecture for Collaborative Filtering of Netnews,” in *Proceedings of the 1994 ACM*

Conference on Computer Supported Cooperative Work, New York, NY, USA, 1994, pp. 175–186.
[36] B. Sarwar, G. Karypis, J. Konstan, and J. Riedl, "Item-based Collaborative Filtering Recommendation Algorithms," in *Proceedings of the 10th International Conference on World Wide Web*, New York, NY, USA, 2001, pp. 285–295.
[37] G. Linden, B. Smith, and J. York, "Amazon.com recommendations: item-to-item collaborative filtering," *IEEE Internet Computing*, vol. 7, no. 1, pp. 76–80, Jan. 2003.
[38] M. Ekstrand, "Similarity Functions for User-User Collaborative Filtering," *GroupLens*, 24-Oct-2013. [Online]. Available: https://grouplens.org/blog/similarity-functions-for-user-user-collaborative-filtering/. [Accessed: 23-Feb-2019].
[39] "Node.js," *Node.js*. [Online]. Available: https://nodejs.org/en/. [Accessed: 23-Feb-2019].
[40] "math.js | an extensive math library for JavaScript and Node.js." [Online]. Available: http://mathjs.org/. [Accessed: 23-Feb-2019].
[41] "RStudio," *RStudio*, 27-Mar-2014. [Online]. Available: https://www.rstudio.com/. [Accessed: 23-Feb-2019].
[42] "Fantasy Football Player Projections | Fantasy – NFL.com." [Online]. Available: https://fantasy.nfl.com/research/projections?offset=0&position=1&sort=projectedPts&statCategory=projectedStats&statSeason=2018&statType=weekProjectedStats&statWeek=1. [Accessed: 26-Feb-2019].
[43] "Projections," *FantasySharks.com*. [Online]. Available: https://www.fantasysharks.com/. [Accessed: 26-Feb-2019].
[44] "Fantasy Football Stats - Points," *CBSSports.com*. [Online]. Available: https://www.cbssports.com/fantasy/football/stats/sortable/points/QB/standard/projections/2016/1. [Accessed: 26-Feb-2019].
[45] "Quarterback Projections: 2016 Week 1 - FF Today." [Online]. Available: http://www.fftoday.com/rankings/playerwkproj.php?Season=2016&GameWeek=1&PosID=10&LeagueID=1. [Accessed: 26-Feb-2019].
[46] mljs, *LIBSVM for the browser and nodejs :fire:. Contribute to mljs/libsvm development by creating an account on GitHub*. ml.js, 2019.
[47] M. Kuhn, "The caret Package," 2019. [Online]. Available: http://topepo.github.io/caret/index.html. [Accessed: 19-Mar-2019].
[48] B. Sarwar, G. Karypis, J. Konstan, and J. Riedl, "Analysis of Recommendation Algorithms for e-Commerce," in *Proceedings of the 2Nd ACM Conference on Electronic Commerce*, New York, NY, USA, 2000, pp. 158–167.

APPENDIX

A.1. Players in the Final Data Set

Table A.1. Number of games by players (min. 5 pass attempts) and season in the final data set

Name	*Number of Games Played*										
	2009	2010	2011	2012	2013	2014	2015	2016	2017	2018	Total
A.J. Feeley			4								4
Aaron Rodgers	16	15	15	16	8	16	16	16	6	16	140
AJ McCarron							4		2		6
Alex Smith	11	11	16	9	15	15	16	15	15	10	133
Alex Tanney							1				1
Andrew Luck				16	16	16	7	15		16	86
Andy Dalton			16	16	16	16	13	16	16	11	120
Austin Davis						9	3				12
Baker Mayfield										14	14
Ben Roethlisberger	15	12	15	13	16	16	12	14	15	16	144
Billy Volek	1										1
Blaine Gabbert			15	9	3	1	8	6	5	5	52
Blake Bortles						14	16	16	16	13	75
Brady Quinn	9			8							17
Brandon Weeden				15	7	2	6				30
Brett Favre	16	12									28
Brett Hundley									10		10
Brian Brohm	1	1									2
Brian Hoyer	2	1		2	2	14	11	6	6		44
Brian St. Pierre		1									1
Brock Osweiler					1	1	8	15	5	6	36
Brodie Croyle	2	1									3
Bruce Gradkowski	6	6	2	1							15
Bryce Petty								4	4		8
Byron Leftwich	3	1		2							6
C.J. Beathard									6	5	11
Caleb Hanie	1		4								5
Cam Newton			16	16	16	14	16	15	16	14	123
Cardale Jones								1			1
Carson Palmer	16	16	10	14	16	6	16	15	7		116
Carson Wentz								16	13	11	40
Case Keenum					8	2	5	10	15	16	56
Chad Henne	14	15	3	9	15	3					59
Chad Pennington	3										3

Table A.1. Number of games by players (min. 5 pass attempts) and season in the final data set (continued)

Name	*Number of Games Played*										
	2009	2010	2011	2012	2013	2014	2015	2016	2017	2018	Total
Charlie Batch		3	1	2							6
Charlie Frye	3										3
Charlie Whitehurst		5	3			7	3	1			19
Chase Daniel					2	1				2	5
Chris Redman	3	1	3								7
Chris Simms	1										1
Christian Ponder			11	16	9	1					37
Cody Kessler								8	1	5	14
Colin Kaepernick				9	16	16	8	11			60
Colt McCoy		8	13	1		5	1			2	30
Connor Cook								1			1
Connor Shaw						1					1
Curtis Painter	2		9		1						12
Dak Prescott								16	16	16	48
Dan Orlovsky			8	1			1				10
Daunte Culpepper	7										7
David Carr	3	1									4
David Fales								1	1		2
David Garrard	16	14									30
Dennis Dixon	1	2									3
Derek Anderson	8	11				5	1	2	1	2	30
Derek Carr						16	16	15	15	16	78
Deshaun Watson									7	16	23
DeShone Kizer									15	2	17
Dominique Davis					1						1
Donovan McNabb	14	13	6								33
Drew Brees	15	16	16	16	16	16	15	16	16	15	157
Drew Stanton	3	4				9	1	2	5		24
EJ Manuel					10	4	2	1	2		19
Eli Manning	16	16	16	16	16	16	16	16	15	16	159
Geno Smith					16	14	1	2	1		34
Greg McElroy				2							2
J.P. Losman			1								1
J.T. O'Sullivan	1										1
Jacoby Brissett								3	15		18
Jake Delhomme	11	5	1								17
Jake Locker			3	10	7	7					27
Jake Rudock									1		1

Table A.1. Number of games by players (min. 5 pass attempts) and season in the final data set (continued)

Name	*Number of Games Played*										
	2009	2010	2011	2012	2013	2014	2015	2016	2017	2018	Total
JaMarcus Russell	12										12
Jameis Winston							16	16	13	11	56
Jared Goff								7	15	16	38
Jason Campbell	16	13	6	3	8	3					49
Jay Cutler	16	15	10	15	11	15	15	5	13		115
Jeff Driskel										6	6
Jeff Tuel					2						2
Jimmy Clausen		12				2	4				18
Jimmy Garoppolo						2		2	5	3	12
Joe Callahan									1		1
Joe Flacco	16	16	16	16	16	16	10	16	16	9	147
Joe Webb		4	3						1		8
John Beck			4								4
John Skelton		5	8	7							20
Johnny Manziel						3	9				12
Jon Kitna		10	1								11
Josh Allen										12	12
Josh Freeman	9	16	15	16	4		1				61
Josh Johnson	5	1	1							4	11
Josh McCown	1		2		7	11	8	5	13	3	50
Josh Rosen										14	14
Joshua Dobbs										1	1
Keith Null	4										4
Kellen Clemens	1		3		9		1		1		15
Kellen Moore							3				3
Kerry Collins	7	10	3								20
Kevin Hogan								1	4		5
Kevin Kolb	3	6	8	6							23
Kirk Cousins				2	5	6	16	16	16	16	77
Kurt Warner	15										15
Kyle Allen										1	1
Kyle Boller	6		2								8
Kyle Lauletta										1	1
Kyle Orton	16	13	8	1	2	12					52
Lamar Jackson										8	8
Landry Jones							3	2	1		6
Logan Thomas						1					1
Luke McCown		1	3				1				5

Table A.1. Number of games by players (min. 5 pass attempts) and season in the final data set (continued)

Name	*Number of Games Played*										
	2009	2010	2011	2012	2013	2014	2015	2016	2017	2018	Total
Marc Bulger	8										8
Marcus Mariota							12	15	15	14	56
Mark Brunell	1	1									2
Mark Sanchez	15	15	16	15		9	3	1		2	76
Matt Barkley					2			7		1	10
Matt Cassel	15	15	9	9	9	3	8	2	2	2	74
Matt Flynn	1	2	1	1	6	2					13
Matt Hasselbeck	14	13	16	7	1	2	8				61
Matt Leinart	4		1	1							6
Matt McGloin					7	1	1	1			10
Matt Moore	6	6	13	1	1			4	4		35
Matt Ryan	13	16	16	16	16	16	16	16	16	16	157
Matt Schaub	16	16	10	16	10	1	2			1	72
Matt Simms					3	1					4
Matthew Stafford	10	3	16	16	16	16	16	16	16	16	141
Max Hall		4									4
Mike Glennon					13	6		1	4	2	26
Mike Kafka			2								2
Mike Vick		12	13	10	6	5	4				50
Mitchell Trubisky									12	14	26
Nate Sudfeld									1		1
Nathan Peterman									4	3	7
Nick Foles				7	11	8	11	2	4	5	48
Nick Mullens										8	8
Patrick Mahomes									1	16	17
Paxton Lynch								3	2		5
Peyton Manning	16	16		16	16	16	10				90
Philip Rivers	16	16	16	16	16	16	16	16	16	16	160
Rex Grossman	1	4	13								18
Richard Bartel		1	2								3
Robert Griffin III				15	13	7		5			40
Russell Wilson				16	16	16	16	16	16	16	112
Rusty Smith		2		1							3
Ryan Fitzpatrick	10	13	16	16	11	12	16	14	5	8	121
Ryan Lindley				6		3	1				10
Ryan Mallett						2	7		2		11
Ryan Nassib						1	1				2
Ryan Tannehill				16	16	16	16	13		11	88

Table A.1. Number of games by players (min. 5 pass attempts) and season in the final data set (continued)

Name	*Number of Games Played*										
	2009	2010	2011	2012	2013	2014	2015	2016	2017	2018	Total
Sam Bradford		16	10	16	7		14	15	2	3	83
Sam Darnold										13	13
Scott Tolzien					3			1	1		5
Sean Mannion							1	1	1		3
Sean Renfree							1				1
Seneca Wallace	3	4	3		2						12
Shaun Hill	6	11		1		8	1	1			28
Stephen McGee		2	1								3
T.J. Yates			5	1	2		3		4		15
Tarvaris Jackson	2	3	15		1						21
Taylor Heinicke										1	1
Teddy Bridgewater						13	16			1	30
Terrelle Pryor				1	10			1			12
Thad Lewis				1	5						6
Tim Tebow		3	12								15
Todd Bouman		1									1
Todd Collins	2	2									4
Tom Brady	16	16	16	16	16	16	16	12	16	16	156
Tom Savage						1		3	7		11
Tony Pike		1									1
Tony Romo	16	6	15	16	15	15	4				87
Trent Edwards	7	4									11
Trevone Boykin								2			2
Trevor Siemian								14	11		25
Troy Smith		6									6
Tyler Palko			5								5
Tyler Thigpen	1	4	1	1							7
Tyrod Taylor				1			14	15	15	3	48
Vince Young	10	9	3								22
Zach Mettenberger						7	6				13

A.2. Parameter Optimization Results for Item-Based CF

Table A.2. Accuracy of item-based CF implementations by similarity functions

similarityFunction	*RMSE*	*MAE*
cosine	8.42	6.64
cosineMeanCentered	8.66	6.83
pearson	8.79	6.94

Table A.3. Accuracy of item-based CF implementations by minimum similarity thresholds

minSimilarity	*RMSE*	*MAE*
0.3	8.47	6.68
0.5	8.76	6.91

Table A.4. Accuracy of item-based CF implementations by use of negative neighbors

useNegativeNeighbor	*RMSE*	*MAE*
false	8.61	6.79
True	8.62	6.79

Table A.5. Accuracy of item-based CF implementations by neighbor sizes

neighborSize	*RMSE*	*MAE*
10	8.78	6.93
20	8.66	6.82
30	8.62	6.79
40	8.57	6.76
50	8.58	6.77
60	8.57	6.75
75	8.54	6.75
100	8.49	6.70
150	8.46	6.69
200	8.47	6.70
250	8.51	6.75
max	8.63	6.80

Table A.6. Accuracy of item-based CF implementations by the number of recent weeks accounted for the target player

targetRecentThreshold	*RMSE*	*MAE*
5	8.73	6.77
10	8.76	6.89
17	8.66	6.83
25	8.50	6.69
50	8.48	6.69
max	8.61	6.80

Table A.7. Accuracy of item-based CF implementations by the number of recent weeks accounted for all players

globalRecentThreshold	*RMSE*	*MAE*
17	9.00	7.07
25	8.83	6.92
50	8.56	6.77
75	8.55	6.77
max	8.41	6.64

Table A.8. Accuracy of item-based CF implementations with the top 10 combinations of parameters

similarityFunction	*minSimilarity*	*useNegativeNeighbor*	*neighborSize*	*targetRecentThreshold*	*globalRecentThreshold*	*count*	*coverage*	*RMSE*	*MAE*
cosinemeancentered	0.3	FALSE	60	5	25	2	0.07%	2.476	1.79
cosinemeancentered	0.3	TRUE	60	5	25	2	0.07%	2.476	1.79
54 rows omitted…									
cosine	0.3	FALSE	150	max	max	1944	71.05%	7.877	6.227
cosine	0.3	TRUE	150	max	max	1944	71.05%	7.877	6.227
cosine	0.3	FALSE	100	max	max	2203	80.52%	7.885	6.230
cosine	0.3	TRUE	100	max	max	2203	80.52%	7.885	6.230
cosine	0.3	FALSE	200	50	max	1075	39.29%	7.894	6.253
cosine	0.3	TRUE	200	50	max	1075	39.29%	7.894	6.253
cosine	0.3	FALSE	200	max	max	1686	61.62%	7.894	6.243
cosine	0.3	TRUE	200	max	max	1686	61.62%	7.894	6.243

A.3. Parameter Optimization Results for User-Based CF

Table A.9. Accuracy of user-based CF implementations by similarity functions

similarityFunction	*RMSE*	*MAE*
cosine	8.68	6.90
cosinemeancentered	8.72	6.93
pearson	9.38	7.48

Table A.10. Accuracy of user-based CF implementations by minimum similarity thresholds

minSimilarity	*RMSE*	*MAE*
0.3	8.76	6.96
0.5	9.13	7.26

Table A.11. Accuracy of user-based CF implementations by use of negative neighbors

useNegativeNeighbor	*RMSE*	*MAE*
false	8.95	7.11
True	8.86	7.04

Table A.12. Accuracy of user-based CF implementations by neighbor sizes

neighborSize	*RMSE*	*MAE*
3	9.150	7.282
5	8.970	7.141
10	8.872	7.055
15	8.850	7.036
20	8.841	7.030
25	8.836	7.025
30	8.835	7.023
max	8.836	7.024

Table A.13. Accuracy of user-based CF implementations by the number of recent weeks accounted for the target player

targetRecentThreshold	*RMSE*	*MAE*
5	9.42	7.47
10	9.20	7.33
17	8.95	7.11
25	8.69	6.92
50	8.54	6.92
max	8.51	6.79

Table A.14. Accuracy of user-based CF implementations by the number of recent weeks accounted for all players

globalRecentThreshold	*RMSE*	*MAE*
17	9.28	7.37
25	9.15	7.28
50	8.83	7.02
75	8.81	7.00
max	8.64	6.88

Table A.15. Accuracy of user-based CF implementations with the top 10 combinations of parameters

similarityFunction	*minSimilarity*	*useNegativeNeighbor*	*neighborSize*	*targetRecentThreshold*	*globalRecentThreshold*	*count*	*coverage*	*RMSE*	*MAE*
cosine	max	max	30	FALSE	0.3	2672	97.66%	6.211	7.755
cosine	max	max	30	TRUE	0.3	2672	97.66%	6.211	7.755
cosinemeancentered	max	max	30	FALSE	0.3	2671	97.62%	6.214	7.760
cosinemeancentered	max	max	30	TRUE	0.3	2671	97.62%	6.214	7.760
cosine	max	max	max	FALSE	0.3	2672	97.66%	6.226	7.772
cosine	max	max	max	TRUE	0.3	2672	97.66%	6.226	7.772
cosine	max	max	25	FALSE	0.3	2672	97.66%	6.229	7.775
cosine	max	max	25	TRUE	0.3	2672	97.66%	6.229	7.775
cosinemeancentered	max	max	25	FALSE	0.3	2671	97.62%	6.234	7.784
cosinemeancentered	max	max	25	TRUE	0.3	2671	97.62%	6.234	7.784

A.4. Features for Support Vector Machine Models

Table A.16. List of all features considered for the SVM models

Feature		
week_of_season	player_10_rushing_yds	team_10_offense_point
player_1_passing_att	player_10_rushing_tds	opp_1_defense_firstDown
player_1_passing_cmp	player_10_fumbles_lost	opp_1_defense_yard
player_1_passing_yds	player_10_score	opp_1_defense_passYard
player_1_passing_tds	team_1_offense_firstDown	opp_1_defense_rushYard
player_1_passing_ints	team_1_offense_yard	opp_1_defense_penalty
player_1_passing_twoptm	team_1_offense_passYard	opp_1_defense_penaltyYard
player_1_rushing_att	team_1_offense_rushYard	opp_1_defense_turnover
player_1_rushing_yds	team_1_offense_penalty	opp_1_defense_timeOfPossession
player_1_rushing_tds	team_1_offense_penaltyYard	opp_1_defense_point
player_1_fumbles_lost	team_1_offense_turnover	opp_5_defense_firstDown
player_1_score	team_1_offense_timeOfPossession	opp_5_defense_yard
player_5_passing_att	team_1_offense_point	opp_5_defense_passYard
player_5_passing_cmp	team_5_offense_firstDown	opp_5_defense_rushYard
player_5_passing_yds	team_5_offense_yard	opp_5_defense_penalty
player_5_passing_tds	team_5_offense_passYard	opp_5_defense_penaltyYard
player_5_passing_ints	team_5_offense_rushYard	opp_5_defense_turnover
player_5_passing_twoptm	team_5_offense_penalty	opp_5_defense_timeOfPossession
player_5_rushing_att	team_5_offense_penaltyYard	opp_5_defense_point
player_5_rushing_yds	team_5_offense_turnover	opp_10_defense_firstDown
player_5_rushing_tds	team_5_offense_timeOfPossession	opp_10_defense_yard
player_5_fumbles_lost	team_5_offense_point	opp_10_defense_passYard
player_5_score	team_10_offense_firstDown	opp_10_defense_rushYard
player_10_passing_att	team_10_offense_yard	opp_10_defense_penalty
player_10_passing_cmp	team_10_offense_passYard	opp_10_defense_penaltyYard
player_10_passing_yds	team_10_offense_rushYard	opp_10_defense_turnover
player_10_passing_tds	team_10_offense_penalty	opp_10_defense_timeOfPossession
player_10_passing_ints	team_10_offense_penaltyYard	opp_10_defense_point
player_10_passing_twoptm	team_10_offense_turnover	
player_10_rushing_att	team_10_offense_timeOfPossession	